Mikko Kurkela

DevOps Capability Assessment in a Software Development Team

Teknologiaosaamisen johtaminen

YAMK

Kevät 2020

KAMK • University of Applied Sciences

Tiivistelmä

Tekijä(t): Kurkela Mikko

Työn nimi: DevOps-Kyvykkyyksien Mittaaminen Ohjelmistokehitys Tiimissä

Tutkintonimike: Insinööri (Ylempi AMK)

Asiasanat: DevOps, Agile, Ohjelmistokehitys, Lean, arviointi

DevOps kyvykkyyksien mittaaminen ohjelmistokehitys tiimissä on tärkeää, jotta tiimi ja sen johtajat ymmärtävät tarkasti tiimin vahvuudet ja heikkoudet uusien menetelmiä käyttöön ottaessa. Tämän opinnäytetyön perustana on State of DevOps -kyselytutkimuksen pohjalta identifioidut DevOps-kyvykkyydet sekä Scrum-viitekehyksen oppaat. Opinnäytetyö on rajattu tarkastelemaan vain yhtä ohjelmistokehitystiimiä osana isompaa organisaatiota.

Opinnäytetyö on luonteeltaan laadullinen tutkimus, joka toteutetiin pääosin strukturoituna kyselynä. Tiimin tämän hetkistä toimintaa arvioitiin tarkkailemalla tiimin toimintaa, analysoimalla tiimin tuottamaa dokumentaatiota, tutkimalla käytössä tiimin käytössä olevien työkalujen sisältöä ja haastattelemalla epäformaalisti tiimin jäseniä. Lisäksi tiimille toteutettiin räätälöity semi-strukturoitu kyselytutkimus, jolla kartoitettiin tiimin DevOps-kyvykkyyksiä ja ketterien menetelmien toteutumista.

Opinnäytetyön tuloksena saatiin kartoitettua laajasti tiimin työskentelytapoja, joita vertailtiin lähdemateriaalien tuloksiin ja suosituksiin. Tiimille annettiin suosituksia tärkeimmistä kehityskohteista, joilla DevOps-menetelmien käyttöönotosta saadaan helpompaa ja hallittavampaa.

KAMK • University of Applied Sciences

Abstract

Author(s): Kurkela Mikko

Title of the Publication: DevOps Capability Assessment in a Software Development Team

Degree Title: Master of Engineering

Keywords: DevOps, Agile, Software development, Lean, assessment

When adopting DevOps practices, it is important for team and its management, to understand the DevOps capabilities of the team. This thesis is based on DevOps capabilities identified from State of DevOps survey and Scrum framework guides. The scope of the thesis is one software development within a large organization.

The thesis uses qualitative methods and semi-structured survey. Team's current behaviors and practices were observed, documentations produced by the team were analyzed, team's development tools were studied and team members were interviewed. Also, a semi-structured survey to measure DevOps capabilities and agile software development implementation were conducted.

On this thesis, team's working practices were widely assessed. The practices were reflected to results of source researches and suggestions. Based on the assessment, team received recommendations, which helps in team's DevOps transformation and make it more manageable.

Table of contents

Glossary

API	Application Programmable Interface
ATTD	Acceptance Test Driven Development
CD	Continuous Development
CDE	Continuous Delivery
CFR	Change Failure Rate
CI	Continuous Integration
CVCS	Centralized Version Control System
DVCS	Distributed Version Control System
eNPS	employee Net Promoter Score
MTTR	Mean Time To Recover
NPS	Net Promoter Score
TDD	Test Driven Development
UAT	User Acceptance Test
VCS	Version Control System
WIP	Work in Progress

Table of figures

1 Introduction

Organizations and their software development teams adopt new software development methods and technologies to improve their software delivery performance. During last decade DevOps has become a widely used approach in software development. The rise of agile development brought focus to the end user. Short delivery cycles with incremental development process pursue development team to deliver faster and more efficiently. However, the operation team wants to keep system stable and keep changing at minimum. DevOps keeps the agile principles in the core, but also tries to break barriers between development and operations world. In the end, development and operations are both trying to satisfy the customer's need. Seamless collaboration is essential; development team and operations team need to be one, DevOps team.

During the last decade, hundreds of tools have emerged to solve DevOps challenges. Organization's DevOps transformation becomes easily too focused on tools, but DevOps is much more than a technical solution. Many different models have been created to describe DevOps, which all tries to grasp the diversity of the term. Also, the academic world has tried to define DevOps with various success. Measuring and assessing something that is difficult to describe is challenging, but luckily there are plenty of common ground within different explanations of DevOps. Also, some studies are made about measuring DevOps.

The aim of this thesis is to find out State of DevOps in certain software development team, which is part of a big organization. The thesis uses research based on annual State of DevOps survey as the base with other relevant studies and suggestions by DevOps practitioners. Team introduction, scope of the study, research questions and the research method are described in chapter 2.

In chapter 3 relevant background studies and used terms are explained. Chapter 4 describes current development practices of the team. Chapter 5 analyses existing metrics and describe State of DevOps survey tailored to the team. Chapter 6 discuss about the findings of the study and give recommendations to improve. Chapter 7 process limitations of the study and executes source analysis. Finally, on chapter 8 whole thesis is concluded.

2 Preface

This chapter introduces the team under research, the scope of the study, represents research questions and describe research methods. Team introduction is kept at a minimum to avoid identifying the company and the team.

2.1 Team Introduction

The team produces a software for industry. The team has 17 members. Approximately 2/3 of the team is located in Finland and 1/3 is located in India. The team is part of a bigger organization with multiple software and engineering teams. The main deliverable of the team is a software product, which is installed on the customer's premises. Accessibility to the production environment may be limited.

The team has been developing its product a few years with agile software development methods. In spring 2019 the team started to improve its development practices. The team has replaced its old build tool with Jenkins continuous integration (CI) and the team’s development process got revamped.

2.2 Research Question and the Scope of the Study

This thesis tries to answer the question: "What is the current state of the team in its DevOps transition?" The answer should cover different dimensions of DevOps. Additional research question "How the team members feel about current DevOps and Agile transition?" Third research question is: “What are the team’s next most important steps to focus on its DevOps journey?”

Even the term, DevOps, is understood to cover whole organization, the scope of the research is only the team. The organization is only studied from the team’s point of view or when the organization directly affects the team’s daily work, in the way that is relevant to the study. The study is tailored to this particular team, but it can be used as the basis for similar assessments for some other software development team.

2.3 Research Method

This study uses a semi-structured questionnaire as a qualitative research method. As the team does not have much collected data available and the population for the survey is small, no reliable assumptions based on qualitative data can be made. Therefore, qualitative method is the right choice for this research.

In qualitative research, the researcher tries to find answers to research questions by an iterative process to get a better understanding of the studied phenomenon. Qualitative research is in dialog between theory and observed reality. The aim of the qualitative research is to improve understanding by gathering and analyzing the data and reflecting it to known theories and generating new understanding and get researcher closer to the answers of the research questions. [1.]

In this study, the survey is used to gather data from team members. Additionally, different metrics from version control system (VCS), project management tool and existing statistics are studied. MS Forms was used as survey platform. Survey questions were mostly on a Likert scale to get quantitative data. Addition to Likert scale questions, open questions were used to examine broader ideas.

Additional to survey, current practices of the team are observed and analyzed. Information for current practices are gathered from documentation, software development tools and by observing the daily work of the team and informally interviewing team members.

3 Background Studies

This chapter introduces relevant background studies and concepts for this thesis. At the begin of the chapter agile software development and Scrum framework is introduced. Second part of the chapter introduces DevOps and different characteristics and technical practices commonly related to DevOps.

3.1 Agile Software Development

Traditional project management views development as a linear sequence of well-defined activities. However, software development process is full of changes and uncertainties, it did not fit into the traditional approach. Term "software engineering" was coined at a NATO conference in Garmisch-Partenkirchen in 1968. Software development was seen to be driven based on the principles and practices seen in engineering. Uncertainties related to software development were attempting to manage with engineering principles. [2.]

Software projects are usually complex and uncertain. Agile project management is trying to manage the complexity and uncertainty by shortening the time frame between planning and execution, recognizing that at the planning time, all the information is not available and empathizing creativity and learning. Moving from traditional project management to agile project management implies dealing with complexity and unpredictability by relying on people and their creativity rather than on processes, moving from command and control to shared decision-making and self-management. Agile Software development is based on principles found in the Agile Manifesto. [2.]

3.1.1 The Agile Manifesto

A group agile of software development enthusiasts, The Agile Alliance, met on February 2001 at the Snowbird ski-resort. The aim of the meeting was to discuss about software development, relax and enjoy good food. Despite the expectations, the group ended up agreeing on a Manifesto

for Agile Software Development. The manifesto contains core values, which all the representatives of different agile software development methodologies were able to agree. The manifesto [2] contains four statements

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

The statements reflect twelve principles behind the statements. Right-hand side of the statements is seen valuable, but the main focus should be on the left-hand side. [2.]

3.1.2 Scrum Framework

Scrum is a workflow framework for highly flexible and adaptive teams. The Scrum was co-created by Ken Schwaber and Jeff Sutherland. First time Scrum was publicly presented at the OOPSLA conference in 1995. Since then Scrum has gained popularity on different fields, especially in software development.[4.] GitLab's 2019 Developer report: DevSecOps ranked Scrum the most practiced development methodology [5]. 60% of the survey takers were GitLab users. Scrum guide is strict only about roles and ceremonies, but leaves flexibility to adjust the work process.

The Scrum guide defines Scrum as "A framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value." Scrum uses iterative and incremental methods to produce and deliver products and solutions with optimized predictability and controlled risks. Scrum has three pillars: transparency, inspection and adaptation. The essence of Scrum is highly flexible and adaptive self-organizing team. [4.]

The Scrum team consists of a Scrum Master, a Product Owner and the development team. The development team is self-organizing and cross-functional. The team develops products in small increments and the whole team is accountable for results. The team's implementation of the Scrum work process (Figure 1) is revised regularly and team should build fast feedback loops for

learning. The core values of the Scrum team are commitment, courage, focus, openness and respect. [4.]

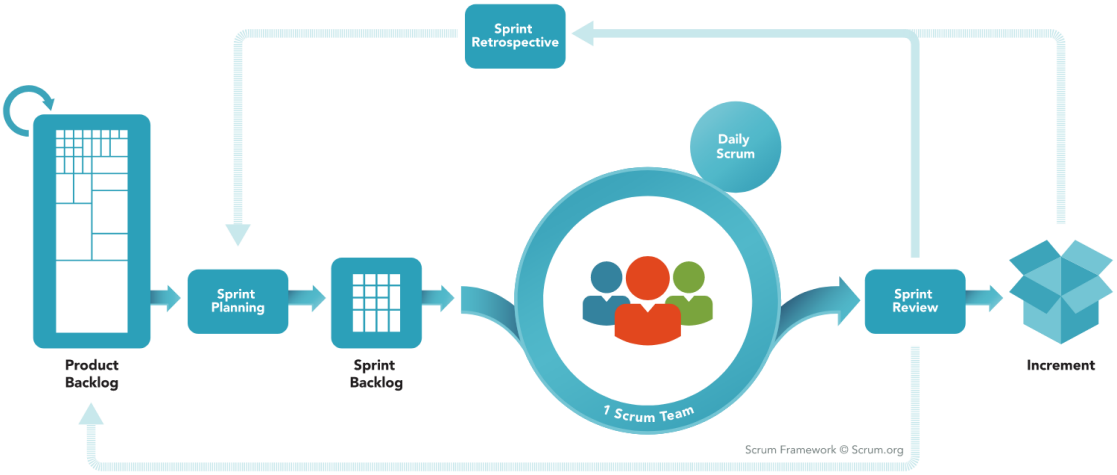

Figure 1 Scrum framework [4]

Product Owner manages work items and organizes them in a product backlog. Product Owner's responsibility is that the development team is working on most valuable items and ensures that the team understands the work items. Product Owner makes sure that the team has correct focus and understanding of the developed product. [4.]

The Scrum Master is a servant leader for the team. The Scrum Master helps to understand Scrum theory and facilitates Scrum events. The Scrum Master can help the Product Owner with the backlog management, but the Product Owner still has the ownership of the backlog. For the development team, Scrum Master acts as a coach and tries to remove obstacles. The Scrum Master also works at the organization level. The Scrum Master collaborates with other Scrum Masters, coaches the organization in Scrum adoption, helps employees and stakeholders to understand and apply Scrum and empirical product development. [4.]

The development team is cross-functional and self-organizing. There is no titles nor hierarchy in the team. The team can decide how it delivers the product increment. Team members may be specialized for certain tasks, but the whole team is accountable. Team size should be small enough that the team can work together efficiently, but large enough to accomplish meaning full product increment in one development sprint.

The Scrum framework defines time boxed events. The events enable transparency and inspection, which are critical for Scrum. "Sprint" is a container event for all other Scrum events. Sprint lasts

one month or less. During a sprint, the team develops an increment for the product. The sprint goal should not be changed during the sprint. If the goal becomes obsolete the sprint may be canceled and a new sprint started. [4.]

A sprint starts with a sprint planning event. In the sprint planning, the team defines its goal and plans how the work is executed. The whole team should participate the planning. The team follows the progress of the sprint on daily Scrum meetings. [4.]

Daily Scrum is time boxed 15-minute event. In the daily Scrum, the development team makes plans for next 24 hours. Daily Scrum meeting eliminates the need for other meetings. Daily meeting is only for high level discussion and detailed problem solving should happen after the meeting, if needed. [4.]

After the sprint is a sprint review meeting. In sprint review, team collaborates with stakeholders about the results of the sprint. The Product Owner explains what has been achieved and which goals has not been reached. The development team discusses about successful occurrences, challenges it ran into and how the issues were solved. Sprint review provides input for next sprint planning. Sprint retrospective is also held after a sprint. In retrospective, the team identifies major items that went well and potential improvements. In the retrospective, the team creates a plan for implementing improvements to the way of work. [4.]

3.1.3 Scrum Anti-patterns

Eloranta, Koskimies and Mikkonen [6] studied common deviations of recommended Scrum way of working. The study identified fourteen anti-patterns. The research also points out that big companies and companies, which has used Scrum longer time, has adopted more anti-patternial behaviors than small companies and Scrum beginners. The fourteen identified anti-patterns are:

1. Big requirements documentation
2. Customer Product Owner
3. Product Owner without authority
4. Long or non-existent feedback loops

5. Unordered Product Backlog
6. Work estimates given to teams
7. Hours in progress monitoring
8. Semi-functional teams
9. Customer caused disruptions
10. Business as usual
11. Invisible progress
12. Varying sprint length
13. Too long sprint
14. Testing in next sprint

The Scrum guide does not give direct answers how to overcome the challenges, but relies on principles and values, which team should follow when creating their own work process. As the Scrum guide says, “Scrum is easy to learn but difficult to master” [4].

3.1.4 Self-managing Teams

A software development team consists several people with multiple expertise. A self-managing team must be able to communicate and collaborate effective ways. In agile software development, self-managing teams are widely recommended. [2.] In the Scrum framework, teams should be self-managing, who invent their own way of work and is collectively responsible for the team outcomes [4]. Self-managing teams can have many benefits over traditionally managed teams. Self-managing teams can improve learning, effectiveness, innovativeness, job-satisfaction and employee turnover.[2.] Takeuchi and Nonaka claims that self-managing teams are essential part of innovative product development.[7.]

Takeuchi and Nonaka define self-managing team as a team, which has three conditions: autonomy, self-transcendence and cross-fertilization. Autonomy does not mean totally uncontrolled.

Management should have enough control to prevent chaos, but still leave space for creativity and spontaneity. [7.] Dybå et al [2] listed five conditions to support self-management:

- Clear, engaging direction
- An enabling performing unit structure
- A supportive organizational context
- Available, export coaching
- Adequate resources

An agile project manager must balance team level autonomy and individual level autonomy. Dybå et all [2] list things that project manage must ensure when successfully leading a self-managing team

- The team has the authority to define work strategies and processes, project goals, and resource allocation
- All team members jointly share decision authority (what group tasks to perform and how to carry them out)
- A team member must have some freedom in carrying out the assigned task

Team leader's tasks in self-managing teams include planning, blocking interruptions, defining the work process, ensuring resources, and set up the technical infrastructure [8]. Team members have high responsibility for their own work and performance. One who knows, should make the decisions. [2]

Iqbal, Omar and Yasin [9] studied factors that an agile team comprise. The study found that most of the factors positively correlate with team productivity. Team leader meetings, unit and regression testing were found to negatively correlate with team productivity, whereas team empowerment, inter-team coordination, requirements as user stories, requirements workshop involvement, clear and ready for development features, test cases and integration testing was found significantly correlating with the productivity of the team. [9.]

3.2 DevOps

At this section, term DevOps is discussed by introducing different models and characteristics based on views by well-known DevOps practitioners and DevOps researches. Two DevOps concepts, Three ways by Gene Kim and CAMS model by Damon Edwards and John Willis are introduced to give clarity to the definitions section. After definition section, different dimensions of DevOps and practices related the dimensions are described. Finally, assessing and measuring the DevOps is discussed.

3.2.1 The Definition of DevOps

Many DevOps practitioners and researcher have defined DevOps by practices that belongs to different categories. Forsgren, Humble and Kim found 24 capabilities related to DevOps under 5 categories (3.3.5) [10, p201-207]. Edward and Willis founded acronym CAMS. Where C stands for Culture, A for Automation, M for Measurement and S for Sharing [11]. Lwakatare, Kuva and Oivo [12] found five characteristic dimensions of DevOps:

1. Collaboration: rethinking and reorientation of roles and teams in development and operation activities
2. Automation: Infrastructure and deployment process automation
3. Culture: Empathy, support and good working environment between development and operations
4. Monitoring: Instrumenting application and aggregating monitored data into insights
5. Measurement: Useful metrics

Lwakatare, Kuvaja and Oivo [12] defined the DevOps as

"A mind-set substantiated by a set of practices to encourage cross-functional collaboration between teams, especially development and IT operations within a software development organization, in order to operate resilient systems and accelerate delivery of change"

The definition was based on online documents and blog posts by DevOps practitioners. However, DevOps practitioners did not completely agree with the definition. The researchers also state that practitioners vary their descriptions of DevOps [12].

In the article "Surprise! Broad Agreement on the Definition of DevOps" Eric Minick discusses about common nominators between DevOps definitions from different analysts, authors, industry and community leaders [13]. He presents a list

- DevOps exits to help the business to win
- The scope is broad, but centered on IT
- The foundations are found in Agile and Lean
- Culture is very important
- Feedback is fuel for innovation
- Automation helps

Minick's list, catches the tone of the term DevOps very accurately. Minick does not even try to give a strict definition to DevOps as there is not common agreement on the definition. However, everybody agrees with core elements and builds their definition around the elements. In this thesis, DevOps is considered as an umbrella term, which includes different technical approaches as well as organizational culture and development process. Next headings introduce two models "Three Ways" by Gene Kim and CAMS model by Edwards and Willis. These models shape the usage of term DevOps in this thesis.

The Three Ways

Gene Kim represented DevOps as three ways (Figure 2). The first way leans to system thinking and breaking organization silos of work. The second way concentrates on creating and amplifying feedback loops from customer to development. The third way is about a culture that support continuous experimentation and learning. [14.]

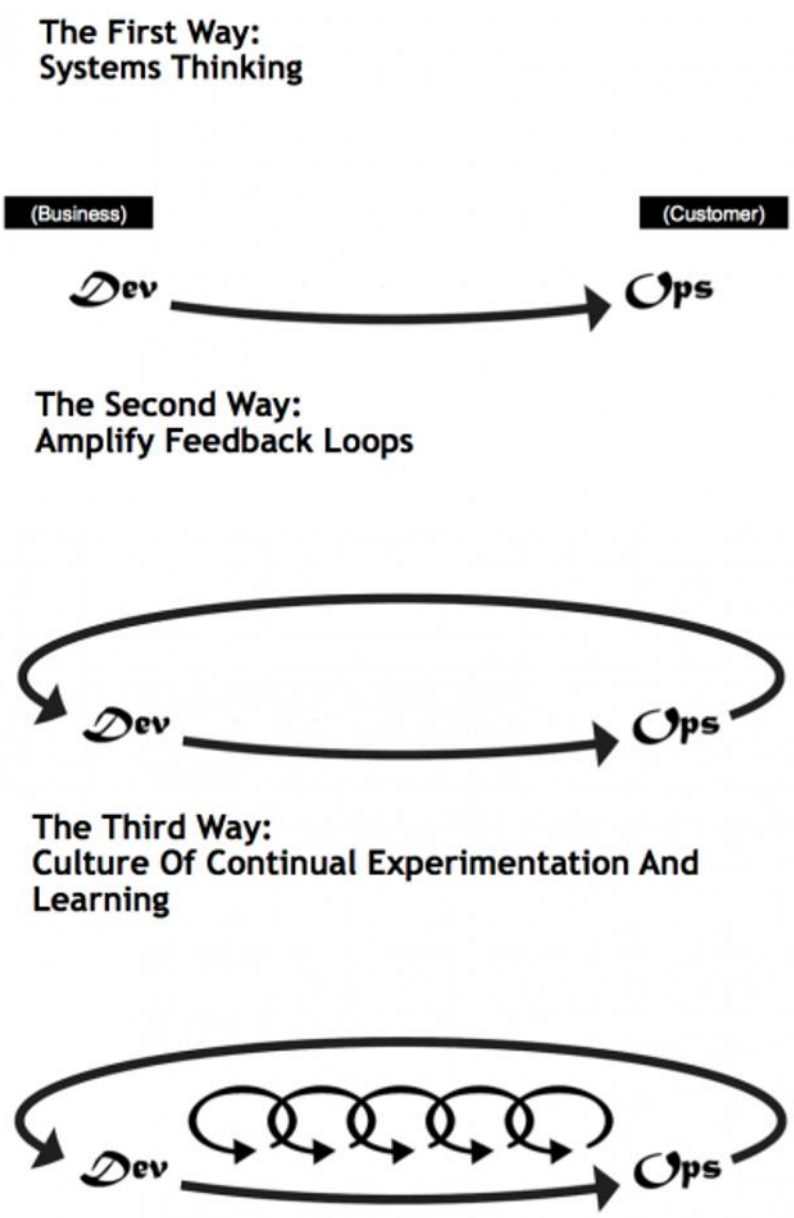

Figure 2 The Three ways [14]

First way includes practices like value stream mapping, making the work visible, reducing patch size and shifting left in the quality and optimizing for global goals. Tools for first way are work in progress limits, continuous integration and continuous deployment and on demand environments. The second way involves culture where all incidents are fixed and root cause is examined. The practices allow the organization to build ever-safer systems of work and detect errors before failure occurs. The second way enables continuous learning. The third way empathizes generative, high trust culture that supports experimentation and risk taking. A culture that supports

learning. The third way also includes scaling up the benefits of new finding to the whole organization. [15.]

CA(L)MS Model

Acronym CAMS was coined by Damon Edwards and John Willis after first US based Devopsdays in Mountain View 2010. C stands for Culture, A for Automation, M for Measurement and S for Sharing. Later Jez Humble added L for Lean. [11] Eveline Oerlich and Forrester Research Inc. added another S for sourcing [16]. However, CAMSS acronym hasn't become as popular as CAMS and CALMS. The model describes DevOps as a phenomenon, which contain all the elements of the acronym. (Figure 3).

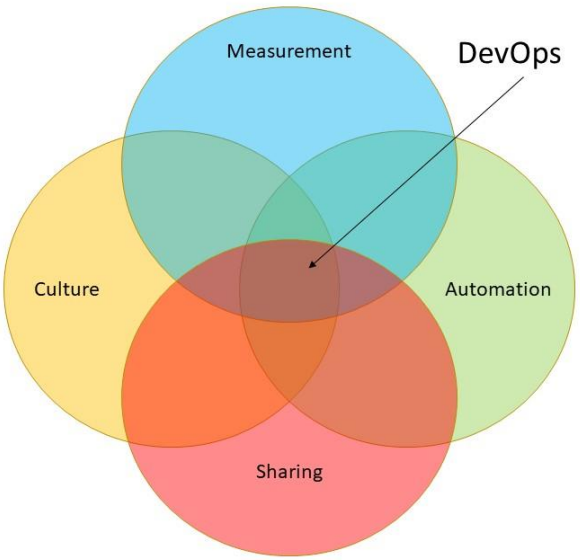

Figure 3 CALMS Model

CALMS acronym shows that DevOps is not just about tools and practices, but also about culture, automation, lean, measurement and sharing combined. John Willis talk on article "DevOps Culture" about organizational culture as an enabler of continuous improvement and lean thinking and the importance of leadership to buy-in the idea. [11.] 2018 State of DevOps survey by Puppet Lab [17] was aimed to find out if organizations who have taken account all four categories of CAMS have got further in their DevOps journey. The 2018 report proposes that CAMS model is useful for benchmarking organization's evolutionary progress. The survey shows that highly

evolved organizations have DevOps culture that spans across multiple departments, the organization has automated services for broad use, measurement is automated and the organization share patterns and best practices broadly across the organization. [17.]

3.2.2 Culture

Organizational culture is an abstraction, but social and organizational situations deriving from culture are powerful. If we do not understand the organizational culture forces, we become victim to them. Understanding the organizational culture helps us to understand the situation we face in our organizational life and it helps us to understand ourselves too. Schein defines culture as "*pattern of shared basic assumptions learned by a group as it solved its problems of external adaptation and internal integration, which has worked well enough to be considered to be valid and, therefore, to be taught to new members as the correct way to perceive, think, and feel in relation to those problems*." [18, p. 3-17]

Westrum's Typology of Organization Cultures

Ron Westrum defines organizational culture as the organization's pattern of response to the problems and opportunities it encounters (Figure 4). Employees creates the culture based on priorities that the leader sets. Westrum showed that organizational culture can predict safety performance in health care. Westrum divided organizations into three dominant types based on style of information flows in the organization. These types are pathological, bureaucratic, and generative. Westrum says that the types are shaped by the preoccupations of the unit's leader. [19.]

Pathological (Power oriented)	Bureaucratic (Rule oriented)	Generative (Performance oriented)
Low cooperation	Modest cooperation	High cooperation
Messengers shot	Messengers neglected	Messengers trained
Responsibilities shirked	Narrow responsibilities	Risks are shared
Bridging discouraged	Bridging tolerated	Bridging encouraged
Failure -> Scapregoating	Failure -> Justice	Failure -> Inquiry
Novelty crushed	Novelty leads to problems	Novelty implemented

Figure 4 Westrum's typology of organizational cultures [19]

In team research by Google [20], researchers found out that it is more important how well the team works together than who are in the team. In the Google's research five most important feature of effective team are described in Figure 5

Figure 5 Features of effective teams [20]

To enhance psychological safety team leaders can solicit input and opinions from the team and share information about persona and work style preferences, and encourage others to do same.

Dependability can be improved by clarifying roles and responsibilities and by developing concrete project plans to provide transparency into every individual work. Regular communication about team goals and ensuring team members understand the plan for achieving the goals and ensuring team meetings have a clear agenda and designated leader can improve structure and clarity. Public praise and giving gratitude when someone help out and positive feedback improves meaning. The impact can be improved by creating a clear vision that reinforces how team member's work contributes to the team's goals and organization's goals and by adopting user-centered evaluation method. [20.]

Variables, which correlated most with team effectiveness at Google are

- Colocation of teammates (sitting together in the same office)
- Consensus-driven decision making
- Extroversion of team members
- Individual performance of team members
- Workload size
- Seniority
- Team size
- Tenure

State of DevOps research identified that technical practices such as continuous delivery and lean development practices has measurable effects on organizational culture. Organizational culture increases organizational performance (Figure 6) [10].

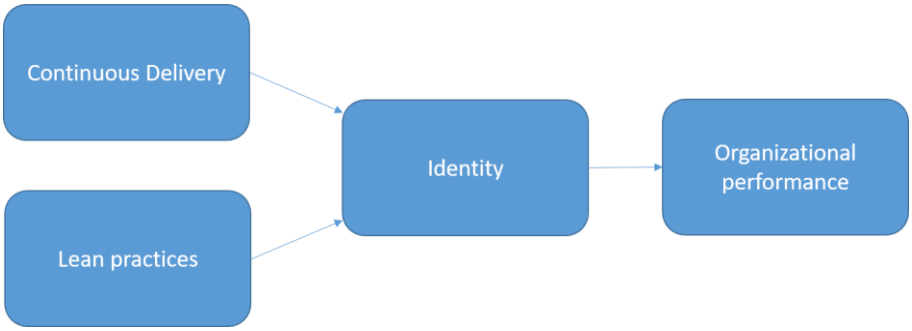

Figure 6 Impacts of technical and lean practices on identity [10]

Transformational Leadership and Servant Leadership

A transformational leader tries to facilitate wide-scale organizational change by appealing follower's values and sense of purpose. Transformational leaders work on getting followers to identify with the organization and work for organizational goals, whereas servant leadership focus on developing the follower's capabilities and performance. [10.]

Leaders has an important role in rolling out the DevOps transformation in organization [10, p. 121]. According to Rafferty and Griffin [21], five dimensions of transformational leadership are

1. Vision. An expression of an idealized picture of the future based around organizational values.

2. Inspirational communication. An expression of positive and encouraging messages about the organization, and statements that build motivation and confidence.

3. Supportive leadership. Expressing concern for followers and taking account of their individual needs.

4. Intellectual stimulation. Enhancing employees' interest in, and awareness of problems, and increasing their ability to think about problems in new ways.

5. Personal recognition. The provision of rewards such as praise and acknowledgement of effort for achievement of specified goals.

The dimensions of transformational leadership were found to have positive associations with affective and continuance commitment, role breadth self-efficacy, interpersonal behaviors, and intentions to turn over. The claims hold partially true on a study conducted by Rafferty and Griffin.

Vision was found to have unique negative association with continuance commitment and role breadth self-efficacy. The researchers debated, if speaking of vision without inspirational communication would cause the findings. [21.]

The research of Rafferty and Griffin also revealed, that inspirational communication was found to positively associate with role bread self-efficacy, affective commitment and interpersonal helping. The same research also shows that intellectual stimulation is positively associated with affective commitment and continuance commitment, which contrast to former research findings where intellectual stimulation were found to have a negative impact to employees. Personal recognition was negatively associated with continuance commitment and supportive leadership did not have significant unique relationship with studied outcomes. [21.]

The study suggests that leaders can have a powerful positive effect on employees by expressing positive and encouraging messages. When expressing vision, inspirational communication should be included or the message may have even negative outcome. Intellectual stimulation can be used to increase employee's affective attachment to an organization, while personal recognition may encourage employees to leave the organization. However, there are research, which suggest that strong continuance committed employees are less likely to make a positive contribution to a firm. [21.]

With the State of DevOps survey, participants who reported most of the five leadership characteristics were more likely from high performing team than those who reported lower level of the characteristics. The transformational leadership scores were found to correlate with the employee net promoter score (Figure 7). [10, p. 120.]

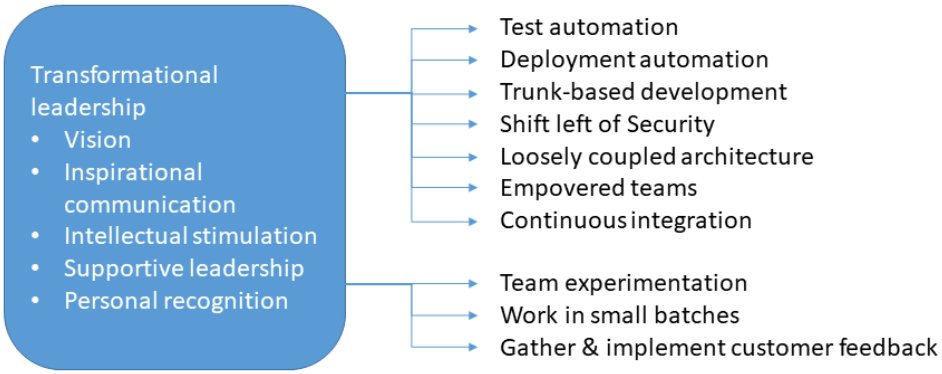

Figure 7 Effects of transformational leadership [10, p. 121]

3.2.3 Lean

Based on Sambinelli's and Borges analysis [22], Lean thinking related citations on software engineer research papers published in 2012-2016 are common. Agile software development methodologies related papers had most citations related to lean thinking. From software development practices, continuous integration, continuous deployment, continuous deliver, DevOps and test-driven development are the most common source of lean thinking related citations [22]. 2016 and 2017 State of DevOps surveys [47][48] found out that Lean product management practices positively impacts on software delivery performance, organizational culture and job satisfaction [10 p. 87-88] (Figure 8).

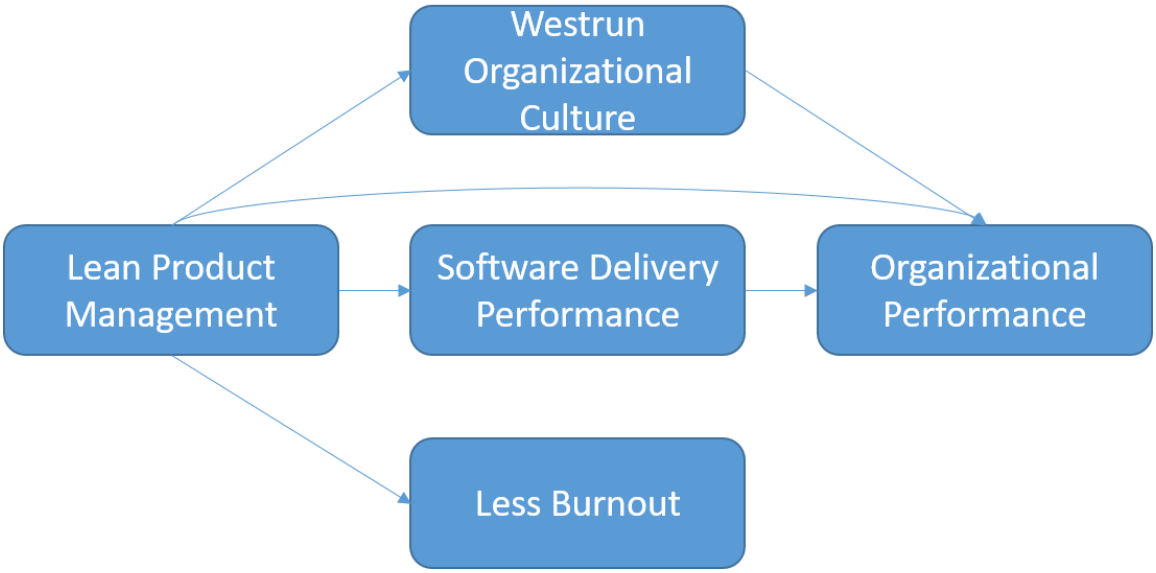

Figure 8 Impacts of Lean product management [10, p. 88]

The State of DevOps research measured work in small batches, making work visible, gathering and implementing customer feedback and team's experimentation. All these Lean product management practices predict software delivery performance, generative organizational culture and decreasing burnout. [10.]

Poppendicks [23] introduces seven lean principles to software development

1. Eliminate waste
2. Amplify learning

3. Decide as late as possible
4. Deliver as fast as possible
5. Empower the team
6. Build integrity in
7. See the whole

Waste is anything that does not add value to the product [23, p. xxv]. According to Lehtonen, there are three types of waste in the software delivery pipeline: defects, waiting and extra features (Figure 9). The amount of these should be minimized. Figure 9 describes where the waste is coming. The defect rate is kept low by fast feedback and automated quality assurance. Waiting time is kept low by automating pipeline steps. Extra features are tried to keep low by getting feedback from end users. [24.]

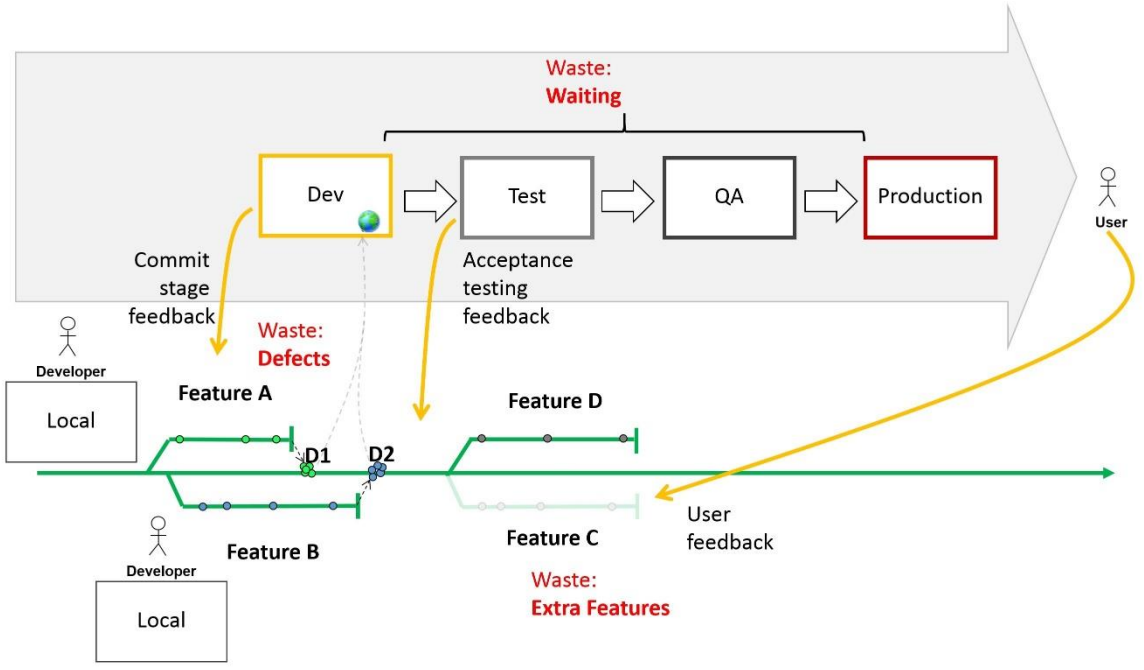

Figure 9 Wastes in deployment pipeline [24]

Poppendiecks, mapped the seven wastes of manufacturing to the seven wastes of software development (Figure 10) [23, p. 4].

The seven wastes of manufacturing	The seven wastes of software development
Inventory	Partially done work
Extra processing	Extra processes
Overproduction	Extra features
Transportation	Task switching
Waiting	Waiting
Motion	Motion
Defects	Defects

Figure 10 Seven types of waste in software development [23]

3.2.4 Continuous Integration and Continuous Deployment

Continuous deployment is an ability to release software into production, frequent and reliable with as much automation as possible [25]. Continuous integration is usually part of continuous delivery. In continuous integration, development code is merged to master/trunk daily [15]. Continuous Integration (CI), Continuous Deliver (CDE) and Continuous Deployment (CD) are all slightly different things. CI practices include committing and merging code into master/trunk frequently. Continuous integration practices include automated building and testing. CDE aims to have software all the time in releasable state but not yet deployed to production environment. In CD changed are automatically deployed to production. [26.]

Characteristics for CI are single source repository, automated builds, automated tests, daily commits to master, a build for each commit, fixing broken builds immediately, fast builds, test in production like clone environment, accessible artifacts and transparent process [27].

The 2014 State of DevOps survey shows that smaller increments and merging the code into the trunk are strongly correlated with IT performance and organizational performance [28]. The State of DevOps surveys also shows that Delivery performance affects to organizational culture [10, p. 218].

Shahin, Zahedi, Babar and Zhu [29] studied how the software architecture affects to Continuous Delivery (CD) and deployment. They found that monolithic architectures do not prevent CD adoption, but CD adoption is easier with small independent deployment units. The study also states that the CD requires high quality tests that run easily. Reusability thinking can cause inter-team

dependencies and more difficulties for testing. The demand for software architects to take account CD requirements, expands the role of architect. Logging, metrics and collaborating with operation personnel and understanding the production environment are key factors for designing operations-friendly architecture. Decomposing a monolithic software to smaller pieces may increase the complexity at the deployment phase. With small deployment units, should be kept independent in terms of deployability, modifiability, testability and scalability. For CD driven architecture, quality attributes such as deployability, modifiability, testability, monitorability, loggability and resilience are essential. [29.]

Version Control System

Version Control System (VCS) is a system that records changes made by the software developers. VCS is also known as Revision Control System (RCS), Software Configuration Management, Source Code Management (SCM), and Source Code Control [30]. VCS keeps track of changes and manages switching between different version of source code or other files [31]. Version control systems have two different approaches, Centralized Version Control Systems CVCS and Distributed Version Control Systems (DVCS). Version control is seen as a prerequisite for CI and CD pipelines [10].

CVCS is a centralized approach where code repository is managed in one central place (Figure 11). In a DVCS each developer has own local repository [30]. According to research done by Brindescu, Codoban and Shmarkatiuk [32], developers do more often commits to DVCS and code commits are even 32% smaller than CVCS repositories.

Version Control System	CVCS	DVCS
Repository	There is only one central repository which is the server.	Every user has a complete repository which is called local repository on their local computer.
Repository Access	Every user who needs to access the repository must be connected via network.	DVCS allows every user to work completely offline. But user need a network to share their repositories with other users.
Example of VCS Tools	Subversion, Perforce Revision Control System	Git, Mercurial, Bazaar, BitKeeper
Software Characteristics that suitable	i. Projects that allow only several users to contribute to the software development. ii. Team located in a single site.	i. DVCS is suitable for a single or more developers because the project repository is distributed to all the developers and this ability offer a great improvement for the projects. ii. It also can be applied for a small or big software projects because it makes less difficult for normal users to contribute to the development. iii. Team located in multiple site or different countries and different timezones.

Figure 11 Differences between CVCS and DVCS [30]

The State of DevOps surveys shows that version control is a key capability in software delivery performance [10, p. 201]. The DevOps handbook suggests that team should put all production artifacts to version control, which includes all application code and dependencies, any scripts, all environment creation tools and artifacts, Dockerfiles, test automation code and libraries, all project documentations, release notes, deployment procedures and all configuration files [15, p. 115-118].

A VCS can also help developers to understand the production environment by storing infrastructure artifacts in infrastructure as a code (IAAC) style and other maintenance scripts in the version control [33].

Trunk-based Development and Feature Branch Development

Version control systems have different workflow models. DevOps literature favors trunk-based development. Feature branch driven workflow models, such as feature branch workflow and gitflow, are also common.

Trunk-based development is a version control branching model where development is done in mainline branch “trunk”. Trunk-based development is seen as key enabler of CI and CD [10, p. 55-56]. Usually commits to trunk is done with merge requests to enable code-review and build checking. With trunk-based development, team can avoid big merge problems. [34.] In trunk-based development, master branch can have unfinished code. Feature toggles can be used to avoid running unfinished code in production [15, p. 171-173].

Feature branch development, developers develop features in own branches. Branch is merged to trunk once the feature is ready. Feature branch development can work well on small teams where the feature branch count is low and when the features are isolated. When feature development touches same files and code, merge conflicts will happen. Solving merge conflicts can be time consuming and frustrating. To avoid huge problems in a feature branch driven development, team must have a good communication so that changes in different feature branches do not cause big problems when the feature is merged to mainline. [35.]

3.2.5 Testing in DevOps

The 2014 State of DevOps report states that automated testing included into continuous integration and delivery pipeline is highly correlated with organizational performance. Automated tests enable fast feedback loop for developers. The feedback loop encourages learning. Easily reproducible test environment and test case helps the developer to fix the problem. [28.]

In DevOps testing share same foundations as testing in a non-DevOps. In DevOps, the whole team is responsible for testing. Also specialized tests like performance and security, which in the traditional development model may have been part of a specialized team, are part of the team's duty. Test specialists in team are responsible to teach testing strategies to the team. Tests facilitates fast feedback loops while also taking care of quality assurance. [36.] Fowler introduced test pyramid (Figure 12) as objection to test cone. In test pyramid, comprehensive unit tests are the base of the quality assurance. Unit tests are cheap to create and fast to run. In the middle are service level tests, which include automated component tests, automated integration tests and automated API tests. These tests are little more costly to create and slower to run. At the top of the pyramid are UI tests. Above the pyramid as a cloud, are manual tests. In test cone, the pyramid is upside down where manual tests are ice cream in the cone. [15, p. 129-133] Various blog posts also provide criticisms towards the test pyramid, but still accepting it as better representation than the test cone model.

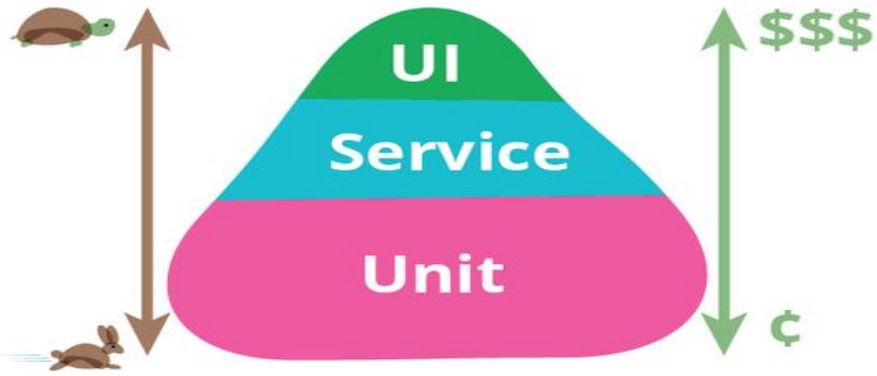

Figure 12 Test pyramid [37]

In DevOps, where fast feedback is crucial, manual testing is in pressure. Human checks are slow and reduce the deployment pipeline performance. However, manual testing has an important role. Some tests make more sense in manual testing than test automation. Exploratory testing and look and feel tests, for example, remain manual tasks. Test engineers are responsible to reveal problem and provoke discussion about potential risks. [38.] Figure 13 represents a model from DevOps handbook, where automatic and manual tests are executed parallel. The model shows different category of tests ran in parallel. The book also suggests running the automated in parallel to reduce execution time. [15, p. 133-134]

Automating the manual tests is one of elements of DevOps. However, only automating all the manual tests do not lead to desired outcomes. Automated tests should be reliable and avoid generating false positive results. False positive results swiftly eat the motivation to keep tests passing. False positives also generate waste in the testing process. Common causes of false positives are problems in deployment to staging environment, slow performance, uncontrolled starting stage and test scenario initialization. [15, p. 133-135]

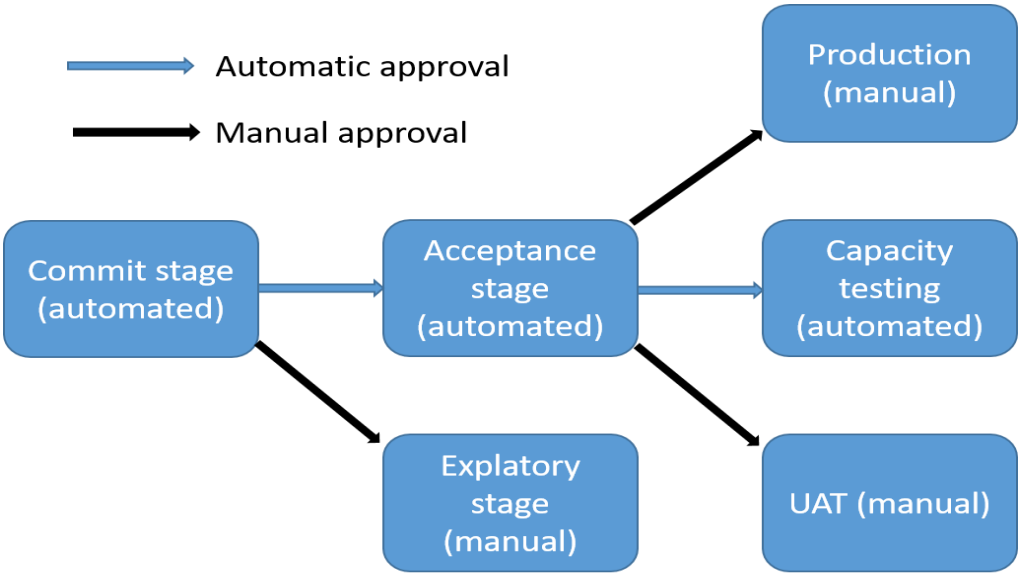

Figure 13 Automatic and manual tests in parallel [15, p. 134]

Shifting Left

In DevOps, practitioners talk about shifting left. Figuratively far left is developer with code and far right is production environment [39]. Shifting left means considering something earlier in the process. Shifting left quality assurance, for example, means that quality inspections are done earlier of the pipeline or built into the development process. Usually shifting left is used when speaking of testing and security, but also any other asset can be paid attention in the early phase of the development. The aim of shifting left is to catch bugs earlier in the development process and shorten feedback loops [40].

The term shifting right is also used in software development. Shifting right means conducting the activity at production environment. For example, shift right in testing can include strategies like dark launches with feature toggles and canary environments. [40.]

Dark launch means releasing a software feature into production, but limiting the access to it. When the feature is proven to increase value, it is scaled up to the whole production environment. The access limitation can be achieved by allowing only a small amount of traffic to the new service (canary testing) and by using feature toggles to turn the feature on/off. Feature toggles increase code complexity, but can help its maintainability. [40.] Feature toggles allow easy rollback, performance degradation on demand and helps to design service-oriented architecture [15,

p. 172]. Feature toggles can also help continuous integration as the feature does not need to be completed before merging to master branch.

Acceptance Test Driven Development

In acceptance test driven development (ATDD), the team writes acceptance tests before implementing a software feature. The main purpose of ATDD, is to facilitate conversation between developers and Product Owners about product requirements. The test writing should include customer perspective, development and testing. Customer perspective brings the focus to what problem is being solved, the development focus on testing broadens the view and help to avoid possible challenges. [41]

The ATDD combined with test driven development (TDD) is an effective way to build quality in the development process. By practicing TDD team can achieve 60%-90% lower defect density while taking 15%-30% longer time to develop. [15, p. 135.]

3.2.6 Measuring DevOps

Mohamed proposed maturity model to measure DevOps maturity level (Figure 14). The Mohamed's model is based on CMMI maturity model. The model contains four dimensions: quality, automation, communications/collaboration and governance. The model also has 5 maturity levels: initial, managed, defined, measured and optimized. To increase the maturity level, the organization must improve all four dimensions. [42.]

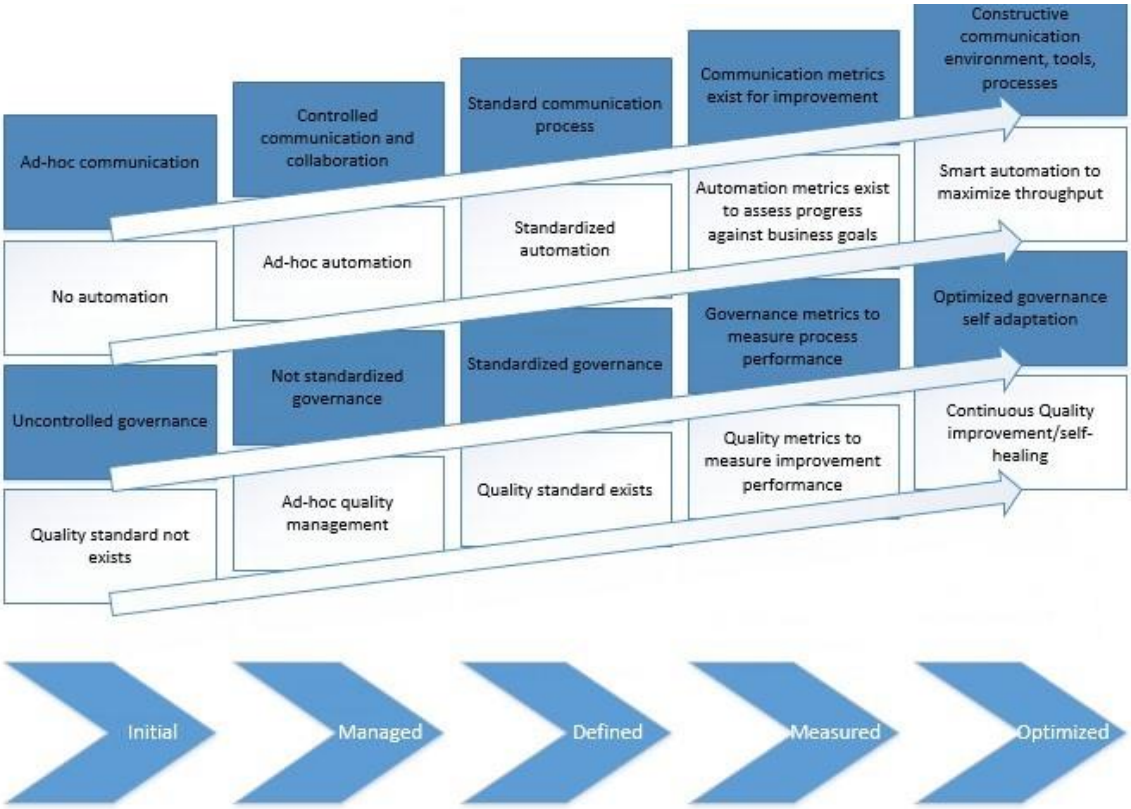

Figure 14 DevOps maturity model [42]

On initial level, there are no process or tools defined for communication, no clear roles between team members. On initial level, decision making is centralized and automation does not exist. Governance is chaotic and processes are not predictable. Quality has no significance and done mostly on an ad hoc basis. [42.]

On managed level, the team communicates a more coordinated way with different stakeholders. Decision making is centralized but decisions are more shared with the team. The team also has better defined roles and responsibilities. However, communication is not shared between teams. Automation and governance are ad-hoc basis and not standardized in organization and processes and customs varies between different teams. Quality starts to have some value but is managed on each project on ad-hoc basis. Proper tools may be used, but are based on project needs. [42.]

On defined level, communication is more concrete. Different departments are informed in a co-ordinated way and involved in the process early. Defined level automation is standardized across the organization. The organization provides support, training and standardized framework. Governance and processes have organization standards. Each project has ability to tailor the standard

process based on its needs while still keeps fitting the organization process framework. Quality also has organization wide standards and tools, which are adopted by teams. [42.]

On measured level, each dimension has metrics, which are used to improve each dimension. Automation helps collecting the metrics and metrics are brought visible. On optimized level, each dimension is optimized towards organization goals and objectives. [42.]

Mohamed also describes how the model can be used. The first is an initial assessment phase, where a gap analysis and assessment of the current state is done. The assessment should identify main pain points. After initial assessment phase, follows a roadmap identification phase where the whole transformation project is planned. From the roadmap identification phase, the organization should have next steps ready: which teams to start, set of processes to be modified, set of tools to be acquired and quick wins. [42.]

After the roadmap identification phase, follows a transformation execution phase. At this phase begins the transformation from current state towards the target state. After the transformation phase comes the verification phase where the transformation is verified by using metrics. Finally, comes the optimization phase where it is ensured that whole organization benefit of the new way of work. Continuous improvement is done organization wide. [42.]

Metrics

In agile and lean software development metrics are used for planning, process tracking, to understand and improve quality, identify and fix process problems and to motivate people. In agile software development metrics are focusing on post-release quality by using pre-release metrics instead of tracking the progress of predefined plans. Usually metrics are concentrating on the actual product and features instead of requirements, specifications and design documents. Metrics can be misused in multiple ways. Improper metrics may guide team for counterproductive patterns and measuring individuals may lead into difficult situations. [43.]

Based on systematic literature review, Kupiainen et al.[43] identified characteristics for good metrics are ease of use and ability to utilize existing tools, metrics which provoke discussion and metrics, which provide visibility of problems. Commonly used metrics are velocity, defect count and customer satisfaction. Software teams rarely use business related metrics like Return of Investment, Net Present Value or Internal Rate of Return. [43.]

In the State of DevOps study, Forsgren, Humble and Kim used four metrics to measure software delivery performance: delivery lead time, deployment frequency, time to restore from failure and change fail rate. Cluster analysis showed that the meters did predict software delivery performance. [10, p. 37-38.] Earlier, Farlay and Humble had stated cycle time as the most important software development metrics [44].

Lead time in the context of software development can be defined in two parts, the first part is time to design and validate the product or feature, and time to deliver the feature to a customer. The first part is hard to measure, whereas the second part of the lead time is easy to measure and has lower variability. [10, p. 14.]

Deployment frequency is how often software is delivered to production or released available for customers. Deployment frequency is used as a proxy metric for batch size as batch size is hard to measure itself. [10, p. 16.]

Restore from failure metric is measured by how fast the team can restore service back in stable production state after a failure. Change fail rate meters the percentage of change into production caused a failure, which requires remediation e.g. service impairment or outage, require a hotfix, a rollback, a fix-forward or a patch. [10, p. 17.]

Lehtonen [24] identified metrics, which can be used to measure continuous software development. Lehtonen presents three metrics to measure latencies in deployment pipeline: development time, deployment time and activation time. Source for the metrics are version control systems, issue management system and production logs and can be automatically generated.

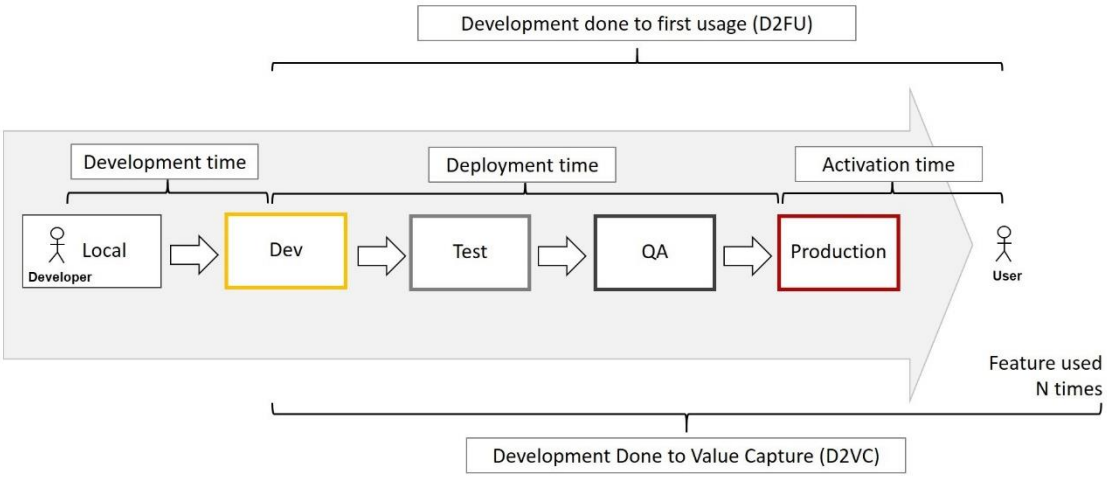

Figure 15 Software process metrics [24]

In the pipeline, which Lehtonen presents, there are five different environments: developer's own environment, local environment, development environment, test environment, QA environment and production environment. Development time is the time it takes for the team to implement a new feature. It can be measured in a feature branch driven development model from branch created with the branch merged with the master branch. [24.]

Deployment time is the time it takes to new feature to be deployed into production environment. The deployment time measurement has two dimensions, the execution time it takes for tools to run the pipeline and time, which includes all the delays caused by project management and QA testing. [24.]

Activation time is the time from releasing to production to the first user to use the feature. The activation time can be hard to measure, without sufficient logging. The activation time metrics gives feedback to the development team about the feature usage, which can be used when deciding which features are developed in the future. [24.]

Lehtonen also recommends metrics for delivery pipeline. These metrics include features per month, releases per month, fastest possible feature lead time. Lehtonen emphasizes visualizing the chosen metrics with graphs and sharing the information to the team with radiators. [24.]

3.3 The State of DevOps Research

In 2013 Nicole Forsgren, Jez Humble and Gene Kim started a research project to answer the question, what capabilities and practices are important to enhance software company's profitability, productivity and market share [10]. The annual state of DevOps report launched in 2012 by Alanna Brown, senior product marketing manager at Puppet Labs [45]. The research project covers findings from the State of DevOps reports from 2014 to 2017. The research goal for each year was different. The response data were not linked between different years [10].

The research was done by surveys as the researchers wanted to acquire a big amount of data from different kind of organizations. The research collected over 23000 survey responses from over 2000 unique organizations. The organization's size varied from small startups to large enterprises. The companies represented many different industries and technologies. Questions were mainly Likert-type questions [10.]

The 2013 survey shows that high performing organizations share two common practices: they use version control and has automated code deployment. The term "code deployment" is covering the deployments not only for production environments, but also to test and staging environments. The version control is used as the single source of truth. High performers use version control also for the infrastructure automation and configuration code. [45.]

The report also covers difficulties and cultural barriers against DevOps implementations. The biggest difficulty in implementing DevOps was that the value wasn't understood outside of their group. Those who have no plans to implement DevOps answered that lack of manager buy-in was the biggest reason. The second highest reason was a lack of team buy-in. The report suggests to overcome the problems is to start small scale communication with other teams. The 2013 report recommendations for DevOps implementation are automation, version control, breaking culture barriers, use of metrics and encourage lateral communication. [10.]

3.3.1 2014 State of DevOps Report

The 2014 State of DevOps survey got over 9200 responses [28]. The 2014 survey was geared to test hypothesis that IT performance does make a change in organizational performance. The 2013

survey was also investigating same things. The 2014 survey validated the findings from the previous year and delineate the meaningful factors. The survey shows that companies with higher IT performance are twice as likely to exceed their profitability, market share and productivity goals. The study did not find a single reason or formula behind IT performance but shows quantitatively that DevOps practices and IT performance enhance organizational performance. [28.]

The organizational performance was measured by asking respondents to rate their organization's relative performance across several dimensions. The survey suggests that job satisfaction is the number one predictor of organizational performance and DevOps practices increase job satisfaction.

The research questions for the year 2014 study were

- What does it mean to deliver software, and can it be measured?
- Do software delivery impact organizations?
- Does culture matter, and how do we measure it?
- What technical practices appear to be important?

The survey shows that: software development and delivery can be measured in a statistically meaningful way, throughput and stability are tied together, an organization’s ability to make software impacts profitability, productivity and market share and the culture and technical practices matter. [10.]

According to the report [28], top 5 predictors of IT performance are

1. Peer-reviewed change approval process
2. Version control for all production artifacts
3. Proactive monitoring
4. High-trust organizational culture
5. Win-win relationship between dev and ops

The survey revealed following top most correlation relations with software delivery metrics

1. Top practices correlated with deployment frequency are
    a. Continuous delivery
    b. User of version control for all production artifacts
2. Top practices correlated with a lead time for changes
    a. Use of version control for all production artifacts
    b. Automated testing
3. Top practices correlated with a mean time to recover
    a. Use of version control for all production artifacts
    b. Monitoring system and application health

Deployment frequency, lead time for changes and mean time to recover from failure was tested to be meaningful measurements. Change fail rate was not significantly correlating with the IT performance. However, high performing organizations had 50% lower change fail rates than medium and low performing IT organizations. The research did not find a link between specific practice and change fail rate. The report states that DevOps maturity was highly correlated with deployment frequency. "The longer the development and operation team practice DevOps, the better they get, leading to higher deployment frequency." [28.]

The researchers chose Ron Westrum's organizational culture typology model as the basis for measuring DevOps culture in organizations. The Westrum's organizational culture was chosen as it forms a "Westrum continuum", which is easy to transcript to Likert type questions. The research shows that the Westrum's organizational culture affects software delivery performance and organizational performance. Based on the survey [28], top predictors of organizational culture are

- Job satisfaction
- Climate for learning
- Win-win relationship between dev and ops
- Version control

- Automated testing

The researcher hypothesized that job satisfaction affects organizational performance. Job satisfaction was found to be the best predictor of organizational performance. The survey shows that automation of menial tasks and empowering the people to make decisions based on feedback loops correlates strongly with job satisfaction. Naturally, high-trust organizational culture and climate of learning were predictions for job satisfaction, but also the correct tools and resources are important. Based on the 2014 report [28], top predictors of job satisfaction are

- High-trust organizational culture
- Climate of learning
- Win-win relationships between operation, development and infosec teams
- Proactive monitoring and auto-scaling
- Use of version control for all production artifacts
- Automated testing

3.3.2 2015 State of DevOps Report

In the 2015 study, the research team reevaluated some of the findings from the 2014 research. New research questions were:

- Do technical practices and automation impact software delivery?
- Does Lean management practices impact software delivery?
- Do technical practices and Lean management practices impact aspects of work that affect our workforce -- such as anxiety associated with code deployments and burnout? [10.]

4976 respondents completed the survey. Respondents were again from different geographic locations and industries and organization sizes varied from small under 100 employee companies to 10000+ employee sized organizations. Five percent of the respondents were women. [39.]

The survey results helped the researchers to form a quantitative definition of IT performance. Chosen metrics for IT performance contains two throughput measures, Deployment frequency and deployment lead time, and one stability measures Mean time to recover (MTTR). Deployment frequency is measuring how frequently the organization deploys code. Deployment lead time is measuring time from code committed to code successfully running in production. Mean time to recover is time required to restore service when a service incident occurs. [39.]

The survey results show that high performing organizations deploy in higher frequency, but also have lower mean time to recover. The research also asked about change failure ratio, but it was not part of the IT performance construct. High performing organizations reported lowest failure rates. Compared to 2014 survey results, throughput measures of high performers did not increase but stability did increase significantly. Researchers assume that the stability increase was due taking the quality factors account in earlier stages of the development. [39.]

For the 2015 study, researchers built two new constructs to model how continuous delivery and Lean management practices affect IT performance and organizational performance. The survey results show that the constructs hold true. Continuous Deliver and Lean management practices, such as visualizing the work, setting work in progress (WIP) limits and use of monitoring tools, reducing batch size and shortening the cycle time, manage team's workload and visualize work queues indeed increase IT performance, which increase business performance. Findings suggest that continuous delivery practices and Lean management practices amplifies each other. [39.]

Software architecture related questions revealed that testability and independently deployable applications and micro-service architecture correlates with high performance. The findings correlate with Shahin et al. found similar results in their research [29]. The 2015 State of DevOps survey found also that it did not matter if software or system is

- Packaged commercial software/Commercial off-the-shelf
- System of records
- System of engagement
- New, not-yet-deployed
- Software with an embedded component that runs on a manufactured hardware device

- Software requiring a user-installed components that runs on the user's machine

All these types of software can benefit of applying DevOps practices, if attention has been paid to design and to enterprise architecture of the ecosystems where they live. [46.]

The study also shows that in high performing organizations, when adding more developers to project, both overall productivity and developer's individual productivity raise. On low performing organizations, individual performance decrease. In mid performing organizations, developer's individual productivity remains same when more people are added. [46.]

Addition to the 2014 State of DevOps finding, the 2015 survey reveals that top 7 measures to correlate to organizational culture are:

1. Organizational investment in DevOps
2. The experience and effectiveness of team leaders
3. Continuous delivery practices
4. The ability of development, operations, and InfoSec teams to achieve win-win outcomes
5. Organizational performance
6. Deployment pain
7. Lean management practices

Leadership was found to correlate with Westrum's organizational culture. Respondents who had effective team leader reported to have generative organizational culture. Effective leadership also correlates with helping teams achieve win-win outcomes, creating feedback loops and use of continuous delivery practices. [46.]

3.3.3 2016 State of DevOps Report

The third year of the research added more technical practices into the study, such as security, trunk-based development and test data management. The research questions for the year 2016 were [10]

- Does the integration of security into software development and delivery help the process or slow it down?
- Does trunk-based development contribute to better software delivery?
- Is a Lean approach to product management an important aspect of software development and delivery?
- Do good technical practices contribute to strong company loyalty?

The 2016 State of DevOps report got over 4600 responses. The background of the respondents remained approximately same as in previous years. 2016 Survey reproduced previous year results; high performing organizations have much higher deployment rate, shorter change lead times and shorter mean time to recover. High performers had significantly improved deployment frequency from previous years while low performers have maintained same deployment frequency. At change failure rate (CFR), low IT performers had a smaller change failure rate than medium IT performers. [47.]

The 2016 survey wanted to find out, if built in quality and security works. Researchers used unplanned work as a metric. Based on survey responses, high performing organizations spend 21% of time for unplanned work when low performers spend 27% of the time. High performers also spend less time (30% vs 35%) on other work, such as meetings, routine maintenance, etc., and more time for new work (49% vs 38%). The study also shows that built in security does not slow down development and high performers spends significantly less time on fixing security issues. [47.]

One of the 2016 State of DevOps report finds is that test data management is important. The report suggests that test data require careful maintenance and teams should try to minimize the data needed for test cases. Instead of big database dumps, test suite should start from empty state and setup the needed data via an application programmable interface (API). [47.]

The researchers wanted to find out if trunk-based development improves delivery performance. Findings suggest that branches should be merged to trunk daily basis. Teams that don't have code freeze periods also achieve higher performance. [47.]

2016 State of DevOps survey was geared to find out, if small batch size, better understanding of the flow of work from the business to customers and customer feedback collection and usage

predicts better IT performance. Results show that all three constructs hold true. The study also shows that smaller batch sizes, and understanding the flow of work through the product development and delivery process goes hand in hand. [47.]

The survey also studied employee's motivation. Employee net promoter score was found to significantly correlate with customer feedback collections and feedback usage in product design, ability to visualize and understand the flow of products or features through development and employee's value and goals match to organizations values and goals. Survey results show that respondents who worked in organizations that uses continuous delivery practices and lean product management practices also agreed with following questions

- I am glad I chose to work for this organization rather than another company.
- I talk of this organization to my friends as a great company to work for.
- I am willing to put in a great deal of effort beyond what is normally expected to help my organization be successful.
- I find that my values and my organizations values are very similar.
- In general, the people employed by my organization are working toward the same goal.
- I feel that my organization cares about me.

People who identified with their organizations predicted better organizational performance. [47.]

3.3.4 2017 State of DevOps Report

The 2017 study was focusing on system architecture and leadership. The researchers also wanted to see if DevOps practices improve non-profit outcomes. The research questions were [10]:

- What architectural practices drive improvements in software delivery performance?
- How does transformational leadership impact software delivery?

- Does software delivery impact not-for-profit outcomes?

The year 2017 survey got 3200 responses around the world. Respondents were working on a different kind of organizations and a different kind of positions. Each year, a percentage reported working in dedicated DevOps teams has risen, now 27% of respondents reported to work on DevOps team. [48.]

The 2017 State of DevOps survey used Rafferty's and Griffin's five dimensional transformational leadership model [21]. Survey respondents who worked on high performing teams reported their leaders have behaviors from all five dimensions, whereas the leaders of low performing teams were reported to have the least characteristic of the five dimensions. The study also shows that transformational leadership also correlates with employee net promoter score (eNPS). The survey did not study Westrum's organizational culture this year, but researchers expect, based on the results of previous years studies, that transformational leadership also predicts generative organizational culture. However, the study shows that transformational leadership is not enough to achieve high IT performance. [48.]

The 2017 questionnaire repeats the IT performance metrics: deployment frequency, lead time for changes and MTR. Also, the change failure rate, which was not part of the IT performance construct was measured. Compared to 2016 results, low performance organizations narrowed the gap in deployment frequency and lead time for changes, but high performers widened the gap in MTR metrics. [48.]

The year 2017 study asked how much the teams have automated their work. Results show that high performing teams have automated significantly higher percentage of their work. Used questions in the survey were about configuration management, testing, deployment and change approval. [48.]

The survey shows that, high performing teams had automated

- 33 percent more of their configuration management
- 27 percent more of their testing
- 30 percent more of their deployments
- 27 percent more of their change approval processes

more than low performing teams. Mid performers reported quite similar percentages as low performing teams, but mid performance seems to do more manual work in deployments and in change approval processes. However, the report states that teams are not good at estimating their level of automation. The report assumes that the more manual work is due freed time to reduce technical debt, which has caused them to institute manual controls around changes. The report advises to not add the manual step even organization has a high temptation to do so. [48.]

The 2017 survey tried to study, if DevOps practices impacts also on organizations non-profit goals. The results show that high performing organizations were twice as likely to achieve the following objectives

- Quantity of products or services
- Operating efficiency
- Customer satisfaction
- Quality of products or services provided
- Achieving organizational and mission goals
- Measures that demonstrate to external parties whether or not the organization is achieving intended results

No matter if the organization was for-profit or non-profit organization. [48.]

Previous year surveys show that comprehensive use of version control, continuous integration, shifting left at security, and automated testing contributes to continuous delivery performance. The 2017 survey concentrated on continuous delivery itself. The results show that continuous delivery contributes to lower deployment pain and higher IT performance. The 2017 survey also shows that loosely coupled architecture and teams significantly improves the ability to practice continuous delivery. The questions to find out the coupling was asking the respondents if they can do testing without integrated environment and if the release can be done independently without another application and services that depends on the module. [48.]

The study shows that the biggest contributors for continuous delivery are related to team empowerment. If team can

- Make large-scale changes to the design of its system without permission from someone outside the team

- Make large-scale changes to the design of its system without depending on other teams to make changes in their own systems, or creating significant work for other teams

- Complete its work without needing find-grained communication and coordination with people outside the team

- Deploy and release its product or service on demand, independently of other services the product or service depends upon

- Do most of its testing on demand, without requiring an integrated test environment

- Perform deployments during normal business hours with negligible downtime

The 2016 survey reveals evidences that trunk-based development correlates with IT performance. 2017 questions investigated it further and tried to find out how long-lived branches should be. The results show that high performing teams have the shortest branch lifetime and the shortest integration times. Also, low performers have the longest branch life time and the longest integration times. The report suggests that teams should merge branches into master daily basis. If the branch time grows longer, the team should review development practices and architecture. [48.]

The 2016 State of DevOps study revealed that, the small batch size and making work visible with status radiators and effectively using customer feedback for product development predicts higher IT performance and lower deployment pain. The 2017 study flipped the model other way and found that IT performance predicts lean product management practices. To be able to deliver faster in small batches helps to gather customer feedback and do more experiments. The research shows that if the team is able to try new things and experiment ideas with real customers without the need to ask permissions from outside, it predicts organizational performance. The experiments should be based on information provided by feedback loops. [48.]

3.3.5 DevOps capabilities identified in State of DevOps research

24 Key DevOps Capabilities in 5 Categories [10]

1. Continuous delivery
2. Architecture
3. Product and process
4. Lean management and monitoring
5. Cultural

Continuous Delivery Capabilities

1. Version control
2. Deployment automation
3. Continuous integration
4. Trunk-based development
5. Test automation
6. Test data management
7. Shift left on security
8. Continuous delivery

Architecture Capabilities

9. Loosely coupled architecture
10. Empowered teams

Product and Process Capabilities

11. Customer feedback
12. Value stream
13. Working in small batches
14. Team experimentation

Lean Management and Monitoring Capabilities

15. Change approval processes
16. Monitoring
17. Proactive notification
18. WIP limits
19. Visualizing work

Cultural Capabilities

20. Westrum organizational culture
21. Supporting learning
22. Collaboration among teams
23. Job satisfaction
24. Transformational leadership

4 Current Practices of the Team

In this chapter, current development practices of the team are described. The chapter describes the team's way of work and tools in use. In this chapter, practices are only observed and analyzing is done in chapter 5.

4.1 Scrum Implementation of the Team

The work process of the team is derived from Scrum framework, but it lacks some of the disciplines of pure Scrum. The team consists 3 development teams, which each has own Scrum Master and Product Owner. Development teams are quite loose, meaning that people may switch from one team to another depending on the skills needed. Each team have responsibility of different parts of the software. Scrum of Scrums has been used to coordinate work between teams. Each team holds own sprint planning, and daily Scrum meetings. The sprint demo has been used together with all development teams.

Retrospective meetings have not been done regularly. Also sprint demos has been canceled time to time. The team is not always able to finish tasks in one sprint time. Unfinished tasks are moved to the next sprint. Sprints rarely has defined goals. Instead, the team operates more on demand basis. The focus of the sprint may have been changed due emerging issues but sprints are not canceled. Before, sprint backlogs have not been easily available, but recently corrective actions have been made. However, the team still does not use information radiators like task boards or visible metrics.

The Scrum guide defines the Product Owner as customer voice and owner of backlogs [4]. Team's documentation defines Product Owner's tasks as

- Defines release content
- Defines sprint content
- Defines feature and bug fix priorities
- Arranges regular backlog grooming sessions

- Approves features before merging code into master branch

The customer's voice role is not emphasized in the team's documentation. According the documentation, feature definitions are the joint work of the Product Owner, UI Designers, the software architect and quality assurance lead. Documentation advice to keep tasks smaller than 3 days amount of work and defined before assigning tasks into sprints. In practice, work tasks can be much larger than 3 days amount of work. Many times, tasks are also not well defined before sprint planning.

From anti-pattern list described in 3.1.3 team has long or non-existent feedback loops, hours in progress monitoring, business as usual. Several other list items can be seen in some degree.

4.2 Development Process

The team has defined development process. The team has definitions of each team role, sprint schedule for daily level and description for each Scrum ceremony. The documents also tell how each release should progress and which actions each role has at each stage of the development. The documents also describe the meaning and the goal of different development phase and meaning and actions for each ticket status.

The team uses feature branch model illustrated in Figure 16. When feature implementation begins, a new branch is created. Once the feature is finished and the merge review approved, the feature branch is merged to master branch. Testing and test automation development is done in feature branch and after the merge, integration test is done on the master branch. Ideally, test automation development and feature development happen same time, but in practice that is not always the case.

A feature is considered as "done" after integration tests are passed. The feature can be deployed in production once software release is public. Specialists takes the new release in use. Activation time can easily grow longer. Activation times are not measured.

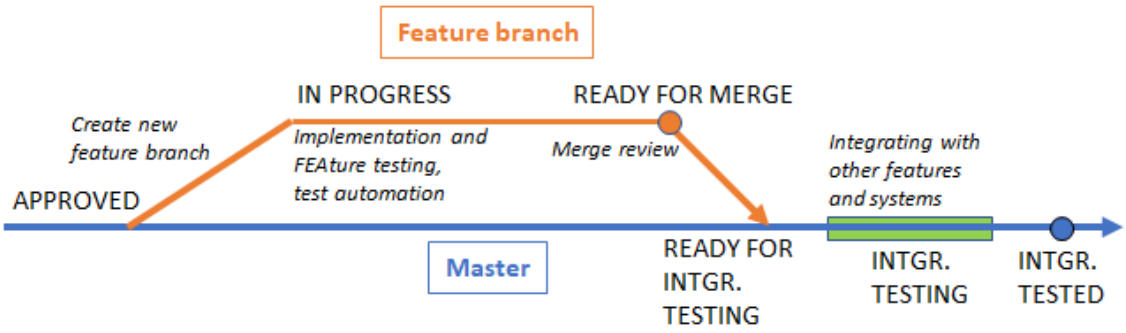

Figure 16 Feature development branching model

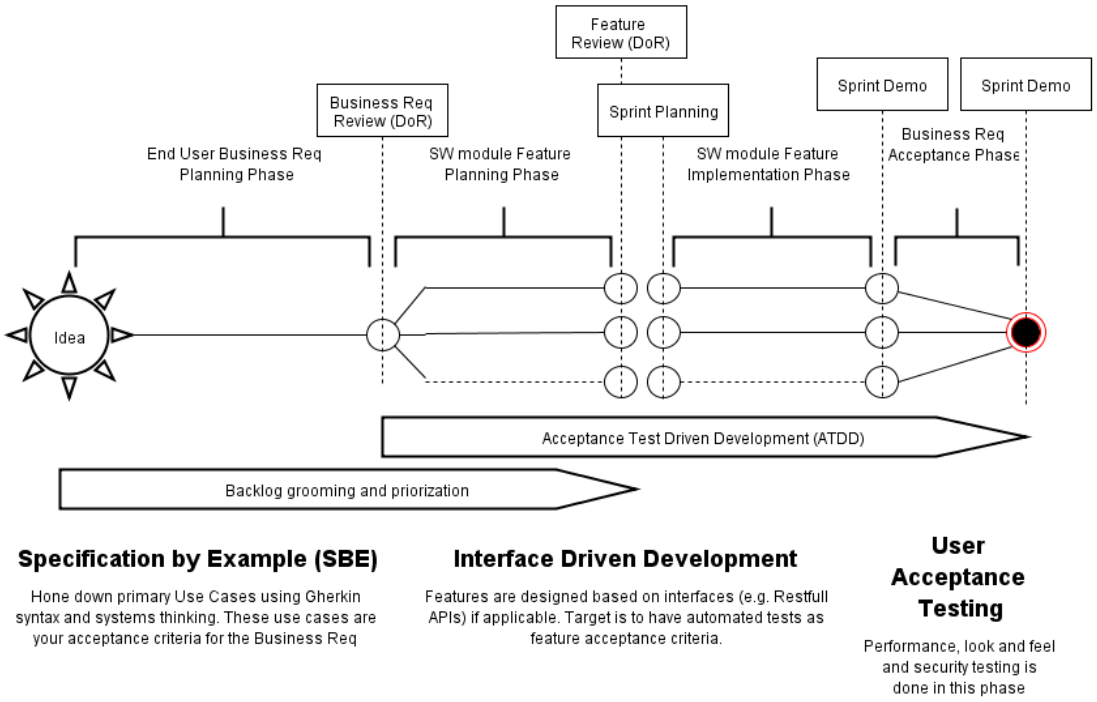

Figure 17 Feature implementation process

Figure 17 illustrates the process from feature idea to implementation. Each idea should be specified as user story specified by example. Development should follow interface driven development and final acceptance is done in user acceptance testing. The development process documentation advice that each story should follow INVEST model

- Independent – they can be developed in any sequence and changes to one User Story don't affect the others.

- Negotiable – it's up for the team to decide how to implement them; there is no rigidly fixed workflow.
- Valuable – each User Story delivers a detached unit of value to end users.
- Estimable – it's quite easy to guess how much time the development of a User Story will take.
- Small – it should go through the whole cycle (designing, coding and testing) during one sprint.
- Testable – there should be clear acceptance criteria to check whether a User Story is implemented appropriately.

The development process has four phases

1. End user business requirement planning phase
2. Software module feature planning phase
3. Software module feature implementation phase
4. Business requirement acceptance phase

Each phase has defined target and roles who own the phase and roles who supports and implements the feature.

Some of the team's documentation was created after DevOps assessment described in 4.7. New documentation describes process, which is not yet working as in documentation suggests. Old documents are partially outdated. However, the documents define process in different abstraction level and therefore complements each other. Team's current practices do not completely follow either old or new documentation, but contains practices from both definitions. The development process in practice is evolving.

4.3 DevOps Practices

On the team, DevOps is only partially applied and is focused only on the development side. Therefore, using term DevOps is a little bit misleading. DevOps in the team can be seen as integrating development and testing activities more closely together with evolving CI pipeline. Operation side is not yet in the picture.

During the year 2019, work on CI pipeline and test automation infrastructure has been done. Also, the way of work has been changed. Before the change, testing started only after the code had been merged to master. Therefore, testing has been always behind the development and feedback time from testing to development grew long. Also, as the untested code was merged into master, the master branch was in releasable state only after code freeze and regression testing period.

Currently, the master is close to a releasable state. However, software has technical debt and multiple known issues. Recently new feature development has been reduced to take down the technical debt. New test automation infrastructure helps to detect regression faster when code changes are made. To run a complete test set on feature branch is not practical because long execution times. The test automation coverage is limited and does not have performance tests. Test coverage is not measured.

Test automation is mainly used for testing whole solution. The team has limited capability to test connections to other services. Component tests are relying on unit tests, which are created by developers. Unit tests do not have coverage analysis enabled.

4.4 Staging Environments

Test environment creation has been automated by using Ansible playbooks and vSphere API. The pipeline relies on pre-created virtual machine template, where all software components are installed automatically. Jenkins tool is used for orchestrating builds and deployments.

Figure 18 describes staging environments, which are numbered from 1 to 4. Environment ones are cattle environments, which are automatically created by Jenkins end-to-end testing job when a new code branch is committed to GitLab. Cattle environments have the same life span as git

branches. Environment twos are master branch test environments. Master test environments are used for running tests whenever new code is merged to master branch and for a nightly regression test run.

Environment threes are UAT pet environments, which are long lived environments for UAT testing. Each tester, UI designer or developer may have own environment, which software can be updated with a Jenkins job. Running the Jenkins job must be triggered manually. The deploy job allows to choose, which git branch software are installed. Environment fours are long living pet environment for release branches. Each release will get own pet environment.

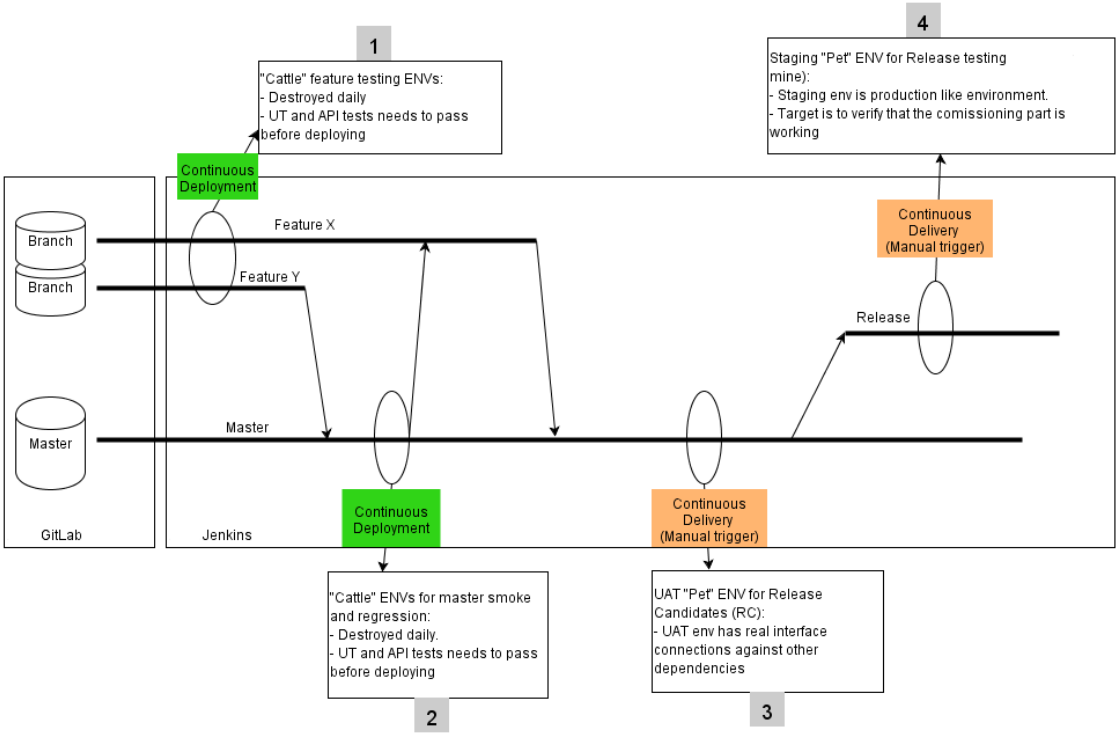

Figure 18 Staging environments

In practice staging environments 2 and 4 has not been used. Instead, master branch has got one pet environment instead cattle environments. Releases have not yet got pet environments. At release phase, release is tested with cattle environments and tester's own pet environments.

The environments are set up on server rack in office, which is in own isolated test network. The other office has limited access to the network. From organization's intranet there is only limited access to the test network. Connectivity difficulties are an obstacle to take full advantage of the test environments.

4.5 Software Code Repositories and CI Pipeline

The team develops multiple software components. Each component has an own git repository in GitLab. Each software component has own Jenkins build pipeline. The build pipeline gets the source code from the git repository, builds the code, run unit tests and push installer into file share. Figure 19 shows an example of Jenkins CI pipeline used by the team.

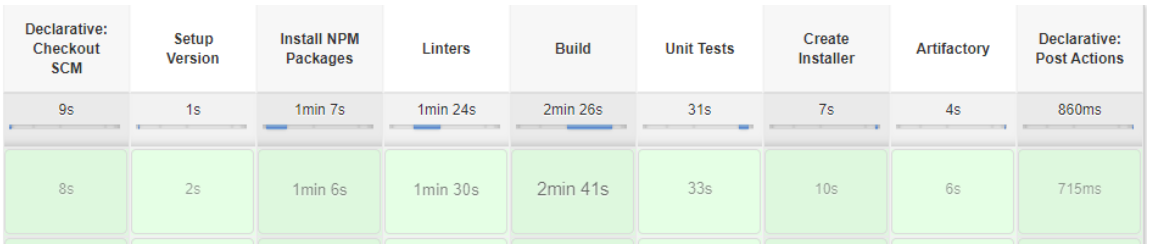

Figure 19 Software component CI pipeline

The CI pipeline contains versioning, package fetching, static analysis, build, unit test, installer creation and pushing installers to file share. CI pipeline contains major parts of a modern CI pipeline, but lacks security analysis, metrics and notifications. Build failures are fixed fast once they are noticed. As the pipeline does not have notifications, failures are not always detected immediately. MTTR metrics does not exist. Some of the software repositories does not have a unit test stage and some do not have a static analysis.

For end-to-end testing, a Jenkins multi branch pipeline is used. The end-to-end pipeline job watches all software component pipelines and test automation repository. The pipeline spawns a virtual machine on the local vSphere cluster, downloads all needed installers from the file share, install the software and run automated test cases. Automated test cases are Robot Framework test cases and Jmeter test cases.

End-to-end test pipeline run time with smoke test set takes approximately 18 minutes. Smoke test set contains 23 test cases. Pipeline takes 3hours 40minutes to run the nightly regression test set. The nightly regression test set contains approximately 450 test cases. The team has increased test automation coverage and the test case count, and test coverage increases every week.

The end-to-end pipeline does not gate builds. Test case errors are analyzed by test engineers. The test engineer creates issues based on findings. The pipeline does not have automatic notifications. Robot Framework test results are published on Jenkins with Jenkins robot framework plugin

and Jmeter test case results are published on Jenkins with Jenkins performance plugin. Test results are not visible in any other source than Jenkins.

4.6 Software Architecture

Software architecture consists of different software modules, but the modules are not isolated microservices. Connection between modules is done by REST API, Windows Communication Foundation (WCF) and SignalR. The technology stack is quite simple and is relying heavily on Microsoft technologies. Overall architecture design has become quite complicated. Development process documentation assigns tasks to a software architect role, but the team does not have a dedicated software architect who would take responsibility of overall software architecture. The developers take architect responsibilities, but the overall architecture view is missing.

The software components are hosted on one server when deployed into production. The deployment server is located on customer's premises. The application can provide data for cloud services, but the team is not developing those services themselves.

Some of the components have an own database but also a shared database is used. All data, including time series data, is stored in SQL databases. Storing all data in SQL database is may not be the most optimal solution in performance vice but simplifies the technology stack. Access to the common SQL database is not centralized. Stored procedures are not used.

As the system is deployed on customer's premises, continuous deployment is difficult and impractical. Deployment is done by manually running installers on virtual machines. Virtual machine creation isn't automated either. The team also has lack of visibility of the deployment process and limited feedback from production environments. Centralized logging service is not in use.

Docker is not used in production but some components has implemented dockerfile for development purposes. Containers are not built in CI pipeline and the team does not have a container registry in use.

4.7 Earlier DevOps Assessment

At the begin of year 2019 an external company did a DevOps assessment for the team. The scope of the assessment was technical. The assessment revealed performance issues, architecture problems, software requirement and life cycle management challenges, lack of metrics and meters, test coverage and test accessibility problems and challenges in the development process and lack of binary repositories, modern CI build tool and manual test environment setup. Also, multiple challenges at operation phase were mentioned. Positive findings included: motivation to change, already identified problems in testing, and software lifecycle management and the autonomy of the team.

Improvement suggestions for the team were to start develop automated test environment and test cases based on ATDD methodology. After the assessment, the team has started to apply test automation and changed development process and adopted new tools.

5 Benchmarking the Team

To find answers to research questions, a survey for the team members was chosen to be the main information acquisition method. For analyzing release frequency, development time and issue closing rate, existing data were used. Questions for the survey were chosen based on State of DevOps survey findings by Forsgren, Humble and Kim [10]. Agile and Scrum related questions are based on principles included to the Agile manifesto [3] and the Scrum guide [4]. Addition to substance questions, survey participant's role, team membership age and working location were asked. The survey questions are concentrating on 24 DevOps capabilities identified on state of DevOps surveys.

5.1 Existing Measurements and Statistics

Release frequency is used to measure team's delivery performance. As the team's product is deployed by an internal specialist on site and is controlled by agreements with clients, software release frequency is better metric than actual deployment frequency. The release frequency of the team was found from the team's project management system.

Year	Release count	Release interval (months)
2016	3	4,00
2017	10	1,20
2018	8	1,50
2019	7	1,71

Table 1 Release frequency

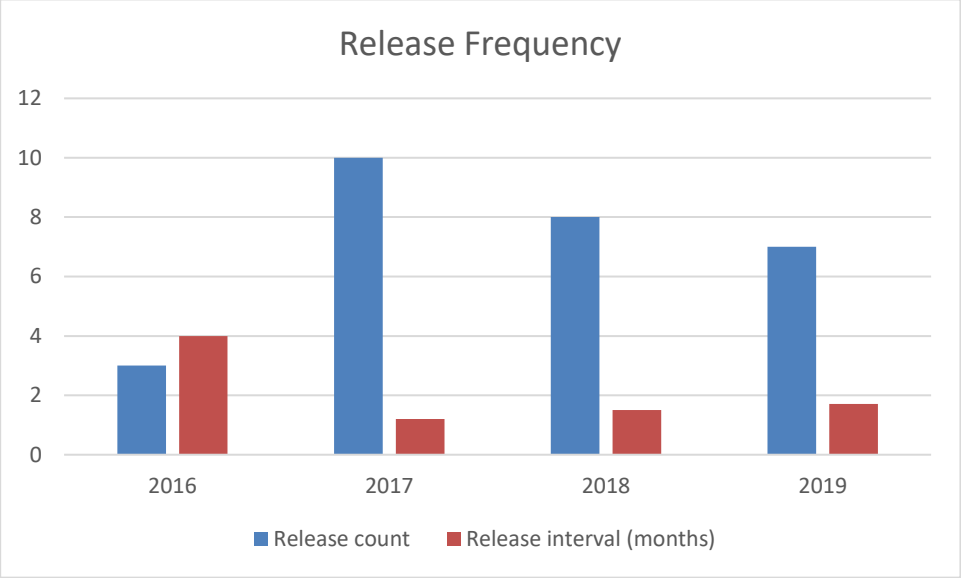

Figure 20 Release frequency

Table 1 and Figure 20 illustrates the delivery frequency statistics of the team. The team has had software releases ranging from 3 releases per year to 10 releases per year. Releases have not been equal sizes. Some of the releases have contained multiple new features and improvements and others only small fixes to existing releases. The release interval of the team had been from one month to three months. When analyzing the past release frequency, one must take account the release content fluctuation. Recent changes in development process do not seem to have had a noticeable impact on delivery frequency statistics. Year 2019 data is not complete and may increase.

On the State of DevOps surveys high performers were having deployments on demand, medium performers once per week to once per month and low performers once per month to once per six months [10]. According to the numbers, the team's delivery performance falls into the low performer category. The deployment environment of the team's product may not be favorable for fast deployments.

Figure 21 Describes closed features and issues for each sprint. Feature closing rate varies from 0 features to 24. Average is 8,7 features per sprint. Not all features are equal sized so measuring the closing rate does not give an accurate view of the team's performance. The statistics show that closing rate fluctuates a lot. Issue closing rate also fluctuates. Release dates can be seen since issue closing amounts at higher issue closing rate. Higher issue closing rate near release can be a signal from late testing process and long feedback loops. Issue closing rate has been higher level

at most recent sprints, which may tell improved testing capacity. However, data do not give very reliable evidence for conclusions.

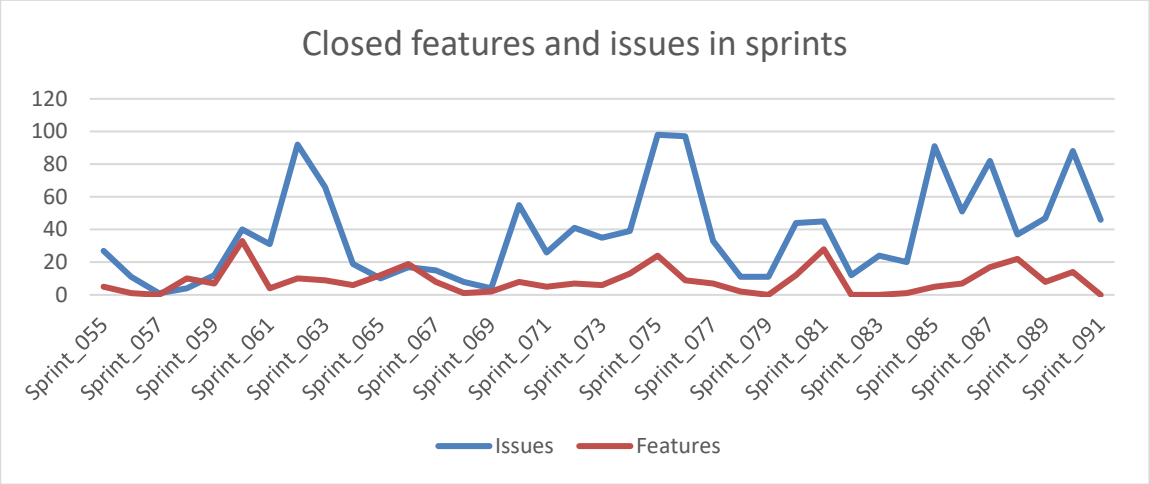

Figure 21 Closed features and issues

5.2 Information Gathered From Tools

Branch lifetime and ticket life cycle metrics do not exist. For this research, only superficial analysis for git repositories, Jenkins pipelines and project management system were executed. By observing git history and Jenkins pipelines, branch lifetime varies from less than one day to months. Issue branches are integrated faster than feature branches.

The feature tickets of the team have different states: “new”, “draft”, “approved”, “in progress”, “ready for merge”, “ready for integration test” and “integration tested”. Development time was measured as the time spent from “in progress” state to “ready for merge” state. Development time should match branch lifetime in the team’s workflow. Unfortunately, ticket status flags are not always used, which prevent analyzing ticket life cycle. Some feature tickets were also reused, which made life cycle analysis more difficult. Due errors in ticket usage, ticket life cycle analysis was not done.

5.3 The State of DevOps Survey for the Team

To get information from the team, semi-structured questionnaire was used. Whole questionnaire can be found from Appendix A. Questions are based on 24 DevOps capabilities identified by Forsgren, Humble and Kim [10]. Also, agile development capability questions were included to find out how well the agile software development practices are adopted in the team. The inquiry also contains additional background information, questions to group the participants based on their role and work age in the team. Some of the questions give also insights to many question categories.

5.3.1 Questions to Measure Continuous Delivery Capabilities

1. All project artifacts are stored in version control
2. Our development process allows continuous integration
3. Changes are merged to master daily
4. I can get test automation results easily after each code change
5. Test automation gates code change requests
6. Our test automation set is comprehensive
7. I can run automated test set easily on demand
8. Our test data is well managed
9. Our test automation results are reliable
10. I can setup test environment myself with certain software installed
11. Our master branch is most of the time in releasable state
12. My tools are adequate to perform my tasks
13. Used technologies are appropriate and interesting

14. IT infrastructure helps me to achieve highest possible outcome

15. Security is built in the development process

Continuous delivery capability questions are concentrating on issues that effects on short feedback loops and effective delivery. Questions should reveal if the testing and development are working seamlessly together. The questions should also give clues about level of automation. Test related questions (4-10) are concerned only test automation as it is found to affect the delivery performance [10, p. 48].

Questions from 12 to 14 are based on finding that the right tools and resources can contribute to job satisfaction, which contributes to organizational performance [10, p. 108-109]. Also, project member has implicated dissatisfaction to the IT infrastructure and performance of the organization's IT support.

Participants were also given the option to answer freely about continuous delivery. Open answer field was added to get insight about how the team members think about continuous delivery.

5.3.2 Survey Questions to Measure Architecture Capabilities

1. Our architecture is loosely coupled

2. Our architecture supports effective continuous integration and continuous delivery

3. We can deploy the application independently of other applications and services

4. The team can do decisions related to architecture

5. We can do testing for components without integrated environment

Architecture is found to be a significant barrier to efficient delivery [10, p. 59-68]. Shahin et al. [29] identified that small independent deployment units helps building the CD pipeline. The study also suggests that architectural decisions affects in deployability, modifiability, testability, monitorability, loggability and resilience. The State of DevOps survey shows that, if a team has the

power to make architectural decisions themselves, it has better deployment performance. Independently testable software components help test automation and allows faster feedback loops [10, p. 204]. Also, open question about architecture capabilities was added.

5.3.3 Survey Questions to Measure Product and Process Capabilities

1. I can get customer feedback easily
2. My work tasks are small
3. My work tasks are well defined
4. I know our definition of done
5. Our development process is efficient
6. Our development process is well defined
7. I know and understand our development process

Customer feedback is essential for continuous improving. Customer collaboration is also one of the main aspects of agile software development. Fast and reliable customer feedback is linked to organizational performance. Small batch size correlates with software delivery performance [10, p. 84]. Lean also emphasize small batch size.

Questions depends on participant's own reckoning. The team does not have any meters how to measure such things. Definition of done and a well-defined process should go hand in hand. Question 7 is added to find out, if the process is communicated to the team. Open question about product and process capabilities was also included.

5.3.4 Survey Questions to Measure Lean Management and Monitoring Capabilities

1. Changes approval process is efficient
2. When change is made, I can get reliable metrics easily and in short time

3. Work in progress limits are applied and followed
4. The work is made visible with information radiators. For example with task boards etc.
5. Metrics are visible and actionable

On product and process capability questions were also related to lean management. Batch size and efficient, continuously improved processes are important in Lean thinking. State of DevOps research revealed that change approval boards reduce software deployment performance [10, p. 205]. Fast feedback loops are at the core of DevOps. Metrics are one way to provide feedback. WIP limits are a way to ensure that the development process is efficient and batch size are small enough. Many different ongoing tasks can also tell, if the worker is shielded from interruptions and unplanned work. Metrics question was also added a visibility dimension. Without good information radiators are in danger to be ignored. Question 4 and 5 should go hand in hand. Open question about Lean management and monitoring capabilities was also included.

5.3.5 Survey Questions to Measure Cultural Capabilities

1. Team search new information actively
2. Messenger are not punished when they deliver news of failures or other bad news
3. Responsibilities are shared
4. Cross-functional collaboration is encouraged and rewarded
5. Failures causes inquiry to reveal root cause
6. New ideas are welcome
7. Failures are treated primarily as opportunities to improve the system
8. Team communicates inspiringly
9. Team has inspiring vision
10. Team stimulates me intellectually

11. Team has supportive working culture

12. Personal recognition is given

Organizational culture is found to affect software delivery performance and job satisfaction [10, p. 218]. Westrum organizational typology questions, questions from 1 to 7, are same as used in state of DevOps survey and the construct was proven to be valid and reliable [10, p. 32-35]. Question word choices were slightly modified from State of DevOps survey. Transformational leadership questions, questions from 8 to 12, measure indirectly the five dimensions of transformational leadership. In the survey form, Westrum's organizational typology questions were in one question block and transformational leadership questions in another block.

State of DevOps research identified that transformational leadership correlates with Westrum's organizational culture typology and employee net promoter score (eNPS) [10, p. 115-121]. Instead of asking directly about the characteristics of the team leaders, indirect questions were chosen to be used. Leaders are key factors to form organizational culture [19]. There are also many other factors that may affect the culture. The questions are based on the five dimensions of transformational leadership [21].

Employee net Promoter Score

1. How likely are you to recommend working in the team to a friend or colleague?

eNPS is found to correlate with business performance and it also indicates employee's loyalty, which is found to correlate with company performance [10]. In the state of DevOps survey, eNPS had two dimensions, organizational and team. As the team has many subcontractors and the scope of this thesis is the team, the question was about organization.

5.3.6 Survey Questions to Measure Agile Development Capabilities

1. I know and understand Scrum framework

2. Our team follows Scrum framework

3. Scrum is applied effectively in our team

4. Scrum ceremonies are strictly time boxed
5. There are no additional meetings to Scrum meetings
6. I know agile principles
7. I follow agile principles in my work
8. Our team is self-managing team

Because, the team follows the Scrum framework, agile development questions are tied to the Scrum implementation in the team. Scrum is a framework for agile development, which leaves a lot of room for different kind of implementations as long as the core pieces are in place. These core pieces are: time boxed Scrum ceremonies, roles (Scrum Master, Product Owner and self-managing team), continuous improvement and agile principles [4]. Questions are concentrating on the efficacy of the team's Scrum implementation. Also, open question about agile development was included in the survey. Customer feedback and batch size questions on product and process capability question set should indicate Scrum implementation efficacy.

5.4 Survey Results

The survey was open to all team members 12 days. Ten members of the team took the survey. Both offices were present. All roles got at least one answer. In this section, survey answers are analyzed only by reflecting to background studies without combining current practice observations from the previous chapter. Current practice observations are only used to verify the conclusions of the survey analysis. Analysis, which combines the survey findings and current practice observations is done in 5.6.

When analyzing the data, "I do not know" answers were excluded from calculations of average, median and standard deviation. Empty answers were counted as "I do not know". Table 2 shows the scoring of the Likert scale answers.

Answer	Points
Totally disagree	1
Somewhat disagree	2
Neither agree nor disagree	3
Somewhat agree	4
Totally agree	5

Table 2 Survery answer scoring

5.4.1 Continuous Delivery Capability Answers

Question	Average	Median	Std.dev	Mode	Min	Max	Idk c.	Idk %
Q1	2,67	2	1,12	2	1	4	1	10
Q2	2,89	3	1,27	3;4	1	5	1	10
Q3	3,11	4	1,36	4	1	4	1	10
Q4	3	3	1,41	2;4	1	5	3	30
Q5	2,4	2	1,14	Idk	1	4	5	50
Q6	2,13	2	0,99	2	1	4	2	20
Q7	3,83	4	0,98	4	2	5	4	40
Q8	2,25	2	0,5	2	2	3	6	60
Q9	4	4	0,58	4	3	5	3	30
Q10	2,17	2,5	0,98	3	1	3	4	40
Q11	3,5	4	1,35	4	1	5	0	0
Q12	3,75	4	0,89	4	2	5	2	20
Q13	3	3	0,87	2	2,3,4	4	1	10
Q14	2,3	2	1,16	2	1	4	0	10
Q15	2,5	2	1,28	2	1	4	2	10

Table 3 Continuous delivery capability answer statistics

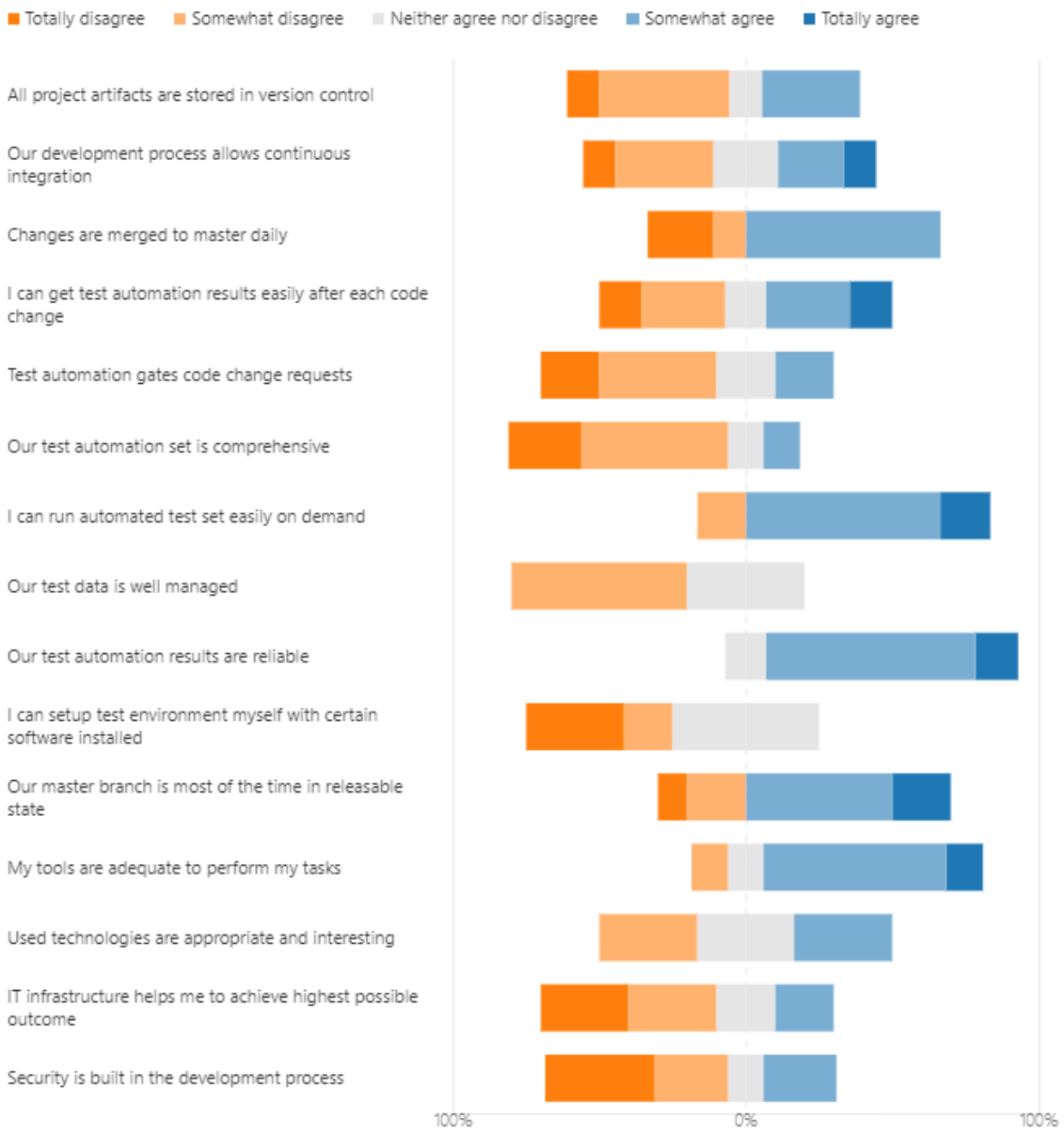

Figure 22 Continuous Delivery capability answers without "I don't know" answers

Free Answers About Continuous Delivery

"Maybe answering “I do not know” as UX designer tells something about how widely CD is utilized within the team."

"Our processes and process discipline needs to be updated/applied before we can do real CD."

"Our software is installed locally in customer premises and networks. Customers want to decide when systems are updated. Continuous delivery in this environment is theoretical to some extent."

"Good improvements lately, so let's continue on the current path"

Continuous Delivery Capability Analysis

All questions got both disagree and agree answers. Top 5 least amount of points are listed on Table 10 and top 5 highest points are listed on Table 11

Question	Average	Std.dev
Our test automation set is comprehensive	2,13	0,99
I can setup test environment myself with certain software installed	2,17	0,98
Our test data is well managed	2,25	0,5
IT infrastructure helps me to achieve highest possible outcome	2,3	0,87
Test automation gates code change requests	2,4	1,11

Table 4 Top 5 lowest points from continuous delivery capability questions

Question	Average	Std.dev
Our test automation results are reliable	4	0,58
I can run automated test set easily on demand	3,83	0,95
My tools are adequate to perform my tasks	3,75	0,89
Our master branch is most of the time in releasable state	3,5	1,35
Changes are merged to master daily	3,11	1,36
I can get test automation results easily after each code change	3	1,41

Table 5 Top 5 highest points from continuous delivery capability questions

All testing related questions got many “I don’t know” answers, which may imply that the new automated testing facility has not been communicated well enough to the team. However, the team seems to trust the results of the test automation. Answers have also shown that in general test automation related questions got low scores except questions about the test result reliability and ability to run test automation on demand.

In general, continuous delivery capability questions answers are deviated. There seems to be possible to improve on a wide area. Also, free answers indicate that team can do better in continuous delivery. It was also pointed out that continuous deployment is not something that the team should aim at the moment instead of continuous delivery.

“Changes are merged to master daily” got “Totally disagree” answers, but also multiple “Somewhat agree” answers. Observations from git usage and ticket systems in 5.2 verifies the answers. The team has many short lived branches, but also many very long lifetime branches. The VCS branching model guides to complete a feature before merging to master, which can lead longer integration time when the developed feature is big.

As expected, the team was not very satisfied to IT support. The test network availability problems, observed in 4.4, also supports the answers.

5.4.2 Architecture Capability Answers

Ques-	Aver-	Median	Std.dev	Mode	Min	Max	Idk c.	Idk %
Q1	2,56	2	1,13	2	1	4	1	10
Q2	2,44	2	0,88	2	1	4	1	10
Q3	2,8	3	1,23	4	1	4	0	0
Q4	3,67	4	1,32	5	1	5	1	10
Q5	3,22	4	1,3	4	1	5	1	10

Table 6 Architecture capability answer statistics

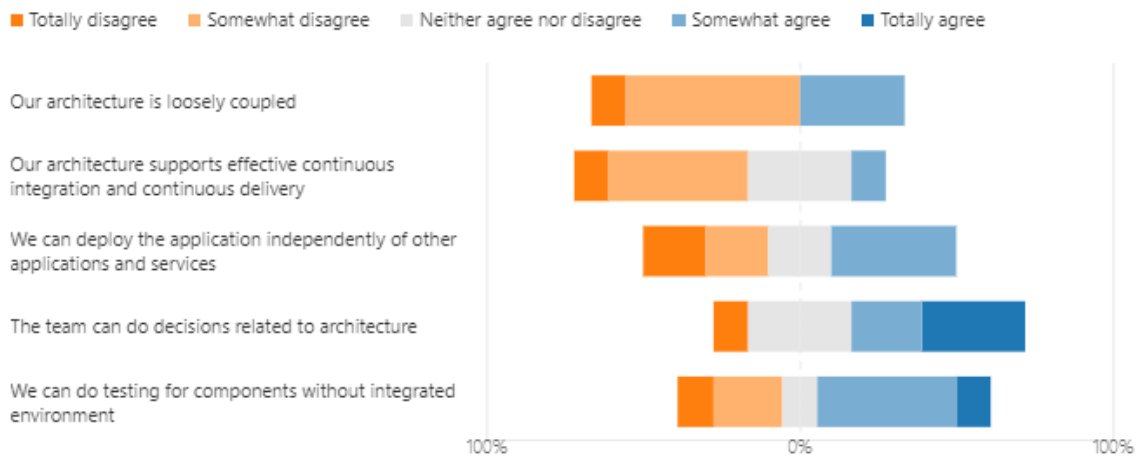

Figure 23 Architecture capability answers without "I don't know" answers

Free Answers About Architecture

“Current architecture has way too much legacy stuff - it should be blown up and redone.”

“Even designers hit the “SW architecture wall” occasionally when proposing designs, so maybe not in the best shape?”

“Old, but ok'ish”

“Monolith.”

Architecture Capability Analysis

Architecture capability question answers were quite deviated. Most survey participants indicated that the architecture is not loosely coupled but still “somewhat agree” option got a significant amount of answers. Participants answered most negatively to questions about how the architecture supports CI and CD. Deployment freedom got most neutral answers, but still some totally disagree answers also. Freedom to make architecture decisions was the most agreed answer. Testing without integrated environment question got strong opinions for and against. Interpretation of “integrated environment” may play a role.

The free answers reveal that the architecture is quite old and have challenges that hinders the work. Studies show that architecture decisions can ease continuous delivery journey [25]. However, based on the answers, there may not be urgent need to replace whole architecture at once. The strangle pattern approach could be used to redo the architecture gradually towards more independent and dependency free components. In strangler pattern method, architecture is changed piece by piece towards target architecture.

5.4.3 Product and Process Capability Answers

Ques-	Aver-	Median	Std.dev	Mode	Min	Max	Idk c.	Idk %
Q1	2,1	2	1,1	1	1	4	0	0
Q2	2,38	2,5	1,3	1	1	4	2	20
Q3	3	3	1,05	4	1	4	0	0
Q4	3,4	4	1,17	4	1	5	0	0
Q5	2,8	3	0,92	3	1	4	0	0
Q6	3,6	4	0,84	4	2	5	0	0
Q7	4,2	4	0,79	4	3	5	0	0

Table 7 Product and process capability answer statistics

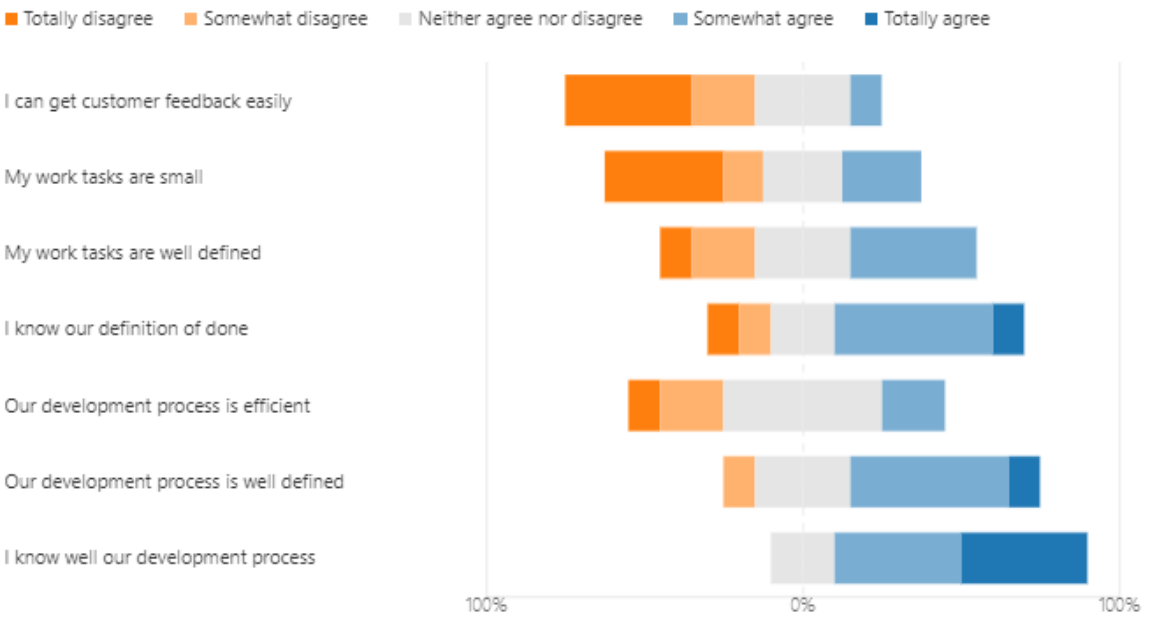

Figure 24 Product and process capability answers without "I don't know" answers

Free Answers About Product and Process

“Processes keep updating, of course development should happen. Not sure if everyone applies the processes though.”

“Slipping from agreed process back to old habits tend to happen too easily.”

"Too much panic-driven instead of long-term-planned. On sprint-level things usually work quite well"

“Sold as waterfall (fixed content and schedule), tried to be implement in Scrum butt agile way.”

Product and Process Capability Analysis

The product and process capability question answers imply that the development process is defined and team members know it well. Based on the answers, the team has problems to follow its development process. Also, the development process doesn’t seem to include customer voice, which is emphasized by agile software development principles [24]. Work task size is also a problem on the team. The process efficiency is seen quite neutral, but a little bit more inefficient than efficient. Some survey participants also would like to have better defined work tasks, but some work tasks seem to be ok. Role in the team may explain the differences between the answers. As expected, definition of done and development process question answers are similar.

Lack of customer voice indicates that the Ops part of the DevOps cycle does not a have working feedback loop. The team should increase communication with people who work with customer interface.

In 4.1 missing retrospective meetings were observed. However, the open answers indicate that process is still updated. Observations and Lean management and monitoring question answers also tell that monitoring and metrics are not in use. The team does not seem to get feedback from process changes.

5.4.4 Lean Management and Monitoring Capability Answers

Ques-	Aver-	Median	Std.dev	Mode	Min	Max	Idk c.	Idk %
Q1	2,6	2,5	1,17	2	1	4	0	0
Q2	2	2	1	2	1	4	1	10
Q3	2,67	3	1,22	4	1	4	1	10
Q4	2,9	2,5	1,28	2	1	5	0	0
Q5	2,4	2,5	0,97	3	1	4	0	0

Table 8 Lean management and monitoring capability answer statistics

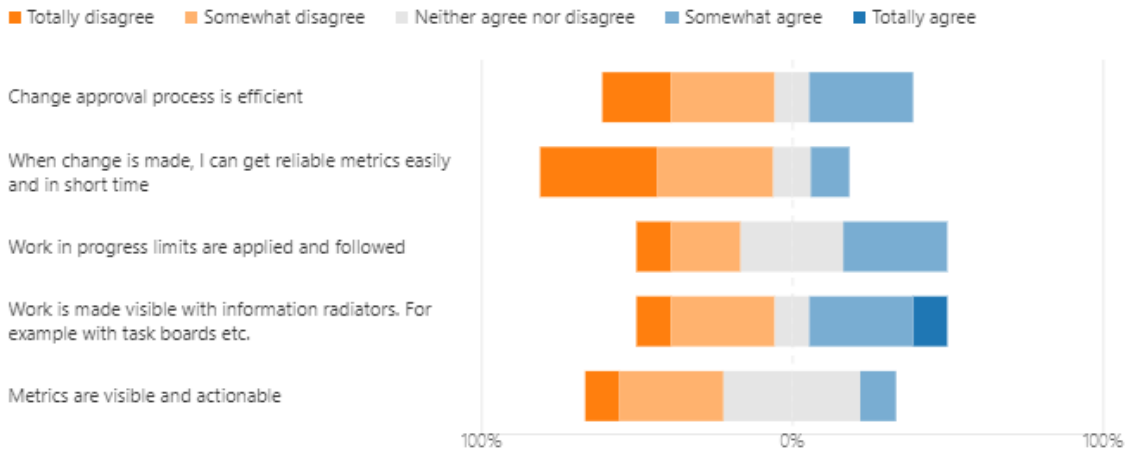

Figure 25 Lean management capability answers without "I don't know" answers

Free Answers About Lean Management and Monitoring

“Polarion is not very easy tool to quickly check status. Prioritizing tasks cumbersome, no kanban style boards in use. Although cannot say how those would fit the team structure.”

Lean Management and Monitoring Capability Analysis

The Lean management and monitoring capability question answers got generally lower than neutral scores. The team can’t get feedback easily when changes have been made. On the other hand, use of information radiator question got very contradictory answers. The answers imply that there might be information available on some tools, but it is not easily accessible by all team members. “Metrics are visible” question recorded mainly disagreeing and neutral answers, which also imply that improvements to metric visibility could be done. The observations are in-line with

the product and the process management questions finding that customer feedback is not easily available.

The product and process capability questions also got lean related questions like task size and process efficiency. Based on the survey answers and missing Scrum retrospective meetings, which were observed in 0, the team does not have Lean management practices in use. The WIP limit question got similar results as the batch size question on the product and process capability question set. Large batch size can make following WIP limits difficult.

5.4.5 Cultural Capability Answers

Ques-	Aver-	Median	Std.dev	Mode	Min	Max	Idk c.	Idk %
Q1	3,6	4	0,7	4	2	4	0	0
Q2	4,5	5	0,71	5	3	5	0	0
Q3	4,3	4,5	0,95	5	2	5	0	0
Q4	3,6	4	1,26	4	1	5	0	0
Q5	3,9	4	1,20	4	1	5	0	0
Q6	4,5	5	0,71	5	3	5	0	0
Q7	4,1	4	0,86	5	3	5	0	0
Q8	3,7	4	0,95	4	2	5	0	0
Q9	3,33	3	0,71	3;4	2	4	1	10
Q10	3,9	4	0,88	4	2	5	0	0
Q11	4,1	4	0,88	4	2	5	0	0
Q12	3,8	4	1,14	4	1	5	0	0
eNPS	7,1	7,5	1,85	9	4	9	0	0

Table 9 Cultural capability answer statistics

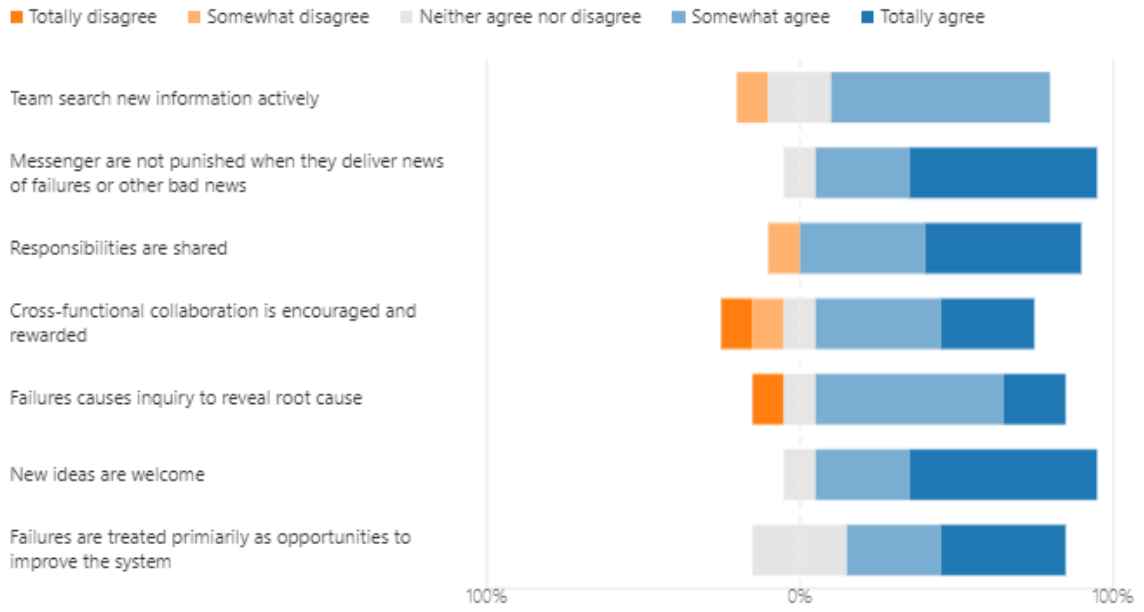

Figure 26 Westrum's organizational typology questions answers without "I don't know" answers

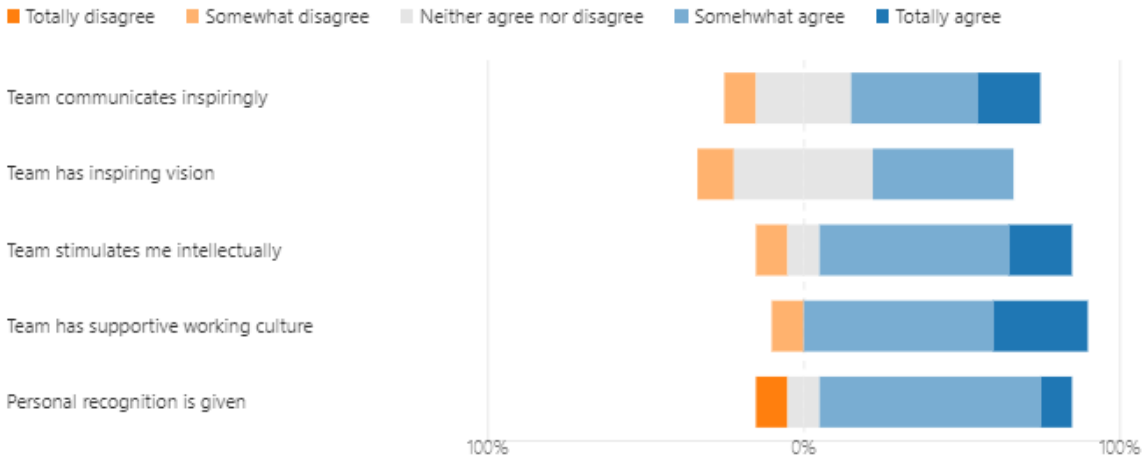

Figure 27 Transformational leadership question answers without "I don't know" answers

Employee net Promoter Score

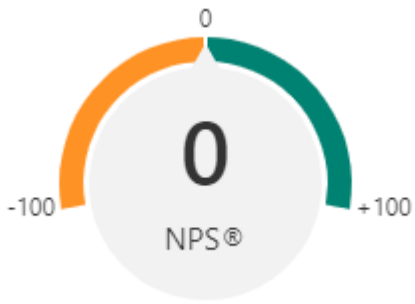

Figure 28 Employee net promoter score

Promoters	**3**
Passives	4
Detractors	3

Table 10 Employee net promoter score

Cultural Capability Analysis

Questions to measure Westrum's organizational culture got high points overall with only a few disagree and neutral answers. Based on the answers, the team seeks actively new information and new ideas are welcome, the team has fear free atmosphere where also negative things can be said. Problems are seen as an opportunity to learn and team shares responsibilities. On the Westrum's scale, the team has a generative working culture.

Transformational leadership questions also got high points. The questions were indirect, but answers show that the team lead has been able to create an atmosphere, which express all the five dimensions of transformational leadership.

eNPS score was 0. Equally many of the survey participants are promoters (score over 8) as detractors (score 6 or under). The average answer was 7,1 which is relatively high when taking account that the answers contained 3 detractor answers. The most common answer was 9 with 3 answers. The team got high points on Westrum's organizational culture questions and transformational questions, but the eNPS question shows that not all team members are satisfied. State

of DevOps studies shows that continuous delivery practices and Lean management practices improves job satisfaction [10]. Based on this survey both areas, continuous delivery and Lean practices, can be improved on the team.

5.4.6 Agile Software Development Capability Answers

Ques-	Aver-	Median	Std.dev	Mode	Min	Max	Idk c.	Idk %
Q1	4,3	5	1,25	5	1	5	0	0
Q2	3,4	4	1,07	4	1	4	0	0
Q3	2,7	2,5	1,06	2	1	4	0	0
Q4	2,8	3	0,79	2;3	2	4	0	0
Q5	2,4	2	1,07	2	1	4	0	0
Q6	4,1	4	0,74	4	3	5	0	0
Q7	3,6	4	0,52	4	3	4	0	0
Q8	3,5	4	1,18	4	1	5	0	0

Table 11 Agile software development capability answer statistics

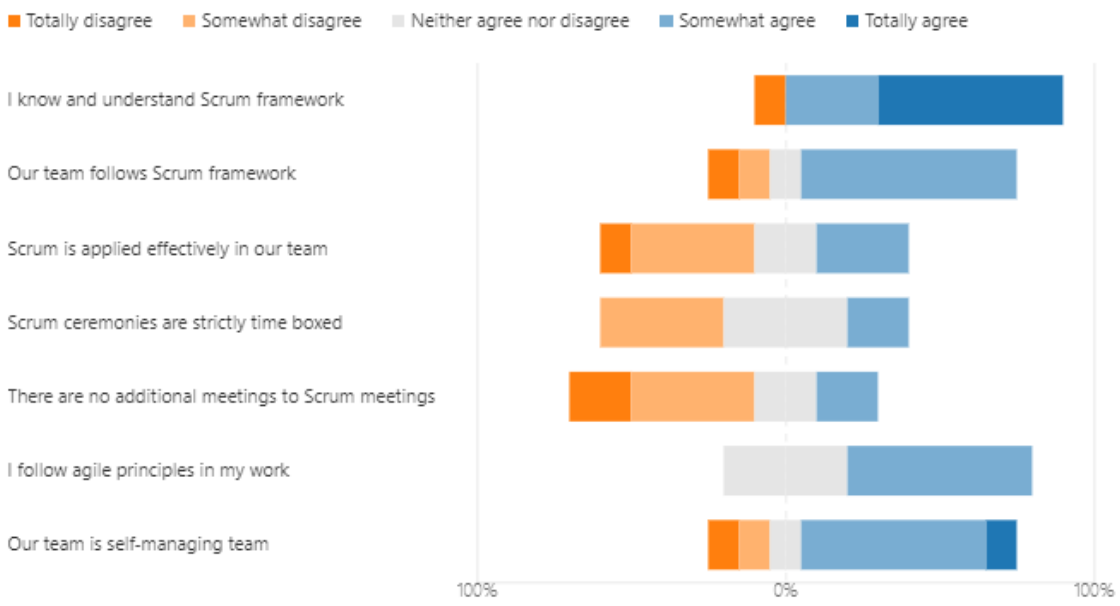

Figure 29 Agile development capability answers without "I don't know" answers

Free Answers About Agile Software Development

“Sold in waterfall, tried to be implemented in Scrum butt. Team has no real power in prioritizing backlog.”

“Due to non-Scrum customers, full Scrum not applicable. Priorities not clearly visible, and may not always even exist. Team a bit in survival mode with work amount and work content.”

Agile Development Capability Analysis

The agile development capability questions answers show that team members know the Scrum framework very well, but has failed to apply it effectively. However, most of the survey participants say that they do follow agile principles in their work. Scrum guide gives quite free hands to teams to implement the Scrum framework. Product and process capability questions and Lean management questions indicated that the team has problems with software requirement management, which can also effect on the agile development capabilities. Requirement management was also one of the findings of the previous DevOps assessment, which was described in 4.7.

As noted in 4.1, the team hasn’t had retrospective meetings for a while, which may have created conditions where the process has slipped towards inefficient and undisciplined implementation. As the team claims to be self-managing, it knows the Scrum framework and the team follows agile principles, it should have keys to improve its’ practices without external help.

The Scrum guide [4] defines the Product Owner as customer voice and the owner of the backlog. Poor results on customer feedback question and small and a well-defined work task questions on product and process capability questions imply that team’s Product Owners does not have a similar role in the team as the Scrum guide describes. The open answer also suggests that the Product Owner does not have the power to manage the backlog as the Scrum guide suggests.

The Scrum guide advices to avoid additional meetings to Scrum ceremonies [4]. Based on the survey, the team organizes many meetings, which may indicate problems in task management and role descriptions.

5.5 The DevOps Maturity Level of the Team

The Mohammed's maturity model gives defined levels for different DevOps dimensions, but as the scope of the thesis is team level it can't be applied completely. However, the model is used as a basis for this evaluation. Sharing information in organizational level is done by a subjective evaluation of a team member's point of view. The team does not have documentation available to support claims. On managerial level, more coordination may happen than assumed in this evaluation, but it is not clearly visible in team level.

Quality

Quality is on the managed maturity level on the Mohammed's DevOps maturity model scale. The team has somewhat defined quality criteria, but definition and acceptance criteria are partly relying on the test lead's and test engineer's feeling of high enough quality level. Quality standards are not shared between other teams. Integrated system testing is difficult due lack of coordination between teams. Manual test cases are defined and described on a project management tool. Automated tests are defined on test automation scripts and project management tool contains links to test automation scripts. Each test set run, leaves the results to project management tool.

Automation

Automation is on the managed maturity level on the Mohammed's DevOps maturity model scale. The team has started to develop test automation and CI pipeline. The team doesn't have other automation practices than the test automation. Automation does not have organization level guidance. Automated metrics are not in use. Also, the automation level is not measured.

Communication and Collaboration

Communication is on the managed maturity level on the Mohammed's DevOps maturity model scale. Communication happens with methods that the team finds useful. The team uses MS Teams as a communication platform with MS SharePoint for file sharing. Also, e-mail is used. Communication does not move vertically between teams, but follows hierarchical model. The team does not have guidance for communication.

Governance

Communication is on the managed maturity level on the Mohammed's DevOps maturity model scale. The team has documented roles and description which tasks the role has in each development phase. The development process is also documented. However, the development process is not followed as it is documented. Also, the documentation for tasks in each process step is not followed. Organization level governance is not clearly visible on a team level.

5.6 Team's DevOps Capability

The team has started the journey on DevOps path, but is still concentrating on improving only on a team level. Good development has been made by involving testing earlier in the process, by creating test automation and by revising the development process. The working culture of the team is one of its key assets. Most challenges are found from hearing the customer feedback, development process and measuring the development and production.

Team has three obvious challenges:

1. Lack of feedback loops
2. Task management
3. Lack of continuous improvement

Each issue is linked to each other's. Lack of feedback loops can be seen in difficulties to get customer feedback and results from changes being made. Lack of feedback loops also includes missing metrics from development and actual production environments. The team does not have any visible metrics; customer voice is only heard through intermediaries. Without customer's voice, task management becomes difficult.

The task management problems are seen as large batch size, unclear feature definitions, wasteful meetings and difficulties to obey the development process. Missed retrospective meetings are causing a lack of continuous improvements. The team does improve its development methods, but without systematical approach and without help of feedback loops.

The feature development process is documented, but the team does not always follow the process. As retrospective meetings have not been done, the reason that the process is not followed remains unknown. Based on the Scrum guide [4], Scrum Master would be responsible that the defined process is followed, but team's Scrum Masters got also many of other roles.

The team is kind of following the Scrum framework, but some key elements of the Scrum are missing. The team has not held retrospective meetings, but the development process is still evolving. However, the evolution has happened without systematic approach and no metrics has been used. The Scrum guide [4] obligates to include one major process development task for each sprint. The team organizes many meetings addition to Scrum ceremonies. The Scrum guide suggests that no other meetings are necessary. The need for additional meetings might tell about challenges in information sharing and role descriptions. Lack of information radiators and problems in task management also gives similar signals.

The survey shows that the team has a generative working culture and the potential to carry out changes. Collaboration between different teams was not measured in this research. Despite, the working culture results, team members see difficulties in task management. Lean development practices are not in use, which is seen in large batch sizes. Organizational level working culture was not in the scope of this thesis.

The team does not have a designated software architect, which would be an obvious role to take responsibility for refining feature descriptions with Product Owners. The architect would have more technical knowledge, which would help chopping development tasks into smaller logical pieces before implementation. The team also uses branch driven development workflow, which encourages completing the new functionality instead of implementing it piece by piece.

The software architecture of the team's product and used technologies are quite old. The team sees need to replace the architecture with more modern one. However, current architecture and technologies does not prevent DevOps advancements, but does make it more challenging than it could be. Better architecture also could help the development process. The team has a lack of visibility to production environment and deployment process. The software does not have modern logging capabilities and configuration management.

The deployment environment is challenging for continuous deployment, but there are no obstacles to aim for continuous delivery. The team provides installers for deployment process and uses

same installers when creating staging environments. However, the team has built automation only to support staging environments, but not to support deployment process.

The testing process of the team has shifted left recently, which helps to create more meaningful test cases. Test automation infrastructure has been built to test different software functionalities. Test automation nor manual test coverage is not measured. Performance testing is limited. Feature requirements do not always have clear criteria to support testing.

6 Recommendations

In the previous chapter the current state of the team's DevOps transition was described. Multiple issues and challenges were identified. If team started to work on all the issues same time, the capacity of the team would become obstructed. Therefore, in this chapter, only a few change recommendations are given. The recommendations are chosen to be a high leverage change, which should help in multiple pain points.

To support continuous improvement and task management, team should resurrect retrospective meetings. To support retrospective meetings, team should start to develop automatically updating metrics, which shows delivery times calculated from completed and ongoing feature and issue development. Metrics itself do not make the change, but combined with information radiators the metrics improves the delivery performance [10]. Metrics also provides a backbone for retrospective meetings. When metrics are automatically generated from tickets, it creates more concrete purpose to follow the development process. When making changes in the development process, clear vision and metrics to evaluate the transformation process should be available. Changes without target and visible metrics rarely succeeds.

For example, development time and deployment time generated from project management tools could tell about development process performance. Open issue count and not released feature count would tell about the software release status. Also, code metrics could be made visible.

The team should focus more on the production phase of the process to gain more knowledge from end users. To shorten long feedback times from production, logging should be improved with automatic monitoring capabilities. When the team starts to develop new software architecture, monitorability and loggability should be taken account. To improve team's knowledge about production environment and end user needs, feature requirements should be written from the end user perspective. The team's development process documentation already suggests that, but it is not followed. Deployment process should be made as visible as possible to the team.

The team has done good development on test automation infrastructure, but the test automation is still not completely integrated into the development process. Accessibility to the test environments should be made easier and scaled up for the whole team. Also, test result visibility should be improved so that each team member can get an idea of the current status of test automation results.

7 Limitations

The scope of the research was on a team level, but the term DevOps also include whole organization. In this study organization was only viewed from team member point of view. The organization always has an effect on how one team can do their work. Further research could be done on organization wide. Sharing the practices is one key pillar of DevOps. Sharing dimension was almost completely neglected from this study.

The survey questions were chosen based on source materials. Questions were targeted to cover the pain points, which were guessed by observing the team and reflection it to source materials. The questions can be used as a basis when measuring other software development teams, but researchers may want to tailor them to be more applicable to their research questions.

Used sources give a comprehensive overview to DevOps. The sources vary from peer reviewed articles and doctoral thesis to DevOps practitioner's blog posts and material produced by enterprises who operates in the field of DevOps. DevOps is still mainly the result of empirical methods and collection of practices that has been found useful. When choosing non-academic sources, the author's reputation in DevOps scene was an important factor. Many academic papers, were also citing same sources as this thesis.

The source material did not include research material to prove effectiveness of Scrum and agile development methods. Therefore, analysis and recommendations based on Scrum and agile development methods are relying on the reliability of used sources. There exist evidences of effectiveness of these methods, but those were not included in this thesis.

The team had not produced much quantitative data and adding tools to mine data from used tools was out of scope of this study. Because data sources were limited, current practices are based on researcher's own observations and team's documentation. When relying on observational data, there are always errors and inaccuracies. When the team has implemented more metrics, current practices can be analyzed more reliably.

8 Conclusion

The thesis used qualitative research methods and semi-structured survey to measure the level of DevOps practices in a software development team. The research studied different aspects of DevOps and agile software development. The study uses annual State of DevOps surveys from year 2014 to 2017 as a base and tailored a custom state of DevOps survey for the team members. The survey results and observations of the team's work practices gave an insight of the teams DevOps capabilities.

Most important findings were that team has a generative organization culture and it has made a progress on implementing DevOps practices on the development side. However, team's challenges on its DevOps journey are related to gathering feedback from production and integrating practices with other shareholders in the process. Team has challenges with feedback loops, task management and continuous improvement.

Next steps on the team's DevOps transitions are enabling faster feedback loops from production to development, enabling continuous improvement and continue improving the test automation and scale it for whole team.

The thesis process was completed while working in the team. The aim was to produce valuable information to the organization, enhance researcher's own understanding of the topic and link the studies with work assignments. The research served all three purposes. Some of the discussed findings changed already during the process. For example, end-user's view has been emphasized and means to get feedback from production to development has been improved. Also more changes have been discussed. The thesis has provided a broader view of the topic into the team, which was one desired outcome.

During the process, the researcher's own idea about the topic has become clearer. The process brought more in-depth view into daily working and inspired to think more scientific way the everyday issues. DevOps is a broad concept and the thesis tried to cover all aspects of it. Inevitably, the perspective has become wider and it has become more natural to discuss also the less technical dimensions of to the topic. During the process it became more evident that change requires changes in whole organization, but still one team can start the changes in wider. The technical solutions require to be accompanied by supportive working culture and efficient leadership.

Sources

1 Aspers P, Corte U. What is Qualitative in Qualitative Research. Qualitative sociology 2019;42(2):139-160.

2 Dybå T, Dingsœyr T, Moe NB. Agile Project Management. In: Ruhe G, Wohlin C, editors. Software Project Management in a Changing World: Springer-Verlag; 2014. Available at: https://www.researchgate.net/publication/263276642_Agile_Project_Management. Accessed: 26.12.2019

3 Agile Manifesto. 2001. Available at: https://agilemanifesto.org/. Accessed 25.12.2019

4 The Scrum guide. 2017. Available at: https://www.scrumguides.org/scrum-guide.html. Accessed 20.8., 2019.

5 2019 Global Developer Report DevSecOps. GitLab Inc. Available at: https://about.gitlab.com/developer-survey/. Accessed: 12.9.2019

6 Eloranta V, Koskimies K, Mikkonen T. Exploring ScrumBut—An empirical study of Scrum anti-patterns. Information and Software Technology 2016 Jun;74:194-203.

7 Takeuchi H, Nonaka I. The New Product Development Game. Harvard Business Review (USA) 1986;64(1):137.

8 Dingsøyr T, Lindsjørn Y. Team Performance in Agile Development Teams: Findings from 18 Focus Groups. 2013.

9 Iqbal J, Omar M, Yasin A. An Empirical Analysis of the Effect of Agile Teams on Software Productivity. IEEE. 2019.

10 Forsgren N, Humble J, Kim G. Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations. 2018.

11 John Willis. DevOps Culture. 2012. Available at: https://itrevolution.com/devops-culture-part-1/. Accessed: 25.12.2019

12 Lwakatare L, Kuvaja P, Oivo M. An Exploratory Study of DevOps: Extending the Dimensions of DevOps with Practices. 2016.

13 Minick E. Surprise! Broad Agreement on the Definition of DevOps. 2015. Available at: https://devops.com/surprise-broad-agreement-on-the-definition-of-devops/. Accessed: 25.12.2019

14 Kim G. The Three Ways: The Principles Underpinning DevOps. 2012; Available at: http://itrevolution.com/the-three-ways-principles-underpinning-devops. Accessed 17.11. 2019.

15 Kim G, Humble J, Debois P, Willis J. The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations. 2016.

16 Oelich E. DevOps Now With CALMSS. Forrester Research Inc. 2015. Available at: https://go.forrester.com/blogs/15-03-02-devops_now_with_calmss/. Accessed: 25.12.2019

17 Mann A, Stahnke M, Brown A, Kersten N. 2018 State of DevOps report. 2018. Puppet. Available at: https://www.thinkahead.com/wp-content/uploads/2018/10/State-of-DevOps-Report.pdf. Accessed: 25.12.2019

18 Schein EH. Organizational culture and leadership. 3. ed. ed. San Francisco, Calif: Jossey-Bass; 2004.

19 Westrum R. A typology of organisational cultures. Quality and Safety in Health Care 2004 Dec;13(suppl 2):ii22-ii27.

20 Understand team effectiveness. Google. Available at: https://rework.with-google.com/print/guides/5721312655835136/. Accessed 1.12.2019.

21 Rafferty AE, Griffin MA. Dimensions of transformational leadership: Conceptual and empirical extensions. The Leadership Quarterly 2004;15(3):329-354.

22 Sambinelli F, Francisco Borges MA. Lean Thinking in Software Engineering: A Systematic Review. International Journal of Software Engineering & Applications 2017 May 30,;8(3):15-32.

23 Poppendieck M, Poppendieck T. Lean software development. Boston [u.a.]: Addison-Wesley. 2003.

24 Lehtonen T. Metrics and Visualizations for Managing Value Creation in Continuous Software Engineering. Tampere University of Technology, 2017. 109 p. (Tampere University of Technology. Publication). Available at: https://tutcris.tut.fi/portal/en/publications/metrics-and-visualizations-for-managing-value-creation-in-continuous-software-engineering(e9e2fbec-0639-401c-ae6d-2c00d6239b54).html. Accessed: 25.12.2019

25 Shahin M. Architecting for DevOps and Continuous Deployment. The University of Adelaide, Australia. Sep 28, 2015.

26 Shahin M, Babar M, Zhu L. The Intersection of Continuous Deployment and Architecting Process. The University of Adelaide. Sep 8, 2016.

27 Fowler M. Continuous Integration. 2006. Available at: https://martinfowler.com/articles/continuousIntegration.html. Accessed: 25.12.2019.

28 Forsgren N, Kim G, Kesten N, Humble J. 2014 State of DevOps report. 2014. Puppet, IT Revolution Press, ThoughtWorks. Available at: https://services.google.com/fh/files/misc/state-of-devops-2014.pdf. Accessed: 25.12.2019.

29 Shahin M, Zahedi M, Babar MA, Zhu L. An Empirical Study of Architecting for Continuous Delivery and Deployment. 2018 Aug 27.

30 Zolkifli NN, Ngah A, Deraman A. Version Control System: A Review. Procedia Computer Science 2018;135:408-415.

31 Version control tools. Plutora. Available at: https://www.plutora.com/ci-cd-tools/version-control-tools. Accessed 21.11., 2019.

32 Brindescu C, Cdoban M, Shmarkatjuk S, Dig D. How do centralized and distributed version control systems impact software changes? ACM. May 31, 2014.

33 Leite L, Rocha C, Kon F, Milojicic D, Meirelles P. A Survey of DevOps Concepts and Challenges. ACM Computing Surveys (CSUR) 2019 Nov 14,;52(6):1-35.

34 Hammant P. Trunk based development: Introduction. 2017. Available at: https://trunk-baseddevelopment.com/. Accessed: 25.12.2019

35 Fowler M. Feature Branch. 2009; Available at: https://martinfowler.com/bliki/FeatureBranch.html. Accessed 1.12., 2019.

36 Faber F. Testing in DevOps. The Future of Software Quality Assurance. 2020. Available at: https://www.researchgate.net/publication/337400749_Testing_in_DevOps. Accessed 26.12.2019

37 Fowler M. Test pyramid. 2012; Available at: https://martinfowler.com/bliki/TestPyramid.html. Accessed 25.11., 2019.

38 Frischknecht C. The Place of Manual Testing in DevOps. 2018. Available at: https://www.tricentis.com/blog/manual-testing-devops/. Accessed 1.12., 2019.

39 Nicole Forsgren, Humble J, Kim G. 2015 State of DevOps Report. Puppet Labs. 2015. Available at: https://www.researchgate.net/publication/302566896_2015_State_of_DevOps_Report. Accessed 25.12.2019

40 Allan Wagner. What is Shift Left Testing? 2015. IBM. Available at: https://www.ibm.com/developerworks/community/blogs/rqtm/entry/what_is_shift_left_testing?lang=en. Accessed: 25.12.2019

41 Acceptance Test Driven Development (ATDD). Agile alliance. Available at: https://www.agilealliance.org/glossary/atdd. Accessed 1.12., 2019.

42 Mohamed S. DevOps shifting software engineering strategy Value based perspective. Iosr Journals. 2015.

43 Kupiainen E, Mäntylä MV, Itkonen J. Using metrics in Agile and Lean Software Development – A systematic literature review of industrial studies. Information and Software Technology 2015 Jun;62:143-163.

44 Humble J, Farley D. Continuous delivery. 6. print. ed. Upper Saddle River, NJ [u.a.]: Addison-Wesley. 2012.

45 2013 State of DevOps Report. Puppet Labs, IT Revolution Press. 2013.

46 Brown A, Forsgren N, Humble J, Kersten N, Kim G. 2016 State of devops report. Puppet, DORA. 2016. Available at: https://www.ciosummits.com/Online_Assets_Puppet_2016_State_of_DevOps_Report.pdf. Accessed: 25.12.2019

47 Forsgren N, Humble J, Kim G, Brown A, Kersten N. 2017 State of DevOps Report. Puppet, DORA. 2017.

Appendices

Appendix 1. The State of DevOps survey for the team

1. Continuous delivery capability

	Totally disagree	Somewhat disagree	Neither agree nor disagree	Somewhat agree	Totally agree	I do not know
All project artifacts are stored in version control	○	○	○	○	○	○
Our development process allows continuous integration	○	○	○	○	○	○
Changes are merged to master daily	○	○	○	○	○	○
I can get test automation results easily after each code change	○	○	○	○	○	○
Test automation gates code change requests	○	○	○	○	○	○
Our test automation set is comprehensive	○	○	○	○	○	○
I can run automated test set easily on demand	○	○	○	○	○	○
Our test data is well managed	○	○	○	○	○	○
Our test automation results are reliable	○	○	○	○	○	○
I can setup test environment myself with certain software installed	○	○	○	○	○	○
Our master branch is most of the time in releasable state	○	○	○	○	○	○
My tools are adequate to perform my tasks	○	○	○	○	○	○
Used technologies are appropriate and interesting	○	○	○	○	○	○
IT infrastructure helps me to achieve highest possible outcome	○	○	○	○	○	○
Security is built in the development process	○	○	○	○	○	○

2. Free thoughts about Continuous Delivery practices in the team

Enter your answer

3. Architecture capabilities

	Totally disagree	Somewhat disagree	Neither agree nor disagree	Somewhat agree	Totally agree	I do not know
Our architecture is loosely coupled	○	○	○	○	○	○
Our architecture supports effective continuous integration and continuous delivery	○	○	○	○	○	○
We can deploy the application independently of other applications and services	○	○	○	○	○	○
The team can do decisions related to architecture	○	○	○	○	○	○
We can do testing for components without integrated environment	○	○	○	○	○	○

4. Free thoughts about current software architecture

Enter your answer

5. Product and process capabilities

	Totally disagree	Somewhat disagree	Neither agree nor disagree	Somewhat agree	Totally agree	I do not know
I can get customer feedback easily	○	○	○	○	○	○
My work tasks are small	○	○	○	○	○	○
My work tasks are well defined	○	○	○	○	○	○
I know our definition of done	○	○	○	○	○	○
Our development process is efficient	○	○	○	○	○	○
Our development process is well defined	○	○	○	○	○	○
I know well our development process	○	○	○	○	○	○

6. Free thoughts about product development and process practices in the team

Enter your answer

7. Lean management and monitoring capabilities

	Totally disagree	Somewhat disagree	Neither agree nor disagree	Somewhat agree	Totally agree	I do not know
Change approval process is efficient	○	○	○	○	○	○
When change is made, I can get reliable metrics easily and in short time	○	○	○	○	○	○
Work in progress limits are applied and followed	○	○	○	○	○	○
Work is made visible with information radiators. For example with task boards etc.	○	○	○	○	○	○
Metrics are visible and actionable	○	○	○	○	○	○

8. Free thoughts about Lean product management practices used in the team

Enter your answer

9. Cultural capabilities

	Totally disagree	Somewhat disagree	Neither agree nor disagree	Somewhat agree	Totally agree	I do not know
Team search new information actively	○	○	○	○	○	○
Messenger are not punished when they deliver news of failures or other bad news	○	○	○	○	○	○
Responsibilities are shared	○	○	○	○	○	○
Cross-functional collaboration is encouraged and rewarded	○	○	○	○	○	○
Failures causes inquiry to reveal root cause	○	○	○	○	○	○
New ideas are welcome	○	○	○	○	○	○
Failures are treated primiarily as opportunities to improve the system	○	○	○	○	○	○

10. How likely are you to recommend working in the team to a friend or colleague?

0	1	2	3	4	5	6	7	8	9	10

Not at all likely — Extremely likely

11. Working culture

	Totally disagree	Somewhat disagree	Neither agree nor disagree	Somehwhat agree	Totally agree	I do not know
Team communicates inspiringly	○	○	○	○	○	○
Team has inspiring vision	○	○	○	○	○	○
Team stimulates me intellectually	○	○	○	○	○	○
Team has supportive working culture	○	○	○	○	○	○
Personal recognition is given	○	○	○	○	○	○

12. Agile software development

	Totally disagree	Somewhat disagree	Neither agree nor disagree	Somewhat agree	Totally agree	I do not know
I know and understand Scrum framework	○	○	○	○	○	○
Our team follows Scrum framework	○	○	○	○	○	○
Scrum is applied effectively in our team	○	○	○	○	○	○
Scrum ceremonies are strictly time boxed	○	○	○	○	○	○
There are no additional meetings to Scrum meetings	○	○	○	○	○	○
I know agile principles	○	○	○	○	○	○
I follow agile principles in my work	○	○	○	○	○	○
Our team is self-managing team	○	○	○	○	○	○

13. Free thoughts of Scrum implmentation in the team

Enter your answer

14. My role in team

- [ ] Development
- [ ] Testing
- [ ] UX Design
- [ ] Lead
- [ ] Product owner
- [ ] Other

15. I work mainly in

- [ ] [redacted]
- [ ] [redacted]
- [ ] Other

16. I have been in project

- Less than 6 months
- More than 6 months but less than one year
- Between one year and two years
- More than two years