Improving the Software Development Process in a Software Development Team - a Case Study

Laxmi Thebe

School of Science

Thesis submitted for examination for the degree of Master of Science in Technology.
Helsinki 28.09.2020

Supervisor

Assoc. Prof. Casper Lassenius

Advisor

M.Sc. (Tech.) Miika Nordström

Aalto University
School of Science

Aalto University
School of Science

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of the master's thesis

Author Laxmi Thebe

Title Improving the Software Development Process in a Software Development Team - a Case Study

Degree programme Master's Programme in Computer, Communication and Information Sciences

Major Software and Service Engineering **Code of major** SCI3043

Supervisor Assoc. Prof. Casper Lassenius

Advisor M.Sc. (Tech.) Miika Nordström

Date 28.09.2020 **Number of pages** 65+8 **Language** English

Abstract

Changes in the context in which software engineering practices are carried out also initiate the need to change in the practices to effectively work as a development team while delivering the software product with the highest possible values. While the thesis was initiated to improve the continuous integration and delivery practices in the case company, the context and the need for the changes in the practices highlighted the need for enhancing the Scrum practices within the software development team. With the design science research methodology approach, the problems in the software development team were drawn during the current state analysis phase followed by a workshop to discuss the findings and select the challenges to tackle as part of the thesis work - both from the case company and development team members' perspective. The results from the current state analysis highlight five core problem areas from which problem area 'Process and Tools' was selected for solving in this project after the discussion with the development team. Despite already utilizing some practices of Scrum, the development team decided to evolve the Scrum adoption with the utmost goal of solving concrete problems in the problem area captured during the current state analysis phase. Semi-structured interviews and surveys were utilized to collect the data, and the findings reveal the potential of the process while suggesting further improvements. Scrum is easier to understand but challenging to master. The process exposes the potential, offers the possibility to respond to the challenges in an agile way while emphasizing the importance of context in shaping the practices and tools which is utilized for software construction.

Keywords Agile Development, Scrum, Distributed Agile Development

Preface

I owe my sincere gratitude and appreciation to all beings whom I happen to share even a brief amount of time in person or by other means. In particular - to my parents for their relentless effort to support my school education, to my friend Harsha and uncle Ganga Siwakoti for being there in my most vital period of life during my high school, to Finland for accepting me regardless of my origin, to my girls for keeping me always motivated and giving me reasons to pursue my dreams, to my colleagues at Vediafi Oy for supporting through the whole project, to Joona Jalovaara for being there to guide in my professional growth, to my advisor Miika Nordström for providing supportive environment and guidance, to my supervisor Casper Lassenius for always motivating and guiding me through this whole project. Without your support and involvement, I would not have been able to be part and bring this project into this state.

Helsinki, 29.08.2020

Laxmi Thebe

Contents

Symbols and abbreviations

Abbreviations

CI	Continuous Integration
CD	Continuous Delivery
API	Application Programming Interface
PR	Pull Request
DoD	Definition of Done
DSRM	Design Science Research Methodology
PO	Product Owner
SM	Scrum Master

1 Introduction

Scrum is a powerful lightweight process framework (Schwaber, 1997) suitable to utilize in complex domains of work where wicked problems are inherent like in the fields of software development. It is simple to understand yet difficult to master but offers the opportunity to try something new, see how it goes, and adapt it in the new light of knowledge and hence repeat - do it again but adaptively.

With the adoption of agile practices in software development, there has always been a focus on constant improvement in the overall software development process with emphasis on three core values - transparency, inspection, and adaptation. Automation is an ultimate outcome of our pursuit of achieving improvement in the aspect of the process - whereas there are other core achievements in improvement in our practice, process, and tools.

The case company has recently started adopting continuous integration and delivery by developing pipelines with the help of the external consultant when the software development team grew from two existing members to the eight members team. This change not only introduced the need for a more robust approach to the usage of tools, but it also established the need to adopt or at least question the existing practices. The initial purpose for the thesis was to develop knowledge and skill to handle the continuous integration and delivery (CI/CD) implementation in future projects, but the study of current state analysis exposed most of the problems in the software development process. As a result of that, the software development team in the case company was mandated for improving the existing Scrum process to solve the current problems before thinking about tools and automation.

Even though the demographic distribution of workforce enabled with technological tools and globalization might have introduced the discussion of having distributed Scrum, we are now forced to work remotely due to the COVID-19 pandemic. Development team members are mostly working from the home office - a kind of distributed work in which the day to day development works is performed from home differentiating from other distributed work settings by creating a work in an individually distributed setting (Luz et al., 2009). The current situation demands to tailor the Scrum process even for the team otherwise collocated in the same office space.

1.1 Research Questions

This thesis aims to explore the Scrum practices used in the case company at the time of undertaking this endeavor by observing the current practices while exploring the problems and implementing a solution to handle those problems. As a part of the DSRM research approach, the application of the solution in the given context is evaluated using an exploratory approach. The thesis undertakes the inquiries to answer the following research questions:

Research problem How to improve the software development practices in the case company?

RQ 1	What are the current problems in the software development team in the case company?
RQ 2	What practices are employed by the software development team in the case company?
RQ 3	What are the changes introduced in the software development team?
RQ 4	How the introduced changes affect the existing problems?

Table 1: The research questions

In addition to answering the research questions, this thesis explores the further possibility of improvement in the software development practices in the case company as part of the continuous improvement approach as the core idea of contributing to the case company while accomplishing the thesis project.

1.2 Thesis Structure

The thesis structure reflects the adoption of the DSRM process model in its content structure. Section 1 presents the research problems and motivation while the case background is introduced by introducing the case company in brief in Section 2. In Section 3, the thesis presents the overview of agile software development and scrum in particular while also discussing the challenges and opportunities of the scrum in other similar situations. Due to the current COVID-19 pandemic situation, the software development team in the case company is forced to work as a distributed team and hence the adoption and challenges of Scrum in distributed settings are also discussed. This enables us to view the implementation in light of the existing practices and the knowledge of literature. The research method utilized to conduct this thesis project is elaborated in Section 4, whereas Section 5 introduces the current state analysis in the software development team in the case company which aims to frame research problems. In Section 6, the selected problems are highlighted while discussing the recommended practices and results of the retrospectives conducted during the Scrum practices. The implementation is evaluated as described in Section 7 while Section 8 discusses the research questions and answers to the questions and the limitations of the research before concluding the thesis project in the final Section 9.

2 Background

As the thesis was conducted in the context of the case company, this section provides an overview of that context and provides a general idea for the thesis on-wards. As the DSRM approach is applied to improve the situation while the context in the case company changed, this section also discusses the newly developed context after explaining the case company in brief. Moreover, it discusses the current practices - which further sets the context for the problem analysis phase of this project.

2.1 Case Company

Vediafi Oy is a growing IT company with two part-time developers less than two years ago now having eight developers in its software development team - some of the developers are working on the project full time while others are working for part time. It offers its services and expertise in the field of logistics by developing various projects and products based on context and clients ranging from individuals to companies and organizations. The company relies on positioning hardware, different sensor devices, and third-party APIs to gather different information to create data-driven smart logistics solutions based on those technologies. Some of the projects have sophisticated architecture and use different kinds of services for their functioning while the applications are mainly developed using Python and JavaScript frameworks.

Moreover, as Vediafi Oy seeks to achieve quality in the software product and improvement in developer productivity, looking into the issue of the improvement of one of the core practices is of vital importance. Shorter lead time is sought after to achieve quality products which are tested by internal members manually before the release of the product. The projects undertaken at Vediafi Oy involve the integration of different services and projects are mostly feature-driven while requirements are changing often. According to the cultural model of the sociologist Ron Westrum, the organization culture in Vediafi Oy is somewhere in the middle of generative and bureaucratic state while focusing on harnessing generative culture which is highly performance-oriented, cooperative, innovative, and shared responsibilities and risks (Google, nd). Inter-team communication is missing but improving all the time and the local team cultures within the software development team are strengthening. The well-being questionnaire is sent monthly within the company with an average rating of 4.2 out of 5 which was also reflected by the developers during the interview discussion when the case of figuring out problems in the software development team was conducted.

The software development projects in Vediafi Oy offer different kinds of challenges and are different in nature and complexities. Some projects are to be delivered or at least achieve some targets in incremental steps which requires delivering software quickly and reliably. The agile software development methodology is used in the software development team whereas a biweekly company-wide meeting is conducted to highlight the general view of the projects which highlights the current status and future expectations from the projects. Slack is predominantly used within the software development team for team communication while emails are mostly used

as a communication channel within the company. G Suite tools are being heavily used for sharing calendars, files, and video communication during this COVID-19 pandemic time in particular.

2.2 New Context - Reasons for Change

While automation and continuous improvement play a vital role in the software development process, the case company focuses on achieving developer productivity and software quality while building products for our projects. As the team seeks to achieve that goal, continuous integration and delivery pipelines are used to automate and enforce the rule that ensures software quality and developer productivity. One of the ways the pipelines enable developer productivity is related to the reduced risks while releasing the new software project artifacts as it is mostly automated and risks are significantly mitigated with the possibility to rollback the changes if something wrong happens. As stated earlier, the software development team consisted of two developers working next to each other and communicating by going over the desk if some issues arose during the day to day work while mostly resolving issues immediately. Due to the growing size of the software development team, the team was working from two different rooms while one of the developers worked remotely before moving to the single big room which hosts all the members of the development team in a single collocated space.

As Scrum promises developer productivity and improved quality by delivering the highest values (Sadun, 2010), Scrum is already implemented to some degree in the company to materialize the vision of increasing developer productivity and enhancing the delivered product quality while working within the deadlines and milestones. With the changes in the size of the team, the need for establishing control and structure to software development activities arises while underlining the need for establishing a software process to achieve business successes (Sánchez-Gordón et al., 2016). While there are simultaneously running multiple projects in the case company often having changing requirements, there is a need to tackle the challenges of selecting and adopting suitable software development methodologies to meet the specific requirements (Flora and Chande, 2014) defined by the context of the case company while focusing on achieving development productivity and quality of code.

With the aforementioned growth, the company hired two external consultants who brought new practices and tools which resulted in the introduction of continuous integration and delivery pipelines while the whole team started to use Jira for task management. The existing CI/CD pipelines were found beneficial in our project when bug fixes require an immediate release to production - and although the deployment to production is manual, it is semi-automated and hence partially focused on delivery. Due to these new practices, the need for more discussion was realized within the team while the company started to feel the need for nurturing in-house knowledge and skills to maintain and implement new pipelines and automation tools.

2.3 Current Software Development Practice

In the case company, the software development practice is evolving as the number of software developers is growing. When there were only two team members, the developers were working in the collocated office space while communicating with the work supervisor frequently while taking away the need for most of the management aspects in software development. As the size of the software development team grew up, the team started using Jira as an agile project management software tool while Bitbucket was already being used as a code hosting platform. The work supervisor was fulfilling the role both as a product owner and scrum master during the start of this thesis study while grooming the product backlogs as a result of discussion with other stakeholders involved in the project when there were no clearly defined team roles. Developers were allowed to add the issues to the backlogs - and the development team did not have any strong conceptual understanding of the backlogs and the software development process. Concerning the practice of the Scrum, the team used to have dailies twice a week and a sprint of usually two weeks' length while having sprint planning at the beginning of the sprint. The work was done as an organized team and the process was called Scrum while having the aspects of both the Kanban method and Scrum in the development practice. The task boards were used mostly like in the Kanban method and the whole team was working with greater flexibility often not having enough discussion on how to register an issue in a Jira for example. As a result of that one, the tasks were often less detailed and missing the proper format for presenting in the Jira even when documenting requirements in user stories or use cases format is recommended (Eloranta et al., 2013). The tasks in Jira were assigned to the developers during the sprint planning to some extent and developers were discussing with each other mostly and taking the task from the board voluntarily. If the task involved coding, a branch was checked out from the master - the active development branch - which follows the naming convention for the branch established quite recently so that the team could achieve the traceability i.e. figuring out which features in the products are related to the merge commits in the code aspect of the product. Regarding the git practices, the development team used feature branch workflow where feature branches are used to create new features and merge back to the master i.e. active development branch once the feature branch is reviewed and approved.

The adoption of CI/CD practice started as a result of having an expert consultant to work with the software development team starting from one of the projects, and the practice in the company could be described with the figure 1. After the code changes are pushed to the Bitbucket repositories, an automated test is run after the environment is set up and code is checked out in the CI/CD platform i.e. CircleCI. Automated tests mostly involve unit testing and the test coverage significantly differs based on projects and the developer who is actively working on the project. If the build is failed due to the issues in the code, the developer who pushed the changes is responsible to fix the builds. Nightly builds are run dailies. During the nightly builds, automated tests are run, the artifact is built as a docker image and pushed to the docker repositories which are automatically set to deploy to the Kubernetes cluster.

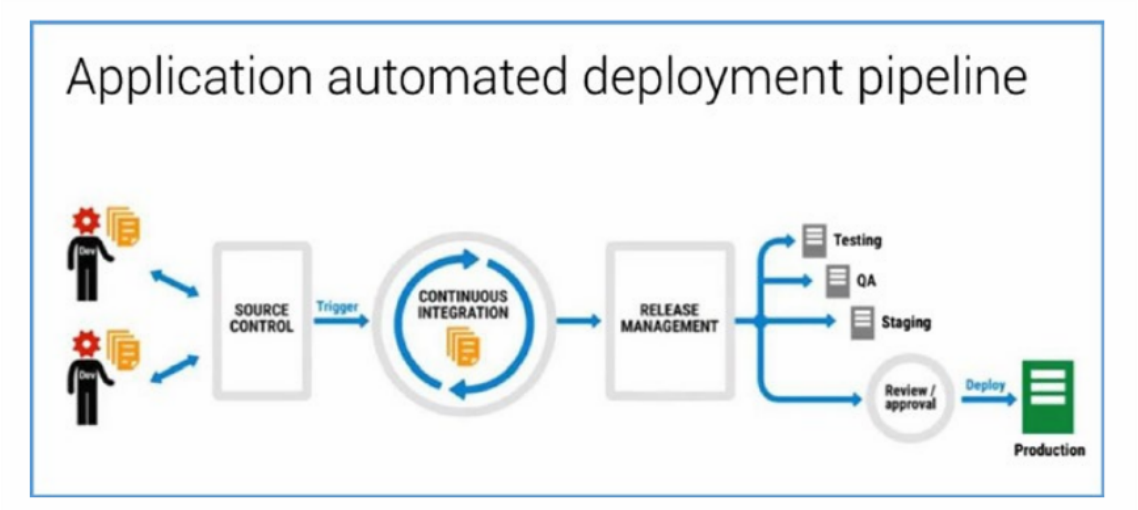


Figure 1: Application Deployment Process (Rossberg, 2019).

The software products are hence released in three different environments - testing, release candidate, and production. As stated already, the nightly builds are released in the testing environment, whereas deployment to release candidate environment and production environment is automated once the code is checked out following branching conventions and pushing the branch to the repositories. Production deployment was not automated at the beginning of the study, but it is automated to the point that it only needs manual approval. The work on implementing end to end testing is discussed but still not implemented. While the importance of developers' involvement in initial deployments is valued by most organizations (Davis, 2019), the presence of the anxiety during the deployment of updates to production is often experienced in the development team while most of the development team members are not particularly aware of the overall process.

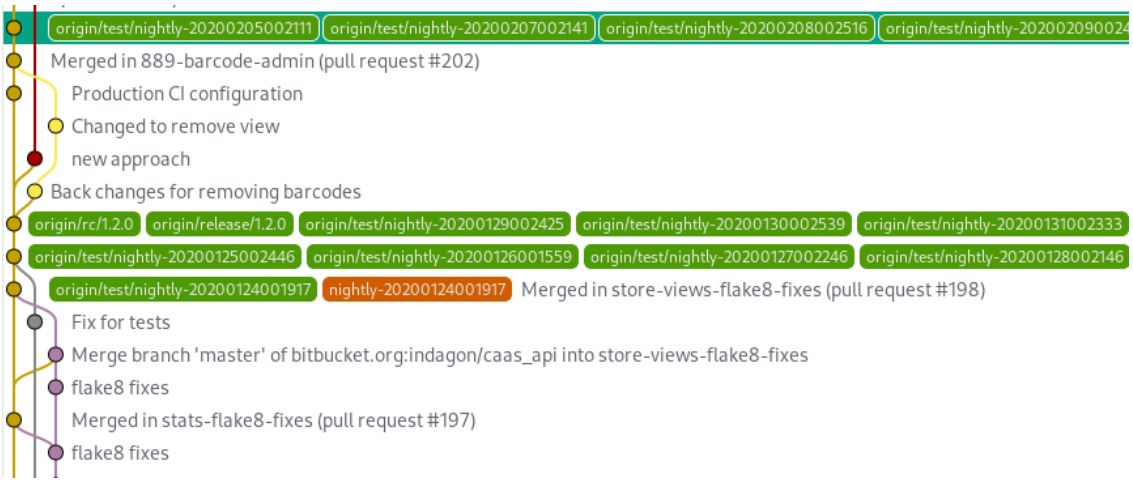


Figure 2: Git workflow and release practice

As a part of this development in CI/CD, there are also tools for monitoring and

alerting but barely used by developers - and the error tracking software report is often neglected for many false-positive reports. At the moment, pull requests are merged manually and some developers are unwilling to merge the pull request - which necessitates the need for an automatic merge if the pull request fulfills the agreed criterion as shown in the figure 2. To raise situational awareness through logging and monitoring, Slack is integrated with other tools so that a developer is notified in relevant Slack channels. Documentation is done in Confluence which is also integrated with Slack.

In conclusion, the software development team has tools to establish better engineering practices while lacking enough discussions or proper utilization of tools and the need for establishing a lightweight process while working in multiple projects within the time-boxed single sprint.

3 Literature Review

The initial goal of the thesis was to improve CI/CD practices in the case company, but the problem analysis phase leads to the need to improve the existing software development process practices in the case company after discussing it with the development team. In addition to guiding the implementation phase of the thesis process, this section provides the rationale and context for different concrete practices employed in the process. Although most of the discussion is around what is Scrum, it provides a review regarding challenges and opportunities of Scrum adoption in addition to discussing distributed scrum as an emergency need to handle the COVID-19 situation.

3.1 Agile Development

"Agile" represents the set of different practices in software development that embraces the iterative approach. The growing popularity of agile development approaches is witnessed in recent years as described by many research papers (Pauly et al., 2015). The agile development approach is proven to increase the productivity of the software development processes while achieving the delivery of done increments in a reduced amount of time (Mahmood et al., 2017). The agile approach allows flexibility in the planning and execution of the project while emphasizing the constant interaction with the clients leading to higher customer values and satisfaction (Ozierańska et al., 2016). As agile software development represents a different set of frameworks and development practices, what is core to the practice of agile software development is the grounded values and principles as listed below as expressed in the 'Agile Manifesto'. With the focus on the preference on the items in the left side over the items on the right side, the agile values are listed as:

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

In addition to the aforementioned values, agile practices are guided by the underlying principles. The underlying principles guiding agile practices are outlined below which also sets the context for defining and adopting Scrum while guiding in the process:

1. Satisfy customer by delivering continuously and early
2. Accept changing requirements
3. Delivering working software frequently
4. Daily collaboration between the development team and business people

5. Trust, support, proper environment for motivated individuals to develop projects
6. Face to face communication
7. Working software as metrics of progress
8. Sustainable development at a constant pace
9. Continuous attention to technical excellence and design to enhance agility
10. Simplicity - the art of maximizing the amount of work not done - is essential
11. Emerging nature of architectures, requirements, designs from self-organizing teams
12. Regular team reflection on being more effective. (AgileAlliance, nd)

As agile involves different sets of frameworks and practices like eXtreme Programming (XP), Scrum, Crystal Clear, Feature Driven Development (FDD), Lean Software Development, Dynamic System Development Methodology (DSDM), and Kanban (Sverrisdottir et al., 2014), Scrum is described in particular highlighting the different practices as described in the textbook implementation of the scrum.

3.2 Description of Scrum

With the wide adoption of Scrum in recent years, it is often misleadingly believed that Scrum is a recently developed process model while in truth it was first presented as "the rugby approach" in 1986 in Harvard Business Review by Hirotaka Takeuchi and Ikujiro Nonaka for the first time. Peter DeGrace and Leslie Hulet Stahl coined this concept as Scrum which itself comes from rugby which means "the quick, safe, and fair restart of a rugby game after a minor infringement or stoppage". In 1996, Schwaber and Sutherland jointly presented Scrum at Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA) while continuously adopting the changes to refine the process. (Rossberg, 2019).

Scrum is a powerful lightweight process framework - simple to understand but difficult to master. As software development involves wicked problems with often changing requirements, Scrum has been a widely used agile software development process. In contrast to the state gate approach to the software development process where it is conducted in multiple phases, Scrum adopts the incremental iterative development approach. "Agile" is the core concept of Scrum and often used as a synonym to Scrum partly due to the wide adoption of the Scrum in the software development process. Consequently, Scrum adheres to agile principles by delivering customer values to the highest possible degree.

Scrum in itself holds Empiricism in its core as described by Steve Porter in three simple and concise yet powerful and beautiful sentences i.e. "Try something new. See how it goes. Repeat." Every empirical process control is rooted on three fundamental pillars i.e.

1. Transparency
2. Inspection
3. Adaptation (Ripley and Miller, 2020)

Having shorter iteration for transparency, inspection and adaptation enable particularly smaller companies in uncertain contexts with changing business requirements to act with agility. As Schwaber mentions, software development involves complexity often manifested due to the three most significant aspects of the software development process i.e. requirements, technology, and people (Schwaber, 2004). Moreover, agile approaches like Scrum are more suitable when projects are complex and requirements are often changing and technology is far from certain by implementing the iterative approach based on empirical process control (Rossberg, 2014).

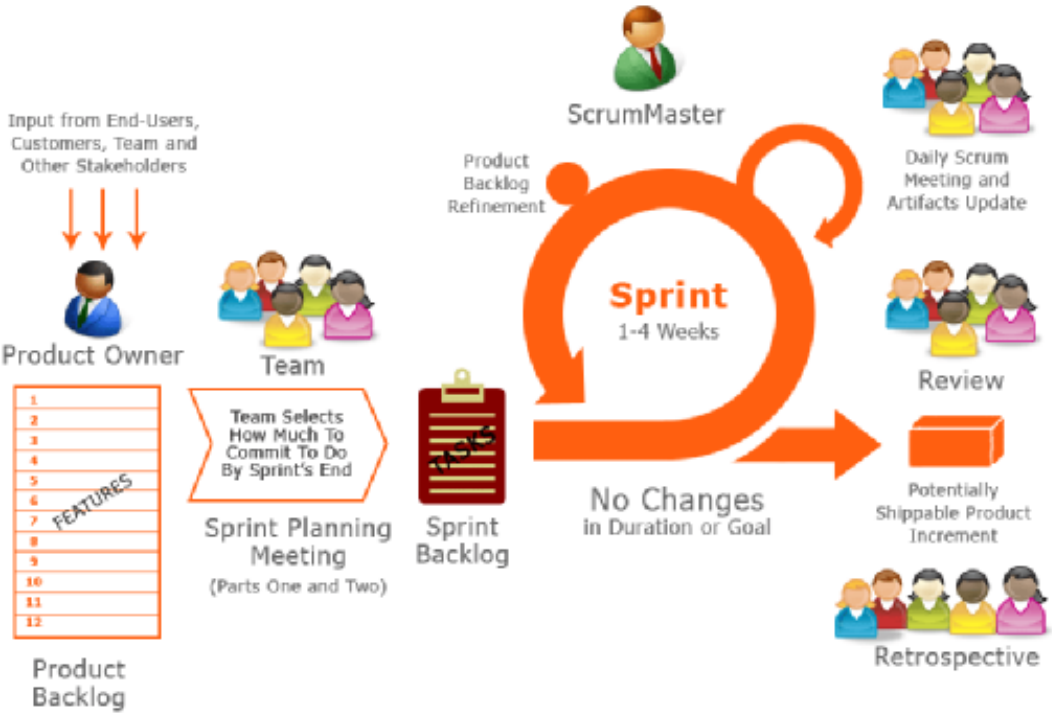


Figure 3: Scrum in Nutshell (EuropeanScrum, nd)

In a nutshell, Scrum is comprised of the development team, scrum master, and product owner working on the product backlogs and taking prioritized product backlog items to the time-boxed sprint where the product backlog items are planned and executed to turn the idea into the potentially shippable increment constantly planning and communicating during daily scrums meetings in a development team while reviewing the increments in the presence of other stakeholders as summarized in the figure 3 while celebrating the values of courage, focus, commitment, respect, and openness. As the adoption of the software development practices involves the transformation, discussion about values and creating the team spirit around the

value plays a vital role as value promotes practices like courage promotes doing the right thing and take risks, focus enables working on prioritized tasks, commitment for achieving sprint goals, respect to work in a cross-functional self-organized team as an independent member, and openness about all the work-related challenges (Dave, 2016).

3.3 Core of Scrum

The core of the Scrum outlined below is what Scrum is composed of as a framework. Being abide by these core elements of the Scrum, the scrum team members understand the importance of the three pillars of the scrum and navigate through those scrum values. The development team consists of a small, agile, and cross-functional team of highly motivated individuals with the ability to organize themselves as a team.

3.3.1 Roles

Scrum team delivers iteratively and incrementally while getting feedback from self-reflection, experiences, or stakeholders, and the team is composed of three different roles i.e. Product Owner, Scrum Master, and Scrum Team.

Product Owner

The Product Owner is the immediate proxy to the clients and represents clients while discussing with the developers and development team while discussing with the clients. The product vision is best understood by the product owner and consequently owns and grooms the product backlog items to deliver the highest possible value to the clients. The Product Owner creates the user stories in the product backlog item - with enough detail to get the work started at least for the next coming sprint with the highest priority items to be included in the next increment. The textbook definition of the product owner roles in Scrum and the actual practice in companies is studied by Sverrisdottir, Ingason, & Jonasson which further highlights the differences in the understanding of the product owner role between organizations despite agreeing on the role of the product owner as an immediate proxy between the clients and the development team. Despite the product owner being the authoritative owner of the product, the cooperation of different stakeholders in shaping the product is further elaborated (Sverrisdottir et al., 2014). Rossberg also highlights the textbook responsibilities and its meaning in real practice like how "Defining the product road map" means "ensuring that the development team understands the product backlog items to sufficient details" in practice (Rossberg, 2019).

Development Team

The development team is composed of 7 ± 2 members and is cross-functional and self-organizing in nature with the ability to deliver potentially shippable product increment at the end of each sprint by turning product backlog items to sprint backlog items. No one can force the Development Team to work from a different set of requirements. Team accountability is highly considered in the development team and the scope of the work is defined in negotiation with the Product Owner. Despite the development team members working independently concerning each other, the

development team shares the accountability collectively as a team (Permana, 2015).

Scrum Master

Scrum Master is a servant-leadership role who facilitates the whole scrum ceremonies and promotes the scrum process in the organization and the team helping the team to improve engineering practices and helping to figure out impediments to deliver greater values. Scrum master also supports product owners with product backlog items and product backlog refinement (Schwaber and Sutherland, 2017). With such a role, the scrum master enables the transparency of artifacts. According to the contribution model in the Scrum development team, scrum master roles involve activities mostly in contentual and managerial realms while most of the technical activities are done by the development team (Ramin et al., 2020). When thinking about developer productivity and saving the time wasted with unnecessary meetings, Scrum Master should focus on how to make scrum ceremonies and meetings meaningful and effective while organizing such events (McKenna, 2016). In conclusion, the role of Scrum Master is facilitation and support to the team to deliver high values effectively.

3.3.2 Artifacts

Scrum artifacts are work and value which provide tangible outcomes of the effort team puts during the sprint and provides the opportunity for inspection and adaptation. There are three artifacts in Scrum as described below:

Product Backlog

The product backlog is a prioritized list of features needed in the product - usually described as user stories or use case diagram. It is a living representation of the product which evolves continuously reflecting the change in the product over its lifetime. The product backlog items at least contain name and description while including other metrics - like order, estimate, and value. The product backlog is refined as a part of an ongoing process in collaboration with the development team and product owner. The product backlog should include the most prioritized items in the top with sufficient details that the development team could take that item to the next sprint and could turn that idea into reality. As all products have some lists of features or enhancements to be made in the future, all the products have some sort of documentation to represent that state and are called with different names like a product road map, comprehensive release plan, and product backlog in the case of Scrum (Fowler, 2019). This artifact is owned by the Product Owner.

Sprint Backlog

The sprint backlog is owned by the development team which contains the product backlog items divided into action plans to achieve the sprint goal. Sprint backlog creates visibility to all the work the development team has to do to meet the sprint goal hence taking away the need to have regular status update meetings. Sprint backlog could be discussed during the scrum daily and changed after the scrum daily with the opportunity for developers to modify it throughout the sprint. As the development team owns the sprint backlog, only the development team members are responsible to change items in the sprint backlog. The product backlog reflects

the lists of features in the product while the Sprint Backlog contains the list of the items to be completed in the time-boxed sprint i.e. the Sprint Backlog exists at the beginning of the sprint and gets discarded once the sprint is over (Fowler, 2019).

Increment

While Scrum adopts the iterative approach of development, development happens in increments where features or functions are broken down into manageable sizes which could be delivered predictably with its sum leading to the production of a working system while fulfilling both functional and quality requirements (Dalton, 2019a). The artifact increment represents the sum of all product backlog items completed by the development team during the sprint by fulfilling the "Definition of Done" criterion. It is an inspectable artifact - and could be demonstrated as the work done in the sprint review meeting while supporting empiricism. The increment is additive to previous increments or at least historically traced over the time in the development of the product (Ripley and Miller, 2020). With a robust definition of done, an increment could be potentially releasable by increasing the agility.

3.3.3 Events

Scrum involves different events in which the development team carries out different activities to produce a working software system as described in the following sections:

Sprint

Every event is time-boxed in Scrum - and sprint being the container of all other events is itself time-boxed with a duration less than a month in which an iteration of Scrum is completed while delivering the product of highest value. Sprint has a sprint goal towards which a development team works over the duration of the Sprint while being guided by the goal. In addition to guiding the development team, the goal could be used to measure the increment produced in the sprint against while inspecting and adapting in the process (Schwaber and Sutherland, 2017). Only the product owner has the authority to cancel the sprint although the decision is generally influenced by other stakeholders of the Scrum. Among the practitioners, the length of 2 to 4 weeks has become a de-facto practice in Scrum (Eloranta et al., 2013).

Sprint Planning

Sprint planning is an event where the development team discusses with the product owner about the tasks to be taken to the upcoming sprint i.e. 'what' of the product backlog items and figures out the 'how' to deliver a product increment by the end of the sprint. The product owner refines the product backlog with the help of Scrum Master if needed so the highly prioritized tasks are in enough detail so that the development team has an understanding of what is to be achieved in regards to the given product backlog items. The development team assesses the capability of what it can accomplish in the upcoming sprint and hence limiting the amount of the product backlog items to be considered for moving to the sprint backlog to break it down as sprint backlog items as part of the 'how' to deliver the product increment and achieve the result by implementing selected user stories (Dalton, 2019b). In addition to having the plans, the Sprint goal is formulated during the Sprint planning

which guides the development efforts and highlights the priority during the upcoming sprint. The following figure 4 depicts sprint planning in a nutshell while highlighting the major idea of sprint planning.

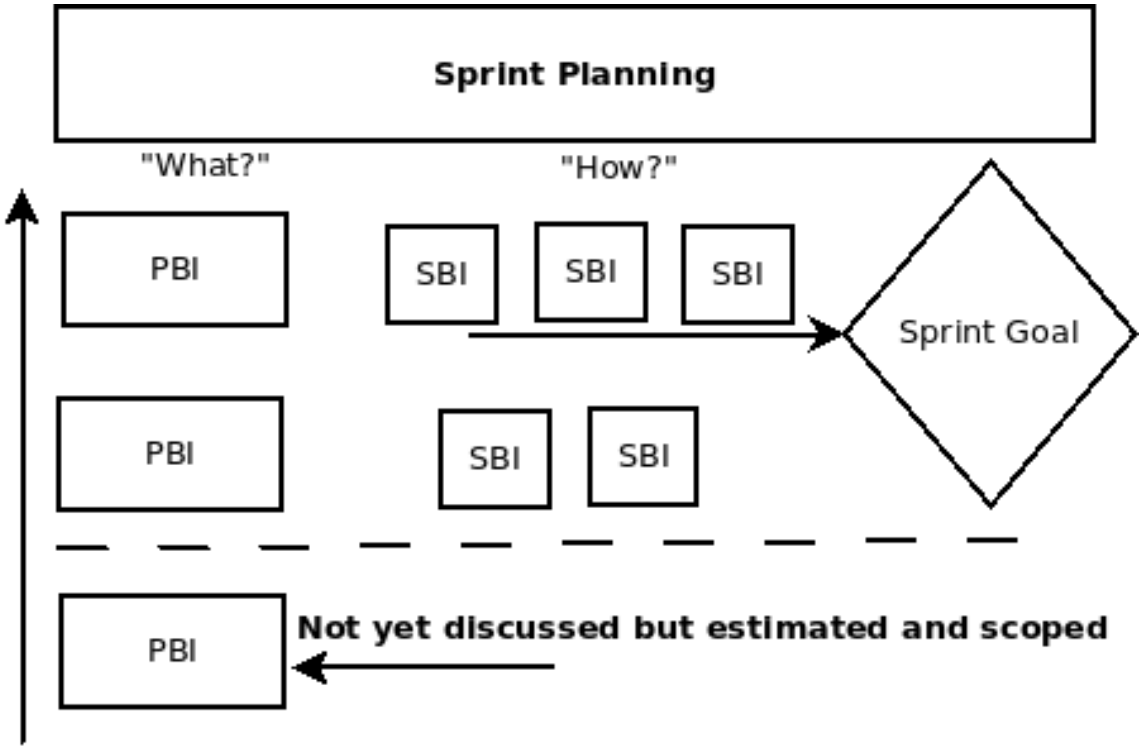


Figure 4: Sprint Planning in Picture (adopted from (Ripley and Miller, 2020))

In addition to the product backlog, sprint planning involves past performance, available working hours, and the increment from the previous sprint as it brings transparency on the planning enabling adaptation - for example, the amount of product backlog items or the accuracy of task estimation. Domain experts could be involved as part of the sprint planning if needed (Ripley and Miller, 2020). As a result of this time-boxed event, the development team should be able to explain how the team self-organizes and achieves the goal of the sprint.

Daily Scrum

A daily scrum is an event in a sprint that occurs each day in the predefined places and time so that there is no hassle or unnecessary effort to organize the event (Pauly et al., 2015). Like other events, it is time-boxed with a duration of 15 minutes which development teams utilize to synchronize and plan the daily effort to achieve the goal for the sprint while playing a vital role in ensuring the regular communication in the development team. During the daily scrum, development team members inform each other by answering what they have been doing, coordinate for the next 24 hours by discussing what they are going to do, and explore the solutions to problems by discussing what kind of impediments they are dealing with (Stray et al., 2013). Even though team members inform each other of what they are doing, daily scrum is for the development team an opportunity for internal planning rather than a status

update of the task. If there are issues that could not be discussed during the short 15 minutes, relevant team members can arrange and solve the issue in additional meetings. Daily scrum plays a vital role - like a 15 minutes half-time break in a football match which can turn around the result - by offering a chance for inspecting and adapting the approach on a day to day basis.

Sprint Review

While daily scrum aligns the development effort with the sprint goal, sprint review aligns it with the whole product development. This time-boxed event is intended to be attended by relevant stakeholders at the discretion of the Product Owner to receive feedback and collaboration from the customers and stakeholders rather than just a status update. The recommended duration for this time-boxed event is of four hours in which time the review on done product backlog items, discussion on a delivery date based on progress, demo, and review of the sprint in itself is conducted. As a result of the sprint review, product backlog could be revised as more inputs from key stakeholders are considered. It is always recommended to have the sprint review even when the development team feels like there is nothing to demo to the stakeholders as it enables transparency while providing inputs for steering the whole product in the future (Ripley and Miller, 2020).

Sprint Retrospective

In the context of the agile software development world, enabling developers to utilize the best engineering practice is one of the major agenda. Sprint retrospective provides such opportunities where the development team, product owner, and scrum master openly discuss people, processes, and tools from the previous sprint with a focus on improvement providing the team agility to adapt processes during the next sprint with the goal of continuous improvement (Marshburn, 2018). This time-boxed event is conducted before the next sprint planning but after the sprint review with the recommended three hours event for one month sprint. After inspecting people, process, tools, and their relationship; the development team identifies key areas of improvements and draws actionable improvement plans in the spirit of the agile manifesto as stated: "At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly (Fowler et al., 2001)".

When the focus is on the software development team, focusing on the effectiveness and accountability of retrospectives is important and could be achieved with documented retrospectives to make it trackable and useful as input for the next sprint planning. Moreover, the effective retrospective is helpful to achieve work satisfaction while improving the development productivity as an approach for software process improvement in agile development approaches (Matthies et al., 2019).

From the perspective of the software development team, the scrum development process starts by collecting the features in the product backlog and moving them in chunks based on the priorities and expected customer values to the sprint backlog which the development team owns and plans daily to change the features into the potentially shippable product increment. The core pillars of the empirical process control i.e. transparency, inspection, and adaptation are achieved by conducting the aforementioned sprint ceremonies while the artifacts and other issues are brought into the discussion amid relevant stakeholders.

3.4 Scrum Adoption

Although Scrum is the most widely used agile methodologies, there is a need for understanding the scrum adoption process as it enhances the ability to tweak the scrum adoption process given the distinguished context in which scrum is adopted. While Scrum adoption challenges explore and guide the scrum adoption process, opportunities encourage and provide hope during the scrum adoption process which would be discussed in more detail following.

3.4.1 Challenges

After going through the literature on the basics of Scrum, it is important to gather knowledge that is relevant to the adoption of the scrum to companies that are developing software and of more or less similar size to the case company as there are always challenging areas during the adoption. According to Eloranta et. al., the scrum adoption challenges can be categorized as either arising due to harmful deviation from practically beneficial practices or not discarding practices that are harmful for the application in given context (Eloranta et al., 2013). While there is an abundance of literature that focuses on critical aspects while implementing agile methodologies and hence often neglecting the aspect of the context an organization and the entities encompassing the organization create around the endeavor of the software development team to adopt Scrum practices. Ozierańska et. al. outlines twenty-three critical factors for agile methodologies adoption under five categories based on project team, psychological and cultural aspects, process and method, environment, and technology. The spatial distribution of the team, remote working, team size, team skill composition, part-time availability of team members, team goals, individual aspirations and attitudes, team chemistry, discipline, members profile, attitude about the workplace, role in the team, business understanding of the product owner, absence of scrum master during the meetings with limited efforts, scrum training, utilized techniques, the team taking several tasks once in Work in Progress, violations of meeting rules, conflicts of business process, team synchronization, limited communication with the customer, influence from outside the team, third party dependencies, continuity of tech stacks, the suitability of current tech stacks are highlighted to affect teams either positively or negatively based on how these factors are affecting the team (Ozierańska et al., 2016). Chow and Cao analyze the success factors and failure factors in the literature and identifies five major critical success factors categorized as organizational structure, process, people, technology, and project factors while defining success based on the perception of quality, scope, time, and cost in software projects (Chow and Cao, 2008). Nerur et al. are quoted to produce almost similar categorizations for successful migrations to agile methodologies i.e. management and organizational structure, people, process, and technology (Ozierańska et al., 2016) - as highlighted by Berczuk with the importance of management support allowing and enabling the development team to embrace the process while figuring out a way for improvement (Berczuk, 2007).

While most research papers discuss some aspects of the challenges in global scrum adoption, Hanslo & Mnkandla reviews the literature and consolidates the list

of challenges after studying twenty-one researches which includes: lack of knowledge/training/skills, organizational culture/mindset, teamwork/communication issues, lack of documentation, budget and schedule constraint, escalating commitment, hard to scale, high management overhead, lack of senior support, work specialization, cross-functional generalist teams, increase stress and workload, lack of quality, lack of top management support, long time to market, low user satisfaction, over-engineered solutions, over-optimistic task estimates, project team size, requirements creep, retrospective inadequacy, and too many meetings (Hanslo and Mnkandla, 2018). In addition to consolidating the challenges, the figure 5 is produced as a Scrum Adoption Challenges Detection Model (SADCM) where nineteen independent variables having either positive or negative impact on scrum adoption are clustered across four constructs i.e. individual factors, team factors, organization factors, and technological factors.

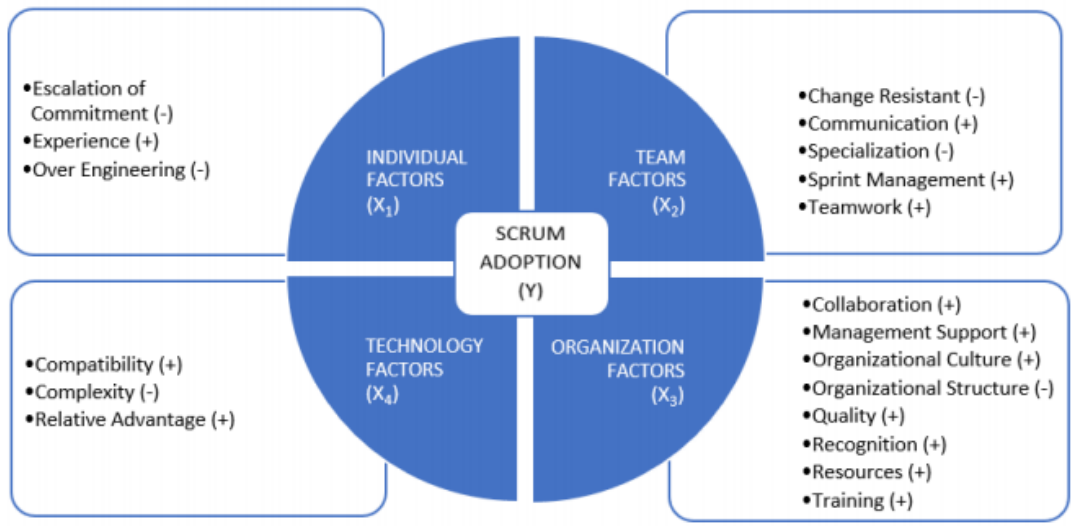


Figure 5: Scrum Adoption challenges (Hanslo and Mnkandla, 2018)

In addition to challenges faced in scrum adoption, anti-patterns are also discussed in the literature and books on Scrum which are evidential in practice after Scrum is adopted. Eloranta et. al. recognized marathon sprint, technical debt in testing, heavy documentation, unordered product backlog, the customer as a product owner, work estimates given to teams, invisible progress, lack of cross-functionality in the team, and customer caused disruptions as anti-patterns in Scrum team (Eloranta et al., 2013). Not having a sprint goal and having the same person as both product owner and scrum master are also recognized as anti-patterns (Ripley and Miller, 2020). Regarding the adoption of the agile methods in general, understanding what kind of projects under what kind of conditions are most suitable to work with agile approaches is also important. As the focus is on building projects around motivated individuals and solid engineering practices, starting scrum adoption could create resistance within the organization (Rigby et al., 2016).

3.4.2 Opportunities

Scrum framework is one of the most widely used agile methodologies. The agile approach, in general, offers flexible planning and execution of the projects while emphasizing constant interaction with customers (Ozierańska et al., 2016) with emphasis on face to face interactions offering improved adaptability and control over risk caused due to over-reliance on rigidly fixed plans (Bhatia et al., 2017). Scrum being one of the frameworks in agile methodologies provides support to the project which is often non-life-critical and requirements are often changing with emergent nature while working in the fast-paced software projects with small teams (Chow and Cao, 2008). In Scrum, the product backlog items are ordered based on the priorities while completing them in the short sprint with constant product backlog refinements. Better visibility of the projects, management of changing priorities, business alignment of IT endeavors, and faster time to market is the key Scrum benefits (ScrumAlliance, nd). Although the Scrum dailies are not for regular updates as the main goal is to organize the whole team for the next 24 hours towards achieving the sprint goal, it provides the opportunity for the development team to reflect on their work while providing the visibility of the works within the development team. In addition to dailies, the Scrum framework provides an opportunity for the development team to improve the engineering practices and improve all aspects of process, tools, and people during the sprint retrospectives. As agile methodology focuses on delivering values by delivering functioning software, it supports the communication between the operation and development team while providing other stakeholders a glimpse of continuously evolving releasable software products. As noted by Humble, high-performing organizations strive for continuous improvements where generative organizational culture is practiced to accept obstacles as challenges to conquer and grow (Humble, 2018). Scrum is indeed a framework that provides organizations with the opportunity and tools to try new things with the openness based on the existing knowledge while striving for improvement i.e. empirical practices based on three pillars of transparency, inspection, and adaptation.

3.5 Distributed Scrum

The Scrum process by book emphasizes the focus on a collocated team (Berczuk, 2007) where the software development team works in an office space with ease of communication with face to face communication being the cheapest channel of communication. Due to globalization and the reach of communication technology and tools, global software development appears to be lucrative when offshore development supports cost-saving, enhances the core competencies, compensates for the lack of resources or knowledge while situating the segment of the development team closer to the market in the globalized world. In addition to having development teams working in a distributed setting, another form of distributed scrum involves individuals working in home office settings regardless of working in a single project as a team or as many teams. While the former brings the issues of agile development in the large in addition to the issues of distribution of efforts working in a single project,

the latter brings the issue only due to the distributed nature of the development efforts. This chapter includes the discussion on challenges and recommendations which apply to both settings rather than talking about opportunities as the situation was forced due to the COVID-19 pandemic.

3.5.1 Challenges

The inherent conflict due to the emphasis of Scrum on the collocated team and distributed work settings is emphasized by various literature. Qureshi, Basheri & Alzahrani quote Scott et al. regarding the challenges software companies face while managing the distributed teams (Qureshi et al., 2018). Sadun highlights the need for detailed documentation and structured communication while mentioning the challenges related to signing agreements, remote access, communication challenges, idle time, motivation and peer feeling, and governance and transparency (Sadun, 2010). In the context of global software development, the distributed teams face challenges to meet the criterion that supports global software development practices like frequent visits, intensive communication with multiple tools, mirroring sites, rotation of senior engineers, and synchronization of work hours (Paasivaara et al., 2009). The difficulties for product owners to transfer the product vision, challenges to share Scrum's visual and physical elements, Scrum meetings challenges, and information sharing challenges are some of the challenges in the settings of home office environments (Luz et al., 2009). While there is no argument about the inherent conflict aforementioned, Berczuk emphasizes on the idea that the distributed nature of the team often highlights the existing process issues rather than being itself a problematic issue while offering recommendations (Berczuk, 2007). Although agile software development does not have strict and formal planning like in the waterfall development approach, planning plays an integral role and face-to-face communication is vital for that reason which is hindered in distributed settings even though several online tools are developed to address this issue (Wang et al., 2010). Regarding the challenges in distributed settings, Ramesh, B., Cao, L., Mohan, K., & Xu, P. outlines 'communication need vs. communication impedance', 'fixed vs. evolving quality requirements', 'people vs process-oriented control', 'formal vs informal agreement' and 'lack of team cohesion' as a context for creating distributed settings more challenging. As an example, informal face-to-face communication is assumed in agile developments which are hindered in distributed settings and the same with the constantly evolving requirements being more planned and fixed in distributed settings. Regarding the people and process, agile development is a people-oriented control process based on team cohesion and motivated individuals instead of process-oriented while there is a need to establish a formal process to achieve control. Moreover, the agreement is formal in distributed settings as opposed to loosely defined contracts in agile settings (Ramesh et al., 2006). Inherent to the distributed setting is geographical dispersion, time and cultural differences (Amar et al., 2019). Qureshi and Sayid highlight the same issues arisen due to the inherent nature of distributed settings i.e. geographical dispersion, time differences, cultural issues, and poor coordination and communication (Qureshi et al., 2018). One particular

aspect of the challenges in Scrum adoption arises from the human tendency to follow scrum practices religiously while forgetting the agile value of focusing on people and interactions over process and tools as more processes and tools are involved to handle the issues in distributed settings (Drummond and JF, 2008). In conclusion, distributed scrum provides opportunities; and in difficult situations like the current situation with COVID-19, the development team is forced to work remotely. Despite having benefits or needs to work in distributed settings, there are challenges inherent to distributed scrum practice due to the differences in assumptions in Scrum with the context created by distributed settings. While most literature discusses challenges and frames the challenges, recommendations are also abundant literature providing guidelines to Scrum adoption in distributed settings.

3.5.2 Recommendations

After understanding the challenges in the distributed scrum, most of the literature focuses on the recommendation for those challenges - while some provide applicable practical tips while others provide some kind of abstract pictures. For example, Sadun categorizes the experience under 'Signing agreements', 'Establishing remote access', 'Overcoming communication barriers', 'Actively managing distributed agile projects', 'Dealing with idle time', 'Achieving motivation and peer feeling', and 'Adapting governance and steering' categorizes while offering the challenges and relevant recommendations. The practical tips include firewall openings, virtual network clients setup, giving offshore people more responsibility based on skills, more physical traveling, exchanging people from offshore to onshore and vice versa, the common language between onshore and offshore, high attention to communication issues, low tolerance for communication blocks and resistance to communication, classifying questions to prioritize which questions need to respond quickly, and peer relationship between onshore and offshore organization (Sadun, 2010). With the focus on incremental adoption, Rayhan and Haque provide useful tips on offshore development such as utilizing tools, focusing on a culture that promotes self-organizing, educating stakeholders about the iterative process, being aware of cultural implication, focusing on bringing change gradually regarding the scrum process, and physical meeting at times (Rayhan and Haque, 2008). As agile is about people and communication, the emphasis is given more on the tools which are required for communication. Cultural understanding and the difference in time zones are often mentioned when the distributed team is located in a different context defined by location and culture. Although Scrum has artifacts that facilitate communication, the iterative process of the development demands communication as opposed to the state-gate approach to development. Berczuk also emphasizes the importance of incremental adoption or improvement in the practices while emphasizing the importance of basing the practice in principles and ensuring that team members understand and embrace the agile values. About the practical aspects of the recommendations, usages of tools like burn-down chart, wikis, and issue tracking to track down the work and communicate, low tech tools like clock, tuned sprint length to match the changing requirements, continuous integration and testing, collective participation in making

decisions like what tool to use, meaningful sprint review to reflect the current state of the project phase, and brief collocation of the team if possible are highlighted (Berczuk, 2007). Moreover, good engineering practices are discussed both to improve the scrum adoption process and in distributed scrum implementation settings too - for example, Ripley & Miller emphasizes the practices like writing tests to check regression failure, end-to-end and unit testing, pair programming, and culture to accept the continuous design and constant refactoring (Ripley and Miller, 2020); and Fowler M. emphasizes the use of continuous integration practices with offshore development (Fowler, 2006). As communication is highlighted as dominant issues in distributed projects and ways of achieving the required level of communication is emphasized, Amar, Rafi-ul-Shan & Adegbile suggests the 5C based theory with five major factors i.e. competency, correlation, comprehension, contentment, and commitment as imperative guidelines while working on scrum-based distributed projects (Amar et al., 2019).

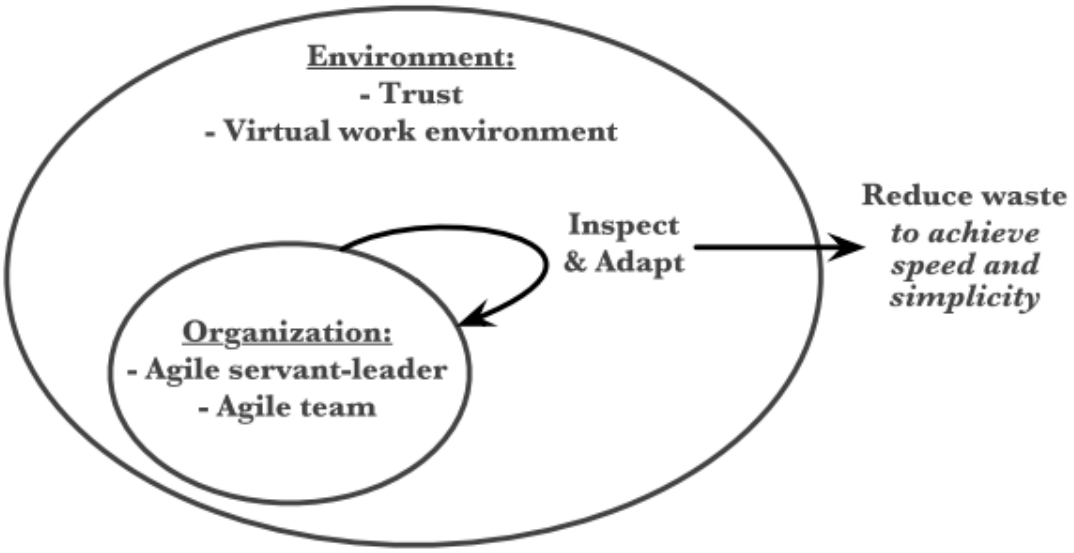


Figure 6: Elements of Scrum to Agile Journey in Distributed Settings (Lous et al., 2018)

To summarize the main points from the literature, it is important to create transparency, inspect and adapt while learning. Although there are challenges while adopting distributed scrum or remote work settings, there are always ways to improve and handle those challenges. Good engineering practices, usage of the tools, understanding of the people and finding a way to get together the team, and improving the communications using tools are some of the recommended ways to thrive in distributed settings while using Scrum as a process management framework. As emphasized by Pauly, Michalik, and Basten, there is a need to tailoring adaptation to make it suitable to development context (Pauly et al., 2015), the demonstration

of the thesis work in the case company should focus on such tailored approach as opposed to following the scrum by the book while continuously inspecting and adapting in the existing organizational and environmental context to reduce waste to achieve development productivity as shown in the figure 6 while conducting the retrospective about distributed work settings itself.

4 Research Method

In this case, the proposal of a research method such as Design Science Research that is adapted to problems in areas such as management would serve to maintain the rigor necessary for investigative research. Most importantly, it might contribute to increasing the relevance of the studies conducted by bridging the gap between what is developed in academia and what is applied in organizations (Dresch et al., 2015).

4.1 Methodology

The purpose of the thesis is to analyze the current state of the problems and figure out ways to solve those problems before implementing the designed measure - hence generating the values to the case company while contributing to the research work. For that reason, a Design Science Approach was used as the foundational methodology in this thesis work. Current state analysis was done using the thematic content analysis although the main purpose of the thesis was to introduce an artifact to solve an existing set of problems. As Design Science Research methodology creates artifacts to solve a particular class of problems as its major contribution to the research, the introduction of artifacts creates a new reality in the context where it is applied - rather than just explaining the existing context (Iivari and Venable, 2009). As described in the DSRM process model diagram 7 given below, the research entry point was context initiated due to the new context in the company as described in the Case Background section. While the nominal process sequence was initiated because of this new context and the initial expectation for the thesis was to improve the continuous integration and delivery practices within the case company, current state analysis and discussion with the software development team in the case company mandated the improvement of the existing Scrum practices. The current state analysis was conducted using the thematic content analysis method, qualitative analysis was used and the collected interview data were coded to find the core problem areas. As a part of the implementation process, scrum adoption guidelines were developed and introduced to the development team which followed the implementation in practice. While the scrum adoption guidelines served as the developed artifact, the implementation of the guideline was the demonstration in practice within the given context. Due to the iterative approach of the scrum framework; the evaluation, feedback, and communication were frequently done within the team although the interview with the development team was conducted as part of the evaluation before communicating the changes that had been made over the period of this project. As described in the DSRM process, the process iteration over the different phases of the process sequence was hence included within the iterative approach of the scrum framework.

Regardless of the research questions, when the researchers' interest lies in producing an artifact to solve a problem as "a means to an end" (Holmström et al., 2009), design science research methodology is one of the appropriate methods. In the case of this thesis project, the pursuit for improvement for the case company emphasizes the need for bringing changes while providing the framework for evaluation for the

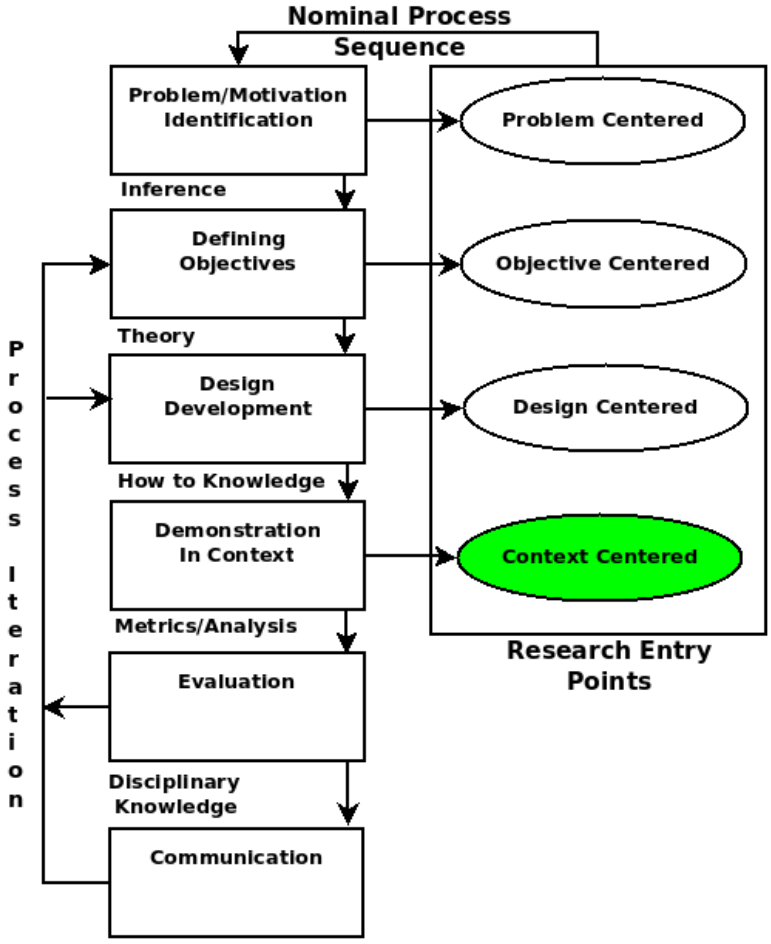


Figure 7: DSRM Process Model (Peffers et al., 2007)

implementation introduced as an ontological basis for exploratory research.

4.2 Data Collection

As the research includes three phases of involvements with the context where the research was applied - i.e. current state analysis phase, demonstration of design in context, and evaluation - data collection was conducted in all phases for different purposes. While some focus group discussion was conducted to create ideas for solving existing problems during the retrospective sessions, the initial data was collected using a standardized template for measuring the maturity level of DevOps adoption in the case company while interviewing the software development team members using a semi-structured format. Semi-structured interviews are suitable for eliciting information with the list of prepared questions while offering flexibility to dig deeper into the issues in a conversational manner guiding the participants (Longhurst, 2003) and driving the discussion in alignment to research interest and understanding of participants. While the context-initiated nature of this thesis process provides the context to create a list of the questions, the suitability of semi-structured interview in

exploring issues with the richness of information from the participants is particularly relevant while the aim of the research process itself is to transform the context where research is being undertaken. As the goal of the research was to transform the context collaboratively working with the software development team, getting deeper into the understanding of the issues faced by both individuals and collectively as a team is important.

In addition to semi-structured interviews, a focus group discussion was conducted during the research process. As focus group discussions are good for concept exploration and generating ideas with the purpose of exploration or triangulation while influencing the members of the focus group discussion (Solcum, 2003), the approach could be utilized in sprint retrospectives while exploring the positive, negative, and improvement area from the past sprints. Since the research approach seeks to establish a sense of ownership to established solutions and values by establishing practices based on informed consensus, the focus group discussion method was utilized while creating the 'Definition of Done'. The time-saving nature of focus group discussion with its ability to synergize the whole team towards optimum consensus if moderated properly is the rationale behind utilizing this approach in the implementation phase of the thesis project. To understand the problems and challenges of Scrum adoption, literature resources were also studied and used as guidelines to shape the implementation as a secondary data source. Despite not having many research papers conducted exactly in a similar context, a lot of valuable insights were available in the literature that guided the initial idea and implementation of the project.

The evaluation part of the research was conducted by utilizing semi-structured interviews which were conducted with five members of the development team who were available. Moreover, a survey was designed and sent to the whole development team using a Likert scale to measure the feeling of impacts in the development team which gathered six responses in total. The interview involved problem discussion and the impact of the adoption process on those problems, understanding of the scrum, and the way for future improvement. With the initial idea of rotating scrum master roles within development team members, discussion about understanding of the scrum is important. After collecting the data in both phases, the data was analyzed in the next phase of the project.

4.3 Data Analysis

In the case of this research, the data analysis was conducted using thematic content analysis by categorizing the ideas expressed in the data to enrich the themes that emerged while utilizing the existing data employing constant comparison during the coding. As thematic analysis is suitable in diverse situations, it can be used as a tool for data analysis for both data and theory-driven analyses (Clarke and Braun, 2013). In particular, data analysis was conducted during the current state analysis phase to determine the root cause of the problem after figuring out five major problem areas. While the goal of the research at that stage was improving the continuous integration and delivery adoption and existing literature was reviewed based on that idea, the presentation of the existing problems with the software development

team mandated the process improvement by adopting scrum at full scale. As a result of that, literature was not reviewed in the thesis process - rather utilized as a guiding knowledge while designing and demonstrating the adoption of the Scrum process. Interesting challenges and recommendations were noted down available in the literature while utilizing them in practice.

The data collected during the evaluation phase was utilized to support or refute the statement that a particular existing problem was solved rather than analyzing the data for any particular purpose. In that way, the evaluation data was familiarized within the light of existing problems - in particular from the problem area which was selected to solve. Despite that, the major idea behind choosing the scrum implementation was to utilize the elements of the scrum to solve the existing problems while discussing the various problems during the scrum retrospective while crossing the boundaries of the selected problems for solution.

4.4 Summary

While the context in the case company provided the case to start the implementation of the design artifact to solve problems with working in a project as a team, the existing problems were drawn by using semi-structured interviews and a standard survey template to conduct the maturity of the current continuous integration and delivery practices. The collected data were analyzed using a thematic content analysis approach and the root cause of the problems was analyzed to be the 'process and visibility'. While the result of the current state analysis and the set of problems from the major problem areas were discussed with the software development team, problems regarding 'process and tools' were prioritized to deal with as a part of the thesis process. Improving the existing practices by adopting all the major elements of the Scrum framework was decided to guide the software development method. The major reason for choosing the scrum framework was the existing use of scrum to some extent already in the case company in addition to having a small team working on constantly changing requirements.

Within the framework of the scrum, different interventions were introduced as a part of the work to improve software quality and developer productivity. For example, the development team started using a single Jira board actively despite working on multiple projects at the same time while the Jira was configured to work the same way across multiple projects the software development team is working on. The Definition of Done was introduced and often updated - while the communication within the development team was conducted more often in team channels than before. The demo of the main project worked during the sprint was demonstrated at the end of the sprint as part of the review. While scrum provided an opportunity to try what works with the development team and supports the goal of the company and project, the development team often neglected the software testing side while focusing more on rigorous reviewing and static code analysis tools.

After the thesis artifact was demonstrated in the context of the software development team, the evaluation was conducted within the development team by using semi-structured interviews and surveys within the development team.

5 Current State Analysis

As described in the 'Case Background' chapter, the intended research was initiated as a result of the growing number of developers in the team - and hence a need to improve the way of working collectively as a team by implementing continuous integration and delivery solutions for upcoming projects. During this phase of the research, a set of questions were asked to assess the current level of practices and the role of developers in achieving the current level of practices in addition to surveying developers about the current level of CI/CD practices.

5.1 Elicitation of Problems

To elicit the problem, a set of questions from a standard template was used for surveying (Appendix B) in addition to a semi-structured interview (Appendix A) - carried out with five developers to figure out the problem developers were facing at the time of interview. Out of the five interviews, four of them were taken on the same day on January 13, while one of them was taken on the 17th of January. The interview and survey reflect the lack of coherence among team members about the knowledge of tools and processes utilized during the software development life cycle and implementation of the existing pipelines. From the perspective of tools, there are plenty of tools set up, but the effectiveness of the tools in its productivity is still insufficient - and developers often tend not to think of tools or processes implemented if it's working as intended and the rationale or the need of the tools being used is not often even discussed in the team. The interview data and survey data were analyzed using the coding as utilized in the thematic content analysis - major problem areas as themes were developed by utilizing the thematic content analysis approach comprised of 'familiarization with data', 'coding', 'searching for themes', 'reviewing themes', 'defining and naming themes', and 'writing up' (Clarke and Braun, 2013) by utilizing the data expressed in the transcribed interviews. The Qualitative Research approach was employed with semi-structured interviews as a way of data collection to enter the world of participants and to understand the problem in the given domain from the participants' perspective. Based on the listed codes, themes were formulated to explain the core area of problems being faced in the case company. The result of such a process is portrayed in figure 8 while the description follows after:

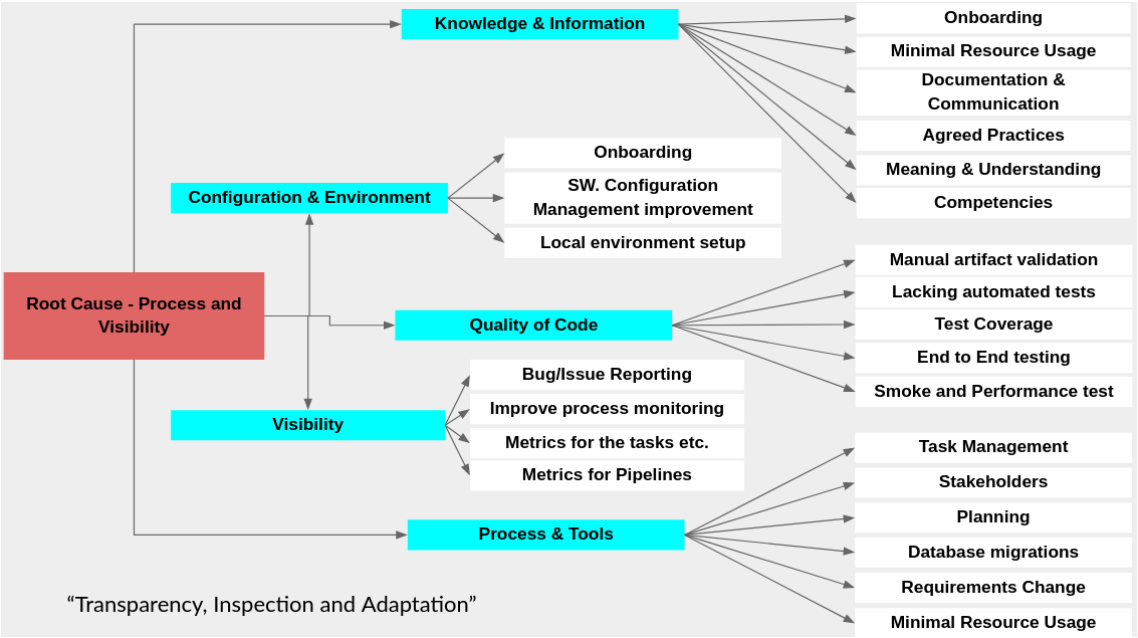


Figure 8: Problem Representation

After analyzing the interview results, five core areas of improvement were highlighted with the root cause being the "Process and Visibility" - fundamentally important for transparency which is a key pillar of empiricism. The five core areas of improvement are described in detail as follows.

Quality of Code

Quality of Code is one of the reasons for implementing the continuous integration and delivery pipelines besides achieving developer productivity. While it could be achieved with code review, test-driven development, static code analysis and implementation of pipelines; quality of code is still suffering due to various factors. Despite one developer mentioning senior developers about pull request guidelines, the case company does not have any particular guidelines over which team has the ownership and adheres to it as its own coherent set of practices.

The CI/CD pipeline includes testing but that phase is allowed to pass even without any test in the code and without being enforced by set coverage level. In addition to that, the development team has not even discussed what quality of code means for the team and what could be done to achieve such quality. A lot of time is spent in manual testing particularly as part of the review jobs to make sure that functionality is working as intended while at times not realizing that the other parts of the system are broken consequently without our notice. There are only unit tests and the amount of tests differs significantly among projects running at the same time even when there is no standard set for quality. Projects lack smoke tests, health tests, or any kind of performance tests although our overall goal is to improve the quality of the product and the code itself.

The team does not have any set of 'Definition of Done' and hence there are no set checkpoints to check the implementation against - often time developers wondering if writing a test is as important as getting other tasks done instead. End to end

testing is now work in progress but it is not implemented in any of the projects. In addition to these conceptual problems, developers pointed out the following concrete problems:

- Code repetition and copying
- Lack of self-documenting code
- Documentation in code with comment and doc-strings
- More tests using cypress
- No test coverage tools implemented
- Ways to integrate testing into workflow missing
- Determining the priority for testing different areas, features, or projects

In conclusion, the software development team is lacking a proper discussion about the quality of code and having an agreed level of tests and coverage to keep the product in a releasable state. Some good practices include the implementation of code integration and code reviewing before merging to the master branch of the project repository. Code review does not make sure that bugs are not introduced while merging the feature branches into the master branch even when code review is supposed to improve the overall quality (Kononenko et al., 2015). Test-driven development is often heard within the development team, but any concrete steps on adopting this practice are not yet taken.

Knowledge and Information

Knowledge and information are vital to achieving both developer productivity and quality of code. The importance of information is particularly visible during the onboarding process - which means when a new member joins the team or when the whole team moves on to the next project. Having this kind of information would be also useful to feel the void created if some developers leave and the task to maintain the existing code relies on the new members of the team.

In addition to lacking proper knowledge management, the team has a lot of tools used during the process which are not properly utilized and the team has not even discussed on way to better utilize these tools. There is a lack of resource utilization in a way that the team has not achieved maximum value through the same tools that are utilized in the practice. Most importantly, there is a lack of agreed practices within the development team - for example, there is no recorded discussion on what the commit messages should be like and the commit messages for build failures fixes are mostly meaningless and provides very little sense of what was the reason for the build failure. When developers have different aptitude and skill levels, setting up common practices or putting effort into developing those practices not only helps the developers in their early stage of career but also the whole team would be benefited. Some developers also mentioned the problems when working with their branch because some other developers have already pushed the changes to that branch without notifying anyone which reflects the lack of agreed practices with

the evidence of surprise. During the interview, some developers suggested having exploratory testing together in the developer team to better understand the whole system and find bugs that otherwise might be neglected.

Competency is one of the aspects of knowledge and plays an important role in improving developer productivity. The case company provides a flexible work environment and provides flexibility to its employees. Despite that contextual setting, some developers feel uncomfortable spending time learning new things at work. Discussion on the team and making what the company offers to employees visible would shed light on the issues that developers might be on the dark side regarding self-development and team development. Moreover, this kind of discussion will promote common understanding and sharing the meaning in the same way as others. The survey reflects the mismatch in the understanding of the current process and tools and how developers understand different aspects differently which highlights the need to promote discussion as a team.

Additionally, significant improvement is required in documentation and communication as Slack is used mostly in personal communication rather than discussing in team channels making some developers have the opinion that there are no collaboration tools at the team level. Many developers think that information is scattered among different channels and sometimes in channels where common things and mostly random stuff are shared consequently causing valuable information to be lost in the world of messages. Moreover, documentation is scattered all around or sometimes not even updated immediately making some developers feel like they are wasting time to figure out something just because someone changed something and have not notified of the change and how the change affects the existing setup. The following specific problems were mentioned during the current state analysis phase:

- Onboarding tutorials
- Local environment setup missing
- Lack of knowledge on how to deploy to testing, staging, and production environment
- Knowledge gap between frontend and backend developers
- Some projects are documented in confluence and some are documented in code repository causing a lack of consistency
- Activities and times for learning and doing new things.
- Personal communication than team communication
- No information flow (what are planned changes and what people have changed)
- Poor online cooperation as a developer might have to wait until face-to-face meeting in the office
- Definition of Done is missing

- Commit message guidelines
- Not enough discussion before next project
- When there is a big design update, there should be a bit of storytelling on how the design works

Since scrum focuses on cross-functional team and software development team is working based on skills and knowledge, it is highly important to break the organizational functional silos and promote internal cooperation over competition. While available frameworks are dealing with knowledge management principles like the four pillar of knowledge management i.e. Leadership, Organization, Technology, and Learning (Mohamed et al., 2004), the major focus of the development team is on discussion, learning, and internal cooperation within the team to promote knowledge and information sharing in the spirit of cross-functionality and self-organization.

Configuration and Environment Configuration and environment provisioning are some of the problem areas which could be improved but possess challenges as well. First of all, virtualization is not used in local development setup while testing, release candidate, and production environment is completely running in the Kubernetes cluster. Developers sometimes face a situation when the code works in the local machine and not in the testing environment - and mostly because of not having enough understanding of how different environments work in the bigger picture.

Not discussing before starting a new project complicates how the setup of projects and how could the development team sync the whole development setup to be consistent with the other environments. The onboarding of the team members would be smoother by doing so - and it also improves the practice in the new project than repeating all the good and bad practices from the previous project as often observed in the existing projects.

Regarding the software configuration management, some of the configurations are done by configuring the environment while the others are included in the code. Some frameworks better support having the configuration in code, but not having enough discussion about how configuration is managed would shed light on how the whole deployment setup is working. Specifically, the following aspects of the problems were mentioned during this phase of the research:

- No discussion on what good practices - even for configuration management - could be taken to the future projects
- Missing guidelines and documentations
- Development environment is set up differently than other environments
- Manual setup of the local development environment.
- Not all developers know release management, versioning, and git workflows
- Not used virtualization in the local development setup

– Software configuration not discussed enough

Version control is being used by the software development team and it has its role in software configuration management. Some technological frameworks support the configuration management in a way that it could be maintained along with the code while other frameworks might need other library support. Workspace management is done privately although some curious programmers are always wondering about other developers' approach to workspace setup - and primarily either Mac or Linux based development platforms are utilized. This does not necessarily mean enforcing the same setup rather to develop a mindset that seeks constant growth as a team. Having a discussion about software configuration management in an agile team is about adopting robust configuration management implementation to enhance the values provided with agile methodology (Moreira, 2010). In conclusion, there are tools and automated continuous integration that supports the software configuration management, but the software development team lacks the necessary cross-functionality required to work as a Scrum team.

Process and Tools

Regarding the process and tools, the development team mostly underutilized the available platform and tools which could be improved by looking for ways to better utilize - instead of relying on individuals' ability to figure out how to use it because most of the tools are used also as a team and optimizing the workflow of the tools around the team would improve the intended goal of the company i.e. developer productivity. The issue about task management in Jira appeared frequently during the discussion - some developers mentioned the inconsistencies in the use of Jira by mentioning how the bug reports coming through the personal communication channels are registered as an issue in the Jira if remembered to do so. As the software development team was working on multiple projects at the same time with separate backlogs for each project, the configuration of Jira with different board settings across multiple projects was often confusing. Moreover, templates for bug reporting could be used so that internal testers in the case company can file the bug reports in a way that communicates meaning to the development team. One of the issues discussed was the frequent requirements change.

Some of the concrete examples of the problem which surfaced during the interview in this phase are given below:

– Not utilizing the possibility to discuss on the Confluence or in the pull request review

– Process and monitoring tools are set up but the data are not utilized

– Confluence have tools for planning release, but not utilized in the team

– All the bugs from different products and different deployment environments are coming in the same Slack channels.

– Bug reporting template is missing - often taking more time for developers to figure out what the bug is even about.

- Requirements change in the middle of the project and quite frequently
- Missing proper channels or tools for getting bug reports from internal testers
- Some tools are not useful - for example, Sentry is not reporting bugs from the frontend applications
- CI/CD pipeline creation is considered as a single task instead of breaking it down into multiple tasks keeping other developers on the dark side of how it is accomplished.
- Stakeholders are not present in any meeting with the development team - and not made part in any sprint events.
- Planning could be improved so that the big feature branches could be trimmed down to be manageable enough task
- Database migrations are complex and often neglected

As described in the current state analysis, one of the issues in the software development team was not having a discussion on what kind of Scrum practice was being used - and hence, not having for example any retrospective or sprint reviews included in the practice. The whole process was missing what to add in the practice, or what to remove to improve as a development team as tweaking and seeing what works and what does not is vital to agile practices. Consequently, both process and toolset were used sub-optimally despite the case company providing support and tools.

Visibility

Like the process in itself, visibility is cross-cutting aspects as it touches all aspects of the issues previously discussed and plays a vital role in achieving the much-needed transparency in the empirical software development process although this topic is discussed only around the finding during the current state analysis phase in this section. As it is mostly bringing out issues in the light for the whole development team, it involves many issues like metrics of the CI/CD pipelines, metrics for the tasks, tracking of the bugs or other issues, and improving the process monitoring. Some of the concrete examples which appeared during the survey and interview are listed below:

- No discussion or visualization about how to measure the CI/CD pipelines itself - even though for example, there is constant refactoring for improving the performance.
- Tasks estimation is missing
- Bug and issues are not tracked - some bugs are just fixed and forgotten
- Build failures are not visualized and discussed

– There are some logging and monitoring tools, but developers might not even consider going through those tools or getting an understanding about the measurement

As transparency is one of the pillars of empirical process control, gaining visibility - i.e. making aspects of the process affecting the outcome to all stakeholders involved (Cho, 2008) - is particularly important. Since the existing process did not involve sprint review and demo, the iterative development did not demonstrate the evolving nature of the product particularly to the stakeholders who had business concerns regarding the projects besides hearing from the proxy product owner about the status update. Additionally, the software development team lacked discussion on what are important metrics and what does the metric represents in practice, hence hindering the visibility within the team itself.

5.2 Presenting and Brainstorming more Problems

After eliciting the problems, the next step in the process was to conduct a workshop with the whole development team where the initial findings were discussed after explaining the purpose of the workshop. As not all developers were included in the interviews, a quick round of brainstorming was carried out to enrich the already established themes. Moreover, the rationale for conducting the workshop within the development team was to create a sense of ownership over the improvement endeavor. The initial findings were qualitatively analyzed using a thematic content analysis method for making sense of data by categorizing interview data into different themes by using classification, summarization, and tabulation process for data-driven analysis. The root cause of the problem was concluded based on the linkages established between categories.

For the workshop part of this phase, two more developers joined the program - one of the developers remotely. The remote developer was included with the whole development team using the Google Meet while the related screen from the computer was shared with. To get organized around the problems as a team, there is a need to brainstorm problems together as a team besides discussing existing problems. To keep the discussion within the limited time and to make all developers feel included even though not all were interviewed, the brainstorming phase was included to enrich only those five key areas of improvement.

As a result of the brainstorming, following additional concrete problems were presented while most of the ideas were similar to the finding during the interview phases besides scrum explicitly mentioned as inconsistently applied to our projects as an issue during this brainstorming phase. Moreover, the software development teams suggested the use of statically typed languages like TypeScript, using test-driven development practices, and regular review of backlog to constantly refine the backlog to have only the most prioritized items in the list.

5.3 Conclusion of Current State Analysis

As the context of the research was initiated due to the increase in the size of the development team, there was still a need to find out the problem areas where improvements are needed with the focus on software quality and development productivity as guiding requirements from the perspective of the case company. The current state analysis was conducted as a stage of problem identification in the design science research methodology. The software development team works as a self-organizing team of highly motivated individuals in Scrum. Gathering problems as a whole team, analyzing the results, and presenting the result to the whole team is founded on that foundational assumption. Despite the initial inclination to improve the continuous integration and delivery practices in the case company, the current state analysis results in five key areas of improvement with 'Process and Visibility' as the root cause of all problems. In conclusion, the current state analysis highlighted 'Quality of Code', 'Knowledge and Information', 'Configuration and Environment', 'Process and Tools' and 'Visibility' as key problem areas with root cause 'Process and Visibility' as the major issues linking all other issues in the given five areas. As part of the problem identification phase of the design science research methodology, current state analysis was conducted and the whole software development team was included in one or another stage of the process to create a shared sense of responsibility towards the solutions to the selected problems.

6 Design Artifact Description

This chapter of the thesis presents the approach to solve the problems and the rationale behind utilizing such approaches. As a part of producing artifacts, first, the team discussed the problems found during the current state analysis phase and then proceeded with the discussion to select the problems to tackle before discussing possible approaches to solve those problems given that the major focus is on code quality and developer productivity. The main approach to solve the problems was to improve Scrum practices and utilize the retrospective in particular to implement solutions to existing problems with the following discussion elaborating on this process in more detail.

6.1 Problems selected for the solution

After detailing more problems, the team was requested to vote on the elicited concrete problems - if they are relevant for achieving our goal of code quality and development productivity based on its significance and our ability to solve collectively as a team. During the same workshop day, voting was conducted after the problem was discussed on its significance and meaning in the context of the software development team. Based on the voting, only issues from three major areas of improvement were selected out of the five areas figured out during the interview i.e. 'Quality of Code', 'Knowledge and Information', and 'Process and Tools'. After the discussion with the team, the supervisor at work was requested to select the problem area to focus on solving the problems given the situation in the case company.

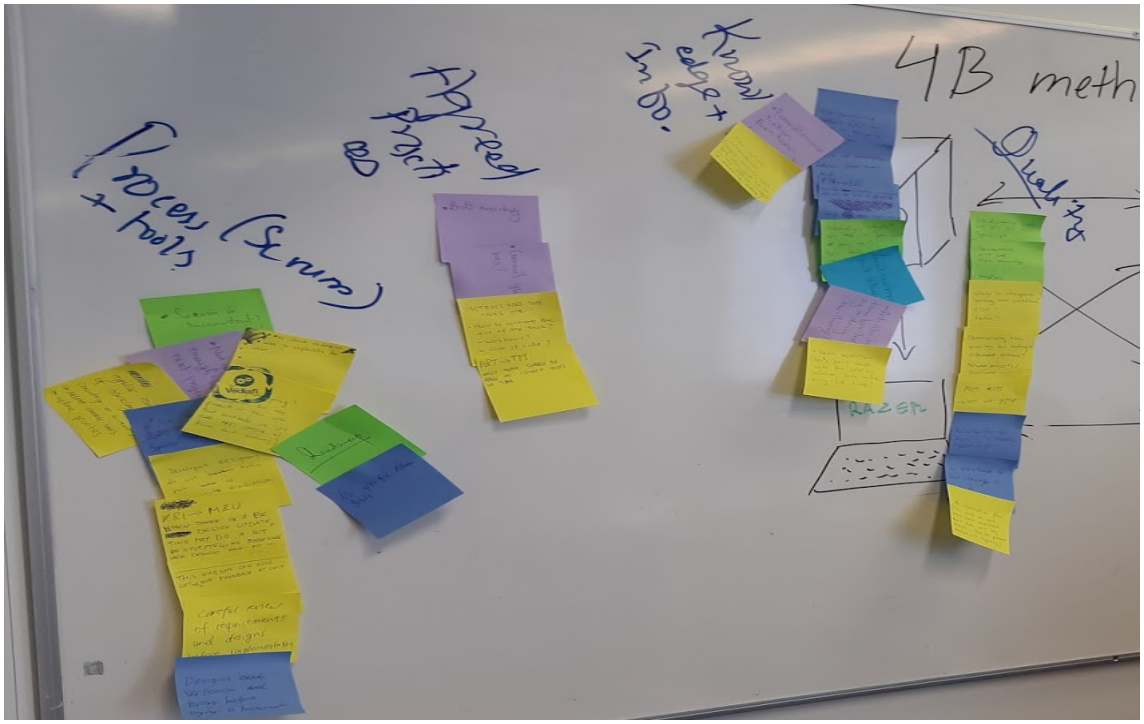


Figure 9: Selection of Problems

As shown in the Figure 9, out of the five key problem areas, only three were selected. Just like during the current state analysis phase, knowledge and information was aggregated with agreed practices as knowledge and information could be generated based on the agreed practices. Although agreed practices could be in itself a cross-cutting aspect touching different problem areas, having a set of agreed practices to enhance the knowledge and share the best engineering practices was of utmost importance in the beginning of this thesis project. With that combination of 'Agreed practices' and 'Knowledge and Information', following concrete problems were selected from the given three key problem areas as shown in the table 2:

Table 2: Concrete Problems in Three Key Problem Areas.

Quality of Code	**Knowledge and Information**	**Process and Tools**
Code repetition	Onboarding tutorials	Inconsistent Scrum use
Self documented code with statically typed language	Workspace setup	Requirements review and design before implementation
Documentation in code	Commit message guidelines	Verification and approval of design before starting development
Integration of testing workflow	Lack of knowledge between frontend and backend	Not enough planning in sprint planning
Prioritization of testing areas to define good enough product for release	Knowledge on how deployment works on different environments	Developers and designers in requirement discussions
End to end testing with cypress	Lack of consistency in using documentation (bitbucket versus confluence)	Metrics for task (estimation etc.)
Test coverage tools	Free time for learning	Ways for internal testers to report bugs
-	Personal communication over team communication	Scrum team owns the Jira board
-	Communication	Clearly defined responsibility
-	Definition of Done missing	Regular review of backlog
-	Poor online cooperation	No prototyping
-	Onboarding discussion for next project	-
-	Design update briefing	-

After selecting the problems within the development team, the team discussed the possible approach to solve the existing problems. As the root cause of the problem during the current state analysis phase was 'Process and Visibility', the discussion was mostly focused on solving the root problems with the rationale that the other problem will be solved as well. The opinion of the workplace supervisor was included in the discussion and the opinion of the team members was heard of. After the discussion, the problems regarding 'Process and Tools' were selected for getting the improvement process moving on and was approved by the workplace supervisor as well. Based on that selected problem area, the solution approach was discussed as described in the next section.

6.2 Mandated Approach to Solution

During the discussion in the workshop, the discussion on the team regarding the problems and their details in the context of improving development productivity and software quality illustrated the need to improve the use of tools and the adopted process. As a conclusion of the discussion, the team decided to fix some of the issues immediately - like talking about development issues in the team channels rather than personal discussion in direct messages. For solving the problems related to the problem area 'Process and Tools', the team decided to experiment with the Scrum and start utilizing Scrum by book to start with instead of just having daily and short and brief planning. Additionally, the team decided to have a review with salespeople immediately after one of the projects is accomplished. The team also decided to create a Slack channel to discuss general technical issues and use message threads instead of new reply messages to make the use of Slack channels more like a Question and Answer channel. As a part of implementing Scrum on a greater scale, the Scrum Blueprint was required to be produced as a part of the thesis work which was used for explaining the development team about Scrum and guiding the overall process of Scrum adoption. One of the rationale to start with the Scrum by book was to include the Scrum Retrospectives to discuss and figure out ways to improve engineering practices while deriving approaches to solve the existing problems in the team.

In conclusion, the team actively participated in problem identification and defining the objectives of the endeavor while brainstorming and mandating the Scrum adoption process even though the current state analysis was conducted to improve the current continuous integration and delivery practices in the case company. In addition to that, the Covid-19 pandemic also forced the entire team to work remotely during the demonstration in the context phase of this study necessitating the study of Scrum in distributed settings.

6.3 Recommended Scrum Practices

After deciding to adopt the Scrum in its improved form in the case company, Scrum Blueprint was introduced with recommendations as mandated by the software development team. The development team was introduced with the Scrum by book with the commitment to try what the team decides, see how it goes, and evaluate if to

continue or not.

Concrete actions taken are listed following:

- Scrum process followed by the book with the agreement to modify if some practices do not make sense while considering that almost half of the developers work mostly part-time.
- The role of Product Owner and Scrum Master would be assigned to different individuals as it facilitates the necessary dialogue between Scrum Master and Product Owner.
- Single team single-board regardless of the multiple projects development team was working on - the development board was consolidated by issues in active sprints in multiple projects. In its optimum use, the Product Owner creates the product features as backlog stories and the development team creates the tasks to accomplish those stories from the development team board (or by going to the respective project board).
- Single team single board rules in Jira to remove the confusion between projects. The global rules were applied to all projects in Jira so that the development team members have the same feeling regardless of which projects board they are checking out.
- Inclusion of planning, review, and retro events and making daily like dailies.
- All the works done by the development team members would be logged in the Jira which not only promotes the visibility of the work the development team is working on but improves traceability and knowledge sharing.
- Creating the definition of done to achieve potentially shippable artifacts by the end of each sprint.
- Clearly defining the goal of the Sprint particularly when the development team has to prioritize even within the projects in the middle of the sprint.
- Utilizing the scrum dailies to raise the major concerns and take time later if needed for further discussion.
- Involving and engaging domain experts and designers and other stakeholders with the development team if and when needed.
- 80/20 rule of planning - the available time is not fully allocated to the development team.
- Documenting sprint retrospectives and creating accountability on the sense that actionable items are achieved by the team members.
- Utilizing sprint retrospectives to review the problems discussed during the current state analysis in small increments.

– Distributed Scrum practices were not explicitly discussed in the beginning although some of the practices introduced during the retrospectives could help to tackle those issues.

In conclusion, the scrum adoption was started following it by the book while being open to change as per the needs of the development team. Since the development team has often changed requirements and priorities among multiple projects running at the same time with almost half of the members in the development team working only part-time, long-term planning was always challenging and hence it was dropped and more focused on sprint goals and its fulfillment. In addition to that, all the sprint events beside daily were conducted by the end of the Sprint on the same day when all the development team members were present. With this work mainly focused on the development team rather than considering the transformation of the whole organization, particular focus was paid on improving the development productivity by figuring out ways of improvement during sprint retrospectives with the agreement on utilizing scrum to experiment and learn while solving the existing problems in the development practices. Given that the software development team is willing to grow and learn while already demonstrating commitments to more productive development work and the team; with the nature of the problems explored during the workshop, adoption of the scrum will help to solve most of the problems despite some of the challenges while experimenting and learning about Scrum itself.

6.4 Summary of the Retrospectives

Before having the concluding retrospective to refine the definition of done and create a solid engineering practice, there were altogether eight retrospectives during the thesis process. In the following sections, the main points of the sprint retrospectives are described per each sprint retrospectives.

Sprint Retro 1 (2020-04-23): Although this was the first sprint retrospectives, the team discussed solving the existing problems while improving the team productivity and quality of the code. The development team agreed on some initial template for the pull request review which is more detailed and easily traceable to the issues related to the features. Non-functional requirements apply to the entire product which could be applied by creating 'Definition of Done' (DoD) items that apply to all backlog items (Merkow, 2019). The initial version of the DoD was created with the development team during this retrospective. Estimation was always the problem, and particularly not cared enough when multiple projects were running at the same time often changing priorities in the middle of the projects. Moreover, the concept of main reviewers was also introduced which helped to have many reviewers who can check the idea of the feature and implementation while giving the main reviewer the responsibility of checking the pull request in more detail. As planning was missing in detail, time estimates were mandated to be estimated with the developers who were present in the design meetings. To consistently use Jira, the development team agreed to create stories for features while having sub-tasks as the breakdown of the story by the development team to get the story done. To improve the testing, the feature's implementation was decided to include the tests also.

Sprint Retro 2 (2020-05-07): In this retrospective, the focus was on improving the workflow while working remotely while briefly reflecting on the previous retrospective. As already mentioned, the development team was already working as a collocated team before going remotely while one of the members was always working remotely. To improve the remote working, the development team decided to have an optional coffee break in the Google Meet daily at 14:00. Furthermore, the team decided to present the idea of being at home the way as being at work with the proper office settings and dresses. Most of the development team members who were working from the office before experienced being distracted during the remote working - and ways to handle distractions were discussed. While the entire planning was virtual, it was decided to discuss more if some of the stories are needed to discuss while mostly preventing from having team discussion when not all team members have concerns. For that reason, the team also suggested having a better description of the tasks with an appropriate label to differentiate between frontend and backend tasks while highlighting the need for a standard for stories that involve both frontend and backend works. For the same reason of not wasting a lot of time for conversation, time estimation was left on the hand of the developers which at least provides an overview of the next 24 hours of work. In case the pull request review becomes lengthy, the main reviewers and the developer were suggested to arrange a meeting over Google Meet to figure out the pull request fixes quicker. Not having dailies was one of the suggested ideas which were not included in creating action plans. Spending more time on sprint retrospective and brainstorming what works better in the remote working situation was one of the suggested recommendations. One of the developers also requested not to add tasks in the sprint as it changes the scope of the sprint without the team being aware of the changes while pointing out the need for including subtasks to the story if the proper implementation of the features could not be achieved without doing extra work. Moreover, providing how many hours of work each member of the development team is available over the period of the sprint was discussed as available working hours directly affects the scope of the sprint.

In conclusion, the team decided to have coffee at Google Meet, synchronous pull request review over several comments in the Slack channels, working only from the sprint board, and more planning before discussion with the whole team during the sprint planning to save the development time.

Sprint Retro 3 (2020-05-21): The goal of this retrospective was to get committed to the plan and in its execution rather than only planning and not following through it with actions. As Scrum values are fundamental to keeping the Scrum in action, the team discussed the values and what value each developer in the team is committed to holding in some form of measurable actions to reflect the commitment of that value. With that context, the DoD was revisited with the team while discussing if it is being applied in the current practices. In particular, writing tests and time estimation in the tasks have been the most challenging part of the development team.

In conclusion, the team discussed Scrum values i.e. commitment, courage, focus, openness, and respect, and actions like daring to ask in team channels and making

the work transparent using Jira boards, etc. Moreover, the same improvement actions were decided to keep working on from the previous two sprints instead of figuring out improvement actions - focus on committing on doing great things until it sticks.

Sprint Retro 4 (2020-06-18): To highlight the most urgent cases, this retrospective was limited to a very short time with the main focus on creating two powerful words - one to describe the positive aspects from the past sprint while the other one to describe the negative aspects. The words thus collected were discussed in the next round where each development team member was allowed to explain the words in one minute each which followed the discussions on how to change those words into positive actions. While scope changes, the complexity of the requirements, new technology, and fixing the implementation before the demo was mentioned on the negative sides; demo, progress, teamwork, communication was highlighted on the positive sides.

Since not all members of the development team work fulltime as mentioned before, all the sprint events were conducted on the same day causing meeting fatigue to the development team members. During the sprint, the development team came up with the idea of demonstrating the artifacts to the product owner; and the demo was done for the first time as a part of the review of the sprint followed by the retrospectives the day before the next sprint planning day. As an actionable item, a recurring calendar event was created where the development team demonstrates the artifact to the product owner at the end of the biweekly sprint.

Sprint Retro 5 (2020-07-01): After conducting the sprint review with the demo again, the development team continued to discuss the weather from the previous sprint. Scheduling meeting without clear agenda, leaving integration or implementation to the last day, and not taking problems and owning them, huge changes in the backend implementation while the frontend is still integrating the backend were mentioned mostly on the cloudy weather while the sunny side included demo, snacks and drinks during the demo, and API design meeting for required endpoints.

As the list of actionable items, the development team decided to create the Slack channels dedicated to announcing changes, scheduled the API design meeting after the sprint planning, and getting snacks and drinks for the demo while making the previous day for review and retro while the next day for the planning.

Sprint Retro 6 (2020-07-15): At this point of the time, all the project boards in Jira are already set to use the same approach from the perspective of the developers - effectively implementing the single board single development team approach. Moreover, Jira automation and rules are applied to make the estimation and time spent necessary while feature stories requiring the story points. The story points are discussed within the team for the features based on the Fibonacci sequence. To better utilize the time, the development team has decided to call the designer for design briefing while providing an overall understanding of the product. The frontend developers decided to split the frontend tasks after the sprint planning (which only discussed the what), while the senior backend developers for the endpoints and backend implementation. Moreover, the team decided to use tagging in the Slack to make sure that all backend developers or frontend developers get the attention to the

issues raised. Since the development team felt the improvement in communication between frontend and backend, the need for keeping up that spirit was highlighted during this sprint retrospectives.

Sprint Retro 7 (2020-07-29): As most developers were on summer vacation, the sprint retrospective was shorter than the others while it reflected the lack of preparation and no need for drastic improvements. One of the constant issues as a result of having a demo part of the sprint review was backend frontend integration. The development team discussed the need for the backend pull request reviewer to use the frontend apps to check the pull request if the pull request concerns the changes in the implementation while not necessarily for new endpoints implementation. Regarding the software quality and test, the backend includes tests while the frontend is missing the test - which is mostly neglected due to the workloads for frontend team members. The development also discussed the need for the backend developers to help the frontend developers if needed. Also to get the product demoable, the team decided not to merge the breaking changes before the demo while keeping some buffer time for miscellaneous fixes before the demo. Furthermore, the development team set the next sprint retrospective as an opportunity to go through the results of all the previous retrospectives and update the Definition of Done and decided to request the designer to attend the sprint planning.

While the thesis focused solely on the development team, the sprint retrospectives provided opportunities for the development team to improve the engineering practices and discuss possible problems and approaches to solve those problems while focusing on providing values to the company and customers efficiently and effectively. Some of the existing problems were solved while the estimation and plannings are still problems if Scrum is thought about doing by the book, but most importantly the culture of the demo was established in the case company that the demo within the development team is grown beyond the development team to include the whole organization.

7 Evaluation

As shown in the DSRM process model, evaluation is a fundamental phase of the design science research which as quoted Peffers et al comprises two activities i.e. demonstration in context and evaluation. While the former demonstrates the feasibility of the artifacts in the context, the later provides the relevance of the solution to solve the problems (Ostrowski and Helfert, 2012). Instead of utilizing the metrics and analysis in this project, a qualitative semi-structured interview was conducted with five developers in addition to utilizing the survey to collect the evaluation data with the goal of the qualitative evaluation of the adoption process to evaluate the effectiveness in solving the existing problems discussed in the current state analysis phase. As discussed in the current software development practice topics, the software development team was already utilizing the scrum practices to some extent indicating this adoption process as evolution instead of new implementation in the context. When the system dimension involves evolution, the artifact evaluation criteria are robustness and learning capability of the system which highlights the ability of a system to respond in changing context with the ability to learn in the given context (Prat et al., 2014). Consequently, the evaluation of the implementation of the artifact in this project focuses on the following three questions:

1. Does it solve the selected and other existing problems?
2. Could it be used to solve future problems?
3. What could have been done better in the future and the demonstration phase?

In the following subsections of this section, these questions are discussed in detail concerning the perception of the developers in the software development team who were interviewed. While the thesis focused on the holistic improvement of the overall process, the evaluation reflects the holistic evaluation based on the perception of the developers interviewed instead of focusing on a more quantitative approach.

7.1 Evaluation based on Problems

While the following Table 3 uses ✓to mention the concrete problems handled during the thesis process and ✗to mention the problems not discussed, the table is followed with the discussion of these issues in details while elaborating other findings regarding the problem areas.

Scrum was consistently used with sprint reviews and retrospectives introduced. In addition to adding additional scrum events into the process, a single development board was created to reflect the work development team was going to undertake during a sprint in a single view. One of the development team members mentioned the absence of a task without assignee in the progress as opposed to before - which also reflects partly the result of a regular review of the backlog. During the problem analysis phase, one developer mentioned the issue of clearly defined responsibility - while frontend and backend people are working in the team - these aspects of the

Table 3: Tackled Problems in 'Process and Tools' Area.

Process and Tools	**Evaluation Discussion**
Inconsistent Scrum use	✓
Requirements review and design before implementation	✓
Verification and approval of design before starting development	✓
Not enough planning in sprint planning	✓✗
Developers and designers in requirement discussions	✓✗
Metrics of task (estimation etc.)	✓✗
Ways for internal testers to report bugs	✗
Scrum team owns the Jira board	✓
Clearly defined responsibility	✗
Regular review of backlog	✓
No prototyping	✗

problem was not discussed instead focusing on working on problems as a single development team. As a result of hiring a part-time designer in a full-time role, the designer is more accessible to the development team than before. Although it is not consistently practiced, the development team highlights the need for design meetings to happen often instead of having the meeting while a new project is introduced or the design undergoes major changes. The development team applied different tools while working remotely - and applied effort to best handle COVID-19 situations. Yet, one of the developers mentions the improvement in teamwork as a collocated team during this adoption process. While the interaction with the designer has improved, interactive prototyping is not yet considered. The development team highlights the better grasp of development efforts for the next sprint - due to all tasks being visible in a single board with clearly formulated sprint goals and regular discussion of the sprint backlog items still unassigned. Regarding the planning, there is mixed feeling in the development team - while they agree on the improvements, they see it as a major area that could be significantly improved.

In addition to these problems; introduction of change-log and endpoints documentation, additional automation, less repetition of code, improvement in semantic aspects of the code, mutual learning, standardized PR template, additional UX tools, and ability to work in multiple projects as a single team more smoothly were often mentioned. Despite having a focus on developer productivity and software quality, software testing is a bit of gray area but understandable due to the constant pressure to deliver in multiple projects. At the beginning of the adoption process, DoD was established at least at the PR level which was partly already enforced due to the evolving CI/CD pipeline and further defined in the PR template. Personal communication over team communication was one of the major problems in the development team which seems to work inconsistently as development team members often highlighted the need for frequently reminding about the importance of communicating in team channels.

7.2 Understanding of Scrum and Future Application

Most of the development team members mentioned the practical knowledge of the scrum while some development team members already possessed the experience and knowledge of working in the Scrum team. The importance of scrum in supporting the daily work while providing a better focus on priorities was highlighted by some development team members. The PO role as a proxy between stakeholders and the development team was reported to improve the productivity of the developers and highly regarded in a positive note. While there was a need for the process to provide structure, rules, agreements, and approach to everyday work and project, the importance of Scrum in bringing people together in an organized way was particularly resonant to the spirit of Scrum which focuses on a collocated team of highly motivated individuals working in a self-organized manner. The importance of Scrum values and the structure provided by the Scrum was reflected in the developers' understanding while neglecting the potentially shippable increments and customer values as the process was primarily focused on the software development team. On the other hand, the case company is growing more customer-driven with the growing number of end customers' products.

In conclusion, most developers would like to continue using Scrum in the future and see the importance of Scrum to provide better values and organization to the development team's teamwork while providing opportunities for improvements. As one development team member mentioned, the development team has three opportunities for improvement - Scrum Daily, Retrospective, and Slack channels; Scrum provides the agility while providing structures to improve while delivering values.

7.3 Lessons Learned

Lessons were learned during this implementation process as evidenced in the interview process. Moreover, I find myself many aspects I could have done better during this process. While this subsection primarily discusses the lessons learned as a team, I feel the importance of rotating the Scrum Master role in the software development team so that the whole software development team could improve and understand the context of our work while heightening the spirit of getting better together - in addition to applying our learning in a different context. Despite the aforementioned improvements, the evaluation reflected the lack of rigor in implementation or at least its consistency. For example, one of the development team members mentioned the problem with documentation as it is often outdated. Communication has improved but in need of the constant reminder to the team so that team channels are more often used than the direct messages. Design meetings are held and reported to be not sufficient - which might mean that there needs to be a discussion in the development team if such meetings are needed. Schedule, deadline, and clarity are often requested by development team members which reflects more of the issues of managing Scrum than Scrum itself. Some development team members have the opinion that PR should be commented a lot for documentation, while others feel the need to merge and get

the work moving on as demonstrated by the PR review practice over Google Hangouts meeting instead of spending a lot of time in asynchronous communication during the remote working. Planning has been always difficult which was further hindered by the team composition of full-time and part-time members. While review, retro, and planning were done in a single day to include all team members in the process, it was particularly challenging for everyone because of the long duration. While review and retrospectives are done now the previous day with planning on the next day, having structures or templates to reflect the clear beginning and ending for the planning was suggested by some team members - which would be cognitively less burdensome to everyone in the team. To improve the practice, some development team members still feel the importance of some small workshops about tools. While some development team members prefer having daily like daily, the others prefer not having it at all reflecting the challenges of working with people. Although the potentially shippable idea of the artifact was discussed in the introduction phase, the adoption of Demo in the practice was not done until later as noticed by some development team members. Some development team members suggested solving multiple problem areas and the other developers suggested introducing small changes one at a time with fewer commitments to start with. One thing the team was very clear from the beginning is to try something new and see if it works or not and then decide whether to keep those practices or not. All development team members experienced stability at work while having some free time devoted to learning also. Some of the factors were outside of the control - like development members unavailability, the disproportionate number between frontend and backend tasks, changing priorities over the different projects in the middle of the sprint. Despite those challenges, the overall process was quite nimble as the development team started having short and regular two weeks Sprint. Some team members mentioned the sprint backlog items being carried through multiple sprints - mostly because some of the tasks from the story were not finished and discussion about completing the whole story was carried out in the retrospective as well. While backend and frontend prefer planning separately, having frequent design meetings is recommended in the evaluation phase as it provides all the members i.e. backend developers, frontend developers, and the designer in the same place. One frontend developer described the improved code quality in general, there was an issue about frontend testing not having enough testing as in the backend code which is understandable given that there are often many frontend tasks in comparison to the backend.

In conclusion; it is vital to understand the three pillars of empiricism, adhere to the Scrum values and agile principles, and most importantly be motivated and self-organized to sharpen skills and ability to handle tools while focusing on creating values for the customers. Despite the challenges in the adoption process, the scrum framework provides agility to work in the changing environment with the goal of continuous improvements and delivery of the highest values product.

7.4 Evaluation Summary

To conclude the evaluation of the artifacts implemented, the survey result will be utilized in this section which validates the general feelings in the development team explored during the meeting rather than implying the effectiveness in positivist lenses. As emphasized by Berczuk of the significance of working as a collocated team before going distributed (Berczuk, 2007), the software development team was already collocated while one of the team members always working remotely from the very beginning of the team formation. As a result of that experience, the team handled the situation of having to work remotely pretty well, and the company guidelines for working remotely during the COVID-19 were also aligned with the distributed scrum recommendation in the software development team. As mentioned in the interview regarding the commitment for Scrum continuation, figure 10 also confirms the developers' sentiments regarding the importance of having scrum events in the Sprint except for one developer who likes to work remotely mostly communicating in Slack rating only one for scrum daily.

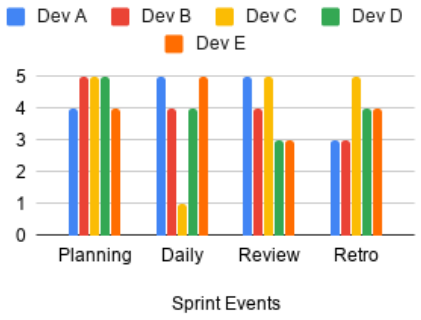


Figure 10: General Rating

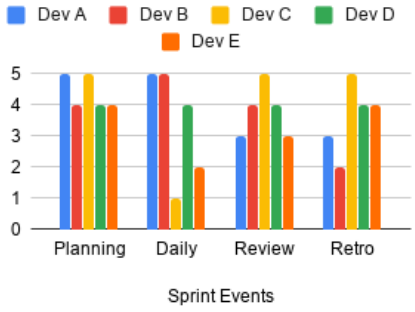


Figure 11: Problem Specific Rating

While developers who preferred working remotely or had to work remotely were against having daily, the one working from the office found having scrum dailies each working day very important both as a general feeling towards scrum daily as in figure 10 and its importance in solving the existing problems in the development team as in figure 11. In general, most developers preferred the single board for the development team regardless of the project development team is working and the skills team acquired during this thesis process on utilizing the Jira features in streamlining the process was noticed by the team. None of the developers mentioned the DoD which is mostly implemented in the continuously evolving CI/CD pipelines although there is already the need to discuss and streamline the team's DoD to have the product in a releasable state all of the time. The following figure 12 reflects the impact of the scrum adoption process in three major problem areas from which problems were selected during the current state analysis phase before primarily focusing on Process and Tools. The data might not be in itself significant, but it reflects the rationale behind choosing to streamline the Scrum process i.e. utilizing the opportunities Scrum framework offers to improve the development practice in addition to verifying

the root cause analysis in the current state analysis phase.

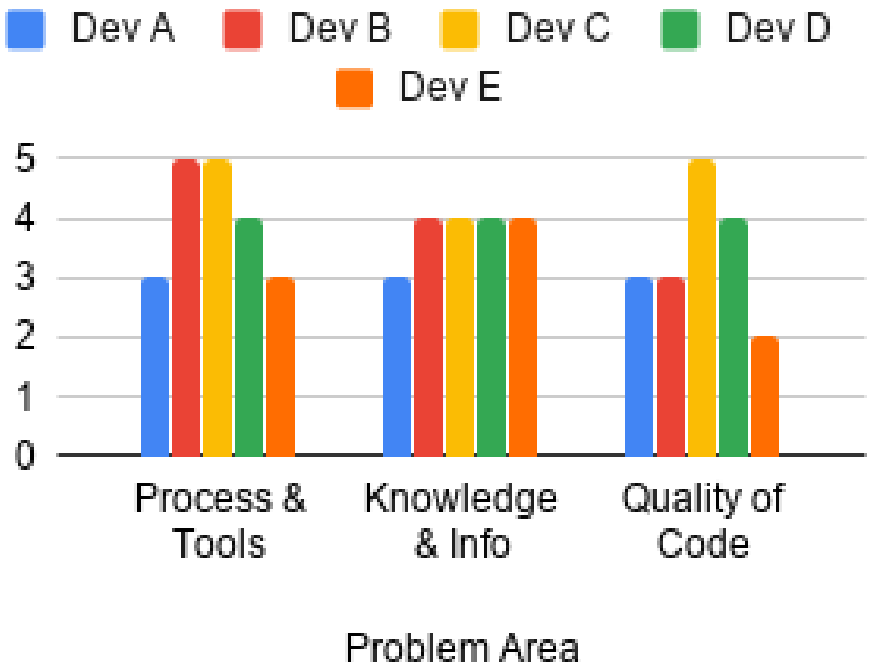


Figure 12: Developer Sentiment About Problem Areas.

While the focus of the thesis was the development team from the beginning and the sprint demo was also conducted within the development team in presence of the Product Owner, the culture of the demo is slowly permeating to the case company. While the whole organization was having the demo, the CEO inquired:

"When is the next demo?"

Despite the challenges and the need for continuous learning during this process, the overall evaluation of the process is positive as it shows the possibility of the process and the ability of the development team to deliver customer values.

8 Discussion

In the DSRM process model, the research process and the result is communicated once the artifact is evaluated. Despite this whole thesis paper reflecting the communication aspects of the process, this particular Section provides the core outcomes of the thesis process. In the following discussion, it presents the answers to the posed research questions and its significance both in the case company and the wider research community.

8.1 Research questions and the answers

When the software development team grew in the size and many different tools were introduced in the development workflow in the case company, the changed context initiated the main question of how to improve or solve problems within the software development team in the case company. Consequently, the research problem was formulated to solve **How to improve the software development practices in the case company?** as the main research question while performing a DSRM approach to study the problems and implement the solutions to improve the software development practices. The following research questions were posed to solve this major research problem which is explained further with the answers discovered during the process.

RQ1 **(What are the current problems in the software development team in the case company?)** focused on finding problems in the software development team in the case company. While the implementation was always based on the collective decision of the development team, the evaluation and problem discussion was conducted using a semi-structured interview with five members of the development team as a tool for exploratory research. The answer reveals five major problem areas 'Quality of Code', 'Knowledge and Information', 'Configuration and Environment', 'Process and Tools', and 'Visibility' while the root cause of these problems being 'Process and Visibility'. The concrete problems are discussed in more details in Section 5. In summary, the finding reveals the lack of the software development team working as a self-organizing team despite having motivated individuals hence not knowing what is happening inside the team. The Survey B result reflects the lack of coherence of knowledge within the development team - out of 9 survey questions, all 5 respondents agreed only on one question as shown in C. In conclusion, the root cause of the problems is a lack of process which supports visibility of the work within the development team hence creating problems in five key areas. The current state analysis phase exposes the lack of what every empirical process control is rooted i.e. transparency, inspection, and adaptation (Ripley and Miller, 2020) - both validating the root cause analysis and selection of the solution approach in this thesis project.

RQ2 **(What practices are employed by the software development team in the case company?)** focused on finding existing practices to acquire a better understanding of the context where the artifacts of the DSRM process is to be demonstrated. The answer reveals some solid engineering practices in the development team like standard application deployment process (Rossberg, 2019) as discussed

in Section 2 - most of them are adopted in the development team but not utilized or not all relevant members of the development team is aware in the worst scenario. Multiple boards in Jira for development work, git feature branch workflow, CI/CD implementation with different tools in the pipeline, etc. were already in practice. Scrum was already used within the software development team but missing most of the elements of the scrum. Scrum dailies were not happening daily, sprint length was not regular, and the scrum used was inconsistent (Ripley and Miller, 2020) as noted by the development team also - mostly because of the team members' schedule. The current sprint was concluded and a new sprint was started in a short amount of time without celebrating all scrum events.

Answering RQ3 **(What are the changes introduced in the software development team?)** introduced the changes initiated as artifacts in the software development team in the case company. At the beginning of the implementation, Scrum basics were presented to the team while providing plans on how to implement the process from now onwards. The implementation is discussed in great detail in Section 6 and Subsection 6.3 provides the concise list of implemented concrete practices during the thesis process. All Development team members interviewed mentioned the single board for the whole development team as the most significant changes as it also initiated other impacts which resulted in overall better knowledge in Jira. For example, the single configuration/settings were established in Jira for all projects the development team was working which simplified the general workflow. The role of Scrum Master and the Product Owner was assigned to different Scrum Team members (Ripley and Miller, 2020) and agreed to utilize rotating scrum master roles from within development team members in the future. Regarding the practices in distributed settings, the development team heavily utilized software tools (Berczuk, 2007), emphasized synchronous communication over the internet, prioritized physical meetings at times when possible (Rayhan and Haque, 2008), and focused on the highest priorities tasks to deliver the highest values regardless of the work settings.

In conclusion, the Scrum team was flexible and open to trying how new things work, decide if it works or not, and then drop or continue based on the results despite the warning of resistance in scrum adoption when the focus is on building projects with motivated individuals adopting solid engineering practices (Rigby et al., 2016). While different kinds of changes were introduced at different times - most of them not even having a significant presence in the development team, the framework for enabling development team members to continuously improve was established during the thesis process.

RQ4 **(How the introduced changes affect the existing problems?)** was presented to evaluate the effect of the changes thus introduced. While the previous research questions pose the rationale and context, this research question evaluates the implementation introduced as a part of the answer to RQ3. The positive changes due to the introduction of the improved scrum process are generally experienced in the software development team as seen in the result of the interview and the survey. Most of the development team members experienced that the Scrum process provided a better structure to the work (Sánchez-Gordón et al., 2016) while providing opportunities for improvements. Development team members experienced more time

for learning and sharing knowledge with others although it was frequently mentioned that there is a need to constantly remind people to discuss in the team channels. As expected in the discussion in the workshop during the current state analysis phase, the impact of improving process was observed in the other problem areas too as discussed in Section 7. In summary, there were perceived improvements in different problem areas with the possibility of further improvements also. Most importantly, the adoption process has established a framework that supports agility and improvement while initiating the culture which is seeking improvements constantly. The sprint events also observed improved discussion within the software development team as the result of dividing the task of product owner and scrum master from a single person to two individuals (Ripley and Miller, 2020).

8.2 Limitations of the research

There are limitations to this research also - despite having improved perception in the development team, it does not map the effect one to one with any of the implemented practices. Limitations are also prevalent due to the nature of interviews and myself being part of the development team from the very beginning - which might distort the perceptions of the people. Moreover, being a single case study research, it is hard to say that it could be generalized to the extent. Implementing the full-scale Scrum costs a lot of time - particularly for the small team which is often working in multiple projects, it is hard to generalize the implementation particularly when the larger context of implementation varies in each case. Regardless of the change in context and the subtle granularity in implementation, the result supports the benefits of Scrum provided in the literature. On the other hand, the context and number of developers interviewed, and the nature of the qualitative interview does not limit the perceived nature of the improvements in the case company. Describing the case company and development company in more detail could have probably supported the generalizability of the research.

While the literature review was not done extensively probing the relatedness of the context in which other researches were conducted, this paper does not exactly add value based on the literature although it provided an idea about different practices which were tried in this process even though most of them were discarded as unsuitable. Moreover, the literature review is scattered to cover the wide range of topics as the situation during the thesis process changed due to COVID-19 which enforced the development team to work in distributed settings. One of the limitations of the research is the selection of interviewees which often represents their interests and gets influenced by different aspects of the larger context in which the development team is operating. As the software development team comprised seven developers who could have been interviewed, five of them were interviewed in each stage. The interview result was matched with the result from the survey where possible as a means for data triangulation. One major limitation of this research is focusing absolutely within the development team instead of the whole organization - even though the organization culture greatly shapes the people working in the organization. One of the major problems discussed during the current state analysis was employees outside of the

development team creating tasks in the development team board and not having proper reporting tools for the bugs - which reflects the need for having organization level discussion within the case company.

In addition to these limitations, the evaluation part reflects competencies in my expertise regarding the implementation and overall research orientation - hence affecting the research negatively. Moreover, being part of the development team while working for this thesis might have some effects because of my relationship with my colleague - despite me always preferring to accept what one truly perceives than what makes me feel happy because ultimately the goal of this thesis is to help the case company by helping ourselves with the improvements in process and tools in addition to our skills and expertise. To tackle these shortcomings, additional attention is paid in selecting interviewees and the number of interviewees as well. Moreover, the data from the retrospective session is also included which presents the results of the discussion taking place with other team members equivalent to the focus group discussion.

In conclusion, there are limitations to the research - lack of strong literature review and my own experience with DSRM being the two major ones. Moreover, being myself as a development team member affects the research because it also affects the people with whom interaction takes place within the team regardless of being critically self-reflective to our interaction with other team members. Despite having relevance and significance within the case company, this research lacks the rigor which is demanded in the scientific community as stated by Dresch, Lacerda, and Antunes from the conception to the communication stage of the research (Dresch et al., 2015). In addition to utilizing the flexibility the case company offers to the software development team to work independently, the relevance of this endeavor could have been increased by including the whole organization in some stage during this process.

9 Conclusion

The purpose of this thesis was to improve the software development practices in the case company Vediafi Oy which involves current state analysis to figure out the problems, a demonstration in context by fully adopting the Scrum with the final part of the project involving the evaluation of the process. Scrum is challenging to master - as software development practices are undertaken in changing context. While this paper involves an exploratory approach to reveal the problems and to evaluate the implementation, the DSRM approach was utilized in the process of the evolution of the currently adopted practices.

While the improvement in practices is a constant search in the development team in the case company as demonstrated by the use of improved tools and automation over the time, most of the problems explored in the current state analysis phase manifested as the result of the adoption of new tools often unfamiliar to most of the members in the development team. When the external context is constantly changing and there are multiple projects with changing priorities and requirements, there is always the need for improvement and agility. While the development team in the case company started the practice in agile approaches, the introduction of the Scrum with its missing elements in the practice provided the framework for improvement - improvement in the developer productivity and software quality (Sadun, 2010). As mentioned by one of the development team members, there are now three opportunities which offer a chance for discussion on improvement:

> "... talking about problems and improvements, we have three places i.e. dailies, retro and maybe the Slack public channels. I feel like dailies can be used to draw attention to serious problems, retro for improving general long term practices, and Slack public channels for practical and small problems." (Participant D)

This project demonstrated solutions to some of the existing problems while exposing the committed effort and demonstrating the artifacts in context i.e. applying the Scrum in practices. Most of the negative issues were related to the planning of the Scrum events itself as a result of the limited effort from Scrum Master (Ozierańska et al., 2016) due to other responsibilities while other aspects of the Scrum and the impact of the adoption process were taken positively in the software development team. Most importantly, this project reveals the ability of the Scrum process to provide an opportunity for the development team to improve the engineering practices while utilizing skills and making assured to actively engage in the holistic development of the practices in a participatory fashion despite the need for tailoring Scrum adoption process.

References

AgileAlliance (n.d.). Twelve principles behind the agile manifesto.

Amar, H., Rafi-ul Shan, P. M., and Adegbile, A. (2019). Towards a 5c theory of communication for scrum-based distributed projects. In *BAM2019 Conference proceedings*. British Academy of Management.

Berczuk, S. (2007). Back to basics: The role of agile principles in success with an distributed scrum team. In *Agile 2007 (AGILE 2007)*, pages 382–388. IEEE.

Bhatia, A., Cheng, J., Salek, S., Chokshi, V., and Jetter, A. (2017). Improving the effectiveness of fuzzy front end management: Expanding stage-gate methodologies through agile. In *2017 Portland International Conference on Management of Engineering and Technology (PICMET)*, pages 1–8. IEEE.

Būcena, I. (2017). Establish devops maturity level. [Retrieved January 13, 2020].

Cho, J. (2008). Issues and challenges of agile software development with scrum. *Issues in Information Systems*, 9(2):188–195.

Chow, T. and Cao, D.-B. (2008). A survey study of critical success factors in agile software projects. *Journal of systems and software*, 81(6):961–971.

Clarke, V. and Braun, V. (2013). Teaching thematic analysis: Overcoming challenges and developing strategies for effective learning. *The psychologist*, 26(2).

Dalton, J. (2019a). Incremental development. In *Great Big Agile*, pages 181–182. Springer.

Dalton, J. (2019b). Sprint planning. In *Great Big Agile*, pages 241–243. Springer.

Dave, W. (2016). Updates to the scrum guide: The 5 scrum values take center stage.

Davis, C. (2019). *Cloud Native Patterns: Designing change-tolerant software*. Manning Publications.

Dresch, A., Lacerda, D. P., and Antunes, J. A. V. (2015). Design science research. In *Design science research*, pages 67–102. Springer.

Drummond, B. S. and JF, J. F. (2008). Yahoo! distributed agile: Notes from the world over. In *Agile 2008 Conference*, pages 315–321. IEEE.

Eloranta, V.-P., Koskimies, K., Mikkonen, T., and Vuorinen, J. (2013). Scrum anti-patterns–an empirical study. In *2013 20th Asia-Pacific Software Engineering Conference (APSEC)*, volume 1, pages 503–510. IEEE.

EuropeanScrum (n.d.). Scrum trainings.

Flora, H. K. and Chande, S. V. (2014). A systematic study on agile software development methodologies and practices. *International Journal of Computer Science and Information Technologies*, 5(3):3626–3637.

Fowler, F. M. (2019). The product backlog. In *Navigating Hybrid Scrum Environments*, pages 59–66. Springer.

Fowler, M. (2006). Using an agile software process with offshore development. [Retrieved July 23, 2020].

Fowler, M., Highsmith, J., et al. (2001). The agile manifesto. *Software Development*, 9(8):28–35.

Google (n.d.). Devops culture: Westrum organizational culture.

Hanslo, R. and Mnkandla, E. (2018). Scrum adoption challenges detection model: Sacdm. In *2018 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 949–957. IEEE.

Holmström, J., Ketokivi, M., and Hameri, A.-P. (2009). Bridging practice and theory: A design science approach. *Decision Sciences*, 40(1):65–87.

Humble, J. (2018). Continuous delivery sounds great, but will it work here? *Communications of the ACM*, 61(4):34–39.

Iivari, J. and Venable, J. (2009). Action research and design science research - seemingly similar but decisively dissimilar. *17th European Conference on Information Systems, ECIS 2009*, pages 1642–1653.

Kononenko, O., Baysal, O., Guerrouj, L., Cao, Y., and Godfrey, M. W. (2015). Investigating code review quality: Do people and participation matter? In *2015 IEEE international conference on software maintenance and evolution (ICSME)*, pages 111–120. IEEE.

Longhurst, R. (2003). Semi-structured interviews and focus groups. *Key methods in geography*, 3(2):143–156.

Lous, P., Tell, P., Michelsen, C. B., Dittrich, Y., and Ebdrup, A. (2018). From scrum to agile: a journey to tackle the challenges of distributed development in an agile team. In *Proceedings of the 2018 International Conference on Software and System Process*, pages 11–20.

Luz, M., Gazineu, D., and Teófilo, M. (2009). Challenges on adopting scrum for distributed teams in home office environments. *World Academy of Science, Engineering and Technology*, 59:308–311.

Mahmood, W., Usmani, N., Farooqui, S., and Ali, M. (2017). Benefits to organizations after migrating to scrum. In *29th International Business Information Management Association Conference*.

Marshburn, D. (2018). Scrum retrospectives: Measuring and improving effectiveness. In *Proceedings of the Southern Association for Information Systems Conference*.

Matthies, C., Dobrigkeit, F., and Ernst, A. (2019). Counteracting agile retrospective problems with retrospective activities. In *European Conference on Software Process Improvement*, pages 532–545. Springer.

McKenna, D. (2016). The scrum framework. In *The Art of Scrum*, pages 27–34. Springer.

Merkow, M. (2019). Secure, resilient, and agile software development.

Mohamed, M., Stankosky, M., and Murray, A. (2004). Applying knowledge management principles to enhance cross-functional team performance. *Journal of knowledge management*.

Moreira, M. E. (2010). *Adapting Configuration Management for Agile Teams: Balancing Sustainability and Speed*. John Wiley & Sons.

Ostrowski, L. and Helfert, M. (2012). Design science evaluation–example of experimental design. *Journal of Emerging Trends in Computing and Information Sciences*, 3(9):253–262.

Ozierańska, A., Skomra, A., Kuchta, D., and Rola, P. (2016). The critical factors of scrum implementation in it project–the case study. *Journal of Economics & Management*, 25:79–96.

Paasivaara, M., Durasiewicz, S., and Lassenius, C. (2009). Using scrum in distributed agile development: A multiple case study. In *2009 Fourth IEEE International Conference on Global Software Engineering*, pages 195–204. IEEE.

Pauly, D., Michalik, B., and Basten, D. (2015). Do daily scrums have to take place each day? a case study of customized scrum principles at an e-commerce company. In *2015 48th Hawaii International Conference on System Sciences*, pages 5074–5083. IEEE.

Peffers, K., Tuunanen, T., Rothenberger, M. A., and Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of management information systems*, 24(3):45–77.

Permana, P. A. G. (2015). Scrum method implementation in a software development project management. *International Journal of Advanced Computer Science and Applications*, 6(9):198–204.

Prat, N., Comyn-Wattiau, I., and Akoka, J. (2014). Artifact evaluation in information systems design-science research-a holistic view. In *PACIS*, page 23.

Qureshi, R., Basheri, M., and Alzahrani, A. A. (2018). Novel framework to improve communication and coordination among distributed agile teams. *International Journal of Information Engineering & Electronic Business*, 10(4).

Ramesh, B., Cao, L., Mohan, K., and Xu, P. (2006). Can distributed software development be agile? *Communications of the ACM*, 49(10):41–46.

Ramin, F., Matthies, C., and Teusner, R. (2020). More than code: Contributions in scrum software engineering teams. *arXiv preprint arXiv:2007.08237*.

Rayhan, S. H. and Haque, N. (2008). Incremental adoption of scrum for successful delivery of an it project in a remote setup. In *Agile 2008 Conference*, pages 351–355. IEEE.

Rigby, D. K., Sutherland, J., and Takeuchi, H. (2016). Embracing agile. *Harvard Business Review*, 94(5):40–50.

Ripley, R. and Miller, T. (2020). *Fixing Your Scrum: Practical Solutions to Common Scrum Problems*. Pragmatic Bookshelf.

Rossberg, J. (2014). *Beginning application lifecycle management*. Apress.

Rossberg, J. (2019). *Agile Project Management with Azure DevOps: Concepts, Templates, and Metrics*. Apress.

Sadun, C. (2010). Scrum and global delivery: pitfalls and lessons learned. In *Agility Across Time and Space*, pages 71–89. Springer.

Sánchez-Gordón, M.-L., Colomo-Palacios, R., de Amescua Seco, A., and O'Connor, R. V. (2016). The route to software process improvement in small-and medium-sized enterprises. In *Managing software process evolution*, pages 109–136. Springer.

Schwaber, K. (1997). Scrum development process. In *Business object design and implementation*, pages 117–134. Springer.

Schwaber, K. (2004). *Agile project management with Scrum*. Microsoft press.

Schwaber, K. and Sutherland, J. (2017). The scrum guide-the definitive guide to scrum: The rules of the game. [Retrieved April 11, 2020].

ScrumAlliance (n.d.). Benefits of using scrum. [Retrieved August 10, 2020].

Solcum, N. (2003). Participatory methods toolkit: A practitioner's manual. [Retrieved August, 13, 2020].

Stray, V. G., Lindsjørn, Y., and Sjøberg, D. I. (2013). Obstacles to efficient daily meetings in agile development projects: A case study. In *2013 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 95–102. IEEE.

Sverrisdottir, H. S., Ingason, H. T., and Jonasson, H. I. (2014). The role of the product owner in scrum-comparison between theory and practices. *Procedia-Social and Behavioral Sciences*, 119:257–267.

Wang, X., Maurer, F., Morgan, R., and Oliveira, J. (2010). Tools for supporting distributed agile project planning. In *Agility Across Time and Space*, pages 183–199. Springer.

A Current State Analysis Questions

1. How do you support the continuous delivery of the software? What are your responsibilities?
2. How do you use git workflow? What are the most common challenges you face while collaborating with other developers?
3. What are the challenging aspects of CI/CD? How do you feel the team/organization/yourself is reacting or acting to the change?
4. How has the CI/CD process evolved over time? What is your role in this development? How have you influenced or been influenced by these changes?
5. Do you see any problems with the current implementation? Any changes to git workflow? Software quality issues? Build and integration process? Do you have any suggestions?
6. Any other comments regarding the company, culture, team, CI/CD pipeline?

B Current State Analysis Survey - DevOps maturity level in technology area

Please, provide your feedback or comments if you have any regarding the particular questions (adopted from (Būcena, 2017)) followed by the feedback question. The question is used to assess the current level in technological area of the DevOps whereas the follow-up questions are used to collect your valuable opinion regarding your choice.

1. **How do you assess the level of deployment automation in current Vedia projects?**

   ❍Manual deployment

   ❍Build automation

   ❍Non-production deployment automation

   ❍Production deployment automation

   ❍Operational and Development teams are regularly collaborating to manage risks and reduce cycle time

   Could you explain your choice? .....

2. **How are issues and bugs tracked in current Vedia projects?**

   ❍No tools or minimal tool usage for issue tracking

   ❍All issues and bugs are tracked

   ❍Issue reporting automation and monitoring

   ❍Activities based on received feedback and data

   Could you explain your choice? .....

3. **How do you assess the level of information and knowledge flow within Dev Team?**

   ❍No collaboration tools

   ❍Project planning tool

   ❍Team/toolset integration

   ❍Knowledge management tool

   Could you explain your choice? .....

4. **How are the required environments provisioned in current projects?**

   ❍Environments are provisioned manually

   ❍All environment configurations are externalized and versioned

   ❍Virtualization used if applicable

   ❍All environments are managed effectively

❍Provisioning is fully automated

Could you explain your choice? .....

5. **How do you assess the process and data monitoring in current projects?**

❍No or minimal monitoring

❍Basic monitoring

❍Integrated monitoring

❍Analytics/Intelligence

Could you explain your choice? .....

6. **How do you assess the validation of the artifacts/deliveries developed?**

❍Manual tests or minimal automation

❍Functional test automation

❍Triggered automated tests

❍Smoked tests and dashboard shared with Operational team

❍Chaos Monkey

Could you explain your choice? .....

7. **Does your organization use Software configuration management (SCM)?**

❍No SCM

❍Standardized SCM

❍Configuration delivered with the code

❍Self-healing tools

Could you explain your choice? .....

8. **How do you perform the build management in current projects?**

❍Manual process for building software/ No artifact versioning

❍Regular automated build and testing; any builds can be recreated from source

❍An automated build and test cycle every time a change is committed

❍Build metrics gathered, made visible and taken into account

❍Continuous work on improvement, better visibility, faster feedback

Could you explain your choice? .....

9. **How is the data management process organized?**

❍Data migration un-versioned and performed manually

❍Changes to DB done with automated scripts versioned with application

❍DB changes performed automatically as part of deployment process

❍DB upgrades and rollbacks are tested with every deployment

❍Feedback from DB performance after each release

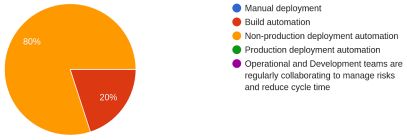
Could you explain your choice? .....

Would you like to provide any open feedback or comments?

C Current State Analysis Survey Answer

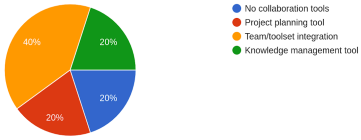
Only the free text answer to the survey questions is utilized along with the interview for thematic content analysis.

1. How do you assess the level of deployment automation in current Vedia projects?
5 responses

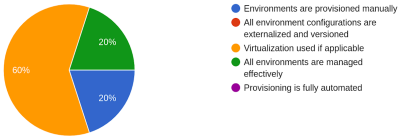


2. How are issues and bugs tracked in current Vedia projects?
5 responses

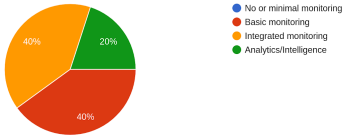
3. How do you assess the level of information and knowledge flow within Dev Team?
5 responses



4. How are the required environments provisioned in current projects?
5 responses

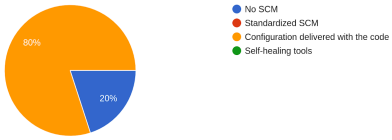


5. How do you assess the process and data monitoring in current projects?
5 responses



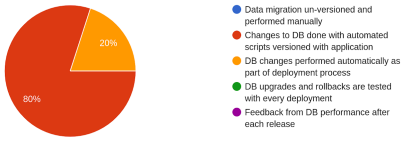
6. How do you assess the validation of the artifacts/deliveries developed?
5 responses

7. Does your organization use Software configuration management (SCM)?
5 responses



8. How do you perform the build management in current projects?
5 responses

9. How is the data management process organized?
5 responses



D Evaluation Questions

Problem Discussion

- What were the problems discussed during the current state analysis phases?
- Do you feel like some of those problems are solved after adopting this process?
- What could have been done better to solve those problems?

Application of Scrum

- How do you feel about your knowledge of Scrum?
- Could you explain the importance/rationale for introducing Scrum practice in our process?
- How do you find the improvement of engineering practice in the Development team over the time?
- What do you remember the most about this Scrum adoption process?

Future Improvement

- Would you like to continue the Scrum process? What could we do better in either case?
- What are the areas we could have improved during this adoption process?

E Evaluation Survey

The aim of the survey is to measure the developers' satisfaction regarding the sprint events and their effectiveness in solving the existing problems. The survey uses Likert scale in which 1 means (contributed to nothing/ do not agree at all) and 5 means (contributed highly/agree very much). Although the free text is optional, your feedback will be highly appreciated.

1. **How do you rate the different Scrum events?** What is your general overall feeling about them in our context? Please, suggest further area of improvement if you lie to suggest any.

   Sprint Planning

   ① ② ③ ④ ⑤

   Any idea for improvements? .....

   Daily Scrum

   ① ② ③ ④ ⑤

   Any idea for improvements? .....

   Sprint Review

   ① ② ③ ④ ⑤

   Any idea for improvements? .....

   Sprint Retrospective

   ① ② ③ ④ ⑤

   Any idea for improvements? .....

2. **How do you rate the different Scrum events in their effectiveness to solve existing problems?** As the reason for adopting Scrum was to solve the existing problems discussed throughout the current state analysis phase, how do different sprint events affect solving those problems?

   Sprint Planning

   ① ② ③ ④ ⑤

   Could you explain your choice? .....

   Daily Scrum

   ① ② ③ ④ ⑤

   Could you explain your choice? .....

   Sprint Review

   ① ② ③ ④ ⑤

   Could you explain your choice? .....

Sprint Retrospective

① ② ③ ④ ⑤

Could you explain your choice? .....

3. **How do you agree about following statements?** If you have comments like if Scrum adoption introduced other challenges, instead of solving the problems or if it solved some different problems in the given problem areas instead of solving the problems you thought there were in the beginning, please provide the comments.

   The process and utilization of existing tools is improved

   ① ② ③ ④ ⑤

   Could you elaborate your choice? .....

   The knowledge and information flows within the team as a single team instead of within individuals.

   ① ② ③ ④ ⑤

   Could you elaborate your choice? .....

   The quality of the code has improved.

   ① ② ③ ④ ⑤

   Could you elaborate your choice? .....