How software developers experience the team performance in Scaled Agile environment? – A Case Study

Saija Maaranniitty

MSc Thesis
Helsinki 23.11.2020

UNIVERSITY OF HELSINKI
Department of Computer Science

HELSINGIN YLIOPISTO – HELSINGFORS UNIVERSITET – UNIVERSITY OF HELSINKI

Tiedekunta/Osasto – Fakultet/Sektion –Faculty/Section	Koulutusohjelma □ Studieprogram □ Study Programme
Faculty of science	Master's Programme in Computer Science

Tekijä – Författare – Author

Saija Maaranniitty

Työn nimi – Arbetets titel – Title

How software developers experience the team performance in Scaled Agile Environment? – A Case Study

Ohjaajat – Handledare - Supervisors

Tommi Mikkonen

Työn laji – Arbetets art – Level	Aika – Datum – Month and year	Sivumäärä – Sidoantal – Number of pages
MSc Thesis	23.11.2020	40 pages + 1 attachment

Tiivistelmä – Referat – Abstract

Agile software development promotes self-organizing teams, close collaboration with client and team members, short iterative development cycles, response to changing requirements and continuous feedback. Originally agile methods were developed for small teams. As large organisations had a need for agile software development methods supporting several teams and collaboration between them, scaled agile development methods were developed. They promise increased productivity and quality, higher customer engagement and alignment between all levels of the enterprise.

This thesis focuses on the software developers perceptions on team performance in Scaled Agile environment. More specifically this thesis studies how team performance is perceived to be impacted in organization that has adopted Scaled Agile Framework (SAFE). The study was designed to acquire descriptive knowledge through empirical studies. This thesis is based on an investigation of a single case company based in Finland. The data for analysis was collected from semi-structured interviews with 5 experienced developers.

The findings show that several factors impact software developers team performance experience. By investigating the point of view of practitioners, we observed that team performance is a concept that evolves over time and is prone to be influenced by surrounding environment. This understanding can be used to adjust software development method related practices and organizational factors. This study provides pieces of advice on what kind of practices support or hamper team performance and how to improve them.

ACM Computing Classification System (CCS):
Software and its engineering → Software creation and management → Software development process management → Software development methods → Agile software development
Software and its engineering → Software creation and management → Collaboration in software development → Programming teams
Social and professional topics → Professional topics → Management of computing and information systems → Project and people management → Project management techniques

Avainsanat – Nyckelord – Keywords

agile software development, scaled agile, SAFe, team performance

Säilytyspaikka – Förvaringställe – Where deposited

Muita tietoja – Övriga uppgifter – Additional information

Software systems

Contents

1 Introduction

Scaled agile development methods, such as Scaled Agile Framework (SAFe), Discipline Agile Delivery (DAD) and Large Scale Scrum (LeSS), are increasingly applied in organisations [OnV19]. They originate from Agile development methods introduced in early 2000s but they aim at scaling and collaboration in larger organisations [BT05]. At the moment SAFe seems to be the most widespread scaled agile method [OnV19].

SAFe promotes to increase productivity, improve quality, enable faster time-to-market and increased employee engagement and job satisfaction. It is separated into three levels: Team, Program and Portfolio. Team is one of the core elements for organizations to achieve the broader business benefits. It is defined as cross-functional group of 5-11 team members who collaborate with other teams and create and deliver value. SAFe declares that high performing teams are created by relaying to Agile Manifesto's values and principles and by giving teams the authority and accountability to manage their own work [LeF11]. Even though SAFe defines only high level/minimal set of practices for teams, organizations may tailor it to make it fit to their circumstances. In addition SAFe gives freedom for every team to have its own framework of implanting agile methodology. Teams can select to follow scrum, Kanban or scrum-ban [LeF11].

Team performance is a miscellaneous concept in which many factors may influence. Human factors such as motivations, skills, satisfaction and personality are obviously associated with the concept [MMR08]. Furthermore teams are almost always embedded in the organizational systems that define context in which team performance occurs [MP17]. Team performance effectiveness and factors related are tied to the nature and effectiveness of the entire organization [GD96].

Despite the prevalence of scaled agile development methods in large product development groups and organizations, it has not been subjected to analysis of impact to team performance. Even though there has been research in the area of scaling agile, SAFe impact on team performance has not been researched. SAFe's influence to more than understanding the pattern of SAFe framework. The added complexity of applying SAFe comes from the fact that it involves dimensions such as business organizations and

management, a context where iterative development history does not exist and a tendency to follow plan-driven approach is prevalent [DP16].

Given the complexity and multilevel nature of input factors that affect team performance, this thesis aims to shed light on the factors that influence team performance at the individual, team and organizational level through the lens of software developer. This study will also discuss obstacles to a successful implementations of high performing teams in organisation following SAFe. A Venn diagram in Figure 1 portrays the scope of this thesis. Area *SAFe* represents the SAFe framework in which software developers do not have thorough visibility. Developers mostly operate on a team and train level and for example SAFe portfolio level is not in the center of attention. Team performance is also a persistent concept which includes several factors.

This thesis aims to understand how developers experience team performance in SAFe environment. First, factors impacting team performance are considered within software development perspective. Next, the SAFe framework is described and the framework is explained and illustrated. The study ascertains if any of applied SAFe practices is perceived to have an influence on team performance. These goals will be examined through a single-case study in a large financial company which has adopted the Scaled Agile Framework. The understanding acquired in this study could be utilized by the case company in order to reinforce practises that support team performance and to revise activities that restrain it.

The objectives of this thesis are the following: 1) to define understanding on team performance 2) to describe SAFe in general and case company setup 3) to carry out analysis of the interview data in the real life setting. The main research questions are:

RQ1: How do software developers experience team performance in SAFe environment?

RQ2: How applied practices support team performance?

RQ3: How does team performance correlate to the applied practices?

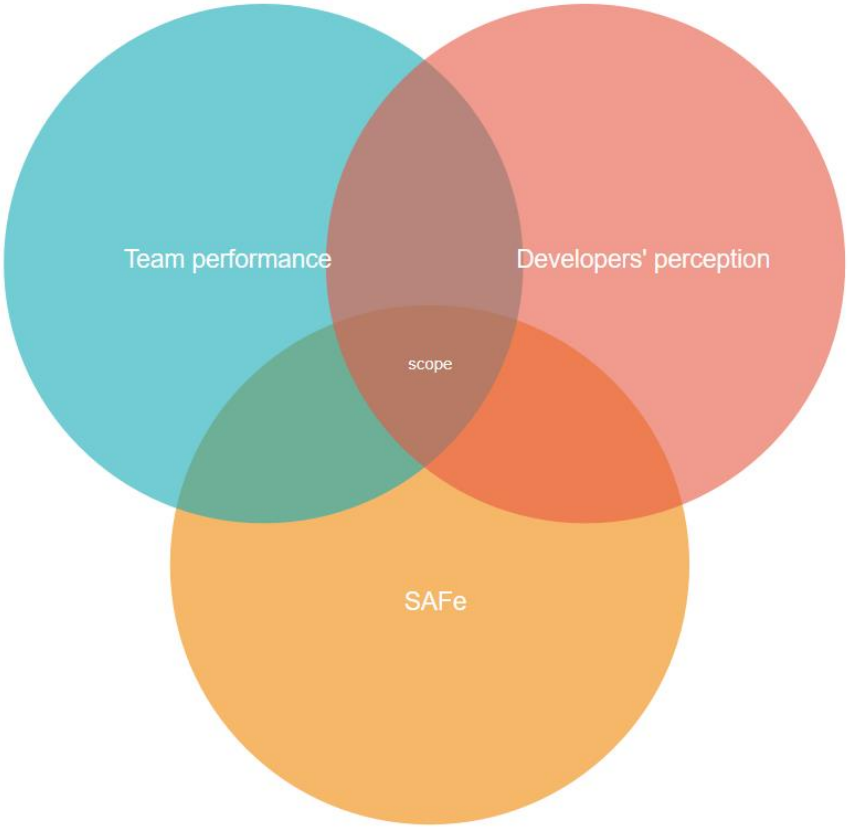

Figure 1: The scope of this study covers the areas depicted as scope. In particular, the focus of this thesis is on how developers perceive SAFe impacts team performance.

This thesis is structured as follows. Chapter 2 describes the background of the research context. It explains the main concept of team performance. Agile and Lean Software development methods are described. Chapter 3 provides background information about the Scaled Agile Framework (SAFe). Chapter 4 presents the overall design of the case study and embodied methodology. Interviews and data collection are also described. Chapter 5 presents the results investigating relationships in the data to understand the potential reasons for team performance experience. Chapter 6 focuses on the discussions. Chapter 7 concludes the work.

2 Agile and Lean Software Development

This chapter contains the required understanding of the background of this thesis. It describes the agile and lean software development framework. Background information about the different principles and aspects upon which SAFe is constructed is provided. This chapter also presents the concept of the team performance in Agile and Lean development.

2.1 Agile software development

Agile software development methodologies emerged 1990's as a response many problems derived from the plan-driven software development method. The Agile approach aimed to respond to evolving and changing requirements with lighter and human oriented software development techniques [LB03]. As traditional software development characteristics include extensive planning, freezing requirements very early in the development life-cycle and phase-by-phase proceeding development cycle (Figure 2), Agile development provides more flexible and light weighted approach [DT08]. In plan-driven software development method iteration length could be three or six months long as in Agile methods iteration length varies between one and four weeks as Figure 2 illustrates [Cas17]. There are several Agile methodologies such as Scrum [SS14], eXtreme Programming (XP) [Bec00], Kanban [Hir08] and Dynamic Systems Development method (DSDM) [Sta97, AWS03]. They all vary in practices, but share common characteristics like iterative development and emphasis on interaction and communication [Cas17].

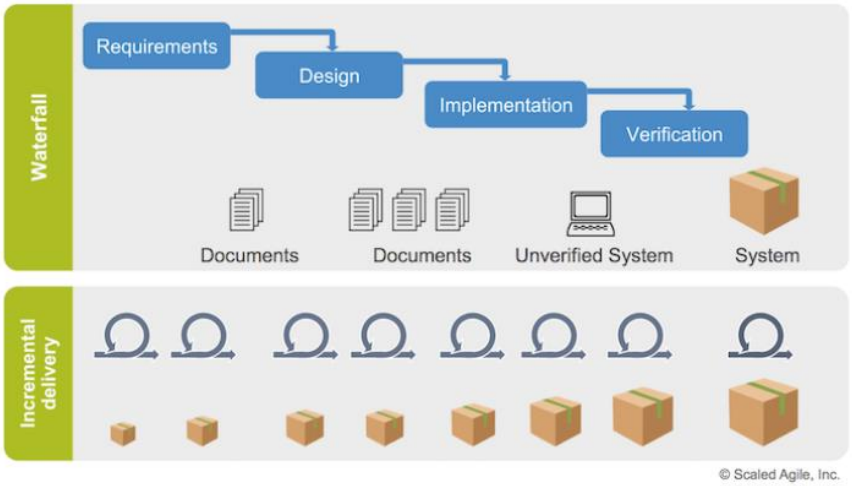

Figure 2: Plan-driven development versus iterative Agile development. [Lef11]

The Agile approach consists of the four values as outlined in the Agile Manifesto [Agile01]:

1) Individuals and interactions over processes and tools
2) Working software over comprehensive documentation
3) Customer collaboration over contract negation
4) Responding to change over following the plan.

Agile values build foundation for the 12 principles that Agile Manifesto defines. These principles are as follows [Agile01].

i. Satisfy the customer through early and continuous delivery of valuable software
ii. Welcome changing requirements
iii. Deliver working software frequently
iv. Business people and developers must work together daily throughout the project

v. Build projects around motivated individuals

vi. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation

vii. Working software is the primary measure of progress

viii. Agile processes promote sustainable development

ix. Continuous attention to technical excellence and good design enhances agility

x. Simplicity – the art of maximizing the amount of work not done – is essential

xi. The best architectures, requirements, and designs emerge from self-organizing teams

xii. At the regular intervals, the team reflects on how to come more effective and then tunes and adjust its behaviour accordingly.

Main objective for agile software development methodologies is working software responding to customer needs. Software development is done in short iterations and outcome of the iterations is presented to customer at the end of each iteration. Short iterations and continuous feedback enables fast reaction to changing requirements. Agile methods address the challenge of unpredictable world, emphasizing close collaboration with the customer and cross-functional team to build working software. Documentation supports software development but is not the goal itself. In addition to communication between individuals, transparency and collaboration are fundamental aspects to obtain success in development process. [TZF11]

2.2 Challenges of Agile Software Development

Despite Agile software development methods are widespread and praised in the software industry, there are some challenges and problems that agile methodology does not address. One of the reported problems with agile software development is a lack of focus on architectural design [PD12]. Architectural design plays important role in the quality of system. It also impacts the system maintenance and lifecycle. Salient part of architectural design's benefits is reuse of architectural knowledge and cross-project re-use of other

software artefacts [PD12]. Falessi et al. argue that support for integrating architectural practices such as software architecture analysis, design and review is regarded as insufficient in agile approach [FCS10]. They claim that the key challenge to associate agile and architectural-centric methods dwell in practical matters of adoption [FCS10].

Erickson et al. reported inconsistency in matching agile software development methods with software standards such as ISO [ELS05]. A documented system is often fundamental part of requirements, in contrast to agile characteristics such as lack of extensive documentation. Furthermore Kajko-Mattsson points out various long-term impacts of reducing documentation [Kaj08]. Undocumented decisions and process activities get easily misunderstood and there emerges difficulties to find a person knowing the answer. Without proper documentation engineers get distracted more often when having to explain the system to stakeholders. Lacking documentation could also lead to a system misunderstanding which may cause defects or unconscious faults [Kaj08]. In larger organizations dependences between teams and systems increases which manifests the need for formal documentation and reduces agility [LMD04]. This brings us to the broader discussion on applying agile practices in larger organizations. Lindvall et al. noted several problems that concern especially large organizations [LMD04]. One of the observations was that large organizations have heavy processes and their supporting systems are too generic and complex to provide needed support. This creates constraints on the development team by limiting practices they can and must use, which then affects the speed and efficiency of software development. Large organizations also need to integrate new technologies and processes with existing ones which creates challenges to efficiently integrate the agile project into its environment [LMD04]. Another constraint is related to several teams that are globally distributed and need to collaborate and coordinate. This arises problems related to communication and coordination between teams [LMD04]. Different time zones affect team availability and might limit cooperative office hours. Another consequence relates to cultural differences between teams [LMD04]. Some cultures differ in their approach to change [AVH17]. On the other hand some cultures tend to be avoid uncertainty, while others cope well with unforeseen events. Furthermore some cultures tend to conceal problems such as technical difficulties [AVH17].

In addition, Boehm and Turner remark that traditionally business and customer relationship is founded on long term road mapping [BT05]. This may bring issues in the form of

contract practises as agile methods emphasize short term planning [BT05]. Furthermore management and stakeholders must move away from life-cycle models and towards iterative and feature centric development to enable shorter term planning [BT05]. In the light of above discussion, it seems there is a need to incorporate methods for large organizations to support agile development.

2.3 Lean software development

Lean development originates from the success of Toyota [PoP12]. It was originally developed for the manufacturing, logistics and product development [PoP07]. The success from Toyota Product Development System gave motivation for applying the practices and principles to software development [PoP12]. Tom and Mary Poppendieck defined Lean Software Development model which builds on same principles as agile software development. Lean Software development model has seven principles that aim to support faster, better quality and smaller budget software delivery [PoP07].

Lean Principles applied to software development are as follows [Pop07]:

1. Eliminate waste
2. Build Quality In
3. Create Knowledge
4. Defer Commitment
5. Deliver Fast
6. Respect People
7. Optimize the Whole

The lean software development attempts to preserve value while eliminating waste and maximizing flexibility in the process [RaS98]. Value is a fundamental element in lean development. Poppendieck has emphasized the value of software can be hard to discover as software rarely has value in and of itself [Pop07]. She points out, the value of software

is not derived solely from the development phase but design and deployment are fundamental as well. She also noted value is either a single time-bound effort; the capability to modify a code base over time tends to be dominant factor in its overall value [PoP12].

Lean development strives for minimum waste to deliver high-quality products meeting customer demand. Waste elimination is seen as the most important principle of Lean thinking [PoP03]. Flow is fundamental part of Lean thinking and closely relates to the principle of optimizing the whole. By optimizing flow, waste can be eliminated. According to Oppenheim work should be organized as a continuous flow that proceeds steadily through all processes without interruptions, rework or backflow [OpB04].

2.4 Team performance in Agile and Lean Development

Performance is an ambiguous concept. It is used in relation to projects, processes, organisations, teams and individuals. This thesis concentrates on the team performance from individual software developers perspective.

The principles behind the Agile Manifesto state that agile projects should be built around self-organizing teams [Agile01]. Dingsøyr et al. define a team as a small group of people with commitment to a common purpose. Team members interact socially and hold themselves mutually accountable. The team has common tasks and team members interact through interdependent tasks [DF16]. Furthermore Guzzo and Dickson address autonomy and authority in their definition of an self-organizing team as follows:

> "Autonomous work groups are teams of employees who typically perform highly related or interdependent jobs, who are identified and identifiable as social unit in an organization, and who are given significant authority and responsibility for many aspects of their work, such as planning, scheduling, assigning tasks to members, and making decisions with economic consequences (usually up to a specific limited value)." [GD96]

Group autonomy is a salient part of self-organized teams. Team members have collective responsibility for the project outcome. Instead of having one person deciding, members of self-managing teams need to agree on a decision [MDD09].

Team performance is a complex concept composed of different elements. Literature on team performance raises several aspects that are important to understand the concept.

Mathieu et al. refer team performance as the process of conducting teamwork [MMR08]. In addition to achieving team objectives, also planning, problem solving, support and development are considered as a part of team performance concept [MMR08]. Similarly, quality, efficiency, knowledge and interpersonal skills are included in performance measure [MMR08]. On the other hand, Salas et al. explain team performance as outcomes of the teams actions irrespective of how the team accomplished the tasks [SSB16]. As many factors external to the team may impact to the success, Salas et al. emphasize differentiation between team performance and team effectiveness [SSB16]. Team effectiveness concept complements team performance perception by taking more holistic perspective as it considers also how team interactions impact the outcome [SSB16].

Team performance can be influenced by various factors. Dingsøyr et al. have analysed the team performance in Software Development in their literature review [DF16]. They report five factors that strongly affect performance, including team coordination, goal orientation, shared mental models, team cohesion and team learning [DF16].

Team coordination includes managing dependencies like shared resources, task assignments, and task relationships [DF16]. Feedback is important for dependency management as dependency detection is not always possible prior the task beginning [DF16]. The ambiguous nature of software development requires teams' to be able to efficiently adapt to changes, as extensive up-front planning is not possible. Frequent interaction and common understanding among members regarding dependencies improves team performance.

In addition task distribution, frequent feedback on work products, coding standards, milestones and critical-path analysis are found to impact team performance [DF16].

Dingsøyr et al. also report that ability to define clear goals and milestones improves team performance [DF16]. Goal oriented leadership has positive impact to team performance and goal oriented leader contributes substantially to individual and team performance [DF16].

Shared mental models refers to common knowledge held by team members and has been found to improve development team's performance [DF16]. For example, understanding of agile software development process such as Scrum could be shared mental model. Shared mental models can help team members to adjust work strategies in accordance with team or task changes [DF16].

Team cohesion is an important factor linking teamwork quality to performance [DF16]. It means that team members feel connected and are driven to achieve a common goal. Characteristics for a cohesive team is interpersonal attraction of team members, group pride and commitment to team tasks. One example of team cohesion impact to teamwork quality and performance, is collective code ownership that has been reported to lead to fewer bugs and thus impact product quality [DF16].

Team learning is a continuous process that enables the team to become responsive and to adjust and adapt its objectives, strategies and processes to circumstances. It increases skills and expertise and improves performance, effectiveness and job satisfaction [DF16].

Salas et al. list five components that endorse team performance and effectiveness: team leadership, mutual performance modelling, backup behaviour, adaptability and team orientation [SSB16]. Table 1 explains each component and related behavioural makers. In addition, three coordinating mechanisms are introduced; shared mental models, mutual trust and closed-loop communication [SSB16]. Coordination mechanisms ensure that team performance core components are consistently updated and relevant information is shared to entire team [SSB16].

Even though the software development work is carried out mostly by teams, variables at the individual and organizational level affect team performance as well [MMR08]. Team effectiveness framework, developed by Mathieu et al., categorizes team input factors into three levels that are transformed into team outcomes; organizational, team and individual level factors. Figure 3. illustrates the team effectiveness framework [MMR08].

A self-managing team's performance depends on not only the team level factors but also the organizational context surrounding the team [MMR08]. The team effectiveness framework identifies organizational level factors associated with team performance such as culture, policies, national culture, organizational goals, organizational structure, training, resources and rewards [MMR08]. On individual level for example individual autonomy, individual roles, skills, resistance to change and work experience have been recognized as variables that impact on team performance in software development [MMR08].

Teamwork	Definition	Behavioral Makers
Team leadership	Ability to direct and coordinate the activities of other team members, assess team performance, assign tasks, develop team knowledge, skills, and abilities, motivate team members, plan and organize, and establish a positive atmosphere.	Facilitate team problem solving. Provide performance expectations and acceptable interaction patterns. Synchronize and combine individual team member contributions. Seek and evaluate information that affects team functioning. Clarify team member roles. Engage in preparatory meetings and feedback sessions with the team.
Mutual performance monitoring	The ability to develop common understandings of the team environment and apply appropriate task strategies to accurately monitor teammate performance.	Identifying mistakes and lapses in other team members' actions. Providing feedback regarding team member actions to facilitate self-correction.
Backup behavior	Ability to anticipate other team members' needs through accurate knowledge about their responsibilities. This includes the ability to shift workload among members to achieve balance during high periods of workload or pressure.	Recognition by potential backup providers that there is a workload distribution problem in their team. Shifting of work responsibilities to underutilized team members. Completion of the whole task or parts of tasks by other team members.
Adaptability	Ability to adjust strategies based on information gathered from the environment through the use of backup behavior and reallocation of intrateam resources. Altering a course of action or team repertoire in response to changing conditions (internal or external).	Identify cues that a change has occurred, assign meaning to that change, and develop a new plan to deal with the changes. Identify opportunities for improvement and innovation for habitual or routine practices. Remain vigilant to changes in the internal and external environment of the team.
Team orientation	Propensity to take other's behaviour into account during group interaction and the belief in the importance of team goal's over individual members' goals.	Taking into account alternative solutions provided by teammates and appraising that input to determine what is most correct. Increased task involvement, information sharing, strategizing, and participatory goal setting.

Table 1: Salas et al. identified five core components that affect most heavily team performance [SSB16]

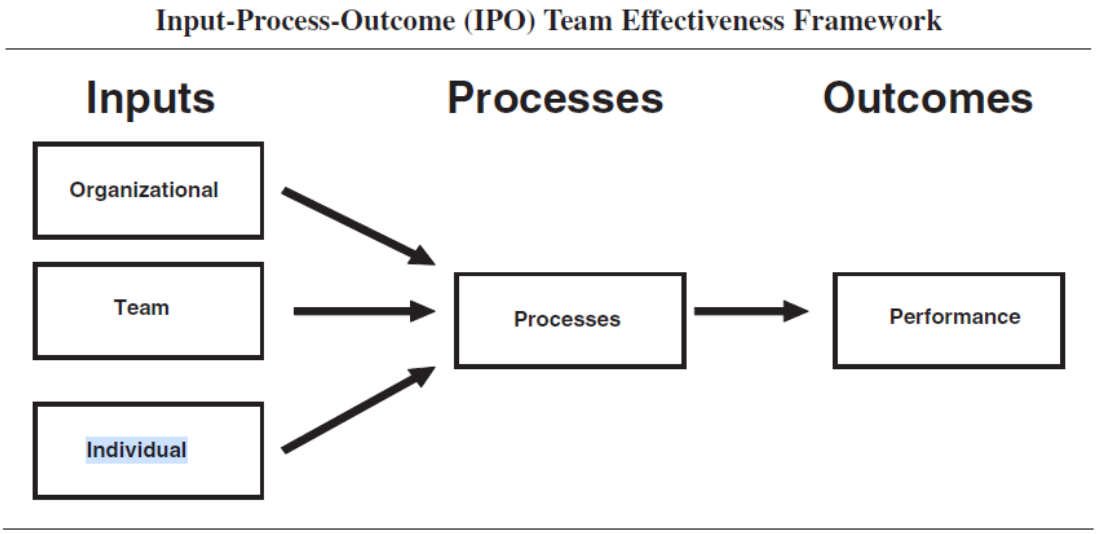

Figure3: Team effectiveness framework [MMR08]

3 SAFe framework

Agile methods were originally developed for small, collocated teams and organizations [BT05]. The scaling agile frameworks are targeted for large organizations to support organizing and managing agile practices in large enterprises [DP16]. Currently there exists several scaling agile frameworks including Disciplined Agile Delivery (DAD), Large-Scale Scrum (LeSS) and SAFe [AR16]. This thesis focuses to SAFe as it is applied in the case organization. This chapter outlines the background of the SAFe. It describes different levels of SAFe as well as related practices in each level. Furthermore, it describes how underlying values and principles are reflect in the framework.

3.1 SAFe background

SAFe is a software development framework that is aiming to scale whole enterprise agile. SAFe was originally developed by Dean Deffingwell and first version was released in the year 2011 [Lef11]. The latest version of SAFe, version 5.0, was released in the beginning of 2020. This thesis refers to SAFe 4.6, as it is the version that is applied in the case company by the time research was done.

SAFe strives to provide large organizations possibility to scale agile the whole organization. It aims to address problems that traditional agile methods do not cover. Big projects executed by large and distributed development organization requires coordination between several teams. Portfolio level management necessitates techniques to organize work together with agile teams [Lef11w]. Simultaneously corporates aspire faster time-to-market, increased productivity and flexibility. SAFe aims to respond the mentioned needs by providing guidance, principles, practices and competencies utilizing Agile and Lean methods [Lef11].

SAFe has four core values that aim to boost the framework's effectiveness: alignment, built-in quality, transparency and program execution [Lef11]. These values guide and drive activities throughout the development process.

Alignment between stakeholders of the vision, roadmap and backlog enables organizations to react changes faster without disruptive competitive forces. Alignment is put into practice through clear content authority lines from portfolio level to team level. Aligned terminology helps to ensure processes, roles and cadence are understood in the same way

across organization. Program Increment (PI) is an iteration that is typically from 8 to 12 weeks long and delivers value through working software. It is one of the SAFe's coordination mechanism that aligns organization and teams to same cadence. PI objectives and iteration goals serve as a way to align and communicate expectations and commitments. Furthermore, cadence and synchronization prevents organization to drift too far from time and economic boundaries.

Built-in quality is key contributor to deliver new products, services or functionalities with increased lead time [Lef11c]. SAFe argues it can be achieved by embedding quality dimensions as a part of the continuous value flow. This is align with Agile principle "Continuous attention to technical excellence and good design enhances agility". Likewise, lean principle calls for a focus on building quality in. Built-in quality means developing a solution or service right from the beginning, throughout the process. This includes for example test-driven development (TDD), collective ownership and continuous intergration (CI).

Without **transparency** decision making is delusive. It is needed to create trust between stakeholders. SAFe reinforces trust through several practices. For example, Agile Release Trains (ART) have visibility to into teams' backlogs and relevant stakeholders participate inspect and adapt meeting [Lef11]. Team commitments are visible and shared with relevant stakeholders. Also shared process, common language and visible progress enhance transparency across organization.

Program Execution refers to teams and ARTs ability to deliver value reliably and efficiently and is key to prosperity [Lef11c]. Focus to execution and continuous value delivery is supported by three other core values. Program execution is supported by leaders and stakeholders whom conform the scope, remove impediments and help reflecting improvement areas.

3.2 SAFe principles

SAFe has nine principles that build on the Agile principles, Lean product development, system thinking and observation of successful enterprises. The purpose of the principles is to marshal teams to move continuously towards shortest sustainable lead time with best quality and value. SAFe practices are grounded in these principles.

SAFe Lean-Agile Principles are as follows [Lef11b]:

1. Take an economic view
2. Apply systems thinking
3. Assume variability; preserve options
4. Build incrementally with fast, integrated learning cycles
5. Base milestones on objective evaluation of working systems
6. Visualize and limit WIP, reduce batch sizes, and manage queue lengths
7. Apply cadence, synchronize with cross-domain planning
8. Unlock the intrinsic motivation of knowledge workers
9. Decentralize decision-making
10. Organize around value.

These underlying principles help organization and teams to navigate in circumstances where recommended practices do not apply. Building software and large systems can be challenging by nature; requirements are not always known beforehand or they may change. The sudden change in the market may require organization to take new direction with short notice. These are examples of circumstances in which teams and organizations may encounter. Even the principles are embodied throughout the framework, teams can also rely on principles when addressing practical problems.

3.3 SAFe levels

SAFe provides the structure by introducing four levels of organization: team, program, value stream and portfolio level as presented in Figure 4 [Lef11]. In essence, all levels are actively engaged to solution development.

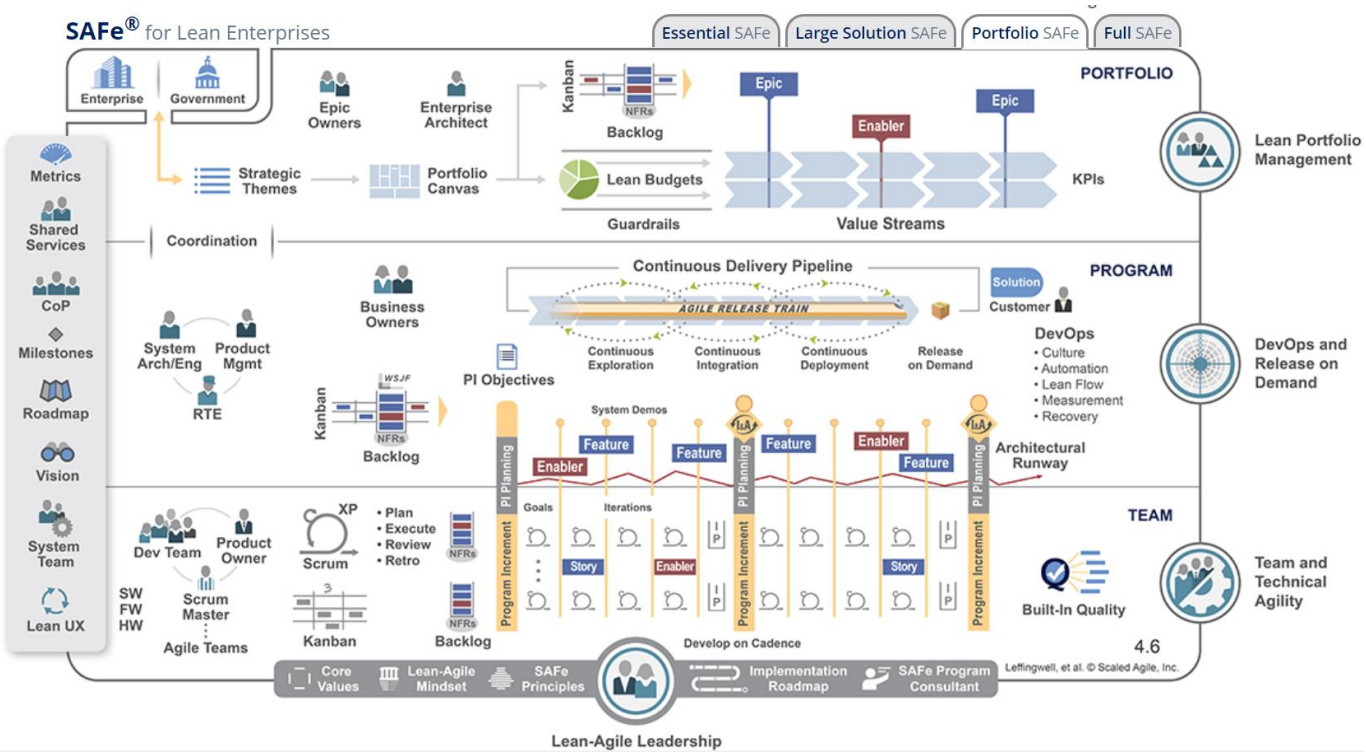

Figure 4: Scaled Agile Framework 4.6. [Lef11]

Agile **team** is one of the key components of SAFe [Lef11a]. Each agile team is collectively responsible for defining, building and testing software in fixed-length iteration and releases. Teams are self-organising and consists of developers, scrum master and product owner. Each team follows Agile methods but iterations have fixed length two weeks and they are synchronized across program level. Teams operate in identical cadence and iteration lengths in order to provide better integration among teams. In addition to traditional scrum ceremonies, SAFe has a Program Increment (PI) planning meeting to synchronize and plan next release in every five or six iterations [Lef11b]. PI Planning is a essential part of SAFe. It is occasion that helps to anchor core values of the framework into practice. SAFe defines team size to consist of five to nine team members and to have a scrum master and a Product Owner (PO). At the team level SAFe utilises practices from Scrum and XP [AR16].

Multiple agile teams form an ART [Lef11]. ART works together on a common value stream for the company. ART is responsible for developing and delivering solutions in a value stream. ART is cross-functional and have all the needed capabilities to define, implement, test, deploy and release solutions. ART operates on SAFe program level. Each

train has Product manager, Release Train Engineer (RTE) and System architect. Product manager is responsible for ART's content and backlog prioritisation. RTE facilitates program level processes and escalates impediments.

Value stream includes the whole sequence from concept to delivery of value [Lef11d]. Value stream coordinates development of large solutions and manages solution intent. Larger value streams may require multiple ARTs as smaller value stream can be realized by a single ART. Value streams are organized around PIs, which are synchronized across all ARTs. SAFe introduces two types of values streams; operational value streams and development value streams.

Portfolio level SAFe aims to align business strategy to portfolio execution through value streams [Lef11e]. It is often used by large businesses with one or more Value Streams. Portfolio backlog is highest-level backlog and it holds implementation approved enterprise initiatives called epics. Epics can either be business or architectural type. Before epics proceed to portfolio backlog they are managed through the Portfolio Kanban. Portfolio Kanban is a tool to visualize and manage epics from ideation to implementation.

3.4 SAFe Program increment

The PI is the key for teams to deliver new functionalities and products [Lef11f]. The program execution is one of the four SAFe core values [Lef11c]. Program increment is a timeboxed interval typically from 8 to 12 weeks [Lef11f]. It includes all SAFe events, such as PI planning event, system demo and Inspect and adapt meeting as illustrated in a Figure 5 [Lef11f]. The PI ensures all teams have the same rhythm and cadence.

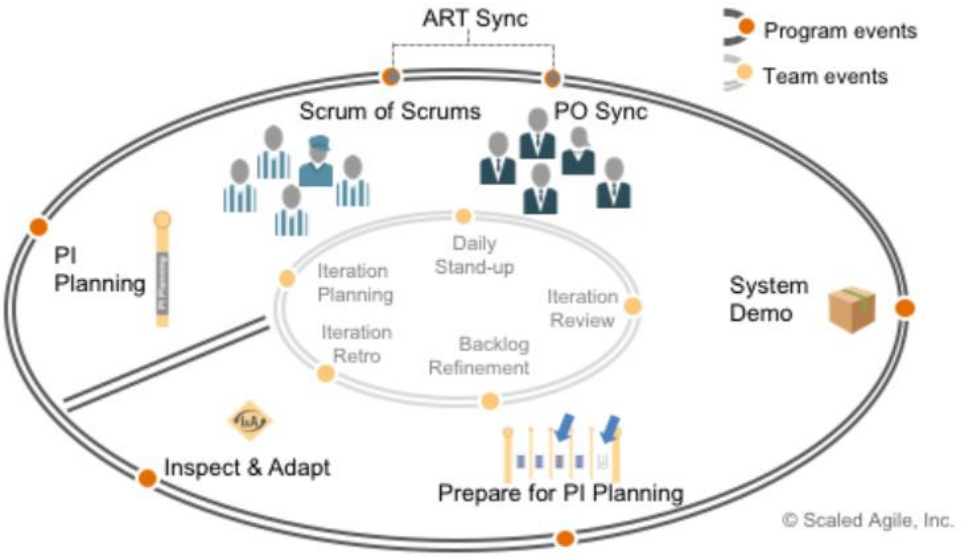

Figure 5: The visualization of PI [Lef11f]

4 Case study

The objective of the empirical part of this study is to compare the literature and theoretical framework with the findings obtained by examining a case. This chapter describes the study context, the research strategy, and the data collection methods adopted for this research. Section 4.1 introduces case company background explaining the motivation for the research. Section 4.2 outlines the research questions investigated. Section 4.3 provides overview of the research method. In Section 4.4 data collection is explained. Data analysis is described in Section 4.5. Finally, in Section 4.6, reliability and validity of the data is assessed.

4.1 Case background

The Case company is a Nordic financial company that offers financial products and services to consumers and businesses. It is globally distributed and has sites in six different locations. It uses SAFe in both business and IT organizations [CAG18]. The need to quickly respond to customers' changing needs and expectations was driving force to adopt scaled agile. The objective for the organization was to be able to deliver customer value faster while reducing complexity. In 2014 the company started to move towards scaled agile by adopting SAFe in first Agile release train with 80 people in various cross-functional roles. By the end of 2019 SAFe was widely used both in business and IT organization.

4.2 Research questions

Increasing competition drives organisations to pursue competitive advantages promoted by scaling agile methods. In the recent years, SAFe have gained increasing popularity in industry [OnV19]. Desirable advantages of the SAFe include improved productivity, reduced time to market, decreased development costs and increasing customer satisfaction [PPL18]. Although previous studies have reported increased productivity in organizational level, there is no existing research focused on a team performance evaluation in SAFe environment. As a team is a key factor that effects the organizations capability to deliver new functionalities or products, deeper understanding of practitioners experience on team performance in SAFe environment is needed. This leads us to the first research question of this paper:

RQ1: How do software developers experience team performance in SAFe environment?

The study aims to gain deeper understanding if SAFe method leads to experience of high team performance. It seeks to understand if the experience of team performance is similar between individual developers. This is a view of the problem from the point of the software developers'.

RQ2: How applied practices support team performance?

SAFe introduces set of practices to be applied in teams and by organization. The problem of the research question can be approached by first determining the practices that sustain and restrain the team performance. Then, by categorizing these factors as constructs, overall picture of their effects can be formulated.

RQ3: How does team performance correlate to the applied practices?

SAFe introduces a set of practices to support large organizations to align and deliver working software. How do these introduced practices impact perceived team performance?

4.3 Research methods

A case study usually aims to understand current phenomenon within it's natural context, explaining causal relationships which are impacted by surrounded complex real life conditions [Yin14]. According to Runeson, case study is suitable methodology for software engineering research since it studies contemporary phenomena in its natural context [RH09].

The purpose of this thesis is to understand how software developers perceive and reason about team performance in organization that has adopted SAFe. This thesis examines antecedents and potential team performance consequences of SAFe. The major contribution of this thesis is to provide insight into how team performance is seen by practitioners. The qualitative research type was selected for this thesis because the aim is to explore current phenomenon. As this study explores the subject in real life setting, case study was chosen as a method. Case study suits well for research aiming to answer how or why something occurs and researcher has little or no control over behavioural events [Yin14]. This thesis is an exploratory single-case study. According to Runeson and Höst exploratory case study tries to understand phenomena through the participants interpretation of their

context [RH09].

Research in this thesis focuses on the level of individuals. The organization's aspect on team performance has not been studied.

4.4 Data collection

There are several methods to collect data for a qualitative research such as interviews, surveys, protocols, stories and observations [Yin14]. In this qualitative case study research data was collected using thematic, semi-structured interviews. In thematic interview researcher has possibility to clarify and deepen the understanding by asking additional questions and asking reasoning for the answers [RH09]. In a semi-structured interview discussion is flexible and leaves room for interviewee perspectives and open conversation [RH09]. A semi-structured interview is a mix of open and closed questions where question order and wording may vary.

Data was gathered by interviewing 5 experienced developers working in case organisation. Interviewees were working in different development teams in order to identify the differences and commonalities. The interviews were conducted as anonymised. This allowed interviewees to speak freely their perspectives during the interviews. The interviews were conducted in Finnish and they were recorded using Skype. Interviewees were selected together with case company manager. The criteria for interviewee selection were as follows: a) they should not work in a same development team b) they should work in different SAFe ART c) they should have at least two year experience from the case company. Once potential candidates for interview were identified, they were contacted by e-mail and asked if they would be wiling to participate the research.

The goals for these interviews were to identify practices that were perceived supportive for team performance perspective and to unveil aspects that are perceived slow down the work. The interview was divided to following themes: background, team, work environment/organization, experience example, team performance. The final interview content is listed in Table 1 and Table 2.

Theme	ID	Question
Background	Q1	When did you start working here?

	Q2	What is your role?
	Q3	How long have you worked with software development?
	Q4	What kind of Agile experience you have?
	Q5	How familiar you are with SAFe?
Team	Q6	Are you part of some team or several?
	Q7	When did you join the team?

Table 1: Information about the interviewees

In the initial phase, interview format was tested with case company's software development manager. Based on the pilot interview, some interview instructions required restructuring to improve clarity. For example research purpose and instructions were modified. In addition, the information about anonymity was decided to be shared prior the interview.

Theme	ID	Question
Work environment / organization	Q8	How is SAFe impacting working habits of your team?
	Q9	How is SAFe impacting working habits of your organization?
Experience examples	Q10	Give examples of successful or unsuccessful work experiences. On what scale did you succeed or fail?
Team performance	Q11	In what way you and why do you think you succeeded?
	Q12	Do you think it have anything to do with SAFe?
	Q13	Can you think of SAFe practices that support / hamper your team performance?
	Q14	Has SAFe improved something in your team?
	Q15	What would you like to be different in order to enable better team performance?
	Q16	Any other comments about SAFe in relation to team performance?

Table 2: Information about the interviews

The data collection was gathered during two week period in March 2020 through individual face-to-face interviews. Five senior software developers were interviewed. The interviewees' experience at the company ranged from 3 to 5 years as illustrated in Table 3. The interviewee software development experience varied between 6 to 25 years and agile development experience varied from 4 years to 15 years. All interviewees were located in Helsinki. The duration of interviews were 1 hour on average. All interviewed persons were promised that only anonymous data would be presented externally and internally in the organization.

Interviewee	**Experience in case company**	Software development experience	Agile experience
A	4 years	6 years	4 years
B	4 years	15 years	11 years
C	3 years	25 years	15years
D	5 years	16 years	9 years
E	4 years	20 years	13 years

Table 3: Information about the interviewees

When doing interviews, there was one researcher asking questions and taking notes. In addition to the notes, all interviews were recorded. During the interview focus was in concrete examples to keep the discussion focused and to get as relevant data as possible. During the interviews the flexibility was kept by using thematic interview techniques to deepen the topics.

4.5 Data analysis

Data analysis started immediately after the first interviews. In the analysis phase, data was gone through by listening interview recordings several times. The interview recordings were transcribed. The transcriptions were written with Microsoft Word. Next, the researcher appended interviews with researcher's notes. The transcribed interviews were printed and reread.

The interview data was analysed using inductive data analysis. Inductive data analysis refers to technique where the data is first collected and then more general themes are derived from the data [Yin14]. In inductive coding researcher does not have preconceived codes or themes but instead codes are developed from the data and themes [Yin14]. Microsoft excel was used to process the data.

5 Results

This chapter describes the findings from the interviews. The findings provide an overview on how developers perceive team performance in the case company. In Section 5.1, findings related to the SAFe framework are presented. Section 5.2 describes organizational aspects that came to the fore at the interviews. Section 5.3 focus on team performance and findings are compared to theoretical considerations found in literature. In Section 5.4, general observations are presented and they are compared to theoretical findings.

Cited interview comments are quoted. Included citations highlight different aspects of interviewees and provide accurate insights into interviewee thoughts.

5.1 Framework practices

SAFe defines a set of practices to support business agility. It involves multiple different organizational units including business, development teams and management.

In response to the questions, *How is SAFe impacting working habits of your team? How is SAFe impacting working habits of your organization?,* all interviewees experienced SAFe did not impact considerably on team operations. In fact, they felt that their team was following scrum, Kanban or scrum-ban and that it was the most obvious actor impacting team practices.

In general SAFe was seen to provide structure for teams and organization. The Biggest impact was seen on macro level, as nearly whole organization followed the same framework and same iteration cadence. On the other hand, developers were not always sure if all organizational units applied the SAFe, especially when there was structural distance in organization. One developer anticipated that some legacy teams might still follow waterfall model as even a change planning could take several months. Respondent generally acknowledged high level coordination to be more transparent when whole organization follows same iteration cycle and process.

Goals and new features were seen coming from a higher level and possibility to impact to the new functionality or its priority was seen somewhat limited. One of the respondents commented that it was possible to communicate if the team wanted to change the priority but the possibility to affect the decisions was low. Respondent was assuming that SAFE would support two way communication better.

All interviewees experienced that PI provides structure and alignment between teams. It was also seen to increase transparency across the organization.

5.1.1 Framework understanding

As visible in the Table 4, only one of the interviewed developers had got official SAFe training. Others had done self-studies and learned by doing. Practical SAFe experience was between 2,5 and three years.

Differences in the framework knowledge were not considered in this research when doing the interviewees. However, it could be that received training affected interviewees interpretation.

Interviewee	SAFe experience	SAFe training
A	2 years	No official SAFe training, self-studies
B	3 years	Leading SAFe, SAFe scrum master
C	2,5 years	No official SAFe training, learning by doing
D	3 years	No official SAFe training, self-studies, learning by doing
E	3 years	No official SAFe training, short introduction organized by case company, learning by doing

Table 4: Information about the interviewees SAFe training & experience

5.1.2 PI Planning

Most apparent SAFe practise seemed to be PI planning and the cadence around it. All developers mentioned PI planning as a prominent occasion associated with SAFe. Even though the PI event was seen to provide alignment and transparency between teams, several doubts were expressed about the PI planning. The following two themes emerged

from the developer responses: *event necessity evolves over the time* and *length of the forecast horizon*.

It became clear from the interviews, that the PI planning **event necessity evolves over-time.** When organization is new with SAFe adoption, face to face PI event is emphasized to be more important. Developers felt that over the time PI planning is not necessity. Furthermore, need for a co-located PI planning was challenged: "At the beginning it should happen face to face. But at the end, we utilize it less." Another comment related to a complexity of the planning in situation when organization was new with the framework: "At least at the beginning when there was a consulting company supporting with SAFe adoption, it felt that PI event ceremony included a lot overhead and did not bring much value for doing." As teams' and organizations' levels of SAFe maturity increased, they started to adjust applied practices. This seemed to receive positive reaction among developers. One of the developers commented about the PI planning as follows: "We don't participate the PI planning anymore. Only Product owner and Scrum master will visit the event once needed. This way we save developers' time. We plan in our own team area and I feel it is better way. Compared to, for example, flying the whole team to Denmark to sleep in the event for two days. This is more efficient way. But when we establish new project it is very good to meet the people. But when it's 13th PI planning, it becomes less important to see face to face. "

The Length of the PI forecast horizon was considered to be too long among interviewees. Developers believed that forecasting was possible only for couple first sprints. One of the developers summarized this as he stated: "Another critique about PI planning is that perhaps we try to forecast too long, 6 sprints is too much. Normally we can forecast couple first sprints, but the last sprints of the PI we do things that we didn't know in the PI planning event. Later sprints do not matter. Only couple first sprints can be done according to a plan."

One software developer thought that organization did not succeed to solve problems related to PI planning and that same issues were present from PI planning to PI planning despite of the feedback.

The purpose of the PI planning is to align teams on the shared mission and vision [Lef11]. Dependencies are one of the key drivers for having the PI planning. In general, PI planning was seen as effective at identifying and agreeing on dependencies. However, several

dependency related problems were identified during the PI execution. This will be discussed in a separate section that follows.

5.1.3 System Demo

One of the SAFe practices that was mentioned by three interviewees was the system demo. It evoked ambivalent feelings among interviewees. Two of the interviewees said it was important way to get feedback and improvement proposal. It was perceived to support the team to achieve good results, which created happiness within the team. Team also felt they were trusted. The key for success was perceived to derive from continuous communication that still left room to do the development without micro level control. On the other hand, another interviewee felt frustration with demoes as he mentioned: "Many times we demo before the functionality is completely ready. So we do the user interface and we have some test data, even it would be mocked. We demo the functionality jus for demoing, just to show we have done something. Even it would be 30 % ready. We might continue working with the functionality for another two or three weeks. And once it's ready we show it again. It still looks the same but this time it really works."

5.1.4 PI execution

SAFe implements short development iterations and longer Program Increments. In case company typical PI is 12 weeks including 6 sprints.

Several hampering effects on team performance were reported to occur during the PI execution. One of the mentioned consequence was slipping from the committed. This was a risk that was mentioned mostly in relation to dependencies. However, it was also mentioned in context of team not being able to deliver committed features during the PI. Most frequently the reason was in unknown occasions that were not known when the team did PI plan. One of the mentioned unforeseen occasion was changes in team members. This might lead to a situation that the high priority work team was committed to was moved to another teams backlog. In practise this meant that some developers had to suddenly stop what they were doing and pick the new more important tasks to support higher level goals.

Another reported factor was the length of the PI. All interviewed developers felt that PI for six sprints was too long period to plan and commit.

5.2 Organizational aspects

Organizational aspects such as culture, policies, organizational goals, organizational structure, training and resources have been found to contribute to team performance. These aspects were also present in comments made by the interviewees. Majority of interviewees felt that conflicting priorities and ability to efficiently solve them prevented teams to deliver new functionality. Empirical data indicates that organizational goals are not always aligned and that the company is lacking systematic way to address potential goal conflicts. Interviewed developers elaborated that the current way to manage conflicting priorities is causing difficulties in delivering features. One interviewee described the situation in this way:

> "The same feature is always part of the plan, but because of the dependencies we never even start it. The basic problem is, that when we are in the PI, we need something from some other team and if the other team doesn't proceed we don't proceed either. And this occurs from PI to PI. The same feature is always part of the PI plan but we never do it because of dependency. "

Interviewed developer also highlighted that team gets negative feedback for not being able to deliver the feature. However, interviewed developer felt that it was not team's problem if the same feature was kept on the backlog without being able to solve the priority conflict with the delivering team.

Another expressed concern related to company's capability to adjust and provide support in different domains. Developers explained that teams usually fought to get support from the design team, but were rarely receiving it in time. This prevented them to proceed with the development. They escalated the problem with design team several times, but the organization was not able to change the structure to more supporting direction. So the features kept being in the PI plan and developers felt even more frustrated.

When talking about priority, decisions related to priorities and organizational goals, interviewed developers expressed concerns if the higher management understood the problems and related implications. One of the mentioned examples was test environment instability. This was the problem that was identified years ago but it has never been fixed. If test environment was down, it paralysed the whole development. As a result, dozens, if

not hundreds, of developers and testers we not able to continue their work. This issue could last from hours to days. One of the interviewees expressed his annoyance, as he stated: “How can such a prevalent attitude nest in the company that we don’t need to react on a problem? How can we show the cost for management to make the issue visible and to get the priority to solve it.”. This finding indicates that bottom-up information flow has severe implications to team performance and should be therefore consider more detailed.

Another finding relates to dependencies towards other teams. Three interviewed developers mentioned dependency towards backend teams to prevent their team to proceed with development. This was seen as a major obstacle for development. One of the developers explained that nearly always when backend and design was in place, team was able to deliver new functionality with speed. This problem could be approached by examining the possibility to establish cross-functional teams.

5.3 Team performance canvas

There appeared to be consensus among developers what success meant: something is released to customers in agreed schedule. Interviewed software developers described factors that they perceived to disrupt or to enhance the team performance. Three top-level categories emerged from the discussions: *goals and objectives*, *support* and *learning.*

Goals and objectives were seen important factor affecting the team performance. Clear goals enabled the team to focus on what is important and to communicate it with stakeholders. If the objective or goal had high priority and it had tight timeline, the performance impact was even more considerable. One interviewee described his success experience as follows: “The latest success story we worked with had very tight deadline. The feature was high priority and everyone understood the importance. Several teams were involved and everyone understood how important is was. We had the vision what should be done but we didn’t know in detail how to do it and how an infrastructure would support the development. But SAFe gave us supporting frame how to agree and collaborate across the team and train borders. The goal was clear and everyone provided support to complete it, even stakeholders outside the ART.”

The above example also clarifies the role of **support**. When the goal has high priority, it is easier to communicate to stakeholders and get support from them. It also clarifies to team members how to prioritise the work and where to focus. Clear goals were also seen

to foster support between team members. If someone was blocked with the task, it was easier to get help from others.

Another factor that was perceive to enforce team internal support related to **learning**. If team members were able to develop cross-platform, it increased team's flexibility. Team was not that dependent on one person who knew particular platform but instead others could contribute and support as well. One developer emphasized the importance as he stated: "One thing that I would change in our setup is that let's forget the roles within the team. We shouldn't have one develop focusing on iOS and one doing Android or backend. Everything is so much more flexible when you think of achieving team goals. In general people are interest in to further develop them self. We should just give a change for them. Team should decide to invest on the learning and team should have positive approach to cross-platform development."

Above quote reflects developer's thoughts about learning. Team had decided to take an action to enhance developers skills and to build cross-platform competence. This made it easier for them to adjust to changing priorities. Following strictly the PI plan with accurately calculated availability and capacity was not needed anymore. Team was able to plan more flexible and focus to goals and priorities instead of estimating each story and task and checking if developer with needed skill had enough capacity in each sprint. This was perceived to improve the whole planning process.

In addition to above mentioned topics, the team stability was mentioned to have positive impact to trust between team members. Team stability seemed to create trust and confidence between team members. One software developer commented the impact of team's long journey together as he stated: "I could say that in our team the competence level is very high. It doesn't matter what topic the functionality is related to, we can get on track in couple of days. And never we have the feeling this can't be done if we just get enough information. Anything can be done."

5.4 General observations

There were many aspects worth mentioning that were discussed during the qualitative part of the research. Additional findings include dependencies and collaboration.

5.4.1 Dependencies

In complex and large enterprise environment, agile teams have many dependencies. Dependencies seem to be the major cause for teams not to be able to perform better. Dependencies were raised by all interviewees and sawn to hamper the team performance. Dependencies are one of the key drivers for the PI event. In general, it seemed that teams were able to address and agree about the dependencies in the PI planning event, but during the PI execution commitments were not always kept. This caused various issues for teams depending on someone else.

One developer explained the situation where poor dependency management resulted in an commitment problem, as he stated: "Typical case is that we do handshake, promise that something is ready in sprint 3, but we don't have any tool for dependency follow-up and we don't monitor that dependency is ready when we need it. Then if we don't get it, we must postpone it."

Another developer explained his experience from the poor dependency management in the PI planning event as follows: "Nowadays PI planning is one day. It's ok but on the other hand we don't map the dependencies on a same level, which is not good thing. And the dependencies are the main thing. And the PI board is so messy that no one understand it's and it seems that no one really cares what is on the PI dependency board."

"At least dependencies were not thought detailed enough during the PI event. During the PI we noticed that we didn't understand the dependency correctly and there are things we didn't consider in the PI planning. It could be team's fault, but on the other hand also other organs could monitor the dependencies. So it slipped also from them."

The reasons for not being able to keep the commitment was reasoned to relate conflicting and changing priorities. In addition, technical obstacles and resourcing were mentioned to prevent delivering committed dependencies.

5.4.2 Collaboration

Even the collaboration was emphasized challenging, there were also many positive reflections. Generally, the more people there were involved to the development process, the more challenging and slower the development was. Also the risk of undelivered commitment increased if several teams were involved or if teams were in different organizational unit.

Another important factor enhancing the collaboration was emphasized to be face-to-face

meeting. All interviewed developers agreed the importance of it. It was mentioned that specially in situations when people didn't know each other's before. One developer expressed the impact of face-to-face by stating: "Earlier collaboration with them was really hard. But right after we met physically the first time, everything started to run smoothly."

Despite the identified problems that relate to team performance, in general the framework was seen supporting collaboration in such a large organization as the case company. One of the developers expressed this by saying: "It gives very good frame for such a large organization. It really supports collaboration across organizational units. If someone is not following SAFe, it is much harder to find common priority because we don't think about prioritization in same way. I can't think of any other way how this large organization could operate better."

6 Discussion

In this section the findings of the thesis are being discussed. First, contributions to research questions are being summarized and discussed. Next, practical implications that emerged from the findings are considered and recommendations for the case company are provided. Finally, validity and future research is discussed.

6.1 Addressing the research questions

This section summarizes and discusses the answers to the research questions. Results presented in Chapter 5 are compared to related research.

RQ1: How do software developers experience team performance in SAFe environment?

The empirical results of this study indicate that the software developers experience team performance as something that evolves over time and is prone to be influenced by environment. They perceive performance to be impacted by factors including individual, team and organizational dimensions.

Software practitioners' perception of the team performance determinant factors were seen to come from contribution outside the team. Conflicting priorities was recognised as having significant impact to team deliveries. There are number of possible explanations for this pattern. The structure of organization may have impact on prioritization conflicts. On the other hand, scattered units may lead to the boundaries that distract common objectives. In some respects, these findings are align with findings of Fagerholm et al., who investigated software developers experience on team performance in Lean and Agile environment [FIK14].

Many of addressed concerns related to collaboration across organization. There were diverse views on the role of the framework. Most of the interviewees perceived it to support collaboration by providing aligned way to communicate and think, while some saw framework not to provide support for detected collaboration issues. The reported experiences has similarities to findings made by Laanti and Kettunen [LK19]. In their study, one of the most significant benefit of SAFe usage was co-operation. However, their research covered also non-software companies. Their survey was distributed through social media, so it could be that also other than software developers responded to it. Therefor respondent views might differ from perceptions of this study's participants.

RQ2: How applied practices support team performance?

SAFe introduces several activities such as short iterations, early feedback and close collaboration. These fundamental activities help teams to align with organizational goals leading to higher performance as observed when researching Agile teams [MP17].

After analysing the different perceptions of team performance in SAFe environment, we observed that the organization adjusted the SAFe framework utilization to fit the realities of the organization. The research findings suggest that this enhanced the team performance experience. This finding was somewhat unexpected because our attention was to research practices recommended by the framework, and not to focus on organizational adjustments. However, this observation is in line with Dikert, Paasivaara et al. literature review about challenges and success factors for large-scale agile transformations [DP16]. They report customizing the agile approach being one of the most mentioned success factors for large scale agile transformations [DP16].

RQ3: How does team performance correlate to the applied practices?

This research question was not addressed because it was not amenable to the qualitative nature of the research setting. Interviews did not answer directly to the research question and either did data analysis extract factors that could be measured and compared. However, interesting observations were made that could contribute to research question.

For example, dependency management was one of the most reported issue. Shortage in dependency management was seen to cause several delays for the teams. It would be possible to measure the impact of dependencies to teams delays and to evaluate if it correlates with team performance. However, the measures should be clearly define in the research approach to be able to make conclusions.

Another interesting finding was, thattTeam members capability to cross-platform development was perceived to increase teams flexibility and to decrease dependencies. It also had positive impact to team members' motivation. This observations is align with the results presented by Magpili and Pazos in their literature review [MP17]. Also SAFe encourages to create end-to-end teams [Lef11].

Above mentioned observations do not directly relate to applied practices but instead it is something that organization could undertake to increase the team performance

6.2 Recommendations to organization

The results lead to recommendations that can help organization to address found issues impacting the team performance. Following areas for adaptions and new practices are suggested for the case company: *Priority conflicts, dependency management, organization structure, team learning*.

Priority conflicts were addressed to be one of the major obstacles for teams to proceed with development. In a case of conflicting priority, the most visible symptom was undelivered dependency. Organization should have a process to determine the conflicting priorities. As the issues was seen harder to solve if priority conflict occurred in different organizational division, the process should cover cross-portfolio patterns. The case company could examine more detailed the process and structure around conflicting priorities.

Dependency management closely relates to priority conflicts. However, the root cause for undelivered dependency could also relate to other obstacles. To detect the dependency related issues, organization should focus to dependency management during the PI execution. It could investigate if the current backlog and dependency management tool provides functionality to trigger an alert if dependency is not solved in agreed time. It could also investigate if the tool could detect a feature that is kept on the PI backlog for several PIs. This should evoke, for example, Release Train Lead (RTL) to investigate the issue more detailed and to escalate the issue it if needed.

Organization structure related issues could be investigated more detail to gain understanding of real root-cause. This research found out that some areas, for example design, were identified to be a bottleneck. Organization could examine how severe issue this is and if there could be improvements done through re-organizing and by adjusting the design related processes.

Another finding related to dependencies between backend and client teams. Proposal for the organisation is to encourage teams to have end-to-end development capability. This would decrease dependencies towards other teams. This would also bring organizational flexibility and adaptability. RTL together with Scrum masters could investigate how to create atmosphere that encourages developers to further extend their skills and to book time for the personal development. This action would contribute also to **team learning.**

6.3 Validity

Validity expresses the degree of trustworthiness of the results and conclusion [RH09]. It reflects if a study results are true and not biased by the researcher presence. In this study, validity threats were analysed through a classification scheme which distinguishes four aspects of the validity: *construct validity, internal validity*, *external validity* and *reliability* [RH09].

Construct validity refers to how well the operational measures reflect the actual construct it intends to measure [RH09]. For example constructs discussed in the interview could be interpret differently by the researcher and the interviewed person. To ensure quality in data collection in this study, interviews were tested with a pilot interview and interview questions were reviewed by thesis supervisor. A potential weakness of this study is that interviewed developers were guessing what they were expected to say or that they discussed issues that they expected researcher wanting to hear. To improve construct validity, the interviews questions referred to real life success and unsuccessful experiences.

Internal validity concerns cause-effect relationship [RH09]. In this research internal validity was enhanced by keeping the interview time span short. Interviews were conducted in three week period.

External validity implies if the results of a study can be generalized beyond the study itself [RH09]. External validity was addressed by selecting interviewees according to before defined participant criteria together with the case company presentative. It was also seen important that all interviewed developers had sufficient time spent with the organization, software and agile development in order to understand it. Moreover, the relative lack of women in developer position in this study reflects proportion in the software development.

Reliability refers to concept to what extent the results are consistent and repeatable [RH09]. In this research, threats to reliability lie in human factors involved. As the researcher that interpret and analyse the results came from the same organization and had long-term experience of it, it could be that she provided more comprehensive view of the subject and perceived results differently. Moreover, the data was collected from

respondents who worked in one site of organization. It could be that respondents were giving monolithic and uniform answers due to site specific characteristics.

6.4 Future research

Findings from this research show that the team performance is a concern that teams and organizations must pursue continuously. Even the desirable team performance level is reached, it should not be treated as a self-evident fact. Team performance evolves and is inclined to comply with surrounding conditions. This understanding gives a ground for the future research to continue exploring the phenomenon.

Furthermore, findings from this study were collected from one company operating in financial industry. For the future research, it would be interesting to repeat the research as multiple-case study focusing to other industries and markets.

7 Conclusions

In this study the aim was to explore how software developers perceive and reason about team performance in SAFe environment. We conducted a case study in which software developers from the company utilizing the SAFe were interviewed. As a result it was recognized that the team performance was perceived to be something that adjusts to prevailing conditions. Many factors were seen to hinder and to contribute to team performance. There was no one factor given why the team was able to perform well. Instead, we identified several arguments that impact practitioners experience about the team performance. First, organization has strong impact on teams and their behaviour. It's ability to adjust and react on identified issues is one of the key elements. Next, the SAFe framework provides good ground for scaling the whole organization agile. However, it should be adjusted to fit the organizations processes and need. The SAFe was seen to bring structure and coordination on high level, which was seen to improve the team performance. Nevertheless, from team and developer point of view there were many performance factors that tie together the people, organization and the framework. Sometimes it was hard to draw a clear line between these categories. For example the dependency management and the prioritisation issues could be seen to be impacted by the organization structure or the framework related issues.

For future research, it would be interesting to repeat the research as multiple-case study focusing to other industries. Another interesting research issue to explore, is to compare team performance experience in such organizations where different scaling Agile frameworks are applied and to compare the results.

References

AWS03 Abrahamsson, P., Warsta, J., Siponen M., Ronkainen, J.,New directions on agile methods: a comparative analysis, In Proceedins of the 25th International Conference on Software Engineering, ICSE ´03, p. 244-254, IEEE, USA, 2003.

Agile01 Agile Manifesto. Alliance: Manifesto for Agile Software Development [online], 2001. http:www.agilealliance.org.

AR16 Alqudah, M. and Razali, R., "A review of scaling agile methods in large software development," International Journal on Advanced Science, Engineering and Information Technology, vol. 6, no. 6, pp. 28–35, 2016.

AVH17 Ayed, H., Vanderose, B., Habra, N., Agile cultural challenges in Europe and Asia: insights from practitioners, IEEE, Proceedings of the 39th International Conference on software engineering, p. 153-162, 2017.

Bec00 Beck, K., eXtreme Programming Explained: Embrace Change, Addison-Wesley, San-Francisco, USA, 2000.

BT05 Boehm, B., Turner, R., Management challenges to implementing agile processes in traditional development organizations. IEEE software 22(5), (2005), pages 30-39.

Cas17 Van Casteren, W., The Waterfall Model and the Agile Methodologies: A Comparison by project characteristics, Paper on software development methods, ResearchGate, 2017, https://www.researchgate.net/project/Paper-on-software-development-methods [03.11.2020]

CAG18 Core Agile Guidelines, Company confidential, 2018.

DP16 Dikert, K., Paasivaara, M., Lassenius, C., Challenges and success factors for large-scale agile transformation: A systematic literature review, The Journal of Systems and Software 119, pages 87-108, 2016.

DF16 Dingsøyr, T., Faegri, T., Dyba, T., Haugset, B., Lindsjorn, Y., Team Performance in Software Development: Research Results versus Agile Principles, IEEE Software, Vol. 33(4), p.106-110, 2016.

DT08 Dybå, T., Dingsøyr, T., Empirical studies of agile software development: A systematic review, Information and software technology, Vol. 50(9-10), p. 833-859,2008.

ELS05 Erickson, J., Lyytinen, K., Siau, K., Agile Modeling, Agile Sfotware Development, and Etreme Programing: The State of Research Journal of Database Management, vol. 16(4), p. 88-100, 2005.

FCS10 Falessi, D. Cantone, G., Sarcia, S, Peaceful Goexistence: Agile Developer Perspectives on Software Architecture

FIK14 Fagerholm, F., Ikonen, M., Kettunen, P., Munch, J., Roto, V., Abrahamsson, P., How do Software Developers Experience Team Performance in Lean and Agile Environments? Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, Article No.:7, (May 2014), pages 1-10.

GD96 Guzzo, R., Dickson, M. Teams in organizations: Recent research on performance and effectiveness. Annual Review of Phychology, vol. 47(1), p. 307-338, 1996.

Hir08 Hiranabe, K., Kanban applied to software development: From agile to lean. http://www.infoq.com/articles/hiranabe-lean-agile-kanban [10.11.2020]

HJ07 Hall, T., Jagielska, D., Baddoo, N., Motivating developer performance to improve project outcomes in a high maturity organization, Software Quality Journal, Vol. 15(4), p. 365-381, 2007.

Kaj08 Kajko-Mattsson, Mira, Problems in Agile Trenches, Proceedings of the Second ACM-IEEE international symposium on empirical software engineering and measurement, ACM, p. 111-119, 2008.

LB03 Larman, C., Basili, V., Iterative and incremental development: a brief history, IEEE Computer, Vol.36(6), page 47, 2003.

LK19 Laanti, M., Kettunen, P., SAFe Adoptions in Finland: A Survey Research.

Lef11 Leffingwell, D., Scaling Agile Framework (SAFe), 2011. http://www.scaledagileframework.com/ [24.02.2020]

Lef11a Leffingwell, D., Scaling Agile Framework (SAFe), Agile Teams, 2011.

https://www.scaledagileframework.com/agile-teams/ https://www.scaledagileframework.com/safe-lean-agile-principles/ [14.11.2020]

Lef11b Leffingwell, D., Scaling Agile Framework (SAFe), Lean Principles, 2011. https://www.scaledagileframework.com/safe-lean-agile-principles [23.11.2020]

Lef11c Leffingwell, D., Scaling Agile Framework (SAFe), Core values, 2011. https://www.scaledagileframework.com/safe-core-values [14.11.2020]

Lef11d Leffingwell, D., Scaling Agile Framework (SAFe), value streams, 2011. https://www.scaledagileframework.com/value-streams [12.11.2020]

Lef11e Leffingwell, D., Scaling Agile Framework (SAFe), Portfolio SAFe, 2011. https://www.scaledagileframework.com/portfolio-safe [14.11.2020]

Lef11f Leffingwell, D., Scaling Agile Framework (SAFe), Program Increment, 2011. https://www.scaledagileframework.com/program-increment/ [21.11.2020]

Lef11w Leffingwell, D., Scaling Agile Framework (SAFe), SAFe White Paper, 2011. Leffingwell, D., Scaling Agile Framework (SAFe), Program Increment, 2011. https://www.scaledagileframework.com/program-increment/ [21.11.2020]

LMD04 Lindvall, M, Muthig, D., Dagnino, A., Wallin, C. Stupperich, M., Kiefer, D., May, J., Kahkonen, T., Agile software development in large organizations, Computer, Vol. 37 (12), p. 26-34, 2004.

MDD09 Moe, N., Dingsøyr, T., Dybå, T., Overcoming Barriers to Self-Management in Software Teams, IEEE Software, Vol. 26(6), p. 20-26, 2009.

MMR08 Mathieu, J., Maynard, M., Rapp, T., Gilson, L., Team Effectiveness 1997-2007: A Review of Recent Advancements and a Glimpse Into the Future, Journal of management, vol.34(3), p.410-476, 2008.

MP17 Magpili, N., Pazos, P., Self-Managing Team Performance: A Systematic Review of Multilevel Input Factors, Small Group Research, Vol. 49(1), p. 3-33, 2017.

OnV19 One, V.: State of Agile Survey. https://www.stateofagile.com/#ufh-i-

521251909-13th-annual-state-of-agile-report/473508 [02.03.2020]

OpB04 Oppenheim, B., Lean Product Development Flow, Systems engineering, vol. 7, 2004

PD12 Prause, C., Durdik, Z., Architectural design and documentation: waste in agile development?, Proceedings of the International Conference on Software and System Process, p. 130-134, ACM, 2012.

PoP03 Poppendieck, M., Poppendieck, T., Lean software development: An agile toolkit. Addison Wesley, Boston, Massachusetts, USA.

PoP07 Poppendieck, M., Lean Software Development, 29th International Conference on Software Enineering, pages 165-166, 2007.

PoP12 Poppendieck, M., Cusumano, M., Lean software development: A Tutorial, IEEE Software, vol. 29, pages 26-32, 2012-09.

PPL18 Putta, A., Paasivaara, M., Lassenius, C., Benefits and Challenges of Adopting the Scaled Agile Framework (SAFe): Preliminary Results from a Multivocal Literature Review, Product-Focused Software Process Improvement, Vol. 11271, p. 334-351, 2018.

Sta97 Stapleton, J., Dynamic Systems Development Method: The Method in Practice, Addison-Wesley, 1997.

RaS98 Raman, S., Lean Software development: Is it feasible?, In Digital Avionics Systems Conference, IEEE, vol 1, pages C13/1-C13/8, 1998.

RH09 Runeson, P., Höst, M., Guidelines for conducting and reporting case study research in software engineering, Empirical software Engineering, 14, pp. 131-164, 2009.

RT92 Rasch, R., Tosi, H., Factors affecting software developers' performance: An integrated approach. MIS Quaterly: Management Information Systems, Vol. 16(3), p. 395-413, 1992.

SS14 Sutherland, J., Schwaber, K., "Scrum Guides", https://www.scrumguides.org [09.11.2020]

SSB16 Salas, E., Sims, D., Burke, C., Is there a "Big Five" in Teamwork?, Small

group research, Vol.36(5), p.555-599, 2016.

TZF11 Tamer, M., Zulkhairi, D., Fauziah, B., Content Analysis on Agile Values: A Perception from Software Practitioners, IEEE Malaysian Conference in Software Engineering, p. 423-428, 2011.

Woh14 Wohlin, Cleas, Guidelines for snowballing in systematic literature studies and replication in software engineering, Proceedings of the International Conference on Evaluation and Assessment in Sofware Engineering, Article No: 38, p. 1-10, 2014.

Yin14 Yin, R., Case study research: design and methods. Sage Publications, 5th edition, USA, 2014.

Appendices

Appendix 1. Interview sturcture

Theme	Guiding question	Purpose
Background	When did you start working here? How long you have worked with software development? What kind of Agile experience you have?	Understand subject's experience from the company and earlier experience
Team	Are you part of some team or several? When did you become involved? (How has the team composition changed along the way?)	Understand participant's previous and current involvement in a team / teams.
Work environment/organisation	How is SAFe impacting working habits of your organisation?	Elicit participans view on the organisation and SAFe
Experience examples	Give examples of successful or unsuccessful work experiences. On what scale did you succeed or fail?	Find concrete examples
Team performance	In what way you and why do you think you succeeded (failed)? What do you think was reason for success / failure? Did the failing somehow benefit the team or organization? Can you think of SAFe practices that support /	

	hamper your team performance? (How do you think SAFe meetings / ceremonies impact your team performance?) Has SAFe improved something in your team? What would you like to be different in order to enable better team performance? Any other comments about SAFe in relation to team performance?	