MASTER'S THESIS | LUND UNIVERSITY 2017

Evaluating the implementation of SAFe with focus on test and quality

Sandra Fridälv

Department of Computer Science
Faculty of Engineering LTH

ISSN 1650-2884
LU-CS-EX 2017-02

Evaluating the implementation of SAFe with focus on test and quality

Sandra Fridälv
Sandra.Fridalv@gmail.com

February 9, 2017

Master's thesis work carried out at
the Department of Computer Science, Lund University.

Supervisors: Christin Lindholm, Christin.Lindholm@cs.lth.se

Examiner: Martin Höst, Martin.Host@cs.lth.se

Abstract

Today it gets more and more popular for companies to implement Scaled Agile Framework (SAFe) in their organizations. SAFe is an agile software development framework that intends to increase productivity and quality, and provide faster time to market. A company that works in a waterfall based development process, is now facing this implementation. The company would like to know which challenges other companies have when implementing SAFe and how they have solved these challenges. Especially they want to know what challenges and changes there are in test and quality when implementing SAFe. With this knowledge, the company can be more prepared if they face similar challenges during their own implementation.

In this thesis work 17 different companies have been interviewed about their experience of implementing SAFe. Results from the interviews were compiled and analyzed to see if there were any similarities between the companies' characteristics and their answers. The analysis resulted in a clear difference in answers between companies with many people involved in SAFe and the companies with fewer people involved in SAFe. It also resulted in some best practices on how to solve the most general challenges that companies have when implementing SAFe. The analysis also gives companies some best practices on how to change their test process and how to work efficiently with test and quality in SAFe. A final overall analysis was made and it resulted in some guidelines for companies that are going to implement SAFe.

Keywords: SAFe, Scaling Agile, Agile process, Waterfall process, Test, Quality, Challenges, Changes

Acknowledgements

I would like to thank my supervisor Christin Lindholm for all her support and guidance during this thesis. Also, I would like to thank my examiner Martin Höst for great feedback on my report. Huge thanks to all involved people at Company X for making this thesis work possible and also for all support and help. Another huge thanks to all the companies that have participated in the interviews.

Contents

Chapter 1

Introduction and background

This chapter provides background information, purpose of this thesis work along with related work and boundaries in this thesis.

1.1 Background

Today many companies implement SAFe [13], Scaled Agile Framework in their organization to increase in productivity and quality, and provides faster time to market. Scaling agile development is a big challenge for many companies so they choose to implement SAFe since it helps companies scale agile development in a organized way. Companies that are going to implement SAFe do not know which challenges they could encounter during the implementation and how they should address these challenges. It can therefore be difficult for companies to estimate the time it takes and the amount of work that is needed to implement SAFe. This thesis work has been conducted on the test unit at a company. On behalf of the company's request, the company name have been replaced with Company X due to anonymity. Company X is now facing the implementation of SAFe and would like to know which challenges and problems that could possibly occur within the test area. With a better knowledge Company X can be more prepared and possibly be able to prevent some of the challenges or have solutions in place before challenges occur.

1.2 SAFe

SAFe, Scaled Agile Framework is an agile software development framework that is provided by the company Scaled Agile. The framework consists of proven success patterns for implementing Lean-Agile software and systems development at enterprise scale. SAFe provides guidance on how to work at four different enterprise levels: Portfolio, Value Stream, Program and Team [14]. More information about SAFe can be found in Section 2.4.

1.3 Company X

Company X is a worldwide furniture store that was founded in the 1940s and has today over 155 000 employees. Company X consists of many different departments where IT is one department. Company X has more than 700 different IT-systems to support their business as well as internal systems for the employees, which are handled by the IT department. There are around 1000 employees and a couple of thousands consultant in the IT department and one part of the IT department is located in Helsingborg. Today Company X is working in a Waterfall based development and some parts work in a agile development. Company X is using ITIL [1], Information Technology Infrastructure Library, for the service management and using PPS [2], Practical Project Steering, for the project management. ITIL is a framework that consists of a set of practices for IT service management and PPS is a steering model that is used for managing projects. Company X have many systems that work together so there are many dependencies between all the systems. All these systems are developed by different teams and with the current working process the deliveries take a very long time. The reason why Company X wants to implement SAFe is so that they can faster deliver new and updated systems to the users. The goal with the implementation is faster delivery to a lower cost in a very structured way.

1.4 Problem definition

To be able to help Company X to be more prepared for their implementation it would be interesting to know what challenges and problems that other companies have encountered when they have implemented SAFe. On behalf of the interviewed companies' requests, the names of the participating companies have not been documented with regard to anonymity. It would also be interesting to identify the major and most common challenges in the test and quality area that Company X could face when implementing SAFe. It is also important to discover and evaluate the solutions that other Companies have made when they have encountered these challenges. There will be changes in the test area when doing the implementation and it would be interesting to find out which the largest changes are, for example roles in the teams, and how to follow up on test and quality efficiently. Since many companies probably have tried and evaluated different test setups on their way to the best testing approach in SAFe, it would be interesting to identify if there are any "best practices" available for how to work effectively with test in SAFe. A final goal is to produce some guidelines that can help companies that are facing the implementation of SAFe.

1.5 Related work

This section contains information about previous work that is related to this thesis. The author of this thesis could not find any previous research about test in SAFe. There are some researches that discuss similar topics and they are described below.

Gandomani, et al. [3] have done a study that highlights the main challenges that a company might have when transforming from a traditional working method to an agile method. A summary of the challenges are described in Section 2.3. Some of the challenges in this article will be used as an inspiration for the questions that will be held in the interviews. There is also another study that is written by Boehm and Turner [4] which discusses the management challenges when implementing agile processes in traditional development organizations.

Talby, et al. [5] have made a number of observations that are highly related to this thesis. This article is based on a study in a large scale project at the company Israeli Air Force (IAF). The article brings up the challenges that the company had with agile testing in the project and also how they managed these challenges.

There is one article about challenges with moving to SAFe that is written by Crain [15]. This article brings up the four biggest challenges of transforming to SAFe, information around the topic of the challenges and tips on how to manage these challenges. This article contains Crain's own experiences and thoughts about transforming to SAFe and it is only published on a website. The trust of the article can be questioned since there is no proof that anyone have reviewed it and that no official research have been done. However Crain has worked for IBM since 1999 where he has lead organizational changes so he is very experienced in agile.

O´Donnell and Richardson [6] have done a similar study on a small Irish company. The purpose of their study was to examine the problems that the company encountered and which benefits they derived by implementing the agile method XP, eXtreme Programming.

1.6 Boundaries

This section describes the boundaries in this thesis work. The boundaries are listed below.

- SAFe describes a lot of different areas but this thesis have focused on the test and quality aspects of SAFe. Other parts of SAFe can be mentioned but are not discussed in detail.

- This thesis work has concentrated on interviewing companies that have implemented SAFe successfully or is in the beginning of the implementation phase. It could also have been interesting to interview companies that have tried implementing SAFe but chosen to stop the implementation because of varies reasons. These companies were not considered in this thesis since these companies are harder to find and they might not be able to answer on all the questions that this thesis is discussing.
- Only companies in Sweden and companies that are located near Sweden, such as Denmark and Finland, have been interviewed in this thesis. The reasons are time difference and cultural differences. It facilitates if the interviewer and the interviewee are close in time zones when doing the interview. Different countries have different cultures and different companies have different views on how a workplace should work. Companies that are located near Company X might have more in common with Company X in these areas so then the result will be more helpful for them.

1.7 Outline of report

The report is divided into 5 chapters which are described below.
Chapter 2 Theory: Contains information about Agile and Waterfall development, SAFe and Company X's test process.
Chapter 3 Methodology: Contains information about the approach of the thesis work, the different interview techniques that have been used and source criticism.
Chapter 4 Result: Presents the results from the interviewed companies.
Chapter 5 Discussion and conclusion: Discusses the results of this thesis and a conclusion is drawn.

Chapter 2

Background

This chapter gives the reader a basic knowledge about the Waterfall development process and the Agile development process. It also provides information about SAFe in general, implementation of SAFe and how testing is done in SAFe. Company X's current and future test processes are also described in this chapter.

2.1 Traditional waterfall development

The traditional waterfall development approach is a sequential approach divided into different phases where each phase is finished before the next one starts. The different phases could be: requirements, design, implementation, test and maintenance [7]. The waterfall development approach is described since the agile development approach is based on it. The different steps in a waterfall development are also executed in the agile development approach.

2.1.1 Requirements

The initial step when starting a new project is the requirement phase. The requirement phase can be divided into three parts: requirements gathering, analysis and documentation. First the team identifies all stakeholders and interviews them and collect all their requirements. Stakeholders are persons that are interested in the project or its outcome, for example the customer. Then the team analyses all the requirements and discuss the requirements with the stakeholders. When the requirements are agreed upon, they are documented and baselined. The analysis and documentation of requirements is ideally not a linear process, since the team should meet the stakeholders several times during the requirement phase to ensure that they have understood the requirements correctly and if any requirements are missing. It is very important that the requirements are agreed on before the team starts on the design phase. Otherwise the team has to go back to the requirement phase every time there is a new or changing requirement. This may invalidate work that was already done and/or cause the project to be delayed [7].

2.1.2 Design

The next step is the design phase, which means the team creates a detailed design for the complete system and also for each of the individual components. The design is made on such a level that the developers can directly translate it into code in the next phase. There are different methods to do the design of the system such as use cases, UML and flowcharts. After the design phase the project moves to the implementation phase. The design has to be finished, reviewed and signed off by the team and stakeholders before the design can be delivered to the developers [7].

2.1.3 Implementation

In this step the design documents are translated into code by the developers. When a component is developed it is unit-tested on its own and the code is reviewed. The integration of the different components is usually at the end of the implementation phase or in the beginning of the test phase. That is the time when things may not fit or work together as planned which means that the team may have to go back to the design phase and make the necessary changes. If that is not included in the project schedule the project will be delayed. It is possible to reduce the risk of an unexpected event by introducing different milestones during the coding phase. When a milestone occurs the system is integrated and needs to provide a certain level of functionality. When the integration is done it is ideal to test that the functionality works correctly [7].

2.1.4 Test

The test phase starts when the developers have declared that they are done with the implementation phase. As mentioned before the different components can be integrated to a system in the beginning of the test phase. It is usual that the team has an integration phase that is owned by the development team, which have to prove that a number of test cases can be executed without bugs before the test team receives the code. To avoid that a large number of testers fail at the same tasks it is helpful if a smaller number of testers start with some basic test cases before the majority of the test team start testing. The purpose of the test phase is to identify bugs in the system before it is released to the user. The right time to hand over the system to the customer or to release it is usually when the defect arrival curve of a project flattens out [7].

2.1.5 Maintenance

This phase starts direct after the solution has been delivered to the customer or made available as a purchasable product. The responsibility of maintenance and support of the system is often handed over to a maintenance team. The maintenance team will help the customer to install the system but also support users of the system when they have problems using the system. Developers will resolve the bugs in the system and release a new version to the customer [7].

2.1.6 Advantages and disadvantages

There are several advantages with the waterfall approach. It is the most efficient way to carry out a project if everything is designed in the beginning, implemented and then works as expected without any integration problems or bugs. Changes should be done as early as possible since it gets more and more expensive the later they are done. When executing the waterfall model thoroughly, a good set of documentation is produced during the requirements and design phases. The documentation is used when the test team verifies that the product works as specified.

It is very difficult to carry out a larger project or a project with some level of innovation without any change. It can be for example that the customer comes up with new requirements when they have seen a prototype. This means that the team needs to go back to the requirements and design phases to add the new requirements and then update the entire code [7].

2.2 Agile development

The agile development approach is an iterative sequential approach where the product is optimized and developed in several short iterations. Iterations help to structure a comprehensive project into smaller manageable units. When working with short iterations it is easier to adapt to new incoming requirements. Each iteration should be maximum four weeks long and will go through a full development cycle including planning, design, coding and test [8]. When working in iterations the team gets forced to stay focused on the present work and not think about what to work on in the future. The rough project outline will guide the teams into the future.

The stakeholders should be involved since this forces the team to continuously ensure that they meet the customers' expectations and deliver value. Continuous feedback from the stakeholders will lead to changes earlier in the process and that the project holds the time frame. The project's progress is measured in the amount of working code, which is forcing the developers to focus on quality. Deliver working code that have been tested directly when it was developed, will provide an early verification if a use case is done or not. Integrate and deliver tested code continuously will avoid issues to accumulate [9].

The teams in an agile development should have the following characteristics:

- **Self-similar:** All the teams are alike and operate within the same corporate culture and vision. The purpose with each team is to create customer value and implement a corporate strategy. Each team organize itself so the teams can be organized entirely differently and apply a different approach to pursue its mission.
- **Goal-oriented:** Each team derives its scope of action from the project's high-level goals and corporate strategy. The team translates the goals into more precise goals so it is important to have clear high-level goals and to make sure that all teams are headed in the same direction.
- **Self-organization:** The teams organize their work in a way that suits them best to accomplish their mission. This can include technical aspects, organizational aspects or social aspects.
- **Self-improvement:** Each team continuously improves and adapts its products and processes based on feedback and input from the customer.
- **Vitality:** The team is making progress in iterations with small increments to the product. The team should focus its planning on the present work to maintain vitality. The team's efficiency is boost because of the extensive iterations, effective collaborations, flexible planning and a lean organization. It also helps to find a creative solution when the unforeseeable happens.

There are several different agile methods that can be used when developing a system. Lean Software Development is one of these methods and it is a concept which evolved from the ideas of lean manufacturing in automotive industry. Concepts that are included are thoughts on eliminating waste, adding customer value and empowering workers. Another method is Scrum which is an extremely efficient and streamlined process of managing and tracking teams. A common method is Test Driven Development which emphasizes the definition of test cases even prior to the actual implementation of the code. A last method is Extreme Programming which is a toolbox of engineering practices like pair programming, or continuous integration [8]. Kanban and ScrumXP are the two methods that are used in SAFe. The major differences between these two methods are:

- **Roles and responsibility:** In Kanban there are no predefined roles in a team because then all team members are able to do any task and help any person that has to much to do. In ScrumXP each team member has a predefined role. The scrum master dictates timelines, product owner defines goals and the team members do the work.
- **Delivery time dates:** In Kanban the products and services are delivered continuously when they are needed. In ScrumXP the deliveries are determined by sprints which are time boxes where a set of work have to be done and be ready for review.
- **Delegation of tasks:** In Kanban the team members pull new tasks once their previous task is completed. In ScrumXP the team pulls a batch of tasked for each iteration.

- **Modifications:** In Kanban, modifications are allowed to a project midstream which makes it possible to prioritize iterations and continuous improvement before completion of a project. In ScrumXP it is not allowed to make changes during a sprint.
- **Productivity:** In Kanban the productivity is measured in the amount of time it takes to complete a project. In ScrumXP productivity is measured by the velocity through sprints. Each sprint relies on the success of the sprint before.

More information about ScrumXP and Kanban can be found in [16] and [17].

2.2.1 Advantages and disadvantages

The team works close to the stakeholders which makes it possible to get feedback and new requirements fast. The agile approach takes care of changes from the stakeholders early in the process since the team gets feedback in each iteration. When working in iterations the team will gain simplicity and will make fact-based decisions rather than wild guesses. The teams focus a lot on quality in each iteration which makes the system quality higher and fewer bugs. The organization is not centralized top-down instead the teams are invited to act as entrepreneurs when taking decisions and this increases the speed of decisions. The teams provide knowledge, skills and talent that the stakeholders might not have. Successful organizations are built from independent, self-organized units that collaborate towards a common set of goals [8][9].

If the customer is not clear with what they want, the development can rapidly get off track which can delay the whole project. Requirements are not as well documented as in waterfall and the design is lacking or confusing since they do design in each iteration. It can be difficult to estimate in the beginning of the project how long time it will take to develop the system. This is because of all possible changes and new requirements that can come up during the projects time [18].

2.3 General difficulties with transforming from waterfall to agile

There are some general difficulties with transforming from a waterfall process to an agile process. According to [3] there are four main areas where companies have difficulties when doing the transformation. The areas are: Organization and management related challenges, person related challenges, process related challenges, and technology and tools related challenges. The challenges with these areas are summarized below:

- **Organization and management related challenges:** Changing the mind-set of persons and their organizational culture is not an easy process. When doing this transformation the management style is changed from “command and control” to “leadership and collaboration”. Another issue is group decision, especially when it comes to resources, alignment of strategic product line and developing and maintain tasks in teams. A project manager's responsibility will change from planner and controller to director and coordinator. Many project managers will have trouble with this change so it will take time to change their mind-sets and they will need good mentoring. One big challenge is documentation since in waterfall, documentation has a very big role when it comes to knowledge management. In agile, the documentation is limited and the team members have a lot of information in their heads. When a company have a distributed development organization, communication is another issue. It gets difficult to have face to face meetings because of the distance.

- **Person related challenges:** To success with agile the team members must value and trust each other. It is common in agile development that there are some individual centered activities for example pair programming. These activities can be a challenge for persons that have worked a long time with waterfall so they would need training, mentoring and a set of work practices. Another challenge is the relation to customers. In agile one of the team members represents the customer and all decisions are based on different attitudes, goals and experience. This can be difficult for some traditional project managers since they are not in charge. In this area, training, coaching and mentoring is very important.
- **Process related challenges:** Changing from a waterfall process to an agile process takes a lot of time and effort, and needs a lot of investment. The waterfall process is based on standard activities and measurements while the agile process is based on uncertain activities which supports rapid development and high quality production. There is no appropriate measurement practice in agile development and this can be a problem for traditional developers. When changing the process, it influence on strategies, tools, roles and techniques. The biggest challenge for the company is to choose which agile method to use. However all of them are based on the same agile values.
- **Technology and tools related challenges:** These challenges are not as big as the others but it can be useful for a company to know about them. The companies should use tools that support incremental evolution, continuous integration, re-working, version management and other agile technologies. Training and investment will help the teams to use these tools [3].

2.4 SAFe

SAFe is an agile software development framework that is scalable and modular. These qualities make it possible to apply SAFe to any size of a system and it allows an organization to apply SAFe in a way that suits their needs. The framework can be applied to both software and system development. It can be applied to a system that needs fewer than 100 persons to make a solution and to a system that needs up to thousands of persons to create and maintain the system [14]. There are four core value that SAFe honours, support and helps deliver: Alignment, Built-in quality, Transparency and Program execution. This report focus on test and quality so this chapter also contains information about built-in quality in SAFe. Information about the other core values can be found in [19]. This section also contains information about mixing agile and waterfall development in SAFe since it might not be possible for all systems to be developed in an agile approach.

2.4.1 Principles

SAFe is based on nine fundamental principles that have evolved from agile principles and methods, Lean product development, system thinking, and observations of successful enterprises. The principles are described in the below sections [20].

Take an economic view

The Lean system builder's goal is sustainably shortest lead time, with best quality and value to persons and society. To achieve this goal it requires a fundamental understanding of the economics of the system builder's mission. If there is a lack of understanding, even a technically competent system may cost too much to develop, take too long time to deliver or have manufacturing or operating costs that cannot support economically efficient value. It is very common that the economic constraints on the system builder's activities are known to only few. When having a centralized knowledge, the workers' everyday decisions are either made without such an understanding, which directly sub optimizes economic outcomes, or escalated to those who have it, which increase delays in value delivery.

SAFe highlights the importance of economics in a successful solution development. If the solution does not meet the customer's or system builder's economic goals then the long-term viability of the solution is suspect. It is very common that the reason for a failed solution is because of failed economics. SAFe have two aspects for achieving optimum economic outcomes and they are: deliver early and often, and understand the economic trade-off parameters for each program and value stream.

Deliver early and often: When choosing a Lean-Agile path they are choosing a model that is based on incremental development and early continuous value delivery. This alone produces a significant economic benefit since value is delivered to the customer much earlier in the process. The value integrates over time since the longer the customer has it, the more value they receive. In the waterfall model the value begins in the end of the planned development cycle. This is illustrated in Figure 2.1. Another aspect is that when the quality is high enough, the things that are delivered early to market are typically more valuable. If it is early enough, it is not available from anyone else and therefor it is worth a lot to the economic buyer.

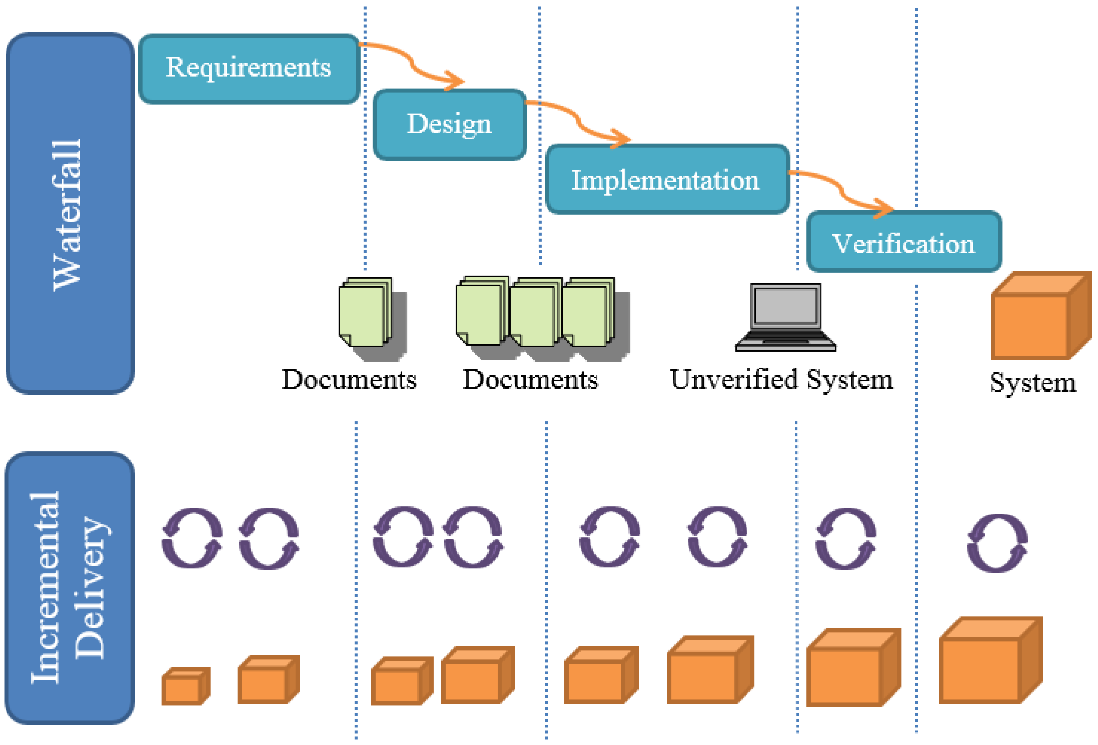


Figure 2.1: Increment delivery vs. Waterfall [21]

Understand the economic trade-off parameters for each program and value stream: It is also important to look at various additional economic trade-offs. There are five primary parameters that can be used to consider the economic perspective on a particular investment. It is important to understand these parameters and the trade-off between them because then it is possible to optimize the life cycle. This is the key to unlocking optimum economic value in development. The parameters are described below and the trade-offs are shown in Figure 2.2.

- **Development Expenses:** is the cost of manpower and materials required to implement a capability.
- **Cycle Time (lead time):** is the time it takes to implement the capability.
- **Product Cost:** is the manufacturing cost (the cost of the goods that are sold), and/or deployment and operational costs.
- **Value:** is the economic worth of the capability to the business and customer.
- **Risk:** is the uncertainty of the technical or business viability of the solution [21].

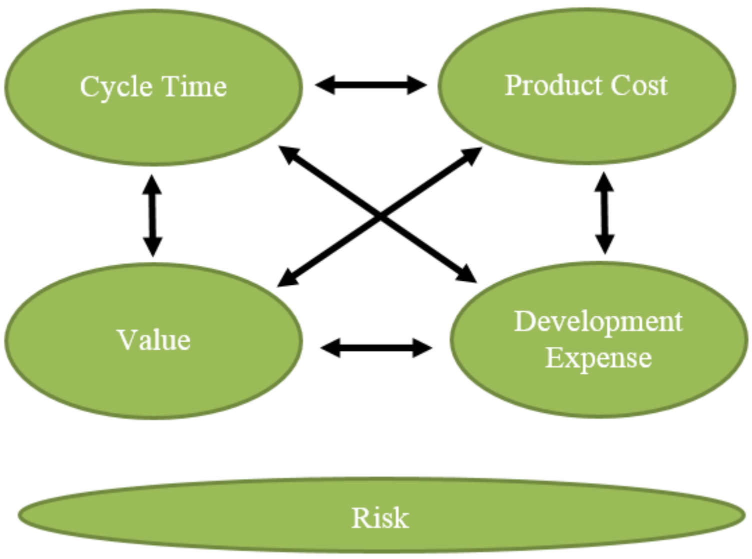


Figure 2.2: Trade-off between the five parameters [21]

Apply system thinking

The three foundational bodies of knowledge in SAFe are System Thinking, Agile development and Lean Product development. System thinking takes a comprehensive approach to solution development. The approach incorporates all aspects of a system and its environment into the design, development, deployment, and maintenance of the system. There are three primary aspects of system thinking and they are illustrated in Figure 2.3. These aspects help leaders and teams navigate the complexity of solution development, the organization and the larger picture of total time to market.

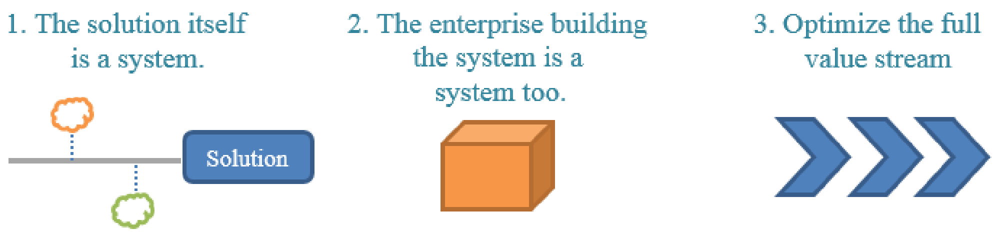

1. The solution itself is a system.
Solution
2. The enterprise building the system is a system too.
3. Optimize the full value stream

Figure 2.3: The three aspects of System thinking [22]

The solution itself is a system: Guidance for development and deployment of complex software and cyber-physical systems is provided by SAFe. These systems are represented by the SAFe Solution object, which is the subject of each value stream and is the physical object that delivers end user value. Some examples of a physical object are: application, satellite, medical device or web site. These tangible systems lead to a number of critical insights:

- The system builders must clearly understand the boundaries of the system, what it is, and how it interacts with the environment and systems around it.

- Optimizing a component does not necessarily optimize the system. It is important that the components do not become selfish and overloads the resources they need such as computing power, memory, electrical power, whatever the other elements need.
- It is important to understand the intended system behaviour and have a high level understanding of the systems architecture if the system shall behave well. The higher level understanding means understanding how the components work together to accomplish the aim of the system. The intentional design of a system is fundamental to systems thinking.
- The interfaces that a value passes through in a system and the dependencies that they create are critical elements of providing ultimate value. It is vital that those interfaces and interactions get continuous attention.
- A system cannot evolve faster than its slowest integration point. The faster the whole system can be integrated and evaluated, the faster the actual knowledge of the system grows.

The Enterprise Building the System is a System too: The second aspect of system thinking is that the persons, management and processes of the organization that builds the system are also a system. That system should also be managed otherwise the components of the organization that is building the physical system will locally optimize and become selfish. This will limit the rate and quality of overall value delivery. This also leads to some organizational systems thinking insights:

- It is a social endeavour to build complex systems so leaders must facilitate the creation of an environment. It is important that the environment makes it possible for persons to collaborate on the best way to build better systems.
- Suppliers and customers must be treated as partners since they are integral to the value stream. They should only be treated as partners after a long-term foundation of trust.
- Locally optimizing teams or functional departments does not optimize the flow of value though the enterprise.
- Value crosses organizational boundaries. Accelerating value delivery requires the elimination of functional individual business functions or the creation of virtual cross-functional organizations, such as an Agile Release Train.

Optimize the full value stream: Value streams are fundamental constructs in SAFe. The entire SAFe portfolio is constructed to be a collection of value streams where each delivers one or more solutions to the market. Each value stream consists of the sequence of steps necessary to get a new concept integrated and deployed via a new or existing system. This aspect is the only way to reduce the total time it takes to go from concept to cash. System thinking requires that leaders and practitioners understand and continuously optimize the full value stream. It is especially important when it crosses technical and organizational boundaries.

Value stream mapping is a primary tool which is a systematic way to look at all the steps required to produce value. If value stream mapping is used then leaders quickly recognize that the actual value added processing steps which consume only a small portion of the total time to market. Some of the processing steps that are added are: creation of the code and components, deployment, validation. This recognition drives these leaders to continuously focus on the delays between steps [22].

Assume variability; preserve options

System builders tend to have a natural tendency to try to reduce variability. When a lot of decisions have been made it is common to think that you are further along in the process then you really are, but that is not always the case. Variability is not inherently bad or good. It depends on the economics associated with the timing and type of variability that determines the outcomes. In the beginning of a project the system builders recognize that very little is actually known except a general understanding of the system intent. It is common that traditional design practices tend to drive developers to quickly converge on a single option and then modify that design until it eventually meets the system intent. This can be an effective approach if one does not pick the wrong starting point. If the wrong starting point is picked then the subsequent iterations to refine that solution can be very time consuming and lead to a suboptimal design. It is a higher risk that your starting point was not the optimal one if it is a bigger and more technically innovated system.

A better approach is to use a Set-Based Design which is illustrated in Figure 2.4. The system builder initially casts a wide net by considering multiple design choices at the beginning. Then they continuously evaluate economic and technical trade-offs and then they eliminate the weaker options and finally decide on a final design that is based on knowledge. This approach leaves the design options open as long as possible which produces more optimal technical and economic outcomes [23].

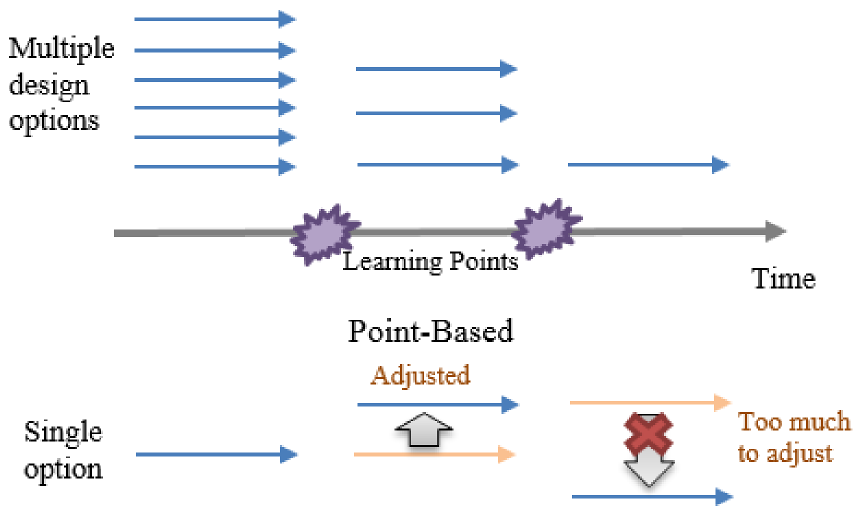


Figure 2.4: Set-Based Design [23]

Build incrementally with fast, integrated learning cycles

In a traditional development the investments cost begins immediately and accumulates until the solution is delivered. It is common that little or no actual value is delivered until all of the committed features are available or that the program runs out of time or money. The process is not designed so that customer gives feedback during the development. Lean system builders approach the problem differently by working within a range of requirements and design options, and build the solution incrementally in a series of short time-boxes. Each time-box results in an increment of a working system that can be tested and evaluated by the system builders and customer. The time-boxes build upon the previous increments and the solution evolves until it is released. The integration points can serve as prototypes for testing the market, establishing usability and gaining objective customer feedback. If needed, the feedback points can allow the system builders to change to an alternative course of action such as one that should better serve the needs of the intended customers.

The primary focus of the system builders are the synchronized integration points. The focus comes from a development process and a solution architecture that is designed in part for that specific purpose. A "pull event" is created by each integration point where the event pulls the various solution elements into an integrated whole. The integration points also pull the stakeholders together because of a routine synchronization that helps assure that the evolving solution addresses the real and current business needs. Each integration point delivers a value that is converting uncertainty into knowledge. The knowledge is about the technical viability of the current design choice and the potential viability of the solution which is based on objective measurements.

Integration points serve as primary mechanism for controlling the variability of solution development since they are an instantiation of basic Plan-Do-Check-Adjust cycle. An example of integration points are displayed in Figure 2.5. The more frequent the point is, the faster the learning is. In development of complex systems, local integration points are used to assure that each element or capability for the system is meeting its responsibilities in contributing to the overall solution intent. Then these local points must be further integrated at the next higher system level and the larger the system, the more integration levels exist. The top level, least-frequent integration point provides for the only true measure of system progress. The system builders understand this and they strive to achieve those points as frequently as possible. All stakeholders understand that the project is in trouble when timing of integration points slip. However this knowledge helps the system builders to realize which adjustments that are necessary to do so that the project is back on track [24].

Base milestones on objective evaluation of working systems

The development of large systems requires investment where system builders and customers have the responsibility to ensure that the investment will deliver the necessary economic benefit. It not, there is no reason to make the investment. It is important that stakeholders collaborate to help ensure the prospective economic benefit throughout the development process. This is possible since the industry has applied a sequential phase-

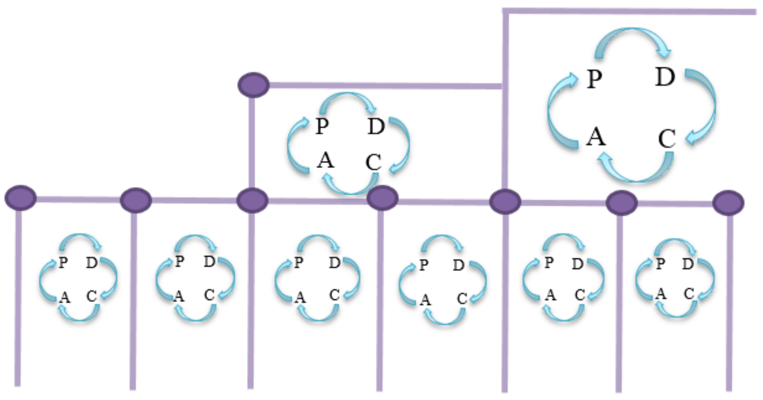


Figure 2.5: Integration points [24]

gated (waterfall) development process. The progress is measured via a series of specific milestones. These milestones are following a logical and sequential process of discovery, requirements, design, development, test and delivery. Sometime this does not work for everybody and the root cause of this problem is the failure to recognize for critical errors within the basic assumption that phase gates reveal real progress and thereby mitigate risk. The four critical errors are:

- Centralizing requirements and design decisions in functions that may not be integrally involved in the solution building.
- Forcing too early design decisions and false positive feasibility. The best choice is made early in the process and the development proceeds under the assumptions that everything is on track. Later on cones the discovery that the path chosen is not actually feasible.
- Assuming that a point solution exists and can be built right the first time. Variability exists in a project and to assume that it will not, will just delay the project.
- Taking up-front decisions creates large batches of requirements, code, tests and long queues. This leads to large batch handoffs and delayed feedback.

The phase gate model does not mitigate risk as intended and a different approach is needed. The system is built in increments where each increment is an integration point that demonstrates some evidence as to the viability of the current in-process solution. A milestone involves a portion of each step: requirements, design, development and testing. An example on milestones is shown in Figure 2.6. Together they produce an increment of value. This is done routinely and provides the discipline needed to ensure periodic availability and evaluation. It also provides predetermined time boundaries that can be used to collapse the field of less desirable options. At these critical integration points different measurements are done and the measurements are subjects to the nature and type of the system that is built. However the system can be measured and evaluated by the stakeholders frequently and throughout the development cycle [25].

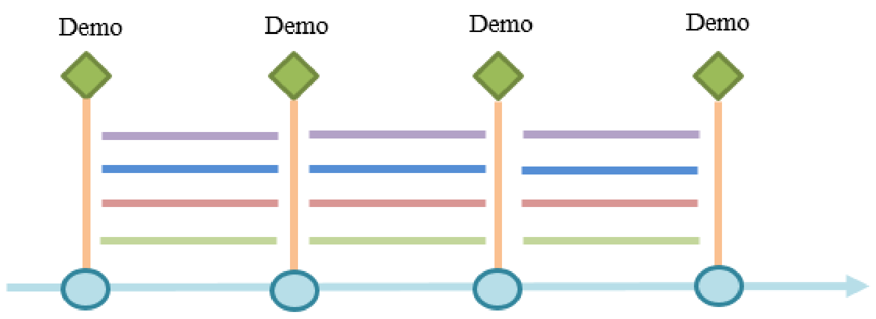


Figure 2.6: Example of milestones [25]

Visualize and limit WIP, reduce batch sizes and manage queue lengths

Lean system builders strive to achieve continuous flow so that they can sustain the shortest lead time. To achieve continuous flow it requires elimination of the traditional development process. The three primary keys for implementing flow are: visualize and limit Work-In-Process (WIP), reduce batch sizes of work items and manage queue lengths.

It is very common to overload teams and programs with more work than they can accomplish. Too much WIP in the system causes multiplexing and frequent context switching. The consequences are that the persons doing the work is overloaded, it reduces focus on any task at hand, reduces productivity and throughput and increases wait times for new functionality. First step is to make the current WIP visible to all stakeholders since it illustrates the total amount of work at each step. It also serves as an initial process diagnostic that shows the current bottlenecks. The WIP can be a wake-up call for the developers so that they start to address the problems. The next step is to start balancing the amount of work in process against the available development capacity. If the WIP limit is reached, no more work should be added. To be able to successfully limit WIP you should have knowledge, discipline and commitment.

Another way to reduce WIP is to decrease the batch sizes of the work such as requirements, designs, code, tests and other work items that move through the system. Small batches go through the system faster and with less variability. The variability increases because of the accumulated variability of the items in the batch. The economically optimal batch size depends on both the holding cost and the transaction cost. The holding cost is the cost for delaying feedback and value. The translation cost is the cost of implementing and testing the batch. To improve the economics, a primary focus is to reduce the transaction cost for each batch.

The last element to achieve flow is to manage and generally reduce queue lengths. Little's law says that the wait time for service from a system is the ratio of the length of the queue divided by the average processing rate. The longer the queue of work that is awaiting implementation by the team, the longer the wait time, no matter how efficient the team is processing the work. The solution to get faster service is to either reduce the length of the queue or increase the processing rate. The fastest way to increase the processing rate is to reduce wait times which is done by reducing the length of the queue. This is accomplished by keeping backlogs short, largely uncommitted and visualized. Reducing queue lengths decreases delays, reduce waste and increases predictability of outcomes. Doing all of these can cause fast and measurable improvements in customer satisfaction and employee engagement [26].

Apply cadence, synchronize with cross-domain planning

Solution development is an uncertain process because if it were not, then the solution would already exist and there would be no place for the next generation of innovations. The uncertainty conflicts with the business need to manage investment, track progress and have sufficient certainty of future outcomes so that they can be able to plan and commit to a course of action. The puzzle with solution development is divided in two: the conflict of uncertainty of outcomes vs the needed certainty of the business. The system builders work in the safe zone where sufficient uncertainty provides the freedom for innovation, while sufficient certainty allows the business to operate. To achieve this, it is important to maintain true knowledge of the current state. This is provided by cadence, synchronization and cross-domain planning.

Cadence is the dependable heartbeat of the process that provides a rhythmic pattern. Cadence transforms unpredictable events into predictable ones, makes waiting time predictable, facilitates planning and provides for more efficient use of resources. It also provides a forcing function and lowers the transaction costs of key events which include planning, integration, demonstrations, feedback and retrospectives.

Synchronization causes multiple perspectives to be understood, resolved and integrated at the same time. Synchronization is used to pull the disparate assets of a system together to assess solution level viability, align the development team and business to a common mission and integrate the customers into the development process. Cadence and synchronization together helps the system builders operate reliably within the uncertainty safe zone.

The most critical event is when periodically all stakeholders gather for cross-domain planning and synchronization. This is called the Release PI and it serves as the key point around which all other events operate. The event serves three primary purposes: Assessment of the current state of the solution, Realign all stakeholders to a common technical and business vision, and plan and commit to the next program increment. Cadence, synchronization and periodic cross-domain planning provides system builders with the tools they need to operate in the safe zone [27].

Unlock the intrinsic motivation of knowledge workers

The Lean-Agile principles of SAFe are a system too and the elements of this system collaborate to create a new and empowering paradigm. It is a paradigm where the knowledge worker is able to communicate across functional boundaries and make decisions based upon an understanding of the economics. The knowledge worker are also able to achieve fast feedback as to the efficacy of their solution. It is important that the knowledge workers participate in continuous incremental learning and mastery. The knowledge workers should also participate in a more productive and fulfilling solution development process. This is one of the most powerful motivations of all.

There are some factors that are important when motivating the knowledge workers. Some of the factors are listed below:

- If the knowledge workers are not paid enough, they will not be motivated. Under-compensation is a major de-motivator.
- After a point, money is no longer a motivator. Then it comes to the point of intellectual freedom and self-actualization. At this point the knowledge workers minds are free to focus on the work and not the money.
- After this point you should not add any compensation elements, it is a de-motivator. It can be seen as an insult to the intellectual integrity or cause the worker to focus on the money, rather than the work.

An important leadership responsibility is to provide for autonomy, while harnessing it to the large aim of the enterprise. The motivation of self-direction must be within the context of the larger objective. Leaders must provide some larger purpose to the workers so that they get a connection between the aim of the enterprise and the workers daily work activities. Being part of a high-performing team is another critical motivational dimension and leaders can inspire teams to do their best by providing:

- The mission, a general and strategic direction.
- Little, minimal or even no specific work or project plans.
- Challenging requirements with the minimum possible constraints as to how teams meet these requirements.

It is important that the workers are heard and respected by their leaders. The workers have so much more knowledge than their manager so they should be connected to the decisions and should be able to give feedback. Leaders should give tough feedback supportively and encourage others to:

- Disagree where appropriate.
- Advocate for the position they believe in.
- Make their needs clear and push to achieve them.
- Enter into joint problem solving with management and peers.
- Negotiate, compromise, agree and commit [28].

Decentralize decision-making

To be able to deliver value in the shortest sustainable lead time, it requires decentralized decision-making. When a decision escalates to higher level of authority it leads to a delay in delivery. However escalated decisions can decrease the decision fidelity because the lack of local context and the changes to fact patterns that appears during the waiting period. Decentralizing decision-making is very good since it reduces delays and improves the flow and throughput of the product development. It also enables faster feedback, more innovative solutions and higher levels of empowerment.

Some decisions should not be decentralized since they are strategic, have far reaching impact and are largely outside of the scope of the teams. The leaders are responsible for the outcomes since they have market knowledge, longer-range perspectives and understand the business and financial landscape that is necessary to steer the enterprise. So some decisions should be centralized and they have the following characteristics:

- Infrequent: decisions that are not made very often and are not so urgent. It is appropriate to have a deeper consideration for these decisions. One example is product strategy.
- Long lasting: decisions that are unlikely to change after they have been made, for example commitment to a standard technology platform.
- Provide significant economies of scale: decisions that provide large and broad economic benefits. Two examples are standard tooling and common way of working.

The majority of the decisions should be decentralized since they do not reach up to strategic importance. These decisions have the following characteristics:

- Frequent and common decisions, for example the prioritization of the team and program backlog.
- Time critical decisions that cause a high cost of delay in the project if it is delayed. Two examples are point releases and dependencies with other teams.
- Decisions that require specific local information and context. The context could be technology, organization and customer or market impact. One example is resolving a significant design problem.

The workers should make these decisions since they have better local context and they have detailed knowledge of the technical complexities of the current solution [29].

2.4.2 Levels

A company can be divided into four levels and the framework provides guidance for work at each level. The levels are Portfolio, Value Stream, Program and Team [13]. The different levels are shown below in Figure 2.7 and permission to include figure is granted by Scaled Agile Inc. [31].

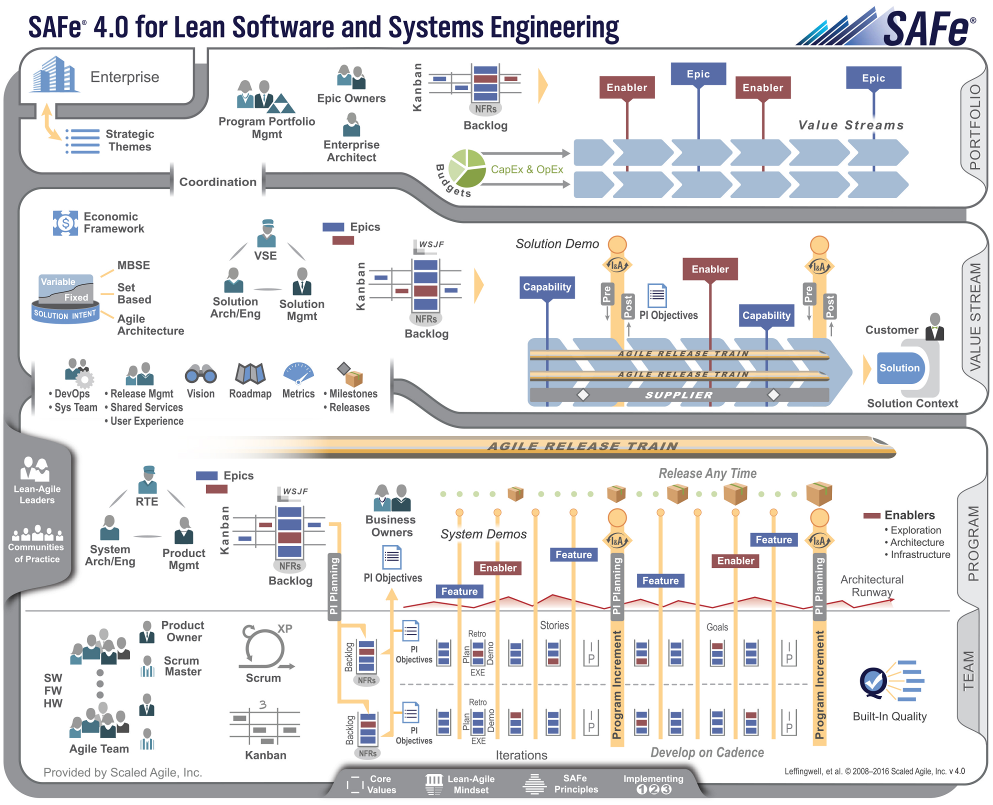


Figure 2.7: The different levels in SAFe [30]

Portfolio level

The Portfolio level is the level with the highest concern in SAFe and it provides the basic design of organizing the Lean-Agile enterprise around the flow of value via value streams. Value streams provide funding for the persons and other resources that are necessary to develop the systems and solutions that meet the strategic purpose. The portfolio level encapsulates all the elements from the value streams and also provides the basic budgeting and other governance mechanisms that are necessary to assure that the investment in the value streams provides and meets its strategic objectives.

The portfolio level is connected to the business via a bidirectional way. One direction provides the strategic themes which connect the portfolio to the evolving enterprise business strategy with specific, itemized business objectives. Strategic themes provide business context for decision-making within the portfolio which affects investments in value streams and serving as input to the portfolio, solution and program backlogs. The strategic themes are created by the business with the key portfolio stakeholders who thereby are among the cornerstones of strategy formation. The other direction indicates a constant feedback of portfolio context back to the enterprise. This updates the enterprise of the current state of the solution set, key performance indicators, along with any strengths, weakness, opportunities and threats that are present at the portfolio level [32].

PPM, Program Portfolio Management is responsible for participating in the establishment and communication of the strategic themes that guide the enterprises' investments and strategies, determine the relevant value streams and allocate budget to them. They are also responsible for defining and prioritizing the portfolio backlog epics, reporting the business on investment and progress, and other aspects of portfolio context [33].

Another of the portfolio level's responsibility is the discovery, definition and administration of cutting initiatives that are required by the business and these are called epics. Business epics capture and reflect the new business capabilities that can only be provided by cooperation among value streams. Epic enablers reflect the architectural and other technical initiatives that are necessary to enable the new capabilities.

SAFe provides a portfolio Kanban system to help manage the flow of work and make sure it is visible to all stakeholders. The Kanban system helps to assure that demand is metered to the actual value stream capacity since it provides WIP limits and it makes the work more visible. The portfolio backlog is the highest-level backlog in the framework and is the place for epics that make it through the Kanban system and await implementation. The backlog only contains the approved and prioritized epics so that the portfolio can achieve market differentiation, operational efficiencies, or better-integrated solutions. Epic owners are responsible for managing the epics until implementation is assured.

The enterprise architect helps provide strategic technical direction that can optimize portfolio outcomes. This is possible since the enterprise architect has the knowledge to work across value streams and programs. Some examples of strategic technical directions are recommendations for epic enablers that harmonize development and delivery technology stacks, interoperability of solutions, APIs, and host strategies [32].

Value stream level

The value stream level is intended for builders of large and complex solutions that require multiple ARTs, Agile Release Train as well as the contribution of suppliers. ART is a long-lived, self-organized team of agile teams (5-12 teams), a virtual organization that plans, commits and executes together. Building complex solutions requires additional constructs, artefacts and coordination. That is why this level contains an economic framework which provides financial boundaries for value stream decision-making, solution intent as a repository for intended and actual solution behaviour. The solution behaviour is a solution context which describes the way the solution fits in the development environment and capabilities which describes the larger behaviour of the solution.

The value stream is organized around PI, program increments, which are synchronized across all the ARTs in the value stream. PI is a cadence-based interval for building and validating a full system increment, demonstrating value and getting fast feedback. The value stream provides cadence and synchronization of multiple ARTs and suppliers. There are three important roles that are necessary for coordinating and advancing the value stream: Value Stream Engineer (VSM), Solution Manager and a Solution Architect/Engineer.

The VSM is responsible for seeing that the value stream is running as it should and make sure that bottlenecks are identified and resolved across the entire value stream. The value stream level meetings are facilitated by the VSM and the value stream Kanban system and the value stream health is monitored by the VSM.

The solution management represents the customer's needs as well as the strategic themes of the portfolio vision. They define capabilities and decompose them into features together with the ART's product management. They are responsible for the value stream backlog and its prioritization as well as the creation of an economic framework to govern the decision-making process for ARTs and agile teams.

The Solution Architect/Engineer is a team that is in charge of defining the overall architecture that connects the solution across ARTs. They also help guide the system architect/engineering team with the architecture developed by the teams in the ARTs.

The solution intent manages the solution behaviour and decisions so all the requirements are stored in the solution intent. The solution context characterizes the environment in which a deployed solution operates. The solutions are built to satisfy the customer's needs and are built up of capabilities and enablers, who are needed to realize the value stream vision and roadmap. The value stream Kanban system manages the capabilities which mean that the capabilities are evaluated and analyzed. The capabilities is then stored in the value stream backlog where they are queued for implementation. The Kanban system limits the WIP, work in progress, which ensures that all the ARTs are synchronized and have the capacity to deliver capabilities together. Large solutions may require suppliers that develop components, subsystems or capabilities for the value stream. The suppliers will then participate in value stream level meetings.

The coordination of multiple ARTs that deliver a solution together is also handled in the value stream level. All the ARTs in the value stream are synchronized around a PI which facilitates the collaboration between ARTs and suppliers. At the beginning of a PI all the ARTs are planned at the same time. The planning is done in PI-planning meetings where the planning is done by the ARTs. At the end of a PI, a solution demo is held. The value stream shows the customer and stakeholders an integrated solution across all ARTs and suppliers. A workshop is held after the demo where they inspect and adapt the process of the value stream [34].

Program level

In the program level the development teams and other resources are applied to ongoing development mission. One of the responsibilities of the program level is to discover, define and develop features and enablers that are required by the business to realize the vision and roadmap. A program Kanban system is used to manage the flow and make it visible for all stakeholders. The system is used to ensure that features are reasoned and analyzed prior to reaching the PI boundary. It is also used to ensure that features are prioritized and that acceptance criteria have been established. The features are then maintained and prioritized in the program backlog and when it is time for implementation they are sized to fit in a PI such that each PI delivers new functionality with conceptual integrity. Features lie between stories and capabilities where stories are handled by a team within an iteration and capabilities are value stream level services that can span ARTs.

ARTs are self-organized and self-managing teams, also called trains, where each team consists of several agile teams. All the trains in an ART plan, commit and execute together however a train does not create or steer itself. Each train needs guidance and direction so that the teams are aligned to a common mission. There are three key roles for the program level: Release Train Engineer (RTE), Product Manager and System Architect/Engineer.

The release train engineer is a leader and the chief scrum master for a train. The RTE helps optimizing the flow of the value though the program by using different mechanisms such as the program Kanban, PI planning and workshop where the process is inspected and adapted.

The product manager is the internal voice of the customer and works with customers and product owners to get an understanding of their needs. With that knowledge they communicate the needs, define the system features and participate in validation. They are also responsible for the program backlog.

The system architect/engineer is an individual or small team that defines the overall architecture for the system, define nonfunctional requirements and determine the major elements and subsystems. They also help define the interface and collaborations among the elements and subsystems.

Programs have a bidirectional connection to the value stream and/or portfolio. The vision and roadmap provide a view of the solutions and capabilities that are going to be developed, reflecting customer and stakeholder needs and the approaches that are proposed to dress those needs [35].

Team level

All SAFe teams are part of one ART and the ARTs are the central construct of the program level. The team level provides an organization, artefact, role and process model for the activities of agile teams. Each team is responsible for defining building and testing stories from their team backlog in a series of fixed-length iterations. The teams use ScrumXP or Team Kanban to routinely deliver systems with high quality and produce a system demo every two weeks. This approach ensures that all the different teams on the ART create an integrated and tested system. The stakeholders can then evaluate the system demo and give fast feedback to the team. The system demo creates a routine where the effort from the different teams are pulled to a specific point which brings the hard work of integration and testing forward and is not left behind till the end.

The team level describes how agile teams power the ART. Each team has 5-9 team members where all the necessary roles are included. ScrumXP roles include the scrum master, product owner, dedicated individual contributors and resources that the team needs to deliver value. Team Kanban roles are not so specific defined however many Kanban teams implement the ScrumXP roles as well. Each team is supported by the program personnel so the team is fully capable of defining, developing, testing and delivering working and tested systems each iteration.

All the teams on an ART are synchronized and integrated so the teams share common iteration start/stop dates and duration. The teams plan and execute two week iterations and each iteration provides a valuable increment of new functionality. This is accomplished via a constantly repeating pattern:

1. Plan the iteration.
2. Commit to some functionality.
3. Execute the iteration by building and testing stories.
4. Demo the new functionality.
5. Hold a retrospective. It is a meeting where the team members discuss their practices and identify ways to improve their process.

Repeat the steps for the next iteration. In the end of each iteration the other teams support the system demo, which is the critical integration point for the ART.

The IP, Innovation and Planning iterations provide the teams with an opportunity for exploration and innovation, dedicated time to plan, retrospect, and learn through informal and formal channels. If it is time for a release the teams perform final system verification, validation and documentation instead of IP. The teams do not plan stories for the IP iteration because it increases flow, throughput and delivery reliability.

There is no rule for how many iterations a PI should be but experience shows that a PI duration of between 8 and 12 works best. The teams use stories to deliver value and the product owner has content authority over the stories creation and acceptance. The stories are the customer's requirements that have been carried through the value stream into implementation. The team backlog contains user and enabler stories which are identified during the PI planning. The primary requirement management processes in the team level are to identify, prioritize, schedule, elaborating implementing, testing and accepting stories [36].

2.4.3 Built-in quality in SAFe

Built-in quality is one of the four core values of SAFe. The Enterprise wants to deliver new functionality as fast as possible and to be able to react to rapidly changing business environments. How fast they can do this depends on the solution quality. The software practices help teams ensure that the solutions they built have high quality and are ready to adapt to change. When enterprises need to respond to change, the software and systems that are built on good technical foundations are easier to change and adapt. This is very critical for large systems since the cumulative effect of minor defects can create unacceptable consequences. To achieve high-quality systems it requires ongoing training and commitment but it has many business benefits:

- **Higher customer satisfaction:** Products and services with low defect-density provide higher levels of customer satisfaction, higher safety, better return on investment and higher confidence for the business.

- **Predictability and integrity of the development organization:** It is impossible to plan new business functionality predictably or to understand development velocity without reliable control over solution quality. The consequences of no control is a lack of trust in the business, reduces individual pride of craftsmanship and lowers morale.
- **Scalability:** Adding more teams and functionality when the components express design intent, exhibit simplicity and have supporting integration and test infrastructure, will result in more business value delivered. If that is not the case, adding more developers and testers will just slow down the enterprise and create unacceptable high variability in quality and release commitments.
- **Velocity, agility and system performance:** Higher quality leads to better maintainability and enhance the ability to add new business functionality more quickly. The team's ability to maintain and improve critical non-functional requirements, is also improved. This includes performance, scalability, security and reliability.
- **Ability to innovate:** The confluence of smart, motivated persons and the environment that they operate brings innovation. High quality creates a technical environment where new ideas can easily be prototyped, tested and demoed.

There are some recommended practices for achieving built-in quality of software, firmware and hardware. In the section below there is a summary of the practices for software. Information about the practices for firmware and hardware are not discussed in this report since the focus is on software. Information about the practices for built-in quality of firmware and hardware can be found on [37].

Built-in quality of software

There are five critical Lean-Agile software engineering practices that support teams to achieve the highest level of quality. These practices are: Continuous integration, Test-first, Refactoring, Pair work and Collective ownership.

Continuous integration: Continuous integration is the practice of merging each developer's code from their workspace into a single main branch of code. This should be done multiple times per day because then all developers on a program are constantly working with the latest version of the code. With this approach all errors and assumptions made among developers and teams are discovered immediately. This helps avoid the risk of deferred system-level integration issues and their impact on system quality and program predictability. Special infrastructure and tools are required for continuous integration. The most important aspect is that the individuals and teams only give themselves credit for fully integrated and tested software.

The teams should perform local integration at least once a day but the responsibility to integrate to the full system and run regression tests are more important. The regression tests are important since they ensure that the system is evolving in the intended manner. This may not be feasible every day so it is better to start by having full system-level integration at least one or two times per iteration. This provides the baseline that is needed for a successful biweekly system demo.

Test-first: This is described in detail in Section 2.4.4.

Refactoring: Refactoring is a technique that restructures an existing body of code which means that the internal structure is altered without the external behaviour being altered. The current code base can be changed at any time because agile welcomes changing requirements, even late in the development. To maintain this resiliency, the teams continuously refactor code in a series of small steps to provide a solid foundation for future development. If omitting refactoring it will quickly create a system that is rigid and unmaintainable which ultimately drives increasingly poor economic outcomes.

Pair work: Pair work, also called pair programming, involves two programmers working at a single workstation. One programmer codes and the other one is an observer which reviews, inspects and adds value to the coding process. The programmers switch roles thought a paired session. First of all, the pair discusses what needs to be done which gives them an understanding of the requirements, design and how to test the functionality. The observer inspects the work, makes suggestions about how to improve the code and point out potential errors. Peer review is done in real time and is integral.

There are many different styles of pairing work done by agile teams where each has value on its own and many can be used in combination. Some teams follow the standards of pair programming for all code development whilst others pair developers and testers together on a story. The developers and testers then reviews each other's work as the story moves to completion. A third option is a more spontaneous pairing, with developers pairing for critical code segments, refactoring of legacy code, development of interface definition and system-level integration challenges. Pair work is also useful for refactoring, continuous integration, test automation and collective ownership.

Collective ownership: Collective ownership encourages everyone to contribute new ideas to all segments of the project. It will be possible for any developer to change any line of code to add functionality, fix bugs, improve designs, or refactor. This is very important since then one person does not become a bottleneck for changes. This is very critical in big systems because they have big code bases and there is a big chance that the original developer is not a part of the team or program anymore. Even if that person is still in the team or program, waiting for one individual to make a change is a hand-off and leads to a certain delay. To be able to implement collective ownership at scale requires following proven, agreed-to coding standards across the program: embracing simplicity in design, revealing intent in code constructs, and building interfaces that are resilient to changing system behaviour [37].

2.4.4 Test in SAFe

Test in agile development differs a lot from the test-at-the-end approach in waterfall based development. In agile development the code is developed and tested in small steps where the development of the test precedes the development of the code. With this approach the tests serve to elaborate and better define the intended system behaviour before the system is completely coded. The quality of the system is built in from the beginning. This approach also alleviates the need for lengthy, detailed requirements specifications and sign-offs, which is the most common way to control quality in traditional software development. The tests in agile development are automated wherever it is possible. Even when they are not automated, they provide a definitive statement of what the system actually does.

Brian Marick, who is one of the writers of the agile manifesto [38], has made an agile testing matrix that guides reasoning about tests. This matrix have been further developed and extended for the scaled agile paradigm. Figure 2.8 describes the agile testing matrix. The matrix contains guidance on what to test and when. The horizontal axis contains business-and technology-facing tests. The business-facing tests are tests that the user understands and are described using business language. Technology-facing tests are written in developer language and they are used to control that the code delivers the behaviour the developer intended. The vertical axis contains tests supporting development or criticizes the solution. Tests that support the development evaluates the internal code and the tests that criticize the solution evaluates the system against the user's requirements.

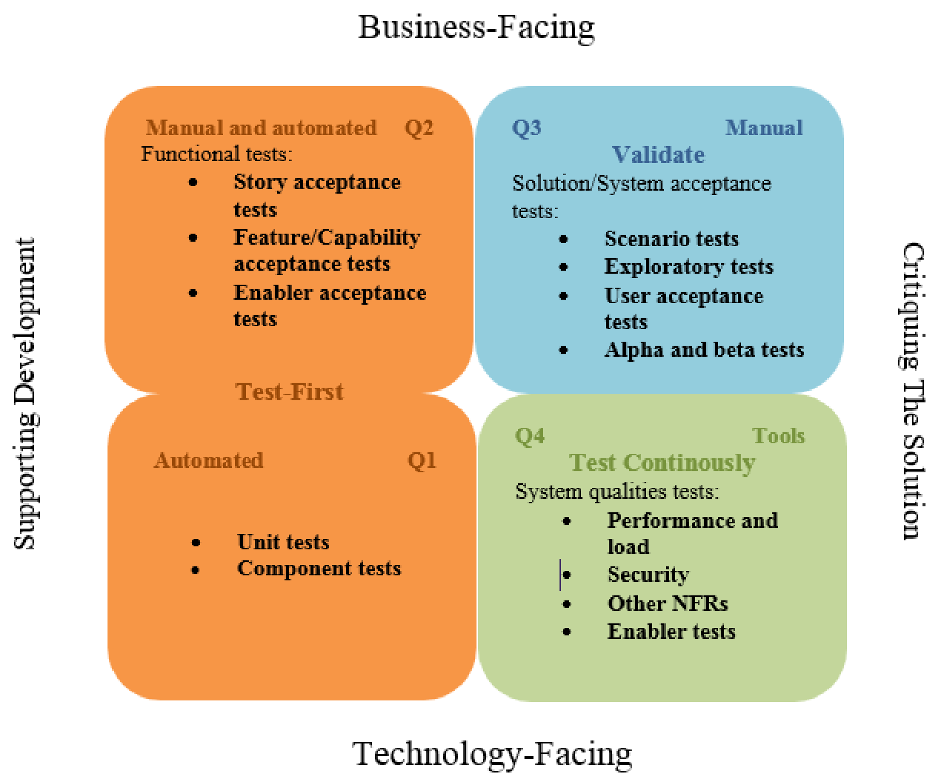


Figure 2.8: Agile testing matrix [39]

The matrix is divided into four quadrates and they enable a comprehensive testing strategy that helps ensure quality. The quadrates are described below:

- **Q1** contains unit and component tests. These tests are automated and they are written by developers. The tests are executed before and after all code changes. This confirms that the system works as intended. By automating their execution, the overhead of executing a large number of tests in different development or test environments are reduced.
- **Q2** contains functional tests and user acceptance test for stories, features and Capabilities. These tests validate that the stories, features and capabilities work as intended by the product owner. Acceptance tests on the feature- and capability-level validate the joint behaviour of many user stories. These tests are automated by the team whenever it is possible and manual testing is only used when it is absolutely necessary.
- **Q3** contains system-level acceptance tests which validate that the behaviour of the system meets the usability and functionality requirements. The tests can include exploratory testing, user acceptance testing, scenario-based testing, final usability testing and more. These tests are often done manually since they involve users and testers that are using the system in actual or simulated deployment and usage scenarios. System-level acceptance tests are used for final system validation and they are required before delivery to the end user.
- **Q4** contains system quality tests which verify that the system meets its non-functional requirements. It is common that these tests are supported by a set of automated testing tools which can for example be load and performance. These tools are designed specifically for this purpose. These tests should be executed continuously since any change to the system can violate conformance with the non-functional requirements.

The functionality of the system is tested in quadrant 1 and 2. These tests can be developed before the code is committed and that is called Test-First. Both test-driven development (TDD) and acceptance test-driven development is included in Test-First. Both methods use test automation to support continuous integration, team velocity and development effectiveness [39].

Test-Driven Development

The focus of test-driven development is that unit tests are written before writing the code. The process is described below:

1. Write the test first. When writing the test first it ensures that the developers understands the required behaviour of the new code.
2. Run the test and watch it fail. There is no code yet to test but this accomplishes two useful objectives. The first one is that it tests the test itself and the tests that harnesses that hold the test in place. The second one is that it illustrates how the system will fail if the code is incorrect.

3. Write the minimum amount of code that is necessary to pass the test. Rework the code if the test fails. It can also be that the test is necessary until a module is created that routinely passes the test.

This practice was primarily designed to operate in the context of unit tests. Unit tests are written by the developers and they test the classes and methods that are used. Working in pairs is used extensively since when two sets of eyes have seen the code and tests, there is a big chance that the module is of high quality. Developers often refactor the code in order to pass the test as simply as possible. This is quality at the source and is one of the main reasons that SAFe relies on TDD.

To prevent Quality Assurance (QA) and test personnel from spending most of their time finding and reporting on code-level bugs, most TDD is done in the context of unit testing. This allows the QA and testers to focus more on system-level testing challenges because more complex behaviour are found there. The complex behaviour are based on the interactions of the unit code modules. There are unit-testing frameworks for most coding constructs a developer is likely to encounter. These frameworks provide a safety belt for the development and maintenance of unit tests and for automatically executing unit tests against the system under development. Unit testing can be accomplished within the iteration since the unit tests are either written before or concurrently with the code, and the unit testing frameworks include test execution automation. The unit test frameworks hold and manage all the unit tests together which means that regression testing automation for unit tests is largely free for the team. Unit testing is one of the cornerstone practices of agile software and all investments that is made towards more comprehensive unit testing will increase the quality and productivity.

Component testing is used to test larger-scale components of the system. Since all components work differently, testing tools and practices for implementing component tests vary. Many teams are using their unit testing framework to build component tests but others may use other testing tools or write fully customized tests in any language or environment. Component tests are also automated and they serve as a primary defence against unanticipated consequences of refactoring and new code [39].

Acceptance Test-Driven Development

The goal with acceptance test-driven development is to have the system work as intended and not to simply have the code do as intended. Many teams find it more efficient to write the acceptance test first and then developing code. These tests emphasize understanding requirements prior to implementation. The purpose with acceptance tests is to make the team understand the specifics of the intended behaviour that a story represent. All the decisions that is made in the conversation between the team and the product owner is recorded as acceptance tests.

A story’s acceptance test is a functional test that intends to ensure that the implementation of the user story is delivering the intended behaviour. The testing is performed during an iteration and if all the new stories work as intended, it is likely that each new increment will satisfy the needs of the users. The feature and capability tests are performed during the course of a PI. Generally the same tools are used however these tests operate at the next level of abstraction. This means that some number of stories are tested to see how they work together to deliver a larger value to the user. There can be multiple story acceptance tests that are associated with a more complex story, and the same goes for features. So there are a strong verification that the system works as intended at the story, feature and capability levels. Functional tests have the following characteristics:

- Written in business language.
- Developed in a conversation between the developers, testers and product owner.
- Tests that verify that the outputs of the system meets the conditions of satisfaction. They have no concern for how the result is achieved.
- Implemented during an iteration where the story is implemented.

The product owner is generally responsible for the efficacy of the tests. If a story is not able to pass its test, the story is carried over into the next iteration. The code and/or the test are then reworked until the story passes the test. Stories, features and capabilities are considered done when they have passed one or more acceptance tests. It is very common that more than one test are associated with a particular story, feature or capability.

There are a variety of approaches to execute acceptance tests since they run at a level above the code. One approach is to handle them as manual tests however it is very common that they pile up very quickly which slows down the team and introduces major delays in value delivery. So to avoid this the teams can automate their acceptance tests and they use a variety of tools for this. They can use the target programming language, natural language that is supported by specific testing frameworks or table formats. The preferred approach is to take a high level of abstraction that works directly against the business logic of the application. This prevents obstruction by the presentation layer or other implementation details [39].

Non-Functional Requirements

Most of the non-functional requirements are system qualities and they are testing the systems non-functional requirements (NFR). It is very common that a collaboration between the system team and the agile team is needed when testing because of the scope and criticality of the NFR. The teams should automate these tests wherever it is possible so that the tests can be executed continuously to help prevent the growth of unexpected technical dept. The accumulated growth of regression tests may consume too much resources and processing time. This can mean that NFR testing may be practical only on occasion. Teams need to create reduced test suites and test data to be able to ensure practicality and continuous use of NFR testing.

Partial testing can be beneficial in increasing system quality. Some benefits with partial testing is listed below:

- The team may spot inconsistencies in the test data or testing approach when they apply reduced test suites locally.

- Teams may create new and unique tests, some of which may be adopted by the system team to help build the larger set.
- Continuously improve testing infrastructure and configurations.
- Teams gets a practical understanding of the impact of NFRs. This helps the team to improve the estimation of business and architectural features.

The environment where the NFRs are tested may not always be available so then the team has to use one of the following approaches to do the tests anyway:

- Using virtualized hardware.
- Creating simulators.
- Creating similar environments.

It is very important to efficiently test NFRs and this requires some thought and creativity. A lack of NFR testing may increase the risk of substantive technical debt or system failure [40].

2.4.5 Implementation of SAFe

To implement SAFe into an organization is a big change for most organizations. Figure 2.9 shows an overview of the three step implementation of SAFe. All the numbered items are described below.

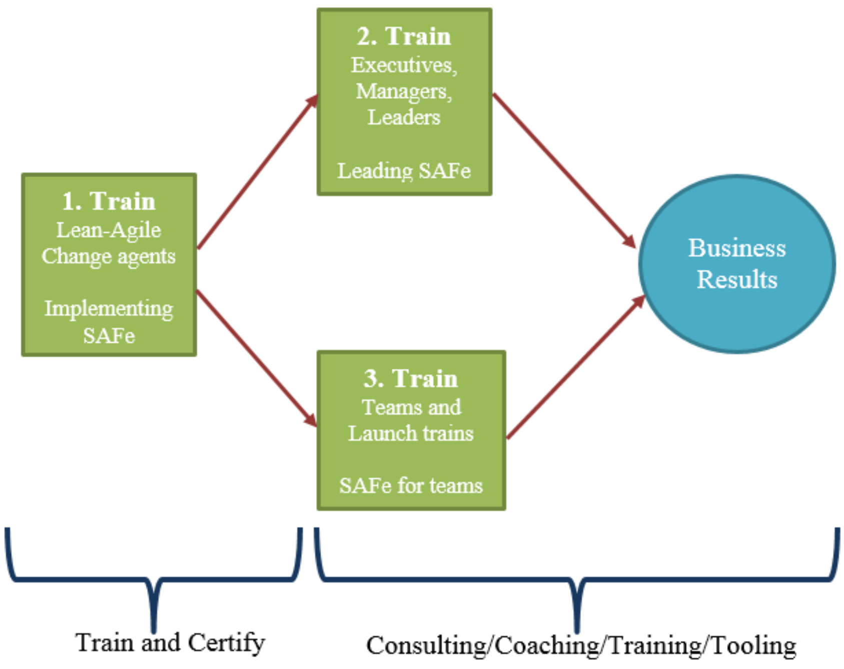


Figure 2.9: Implementation of SAFe [41]

1. Train Implementers and Lean-Agile Change Agents
To have a successful adoption of SAFe when the scope, challenge and impact of rollouts are given, requires that most enterprises use a combination of internal and external change agents, leaders, mentors and coaches. All of these persons have to be skilled in teaching and delivering SAFe. This can be achieved by entering a class at Scaled Agile, Inc. Then the attendees will be able to lead an enterprise Agile transformation with SAFe, implement SAFe, launch ARTs, monitor and continuously improve the trains via inspect and adapt workshops.

2. Train all Executives, Managers and Leaders
It is very important that the executives, managers and leaders understand what is required to lead a transformation to SAFe, how and why SAFe works. Scaled Agile, Inc. also provides a course for this and after the course the participants will be able to:

- Adopt a Lean-Agile mind-set.
- Apply Lean and Agile principles. Also understand, exhibit and teach these principles.
- Understand the principles, roles, activities and artefacts of SAFe.
- Unlock the internal motivation of knowledge workers.
- Learn the practices and tools of adamant improvement. Also to teach employees problem-solving and corrective-action skills.
- Become practical with the new process adoption, eliminate obstacles and facilitate organizational change management.
- Take the responsibility for the success of the Lean-Agile implementation.

3. Train Teams and Launch Agile Release Trains
The ART is the mechanism that primary deliver value in the enterprise. It can be very difficult to start these ARTs but there is a specific course that is proven successful. The course contains the following parts:

- Organize the members into Agile Teams and training them simultaneously in the principles of Lean, Agile and SAFe.
- Trains the teams together with a course that is distributed by Scaled Agile, Inc.
- Adjusts the teams on the ART to a common goal. Also spends two days in face-to-face support of planning the next PI.
- Introduces the future product owners and scrum masters to the skills and the unique activities for their roles in the new Agile Enterprise.
- Builds context and a cadence-based, rolling-wave planning and delivery model. The model continuously incorporates business objective setting and program commitments, effective and reliable program execution, and adaptive feedback.

Once the enterprise has gone through these three parts and has a few Agile Release Trains rolling, a variety of consulting activities may be applicable and beneficial. The activities could be coaching the train, training specialist roles and continuous improvement. The coaches share their knowledge and experience which helps the teams and individuals improve their new found skills. They can help them with the following:

- Build the organization's Lean-Agile capabilities by providing program consulting and team coaching.
- Facilitate the ART readiness which includes refinement of the program backlog and more.
- Facilitate inspect and adapt workshops.
- Facilitate Value Stream mapping and Portfolio planning workshops.
- Implement relevant metrics and governance.
- Mentor executives, managers and other stakeholders.
- Shadowing and mentoring Release Train Engineers.

It is also important to train specialists in the principles and practices unique to their role. Prospective product owners and scrum masters are also included in the training. There are several courses that are distributed by Scaled Agile that the specialists can enter to get their training. Once the transformation is ongoing, there are a variety of opportunities for sustaining and enhancing improvements. Improvements can be done in speed and quality which can be best facilitated by the comprehensive community of skilled professionals [41].

2.4.6 Mixing Agile and Waterfall development in SAFe

It can be difficult for an organization to go all in with agile development for various reasons such as dependencies on external partners who are using waterfall development or internal culture that requires time to shift to an agile mind-set. The biggest challenge is to know where in the process you should integrate. Three different scenarios will be discussed in this section along with how to address them.

Before describing these scenarios, some important aspects of the program should be explained. These aspects should be the same regardless of whether there is a mixed situation or not. One aspect is that the PMO (Program Management Office) can provide governance across both the waterfall and the agile programs. By having a common approach, governance will allow key decisions to be made on the metrics being used. A key aspect of the PMO is that it should adopt a mind-set that performs the same core functions of a traditional PMO but does it in an agile manner. The traditional PMO contains strategy and investment funding, program management, and governance. This is shown in Figure 2.10.

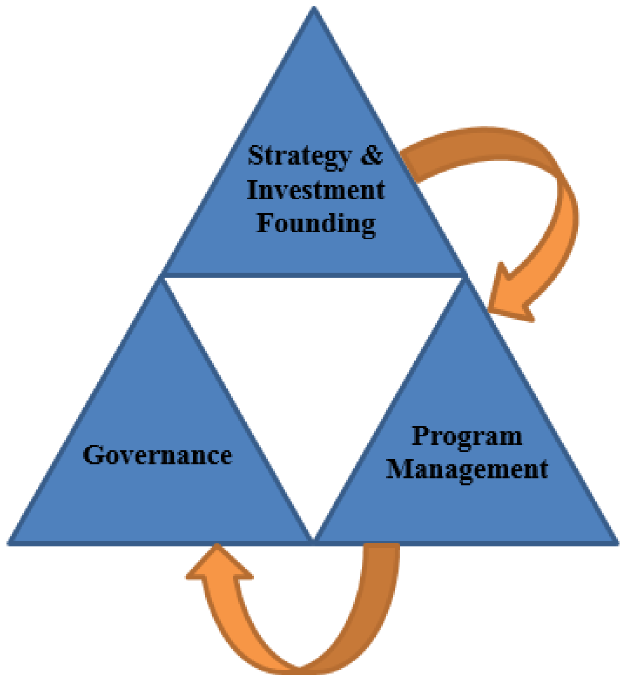


Figure 2.10: PMO [42]

There are some key agile mechanisms that can be used to perform each of the traditional PMO functions. These mechanisms are:

- Strategy and investment funding:
  - Decentralized decision-making.
  - Continuous value flow.
  - Lightweight business cases.
  - Decentralized rolling wave planning.
- Program management:
  - Agile estimating and planning.
  - Agile Release Trains.
  - Agile Program Management.
- Governance:
  - Objective, fact-based measures and milestones.

The program may need to adopt some of these mechanisms gradually but the focus should be to continually find ways to evolve the PMO towards agile/lean thinking [42].

Mixed model scenarios

There are three different scenarios that a company can have when mixing Agile and waterfall development. The three mixed model scenarios are called: independent teams, low dependency and high dependency. Independent teams means that the waterfall and agile programs are completely independent of each other and just need to be deployed in a common release. Low dependency means that some dependencies exist but constant synchronization is not required for success. High dependency means that there are substantial dependencies throughout the development. For each program to success, a high level of interactions and synchronizations are required.

Synchronization is required for each of the scenarios to ensure that the program progresses as planned. The synchronization is on different level for each of the scenarios. Synchronization at the portfolio level is sufficient in the independent scenario. In the low dependency scenario more frequent synchronization is required. It should minimum synchronize at the potentially shippable increments (PSIs) which are defined in the program level. PSI is the output from each sprint which means that all the work has been done for the implemented feature and it can be shipped to the main branch. High dependency scenario requires frequent synchronization between teams and these occurs within the defined cadence at the Team level. The scenarios are described below.

Scenario 1: Independent teams
In this scenario it is not required with synchronization throughout the project since there are no deep dependencies between the programs. Sometimes these teams can deliver into production at different times or they come together at the end so they are included in the same release in order to maintain consistency with the user community. These teams are synchronized at the Portfolio level since the functionality that is delivered by each team does not have to be integrated prior to release.

Even though the teams are independent, it is still important for the PMO to be able to monitor progress of each team in a similar manner. To accomplish this, a common governance structure can be established which defines the same milestones with different definitions for the teams. Four common milestones are Planning, Design, Integration and Release. The Release milestone is self-explanatory however the planning and integration milestones will have a different meaning between the two teams. Figure 2.11 is showing the synchronization between two independent teams.

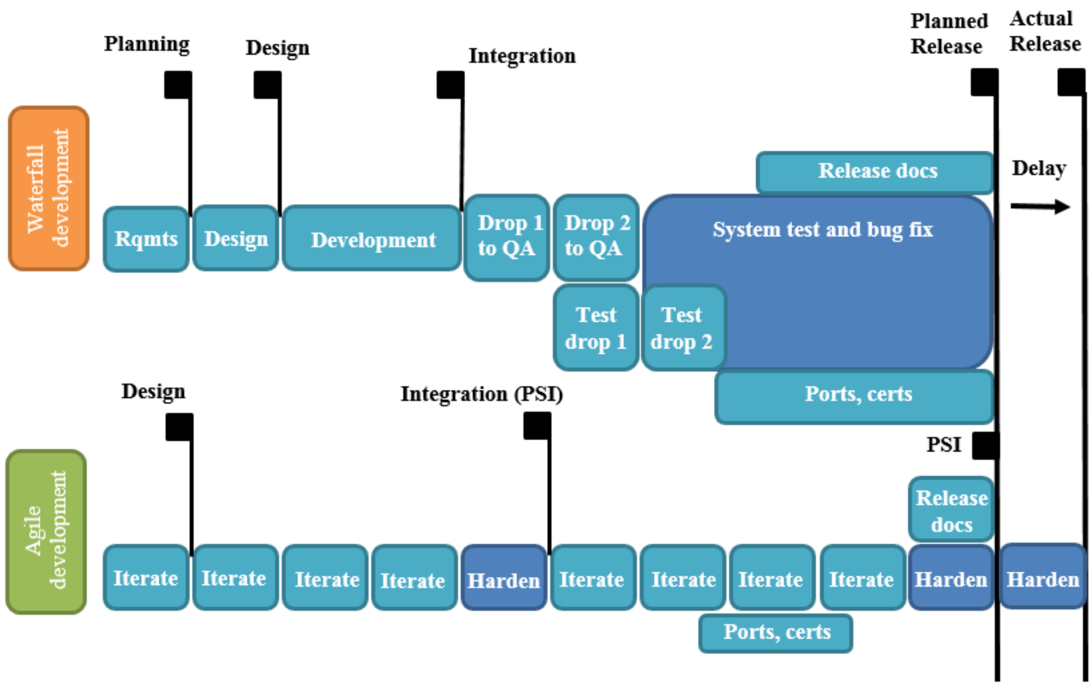


Figure 2.11: Scenario 1 [42]

Scenario 2: Low dependency teams
This scenario is slightly more complicated since it introduces temporal synchronization of the governance milestones. This is necessary so that the teams can ensure that their individual components will integrate well. Where possible, programs strive to align these milestones with iteration or PSI boundaries at the Program level. The agile team has a faster "clock speed" than the waterfall team so it is easier for the agile team to adjust its plans to align with the waterfall milestones. It is possible that this may introduce another hardening iteration into the PSI to allow for an integration point or to remove a development iteration which may allow an earlier integration point. It would be ideal for the waterfall team to adopt a more incremental approach to develop the product. This would remove the long release cycles and synchronization, integration and fast feedback comes earlier in the process. Figure 2.12 is showing the synchronization between two teams with low dependency.

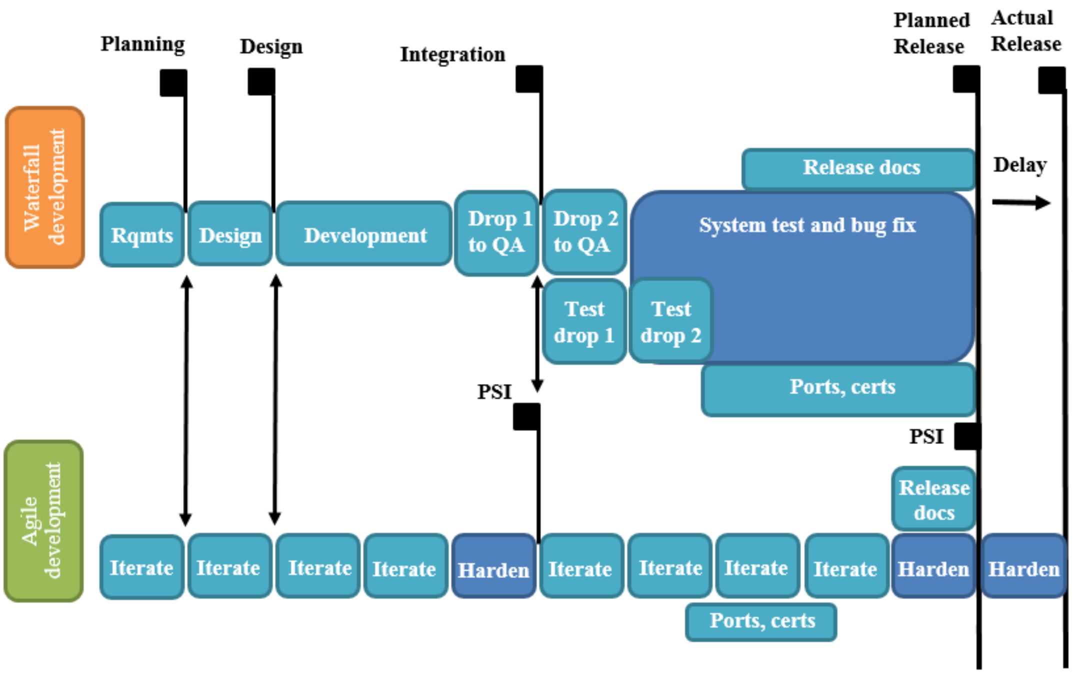


Figure 2.12: Scenario 2 [42]

Scenario 3: High dependency teams
This is the most complicated scenario which requires the highest level of synchronization. The synchronization is required at the Team level and should happen at least every iteration boundary. In some cases, daily interactions may be required for key issues or dependencies and they could occur at both the technical level or at the business level. The main key is that the agile teams should not wait for the waterfall teams to complete their entire development cycle before they begin to integrate. The agile teams have built-in events for these regular synchronizations such as PSI planning, iteration planning, daily stand-ups and iteration demos/retrospectives. Because of the high volume of dependencies it is required to have a detailed tracking of the dependencies and associated risks/issues. It will be very common for the agile team to make assumptions in order to make progress on its iterations while the waterfall team progresses towards an integration point. It is very important that these assumptions are validated on a regular basis and if necessary, are used to drive the appropriate re-factoring within the agile team for assumptions that was not true. Figure 2.13 shows the synchronization between two teams with high dependency [42].

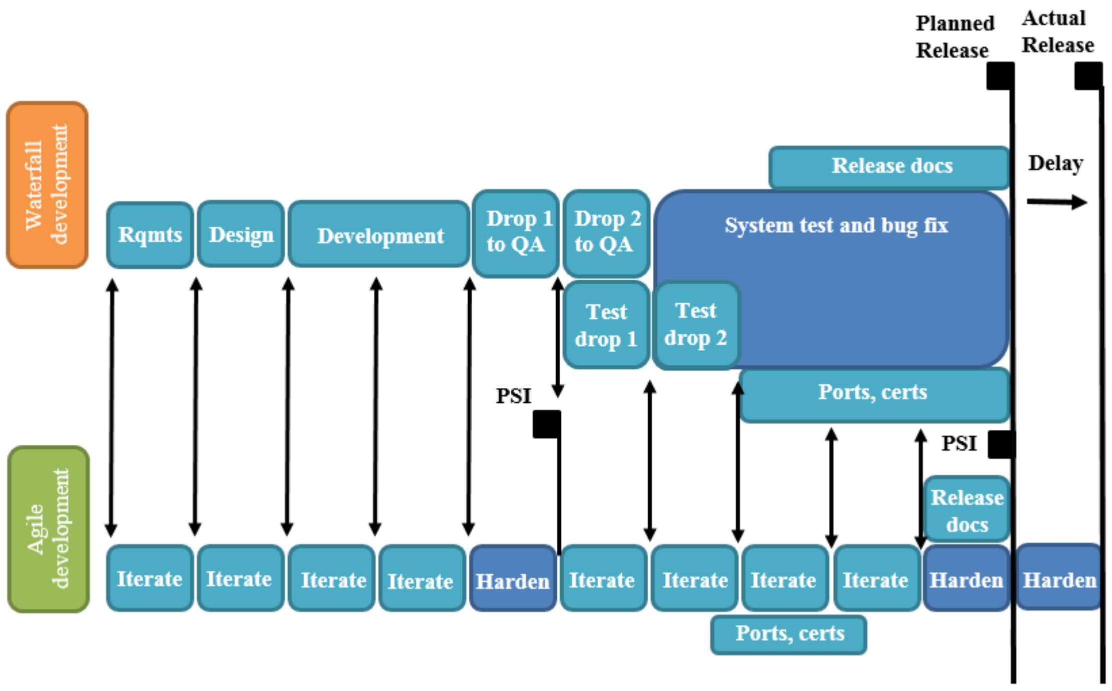


Figure 2.13: Scenario 3 [42]

Technical strategies

It is important to have technical strategies when using a mix of agile and waterfall development because they provide better risk management. The primary goal for establishing effective collaboration between these teams, is to validate all assumptions as early as possible. If the problems are discovered near the end it could have a dramatic effect on the expected outcome. There are four suggestions on mechanisms that teams can use to address this mixed mode of operation. The suggestions are: Joint requirements and design workshops, Mockups and designing to interface, Frequent integration, and Use design patterns and refactoring.

Joint requirements and design workshops

Waterfall development is largely based on large initial requirements and design but these activities can be conducted in a highly collaborative manner. All dependencies are identified and resolved when representatives from both the agile and waterfall teams meet together in front of a whiteboard with all requirements. Some tips on how to run such a workshop are listed below:

- The team should do some work before the meeting. It is not necessary to pursue all dependencies but they should try to build a comprehensive view of the entire initiative. When doing that, the dependencies will pop up. The same process is then done together with the other team. This is necessary to obtain the right understanding and context as preparation for effective joint meetings.
- Use whiteboards, walls and flipcharts extensively.

- It is important to give examples when specifying because this derives concrete system behaviors and eliminates ambiguities. It is also useful to use a demo-driven approach and these examples should be saved and used by both teams over the course of the project.
- Then identify dependencies and plan for integration points.

Mockups and designing to interface
Mockups allow agile teams to test interface assumptions before the system/code is available. A physical integration helps a lot and the interface becomes the most real thing possible when one of the two teams only deliver working software in the end. Some tips are listed below:

- Mockups are useful when they bridge both the system integration and personal communication gaps between the teams. Interfaces are simulated by the mocks and the interfaces should first be discussed at the design workshop. The best scenario is when the waterfall teams create mockups for the agile teams. When this is not possible, the agile teams will do it but the mockups have to be reviewed by the waterfall teams so that the mocks not only test a set of assumptions.
- The teams should design to interfaces when using mocks. It is highly recommended to couple mockup development with the interface structure that the programming languages provide. Then the developers can simply replace each mockup with real implementations, once they are available. The teams should capture all inconsistencies as soon as they are found.

Frequent integration
The code and the assumptions that went into an integration are tested for each frequent integration. The mockups wants to implement the logic that is associated with the mockup interface which fosters integration. An integration can be full or partial but the important part is that teams integrate with no fear. When an integration fails, the teams should be driven by an inner desire to fix the integration errors first and then after that, go back to their own agenda. This is nothing like continuous integration however it still has huge value even though its on-demand and manual, not automated.

Use design patterns and refactoring
System architects who operates in waterfall environments often claim to use design patterns in their systems. Agile has a different interpretation of pattern usage which has very important consequences. Patterns are important because of the connatural complexity associated with software development and our inability to predict either requirements or system design. Some systems behavior will inevitably change over time of the implementation. The patterns provide different design approaches that are more immune to change and are more suitable to refactor. Some design patterns foster separation of concerns and some of them are called: Bridge, Adapter, Decorator, and Chain of responsibility. To foster the use of mockups, testability, and early integration as instantiation of the entities can be used to separate them from their use. Facade patterns fosters early validation of inte-

gration with legacy systems. The patterns should not be considered as building blocks for the system. They should be considered as ways to isolate variability [43].

2.5 Test at Company X

As mentioned before Company X is using a waterfall based development approach. The information in this section has been retrieved from Company X's internal documents that are only available for the company. Their current testing process is risk based and divided into three main phases: Planning and preparation, Execution, and Acceptance. These phases are all done in a sequence starting with planning and preparation, followed by execution and an acceptance phase. Planning and preparation is the phase where the test team performs a Product Risk Analysis regarding requirements or change requests to implement. They also defines a test strategy and test plans based on the requirements and identified product risks, prepares test cases, orders needed test environments and coordinates test needs between different teams.

Execution is the phase where planned test levels are executed, reports are created in parallel with iterative business demonstrations to provide early validations and test results are being measured and followed-up. The final step in this phase is that an Operational Readiness Test (ORT) is executed as the final test level before handover to the acceptance phase

The last phase Acceptance starts with a test delivery walk-through to secure that the test delivery is complete and acceptance criteria met. Then Business Acceptance Test (BAT) is done to verify that the requirements are fulfilled, followed by User Acceptance Test (UAT) to validate that the delivered solution fulfills the end-user needs. BAT and UAT results are evaluated and summarized in a final report and the phase ends with test environment clean-up and a lessons learned activity. This section contains information about how Company X's future testing process is intended to work.

2.5.1 Future test process principles

The future test process will have two mayor principles where one is that the test process should be integrated with the agile approach which means that:

- The test activities should be totally integrated into the development process.
- All individuals that are involved will be prepared to perform the test activities.
- Testing is the driving force to provide continual information on the product's quality.
- Test automation/monitoring/testability enablers is becoming increasingly important.
- Testing must be incorporated in the definition of done.

The second principle is that the test process must adhere to Company X current test process which means:

- Product risks are a driving force.

- Business involvement.
- Operational readiness focus.

2.5.2 Future test process

The structure of the future process, which is shown in Figure 2.14, have been developed by people that work with the implementation of SAFe at Company X. A development team develops a team increment, also called team feature that can consist of one or more stories. At the same time the test team tests all the stories and develops the automated tests for that increment and for the future integration. When it is time to deliver a new team increment to the system increment, the test team run both their automated tests and the manual tests. If any test fails the development team reworks the code so that the tests pass. When all tests have passed, the latest version of the system increment is integrated with the team increment on a separate branch. When the integration is done the test team runs some integration tests and if all tests pass they will push the integration to the system runway. If any integration test fails the development team pulls back their team increment and rework the code until it passes all the tests. All the tests that are on the team level is called team tests.

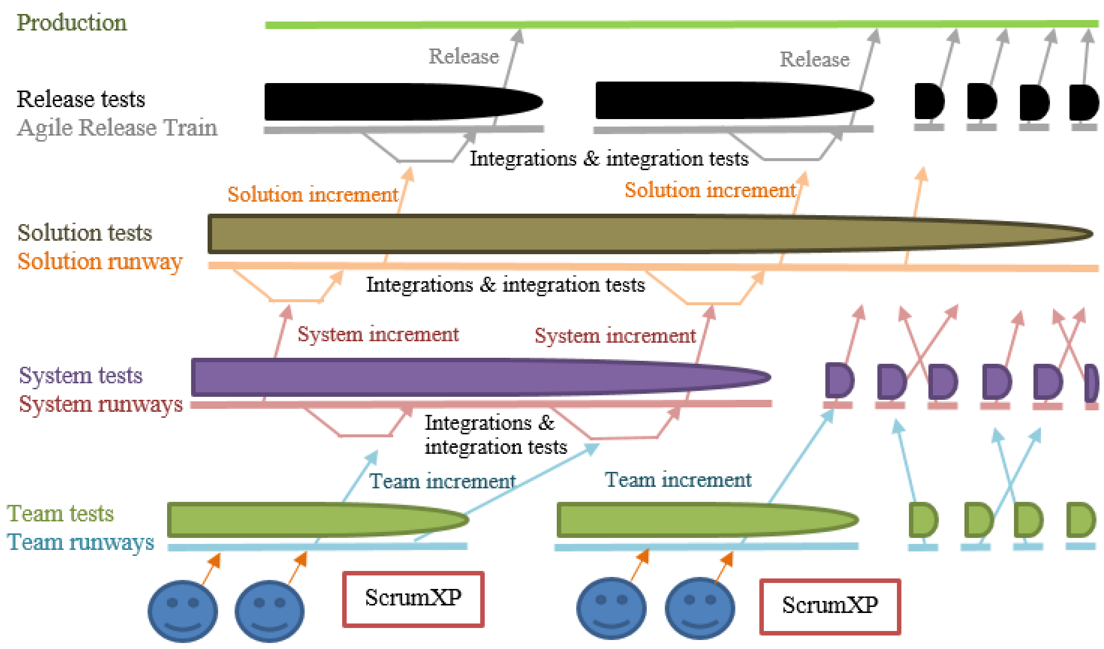


Figure 2.14: Company X's intended test process

A system increment consist of integrated team increments from all the teams that develop that system. During the development of all team increments the system test team will test the system increment each time there is a new team increment delivered to it. They will also develop the automated tests for the system increment and for the future integration. When it is time to deliver a new system increment to the solution increment, the system is tested. If all the tests passes, the system increment is integrated with the latest version of the solution increment on a separate branch. However if the tests fail the system increment is reworked until all the tests pass. When the integration is done some integration tests are executed and if all the tests pass the integration is pushed to the solution runway. All the tests that are on the system level is called system tests.

The solution increment consist of integrated system increments. During the development of all system increments the solution test team will test the solution increment each time there is a new system increment delivered to it. They will also develop the automated tests for the solution increment. All the tests that are on the solution level is called solution tests. When it is time to deliver a new release, the current solution increment is tested with the release tests. During the development of all the solution increment, the release test team develop the automated tests for the release. When all the tests have passed the release is delivered to the end user.

The goal with the solution increment is to always be releasable. This is achieved by allowing developers to take responsibility for the quality all the way to production. It is important with fast feedback which means that continuously reduce the time it takes between that a developer makes a code change and that potential errors are reported to that developer. It is also important to promote the tested and releasable code in such a way that everybody know where the latest releasable point is.

So the test process will consist of four different levels of tests: team test, system test, solution test and release test. The team and system tests work in the same way. Each test level is explained below:

- **Team/system test:** They check if the acceptance criteria are met. This means that they check if the functional requirements and non-functional requirements are met. The tests in the team are based on the built-in quality principles from SAFe: Continuous integration, Test first, refactoring, pair work and collective ownership. When performing these the team and system tests focus is on tests that can assist in the mitigation of the highest risks as early as possible. Service virtualization is used when performing the integration tests and when the dependent components are unavailable. Service virtualization is a method where you emulate the behaviour of a specific component. It gives the test team access to dependent components that are needed to perform the tests. The goal with these tests are to get feedback of valuable defect reports towards development stakeholders and valuable feedback regarding known issues and risk areas towards solution and release stakeholders.
- **Solution test:** They check if the solution is kept stable and releasable. When performing the tests the focus is on tests that were found to be out of scope in earlier increments. The tests, data and integrations are set up as production-like as possible. Before and after the integration tests are performed and the testers make non-functional risk based investigations and risk based functional tests. These assignments are session based tests which is a software test method. The purpose of these tests are to combine accountability and exploratory testing to provide rapid defect discovery, creative test design, management control and metrics reporting. A part of the integration testing is to perform solution regression test which verifies that the software that was developed and tested previously, still works correctly after

it was changed or interacted with other software. The goal with these tests are to get feedback of valuable defect reports which will give an overall solution perspective towards quality control stakeholders. It will also give valuable feedback regarding known issues and risk areas towards scrum teams and release test.

- **Release test:** They check that the release is in a GO-state for production, which means that the release is ready to be delivered to production. When performing these tests the focus is on tests/checks that were found to be out of scope in earlier increments. This is the final check-up which has the main focus on static review of documentation from earlier tests in combination with random sample tests. The release tests consist of three different tests: risk based performance test, user acceptance test and operational readiness test. Risk based performance tests are tests that determine how a system performs when it comes to responsiveness and stability. They are prioritized based on the risk of failure, the function of their importance and likelihood or impact of failure. User acceptance tests are tests that determine if the user's requirements are met. Operational readiness tests have the purpose to find defects in the business's readiness. It ensures that the business is ready for the system that is tested to go live such as server power and what to do if the system shuts down. The goal with these tests are to provide decision support regarding release GO/No-GO.

2.5.3 Connection to SAFe

The process of building a releasable solution in Company X is the same as the process in SAFe. The team increment is the team's finished iteration backlog stories which have been demonstrated at the Team Demo. This is done at least every iteration. The system increment is the team's completed backlog items which are integrated and demonstrated at the System Demo. This is also done at least every iteration. The solution increments are the ARTs assets which have been integrated, tested and demonstrated at the Solution Demo. This is done at least every PI. The release includes additional activities such as system validation, documentation, etc. This is done when required by the business. Figure 2.15 shows the connection between Company X's process and SAFe's process.

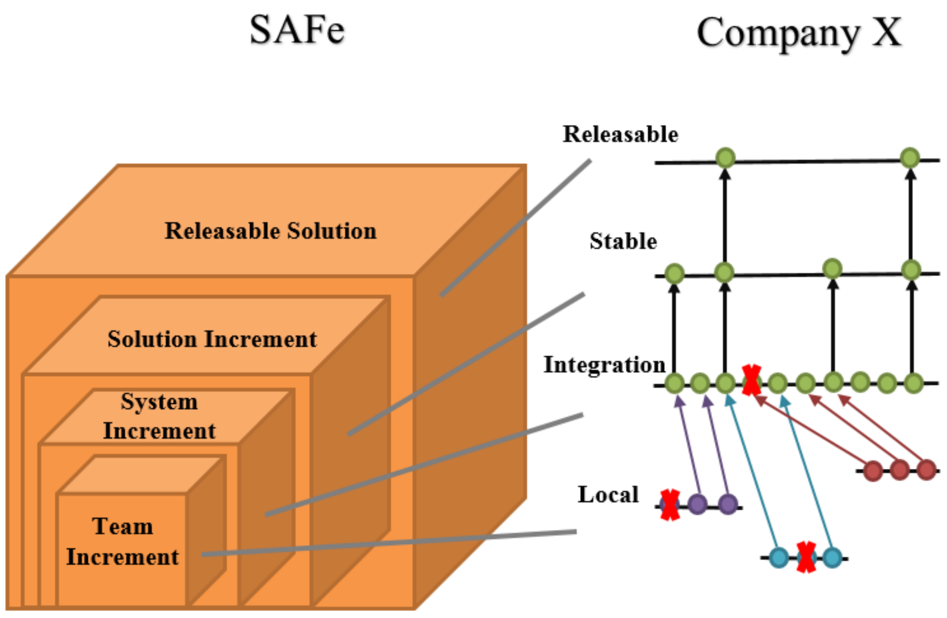


Figure 2.15: Connection to SAFe's test process

Chapter 3

Methodology

This chapter contains information about the methodology that was used to retrieve the results of this thesis work.

3.1 Approach

The approach of this thesis work have been a sequential approach which contained four phases which is demonstrated in Figure 3.1. The four phases was Theory, Preparation, Data collection and Compiling results. The report have been written during all of the phases in this thesis.

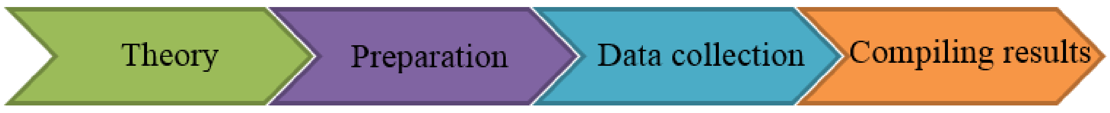


Figure 3.1: The process of the thesis work

3.1.1 Theory

The first phase of the thesis work was theory which consisted of literature study and learning about SAFe. The aim of this phase was to find useful literature about the waterfall and agile process models, and about SAFe. The main purpose of this phase was to learn and understand SAFe so that the author of this thesis had an understanding of SAFe and could ask the right questions in the interviews. Another goal with this phase was to see if similar evaluations about SAFe have been made before. The study resulted in information about SAFe that is described in Section 2.4 and also in similar evaluations that are mentioned in Section 1.5 related work. During this phase all literature about SAFe was found on the SAFe homepage [13]. Literature about Agile and Waterfall process models, and related work was searched on using several databases. Examples of the databases that was used are the Lund University digital library archive search tool LUBsearch and google digital library search tool google scholar. Initially searches was done on some key words such as SAFe, scaling agile and agile test process. Then references on the relevant articles was inspected in order to find more relevant literature.

3.1.2 Preparation

Next phase after the Theory was the Preparation phase. The main purpose of this phase was to gather some interview questions from involved persons at Company X and to find companies to interview. A focus group that consisted of 6 involved persons at Company X, was held to gather some interview questions. More information about focus groups can be found in Section 3.2.2. The involved persons consisted of persons that are involved in the SAFe implementation and persons that work within test at Company X. At the same time, research about companies that uses SAFe was conducted. The companies was found on the SAFe homepage and from different consultants that had worked with implementing SAFe on companies. The companies that was chosen have implemented SAFe and are among others located in Sweden, Denmark or Finland. Meetings was then booked with persons from the different companies. This phase resulted in interview questions and 17 companies to interview.

3.1.3 Data Collection

After the Preparation phase the different interviews were conducted and a survey was sent out. 20 persons from 17 different companies were interviewed about their experience of implementing SAFe. The purpose of this phase was to get answers on the questions and to hear about companies experiences of implementing SAFe. The focus group came up with a lot of questions so the author of this thesis had to prioritize the questions. The questions that got high priority was the questions that were related to the aim of this thesis work and that were interesting. The majority of the interviews were individual interviews but there were some group interviews and a survey. Information about the different interview techniques that was used and about the survey can be found in Section 3.2. Many of the individual interviews were held by phone since many of the companies that were interviewed are not located near the author of this thesis. The interviews were recorded with permission from the interviewee. The records made it easier for the author of this thesis to remember what was said in each interview. The interviews took between 1 to 2 hours depending on how long time the interviewee could set aside for an interview. Unfortunately the survey gave no results since there were too few persons that answered on it. It is hard to get people to take the time and answer a survey. One of the persons that work at

Company X was responsible for sending out the survey to people that work with test. This could be one of the reason why the survey gave no result since the author of this thesis could not effect who the survey was sent to. This phase resulted in answers on the most important questions and an understanding on how different companies have implemented SAFe.

3.1.4 Compiling results

The last phase was the Compiling results phase where the results from the interviews were compiled. The purpose of this phase was to get a summary and a conclusion on how the different companies have implemented SAFe and what challenges they had during the implementation. The records and notes were reviewed and written down and then a compilation was made of all the companies' results which can be found in Chapter 4. When doing the compilation the author of this thesis went through each question and reviewed all the companies' answers on that question to see which the most frequent answers were. The author of this thesis then saw some patterns in the answers that some companies said which lead to two different analyzes. The first analysis was conducted to see if there were any connections between the amount of persons involved in the implementation and how the companies have answered. First the author of this thesis grouped the companies according to amount of people that are involved in SAFe. Then a review of the answers in each group was made to see which answer that was most frequent. The grouping is explained in Section 4.1. The second analysis was to see if there were any connections between the companies' background and the companies' answers. The background in this context means if they had an agile, waterfall or any other process model before the implementation. The author of this thesis saw patterns on some of the questions where the answers differed a lot between companies with different backgrounds and amount of people involved in SAFe. After the compilation of the results an analysis and discussion of the results were made which can be found in Chapter 5. The overall analysis resulted in some guidelines for companies that are going to implement SAFe. This phase resulted in a compilation of the results and an overall understanding of the effect the amount of persons and the background of a company has on the implementation.

3.2 Data collection techniques

This section contains information about the different techniques that have been used or been considered when collecting information from the selected companies. There are different types of interviews: Fully structured, Semi-structured and Unstructured. In a fully structured interview the interviewer only asks predetermined question that is in a pre-set order. The person that is being interviewed can ask some response questions if need. In a semi structured interview there are some predetermined questions that might not be in a pre-set order. The questions can be changed, explanations can be given and new questions can be added if the interviewer feels that it is necessary. Unstructured interview have no predetermined questions but the interviewer has a general area that they like to discuss. The interviews in this thesis work were semi-structured since the author of this thesis wanted to have some predefined questions but also have the opportunity to change a question or add new questions if necessary.

There are different types of questions that you can ask at an interview: closed, open and scale. Closed questions forces the interviewee to choose from one or more fixed alternatives. Open questions have no alternatives and no restrictions on the content of the reply. Scale questions are questions or statements where the interviewee chooses the answer on a scale. In this thesis, the type open questions have been used so that the interviewee can answer as freely as possible and the answers gets as specific as possible [10].

3.2.1 Interview

In this thesis, the semi structured interview technique have been used for the individual and the group interviews. The structure of a semi structured interview can be to first have an introduction where the interviewer explains the reason for the interview. Then the topic of the interview is presented and the interviewer asks the questions. When asking the questions the interviewer can have some associated prompts but it is not necessary. After the questions have been answered the interviewer say some closing comments and then the interview is over.

It is very common that individual interviews are held by phone [10]. There are many advantages with having telephone interviews such as high response rate, correction of obvious misunderstandings, possible use of probes, etc. There is a lack of visual cues which may cause problems in interpretation however it is a good alternative when there is a geographical distance between the interviewer and the interviewee [10]. The majority of the individual interviews in this thesis have been held by phone due to geographical distance.

3.2.2 Focus group

A focus group is a group interview on a specific topic. In this thesis the questions for the interviews have been gathered by using a focus group which in this case was an unstructured group interview. The author of this thesis chose a focus group since the persons in the focus group can be inspired from each other's questions which will lead to more questions. It can also lead to a discussion which will give better and more precise questions but the focus group is guided by the interviewer. A focus group can either be homogeneous, which means that the group has common background, position or experience, or heterogeneous, which means that the group differs in those areas. In this thesis the group was a homogeneous group since the focus was about what questions Company X wanted to have answers on. The author of this thesis was the moderator which is the person that runs the focus group. The moderator took notes about the different questions that the focus groups wanted to have answers on [10]. It resulted in many questions so the author of this thesis had to prioritize the questions. The questions that were asked in the interviews are shown in Appendix A.

3.2.3 Survey

A survey is carried out by an interviewer which is the person that wants you to answer the questions. There are different kinds of surveys such as postal survey, telephone survey and where the interviewer is knocking on your front door or stops you in the street. A postal survey is a survey that is sent by email and a telephone survey is a survey that is held by phone. A postal survey have been used in this thesis work since it was targeting a specific group. The specific group consisted of people working with software testing. It is unfortunately very common with low response rate on surveys. Surveys are used everywhere and for many different purposes so it is likely that you will have an understanding of what the term means. This wide range of studies that have been labeled as surveys have made it difficult to give a concise definition of a survey. Some typical features of a survey are:

- Uses a fixed and quantitative design.
- In a standardized form collects a small amount of data from a large number of individuals.
- Makes a selection of representative individuals from known populations [11].

In this thesis work an electronic survey that consisted of questions about test and quality in SAFe was used. Google forms was used for the survey and the questions were the same as the test and quality questions that were used in the interviews, see appendix A.

3.3 Source criticism

The majority of the literature that have been used in this thesis have been found on the internet. All literature about SAFe have been found on the SAFe homepage which can be both positive and negative. The positive aspect is that the information comes from the creators of the framework who knows the most about the framework. The creators may not see any negative aspects about their framework and even if they did, they would not write about is since they want companies to use their framework.

Literature about waterfall and agile process models have been found in different places such as books and internet. The webpage that lists some of the disadvantages and advantages about agile process model is a website that offers study material that is needed to pass the ISTQB (International Software Testing Qualifications Board) foundation level certification. This website is made by a person that have the certificate that wants to help others when studying. He does not want to sell anything but it is still a second part information so some of the information could be his own thoughts and not facts. The literature for related work are of different reliability but the majority are from articles that have been published. The articles that have been published can still be angled since the writer want people to read the article. One of the sources that have been used for related work is from a website that is almost like a blog where one person write about his experience of SAFe. This is not as reliable as the other sources since it is one person's experience and thoughts.

The books that have been used both for the agile and waterfall process models, and for the interview techniques are of great reliability. However a writer can always angel the content so it is good to look at several sources to see that more than one source has the same facts.

The interviews have been carried out on different persons at different companies. The information that is given from each interviewee is very personal and may not reflect the whole company's view on SAFe. Different roles have different experiences of SAFe so to get a more accurate result the interviewer could have interviewed more than one person from each company and persons with different roles. When doing this source criticism [12] have been used for help.

Chapter 4

Result

This chapter contains the most frequent answers that the companies have answered on different questions. All the answers can be found in appendix A. This chapter also contains an analysis of the results to see if there are some similarities between the characteristics of the companies and the answers they have said.

4.1 Amount of people involved in SAFe

The interviews have been conducted on 20 persons from 17 different companies and consultant firms. The interviewees had very different roles but the majority had some management position. Few of the interviewees were testers, developers, scrum masters or product owners. All the companies that have been interviewed were in different sizes and have different backgrounds. The companies have been divided into four groups when it comes to the amount of persons that are using SAFe:

- Small: Companies with less than 100 persons.
- Medium: Companies that has between 100 and 500 persons.
- Large: Companies with over 500 persons.
- Consultants: Consultant firms that have helped other companies with implementing SAFe.

This section contains the answers from all the companies but they are grouped in the different groups. This makes it possible to see if there are any similarities between the answers and the amount of persons that are involved in SAFe.

4.1.1 The companies

There are three companies in Small, four in Consultants and five in both Medium and Large. This is described in Table 4.1. The Consultants have been involved in a handful companies implementations of SAFe. Their experience is that it is most common that companies have between 100-500 persons that are using SAFe. Companies often start small and expands their implementation when they see that it works.

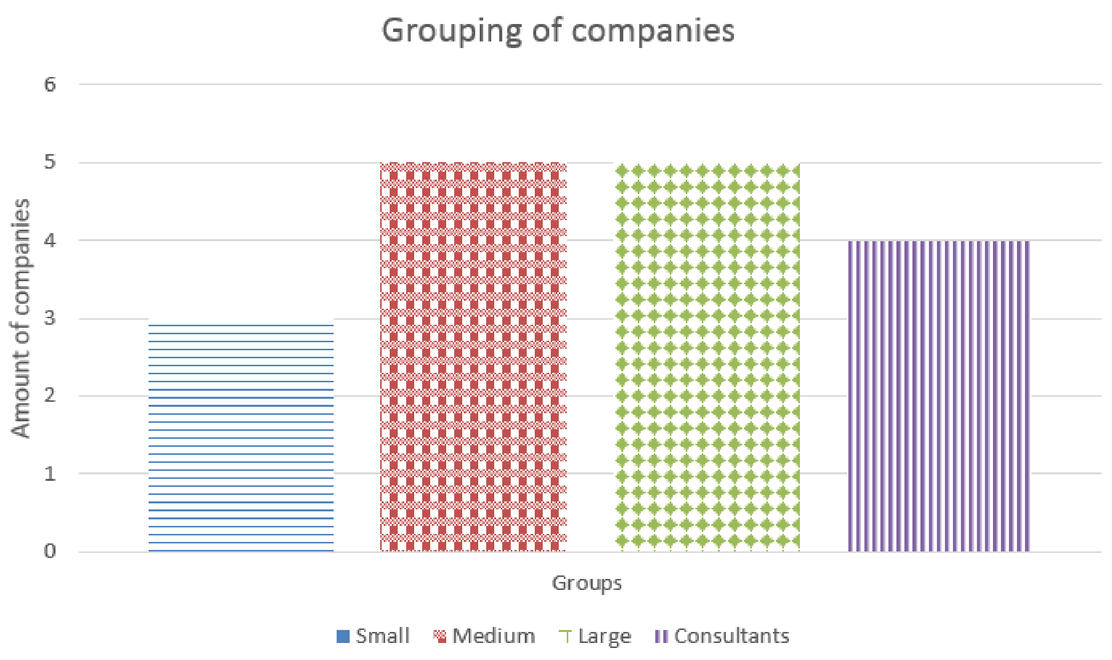


Figure 4.1: Grouping of the companies

The majority of the companies are located internationally but there are some exceptions. One company is located mainly in one location but has one outsourced part that is located in another location. Two of the companies have all the persons that are using SAFe located at the same place and two have them located in the northern of Europe. The consultants' experiences are that the majority are located internationally.

Suppliers

The majority of the companies are responsible for everything themselves or have only some small parts outsourced to suppliers. Large have suppliers that are differently involved in SAFe. Some of the companies have more than one part outsourced to suppliers and some of them just have one part that is outsourced to suppliers. The companies that uses many suppliers said that it is a challenge because the suppliers have to work in the same way as them which is not always the case. There is only one company that has everything outsourced to suppliers. Medium are responsible for all or most of the development that is involved in SAFe. Some of the companies have one part that is outsourced or have outsourced during periods. Small are responsible for everything themselves except for one company that has one product that is outsourced to suppliers.

Three out of four consultants said that it is very common that companies have suppliers when using SAFe. All of the consultants agree that it is not so successful to have suppliers and that companies often bring in help because they have suppliers. The consultant that said that it is most common that companies does not have suppliers, agreed that it goes faster when companies does not relay on other companies. Other negative aspects of having suppliers are: lower quality, the company has not the same overview of what is done, culture clashes and it requires a lot of initial investments both in time and money to make this work. One consultant said that smart companies realize that it is not worth having suppliers even if it is cheaper because the quality gets a lot worse. So the consultants' tips companies to take home the development to the company since the more complex operation the harder it gets.

Amount of products/systems involved in SAFe

The companies have answered differently when it comes to amount of products/systems that are involved in SAFe. Some companies have answered in amount of systems and others have answered in amount of products. The dependency is that a system consist of several products. Large has different amount of products/systems that are involved in SAFe. 2 of the 5 companies have answered that they have over 100 products that are using SAFe. The other 3 have either 5-10 systems or they have around 50 systems. Medium is also divided in the amount of products or systems. One company answered that they have 1 system and the other companies answered in amount of products. 2 of the companies said that they have 50-70 products and the other 2 have 4-5 products. Small has all answered in amount of products and they have between 4-15 products.

4.1.2 SAFe implementation in general

This section contains information about how the companies have answered on questions about the SAFe implementation in general.

The companies background

Large had a mix of different backgrounds. 3 of the companies have a waterfall background, one have a Lean background and one have an RUP background. Some of the companies that have a waterfall background had some teams that worked agile before the implementation. Medium also has a mix where 3 of the companies have an agile background and 2 have a waterfall background. One of the companies that had agile background were working RUP on a higher level before implementing SAFe. In Small all have an agile background. 2 of the consultants said that it was most common with waterfall and two said agile. This is illustrated in Table 4.2.

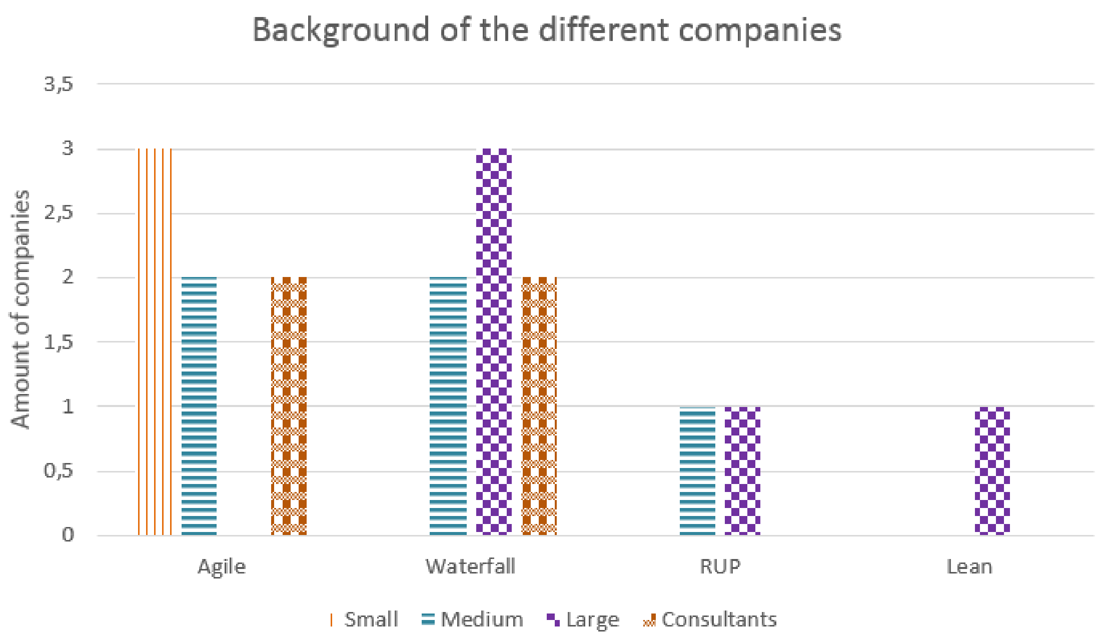


Figure 4.2: Background of the companies

General challenges

The companies that have implemented SAFe have had some challenges when doing the implementation. The challenges differs a lot from the different companies but some similarities are there in the different groups. The three most general challenges in Large are shown in Table 4.3. The challenges are to convince workers that are resistant and to get continuous integration and test automation in place. The companies have also solved these challenges in different ways. Table 4.4 shows the three most general ways of solving the challenges in Large. The most general ways are to bring in coaches that helps them, train persons in their roles and to educate persons.

In Medium, the most general challenges are shown in Table 4.5. They had challenges with workers going back to the old way of working when they were stressed, persons being very confused about their new roles and to get everybody to understand that the teams take the decisions and have the responsibility. Table 4.6 shows the three most general ways of solving the challenges in Medium. It is most general to bring in coaches for help but also to have retrospectives which means that they learn by their own mistakes. The third way is to have patience because it takes time to do the implementation.

The three most general challenges in Small are shown in Table 4.7. The challenges are that workers had some challenges with understanding what was required of their new roles, to find and change tools, and that stakeholders were worried that they were going to lose agility because the planning is far ahead. The most general ways of solving challenges in Small are shown in Table 4.8. Retrospective, to get feedback faster from the teams and customers and to have patience are the most general ways in Small.

The three most general challenges that include all the groups are to change the tools, that the teams easily go back to the old way of working and that some roles had difficulties to let go of their responsibilities. The consultants had very different experiences with which challenges companies have. Two challenges were general between the consultants where one is that management had to understand what SAFe was about and that they want to do the implementation. The second challenge is to know which changes the company needs to do since they do not have to change things that already work. The consultants are agreed on that the most general way of solving the challenges are to educate persons and bring in coaches that help them.

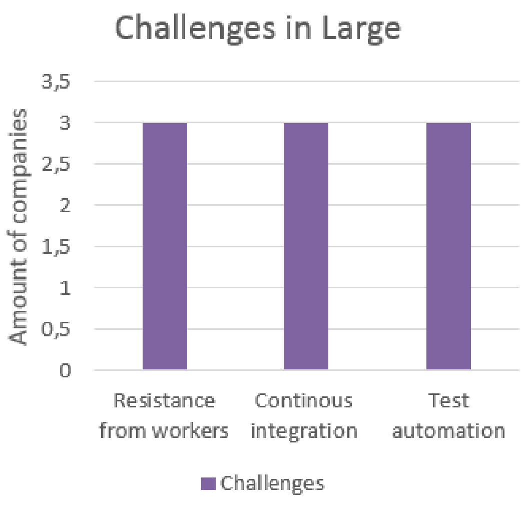


Figure 4.3: Challenges in Large

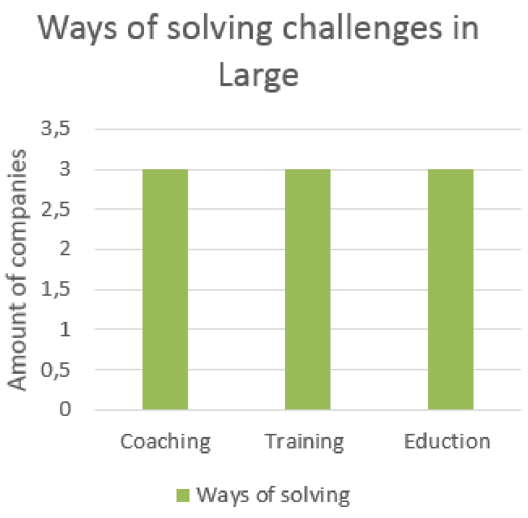


Figure 4.4: Ways of solving in Large

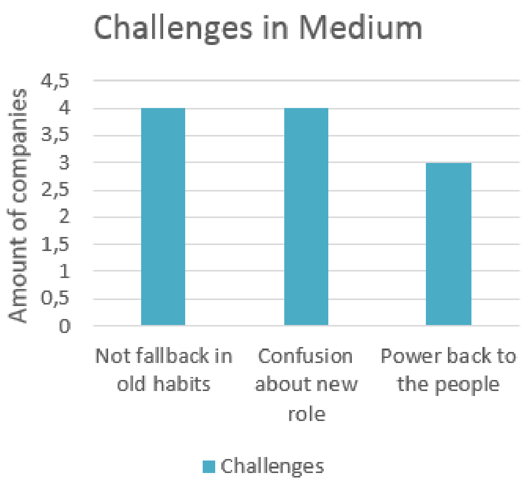


Figure 4.5: Challenges in Medium

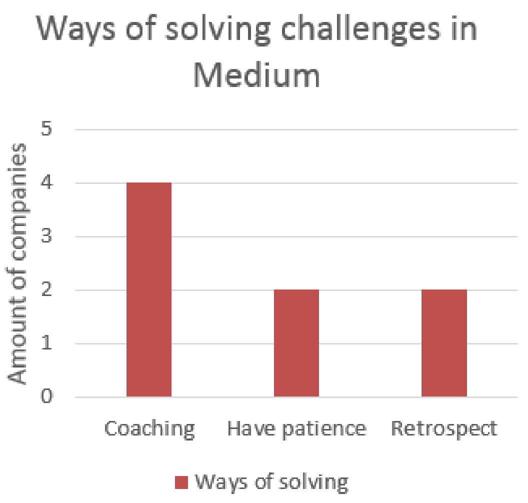


Figure 4.6: Ways of solving in Medium

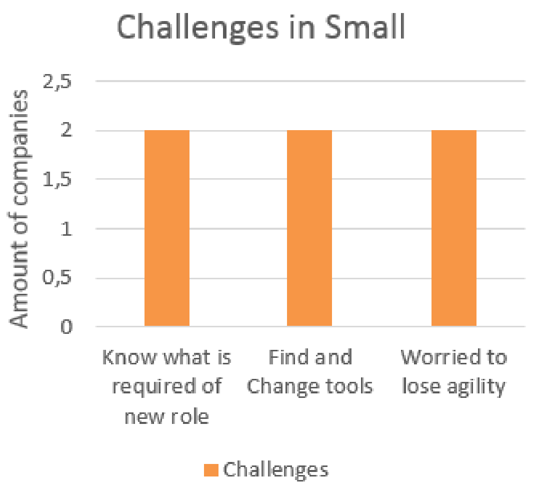


Figure 4.7: Challenges in Small

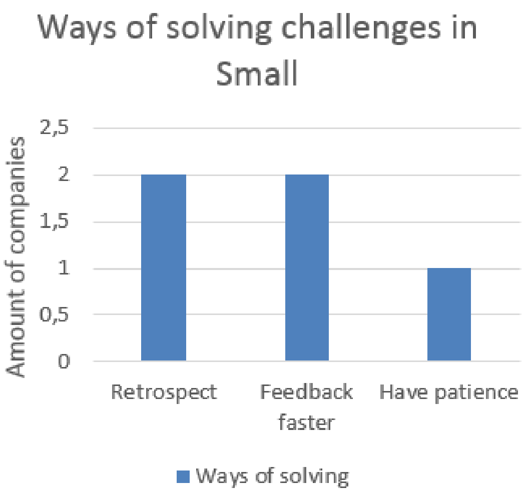


Figure 4.8: Ways of solving in Small

Implementation

All companies implemented SAFe in parts of the organization but many have the goal to implement in the whole organization. The consultants agree with this and said that most companies start with implementing in one part of the organization and when they see that the process works they expand to other parts.

The companies have chosen SAFe for many different reasons and the three most general reasons for each group are described in Table 4.9 to 4.11. For Large one reason is that they have done it in another part of the organization before so it was an inspiration for other parts. Two other reasons are that they do not know why they chose SAFe and that they realized when they had scaled that they worked very similar to SAFe so they took inspiration from SAFe after that. Medium had some other reasons for choosing SAFe such as that the initiative came from the teams, that SAFe has a good structure on doing things and that there is good documentation on SAFe. Small had very different reasons for choosing SAFe. None of the companies in Small had the same reason so Table 4.11 shows the three reasons that each company said. The reasons are that it is the most famous framework for scaling agile, that the management decided that they should use it and that they got a recommendation to use SAFe. There were some reasons that companies from all the groups said. The three most general reasons from all the groups are that it was a management decision, the most described framework and that it gives a good structure which big companies need. The consultant mentioned the same reasons as the groups and they said that SAFe is a good starting point so that the companies does it right from the beginning.

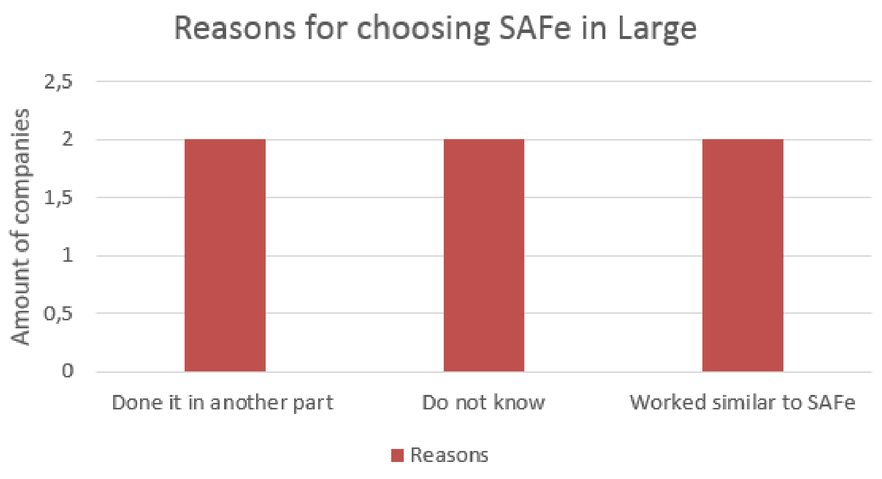


Figure 4.9: Reasons for choosing SAFe in Large

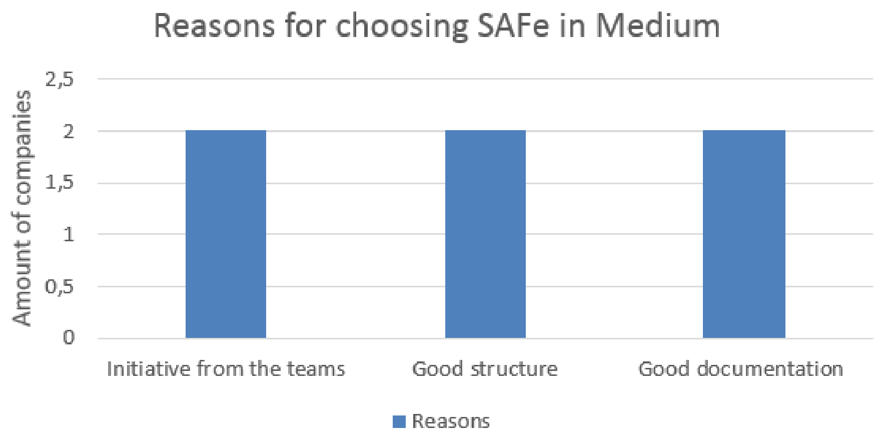


Figure 4.10: Reasons for choosing SAFe in Medium

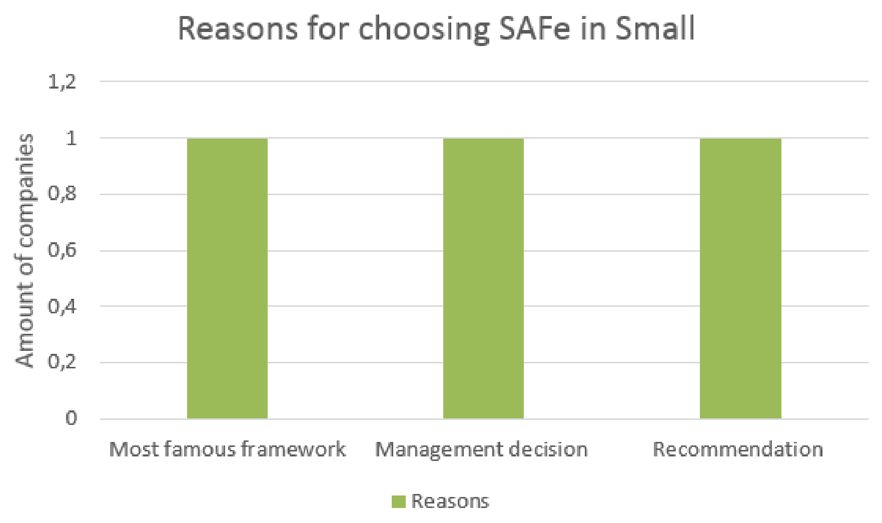


Figure 4.11: Reasons for choosing SAFe in Small

The companies have had different goals with implementing SAFe. The most general goals for all the groups are shown in Table 4.12. They really wanted to increase in productivity so that they could deliver faster and more to the customer. To increase quality and get better transparency were also two important goals. Medium and Small also wanted to have better cooperation between the teams and get a better structure. The consultants said the same goals as the companies did. They said that the most general goals are that companies want to have better efficiency and increase quality.

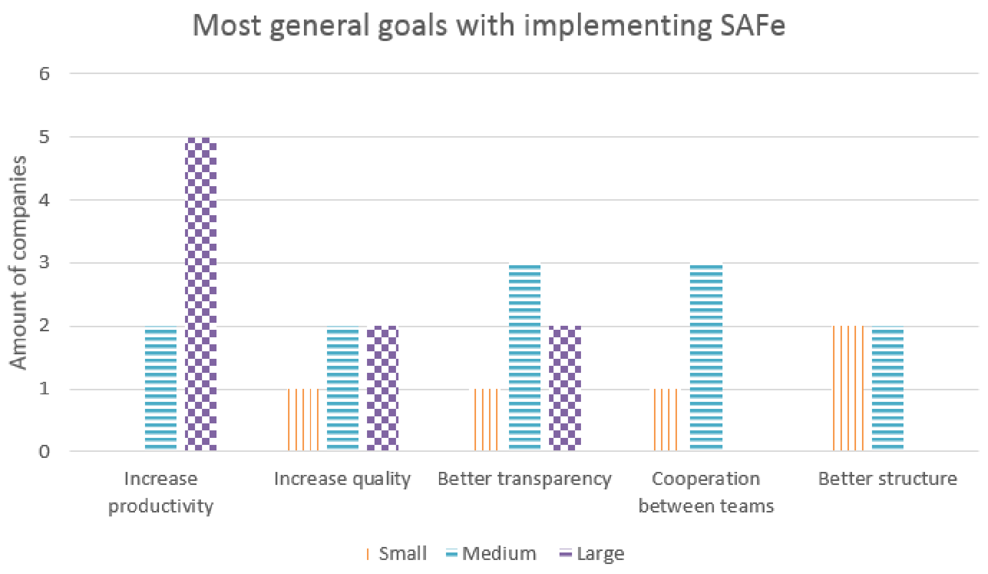


Figure 4.12: Most general goals with implementing SAFe

An important part when implementing a new framework is to follow up the implementation. The companies should measure and compare to see if the implementation have fulfilled their goals. The groups have answered differently on this aspect. Large had two ways of following up the implementation that more than one company mentioned. In Table 4.13 can be seen three ways of following up the implementation in Large. To use different KPI and that companies do not have any dedicated measurements are the two most general ways. The KPI's can be lead time, amount of customer complaints, delays and so on. Retrospective on each level was a third way of doing follow up on the implementation in Large. Medium's most general ways of doing follow ups are shown in Table 4.14. They measure employee satisfaction, how much of the work that was planned in a PI was really done, and that the companies do not do any follow up. The companies in Small had many different ways of doing follow up and three of them are shown in Table 4.15. As the other groups the companies in Small also said that they did not do any dedicated measurements. They also said that they had a hard time to compare since they measured different things before the implementation and that they had a feeling that they have fulfilled their goals. The most general answer from the companies in all the groups was that they did not have any dedicated measurements for following up their implementation. The consultants said the same ways as the companies and that it is very common that companies have no relevant measurements. All the consultants agreed that it is very good if the companies do some measurements and compare so that they see that the implementation were worth doing.

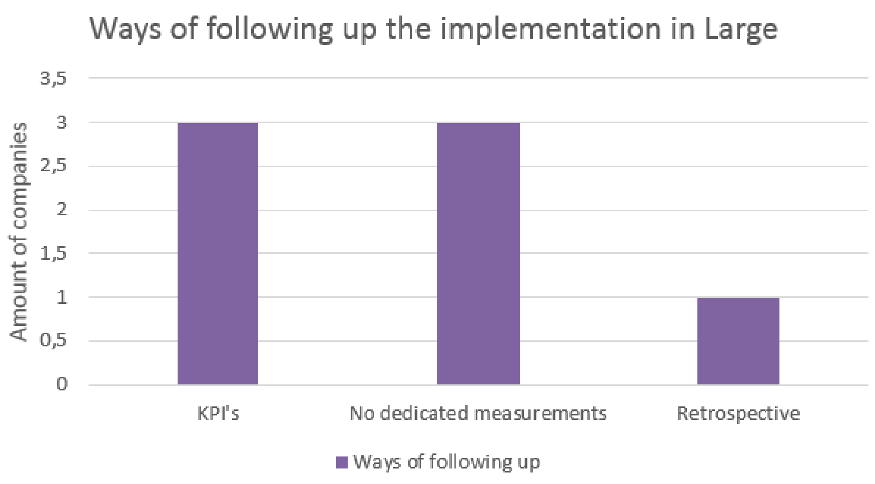


Figure 4.13: Ways of follow up in Large

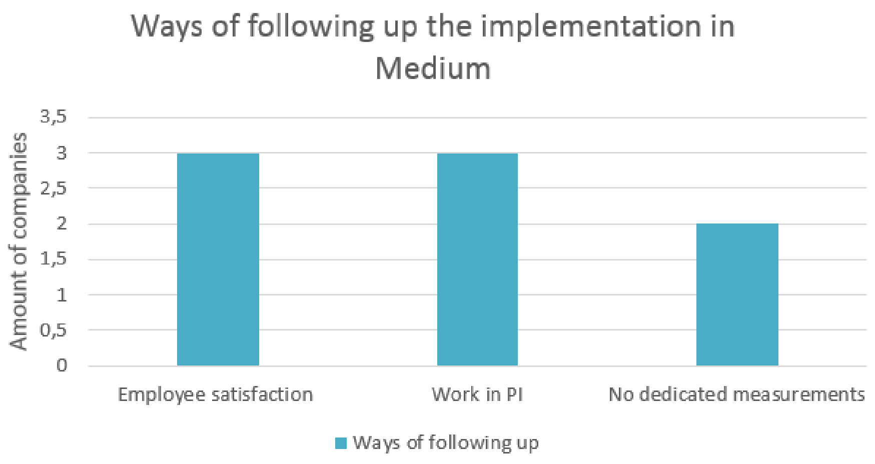


Figure 4.14: Ways of follow up in Medium

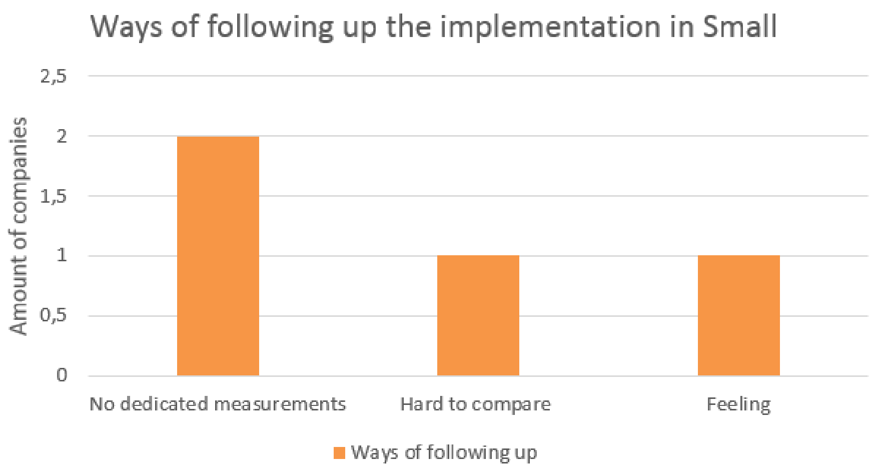


Figure 4.15: Ways of follow up in Small

When doing the implementation some companies may have to do some organizational changes. All the groups answered similar on which changes they had to make. Table 4.16 shows the companies different changes. Many of the companies have had to do cross-functional teams but also to reorganize the organization completely and to give persons new roles. A cross-functional team is a team that consist of people with different functional expertise, which works toward a common goal. An example is a team that consist of developers, testers, UX people and architects. Some companies also have had the change that the teams are responsible for everything and some companies did changes before they did the implementation. One general change that all the groups said was that they had some shared teams that helped the other teams. The consultants said that it depends on which changes the companies have done before but it is very common that companies have to do some changes. They all mention that they have to do cross-functional teams and that persons get new roles.

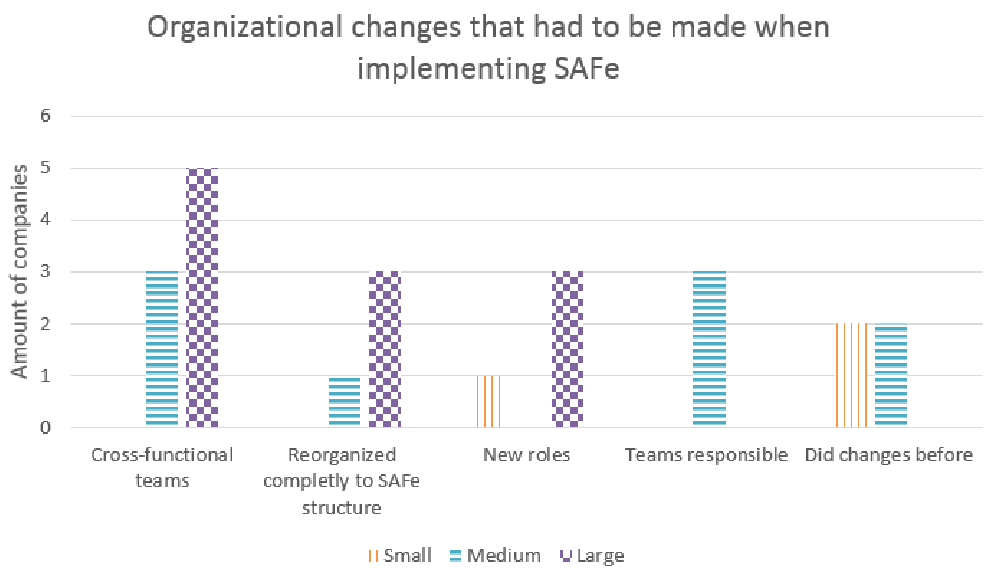


Figure 4.16: Organizational changes when implementing SAFe

When a company is implementing SAFe they need to have cross functional teams. If a company does not have that before the implementation they have to do it when implementing SAFe. Table 4.17 shows the different roles that the companies in the groups have. Not all companies have testers in the teams so then the developers are responsible for doing the tests as well. Some of the companies have a separate test team that helps the developers with test when they have too much to do. The consultants said the same roles as the companies did. They said that it is very good if the testers are in the teams but that some companies does not have that.

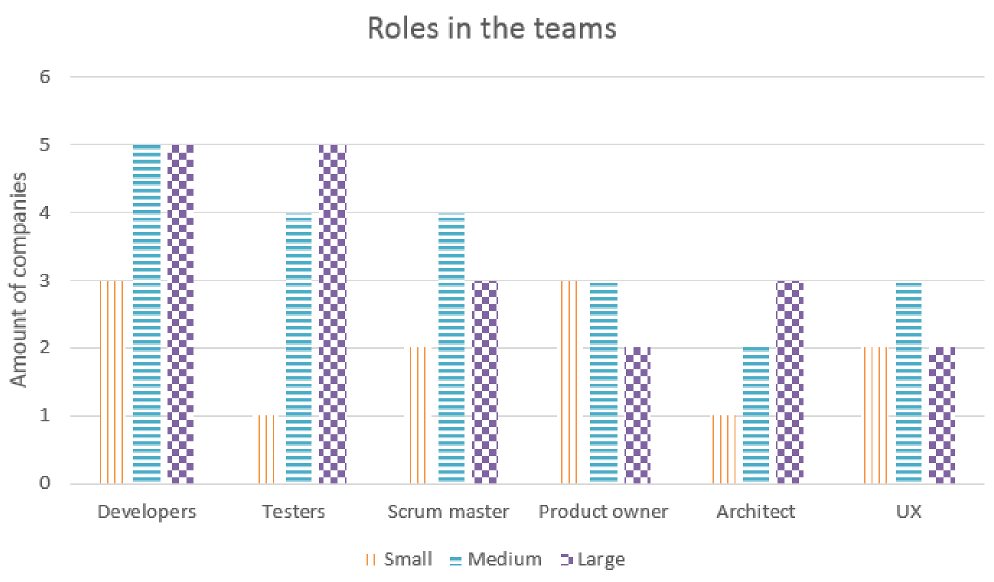


Figure 4.17: Roles in the teams

A company will never be completely finished with implementing SAFe since it takes a long time to implement and that the company can always improve their way of working. It has taken differently long time for the companies in this study to get SAFe to work. The companies have answered in different ways, some answered in how long time it took from decision to first PI, others answered how long time it took to get something that was structured and some answered in both ways. In Table 4.18 is how long time it took for the companies in all the groups.

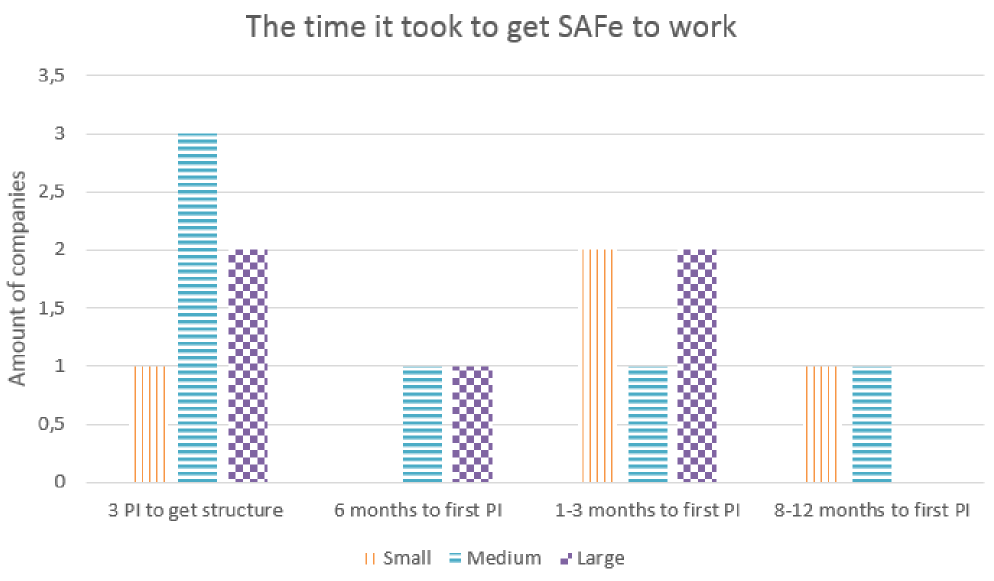


Figure 4.18: Time it took to get SAFe to work

4.1.3 SAFe implementation, test and quality aspects

This section contains information about how the companies have answered on questions about test and quality in the SAFe implementation.

Challenges and changes

When implementing SAFe companies have some challenges with test and quality. Table 4.19 shows the four most general challenges that the companies had in test and quality. Many companies had challenges with automating tests and to get continuous integration in place. Some companies in Medium had challenges with getting the team to understand that the whole team is responsible for test and quality. The developers did not take responsibility for the test and quality and put it on the testers. Coordination between teams was also a challenge for some companies. The consultants agreed that the companies have challenges with automating tests and that the whole team takes responsible for test and quality.

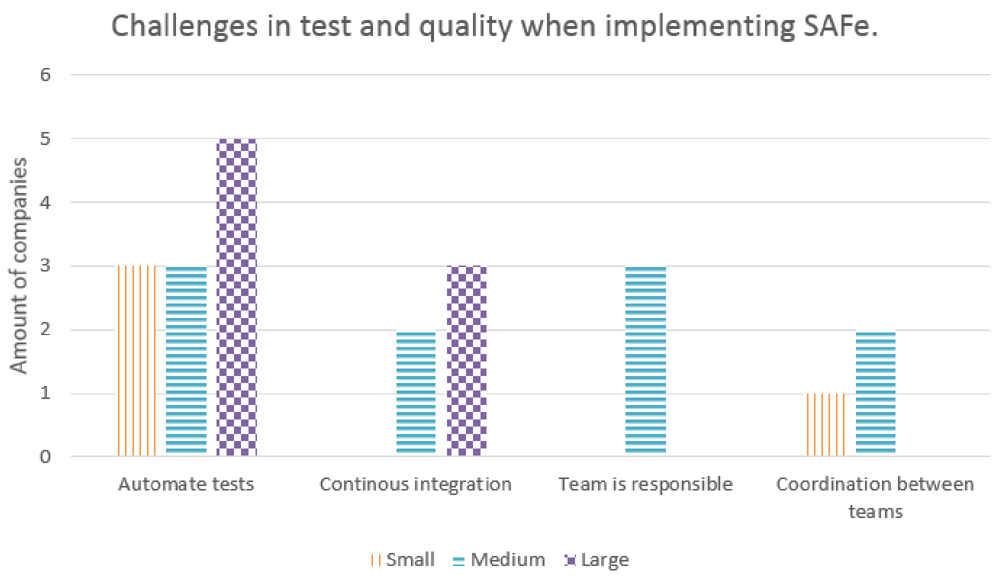


Figure 4.19: Challenges in test and quality when implementing SAFe

These challenges have been solved in many different ways by the companies. Table 4.20 shows the most general ways of solving the challenges for each group. There were a big difference in the ways of solving the challenges. Many brought in consultants that coached them with their challenges and got educated in these areas. To have test communities and get a DevOps team was other ways of solving challenges. The consultants agree with the ways of solving but they emphasize that companies should automate as many tests as possible because then they get better quality.

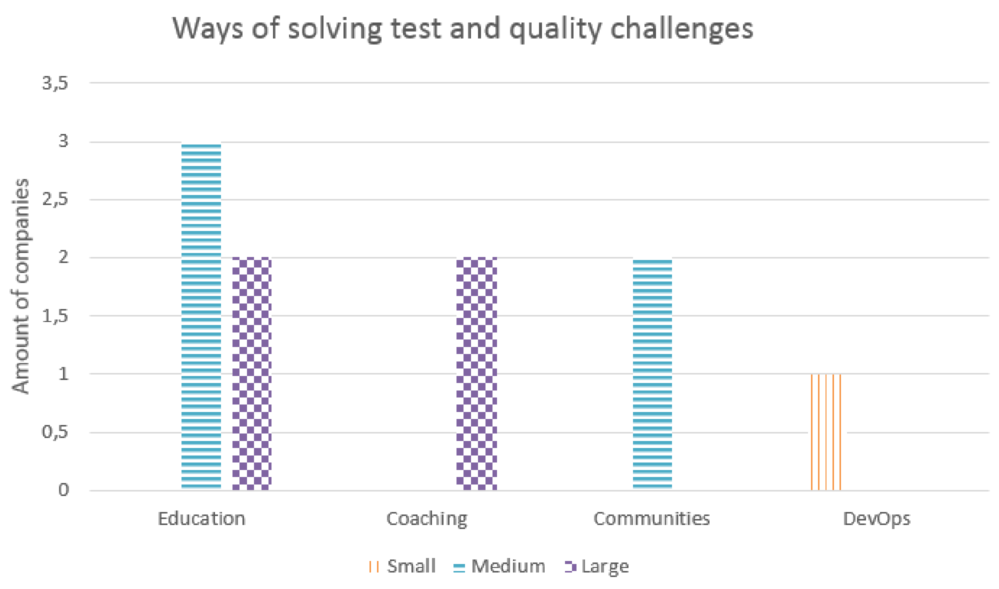


Figure 4.20: Ways of solving test and quality challenges

The companies have to change their test process when implementing SAFe. Some companies have to do more changes than others. Table 4.21 shows the most general changes that the companies in the different groups have made. There are four general changes that the companies had to do. Some of the companies had to change from having one test department that did all the tests to have the tests in the teams. Many of the companies change to have the test in parallel to the development instead of having it in the end of a release and to have more automated tests instead of manual tests. Another change was to have definition of done on each level in SAFe. The consultants said that it depends on what changes the companies have done before the implementation. The most general change between all companies is to have more automated tests.

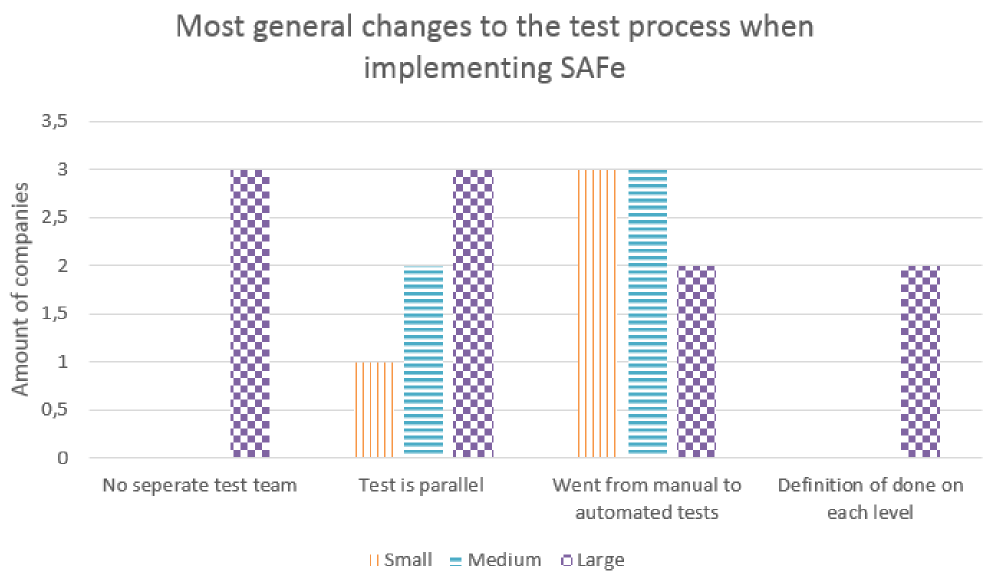


Figure 4.21: Most general changes to the test process

It is important that the companies control so that the changes gives some good effect. Table 4.22shows the most general effects that the changes to the test process had for the companies in the different groups. There are three general effects of changing the test process. The companies find defects/bugs earlier and much faster now. They also have less defects/bugs in the production so the system is more stable and some companies have not looked into this yet. The consultants confirms that companies finds defects/bug much earlier and faster when they change the test process.

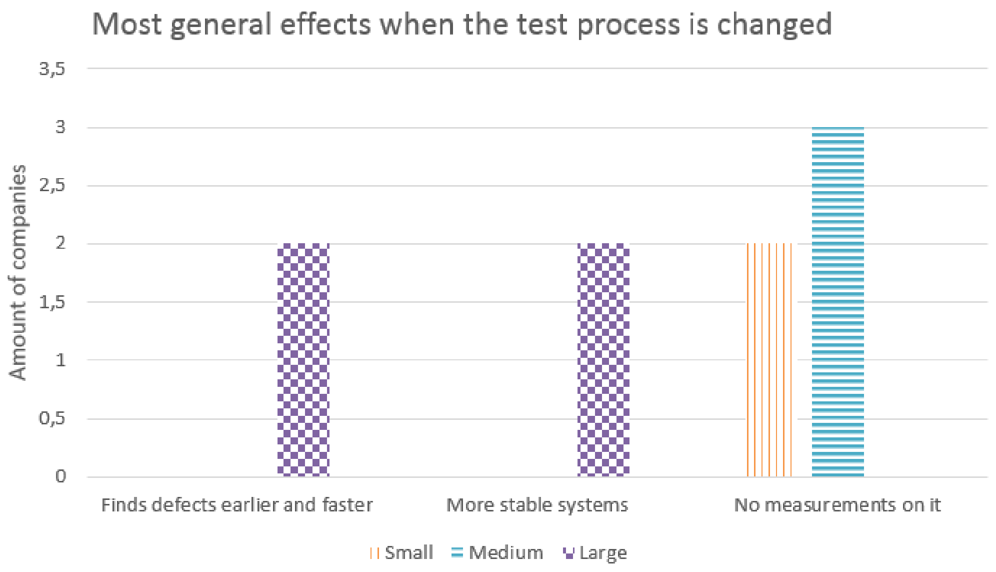


Figure 4.22: General effects when the test process is changed

General facts about test

It is very common that companies outsource different parts to suppliers such as test. The majority of the companies that have been interviewed have all tests in-house. Table 4.23 shows if the different groups have test outsourced. The companies that have tests mostly in-house, have just one part or one kind of test outsourced and the company that have partly outsourced have many parts outsourced. The consultant said that it is most general that companies have their tests in-house and that the companies that have it outsourced bring the tests home.

All the companies that were interviewed did some measurements on test and quality. It differs a lot in the amount of metrics and if it is decided that everybody should measure the metrics or not. Table 4.24 shows the most general metrics that the different groups measure when it comes to test and quality. The most general metrics are amount of defects, defects found by the customer, defects that are waiting on, defects that gets fixed, defects that are found by the team and code coverage. The consultants said that it depends a lot on what kind of product it is that the company develops but that it is very important to get customer feedback as early as possible.

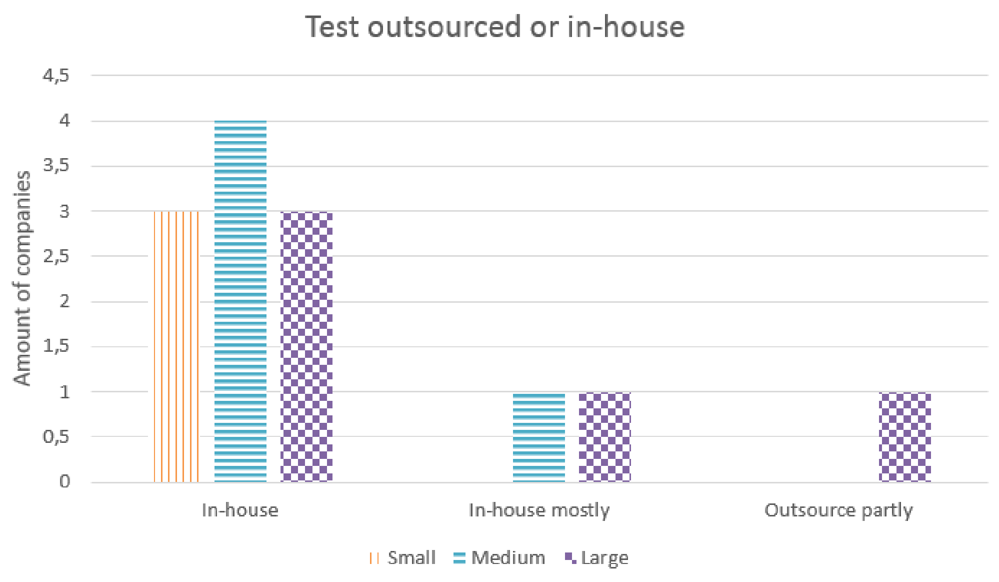


Figure 4.23: Test outsourced or in-house

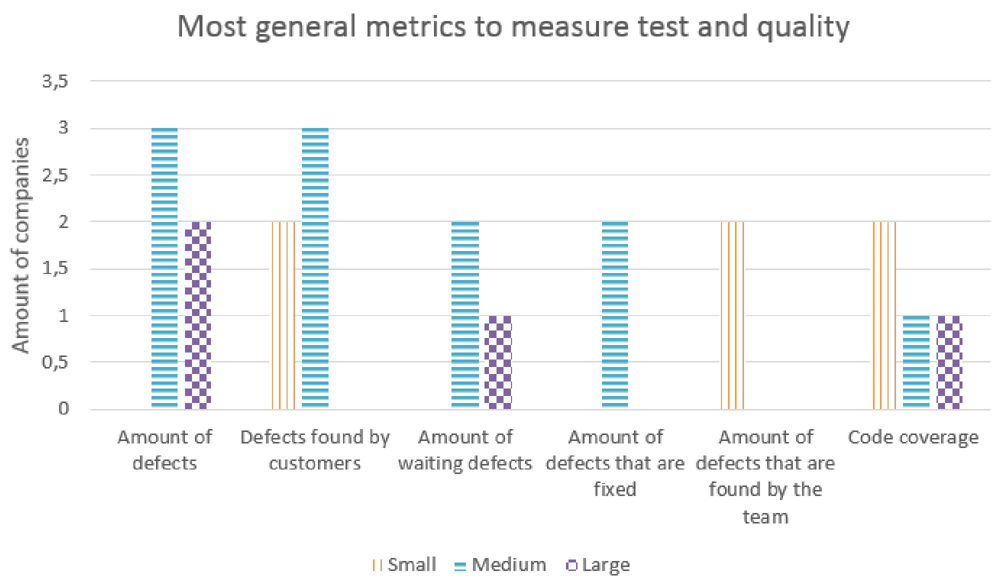


Figure 4.24: General metrics to measure test and quality

The majority of the companies had to change test tools when implementing SAFe. Some of the companies had done changes before the implementation. Table 4.25 shows the most general tools that the companies use for test. The three most general test tools to use is Jenkins, that the test tools depends on what language they are developing in and HP-ALM. The consultants also mentioned Jenkins and that it depends on what language they are developing in.

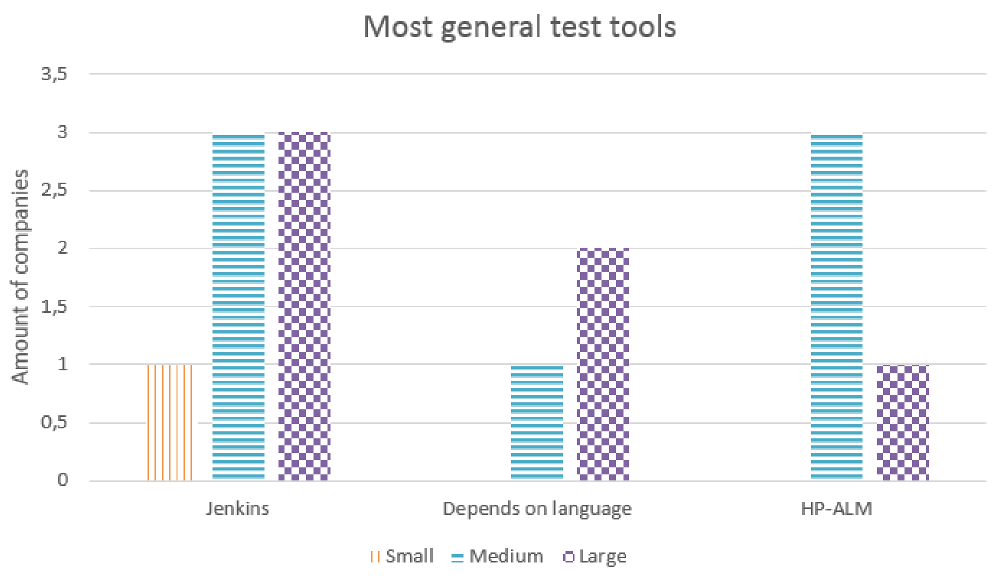


Figure 4.25: Most general test tools

4.2 Waterfall or agile background

The companies that have an agile background have to do less changes in their process when implementing SAFe since they already have some things in place. Two examples of things that are already in place are that the teams are working in small iterations and that they have cross-functional teams. It is more common that the companies that have an agile background does not have any dedicated testers in their teams. Instead they have developers that also do the tests. None of the companies with a waterfall background have that structure in their teams. When it comes to the amount of time it took to get SAFe to work, there were no difference between companies with different background. Some challenges that the companies with a waterfall background have which the companies with an agile background does not have are:

- Resistant among co-workers. The co-workers do not like to work in cross-functional teams and to work in iterations. Companies with an agile background does not have so much of this challenges since they already work like that.
- Managers do not want to let go of their responsibility. Many of the managers on the companies that have an agile background have already embraced this change since the teams are responsible in an agile process model.
- Get the team to realize that they have the whole responsibility for quality. The cross-functional teams have the whole responsibility of the quality together. Teams that already work agile knows this but the companies that worked waterfall before had problems that the teams thought that the testers are responsible for the quality.

The companies that had a waterfall background had to do more organizational changes than the companies with an agile background. This is because the companies with an agile background had already some of the levels in SAFe in place.

Chapter 5

Discussion and conclusion

This chapter contains a discussion and a conclusion of the results in this thesis. It also contains some improvements and tips on future work that can be done in this area.

5.1 Discussion

When reviewing the result it is clear that some challenges, approaches and changes are general for many of the companies. To successfully implement and use SAFe, companies should not have too much of their development and testing outsourced. It requires a lot more work and time to make SAFe work with suppliers. The companies had different backgrounds when implementing SAFe. Companies that had an agile background did not have to do as many changes as the companies with a waterfall background. It could be a good idea to start with implementing an agile method on a team level and then implement SAFe when they see results of working in an agile process. The companies chose SAFe for different reasons such as SAFe has a lot of documentation and it was management that decided it. The companies that had fewer people involved in SAFe had more knowledge about why they chose SAFe than the companies with many people. This might be because of the size which means that the more people that are involved the more people do not know why they chose SAFe. It is good if the companies are clear about why they are using SAFe and why they chose SAFe because it motivates the co-workers better.

5.1.1 Challenges and changes

When it comes to challenges that the companies have it differs a lot depending on many factors such as amount of people and background. The more people that are involved in the implementation, the more resistance there is from the co-workers and it is harder to get test automation and continuous integration in place. The more people that are involved the harder it gets to make changes, teach everybody and to convince everybody that SAFe is the right way to go. When the implementation only effects a small amount of people the challenge is more to get everybody to know what is required of their new role. When it comes to different background it is no surprise that companies with different backgrounds have different challenges when implementing SAFe. Companies with a waterfall background have more changes when it comes to resistant co-workers and that managers do not want to let go of their responsibility. As mentioned before, the agile teams do not have these challenges since the co-workers and managers have already embraced these changes.

The companies have also solved changes differently depending the amount of people that are involved. The companies with many persons involved have solved challenges by bringing in coaches and educating people. Companies with fewer persons involved have learned by their own mistakes and during time. It seems like companies that make a big investment in amount of persons also makes a big investment in how they solve challenges. The companies that try out SAFe a bit first do not make so much investment in coaches and education. When doing the implementation all the companies had to do some organizational changes. The companies that had more people involved had to do bigger changes such as reorganize the whole organization and everybody got new roles. The companies with fewer people had done some of the changes before the implementation and that the teams were also responsible for the maintenance. It seems like the more people that are involved, more changes are necessary to do in the organization. This can also depend on the companies' backgrounds since companies with an agile background needs to do less changes than the companies with a waterfall background. The companies with an agile background already have some of the levels in SAFe in place.

5.1.2 SAFe implementation

As mentioned above in the result chapter, all the companies implemented SAFe in parts of their organization. It seems to be a smart approach as they do not take on too much that they cannot handle. For some companies it is not possible to implement SAFe in all parts of their organization for various reasons. All the companies had similar goals with implementing SAFe but few of them followed-up if the implementation fulfilled their goals. The companies with many people involved had some defined KPI' that they measured and compared. The more people that are involved the more measurements are needed because it gets harder to get an overview on how the process work when many people are involved. Companies should do some measurements so that they can see that the implementation is successful and fulfills their goals.

One thing that was surprising was the amount of time it took to implement SAFe. It did not differ a lot between the groups on how long time it took. One would think that it would take longer time if more persons are involved. It was also a bit surprising that it did not get particularly faster for companies with an agile background to get SAFe to work compared to the companies with a waterfall background. One would think that it should go faster for companies with an agile background since they already have some things in place such as cross-functional teams.

5.1.3 Test and quality aspects

When it comes to the tests aspects almost all companies had challenges with test automation. The companies with many people involved had challenges with continuous integration but not the companies with fewer people involved. This might be due to that those companies have not tried to implement continuous integration yet but this was nothing that the thesis writer asked. The companies with fewer people involved had more problems with getting the teams to cooperate. When it comes to solving the problems, the companies did almost the same as for the other challenges. The companies with many people involved took in consultants and educated people, and the companies with fewer people involved had test communities and a DevOps team. The companies with fewer people involved do not invest in education or coaches, but instead try to solve the challenges on their own with different approaches.

The majority of the companies' test processes changed when implementing SAFe and almost the same things changed for all the companies. To test in parallel and to do automated tests instead of manual tests was the most general changes. The test process does not differ a lot between the companies and that is because the test process should work in a certain way when working according to SAFe. However the companies have seen different effects on the changes that they have made. Companies with many people involved saw that they found defects/bugs faster and earlier and that they had more stable systems in the production. The companies with fewer people involved had not done any measurements on this yet. It seems like companies with more people involved have more measurements and this could be because of the amount of people that are involved. Many of the companies measure test and quality now when they have implemented SAFe. Some of the metrics that they use are the same for all companies but the metrics that differs depends on what product they deliver and not on the amount of persons that are involved. Lastly, when it comes to the tools, all companies have done some changes. The companies with few people involved did not use the same tools as the companies with more people involved. This can be due to that some tools are good to use when handling few people. The companies with few people involved might have to change tools when they get more people involved in SAFe.

5.2 Guidelines

The results leads to some guidelines that can help and facilitate for companies that are going to implement SAFe. The different guidelines are listed below.

- **Suppliers:** Companies should have as little as possible of their development and testing outsourced to suppliers. When having many suppliers it results in for example lower quality, it takes longer time and the company does not have the same overview.
- **Amount of people:** When implementing SAFe, a company should start small and then expand when they see that the implementation gives the results that they wanted. When starting with many people it results in more challenges and that it takes longer time.
- **Have patience:** It takes a long time to get SAFe in place so the companies should have patience because it is not an easy task.
- **Bring in help:** Companies should not be afraid to bring in help because sometime the best solution is to bring in some expert. It is important that the experts has the right competence because otherwise it can give the opposite effect than what the company wanted.
- **Follow up:** It is very important that companies follow up on their implementation and the changes that they make in their organization. If companies do not follow up they will not know if the changes gives the results that they wanted.
- **Test automation:** Companies should automate as many tests as possible and wherever it is possible. This gives better quality and makes it possible to verify and validate the system much faster.
- **Test in parallel:** It is important to test in parallel to the development because then the new functionality is tested as early as possible. This leads to better quality and less time for fixing bugs in the system.

5.3 Conclusion

Companies will always encounter challenges when implementing a new framework. That is also the case when implementing SAFe. The most general challenges are: changing tools, prevent people from going back to the old way of working and that old roles have difficulties to let go of their responsibility. The most general ways to solve this are to bring in consultants, send people on education about SAFe and to have patience because it takes time to implement SAFe. When it comes to challenges in test and quality area the most general are to get test automation in place, get the whole team to understand that they are all responsible for test and quality, and to get continuous integration in place. The most general way of solving challenges in test and quality area is to take in coaches to help them. Some companies created communities for testers and developers to solve problems between different teams.

The companies had to make some changes when implementing SAFe. They had to do some organizational changes to get everything to work. Some example of organizational changes is cross-functional teams and to give people new roles. The most general roles in the teams are developers, testers, scrum master and product owner. Many of the companies do not follow up their implementation and the changes they make. The companies should do some measurements so that they can see that the implementation is fulfilling their goals. When doing changes to the test process some companies follow up the implementation with some KPI's and metrics. Some examples on metrics are amount of defects, amount of defects that the customer finds and code coverage. The best way of working with test in SAFe is to have as many automated test as possible and to test in parallel to the development. To implement SAFe takes a lot of time, so companies that are going to implement SAFe have to know that it requires a lot of patience. It also requires a lot of investment from a company to succeed with this but the investment will pay off since they deliver faster and have much better quality.

5.4 Future work

As mentioned in boundaries, some aspects have not been considered in this thesis. It would be possible to make some research about what challenges and changes companies had in other areas than test and quality when implementing SAFe. Many companies mentioned having challenges with change management, so that would be a good area to do research in. It would also be interesting to do some research about companies that have tried implementing SAFe but stopped the implementation for some reason. It would be a good complement to this thesis to see what challenges that stopped people from implementing SAFe.

Companies have said why they chose SAFe and it would be very interesting to do more research about this. Some parts of their old work process did not work as they wanted and this contributed to why they chose to implement SAFe. It would be interesting to know which these parts were because then it would be possible to do an analysis about this. The analysis would result in knowledge about if it was possible for the company to only have done some smaller changes to achieve their goals instead of doing the big change of implementing SAFe.

Another research that would be very interesting would be to see if SAFe is better suited for certain companies. SAFe might be better suited for companies with certain products, sizes or structures. In this context, structures means for example if they have a lot of their development and test outsourced or a lot of hardware dependencies. It would also be interesting to compare SAFe with other frameworks to see which are SAFe strengths and weaknesses. This would help companies to see if some other framework suites better for them than SAFe. This would help companies that want to do a change and are facing the choice of which framework to use. With that research a company could easily pick a framework that suites them best.

Bibliography

Books and articles

[1] Orr, A. (2011). *Introduction to the ITIL Service Lifecycle.* 3:rd ed. Norwich:The Stationery Office, pp. 3-7.

[2] Tonnquist, B. (2009). *Project management - a complete guide.* Aarhus:Academica, pp. 335.

[3] Gandomani, T., Zulzalil, H., Ghani, A., Sultan, A. and Nafchi, M. (2013). *Obstacles in moving to agile software development methods; at a glance.* Journal of Computer Science, 9(5), pp. 620-625.

[4] Boehm, B. and Turner, R. (2005). *Management challenges to implement agile processes in traditional development organizations.* IEEE Software, 22(5), pp. 30–39.

[5] Talby, D., Keren, A., Hazzan, O. and Dubinsky, Y. (2006). *Agile software testing in a large-scale project.* IEEE Software, 23(4), pp. 30-37.

[6] O'Donnell, M. and Richardson, I. (2008). *Problems Encountered When Implementing Agile Methods in a Very Small Company.* In: Software Process Improvement. Vol. 16. Ireland: Springer, pp. 13-24.

[7] Hansmann, U. and Stober, T. (2010). *Agile Software Development - Best Practices for Large Software Development Projects.* Berlin:Springer, pp.15-33.

[8] Hansmann, U. and Stober, T. (2010). *Agile Software Development - Best Practices for Large Software Development Projects.* Berlin:Springer, pp.1-14.

[9] Hansmann, U. and Stober, T. (2010). *Agile Software Development - Best Practices for Large Software Development Projects.* Berlin:Springer, pp.171-175.

[10] Robson C. (1993). *Real World Research, A Resource for Social Scientists and Practitioner–Researchers.* Oxford:Blackwell, pp. 269-291.

[11] Robson C. (1993). *Real World Research, A Resource for Social Scientists and Practitioner–Researchers*. Oxford:Blackwell, pp. 227-268.

[12] Alexanderson, K. (2012). *Källkritik på Internet.* Ödeshög:DanagårdsLiTHO, pp.7-44.

Websites

[13] scaledagileframework.com, (2016). Scaled Agile Framework Official Website. [online] Available at: `http://www.scaledagileframework.com` [Accessed 26 Sep. 2016].

[14] scaledagileframework.com, (2016). About SAFe. [online] Available at: `http://www.scaledagileframework.com/about/` [Accessed 6 Sep. 2016].

[15] Crain, A. (2016). 4 biggest challenges in moving to Scaled Agile Framework (SAFe).[online] TechBeacon. Available at:`http://techbeacon.com/moving-safe-scaled-agile-framework-4-biggest-challenges` [Accessed 20 Oct. 2016].

[16] scaledagileframework.com, (2016). ScrumXP. [online] Available at: `http://www.scaledagileframework.com/scrumxp/` [Accessed 18 Oct. 2016].

[17] scaledagileframework.com, (2016). Kanban. [online] Available at: `http://www.scaledagileframework.com/team-kanban/` [Accessed 18 Oct. 2016].

[18] istqbexamcertification.com. What is Agile model – advantages, disadvantages and when to use it?. [online] Available at: `http://istqbexamcertification.com/what-is-agile-model-advantages-disadvantages-and-when-to-use-it/` [Accessed 9 Sep. 2016].

[19] scaledagileframework.com, (2016). SAFe core values. [online] Available at: `http://scaledagileframework.com/safe-core-values/` [Accessed 19 Oct. 2016].

[20] scaledagileframework.com, (2016). SAFe principles. [online] Available at: `http://scaledagileframework.com/safe-lean-agile-principles/` [Accessed 16 Sep. 2016].

[21] scaledagileframework.com, (2016). Economic view. [online] Available at: `http://scaledagileframework.com/take-an-economic-view/` [Accessed 16 Sep. 2016].

[22] scaledagileframework.com, (2016). System thinking. [online] Available at: `http://scaledagileframework.com/apply-systems-thinking/` [Accessed 16 Sep. 2016].

[23] scaledagileframework.com, (2016). Set based design. [online] Available at: `http://scaledagileframework.com/assume-variability-preserve-options/` [Accessed 19 Sep. 2016].

[24] scaledagileframework.com, (2016). Integration points. [online] Available at: `http://scaledagileframework.com/build-incrementally-with-fast-integrated-learning-cycles/` [Accessed 19 Sep. 2016].

[25] scaledagileframework.com, (2016). Milestones. [online] Available at: `http://scaledagileframework.com/base-milestones-on-objective-evaluation-of-working-systems/` [Accessed 20 Sep. 2016].

[26] scaledagileframework.com, (2016). Limit WIP. [online] Available at: `http://scaledagileframework.com/visualize-and-limit-wip-reduce-batch-sizes-and-manage-queue-lengths/` [Accessed 20 Sep. 2016].

[27] scaledagileframework.com, (2016). Apply cadence. [online] Available at: `http://scaledagileframework.com/apply-cadence-synchronize-with-cross-domain-planning/` [Accessed 20 Sep. 2016].

[28] scaledagileframework.com, (2016). Motivation of workers. [online] Available at: `http://scaledagileframework.com/unlock-the-intrinsic-motivation-of-knowledge-workers/` [Accessed 20 Sep. 2016].

[29] scaledagileframework.com, (2016). Decentralized decision making. [online] Available at: `http://scaledagileframework.com/decentralize-decision-making/` [Accessed 20 Sep. 2016].

[30] scaledagileframework.com, (2016). SAFe poster. [online] Available at: `http://scaledagileframework.com/posters/` [Accessed 12 Nov. 2016].

[31] scaledagileframework.com, (2016). Permissions. [online] Available at: `http://scaledagileframework.com/usage-and-permissions/` [Accessed 12 Nov. 2016].

[32] scaledagileframework.com, (2016). Portfolio level. [online] Available at: `http://www.scaledagileframework.com/portfolio-level/` [Accessed 12 Sep. 2016].

[33] scaledagileframework.com, (2016). PPM. [online] Available at: `http://www.scaledagileframework.com/program-portfolio-management/` [Accessed 12 Sep. 2016].

[34] scaledagileframework.com, (2016). Value stream level. [online] Available at: `http://www.scaledagileframework.com/value-stream-level/` [Accessed 12 Sep. 2016].

[35] scaledagileframework.com, (2016). Program level. [online] Available at: `http://www.scaledagileframework.com/program-level/` [Accessed 13 Sep. 2016].

[36] scaledagileframework.com, (2016). Team level. [online] Available at: http://www.scaledagileframework.com/team-level/ [Accessed 13 Sep. 2016].

[37] scaledagileframework.com, (2016). Built-in Quality. [online] Available at: http://www.scaledagileframework.com/built-in-quality/ [Accessed 12 Oct. 2016].

[38] agilemanifesto.org, (2001). Agile Manifesto. [online] Available at: http://agilemanifesto.org/ [Accessed 18 Oct. 2016].

[39] scaledagileframework.com, (2016). Test-first. [online] Available at: http://www.scaledagileframework.com/test-first/ [Accessed 9 Oct. 2016].

[40] scaledagileframework.com, (2016). Non-functional requirements. [online] Available at: http://www.scaledagileframework.com/nonfunctional-requirements/ [Accessed 9 Oct. 2016].

[41] scaledagileframework.com, (2016). Implementing SAFe. [online] Available at: http://scaledagileframework.com/implementing/ [Accessed 28 Sep. 2016].

[42] scaledagileframework.com, (2016). Mixing Agile and Waterfall. [online] Available at: http://www.scaledagileframework.com/mixing-agile-and-waterfall-development-in-the-scaled-agile-framework/ [Accessed 11 Oct. 2016].

[43] scaledagileframework.com, (2016). Technical strategies. [online] Available at: http://www.scaledagileframework.com/technical-strategies-for-agile-and-waterfall-interoperability-at-scale/ [Accessed 12 Oct. 2016].

Appendices

Appendix A

Questions asked in the interviews

Company related questions:

1. How big is the company?
    (a) Amount of employees in total?
    (b) Amount of employees that work in SAFe?
2. Where is the company located?
    (a) International or just in Sweden?
    (b) Is everybody that works in SAFe located at one place or are they spread out?
3. Do you have suppliers that deliver parts of your systems?
    (a) Do you have suppliers that are involved in SAFe?
4. How many products/systems do you have?
    (a) Do you have many system integrations?
    (b) How many systems/products are involved in SAFe?

Consultant related questions:

1. How many companies have you been involved in their SAFe implementation?
2. How big were those companies?
    (a) How many employees work in SAFe?
3. Where were the companies located?

   (a) International or just in Sweden?
   (b) Were everybody that works in SAFe located at one place or are they spread out?
4. Which is most common, having suppliers or develops everything themselves?

SAFe implementation in general:

1. How did you work before you implemented SAFe?
   (a) Agile, waterfall..?
2. Why did you choose SAFe?
   (a) How did you hear about SAFe?
   (b) Did you look at other frameworks before you chose SAFe?
3. Which challenges did you have generally when implementing SAFe?
4. How did you solve the challenges that you were facing?
5. Have you implemented SAFe in the whole organization or just in parts of the organization?
   (a) If just parts: Why just parts?
6. What was your goal with implementing SAFe?
   (a) Have you reached those goals?
7. How long time took it for you to implement SAFe?
8. How do you follow up the implementation?
   (a) Have you done any comparison if it goes faster or not with SAFe?
9. Did you have to do any organizational changes when implementing SAFe? Which?
10. Which roles do you have in a team?

Implementation, Test and quality:

1. Are your tests done in-house or outsourced?
   (a) If outsourced: is it the same team that does both test and development?
2. Which challenges did you have within test and quality when implementing SAFe?
3. How did you solve these challenges?
4. Did you test process change when implementing SAFe?
   (a) How did it change?
5. Have you been able to show an effect where you find more errors now when your test process changed?

6. How do you measure test and quality?
   (a) Which metrics do you look at?
   (b) Which KPI's have changed when implementing SAFe?
7. Did you have to do any changes when it comes to tools within test when implementing SAFe?

Institutionen för Datavetenskap | Lunds Tekniska Högskola | Redovisas 13 JANUARI 2017
Examensarbete: Evaluation of the test area when implementing the framework SAFe
Student(er): Sandra Fridälv
Handledare: Christin Lindholm(LTH)
Examinator: Martin Höst

Utmaningar med att införa ett arbetssätt

Populärvetenskaplig Sammanfattning av ***Sandra Fridälv***

Idag inför många företag det nya arbetssättet SAFe för att öka produktiviteten, kvaliten och för att kunna leverera snabbare till marknaden. Företag som står inför detta införandet vet inte vilka utmaningar de kan stötta på, hur de bör lösa dessa utmaningar och hur testningen ändras. Detta examensarbete beskriver utmaningarna och hur man kan lösa dom.

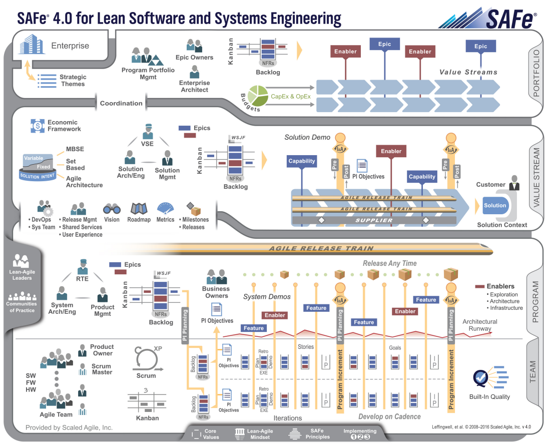
Införa nytt arbetssätt

Konkurrensen mellan företag är väldigt hög idag så för att företag ska kunna konkurrera måste de har hög kvalitet och kunna leverera fort till marknaden. Det finns olika tillvägagångssätt för att bli mer konkurrenskraftig, bland annat att införa ett nytt sätt att arbeta på i sin IT organisation. Ett alternativ som blir mer och mer populärt är arbetsättet SAFe som hjälper företag att jobba mer effektivt på alla nivåer i ett företag. Figuren visar hur SAFe fungerar på de olika nivåerna. Detta skiljer sig en del från andra arbetssätt som bara fokuserar på team nivån. SAFe beskriver hur företag ska arbeta för att få bättre kvalitet och kunna leverera fortare till marknaden. Att införa detta arbetssättet är väldigt komplicerat och det tar väldigt lång tid att införa vilket gör att företagen måste ha en del tålamod. Företaget jag har gjort mitt examensarbete på ska påbörja införandet av SAFe och de undrar om det finns några generella utmaningar med införandet, hur företag i så fall har löst dessa utmaningar och hur de har gått tillväga med testningen i det nya arbetssättet.

I mitt examensarbete har jag intervjuat 20 personer från 17 olika företag för att höra om deras erfarenheter av att införa SAFe. Jag har haft ett fokus på test och kvalitet då jag gjorde mitt examensarbete på test avdelningen på ett företag. Resultatet visar att företag behöver göra en hel del ändringar när de inför SAFe och detta medför en hel del utmaningar. Att arbeta enligt SAFe är en stor omställning för många medarbetare då de skiljer sig en del från hur företaget jobbade innan. Många medarbetare tycker det är jobbigt med en så stor förändring och detta kan då leda till att medarbetarna går tillbaka till det gamla, bekväma sättet att arbeta på. Företag kan vara tvungna att byta program som de jobbar i när de inför SAFe eftersom programmen är kopplade till sättet man arbetar på. Det är en stor utmaning och det tar lång tid att lära alla att använda de nya programmen. Företag har oftast löst dessa utmaningar genom att utbilda folk och tagit in coacher som hjälper medarbetarna i början.

Vid införandet av SAFe har företag även en del utmaningar när det gäller test och kvalitet. Företag som inte har haft automatiska tester innan tycker att det är svårt att få det att fungera. Det kräver en hel del träning och utbildning av medarbetarna. En annan utmaning är att få utvecklare och testare att förstå att de tillsammans är ansvariga för test och kvalitet. Många på företagen ser testarna som ansvariga för test och kvalite men utvecklarna har lika mycket ansvar för det. Detta brukar företag också lösa genom att ta in coacher som hjälper medarbetarna.

Resultatet från intervjuerna sammanställdes och analyserades för att se vilka utmaningar som är mest vanliga och vilket som är det vanligaste sättet att lösa de på. Analysen visade också att där finns kopplingar mellan antalet personer som var involverade i SAFe införandet på ett företag och vilka utmaningar de hade. Med hjälp av mitt examensarbete kan företag som står inför införandet av SAFe vara mer förbereda och förhoppningsvis lösa utmaningarna fortare. Även företag som vill förbättra sitt arbetssätt kan ta nytta av mitt examensarbete då de kan identifiera utmaningar de har och då ta inspiration från hur andra företag har gått tillväga för att lösa det.


SAFe® 4.0 for Lean Software and Systems Engineering
SAFe®
Enterprise
Strategic Themes
Program Portfolio Mgmt
Epic Owners
Enterprise Architect
Kanban
Backlog
Coordination
Value Streams
PORTFOLIO
Economic Framework
Solution Demo
Solution Arch/Eng
Solution Mgmt
Customer
Solution Context
VALUE STREAM
AGILE RELEASE TRAIN
Lean-Agile Leaders
Communities of Practice
RTE
System Arch/Eng
Product Mgmt
Business Owners
System Demos
Enabler
Feature
PI Objectives
PROGRAM
Product Owner
Scrum Master
Scrum
Kanban
Agile Team
Iterations
Develop on Cadence
Built-In Quality
TEAM
Core Values
Lean-Agile Mindset
SAFe Principles
Implementing
Provided by Scaled Agile, Inc.