UPPSALA
UNIVERSITET

UPTEC STS 21013
Examensarbete 30 hp
Mars 2021

Listening in on Productivity

Applying the Four Key Metrics to measure productivity in a software development company

Johanna Dagfalk
Ellen Kyhle

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Listening in on Productivity

Johanna Dagfalk & Ellen Kyhle

Software development is an area in which companies not only need to keep up with the latest technology, but they additionally need to continuously increase their productivity to stay competitive in the industry. One company currently facing these challenges is Storytel - one of the strongest players on the Swedish audiobook market - with about a fourth of all employees involved with software development, and a rapidly growing workforce.

With the purpose of understanding how the Storytel Tech Department is performing, this thesis maps Storytel's productivity defined through the Four Key Metrics - Deployment Frequency, Delivery Lead Time, Mean Time To Restore, and Change Fail Rate. A classification is made into which performance category (Low, Medium, High, Elite) the Storytel Tech Department belongs to through a deep-dive into the raw system data existing at Storytel, mainly focusing on the case management system Jira. A survey of the Tech Department was conducted, to give insights into the connection between human and technical factors influencing productivity (categorized into Culture, Environment, and Process) and estimated productivity. Along with these data collections, interviews with Storytel employees were performed to gather further knowledge about the Tech Department, and to understand potential bottlenecks and obstacles.

All Four Key Metrics could be determined based on raw system data, except the metric Mean Time To Restore which was complemented by survey estimates. The generalized findings of the Four Key Metrics conclude that Storytel can be minimally classified as a 'medium' performer. The factors, validated through factor analysis, found to have an impact on the Four Key Metrics were Generative Culture, Efficiency (Automation and Shared Responsibility) and Number of Projects. Lastly, the major bottlenecks found were related to Architecture, Automation, Time Fragmentation and Communication.

The thesis contributes with interesting findings from an expanding, middle-sized, healthy company in the audiobook streaming industry - but the results can be beneficial for other software development companies to learn from as well. Performing a similar study with a greater sample size, and additionally enabling comparisons between teams, is suggested for future research.

Handledare: Maria Verbitskaya & Jakob Wolman
Ämnesgranskare: Davide Vega D'Aurelio
Examinator: Elísabet Andrésdóttir
ISSN: 1650-8319, UPTEC STS 21013

Acknowledgement

We would like to acknowledge everyone that played a significant role in the accomplishment of this Master's thesis project, done in collaboration with Storytel, as part of the Sociotechnical Systems Engineering program (STS) at Uppsala University.

First and foremost, we want to thank our supervisor and subject reviewer *Davide Vega D'Aurelio*, for support and valuable advice. Secondly, the project would never have been possible without our supervisors at Storytel; *Jakob Wolman* and *Maria Verbitskaya*. Thank you for inspiring leadership and guidance throughout the process.

Lastly, of greatest importance for this thesis is the cooperation with all employees at Storytel who have answered our survey and participated in interviews. We are grateful for the assistance and input from each and every one of you.

Johanna Dagfalk & Ellen Kyhle
March, 2021

Populärvetenskaplig sammanfattning

För att förbli konkurrenskraftig inom mjukvaruutvecklings-branschen idag måste företag, utöver att anpassa sig till det snabbt förändrande teknologi-landskapet, kontinuerligt bli mer produktiva. Ett av de företag som står inför dessa utmaningar idag är Storytel - en av de starkaste spelarna på den svenska ljudboksmarknaden - med ungefär en fjärdedel av sina anställda inom deras tech-avdelning, och med en snabbt växande arbetsstyrka.

I denna uppsats kartläggs Storytels produktivitet med hjälp av fyra nyckelmått (the *Four Key Metrics*) - Deployment Frequency, Delivery Lead Time, Mean Time To Restore och Change Fail Rate - i syftet att öka förståelsen för hur Storytels Tech-avdelning presterar. En klassificering utförs av vilken prestations-kategori (Låg, Medel, Hög, Elit) som Storytel tillhör genom att djupdyka i systemdata med huvudfokus på Storytels ärendehanteringssystem Jira. För att vidare undersöka olika faktorer som påverkar produktivitet på Storytel skickades en enkät ut till hela tech-avdelningen, vilket genererade värdefull insyn i kopplingen mellan faktorer (kategoriserade i Kultur-, Miljö- och Processfaktorer) och estimerad produktivitet. Tillsammans med denna datainsamling utfördes även flertalet intervjuer med anställda för att samla ytterligare kunskap om Storytels tech-avdelning, och för att förstå potentiella flaskhalsar och hinder mot en högre prestation.

Samtliga *Four Key Metrics* kunde bestämmas med hjälp av systemdata, förutom Mean Time To Restore, som istället kompletterades med hjälp av enkät-uppskattningar. Man fann att prestations-kategoriseringen skiljer sig beroende på vilken service eller tech-stack inom avdelningen som undersöks, men de generaliserade fynden från alla *Four Key Metrics* konkluderar att Storytel minimalt kan klassificeras att prestera på medelnivå. De faktorer som avgörs påverka *the Four Key Metrics*, validerade genom statistik faktoranalys, är ‘Generative Culture’, ‘Efficiency (Automation and Shared Responsibility)’ samt ‘Number of Projects’. De huvudsakliga flaskhalsar som hittas är relaterade till ‘Architecture’, ‘Automation’, ‘Time Fragmentation’ och ‘Communication’.

Genom att beskriva utgångsläget för prestationsnivå utifrån dessa fyra mått, och följa upp förändringar genom att analysera måtten och vilka faktorer som påverkar dessa så kan ett team förbättra sin mjukvaruutvecklings-process och uppnå bättre affärsresultat. Denna uppsats bidrar med intressanta fynd från ett expanderande, medelstort och framgångsrikt företag inom ljudboksmarknaden - men resultaten kan även vara lärorika för andra företag inom mjukvaruutvecklings-branschen.

Abbreviations and Important Concepts

Actionable Agile - Actionable Agile is a tool that enables flow charts for metrics such as WIP, throughput, cycle time, and work item age based on for example Jira data.

Agile - Agile is an iterative approach to project management and software development that helps teams deliver value to their customers faster. An agile team delivers work in small increments.

Android - a mobile operating system primarily for touchscreen mobile devices such as smartphones and tablets.

Backend - development concerning the server-side focusing on databases, algorithms and system optimization - namely the portion of systems that you don't see.

Batch size - in software delivery i.e. the amount of code being deployed on average.

Bartlett's test of sphericity - can be used to test that items are unrelated and therefore unsuitable for detecting a structure in Factor analysis. Small significance values (below a threshold of 0.05) indicate that items in the dataset are sufficiently correlated and therefore that factor analysis can be useful.

Bottleneck - some limiting resource with a capacity equal to or less than the demand placed upon it in a system.

Cycle time - In this thesis, the cycle time is the amount of time from work started to work delivered. Generally, the cycle time can refer to fewer steps of the delivery cycle than Delivery Lead Time.

Direct oblimin - Factor Analysis rotation method based on the assumption that the factors are correlated to each other, used to obtain new sets of factor loadings (high loadings maximized) to reach the simplest and most interpretable structure.

DORA - Google's DevOps Research and Assessment team introducing the *Four Key Metrics*.

EFA - Exploratory Factor Analysis: a modelling technique used to discover the number of underlying factors that are influencing variables.

Eigenvalues - Eigenvalues are a set of scalars associated with a linear system of equations. Eigenvalues > 1.0 is used in factor analysis to extract how many factors to retain. Factors less than 1.0 are considered unstable, accounting for less variability than one single item.

eNPS - Employee Net Promoter Score: Conventional method used to rate employees satisfaction with work and loyalty to their employer. It is based on the percentage of employees rating their likelihood to recommend their company for others.

E-factor - Environmental Factor: the fraction of uninterrupted hours at work in proportion to total hours.

Four Key Metrics - Balanced and comprehensive measuring framework for productivity in software development organizations. Consists of Delivery Lead Time, Deployment Frequency, Mean Time To Restore and Change Fail Rate, developed by DORA.

Frontend - development concerning the client-side focusing on conversion of data into graphical interfaces. It involves everything the user experiences directly, such as the visual and interactive side of a system however not the design - but functionality of designs.

Github - internet hosting of code repositories for software development collaboration and version control.

Google cloud platform - suite of cloud computing services that provides infrastructure as a service, platform as a service and serverless computing environments.

iOS - a mobile operating system created and developed by Apple Inc. exclusively for its own hardware, powering most of the company's mobile devices.

Jira - A Case Management System that allows for Agile project management and involves features for planning, distribution of tasks, tracking, prioritizing, and reporting among lots of other features.

KMO - The Kaiser-Meyer-Olkin (KMO) test is a measure of whether a dataset is appropriate to analyse with factor analysis. The score indicates to what degree the items in the dataset are related, by testing the partial correlations among the items.

Lean management - an approach to managing and organizing work that aims to improve a company's performance, involving the employees in improving the work environment. Includes several principles but relies on three simple ideas: to deliver value from your customer's perspective, eliminate waste (things that don't bring value to the end product) and continuous improvement.

Little's Law - The relation between throughput, WIP, and Cycle Time based on the formula: Cycle time = WIP / Throughput.

PAF - Principal Axis Factoring: Extraction method in Factor analysis which seeks to find the least number of factors that can account for the common variance in a set of items.

P-value - statistical measurement that indicates the level of significance of the relationship between correlated factors, used in spearman rank-order correlation. The lower the p-value, the greater the statistical significance of observed difference.

Raw system data - in this thesis referring to data from the case management system Jira.

R-coefficient - A rank correlation coefficient (r_s), ranked between +1 and -1, indicates the strength and direction of the relationship between correlated factors, used in Spearman rank-order correlation.

Slack - business communication platform offering features such as chat rooms (channels), private and public groups and direct messaging.

Software development - the process of conceiving, specifying, designing, programming, documenting, testing, and bug fixing involved in creating and maintaining applications, frameworks, or other software components.

Spearman rank-order correlation - assesses the relationship between two factors without having to take normality of distribution or equal variance of data into consideration.

SPSS - Statistical Package for the Social Sciences: Statistical Software Platform developed by IBM.

Tech-stack - a set of technologies an organization uses to build a web or mobile application. It is a combination of programming languages, frameworks, libraries, patterns, servers, UI/UX solutions, software, and tools used by its developers.

Test club - Cross-sectional cooperation of testers between teams with the purpose of sharing knowledge. Responsible for testing during Freeze time.

Throughput - the units of work (tickets) that are completed within a set period of time.

UI - User Interface: including the visual touchpoints that allow users to interact with a product involving for example combinations of colors, animations and typography that results in aesthetically pleasing usage

UX - User Experience: including the full experience of users contact with a product involving structural design solutions that results in effective usage.

WIP - Work In Progress: the stories or tasks that are currently awaiting completion. Crucial component of Agile development.

Table of Contents

1. Introduction

Software development is an area of study characterized by constant change, and companies need to keep up with the latest technology to stay competitive in the industry. To be able to hold onto the market share and continuously deliver products and services of high value to customers, it is often crucial for companies to increase productivity.

Technology is fundamental in the audiobook streaming industry, and one of the strongest players in the Swedish audiobook streaming market right now is Storytel. Storytel is an audio- and ebook streaming service that is available in close to 30 countries distributed over three continents. Like traditional media, the time came for the book to be digitized, and today the revenues from audiobooks equate to 50% of the market for fiction books. The audiobook industry is characterized by growth, estimates say that the market will grow at least 15% per year (Storytel AB, 2019a).

Tech development is the enabler for a well-functioning subscription streaming service, and in order to be a leader in the audiobook industry it is not enough to have a wide range of book titles, but you also need to have a dominant application (Boktugg, 2020). Currently about a fourth of all employees at Storytel belong to the Tech Department, and the number of Tech employees is rapidly increasing. During 2020, Storytel's Tech Department has increased its workforce from about 100 to 160 employees. As the department increases in size, their teams grow bigger and more features are developed.

Outside the Tech Department, Storytel has an Intelligence Department with the purpose 'to provide data-driven insights regarding the business, customers, and content across the organization'. They are successful in monitoring the productivity of their organization based on these terms with 'business metrics', which are helpful to look at when it comes to deciding about the future and roadmap for the developers' agenda (Storytel, 2021b). At the moment, Storytel is however not utilizing the data that exists for generating insight regarding the flow of work and information in the Tech Department. More employees are recruited continuously, which is generally assumed in the software development industry to equal a higher level of productivity (Brooks, 1995). That might be the case, but having a balanced measuring framework covering the Storytel Tech Department's productivity could validate such assumptions.

Appropriate tech metrics should be balanced and include all necessary dimensions. Dimensions that should be covered are, for example, responsiveness, stability, quality and predictability to enable a holistic view of the current state within a team or a project. By monitoring organizational performance, it is possible to influence and improve organizational productivity. One approach for measuring the performance of a software development organization was recently developed by Google's DevOps Research and Assessment team (DORA) known as the *Four Key Metrics*. Using these metrics can be valuable for historical comparison of the state of the organization, furthermore discovering trends and patterns available which in turn

can be used to evaluate changes made to the organization or serve as the groundwork for learning about how to streamline procedures (Forsgren, Humble and Kim, 2018).

1.1 Aim and research questions

With the purpose of understanding how Storytel's Tech Department is performing, this thesis aims to map Storytel's productivity defined through the *Four Key Metrics*. The *Four Key Metrics* are Delivery Lead Time, Deployment Frequency, Mean Time To Restore and Change Fail Rate, and constitute a balanced framework that measures both the tempo and the stability of the software development process. By measuring these key metrics, a software development team can be classified into one out of four performance categories: Elite, High, Medium, and Low. By creating a performance baseline from these metrics and tracking changes through analyzing them, a team can improve on their work process and achieve better business outcomes.

The following research questions will therefore be investigated:

- Where does Storytel rank in the software development performance categories based on the *Four Key Metrics*?
- What human and technical factors have an impact on Storytel's software development performance?
- What bottlenecks exist that hinder Storytel from being a better performer in terms of a higher performance category?

1.2 Implementation

In the following thesis, a classification is created into which performance category the Storytel Tech Department belongs to. This is done through a deep dive into the raw system data existing at Storytel, mainly focusing on the case management system used at Storytel called Jira (Atlassian, 2019). To look into the factors influencing productivity, a survey of the Tech Department was conducted. The survey took about 10 minutes to respond to and approximately 50% of the Tech Department answered the survey, giving valuable insights in the connection between technical and human factors and perceived productivity. Along with these data collections, interviews with Storytel employees were performed to gather knowledge about the Storytel Tech department in-depth and to understand bottlenecks and obstacles. While some analyses are looking at the team level to validate some findings, the overall focus has been on the department as a whole.

1.3 Thesis structure

Following this introduction (Section 1), the thesis begins with a background of the Storytel context needed in order to create a basic understanding (Section 2). Thereafter, the findings of a literature review on the area of productivity and metrics within software development are presented in the theoretical framework in Section 3. Among lots of factors, reasoning concludes which factors are interesting to look into specifically for the Storytel context. In the concluding part of the theoretical framework, the chosen factors are visualized in the research model together with the metrics. The methodology and implementation (Sections 4 and 5) present every step of the approach of data collection and analysis. Thereafter, the empirical results are given along with some fact-founded analysis (Sections 6, 7 and 8). Results and insights in the Storytel Tech Department are conferred, followed by *the Four Key Metrics* estimations and factor analysis and the bottlenecks discussion. Conclusion (Section 9) wraps up the thesis with reasoning on lessons learned, limitations, and future research.

2. Background: Storytel context

In this section, we present more knowledge about the Storytel Tech Department concerning how the organizations and teams are structured. Further, elaboration is made on aspects like which tools are used in the organization and which workflow stages are used in their software development process.

2.1 About Storytel's Tech Department

During 2020, Storytel's Tech Department increased its workforce from about 100 to 160 employees. This means that they now constitute about one-fourth of all employees at Storytel (Storytel, 2021a). Along with their growth, the tech organization has also been going through a lot of organizational changes. Today there are eleven different teams each with unique focus areas. The number of employees within each team ranges from 5-25. Within the team, there can be several crews with corresponding Crew Coaches (corresponding to Scrum Master), and each team has one Tech Manager. The tech manager's main responsibility is growing the team and the talents in it. How deep they are involved in the feature development is up to each team. Apart from Tech Manager and Crew Coaches, there are different roles within the crews such as developers working on different systems and multiple stacks, testers, and UX/UI designers (Storytel, 2021c). How many people in each role there are depends on the focus area of the team.

The structure of the teams has changed along with the size and needs of the department. Starting off as just a few people in the Tech Department, several reorganizations have happened since then. About two years ago Storytel switched from being divided into tech stack-specific teams (backend, Android, iOS, and web) where dependencies between each other were inevitable, to three different teams with different focus areas which were related to the end-user journey and business metrics. These changes were aimed at reducing dependencies and creating autonomous, independently functioning teams. The second reason was to decrease the number of stakeholders necessary to manage for each team. The teams now all had their own different backlogs, which made prioritizing easier. To adapt to the rapidly growing Tech Department, these three teams were incrementally split during 2020 to make the work of each team more easily managed *(Interview 1: Product Manager, 2020)*.

Most of the current 11 teams are connected to some specific parts of the user journey - from discovering the service, creating and paying for an account, finding and listening to their first audiobook until finally becoming a frequent user. Cross-sectionally between teams there are *clubs*, such as UX club, Test club, and iOS club (Storytel, 2021c). The purpose of these is that people working in similar tech stacks on different teams can share knowledge and have a place to meet and cooperate, such as in regular meetings or dedicated Slack channels (*Interview 3: Developer, 2020)*. Currently, a lot of architectural decisions are made in the clubs *(Interview 16: Tech Manager, 2021)*.

2.2 Communication Tools

The following paragraphs will cover aspects of the Storytel context related to communication. Storytel does not restrict its Tech Department teams to use universal models or tools within its organization, neither frameworks nor programming languages. They generally use a bottom-up approach, giving the development teams the power to make these decisions based on their own expertise, interests, future surveillance, and in an experimental and explorational way. The increase of employees has affected the communication routines and the amount of teams is strongly correlated with the communication quality, efficiency and effort needed. The following tools have either been used to collect data in order for analyses on productivity to be made or are found important to gather understanding on the context in which the Tech Department operates.

Since all teams have their own managers, and a diverse setup of roles and responsibilities, using the same methods to derive their productivity is difficult. Some are frequent users of Storytel's case management system, and some are not. Documentation standards, commit structures, and contribution guidelines on Github differ among the teams. However, the communication tool Slack (Slack, 2021) was introduced at Storytel in 2016 and has been the main channel for communication for all teams in the Tech Department since then. While email as a means of communication is very prevalent in other parts of the organization, it is rarely used within the Tech Department. The communication pattern through Slack somewhat characterizes the culture of the department. Internal communication is mainly handled in a quick setup with low formality. Anyone can easily message any other person directly, or post questions in open channels to find answers. Responses are usually fast and ease simple cooperation both within and between teams and crews.

Storytel's Human Resources Department (HR) additionally utilizes a survey tool to gather insight from the whole organization. Surveys are sent out to employees via email on themes such as 'Wellbeing', 'Leadership', and 'Feedback'. While some surveys are sent frequently, others are sent out once as part of an investigation into a specific focus area. For this study, we have analyzed HR survey data specifically gathered from the Tech Department in 2019 and 2020, see Appendix 4. This data will hereafter be referred to as HR survey data.

The case management system used at Storytel is called Jira (Atlassian, 2019). In general, Jira is used on a daily basis by both developers and Crew Coaches. It allows for Agile project management and involves features for planning, distribution of tasks, tracking, prioritizing, and reporting among lots of other features. In Jira, you can design your own workflow and use several plugins to design your own system of integrations with other tools. Generally, each team has one or several projects in Jira. In some cases, each crew has its own specific project.

In the Jira projects, work is organized between different boards. Each team chooses its own structure, but for example, there can be one board for representing the roadmap of the team as an overview when it comes to prioritizing among *Epics*, a larger body of work that can be divided into a smaller number of tasks (Atlassian, 2021). These tasks are called *stories* (or

tasks) and the rule of thumb is that no *story* should take longer to finish than one sprint of three weeks *(Interview 6: Crew Coach, 2021).* Then the highest rank granularity is found in the developer's board, where the crew members join and concretize the *stories* into smaller *tickets* or *subtasks* (see Figure 1). One ticket should preferably not take longer than two workdays to finish *(Interview 6: Crew Coach, 2021).* At the end of each sprint, boards are cleared, closed, or archived.

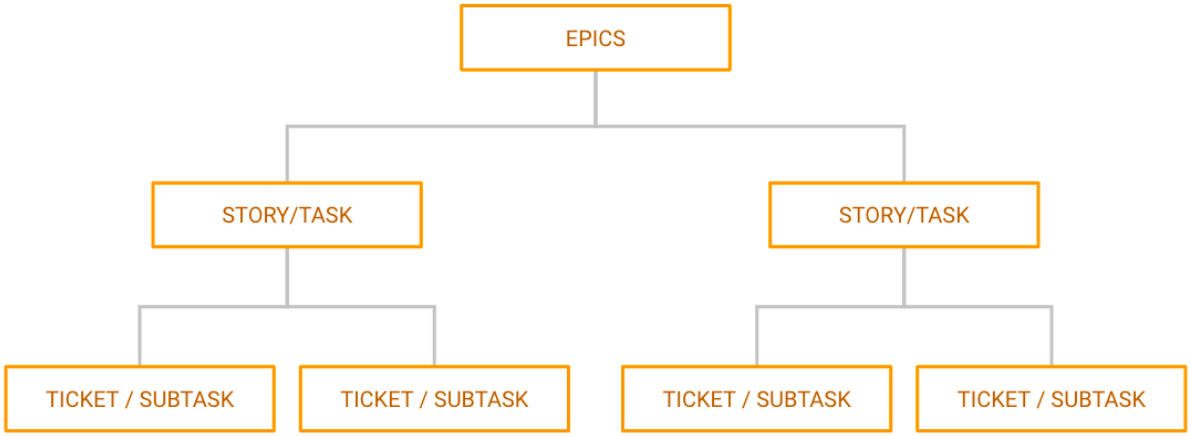

Figure 1. Representation of the hierarchy of Jira issues.

In Jira, automated reports are available. However, they do not allow for much specifications or interaction. Instead, a plugin was used in this thesis called Actionable Agile (Actionable Agile, 2021). This was implemented in Jira for a short trial, but Actionable Agile Analytics was also possible to use separately with imports from Jira. Actionable Agile enables flow charts for metrics such as Work-In-Progress, throughput, cycle time, and work item age and allows for filtering, zooming, enabling different workflow stages, and more.

Github (Github, 2021) is Storytel's code repository. At this moment, there are about 200 collaborators involved in the organization account and over 550 repositories. Among the top languages used are Java, C#, Go, JavaScript, Kotlin, Python, Shell and Jupyter Notebook. The total number of languages used in the organization is around 25, however, they are used to varying degrees. Some are abandoned and some are only maintained but not involved when creating new features.

Several other tools are available depending on the role that you have, for example, tools that are specific to the work of a UX designer. Examples of commonly used tools at Storytel are Delibr (Delibr, 2020); a Jira plugin mainly used for writing specifications and requirements, and Miro (Miro, 2021); a visual collaboration whiteboard suitable for meetings, brainstorming, and workshops. When it comes to these tools, the teams - and in some cases, individual developers - are free to choose their own tools as a part of the ambition to embrace creativity and curiosity to try new things. The same idea applies to programming languages.

Each month, there is a meeting for the entire Tech Department called Monthly Tech. The content of this meeting has partially changed over time. It used to be a check-up meeting where every team had the chance to present what they were working on so that everyone could be up to date on what was going on in the department. With Storytel's growth, this has been set aside in favor of welcoming new employees and general sweeping updates of the most important notices. There are currently too many teams to practically have a proper presentation from each of them every month *(Interview 2: Crew Coach, 2020)*.

2.3 Development process

The following paragraphs will present the aspects of the Storytel Tech Department, for example regarding workflow stages in their software development processes and strategic choices concerning architecture and automated testing.

2.3.1 Services and architecture

Storytel maintains several different services. They maintain an audiobook streaming mobile application for both Android and iOS, which is their main service. In addition, they have internal web tools for employees and external tools for creators such as authors, narrators, and publishers. Furthermore, Storytel maintains a customer web page, several payment-related systems, APIs for partners, and databases.

Storytel is currently going through a migration process, switching from a local server platform to the cloud-based Google Cloud Platform (GCP) (Google Cloud, 2019). This was an initiative that started about 5 years ago, partly due to the ambition of decreasing their climate footprint and one step in the right direction of their sustainability agenda (Storytel AB, 2019b).

2.3.2 Deployment pipeline

The deployment pipelines differ between the Storytel services. For the mobile applications there is a new release every third week according to a schedule that involves time for testing (freeze dates) and coordination with the AppStore (Apple, 2021a) for iOS (Apple, 2021b) and GooglePlay (Google Play, 2021) for Android (Android, 2021). However, the final rollout happens in stages. The complete rollout, which means availability (not reachability) for 100% of the customers, is usually performed after 7 days. Not all customers update their mobile applications every third week, but each release usually has time to reach about 85% of the customers before it is time for a new release *(Interview 10: Tech Manager, 2021)*. This means that it is not straightforward to keep track of which features have an impact on business metrics. This further supports the assumption that business metrics are not sufficient for measuring productivity within the tech teams. There is a long latency and delay to see feature-related changes in the business metrics *(Interview 1: Product manager, 2020)*. To make it more complex, all features are not available in all markets. The release versions do not differ between countries but features can be disabled through a feature flag system *(Interview 16: Tech Manager, 2021)*. Even though the number of 85% reached customers for each release is quite high, this number takes up to three weeks to reach - explaining why released to production does not necessarily mean reaching end-users.

While the routines for app releases have been developed and reached some maturity within the organization, there are initiatives in different stages to introduce similar routines for other services in the company *(Interview 10: Tech Manager, 2021).* For the legacy platform, releases are given a version number and are deployed at an interval of one week *(Interview 8: Test Lead, 2021).* For the internal and external web tools, changes are being deployed more frequently as the teams usually can release new features independent of other teams. Generally, for teams working on these, there are deployments every week *(Interview 3: Developer, 2020).* These releases are communicated to affected users to varying degrees, with a decreasing trend - but they lack version control *(Interview 3: Developer, 2020; Interview 6: Crew Coach, 2021).*

2.3.3 Test pipeline

Since one year ago, Storytel has one person employed as Test Lead. This role grew from the necessity of a coordinator to keep all the testers at Storytel organized, which had become quite many along with the growth of the Tech Department. The role was established with the aim to increase the level of tests and the overall quality. With a background as a tester, it also means that the Test Lead can help or temporarily replace someone in the test organization. In conclusion, this role has both a strategic and operational focus. Initiatives in Storytel to improve the testing organization are based on the theoretical concept to *shift left* - meaning that testing should be included earlier in the development cycle - and to expand the degree of automation in the testing activities *(Interview 8: Test Lead, 2021).* Storytel is striving towards further automating regression testing, but is still in early phases, with the gain of being able to repeat tests often and cheaply *(Interview 8: Test Lead, 2021).*

In order to strengthen the relationship between the Tech Department and customer support, and to some extent enable customer feedback-driven work, the new role within customer support called the Global Support Technical Administrator, appeared in March 2020. Cooperation is mainly orbiting what is called the *Bug Refinement Sessions,* happening each Friday *(Interview 11: Customer Support, 2021).* At this meeting, there is a chance to discuss bug prioritization, Customer Service insights into present bugs, and lift overall questions between the Customer Support representative, Crew Coaches, and testers. If it is not possible to wait until this meeting due to the urgency of the bug, Slack channels are used *(Interview 8: Test Lead, 2021).*

The testers are team-specific during the development phase, and they either test source code themselves during this time or they serve as a coach to the developers to manage their own testing. When it is time for releasing the applications a freeze date tells the teams when it is not possible to push new code, as testing commences. This exists so that the whole test club can test the upcoming release material together. After the freeze date, it no longer matters which part of the code belongs to which team, as they are encouraged to test each other's teams' work *(Interview 8: Test Lead, 2021).*

3. Theoretical framework

In this section, the findings of a literature review on the area of productivity and metrics within software development are presented. Among a lot of potential factors theorized to impact software development productivity, reasoning concludes which factors are interesting to look into specifically for the Storytel context. In the concluding part of the theoretical framework, the chosen factors are visualized in the research model together with *the Four Key Metrics* chosen to measure productivity.

3.1 Productivity

Productivity in the field of software development is notoriously challenging to measure because of the complexities of the tasks and processes it involves. The traditional definition of productivity as being output divided by input may sound straightforward, but defining what constitutes input and output in a software development process presents many challenges. The output needs to be evaluated in terms of both quantity and quality, among other dimensions. Regarding the input, the key ingredient in a software development process is *people*, and the qualities and skills of people are also famously difficult to quantify. (Wagner and Ruhe, 2018)

There has been a lot of research done on the area of productivity within software development over the years, and consequently, efforts have been made by researchers to collect and review these findings. Wagner and Ruhe (2018) conducted a systematic review intended to overview productivity factors in software development. They have collected hundreds of relevant studies and present them with a timeline perspective which serves as valuable groundwork for future research in the field of measuring productivity. However, the large amount of influencing factors presented in their research highlight the difficulties in finding a simple measurement tool (Wagner and Ruhe, 2018).

According to Wagner and Ruhe (2018), literature within the software engineering productivity area has had a strong emphasis on mostly technical factors. Consequently, Wagner and Ruhe (2018) have been careful to also analyze human-related, ‘soft’ factors, hereby referred to as *human factors*, with equal detail. The importance of involving these human factors for productivity surfaced during the ’90s, partly because of the comprehensive work on the influence of soft factors by DeMarco and Lister (Wagner and Ruhe, 2018). Wagner and Ruhe (2018) present the human factors and technical factors separately, but highlight that the line between these can sometimes be fuzzy. The factors are listed in their paper with a short description but do not involve details of how factors may affect productivity positively or negatively. The human and the technical factors are further divided into categories. The five categories within the human factors are: corporate culture, team culture, capabilities and experiences, environment, and project-specific factors. The technical factors are divided into the three categories of product, process, and tools (Wagner and Ruhe, 2018).

3.2 Measuring productivity

3.2.1 Background on measuring productivity

Defining metrics to measure productivity and quality in software development has been an important research area for many decades. In the book *Accelerate* (Forsgren, Humble and Kim, 2018), the authors discuss the flaws of a few traditional attempts to measure productivity in software development, such as lines of code, velocity, and utilization, which are all relatively ineffective and misleading for different reasons. According to the authors, measuring *lines of code* - historically a rather favored method - sets an incentive for developers to write bloated software that in turn requires more maintenance and a higher cost of change. Using *velocity* as a metric of productivity is a relative and team-dependent measure, which can cause teams to try and inflate their estimates by working on completing as many tasks as possible while avoiding collaboration with other teams - as to not increase others' velocity at the expense of their own. Finally, the flaw in measuring *utilization* as an indicator of productivity is that when an entire team is working at full capacity, there is no spare capacity that can handle changes to the plan such as unexpected workloads or improvement work. Ultimately, having a utilization rate close to 100% leads to teams taking exponentially longer to get work completed. The authors argue that a successful performance metric should avoid these pitfalls by fulfilling two key requirements: they should focus on global outcome to ensure that teams are not competing against each other; and they should focus on *outcome* rather than output, work that contributes towards achieving organizational goals (Forsgren, Humble and Kim, 2018).

Meyer et al (2014) emphasize that there might not be a single or simple measure for a developer's productivity. Wagner and Ruhe (2018) share their concerns, and refer to Ramirez and Nembhards saying that 'it seems to be a common agreement that to date there are no effective and practical methods to measure knowledge workers' productivity'. However, there are strong incentives to attempt to make these measurements, which is why researchers and organizations keep trying. First of all, measurements can prompt action. Secondly, they can serve as the foundation for goals and aligning actions accordingly. They are also important for advocating when seeking investments and to justify and confirm actions (Github, 2019).

When discussing productivity, it is common that some vocabulary is used interchangeably. Words like productivity, performance, efficiency, and quality are used synonymously. Moreover, it is important to remember that measuring commercially can differ quite a lot from measuring academically (Construx Software, 2016). There are not only difficulties connected to how or what to measure, but also how to evaluate the results. Furthermore, one must take into account the potential risk of unwanted effects from implementing measurements.

A good productivity metric should indicate factors that are within the teams' influence to change and feel relevant to the individuals involved. They should be linked to company strategy so that the output it measures is aligned with organizational goals. To ensure sustainability, they should also be of low cost and effort to capture. As no single metric can capture enough information to give a good indication of team productivity, it also needs to be balanced by other complementary metrics. Bad metrics tend to pit teams against each other and focus on local

outcomes rather than global outcomes. If the metric is linked to personal reputation, it also runs the risk of causing detrimental social effects within the team. (Øredev Conference, 2015)

A risk of using metrics is that if they are set up as a target, they risk being abused. If good metrics are in place, their abuse will generally lead to desirable outcomes (Øredev Conference, 2015) Attitudes towards metrics may also vary. In a study by Meyer et al. (2014) 10% of participating software developers stated that they do not think it is possible to measure productivity, that they have privacy concerns, or that they believe that the measuring itself might in fact lead to a decrease in productivity.

3.2.2 Four Key Metrics

Metrics intended to capture productivity quantitatively need to be balanced to ensure that an organization gains any real value and insight from using them. They should include dimensions such as responsiveness, stability, quality, and predictability to enable a holistic view of the current state within a team or a project. This requires multiple complementary metrics (Øredev Conference, 2015).

In the book *Accelerate* (Forsgren, Humble and Kim, 2018), the authors propose the *Four Key Metrics* to quantitatively measure productivity and indicate performance level in the software development context. The focus of these metrics are on global rather than local, team-level outcomes, and aim to measure outcome rather than output that does not actually contribute to organizational goals. The idea is that by measuring these key metrics, a software development team can be classified into one out of four performance categories: Elite, High, Medium, and Low. In Figure 2 the categories are represented in a matrix, with corresponding intervals separating the different categories. Each of the top three categories are divided by time spans, indicating speed for Delivery Lead Time and Mean Time To Restore, and frequency for Deployment Frequency. The metric Change Fail Rate is divided by the percentage proportion of 'failed' changes to a service. The matrix and intervals are constructed by the DORA team based on their research findings on software development organizations placing on a global scale.

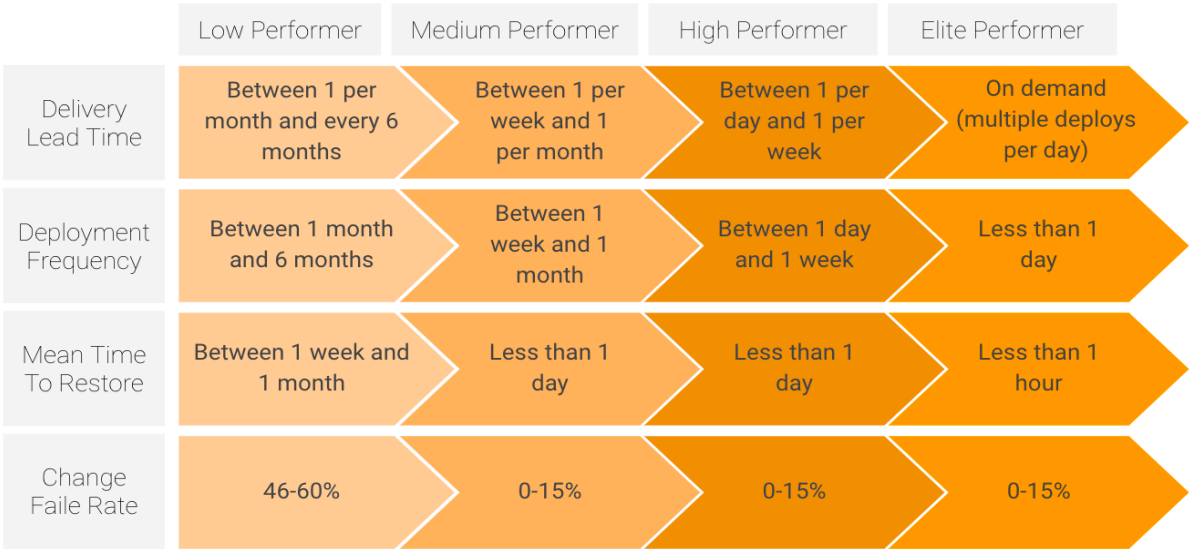

Figure 2. The Four Key Metrics and their respective classification into four performance categories.

The first proposed metric is ***Delivery Lead Time***. When measuring the lead time it is often not clear where to begin, due to the difficulty of defining what constitutes the beginning of the product development process. One relatively stable metric is using the delivery part of the lead time, as opposed to beginning with the product design and development phase. This includes the building, testing, and deployment and can be translated to the time it takes to go from code committed to code running in production. Shorter product Delivery Lead Times are preferable as they enable a quicker feedback loop and consequently faster course correction (Forsgren, Humble and Kim, 2018).

The second metric is ***Deployment Frequency***, used as a proxy measurement of batch size, i.e. the amount of code being deployed on average. By reducing batch size, one can enable faster cycle times, accelerate feedback loops and reduce overhead and risk. As the batch size is not made up of visible inventory in software development it is tricky to measure, and therefore Deployment Frequency, defined by a software deployment to production or an app store, is used to approximate batches (Forsgren, Humble and Kim, 2018).

As these two metrics are indicators of the *tempo* of the product development, they need to be balanced by measures indicating the reliability and quality of the developed product, namely the *stability*. This allows for a more complete picture to be derived, and for impacts and tradeoffs between the metrics to be found (Øredev Conference, 2015). Reliability is generally measured as the time that passes between failures, but as failures are impossible to avoid in modern software services and products as systems are becoming increasingly complex, the interesting measure instead becomes the time it takes for service to be restored in the inevitable case of failure. The third metric is therefore defined by Forsgren, Humble and Kim (2018) as the ***Mean Time To Restore***.

Finally, the fourth metric is a measure of quality defined as ***Change Fail Rate***. It is measured as the percentage of changes made to the primary service or application that result in either degraded service or a need for remediation such as a patch, roll-back, or a hotfix. (Forsgren, Humble and Kim, 2018)

This framework can be applied to indicate what performance level an agile software development organization is at compared with other companies on the market, as well as allow for an unbiased historical comparison of the state of the organization. Using the metrics, internal trends and patterns can be observed over time and in turn be utilized to indicate what kind of impact different decisions and events have had on productivity in the organization.

3.3 Categories influencing productivity

Based on the findings in Section *3.2 Measuring Productivity* of what makes a good metric, the factors were narrowed down to what the metrics should cover, fit for the Storytel context. Based on the findings of Wagner and Ruhe (2018) and Forsgren, Humble and Kim (2018), eight categories of factors influencing productivity were extracted (see Figure 3). In this thesis, the

most relevant categories within the human factors are found to be the corporate- and team culture factors, since they are company-wide and involve the team. Capabilities and experience, on the other hand, are related to the individual and are therefore most reasonable to exclude. Between project-specific factors and environment factors, the environment-related ones are deemed more interesting - since these will more likely continue to be relevant in the future. For the technical factors, it is concluded that good metrics are focused on the process rather than the product (Github, 2018). The factors connected to the choice of tools would preferably be excluded in favor of a measurement framework that can be relevant no matter what software development tools are currently trending. Since Storytel is flexible regarding tools and product-related factors and allows these to be easily interchangeable, the product and tool category can be considered less significant for this thesis. The chosen categories (Culture, Environment, Process) most relevant for the Storyte context are highlighted in a darker color, and the corresponding factors that will be the focal point of this study and are visualized in Figure 3. In the following sections, each factor will be described and contextualized.

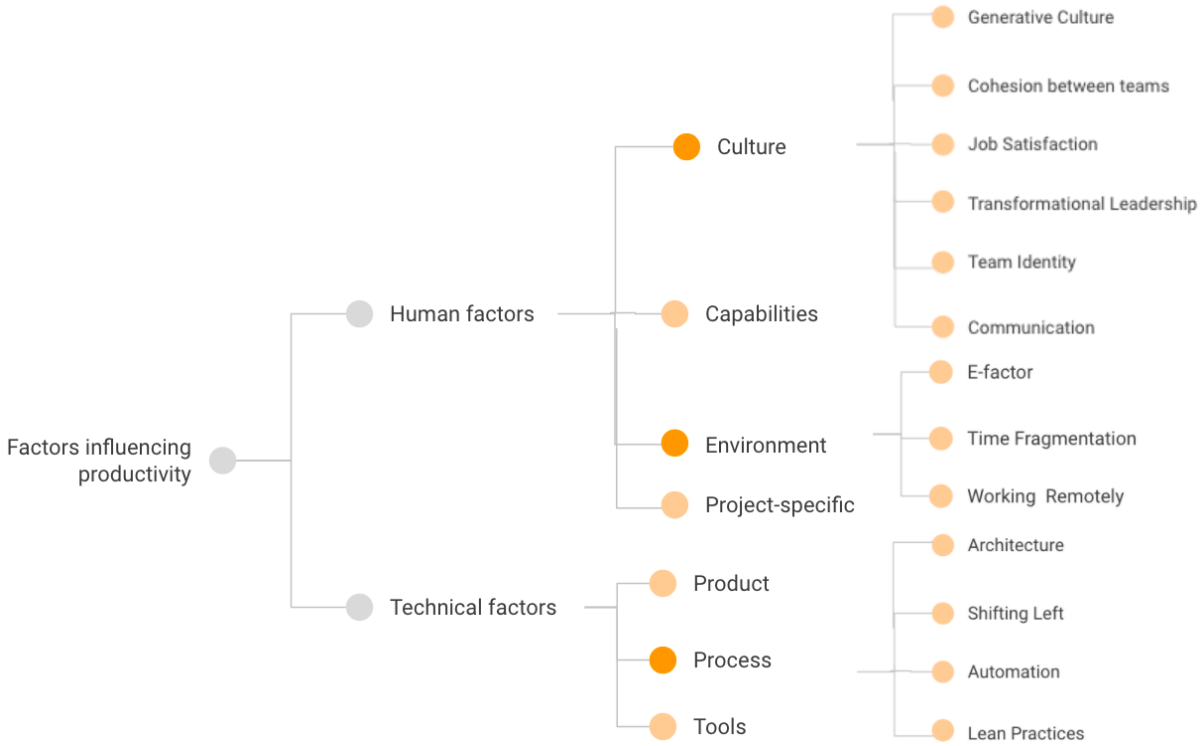

Figure 3. Factors influencing productivity from chosen categories

3.3.1 Culture Factors

Academic literature has long recognized the impact of *culture* on productivity and quality in software development organizations (Mathew, 2007). In this subsection, a number of factors that are theorized to measure the culture in an organization are described. These are generative culture, job satisfaction, transformational leadership, team identity, cohesion between teams, and communication.

Westrum (2004) found that cultures that optimize information flow, also known as *generative* cultures, were found to be particularly predictive of desirable organizational outcomes. Westrum introduced 'The three cultures model' in which three typical patterns are identified in organizational cultures. The first culture is power-oriented and pathological, in which cooperation is low, novelty is crushed and responsibilities are shirked. The second culture, in the middle of the spectrum, is distinguished by bureaucracy and rules and marked by modest cooperation and narrow responsibilities. The third and final culture is generative and performance-oriented within which risks are shared, cooperation is encouraged and novelty is implemented. The concentration in the organization is on the mission, rather than positions and individual people. Westrum emphasizes that the flow of information needs to be timely and presented in such a way that it can be used efficiently and provide the right answers to the questions that the receiver needs answered. (Westrum, 2004). The culture needs to promote meaningful work, psychological safety, and clarity to generate high-performing teams. A generative culture is listed by Forsgren, Humble and Kim (2018) as one of the capabilities found to drive higher software delivery performance, organizational performance, and productivity. In order to measure Westrum cultures accordingly, they have tested seven statements related to the dimensions in Table 1. to be both valid and reliable.

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 Leads To Scapegoating	Failure Leads To Justice	Failure Leads To Enquiry
Novelty Crushed	Novelty Leads To Problems	Novelty Implemented

Table 1. Westrum's culture framework with three different types of cultures.

According to Forsgren, Humble and Kim (2018), *job satisfaction* - signified by employees feeling that their work is meaningful, that their judgement is valued, and that they have access to the right tools and resources to perform their job - is a predictor of organizational performance. Engaged employees that bring the best of themselves to work produce better work results which consequently results in a higher software delivery performance. The feeling of fulfillment in one's job is an emotional state and naturally a perceptual measure that cannot be directly quantified, but a commonly used proxy metric is to measure Employee Net Promoter Score (eNPS). The idea behind eNPS is to ask how likely it is that an employee would recommend their company as an employer to a friend or colleague on a scale and that this score reflects the respondent's level of satisfaction with their employer. The eNPS score can be calculated from 5 point scale survey answers by retracting the share of 'detractors' (those who score in the bottom range, between 1-3) from the share of 'promoters' (those who score in the

top range, 5) (Sedlak, 2020). An eNPS score can range from -100 to 100, and generally, scores between 10 and 30 are considered 'good'. A score above 50 is considered 'excellent' (Madhavan, 2019).

Forsgren, Humble and Kim (2018) additionally found that the style of leadership in a team has a measurable and significant impact on organizational productivity and software delivery. The model of *transformational leadership* has been emphasized and embraced as a way an organization can encourage its employees to exceed expectations. Transformational leaders motivate their followers and 'transform' their attitudes, beliefs, and values (Rafferty and Griffin, 2004). Rafferty and Griffin (2004) identify five characteristics of a successful transformational leader that are highly correlated with performance. These characteristics are vision, inspirational communication, intellectual stimulation, supportive leadership, and personal recognition. In a study by Ali, Farid and Ibrarullah (2016), transformational leadership was additionally found to have a significant effect on job satisfaction and organizational commitment. Transformational leadership can be measured directly by asking team members to what extent they perceive their leaders to exhibit these characteristics (Forsgren, Humble and Kim, 2018).

Demarco and Lister (1987) argue that teams with a strong *sense of identity* are more effective because the team members are more directed. The reason that teams with a strong sense of identity are more likely to have aligned goals, and in turn are more likely to attain those goals. Strong team identity can be signified by members having a joint feeling of ownership of the product, and that they feel that they are part of something unique and that they take enjoyment in their work.

Aligned with Demarco and Lister's (1987) line of argument in the previous paragraph and Westrum's (2004) finding that cultures that optimize information flow drive performance, it can be theorized that *cohesion between teams* is a factor that similarly influences productivity. Insight into what other teams are working on and corresponding transparency into one's own team can promote cooperation, information flow, and cohesiveness between different teams within the organization, and in turn promote organizational performance.

A large software development project typically includes a lot of requirements to fulfill and a diverse set of roles, and therefore a good *communication structure* is fundamental. To meet requirements and divide the workload, projects need to be divided up into multiple tasks, many of which might be interconnected in a chain. In the book *The Mythical Man-Month: Essays on Software Engineering* (1995), author Frederick P. Brooks finds that most tasks within software engineering projects are tasks with complex interrelationships, and therefore they become more and more time-consuming the more people you add to the task. As a general rule, Brooks argues that assigning more software developers to a project with the purpose to speed up the process will lead to a further delay because of the time it takes for the new recruits to learn about the project and the increased communication overhead. This simplified observation is known as Brook's Law (Brooks, 1995).

While there is a common belief within the field of software engineering that efforts on communication should be reduced as they hamper productivity due to interruptions, Wagner and Ruhe (2018) suggest the opposite. They find that several studies advise that higher communication intensity is positively correlated with successful projects, and that the communication efforts should therefore strongly correlate with the increasing number of people in the organization (Wagner and Ruhe, 2018). The importance of communication can be derived from Conway's law, based on Melvin Conway's publication 'How do committees invent?' (1968), proposing that an organization's communication structure will inevitably be mirrored in the software systems that are designed within the organization (Brooks, 1995). This basically means, that in order for a software module to function, the authors developing it must communicate frequently.

3.3.2 Environment Factors

The work environment, both in terms of physical as well as time-management and workflow-related components, is naturally a significant aspect of an employee's work life and consequently their day-to-day productivity. In this subsection, factors theorized to measure impactful aspects of the organizational environment are described. These are time fragmentation, E-factor, and working remotely

.

Fragmentation of employees' time is brought up by Demarco and Lister (1987) as one of the main obstacles for efficiency and productivity, and mention this as being a consequence of when people are involved in too many projects. They argue that a good work environment should afford employees to work uninterrupted in a flow. Similarly, Meyer et al (2014) highlight interruptions and switches and how they concern productivity. Switches can be separated into different kinds. Task, activity, and context- switches all have different impacts on productivity and can be of both positive and negative character for the individual as well as for the team. An interruption from coding for one developer, for example, to review code from someone else in the team, can possibly prevent a bottleneck for a teammate. A task switch for the individual is therefore not necessarily negative for the productivity of the whole team (Meyer et al., 2014). The impact of the number of uninterrupted hours a software developer has access to in regards to productivity has been contested in different studies. Meyer et al. (2014) studies showed that over half of the developers' time was spent in interactive activities other than coding. Wagner and Ruhe (2018) present the same estimate to be that a third of the time the typical software developer is not working explicitly with technical work.

Following Demarco and Lister's (1987) idea that uninterrupted hours is a prerequisite for a productive work environment, the collection of uninterrupted hour data can be a meaningful metric of how good or bad a work environment is - they name this metric the Environmental Factor, or the *E-Factor*. They argue that when there is a low number of uninterrupted hours in proportion to total hours, approximately below 40%, this can imply reduced effectiveness and frustration among employees. A number above 40% indicates an environment that allows employees to get into a flow when they need to.

Things like communication patterns, performance management as well as the work itself undergo a transformation when an employee starts *working remotely* (Watad and Will, 2003). Bloom et al. (2014) have investigated whether working remotely affects job performance. In the study of Ctrip, a company located in Shanghai, the authors found that the introduction of remote work increased the performance of employees by 22 percent. One reason for the increased performance, the authors suggest, is because the remote workers worked more minutes as they took fewer breaks. Another reason was found to be connected to a quieter and more convenient working environment. They conclude that tasks requiring concentration may be best undertaken at home, whereas other tasks involving teamwork may be best undertaken in the office. Naturally, this is depending on the employee's individual prerequisites at home and living situation whether working from home allows for more uninterrupted hours than at work, or fewer.

Individual effects of working from home, Bloom et al. (2014) identified as fewer redundancies and a significant increase in job satisfaction. Harpaz (2002) and Bellman and Hübler (2020) continue to write about the advantages and disadvantages of the individual working from home. Among other things, individuals experience more flexibility, better time management, and savings in expenses and travel time. On the other hand, the individual may also experience a feeling of isolation, a poorer division between work and private life, and a lack of professional support (Harpaz, 2002).

3.3.3 Process Factors

Factors belonging to the process category measure technical aspects of the software development process. Those estimated to be most relevant to the Storytel context are described, mainly based on Wagner and Ruhe's (2014) and Forsgren, Humble and Kim's (2018) research. These factors are mainly part of the concept of continuous delivery, including architecture, 'shifting left', automation, and lean management practices like visual management and limiting Work-In-Progress.

Continuous delivery is described by Forsgren, Humble and Kim (2018) as the ability to release all kinds of changes to production 'quickly, safely and sustainably' and is supported by their research to have a measurable impact on software delivery performance. It is about increasing throughput while simultaneously lowering risks, and promotes prioritizing keeping software deployable over working on new features, ensuring that feedback on quality and deployability of the system is available to everyone on the team, and working in small batches. Continuous delivery is implemented by adopting a number of different practices related to automation, security design, and architecture. Eleven contributing components are mentioned by Forsgren, Humble and Kim (2018), and four of those found most applicable to this study are described.

One of the practices of continuous delivery is *loosely coupled architectures* (also known as *microservices*) which allow organizations to achieve better delivery performance and reduce the pain of deployment. In microservice architectures, services and applications are units that can be deployed or released independently of services it depends on, and services that depend on it (Forsgren, Humble and Kim, 2018).

Another aspect that is closely tied to continuous delivery is the move to '*shift left on security*', i.e. address security concerns earlier in the development process in order to build more secure systems and achieve higher levels of software delivery performance. Traditionally security testing is done after development is complete, which typically means that if significant issues are discovered - such as architectural flaws - they are expensive to fix. Furthermore, when testing activities are carried out towards the end of each development cycle and since development processes are rarely completed on time - the testing process tends to suffer most by being cut off. Additionally, the effect of 'shifting left' has been observed to improve communication and information flow (DevOps Research and Assessment, 2021).

Automation is another key feature to continuous delivery, both in regards to testing and deployment. *Test automation*, performed alongside a degree of manual testing, can be used to increase test reliability and regularity, which leads to lowered risks and increased quality. *Deployment automation* similarly enables more reliable and risk-free deployment to production. The impact of both of these factors can be indicated by investigating the percentage of automation in their respective pipelines (DevOps Research and Assessment, 2021).

Complementing the principles of continuous delivery, a set of practices categorized as *lean management* is proven by Forsgren, Humble and Kim (2018) to improve software delivery performance, decrease burnout, and lead to a more generative culture. Out of the four practices described by the authors, the two deemed most relevant are emphasized below (left out are '*Feedback from Production'* and *'Lightweight Change Approvals'*.)

Visual management entails enabling greater visibility for the team into their collective work through key productivity and quality metrics, which can promote a greater understanding of the flow of the entire work process. Metrics (for example lead times and failure rates) are presented on dashboards or other visual displays. Teams that are proficient in implementing work visibility have a greater understanding of how their work moves from idea to customer, and are in turn empowered to improve their workflow (Forsgren, Humble and Kim, 2018).

Limiting work-in-progress, the number of tasks team members are working on, drives process improvement and increases throughput. These lean management practices protect teams from becoming overburdened and expose obstacles to the flow of work. Interestingly, it has been observed that solely constricting the number of Work-In-Progress, hereby referred to as *WIP*, in a team does not in itself have an impact on software delivery performance, but only when this practice is combined with the use of visual displays a strong positive effect can be observed (Forsgren, Humble and Kim, 2018).

3.4 Self-rated productivity

How a developer perceives their own productivity can be of interest in relation to more quantitative metrics of productivity. Productivity can be measured on several levels, both organizational, team-specific, and individually and in this thesis, emphasis will be placed on organizational and team-specific levels of productivity. However, it can still be valuable to account for how developers perceive themselves as productive as a helpful support to measure and assess productivity on higher levels (Meyer et al. 2014). Studying the developers' perceptions of their own productivity - besides hypothetically being indicative of organizational productivity - might also indicate the culture and attitudes of the organization.

Furthermore, the developers' feeling of the productiveness of the organization can have an indirect impact on productivity, since it can affect the mood of a developer and eventually the performance level. There are studies showing the causal link between human-well being and human performance, that provide evidence that happiness makes people more productive (Oswald, Proto and Sgroi, 2007), as well as the opposite, that depression and beliefs about cognitive confidence independently predicted behavioral procrastination (Spada, Hiou and Nikcevic, 2006). Some suggest that fluctuating emotional states of humans should be seen as input in designing a model that seeks to increase the productivity of a system (Pakdamanian, Shiyamsunthar and Claudio, 2016).

Through a survey and observational study, Meyer et al. (2014) gathered perception data. The survey showed that developers think about productive days in terms of ones in which many or big tasks are completed without significant context switching or interruption. Developers also like to organize their work to get in “the flow” so as to have few interruptions and context switches. However, from observational data, it was found that significant context switching between tasks and activities can occur with developers still perceiving themselves as productive (Meyer et al 2014). Therefore, it would make sense to communicate and support developers in reflecting upon their productivity and sharing best practices for work habits - to achieve and support the feeling of productiveness.

3.5 Throughput and finding bottlenecks

Throughput - the units of work (tickets) that are completed within a set period of time - can also be valuable to look at as an indicator of productivity. Meyer et al. (2014) asked software developers about which measures might be helpful to them to assess their productivity, and the metric with the highest rating was found to be ‘The number of work items (tasks, bugs) I closed’, which supports that throughput can be used as a complementary measure to the *Four Key Metrics*. Developers were also interested in the value of their work. Participants mentioned that performing useful, necessary, and interesting work and having the feeling of being necessary to the team or product is very important, i.e. the feeling of being productive is important for motivation to continue to be productive.

There are similarities between agile and lean software development that goes back to traditional manufacturing methods. Just like in manufacturing, the continuous flow in the assembly line

is what quickly delivers value to customers and makes money. The development process can be seen as a continuously running factory that collects, implements, tests, and releases requirements (Petersen and Wohlin, 2011). Within the field of software development, inventory is generally of a more perishable nature compared to manufacturing. Produced work will be outdated at a quicker rate as market requirements and customer needs change rapidly. This calls for even faster throughput, i.e. delivery of working code into production that can be generated into value (Anderson, 2004).

The relation between throughput, WIP, and Cycle Time (the amount of time from work started to work delivered) is described in Little's law, with the formula Cycle time = WIP / Throughput (The Agilist, 2014). This concept is one of the main theories behind the agile methodology, meaning that the lower WIP you have combined with the higher throughput, the faster features can be delivered. To be able to decrease WIP and cycle time, bottlenecks need to be found and managed. Bottlenecks can be defined as some limiting resource with a capacity equal to or less than the demand placed upon it in a system. Cycle times grow if e.g. testing is a bottleneck. WIP grows e.g. if there is a dependency issue that means one feature cannot be finished until collaboration is performed, so the developer starts with another task meanwhile. It is when analyzing the process flow, bottlenecks can be identified in order to improve throughput (Petersen and Wohlin, 2011).

3.6 Research model

Based on the literature review, relevant variables theorized to impact software development performance are used to construct a research model, visualized in Figure 4. Arrows visualize the theorized impact on productivity as a whole, and not necessarily one specific metric. The *Four Key Metrics* are used to measure productivity in a sufficient, comprehensive, and balanced way.

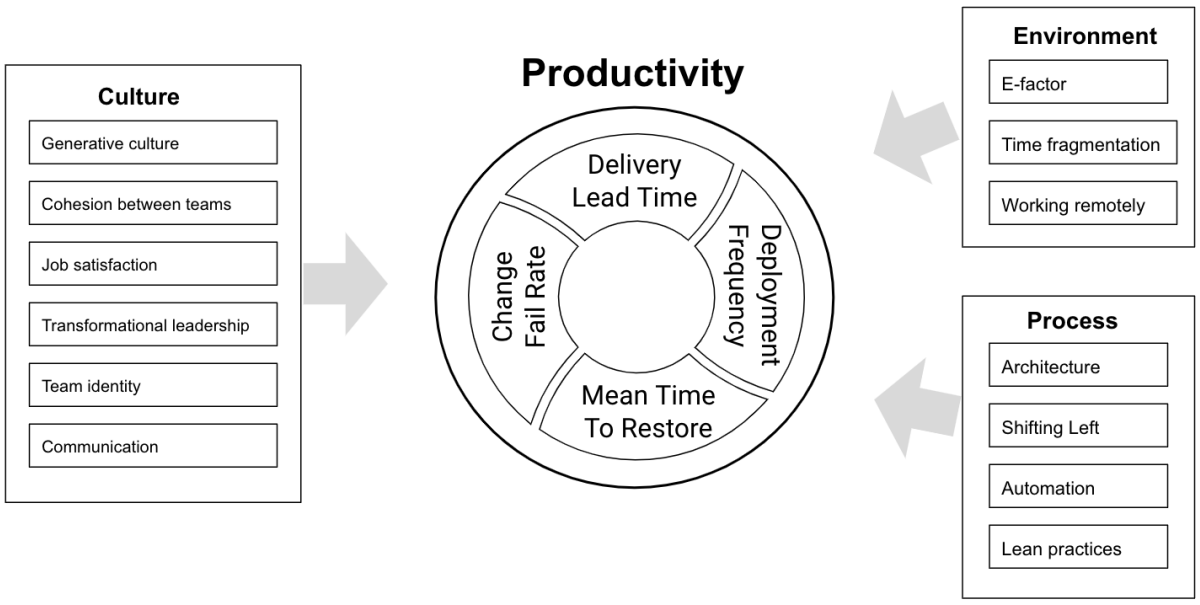

Figure 4. Research model describing the relationship between influencing factors and productivity, measured by the Four Key Metrics.

4. Method

This chapter describes the methods used and related decisions made during this study. Explanations are made to validate the choices of research design, data collection methods, and statistical measurements. It also elaborates on how the implementation went and a summary of which obstacles got in the way.

4.1 Research design

Forsgren, Humble and Kim (2018) describe the differences between methods of collecting data from the software development process. Two options were considered; looking directly into the system tools and data logs to get insights about influencing factors or gathering this data through a survey. Both are primary research, as the data is collected first-hand, and depending on what kinds of questions are asked in the survey, both methods can be quantitative.

Quantitative methods are suitable when studies are related to quantity, frequency, or how usual a phenomenon is (Trost, 2012), but it can also be used to find variation in variables (Djurfeldt, Larsson and Stjärnhagen, 2010). System data can be found in systems or tools used by the organization, as well as from surveys. If a survey asks questions that capture responses in a numerical format, or on a Likert scale, it is System data. There are advantages with both kinds of methods, but Forsgren, Humble and Kim emphasizes that collecting *survey system data* can actually obtain more benefits than only using *raw system data*.

Raw system data is limited to reflecting what is happening inside the system boundaries, while people have the ability to see the surrounding context (Forsgren, Humble and Kim, 2018). Asking questions to individuals in the Tech Department can create a more holistic view of the system than the more simplistic representation that raw system data would generate. The vocabulary regarding productivity in the area of software development can be confusing as definitions and perceptions diverge, and different terms are often used interchangeably. By using surveys there is a large opportunity to avoid misinterpretations (Forsgren, Humble and Kim, 2018). However, it means that the survey needs to be carefully worded to avoid differing interpretations and consequently unreliable data.

To look into the factors influencing productivity and evaluate the research model, a survey was conducted. The survey took circa ten minutes to respond to and approximately 50% of the Tech Department answered, providing insights into the connection between technical and human factors and perceived productivity.

Factor categories theorized to influence productivity are measured in the survey using *latent constructs*. Using a latent construct is a way of measuring something that cannot be measured directly, such as culture. Instead, questions are asked that can capture indicator variables that *represent* the underlying construct. These are also called *manifest variables* (Forsgren, Humble and Kim, 2018). For example, the latent construct ‘Culture’ can be indicated by measuring the manifest variables ‘Job Satisfaction’, ‘Transformational Leadership’, etcetera. Through

secondary research, eleven manifest variables have been extracted that are hypothesized to have an influence on productivity.

Consequently, these can also be interpreted as latent constructs that cannot be measured directly. Instead, multiple secondary manifest variables or *items* have been set to describe each of these constructs. An item can be measured directly through a question.

In this thesis, a literature review was performed in order to overview previous findings on influencing factors on productivity in the software development process. Based on these findings, a research model was constructed with those factors determined most relevant in Storytel's context. Furthermore, an online survey was used to collect data from Storytel's Tech department to evaluate these influencing factors and their respective impact on software development productivity. Raw system data has been analyzed to complement the survey data in estimating the *Four Key Metrics*. Apart from collecting information from internal documentation, interviews have been performed in both an exploratory and confirmatory manner to gather qualitative data. All of them have been performed in a semi-structured way, where questions and themes have been prepared in advance. Because of the varied data collection and strategies used to fulfill the research questions, each method is described respectively.

4.2 Data collection

Data has been collected in both qualitative and quantitative manner. The quantitative was performed through both a survey and system data analysis. The qualitative collection included 15 interviews with people in different roles of the tech organization.

4.2.1 Quantitative data

4.2.1.1 Survey

Applying statistical analysis to these latent constructs makes it possible to ensure that there is validity in the chosen theoretical approach, as well as validity and reliability in the operationalization of the theoretical background. The survey involved a section of demographics capturing which team and role the respondent belongs to, a section of comparative historical perspective for those who have been employed for more than 12 months, a section covering the factors defined in the research model, and a section where the respondent estimates the *Four Key Metrics* in their team. The section with *Four Key Metrics estimations* was done in order to complement the raw system data in several aspects. We set out to measure eleven factors in the questionnaire using a total of 32 questions (or items). Most questions (23/32) were five-point Likert scale questions; statements to which the respondent answered by choosing an option between 1 ('Strongly Disagree') and 5 ('Strongly Agree') (SurveyMonkey, 2018). The complete questionnaire can be found in Appendix 1, and the factors can be found below:

- Number of Projects (3 items)
- Generative Culture (6 items)
- Team Cohesion (3 items)
- Job Satisfaction (2 items)
- Transformational Leadership (3 items)
- Team Identity (3 items)
- Communication (2 items)
- E-Factor (4 items)
- Architecture (2 items)
- Automation (2 items)
- Lean Management (2 items)

When designing the questionnaire, finding the correct latent constructs and manifest variables is not enough, as it is the actual questions in a survey that are the indicators for the theoretical concepts. The operationalization is of great importance, so the questions need to be carefully worded. Common weaknesses in survey questions presented by Forsgren, Humble and Kim (2018) are the following:

- Leading questions
- Loaded questions (means that there is no option to choose the right answer, so the respondent is forced to lie by choosing one of the existing alternatives)
- Multiple questions in one
- Unclear language

In shaping the questionnaire, we paid careful attention to the order, the complexity, and the number of questions to ensure that they were hard to misinterpret and that they serve to answer the research purpose. (Esaiasson et al. 2017).

To be able to generalize the results for a sample population, there are four potential sources of error to take into account. These errors include (a) sampling error which will be present when certain members of the population from which responses are obtained are deliberately excluded, (b) non-coverage error is the error that the survey is not participated in by some part of the population, (c) non-response error comes from the fact that some members of the population do not respond to the survey for various reasons, and (d) measurement error refers to the discrepancy between undiscovered, underlying variables such as certain opinions or behaviors, and the observed survey responses. This might stem from questions being phrased so that they cannot be answered correctly, or from some hidden motivation in respondents to provide inaccurate answers (Dillman, 1991).

The non-response error is traditionally viewed as the major problem for mail surveys. To maximize response rates and maintain quality responses while still minimizing the risk of non-response error, an approach called Total Design Method (TDM) designed by Dillman (1991) was applied. This framework posits that respondents will respond to a larger extent if they perceive that the benefits of responding are greater than the cost. Thus, every part of the survey

was designed with three considerations: making the questionnaire appear easier and less time-consuming, making it interesting for the respondent to fill out by adding attention-grabbing questions, and focusing on increasing trust, for example by using some official format on the survey. The specific TDM design recommendations that are applicable to online surveys are the following: order questions so that the interesting ones that are related to the topic in the survey description come first; use graphical design and question-writing principles to facilitate the respondent's task of reading and answering questions, and include an explanation on how confidentiality is protected (Dillman, 1991).

In a survey-based study, it is important to test the questionnaire. By performing a pilot study the researcher can ensure that the questions are designed in a way that the respondents can understand and that they provide information that answers the research questions (Bryman, 2011). A pilot study was conducted involving students with terminology knowledge, and our supervisors. This gave us an opportunity to revise unclear questions and improve the format.

The questionnaire was sent through Google Forms. It is easy to set up and supports good presentation management of answers. Furthermore, it is beneficial in the way that respondents can decide upon when and where to answer the survey, which ensures anonymity and confidentiality (Bryman, 2011). Expectations of a rather small share of responses prompted us to send out the survey to the entire Tech Department. We cannot enlarge the target group in order to receive a certain amount of responses, but we can avoid choosing a smaller number of people, for example, based on selected teams. Furthermore, the survey is distributed to all roles within the department. Surveying everyone in the teams provides perspectives ranging from manager roles to those closer to the actual value-creation in the software development process, such as developers.

4.2.1.2 Raw system data

To classify Storytel's performance, system data was analyzed using several different data sources (described in Table 2). These were chosen over in favor for example Github because of their ease in accessibility and no need for help from administrators at Storytel. Each system or tool will be presented individually.

Source	Type of data
Actionable Agile	The *Cycle Time Scatterplot* visualizes how long it takes to finish a single item of work once started. The number of started but unfinished work items over time is shown in the *WIP run chart.* The number of finished items in a period of time is shown in the *Throughput run chart.* (Actionable Agile, 2021).
HR Survey Data	Data has only been gathered at a department level to avoid spreading sensitive information connected to specific teams. A limitation was made to only study surveys from 2020, and in some instances from 2019. The response rate is relatively steady within the Tech Department at around 80%.

Table 2. System data sources utilized in this thesis.

4.2.2 Qualitative data

When preparing interview questions, templates were made. The templates served as a tool for conducting semi-structured interviews. For this study, semi-structured interviews were preferable, with a combination of fixed themes but still having the possibility to ask further questions (Bryman, 2011). Generally, there is a risk with interviews that answers will be affected by the phenomena social desirability (Bryman, 2011). However, it is of great value to be able to ask unique follow-up questions (Eriksson and Wiedersheim-Paul, 2014). The templates were structured containing central themes and questions covering the most important concepts needed to answer the research questions (Dalen, 2008). Based on previous interviews, the template was adjusted along the way.

Interviewees were asked to participate, with a short introduction to topics and purpose of the study. Interviewees in a wide spectrum of roles and responsibilities have been covered. In the end of each interview we asked to be recommended who the interviewee thought would be appropriate for us to talk to next. Furthermore, supervisors helped us out finding good candidates.

4.3 Analytical methods

Several statistical methods have been applied to analyze the data obtained from our survey and these are explained in the following subsection.

4.3.1 Statistical methods

Factor analysis is a statistical method in which the key concept is that multiple observable variables have similar patterns because they are all associated with some latent variable - a variable that the researcher is interested in but which cannot be directly measured. The goal is to understand to what extent some measurable items can reflect this hypothetical construct.

Exploratory Factor Analysis (EFA) is a modelling technique used to discover the number of underlying factors that are influencing variables, and to analyze what factors seem to correlate with each other. It is used to develop theories. The goal of EFA is to identify the groups of items that when jointly considered explain as much of the observed covariance as possible. Each of these groups are called a *factor* or a latent variable. It should be noted that EFA is not a statistical way to conclude whether the extracted factors are correct or not, and is commonly used when the researcher has no hypotheses of the underlying factors (Yong and Pearce, 2013). One of the most widely-used methods of extraction in factor analysis is *Principal Axis Factoring* (PAF), which seeks to find the least number of factors that can account for the common variance in a set of items. A common way of deciding how many factors to extract is by using the *Kaiser criterion*; all factors with an eigenvalue below 1.0 are dropped (an eigenvalue of 1.0 is equal to the information accounted for by one single item) (Field, 2009). After extraction, each item obtains a *factor loading* for each factor (between -1 and 1), which represents to what extent the item correlates to the factor (Mabel and Olayemi, 2020).

The procedure begins with defining individual constructs theoretically. To account for unidimensionality between and within construct error variance, at least four constructs and three items per construct should be defined. Then the validity of the model needs to be assessed, which is performed by comparing the theoretical model to the reality model and evaluating how well the data fits (Statistics Solutions, 2013).

After an initial solution is obtained through factor analysis, the loadings on each factor are *rotated* to obtain a new set of factor loadings. This is in order to maximize high loadings and minimize low loadings so that the simplest possible interpretable structure can be achieved. One common rotation method is *direct oblimin*, which is based on the assumption that the factors are correlated to each other.

Before proceeding with interpreting the extracted factors, tests to assert that the dataset is appropriate to analyze using factor analysis should be evaluated (IBM, 2021). The Kaiser-Meyer-Olkin (KMO) measure of sampling adequacy is standard test procedure for this purpose. The KMO-score indicates to what degree the items in the dataset are related by testing the partial correlations among the items. As a general rule, the KMO score should be at least 0.60 to justify that factor analysis makes sense (Schwarz, 2011). Furthermore, Bartlett's test of sphericity can be used to test the hypothesis that the items are unrelated and therefore unsuitable for detecting a structure. Small significance values from Bartlett's test of sphericity (below a threshold of 0.05) indicate that items in the dataset are sufficiently correlated and therefore that factor analysis can be useful (IBM, 2021).

To further analyze correlations between factors without having to take normality of distribution or equal variance of data into consideration, *Spearman rank-order correlation* can be applied. Spearman's rank-order correlation assesses the relationship between two variables. A rank correlation coefficient (r_s), ranked between +1 and -1, indicates the strength and direction of the relationship, and the p-value (p) indicates the level of significance of the relationship. A p-value lesser than 0.05 indicates that the relationship between the two variables is significant,

i.e. there is less than a 5% chance that the strength of the relationship found (r_s) happened by chance (Al-jabery et al., 2020).

4.3.2 Validity and Reliability

The purpose of testing validity is to give the researchers a high degree of confidence that the chosen methods are useful in finding scientific truth. There are several different types of validity, and in this subsection content validity, construct validity and reliability will be discussed (Straub, Gefen and Boudreau, 2004).

Content validity centers on the question whether the measures chosen to capture the construct are wisely chosen. They need to represent the construct well and capture its essence. If measures that do not represent the construct well are included, measurement error is likely to happen. On the other hand, if measures are omitted, the error will stem from this exclusion. Generally, content validity can be tested through literature review (Straub, Gefen and Boudreau, 2004).

Construct validity centers on the issue of measurement *between* constructs, and whether the measures capturing the construct are 'balanced' or not. If measures are grouped together in different manifest constructs, we would want to be assured that these variables are most closely associated with other variables in the same construct. If construct validity is established, the researcher can rule out the possibility that the constructs - which are artificial, intellectual constructions that the researcher cannot observe - are not being captured by the choice of measurement instrumentation (Straub, Gefen and Boudreau, 2004). EFA can be used to evaluate construct validity; items that do not empirically belong to the constructs can be identified and eliminated (Knekta, Runyon and Eddy, 2019).

In contrast to construct validity, *reliability* is the issue of the measurement *within* a construct. The metrics chosen to measure a specific construct may involve aspects of a construct that are different to the degree that they do not correlate (Straub, Gefen and Boudreau, 2004). It can be viewed as a statistical measure of the reproducibility of the survey data. By ensuring reliability, the researcher avoids instability of responses over time and questions being perceived differently by different respondents (Litwin, 1999). *Cronbach's alpha* is one of the most common indicators of scale reliability in factor analysis such as EFA, and can be viewed as a measure of internal consistency (Osborne, 2014).

5. Methodology

In this section, the implementation of the method is presented. Demographics, datasets and obstacles in data collection are also conferred. A timeline of the implementation can be found in Figure 5.

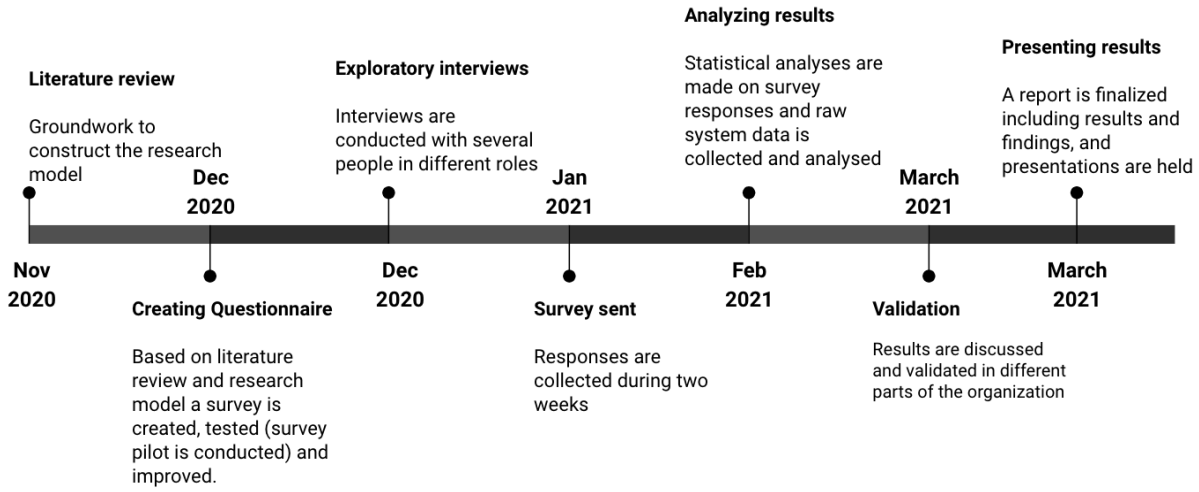

Figure 5. Timeline of the implementation process of this thesis.

5.1 Survey

After the literature review, one of the first tasks was to construct and test the survey. Then finally, the survey was sent out in January 2021 and was open for answers for two weeks, with a cover letter describing purpose, data usage and privacy concerns. An additional encouragement was written from our supervisors to promote the benefits of the survey for the Tech Department. Several reminders were sent in Slack channels, and we got a slot during a Monthly Tech Meeting to promote the survey.

The survey has been carried out according to the Swedish Research Council guidelines that involves demand for information, consent, confidentiality and utilization (Lindstedt, 2017). First of all, respondents were *informed* of the purpose study and that participation was completely voluntary in the cover letter (Esaiasson et al., 2017), and that they give their *consent* to participate when they send in their answers. Regarding *confidentiality*, names were not collected, however team affiliation and role was. Raw survey data was only available to the authors. Finally, the records were and will not be *utilized* for anything else than the purpose of this scientific study (Lindstedt, 2017).

The statistical analyses applied to the survey data have mainly been performed using the statistical software platform SPSS (Statistical Package for the Social Sciences) (IBM, 2019).

5.1.1 Descriptive statistics

The survey was sent to all employees within the Tech Department, and no matter what their role was, everyone was encouraged to answer. Half of the department (75 employees) answered the survey. The total number of people in the Tech Department is represented in the historical

graph, see Figure 6, showing the increase of approximately 60 people per year during 2019 and 2020.

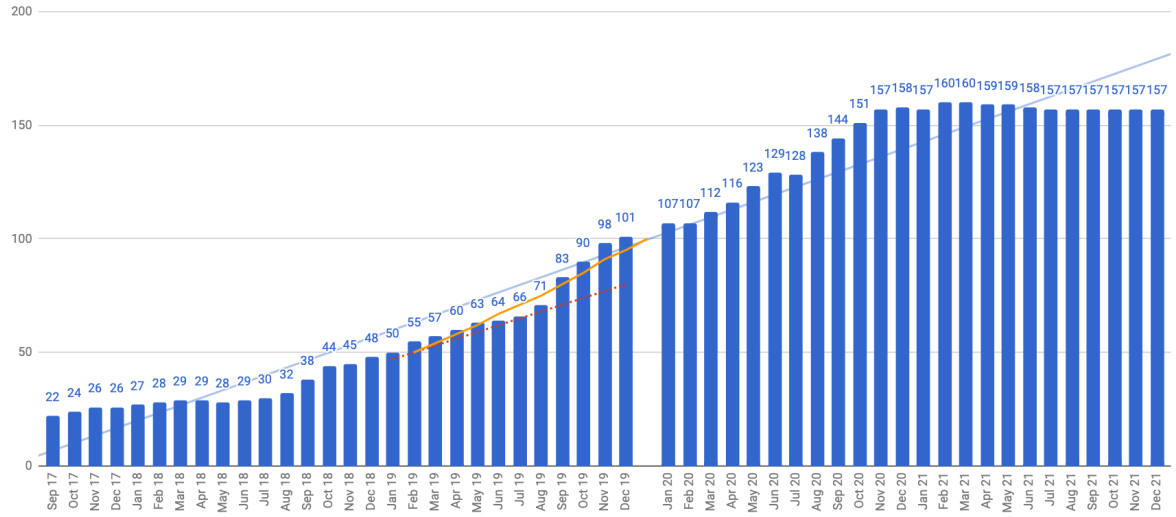

Figure 6. Tech Department Employee Tracking document for 2018-2020, indicating the number of employees each month.

The distribution between roles among the respondents in categories with 5 or more respondents, are shown with the respective number of people in each category in Table 3, equivalent to circa 75% of the respondents.

Role	N	Percentage
Backend Developer	25	33.3 %
Crew Coach	8	10.7 %
Frontend Developer	6	8.0 %
IOS developer	7	9.3 %
Tech manager	5	6.7 %
Full-stack Developer	5	6.7 %

Table 3. Responses to question 'What is your primary role?'. Role categories with 5 or more respondents shown - circa 75 % of the population (56 out of 75).

To validate that the distribution of roles in the survey data reflected the proportions of roles in the Tech Department, a graph was created from internal documents for comparison (see Figure 7). It was found that the majority of Back-end Developers among the respondents, mirrored the actual distribution of roles in the Tech Department. From this data, it could further be assumed that Crew Coaches are more prone to answer this kind of survey, since they are the second largest group of respondents - but a smaller percentage among all employees. For example, this

is compared to that almost no testers answered the survey while this role has a larger percentage in the company. There are more iOS developers than Android developers, and this is mirrored among the respondents.

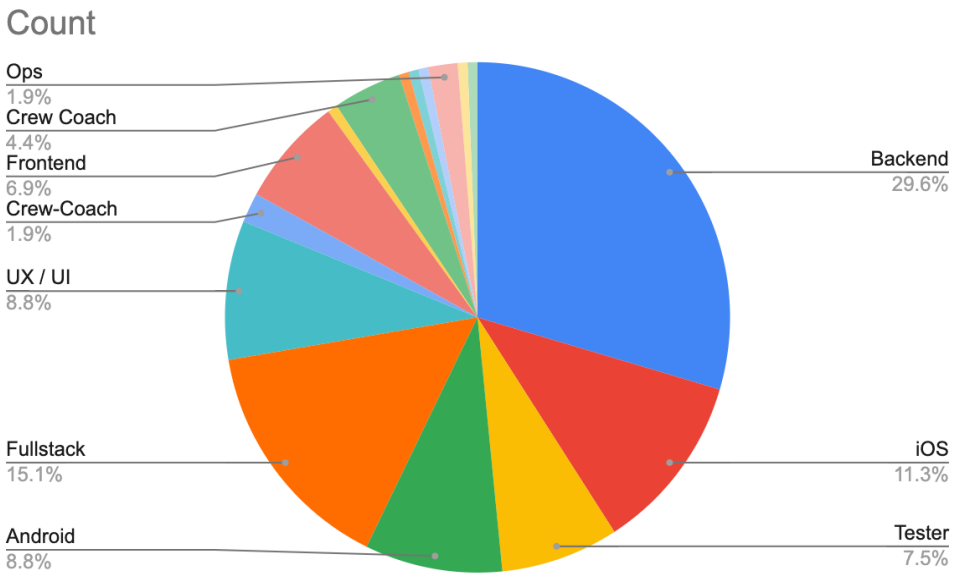

Figure 7. The distribution of roles among all employees at the Storytel Tech department according to the internal Tech Department Employee Tracking document.

Looking at the distribution of respondents, one can conclude that it is not enough data to be able to draw conclusions from comparisons between roles and their answers. Regarding the distribution among teams from respondents, Figure 8 shows an expected distribution. The teams differ in size, so it is reasonable that the number of respondents differ. To not reveal sensitive information, the names of each team have been anonymized. In the end, comparisons between both roles and teams were never performed.

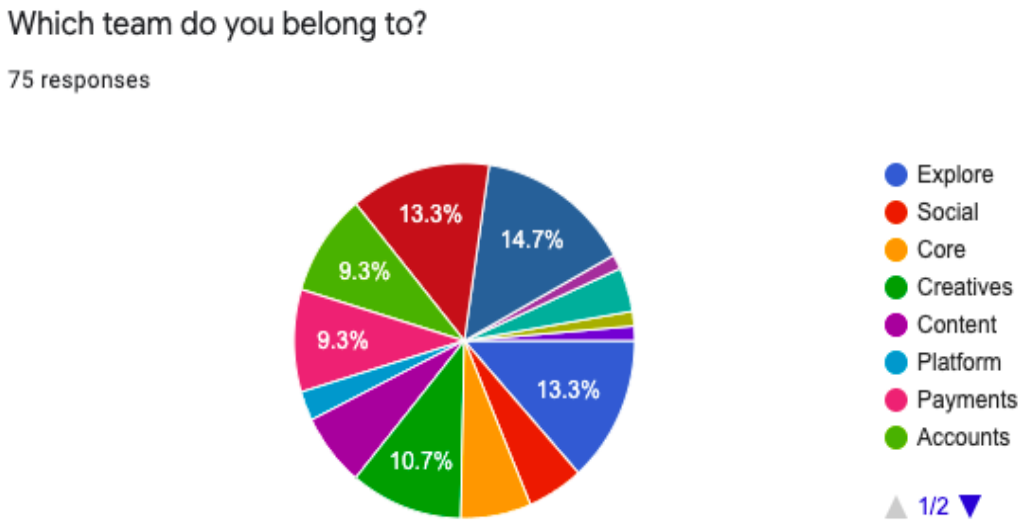

Figure 8. The distribution among survey answers to the question 'Which team do you belong to?'.

If the respondent had been employed more than 12 months, they were able to answer a complementary section of questions related to a historical perspective. As can be summarized from Table 4, this applied to a little more than half of the respondents. The distribution among respondents employed time has the same proportions as the whole Tech Department. Among the 160 employees in the beginning of 2021, 100 employees have been employed for more than 12 months. This equals 62%, compared to the 54% in the survey. In conclusion, recently employed personnel are slightly more prone to answer this kind of survey.

Time Employed	N	Percentage
Less than 6 months	19	25.3 %
Between 6 months and 1 year	16	21.3 %
Between 1 and 2 years	23	30.7 %
Between 2 and 5 years	9	12 %
More than 5 years	8	10.7 %

Table 4. Responses to question 'How long have you been at Storytel in total?'.

The demographics questions were added after careful deliberation, as having to give potentially identifiable information might impact people's tendency to take part in the survey because complete anonymity subsequently is not ensured. The anonymization is not complete since combining answers could potentially reveal sensitive information (if looking at for example team affiliation, role and time in Storytel combined, there is not a large amount of options on who the respondent is). Since the team affiliation proved to be difficult to use due to time constraints and lack of opportunity to find the corresponding raw system data, this could potentially have been omitted. However, if this could have increased the response rate is hard to tell.

5.2 Factor Analysis

Some factors were only measured using two items, in order to simplify the questionnaire and in turn increase the response rate. This is theorized to be an insufficient amount to account for unidimensionality between and within construct error variance, and can be viewed as a potential flaw in the survey design (Statistic Solutions. 2013)

5.2.1 Pre-processing

To perform the EFA, the dataset containing the survey responses is loaded into SPSS. The majority of survey items are measured on a Likert scale between 1 and 5 (a total of 23 out of 32 items), and do not require further pre-processing for analysis. Some items are re-coded into new variables in order to translate nominal responses into an ordinal scale, and responses

stating ‘I don’t know’ are filtered out. For example, the item ‘How many projects are you working on right now simultaneously?’ with responses ranging from ‘1-2’ to ‘More than 10’ is re-coded into a new variable with corresponding responses but on a scale from 1 to 6. Similarly, questions regarding automation phrased ‘What is, in your estimate, the percentage of tasks related to X that are automated in your team?’ with options ranging from ‘0-20%’ to ‘81-100%’ are also re-coded into a six point scale.

Two new variables were created using mathematical expressions combining other variables. The new variable ‘E-factor’ was calculated as the fraction between the items ‘How many complete full hours without interruptions do you have?’ and ‘How many hours is your average work day?’ to express the fraction of uninterrupted hours on an average work day. The new variable ‘Fraction desired uninterrupted hours’ was calculated as the fraction between ‘How many consecutive uninterrupted hours would you prefer to have on a regular working day?’ and ‘How many complete full hours without interruptions do you have?’ to reflect the fraction of average uninterrupted hours to the respondent’s preferable amount of uninterrupted hours. The three included items (‘How many complete full hours without interruptions do you have?’, ‘How many hours is your average work day?’, ‘How many consecutive uninterrupted hours would you prefer to have on a regular working day?’) were subsequently left out of the factor analysis as these are highly correlated with the new variables.

Survey items connected to demographics and historical perspective were also excluded as these were not intended to measure any factors. The full list of factors and corresponding variables are presented in Appendix 2. The item ‘How many projects are you working on right now simultaneously?’ (NP1) was removed as it was highly correlated (0.81) with the item ‘How many projects have you worked on during the last three months?’ (NP2). According to Field (2009), highly correlated items (as mentioned in Section *4.3.1 Statistical Methods*; above 0.9) may create problems because of multicollinearity, i.e. a near perfect linear relationship between two or more variables in the data. Although the variable score of item NP1 was not above 0.9, it was removed as it was the only score in the correlation matrix exceeding 0.8. It was also deemed as not adding any significant information that item NP2 did not already inform, as both items indicated the respondent’s involvement in what number of projects during a relatively short time span (last three months versus currently).

The item ‘In my team messengers are not punished when they deliver bad news’ (GC3) was also removed due to a very low KMO-score (0.388) (see Section *4.3.1 Statistical methods* for a description of the method) in the Anti-image correlation matrix. According to Field (2009), all variables with an individual KMO-score below 0.5 should be considered for removal as the proportion of variance explained by the variable is very low (Field, 2009). The final list of the 29 remaining items are listed in Appendix 2.

5.2.2 Factor Analysis Operationalized

To perform the factor analysis, Principal Axis Factoring (PAF) was conducted on the 29 items with oblique rotation (Direct Oblimin). The rotation method was used in preference over an

orthogonal method as it allows for some degree of correlation between factors. The Kaiser-Meyer-Olkin results (0.690) verified that the sample size was adequate for analysis (above 0.6 is classified as 'mediocre' and above 0.7 is 'good' according to Field (2009)). Additionally, Bartlett's test of sphericity indicated that correlations were sufficiently large to perform PAF ($\chi^2 = 657.899$, $p = < 0.001$).

Applying Kaiser's criterion of extracting factors with an eigenvalue exceeding 1.0 (Field, 2009), ten factors were retained for the final analysis, and aggregated these explained 60.28% of the variance. Table 5 shows the factor loadings after rotation. Initially, those with factor loadings less than 0.4 were suppressed, based on Steven's (2002) suggestion that this is an appropriate cut-off for interpretive purposes. At a later stage, a threshold of 0.35 was applied supported by Osborne (2014) who mentions that this is preferred by some EFA authors, and because this slight adjustment made a meaningful difference in the analysis pertaining to how many items that were sufficiently loaded to be included in the factors (5 additional items). In those cases where an item had a loading above the threshold of 0.35 on more than one factor (TI2 and GC4), the item was interpreted as only belonging to the factor in which it had the highest score in order to simplify overall interpretation.

Included items (with factor loadings > 0.35)	**Factor loading (including loadings on secondary factors)**
Factor 1	
TI1 - My team is collectively working toward the same goal	.806
TI2 - I know the reason for all features developed in my team	.483 (-.467 on Factor 4)
C1 - Communication is efficient in my team	.391
Factor 2	
NP2 - How many projects have you worked on during the last three months?	.911
NP3 - How many projects have your team been involved in during the last three months?	.654
Factor 3	
TL3 - My manager regularly gives med actionable feedback	-.910
TL1 - My manager challenges me to see problems from new perspectives	-.849
TL2 - My manager notices me	-.696
Factor 4	
GC5 - In my team, failure causes inquiry so that we can learn from the experience	-.775
GC4 - In my team, cross-functional collaboration is encouraged and rewarded	-.480 (-.458 on Factor 7)
GC6 - In my team, new ideas are welcome	-.451

GC1 - In my team, information is actively sought	-.443
Factor 5	
JS2 - I would recommend my team as a place to work	-.872
JS1 - I would recommend my workplace s a place to work	-.651
TI3 - I am proud to be a part of my team	-.495
Factor 6	
TC1 - In my team we put effort into facilitating work for other teams	.759
AU1 - What is, in your estimate, the percentage of tasks related to deployment that are automated in your team?	.649
AU2 - What is, in your estimate, the percentage of tasks related to testing that are automated in your team?	.529
GC2 - In my team, responsibilities are shared	.519
LM2 - In my team the ambition is to keep the number of WIP to a minimum	.376
Factor 7	
LM1 - I have access to visual displays showing the status and/or flow of work within my team by some metrics.	-.558
C2 - How often do you interact with members from other teams for inspiration and/or assistance for a task you are working on?	-.367
Factor 8	
EF3 - Fraction Desired Uninterrupted Hours	-.558
EF1 - E-Factor	-.367
Factor 9	
TC3 - I wish I had more insight into what other teams are doing.	.725
TC2 - I have a good insight into what other teams are doing	-.352
Factor 10	
EF2 - Switching between tasks can be good in terms of being productive	.621
AR1 - Features developed in my team can be tested and deployed without being dependent on other teams.	.412

Table 5. Survey items categorized into factors according to the factor analysis, with corresponding factor loadings after rotation.

5.2.3 Factor Analysis Reliability

As factor analysis is used to validate the questionnaire, the *reliability* of these factors need to be measured. This is performed using Cronbach's α to indicate the factors' internal consistency.

Generally, researchers agree that an alpha of 0.7 is ‘adequate’ and above 0.8 is ‘good’. (Osborne, 2014). According to Hinton, Mcmurray and Brownlow (2014) a score above 0.5 can also show moderate reliability. The results from the performed Cronbach’s α test can be found in Table 6.

Factor	Item-codes	Cronbach’s α
1	TI1, TI2, C1	.760
2	NP2, NP3	.778
3	TL1, TL2, TL3	.869
4	GC1, GC4, GC5, GC6	.720
5	JS1, JS2, TI3	.788
6	TC1, AU1, AU2, GC2, LM2	.747
7	LM1, C2	.355
8	EF1, EF3	.546
9	TC2, TC3	.644
10	EF2, AR1	.353

Table 6. Cronbach’s α reliability test

Factors 1 through 5 all have adequately high reliabilities, with Cronbach’s α > .70. As seen in Table 5, factor 6 had a few items scoring below 0.3 (.263 and .273) in the inter-item correlation matrix, which indicates that those particular items do not correlate very well with the scale overall and may cause unreliability. This was ignored due to a relatively high Cronbach’s α score. Factor 7 and Factor 10 scored low, both significantly below the thresholds for Cronbach’s α, and were discarded. From an interpretive standpoint, these two factors also contained the least clear sets of items from a variation of separate theoretical constructs. Factors 8 and 9 scored relatively low, and do not pass the general rule of thumb for Cronbach’s α of a score greater than 0.7, although they do pass the ‘moderate’ threshold of 0.5, and therefore these factors are considered reliable in this analysis.

5.3 System data

5.3.1 Obstacles in Four Key Metrics estimation

In order to estimate the *Four Key Metrics*, a combination of both raw system data and survey data are used in a complementary manner. This was performed since estimates of the *Four Key Metrics* may vary from person to person due to differing perceptions and access to information, and because solely using raw system data can be erroneous and unreliable due to differing procedures between teams, recent organizational restructurings etc. This subsection will

summarize the difficulties and obstacles for the raw system data estimation of each metric respectively. A comprehensive explanation of the obstacles can be found in Appendix 3. The tempo metrics are Delivery Lead Time and Deployment Frequency, and the stability metrics are Mean Time To Restore and Change Fail Rate.

Obstacles related to Tempo Metrics mostly revolve around that the teams in the Storytel Tech Department have a very high degree of autonomy. As a consequence of this, the way different teams work often differ significantly from each other. As a consequence, there were difficulties in trying to extract values of the *Four Key Metrics*. There are few routines, procedures, and standards that span the entire tech organization, which makes generalization difficult. Another obstructing factor in finding reliable data from a longer period of time was due to Storytel's multiple organizational restructurings during the last year. Some teams are only a few months old and cover new focus areas, and others were previously part of a larger team that was split into different focus areas. These characteristics have resulted in obstacles such as:

- There are no standardized guidelines at Storytel regarding what workflow stages a ticket should pass in Jira
- The usage of the workflow stage 'Waiting For Release' is different.
- Differences in what the workflow stage 'Done' implies in different Jira projects.
- Workflow stages and their meaning for each team, have changed over time
- Differences between teams regarding the size of a ticket in Jira.
- Not possible to give one single estimate that generalizes the entire Tech Department.

When discussing the obstacles related to the Stability Metrics - Mean Time To Restore and Change Fail Rate - the numbers are heavily dependent on defining what constitutes a failure. A failure could be a bug that the customer rarely notices (or believes is a design choice), a bug that actually impacts the user but has a work-around, a bug that impacts the user that does *not* have a workaround, or it could be related to degraded performance, downtime, partial or whole system disruption. Several options were considered but none of the attempts were sufficient in estimating the Mean Time to Restore. The options for estimating Mean Time To Restore are presented below, but the extensive explanations are given in Appendix 3.

- **Option 1:** Measure the lead time in Jira for bugs with priority 'critical' or 'blocker'
- **Option 2:** Measure the lead time of system disruptions logged at status.storytel.com
- **Option 3:** Create a timeline for hotfixes based on information from slack
- **Option 4:** Create a timeline for requests of hotfixes based on information from slack

Option 1 was chosen due to its reasonable simplicity, and to avoid the time-consuming action of reading through and manually evaluating Slack channels for information that would be required for option 3 and 4. This was carried out, but proved to not indicate failures that caused disruption of sufficient size. Option 2 was thereafter tested, but concluded faulty. Option 3 and 4 are still unexplored.

The Change Fail Rate is, similar to the Mean Time To Restore metric, particularly difficult to define as the central question becomes what constitutes a failure. The options considered are based on the same definitions of failure as the Mean Time To Restore metric, but by looking at the relative proportion of failures, rather than the timeline of a failure from occurrence to remediation.

One problem with the Change Fail Rate metric is the idea of whether a whole batch (for example all code related to a specific iOS app release) should be considered to be a 'failure' if there is one serious bug leading to failure among hundreds of issues involved in the batch, or not. The chosen method of calculating the Change Fail Rate from the proportion of bugs to the monthly throughput neglects to consider batch size due to the increased complexities of taking this into account. This was chosen in favor of a simpler calculation that can be performed and compared easily over time, leaving the aspect of batch size to be presented in the Deployment Frequency metric. This decision is supported by the logic that the metrics balance each other and should not be analyzed separately, but joint together.

5.3.2 Preprocessing of Raw System data sets

The earlier mentioned systems and tools were chosen partly based upon the accessible data sets. The different datasets are presented in this section.

Jira has been used for several years, and the data-set from all time use of Jira consists of circa 27 000 issues. Among these, the 5 largest issue categories are displayed in Table 7.

Number of issues	Category
7188	Stories
5849	Tasks
6499	Sub-tasks
4717	Bugs
913	Epics

Table 7. The 5 largest categories of issues in Jira and their respective amount.

However, the entire dataset was not used in every analysis. To overview recent trends, only 2019-2020 were analyzed in depth. Depending on the distribution of the data, outliers were excluded. Furthermore the 85th percentile is the one represented in the following numbers.

For tempo metrics related analyses, all issues apart from epics and bugs were used. Because of constraints of the system causing overload, unnecessary issues were examined and omitted. Therefore, epics (large features equivalent to heavy time consumption) were excluded because

of its ticket size. Furthermore, bugs were excluded. At the Storytel Tech Department bugs are filed because of formality rather than the attempt to solve them as soon as possible, if they have a low priority. As a lot of bugs are placed in the bottom of the backlog for a very long time, including the whole set of bugs could potentially skew the lead time results. After deleting bugs and epics, the final dataset for tempo metrics contained approximately 21 000 issues. For stability metrics, the dataset included only bugs, filtered on a certain priority label which will be further explained later in this section. The stability metric dataset contained 420 issues.

Another pre-processing involved the classification of workflow stages. Since every team and crew choose their workflow themselves, and Jira allows you to design the process from scratch, there are a lot of workflow stages to manage when overviewing all issues created in Jira. Classification was performed, mapping each existing stage into one of the four chosen stages; either 'Backlog', 'To Do', 'In Progress' or 'Done'. This means for example that 'Waiting for Test' or 'Waiting for Release' were lumped together into 'In Progress'. The flexible options became fewer, but it was necessary to be able to manage and operate on the very large dataset.

Further looking into the data, one large disruption was found. In February 2020, a large amount of resolved issues happen at once, resulting in a large gap in the graphs. In Figure 9 an example is shown, where the daily number of work in progress decreased into half (the green line indicates the trend over the blue dots that is the daily WIP number). By looking at the details of the graph we found that issues connected to one team in particular were responsible for the big heap. Questioning the Crew Coach of the team cleared that they had configured its Jira tickets differently, so that they were never classified as "Done". A large heap of finished issues were cleaned up causing the data to become significantly skewed. Since the impact of this data on other graphs are unknown, this team with all its related issues was chosen to be excluded from the dataset.

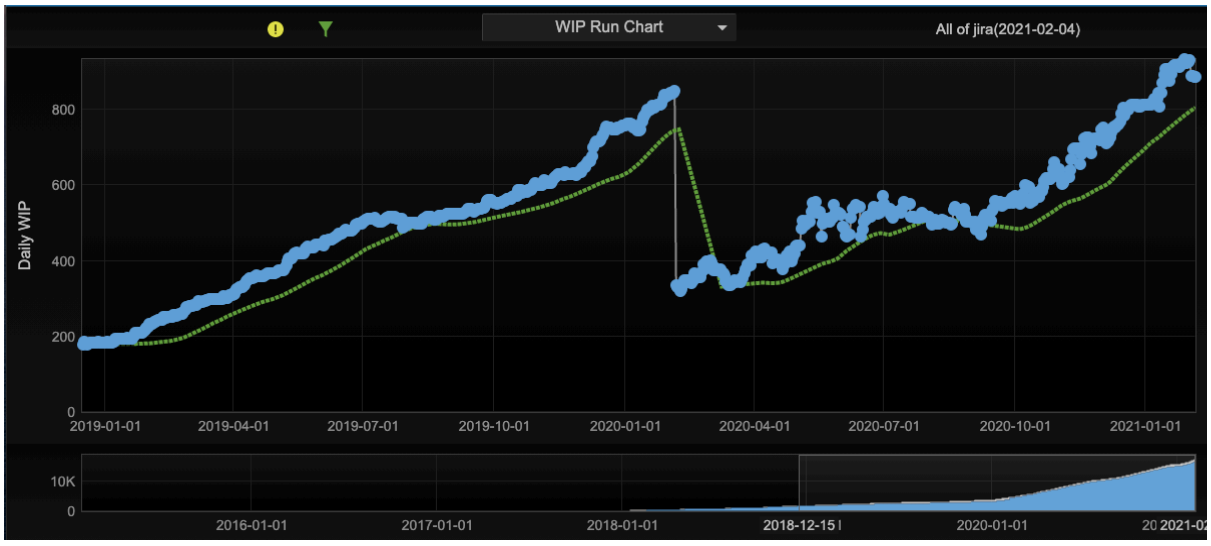

Figure 9. Visualizing the data in a WIP run chart before excluding one team from the data set.

The data from HR surveys was processed by HR before handed over to us, in order to protect sensitive information. The accessed dataset contained questions and answers from the Tech

Department, response rates and some averages. A majority of HR survey questions are answered allowing the respondent to indicate their level of agreement between 1 and 100. These answers are summarized by an average score of agreement in the data. The surveys deemed relevant for this study have been categorized into relevant factors. A table summarizing the included surveys can be viewed in Appendix 4.

5.4 Interviews

The interviews conducted in this thesis were approximately 45 minutes long. With the consent of the interviewee, the interviews were recorded. This was not only because the focus enables listening rather than writing down the answers, but also because it added value through the fact that it was possible to use direct quotation afterwards (Bryman, 2011). A summary of conducted interviews can be found in Table 8.

No.	Role	Date
1	Product Manager	Nov 27, 2020
2	Crew Coach	Dec 1, 2020
3	Developer	Dec 2, 2020
4	Tech Manager	Dec 7, 2020
5	Customer Support	Dec 15, 2020
6	Crew Coach	Jan 8, 2021
7	Developer	Jan 8, 2021
8	Test Lead	Jan 11, 2021
9	HR	Jan 12, 2021
10	Tech Manager	Jan 18, 2021
11	Customer Support	Jan 18, 2021
12	Developer	Jan 25, 2021
13	Crew Coach	Jan 25, 2021
14	Developer	Jan 27, 2021
15	Tech Manager	Feb 1, 2021
16	Tech Manager	Mar 7, 2021

Table 8. An anonymized list of the conducted interviews.

6. Results - *Four Key Metrics*

In this section, the raw system data estimation of the *Four Key Metrics* is used to answer the first research question '*Where does Storytel rank in the software development performance categories based on the Four Key Metrics?*'. In the last paragraphs of this section, *6.4 Comparison of the Four Key Metrics,* the estimated productivity gathered in the survey is set against the raw system data and a comparison is made.

To be able to answer the final research question, '*What bottlenecks exist that hinder Storytel from being a better performer in terms of a higher performance category?'*, it is not enough to estimate the *Four Key Metrics* current situation, there is also a need to understand the context for *why* they are currently performing this way, what the performance level has previously looked like and if there are tendencies that support certain expectations for the future. Furthermore, being able to bring reasonable explanations for why they are categorized in a certain way, increases validity. If no explanations can be found, it is additionally a warning sign in terms of reliability. This kind of estimations are therefore included in this section.

6.1 Tempo Metrics

When discussing the two key metrics that indicate *tempo* - Delivery Lead Time and Deployment Frequency - these vary between the different services and tech stacks at Storytel. The different services are separated when needed in a simplified representation and not the exact reality. The app related services are fairly easy to distinguish from the rest, but the other services (see Section *2.3.1 Services and architecture*) are sometimes presented grouped together as 'non-app related services' - even though there are differences among these as well. Sometimes further distinctions are made, like services on 'the legacy platform', internal and external web tools, and backend. The ambition is to cover most services, but sometimes there are gaps or overlaps. The exact distinctions and corresponding numbers are not the important conclusions here, rather that there exists differences within the tech stacks and services in the Tech Department.

6.1.1 Delivery Lead Time

The Delivery Lead Time at Storytel Tech Department is between *one week and one month,* which means they are a medium performer (see Figure 10). Interestingly, this is one of the metrics that according to the raw system data has changed the most if looking at the historical perspective. During 2019, the lead time is 10 days which indicates Storytel as being a medium performer. However, for 2020 the lead time has almost doubled and is now about 19 days (for 85% of the issues, outliers excluded).

Not only the number of issues have significantly increased during 2020, so has the average cycle time. It has not been possible to separate issues for apps due to insufficient usage of labels, why the numbers includes tickets from all tech stacks. However, the Cycle time of 19 days is almost corresponding to the app release schedule with three week intervals. Regarding the increasing trend from 10 days 2019 to 19 days 2020, the efforts on improving test coverage

could be a potential explanation. The Test Lead role was introduced in the beginning of 2020, and the number of employed testers has almost doubled since the beginning of 2019. During interviews it is conveyed that the ambition to decrease the interval to two weeks is becoming less and less realistic. When internal discussion started whether to increase Deployment Frequency to two weeks, the cycle time was 10 days according to the data, which could explain the optimistic approach. Since then, Delivery Lead Time has adapted to Deployment Frequency for example because standards have been set higher.

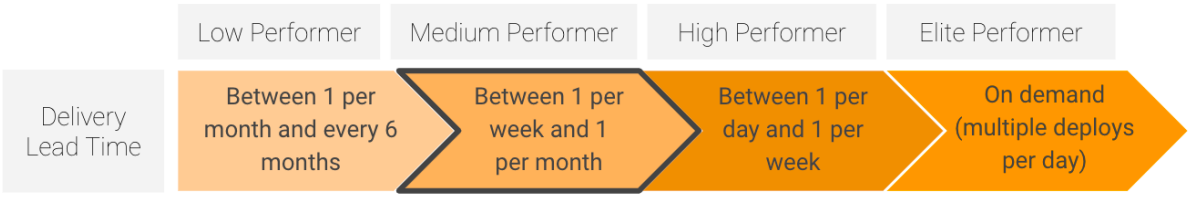

Figure 10. The estimate of the present classification of Delivery Lead Time.

6.1.2 Deployment Frequency

The Deployment Frequency at Storytel Tech department cannot be conclusively generalized. Apps deploy every third week according to schedule, wherefore they are categorized into deploying between *once per week and once per month.* The legacy platform is updated once a week according to schedule, i.e. deploys happen between *once per day and once per week.* Apart from these, other deployments occur more frequently. Backend-related deploys, or web deploys connected to internal or external web tools often happen weekly and in some instances daily. However, this occurs especially when a new feature or tool is under construction, before being launched to users. Therefore, fast deployment is possible since it does not risk to entail any disruption or outage.

In conclusion, some deploys are scheduled and some are not. With this in mind it is reasonable to determine that the Deployment Frequency defines Storytel's Tech Department to be both a medium and a high performer depending on the service (see Figure 11).

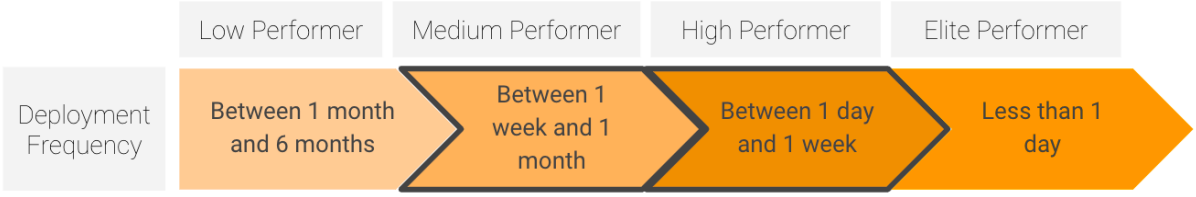

Figure 11. The estimate of the present classification of Deployment Frequency.

The Tech Department's Deployment Frequency has stayed the same for the last couple of years. The apps have deployed every third week since at least January 2019 according to documented release history (Storytel, 2021f). For the legacy platform, the weekly scheduled releases is a newly introduced routine. There are ambitions to increase the routines for other services, and to involve testing time periods for the releases already happening weekly. One improvement

regarding Deployment Frequency is for the unscheduled deployments for web tools that previously affected services crossing over several teams. More coordination was needed in order to not affect other teams' code, as updates could not be performed individually. These dependencies have decreased along with architectural changes into microservice usage *(Interview 7: Developer, 2021)*. The ambition of the company is to use the standard deployment pipeline instead of hotfixes when there is a disruption or bug fixed (Storytel, 2021e).

6.2 Stability Metrics

6.2.1 Mean Time To Restore

When defining what constitutes a failure we deliberated several different approaches. Two attempts to estimate the Mean Time To Restore were carried out, but both were discarded because of uncertainty. The first attempt, measuring the lead time in Jira for bugs with priority 'critical' or 'blocker', ended up being inefficient in terms of impact on users. This could for example be due to the assumption that the majority of bugs in Jira are in production might be incorrect, or that the guidelines for priority labels on bugs are outdated. The second option, measuring the lead time of system disruptions logged at status.storytel.com, was found to not measure the desired type of failure. These attempts and the obtained data can be found in Appendix 5. The remaining options are still unexplored due to time constraints.

Conclusively, no final estimate was retrieved for the Mean Time To Restore (see Figure 12). The raw system data were supposed to complement the survey estimated Mean Time To Restore to create a holistic and nuanced estimation. Fortunately, the survey estimates had rather small variations and this data could be used in the factor analysis.

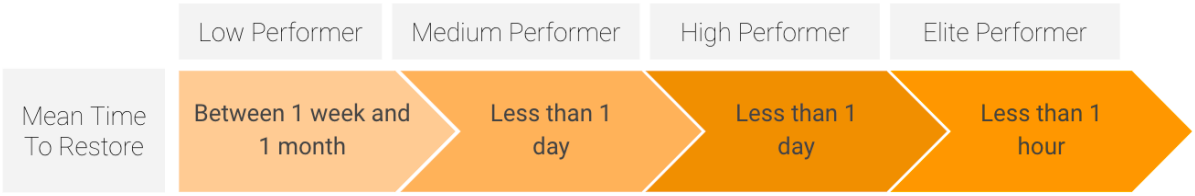

Figure 12. No estimate has been possible to define Mean Time To Restore.

6.2.2 Change Fail Rate

Change Fail Rate, defined as the percentage of deployments made that lead to a degraded service which in turn requires immediate remediation, is another measure indicating quality. As mentioned, at Storytel, deployments rarely lead to service degradation. The large majority of issues that would need immediate action if deployed are caught in the earlier development stages *(Interview 8: Test Lead, 2021; Interview 10: Tech Manager, 2021)*.

The Change Fail Rate is closely related to testing and how many bugs are caught before release to production. If looking into the raw system data, the total number of filed bugs have increased, but so has the total throughput (see details in Appendix 8). For the proportion of critical and blocker bugs among the monthly total throughput of all issues, the increase constitutes for one percentage point between 2019 and 2020 - but still far below the level of 15%. However, it is

important to note that it is not necessarily concluding that a larger proportion of bugs are produced, but that more bugs are found and filed due to improved routines and amount of testers.

As previously concluded, bugs prioritized as critical or blocker are found to not be a proper definition of a failure at Storytel. However, since it is a broader definition of failure (less impactful bugs are included) and this proportion is already below 15%, it can be used as a basis to infer that Storytel is already performing well, having a Change Fail Rate below 15%. Unlike the other metrics, this is a common rate for both Elite, High and Medium performers (see Figure 13).

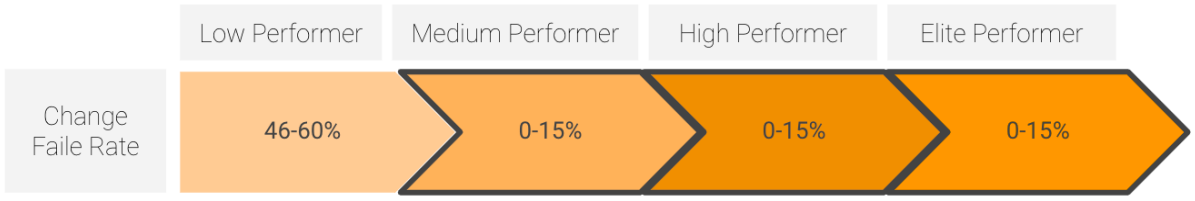

Figure 13. The estimate of the present classification of Change Fail Rate.

6.3 Summary of the *Four Key Metrics* estimation - from raw system data at Storytel

In conclusion, from the found data Storytel is today performing as a medium or high performer (see Figure 14). Notice that this is based on present performance. The historical perspective suggests some changes *within* the categories rather than any major changes between the categories.

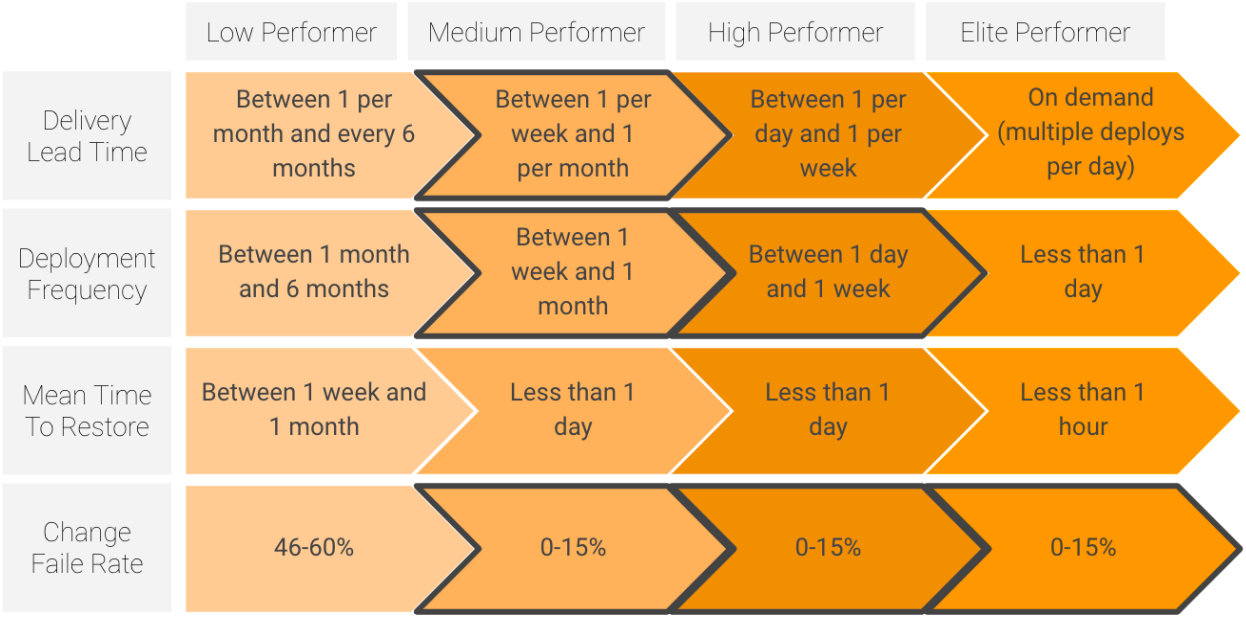

Figure 14. Summary of the Four Key Metrics estimation from raw system data at Storytel

6.4 Comparison of the *Four Key Metrics* - System Data vs Survey Estimates

In the survey, respondents were asked to give an estimate of each of the *Four Key Metrics* based on their team in order to complement the raw system data in several aspects. The main reason for this was to provide the opportunity to analyze how the factors theorized to impact productivity correlate with the *Four Key Metrics* through statistical measures. Additionally, it serves to further contextualize the raw system data. As described in the methodology section, gathering data through a survey can create a more holistic view than what the raw system data can generate (Forsgren, Humble and Kim, 2018). However, during interviews, some employees have indicated that they believe developers will likely contribute with a large spread in estimates, and that some will be far away from reality *(Interview 8: Test Lead, 2021)*. Lastly, it serves to validate the raw system data - if the data provided by the survey shows very different metric values from the raw system data, this would be a strong reason to believe that the raw system data is likely untrustworthy.

From analyzing the survey responses, a relatively large share of respondents opted out of giving an estimate. On the two tempo metric questions about Deployment Frequency and Delivery Lead Time, 9.5 % and 8.2 % respectively responded 'I don't know'. The rate of this response was even higher for the Stability metrics - 16.7% in Mean Time To Restore and 19.4% in Change Fail Rate. These high percentages indicate that there is likely a degree of uncertainty and guesswork from a number of respondents who chose other options than 'I don't know'. This emphasizes that the survey estimates of the *Four Key Metrics* should be treated as biased estimates, and not as fact.

6.4.1 Delivery Lead Time

The survey data on Delivery Lead Time partly contradicts the raw system data which classifies Storytel as a medium performer. The majority of survey respondents (34.2%) estimate that their team's Delivery Lead Time is between one day to one week, which classifies them as a 'high' performer. The second most frequent answer (27.4%) is one week to one month, which should be the majority answer in accordance with Jira data (see Figure 15). When accumulating the share of respondents that estimated that their average Delivery Lead Time is shorter than one week (ranging from 'Less than one hour' and 'one day to one week'), the result is that 61.1 % of respondents classify their team's performance in the 'high' to 'elite' range according to the classification of the *Four Key Metrics*.

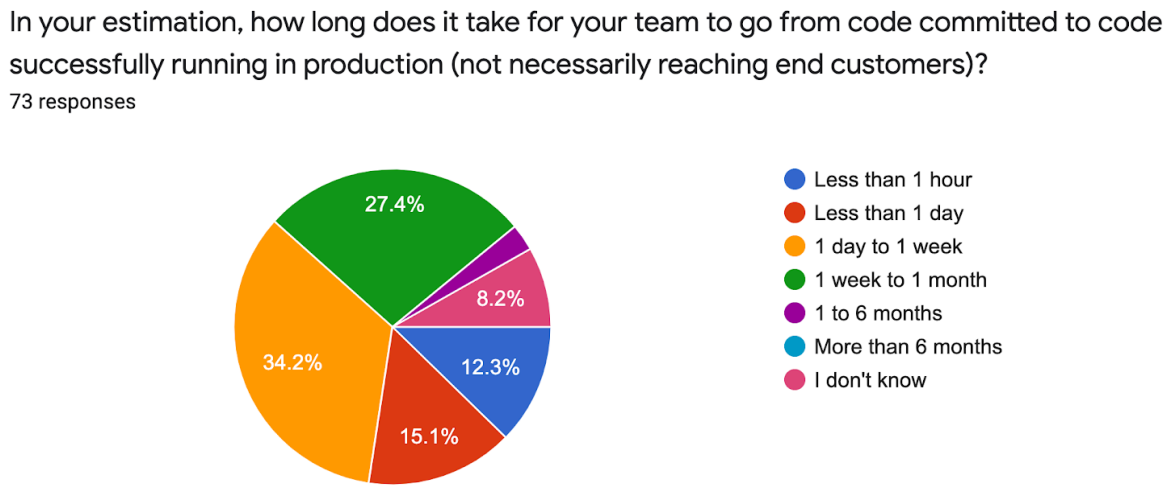

Figure 15. Survey responses for Delivery Lead Time.

There are several possible reasons for this discrepancy. It might be that respondents tend to underestimate how long their teammates spend on each task on average, and mainly respond based on their individual experience because the respondent's focus is only the part they are mainly responsible for in the development cycle. This is interesting, as it either shows lack of team perspective and bad awareness of how well the team is doing - or lack of communication and that the state of the teams is not properly signaled. It could also be both of them. Similarly, how the respondent interprets the wording 'code successfully running in production' and how this differentiates to the workflow stages included in the Jira analysis may affect the results. For example, an iOS developer might interpret code and corresponding Jira ticket as 'successfully running in production' when their code is added to the version that will be deployed to the App Store on the next scheduled release - whereas the Delivery Lead Time extracted from the Jira data may also include the time in which the ticket that reflects that iOS code is finished but is still waiting for the next scheduled App release. This is due to differences in what workflow stages different teams use, and how they define 'Done'. Lastly, it might be that results are skewed due to the fact that only half of the Tech department responded to the survey.

Some discrepancies are expected, since Delivery Lead Time differs among the tech stacks. From the survey data, for example among developers, the Backend and Frontend Developers estimate a shorter Delivery Lead Time than Android developers corresponding to expectations. Since Backend Developers are the majority role both in the Tech Department in total and among the respondents, this will affect the average estimates of Delivery Lead Time (both in survey and raw system data collection). However, apart from Android developers, the overrepresented roles among the longest estimated Delivery Lead Times are Crew Coaches, project managers and testers. Arguably, these roles might have a better overview over the whole process than the developers.

In conclusion, based on the survey data there are indications that Storytel has a lower score (shorter time) in the Delivery Lead Time metric than what is found by the raw system data. A majority of respondents indicate that the average Delivery Lead Time for their team is less than one week, which would classify Storytel as a 'high' performer.

6.4.2 Deployment Frequency

The raw system data regarding Deployment Frequency, indicating that Storytel deploys between once per day and once per month and categorizing them between a 'medium' and 'high' performer, is overall corroborated by the survey results (by 75.7 % of respondents). 44.6 % respond that they deploy between once per day and once per month and 31.1% respond between once per day and once per week (see Figure 16).

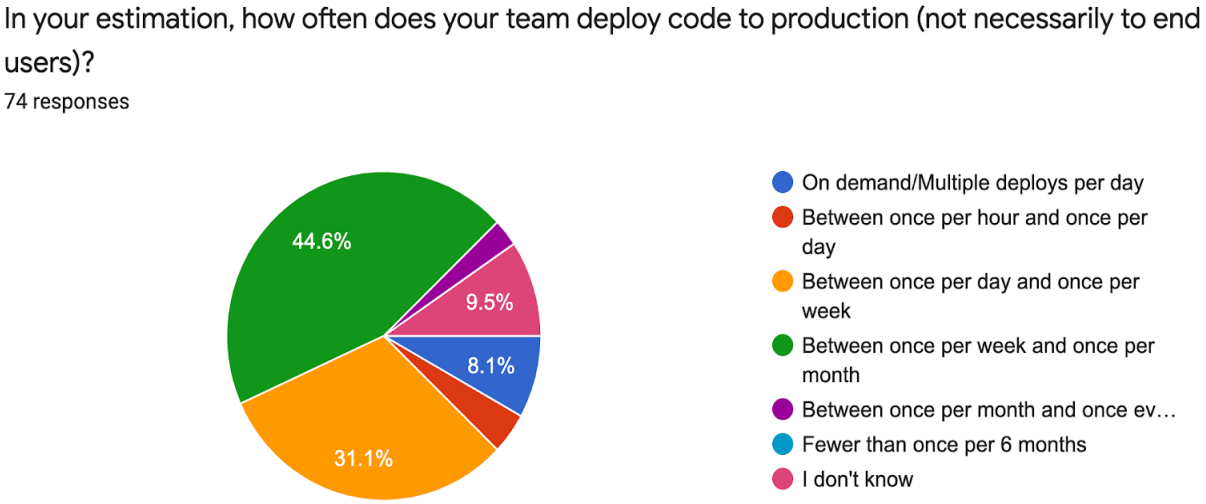

Figure 16. Survey responses for Deployment Frequency.

The third largest share (8.1%) - when excluding those who responded 'I don't know' - indicate that their team deploys multiple times per day, which would categorize their team as an 'elite' performer. As previously discussed, the differences in Deployment Frequencies at Storytel is largely due to the differing procedures between teams developing the iOS and Android apps, and teams involved in developing other Storytel services. It might be additionally due to other influencing factors, but these are difficult to separate from the aforementioned differences.

6.4.3 Mean Time To Restore

A large majority (50 %) of respondents, estimate that the Change Fail Rate of their team is less than one day (see Figure 17). Unfortunately, these numbers cannot be corroborated by the raw system data findings due to a gap in the findings. The survey data however classifies Storytel as a high/medium performer. Furthermore, 20.8 % of respondents estimate that issues are resolved in less than an hour value, which would indicate that those respondents belong to

teams that can be classified as ‘elite’ performers. However, these estimates are in a minority, and Storytel is classified as a high/medium performer.

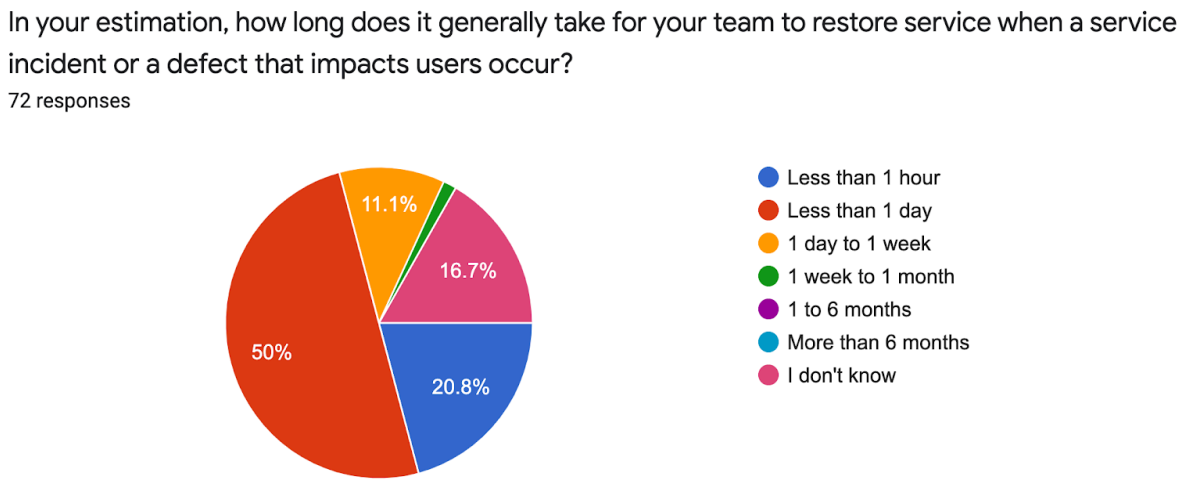

Figure 17. Survey responses for Mean Time To Restore.

6.4.4 Change Fail Rate

A large majority of respondents in the survey (68.1 %) estimate that the percentage of deployed changes that result in a degraded service and require immediate remediation is between 0-15 % (see Figure 18). This is in accordance with the raw system data, and verifies that Storytel is not a ‘low’ performer in this metric.

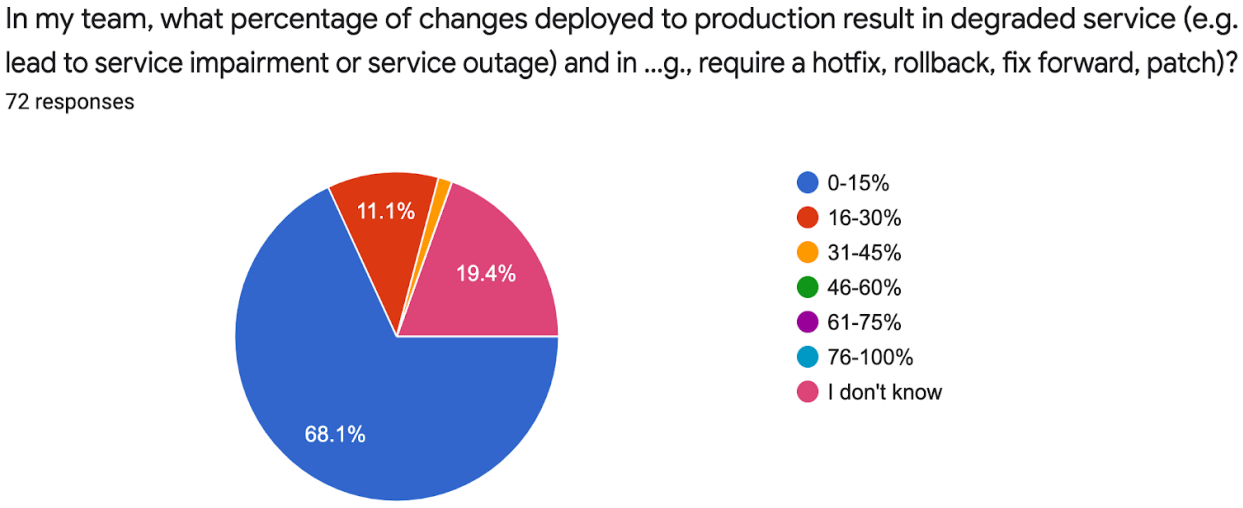

Figure 18. Survey responses for Change Fail Rate.

7. Results - *Factors*

This chapter presents the results of the factor analysis and its correlation with the *Four Key Metrics* made of the survey responses in SPSS. Ultimately, it answers the second research question of this thesis '*What human and technical organisational factors have an impact on Storytel's software development performance?'.*

7.1 Results from Factor Analysis

The survey was constructed based on content validated factors with the help of a literature review. The factor analysis has been performed to evaluate construct validity (see *4.3.2 Validity and Reliability*). To interpret the factors, the contents of the items included in the factor are scanned to identify common themes excluding those deemed unreliable (Factor 7 and Factor 10). Overall, the factors the survey intended to measure seem to be reflected to a high degree in the factor analysis. Factors 1-5 and 8-9 mainly consist of variables from the same theoretical constructs, see Table 5. The same final factors from the factor analysis, but with respective thematic interpretations, are shown in Appendix 9.

Factor 1 contains two items from the construct Team Identity and one item from the construct 'Communication'. This seems reasonable as T1 and T2 are both centered around phenomena where good internal communication can be assumed to be a prerequisite (collectively working toward a shared goal and knowing the reason for features developed in the team, respectively). The theme of the factor is interpreted as mainly representing the construct Team Identity, and named accordingly. Factor 2, 3 and 4 loaded items corresponding to only one construct each, and were therefore named 'Number of Projects', 'Transformational Leadership' and 'Generative Culture' respectively. Factor 5 contains the two items that are meant to measure job satisfaction in the questionnaire, and one item that belongs to the construct 'Team Identity' - the item 'I am proud of being part of my team' (TI3). Factor 5 is still named 'Job Satisfaction', since TI3 can intuitively be connected to job satisfaction.

Factor 6 contains items from four different theoretical constructs. This can be interpreted as a new factor that we did not intend to measure, but that these items might reveal an undiscovered construct and an interpretation is therefore necessary. 'In my team, responsibilities are shared' (GC2) correlated much more strongly with the variables in Factor 6 than with the rest of the 'Generative Culture' variables in Factor 4. The item TC1 'In my team, we put effort into facilitating work for other teams.' can furthermore be linked to GC2 through the common theme of shared responsibility. AU1 and AU2 are both indications of automation in the development process, in deployment and testing respectively. Finally, LM2 'In my team the ambition is to keep the number of Work-In-Progress to a minimum' - connected to the theoretical construct 'Lean Management' - is theorized to be an indication of an efficient development process. The rest of the items in Factor 6 can be interpreted connected to aim towards efficiency as well, why Factor 6 subsequently is named 'Efficiency (Automation and Shared Responsibility)'.

Factor 8 and 9 both only contain items from one theoretical construct each and were consequently named ‘E-Factor and Time Fragmentation’ and ‘Team Cohesion’. In Factor 9 ‘Team Cohesion’, the two included items have opposite loading; TC2 (‘I have a good insight into what other teams are doing.’) is negatively correlated with TC2 (‘I wish I had more insight into what other teams are doing.’). This indicates that respondents that do not already experience that they have good insight into other teams, generally wish that they had better insight.

Some constructs that the questionnaire aimed to measure - Communication, Automation, Architecture, and Lean Management - do not show up clearly in the factor analysis. While Communication, Automation and Lean Management items partly loaded onto other reliable factors, Architecture is not included anywhere. This may be due to reasons such as an insufficient number of questions to measure these constructs, poorly phrased questions or other design flaws. It could also be interpreted as support that this might not be a prevalent factor. In any case, we can conclude that no support exists from the factor analysis that this construct has an impact on the respondent's answers.

Conclusively, no factors from the factor analysis are connected purely to constructs belonging to the Process category. These constructs were only measured by six items out of a total of 32 items, and only two items per theoretical construct, so this is likely primarily caused by the survey design. As a result, no reliable conclusions can be drawn on regarding how this category correlates with productivity more or less than the other categories.

These factors represent the most prevalent underlying aspects that explain the majority of the variance in the survey data.

7.1.1 Factors influencing the Four Key Metrics

To analyze what factors influence the *Four Key Metrics* at Storytel, the factors generated from the factor analysis are used. As section 6.4 (‘Comparison of the *Four Key Metrics* - Raw System Data and Survey Estimates) supports, the estimated values of the *Four Key Metrics* in the survey are sufficiently similar to those gathered from the raw system data that they can be treated as a proxy for the raw system data. The survey data has provided a more holistic view than what the raw system data would have contributed to. Subsequently, correlations between these estimates and the factors validated by the factor analysis can generate valuable insight.

New factor variables with the corresponding factor scores for each respondent were created automatically in Principal Axis Factoring using the Regression method. Regression was chosen as this procedure maximizes the validity of the estimates (Distefano, Zhu and Mîndrilã, 2009). The new variables were used to analyze potential correlations between the factors and the respondents’ estimates of the *Four Key Metrics*. The correlations were analyzed using Spearman’s rank correlation coefficient as neither the factor scores or the *Four Key Metrics* estimates are normally distributed. The significant correlations are displayed in Table 9. The smaller the p-value is, the stronger the evidence for the correlation is (Glen, 2021).

Four Key Metric	Correlated factors	Spearman Significance
Delivery Lead Time	Factor 4 - Generative Culture	r_s = - 0.409**, p = 0.004
Deployment Frequency	Factor 6 - Efficiency (Automation and Shared Responsibility)	r_s = - 0.295*, p = 0.042
Mean Time To Restore	Factor 2 - Number of Projects	r_s = - 0.359*, p = 0.018
Change Fail Rate	-	-

Table 9. Factors correlated to the Four Key Metrics (Regression method).
** p-value less than 0.05*
*** p-value less than 0.01*

Surprisingly, among the 8 valid factors extracted in the factor analysis, only three significant correlations were found between factors and the *Four Key Metrics*. This may be caused by previously discussed flaws in the survey design (see Section *5.2 Factor Analysis*), and it is likely more factors that would have shown statistical correlations if there had been more items covering them in the survey. However, only these three correlations can be conclusively confirmed to have an impact on the *Four Key Metrics* at Storytel, based on our survey.

7.2 Investigating the correlated factors

In this section, results gathered from our survey, interviews, and raw system data regarding factors influencing productivity at Storytel are presented to support the results from the factor analysis correlation with the *Four Key Metrics.*

7.2.1 Generative Culture impact on Delivery Lead Time

The factor 'Generative Culture' shows a negative correlation with the metric Delivery Lead Time (r_s = - 0.409**, p = 0.004). This indicates that shorter delivery lead times within the Tech Department are correlated with a higher degree of generative culture, i.e. a culture that optimizes information flow (see items in Table 10).

Factor 4: *Generative Culture*
GC5 - In my team, failure causes inquiry so that we can learn from the experience. (-.775)
GC4 - In my team, cross-functional collaboration is encouraged and rewarded. (-.480)
GC6 - In my team, new ideas are welcome. (-.451)
GC1 - In my team, information is actively sought. (-.443)

Table 10. Items included in Factor 4 with corresponding factor loadings.

In our survey, the average score of the four items included in this factor indicates that Storytel's Tech Department can be classified as having a generative culture overall. All items have a mean value above 4 out of 5 (average score 4.26), suggesting the average score is between 'Agree' and 'Strongly Agree'. This indicates that the concentration in the organization is on the mission, rather than on positions and individual people. Furthermore, this means they are a performance-oriented organization within which risks are shared, novelty is implemented, and cooperation is encouraged (Westrum, 2004). Notably, these organizational qualities are reflected in one of Storytel's mission statements regarding employees, which states; 'Attract and develop exceptional people by nourishing a diverse workplace built on *trust, innovation,* and *collaboration*'.

Findings from HR survey data further corroborate the findings that the culture at Storyel's Tech Department is generative. According to Westrum (2004), a generative culture needs to promote meaningful work and clarity to generate high-performing teams. HR survey data from 2020 shows that Storytel's mission and vision statements are well-communicated and found to be inspiring. Additionally, both of these have increased slightly since 2019. These are arguably important components in promoting meaningful work (*Vision & Mission* - HR Survey Data, 2019-2020). In another survey from 2019, employees overall strongly agree that their work feels meaningful and that they view it as 'creating change for the better' (*Team Efficiency Tech* - HR Survey Data, 2019).

The item GC5 - 'In my team, failure causes inquiry so that we can learn from the experience' has a significantly larger factor loading (-.775) than the other items (the second largest factor loading is GC4 with -.480); and subsequently can be inferred to have the largest degree of correlation with Delivery Lead Time. Due to this, we looked for more information regarding the Storytels' Tech Department's ability to learn from previous failures.

The capacity to learn from failures or mistakes within an organization can be connected to how feedback is handled within the organization. From two HR surveys during 2019 about leadership, we found that *feedback* is among the top-rated skills of which employees agree managers could improve (Leadership at Storytel - HR Survey Data, 2019). Supporting this finding, from 'Transformational Leadership' in our survey one item directly connected to feedback (TL3 - 'My manager regularly gives actionable feedback that helps me improve my performance.') received the lowest average score (3.2/5) out of the three items. Looking into feedback in more detail, the numbers from an HR survey from 2020 tells us that feedback is given to colleagues more often than to managers, but that employees are equally comfortable giving it to both. However, the amount of feedback received from managers is generally less than what employees desire (Feedback - HR Survey Data, 2020).

One initiative to learn from mistakes within app releases at the Tech Department is a meeting happening one week after each app release. The meeting aims to evaluate and discuss pros and cons with the recent release and to document improvement areas and learnings *(Interview 10: Tech Manager, 2021)*. For large issues or disruptions, there is an additional procedure implemented in which a document called a '*Post Mortem*' is filled out, typically by a senior

developer or a Tech Manager. This is performed to map specifically what went wrong, why it happened, and what was done in order to prevent the same issue from occuring in the future. (*Interview 8: Test Lead, 2021)*.

While 'Generative Culture' is the factor that is found to have the largest impact on Delivery Lead Time, and Delivery Lead Time is the metric that changed the most between 2019 and 2020, from 10 days to 19 days - none of the findings indicate that Storytel's culture has changed significantly during this period. While there are likely several components contributing to the increase in Delivery Lead Time metric since 2019, the lack of a well-functioning feedback culture might be a growing source of error as the Tech Department increases in size. Lastly, the need of more reflective and retrospective procedures might grow bigger.

7.2.2 Efficiency, Automation and Shared Responsibility impact on Deployment Frequency

The factor 'Efficiency (Automation and Shared Responsibility)' shows a negative correlation with the metric Deployment Frequency (r_s = - 0.295*, p = 0.042). This suggests that the items included - spanning the theoretical constructs team cohesion, automation and lean management - are indicated to have a combined impact on the frequency with which the Tech Department deploys code to production. High scores on these items are connected to more frequent deployments (see items in Table 11).

Factor 6: *Efficiency* (*Automation and Shared Responsibility)*
TC1 - In my team we put effort into facilitating work for other teams. (.759)
AU1 - What is, in your estimate, the percentage of tasks related to deployment that are automated in your team? (.649)
AU2 - What is, in your estimate, the percentage of tasks related to testing that are automated in your team? (.529)
GC2 - In my team, responsibilities are shared. (.519)
LM2 - In my team the ambition is to keep the number of WIP to a minimum. (.376)

Table 11. Items included in Factor 6 with corresponding factor loadings.

Both of the items that were intended to measure Automation (AU1 - 'What is, in your estimate, the percentage of tasks related to deployment that are automated in your team?' and AU2 - 'What is, in your estimate, the percentage of tasks related to testing that are automated in your team?) are included in the factor that is found to have an impact on Deployment Frequency. From interview material, this is not an unexpected finding.

According to several employees, the largest obstacle in reducing the interval between deployments is the manual and time-consuming regression tests required before each release. *(Interview 8: Test Lead, 2021; Interview 10: Tech Manager, 2021)*. Data logged from the app release processes indicates how time consuming the release-related activities are in the test

pipeline. During the first nine months of 2020, testers collectively spent approximately ten times as many hours on release-related activities compared to iOS and Android developers (Storytel, 2021d). These numbers support that manual regression testing is by far the most time consuming activity, interfering with the initiative to decrease the release interval. Conclusively, a lot of time is spent by the testers to assure quality during freeze time (when new code is no longer pushed, and most testing activities are carried out). If *test automation* increased, Deployment Frequency could increase.

Furthermore, *deployment automation* enables more reliable and risk-free deployment to production (DevOps Research and Assessment, 2021). Interviewees emphasize the benefits of introducing Deployment Monitoring and tools to assist troubleshooting which is now a time-consuming activity that hinders developers, and especially Backend Developers, from deploying often and in certain situations *(Interview 15: Tech Manager 2021; Interview 14: Developer 2021).*

Another item included in this factor is LM2 - 'In my team the ambition is to keep the number of WIP to a minimum' (factor loading: .376) which indicates that teams that implement this practice at the Tech Department, in combination with the other factor items, generally have an higher Deployment Frequency than other teams. Deployment Frequency is used as a proxy measurement of batch size, and limiting WIP is connected to reducing batch sizes. This subsequently enables faster cycle times and exposes potential obstacles to the flow of work (Forsgren, Humble and Kim, 2018). However, according to Forsgren, Humble and Kim (2018), this factor is only observed to have an impact on software delivery performance when it is combined with a team's use of visual displays of productivity and quality metrics. Interestingly, from Jira data it was found that teams' WIP does not correlate with lower cycle times (see Appendix 6) and visualization on work flow is expressed by our survey respondents as lacking. Based on these findings, further efforts spent on visualizing metrics in teams where the practice of limiting the number of WIP is implemented, could potentially increase process improvement and throughput at Storytel even more.

Item GC2 - 'In my team, responsibilities are shared' is also included in this factor, and consequently indicated to have an impact on Deployment Frequency. A team with a strong *sense of identity* is signified by a joint feeling of ownership (Demarco and Lister, 1987), which can be assumed to promote responsibility sharing within the team. In the three questions in our survey that represent 'Team Identity', the average score is high (4.36/5), indicating that most employees at the Tech Department feel a strong sense of team identity. This is further supported by the HR Survey Data. Employees answers regarding having trust in that their teammates follow through on their respective tasks are very highly scored. However, questions regarding responsibility distribution, communication about delays, and visibility into projects within the team are scored significantly lower (Team Efficiency - HR Survey Data, 2019). Sharing responsibilities within teams when it comes to testing is important for possibilities to increase Deployment Frequency. Storytel's goal is not to completely eliminate manual testing in favor of automation or outsourcing, as it adds value when developers are forced to look at the code, add several sets of eyes to a problem and spread knowledge through testing *(Interview 8: Test*

Lead, 2021). Additionally, if all team members are involved in testing and facilitate the tester's responsibility for test coverage, the team can be more efficient. In conclusion, working on responsibility sharing and communication within the teams could further drive software delivery performance.

Finally, item TC1 - 'In my team we put effort into facilitating work for other teams' is correlated with higher values of the metric Deployment Frequency. Being able to facilitate work for other teams - for example by sharing experiences, having a shared code base and assisting in solving problems for other teams - is dependent on having insight into what other teams are doing. Theory suggests that insight into what other teams are working on and corresponding transparency into one's own team can promote cooperation, information flow and cohesiveness between different teams within the organization, and in turn promote organizational performance (Westrum, 2004).

The Monthly Tech Meetings used to be an informative session to increase insights in other teams, but along with the Tech Department's growth over the last year, the Clubs (knowledge sharing area existing between teams for example the Test club, the iOS club and the UX club) has become the most important platform to enable insights across team boundaries (*Interview 3: Developer, 2020)*. From seven HR Surveys from 2019, employees score a question regarding if they think that they have enough insight into what other teams are doing comparatively low (Tech work scope retro - HR Survey Data, 2019). Data from our survey further supports a comparatively low insight in other teams as item TC3 - 'I wish I had more insight into what other teams are doing.' has a relatively high average score (3.96). Additionally, interview data suggests that employees' level of insight into other teams have significantly decreased as the organization has grown *(Interview 2: Crew Coach, 2020; Interview 3: Developer, 2020).* For example, there are indications that general information and posts in Slack channels or other communication tools is not reaching everyone in the department *(Interview 13: Crew Coach, 2021).* This aspect, in its connection with enabling teams to facilitate work for other teams, may have a negative impact on efforts to increase the Deployment Frequency in the Tech Department.

7.2.3 Number of Projects impact on Mean Time To Restore

The factor 'Number of Projects' shows a negative correlation with the metric Mean Time To Restore (r_s = - 0.359*, p = 0.018). This indicates that respondents involved in a larger number of projects generally report a lower average Mean Time To Restore within their team (see items in Table 12).

Factor 2: ***Number of Projects***
NP2 - How many projects have you worked on during the last three months? (.911)
NP3 - How many projects have your team been involved in during the last three months? (.654)

Table 12. Items included in Factor 2 with corresponding factor loadings.

Initially, we found this finding to be counter-intuitive as, according to Demarco and Lister (1987), one of the main obstacles for efficiency and productivity is *time fragmentation;* which they describe as a consequence of when employees are involved in too many projects at once. However, as Meyer (2014) suggests, task-, activity-, and context switches all have different impacts on productivity, and can be of both positive and negative character for the individual as well as for the team. As supported by our findings, task switches might be of a positive character in relation to Mean Time To Restore.

To further investigate the effect of time fragmentation at Storytel, we can analyze the E-Factor, claimed by Demarco and Lister (1987) to be an indicator of a productive work environment in which employees are allowed a sufficient amount of uninterrupted hours. Based on our survey, the E-Factor is calculated to be 46% (standard deviation 22 %) at the Storytel Tech Department. This percentage is above the suggested benchmark of 40%, which indicates that the average Tech Department employee works in an environment that allows them to get into an uninterrupted flow and that consequently promotes effectiveness and reduced frustration. However, the relatively large standard deviation of this item indicates that the E-Factor differs a lot between employees.

We have an additional reasoning that supports the correlation between Mean Time To Restore and Number of Projects that the restoring process is likely facilitated if the involved employees have recent experience of working on the affected code, which is more probable if they are involved in many projects. Employees involved in several projects at once have good insight into the code that they are currently working on, and can subsequently help locate the problem and find the solution more quickly if the issue is in one of these projects. Furthermore, they can prioritize solving issues easier than developers working on only one project, as they have a more holistic view. However, we have not had the opportunity to investigate this reasoning.

7.2.4 No factors impact Change Fail Rate

No factors are found to significantly correlate with the Change Fail Rate metric. Based on the distribution of survey answers, this is likely due to the design of the question and its options. Excluding respondents that opted out of giving an estimate by responding ‘I don’t know’ (almost 20%), 84% of employees report that the estimated Change Fail Rate in their team is between 0-15% which means that there is a very low degree of variance in the answers. Arguably, within the range of 0-15%, there are likely several factors impacting the Change Fail Rate across the Tech Department. Consequently, we are left to analyze material from interviews when it comes to what factors might have an impact on this metric.

Based on one interview in particular, there is an idea that Change Fail Rate is kept low because of the commitment and skill of people at Storytel’s Tech Department rather than a fail-safe process *(Interview 8: Test Lead, 2021).* This means that people feel a willingness to produce quality and responsibility for a well-functioning service, and that this keeps the Change Fail Rate low. The underlying reasons for this can be high job satisfaction, good mood and meaningfulness among other things. Engaged employees that bring the best of themselves to

work produce better work results. If job satisfaction is high, people tend to be more engaged and driven to deliver good work, which consequently results in a higher software delivery performance. Job satisfaction is further correlated with employees feeling that their work is meaningful (Forsgren, Humble and Kim, 2018).

First of all, both of the questions intended to measure job satisfaction in our survey scored highly (The items JS1 - 'I would recommend my organization as a place to work.' and JS2 - 'I would recommend my team as a place to work.' scored 4.64/5 and 4.59/5 respectively). From HR surveys the eNPS score was calculated twice during 2020, equal to 42 in the first quarter of 2020 and an increase to 49 in the last quarter. This reflects a really high and increasing Job Satisfaction at Storytels Tech Department considering that 50 is a benchmark for 'excellence' (Madhavan, 2019). Furthermore, respondents reported a very high score to a question about whether they would re-apply for their current job (Job Satisfaction - HR Survey Data, 2020). Additionally, transformational leadership also has a significant effect on job satisfaction and organizational commitment (Ali, Farid and Ibrarulla, 2016) and according to our survey, this style of leadership is indicated to be prevalent at Storytel. The questions measuring the transformational leadership of managers (items TL1, TL2 and TL3; see Appendix 10) scored an average of 3.6/5. Hence, transformational leaders might also be an underlying reason for committed, satisfied employees at the Tech Department, and subsequently a contributing factor to a low Change Fail Rate.

Secondly, according to an interviewee, mood and attitude is suggested to be of large importance for both individual and team performance at Storytel *(Interview 15: Tech Manager, 2021).* While our expectations were that people would report generally feeling 'worse' at work during the ongoing pandemic and that loneliness from remote work would affect the mood - the weekly HR question 'How was your week at work?' score, assumed to infer the general mood and wellness of employees, is not significantly lower during 2020 than 2019 (Weekly Tech Question - HR Survey Data, 2019-2020). Lastly, from HR survey data it is confirmed that work feels meaningful and in line with skills and interest at Storytel's Tech Department. An interesting hypothesis mentioned during interviews as to why recruiting to the Tech Department at Storytel is easy is that the literature industry, compared to a lot of other tech companies, is appealing when it comes to 'doing something good'. This is supported by a high score to the question 'Do you see your work as creating change for the better?' (Team Identity - HR Survey Data, 2020). Conclusively, levels of job satisfaction, general mood and intrinsic meaningfulness are high at the Tech Department, which could influence employees to feel responsible and engaged to keep the Change Fail Rate low.

8. Results - *Bottlenecks*

To answer the third research question *'What bottlenecks exist that hinder Storytel from being a better performer?'*, we have in this section analyzed interview and survey data. The impact of the described bottlenecks on the *Four Key Metrics* is not statistically confirmed, but they add context to what hinders Storytel from being a better performer in terms of software delivery performance. The bottlenecks - defined as limiting resources equal to or less than the demand placed upon them in a system - are described according to their effect on tempo metrics versus stability metrics respectively, as well as a summary of those highlighted in the survey data.

8.1 Tempo metrics

We found that Storytel is currently a medium to high performer in Deployment Frequency, depending on the service. There are several potential bottlenecks connected to this metric, and in particular related to the app release process, in which they are decidedly a medium performer with a three week deployment cycle. During 2020 and going into 2021, the Storytel Tech Department has maintained an ambition to reduce the time between deployments and adapt to a two week release cycle for the apps instead of three - but this aim is constrained by several factors.

Along with the market growth, there is a growing amount of tasks related to each new app release that employees from the Marketing and Communications departments' need to finish, which becomes more time-constrained with a reduced release cycle. Another technical time-sensitive obstacle is the time needed to receive valuable feedback from users about bugs and other problems to be able to incorporate it into the next release, as well as the minimum time required for the Android version to be in their beta program for it to be of value. Additionally, the app release cycle is further limited by the review processes of the AppStore and the Google Play Store, which generally take up to a few days *(Interview 10: Tech Manager, 2021)*.

Due to the aforementioned reasons, there are differing opinions on whether changing the Deployment Frequency for the apps from three weeks to two is desirable. It would require a joint effort from a lot of different employees in diverse roles to drastically change the current process in order to reach this goal. One interviewee mentions that the present Deployment Frequency may be a local optima for Storytel *(Interview 10: Tech Manager, 2021)*.

From the raw system data it was found that the Delivery Lead Time has increased from 10 days to 19 days from 2019 to 2020. Arguably, this change during 2020 has additionally counter-acted the ambition to increase the Deployment Frequency to releasing every second week. One interviewee expresses that although things are generally more time-consuming now than they used to be in terms of producing and deploying features, the quality of the features and the process have likely improved *(Interview 13: Crew Coach, 2021)*, which suggests that it has become increasingly difficult to compress the current time frame further without compromise. The opportunities for Storytel to become a better performer within app-related Deployment Frequency are inhibited by these bottlenecks.

Currently, not restricted to the app-related services, the architectural infrastructure at Storytel is an additional bottleneck to improving Deployment Frequency and Delivery Lead Time. The Tech Department has over the past few years been going through a platform migration from their old legacy platform to the Google Cloud Platform, in order to, among other reasons, transfer to using microservices and improve scalability and modularity *(Interview 7: Developer, 2021)*. The growth of the organization and rapidly increasing number of employees made it eventually impossible to work all together in one single code base, and prompted the need for microservices.

Before introducing microservices, when teams were working on the same code bases to a larger extent, a release from one team led to an update for every other teams' service as well. Before releasing, some heads-up and checkups had to be performed in order to avoid implicating the work of another team. Since not too long ago, this is not the required workflow anymore *(Interview 6: Crew Coach, 2021).* This is mainly due to the increasingly microservices-based architecture which enables reduced team dependencies, and can potentially subsequently allow for both increased Deployment Frequencies and shorter Delivery Lead Times by making the release process less laborious. It would additionally broaden opportunities for increased automation in some aspects (monitoring and modular testing) *(Interview 8: Test Lead, 2021; Interview 14: Developer, 2021)*. Decisions on transferring to microservices have mainly been decided in clubs and teams and this is how it will continue for now *(Interview 7: Developer, 2021).* However, there is a newly formed team at Storytel that is now mainly responsible for completing the migration from the legacy platform to the Google Cloud Platform. The upcoming and further architectural improvements can continue to provide opportunities for better performances in Deployment Frequency and Delivery Lead Time.

8.2 Stability metrics

The amount of testing that is possible to automate, but is still largely being carried out manually, is a bottleneck - especially in the app test pipeline *(Interview 14: Developer, 2021)*. For improvement in the Change Fail Rate metric and increase in throughput, the testing process overall is a crucial phase in which improvements could be made. Testing and finding bugs is a shared responsibility, however guided by the test club. Supportive in finding bugs are also the users, to whom it is preferable to have a good communication channel. The routines and strategies of communication channels from users differ among the services.

For the apps' communication channel with its users, there is a customer service function at Storytel that apart from helping customers - serve as a feedback collector together with the app ratings. The customer service functions are working in different local teams with one global customer support team facilitating coordination, where one person is responsible for the main communication with the Tech Department *(Interview 5: Customer Support, 2020).* A weekly meeting called the Bug Refinement Session was introduced a few years ago, which have become useful in terms of prioritizing the bugs that are noticed by the end users - which without

the help of Customer Support is harder to figure out *(Interview 8: Test Lead, 2021)*. The meeting has probably improved Mean Time To Restore for bugs in general, but not for the most urgent bugs causing a disruption or large impact on users. These are communicated in Slack channels; however among other types of less prioritized issues, which arguably might diminish their visibility to developers that need to personally keep track of these support channels to quickly be alerted to fix urgent issues related to their code. If urgent issues happen outside of office hours, an on-call club is notified (*Interview 16: Tech Manager, 2021)*.

For internal and external web tools with employees and creators as users, there is no Customer Support function. Several Slack channels are used to get in contact with each team, different channels depending on the urgency of the support request. An improved cooperation process with Customer Support for apps, and introducing more coordination among support errands within internal and external web tools, could potentially scale down the size of this bottleneck and increase the Mean Time To Restore in terms of reducing the time from an issue being discovered until it is fixed.

Again, for some non-scheduled deployments there are plans to introduce routines to for example include scheduled time for testing. If routines for testing and monitoring are in place to catch problems before they are released into production, Change Fail rate could decrease. Incorporated in these plans is furthermore to add versioning. Version control is connected to quality assurance and testing, and could possibly decrease Mean Time to Recover, since finding issues is facilitated.

8.3 Bottlenecks - data from our survey

In an open ended question in our survey phrased *'In your opinion, what is the biggest bottleneck slowing down the work in the Tech Department?'*, 50 respondents shared their reflections. Interestingly, the answers were quite diverse. Some comments reflect bottlenecks mentioned in interview data, and additional bottlenecks are also mentioned.

Two of the most frequently mentioned bottlenecks in the survey are legacy codebases; source code that is outdated and no longer supported, and technical debt; i.e. the future cost of prioritizing fast delivery over high quality code. Legacy codebases and technical debt can be viewed as interrelated as they both stem from prioritizing delivering new features fast over maintaining and updating the underlying codebase and the existing features. This is further commented on by one respondent who says that too little regard is given to the maintainability of the systems and features that they build, which leads to error-prone solutions and a need for frequent patches and bug fixes. Another respondent credits technical debt - expressed by the large amount of features and services that need continuous maintenance - with leading to increasing amounts of WIP, which often exceed the intended WIP limits within the team. Conclusively, these bottlenecks relate to both tempo and stability metrics, as legacy codebases and technical debt lead to decreased quality as well as additional time-consuming activities in the test-pipeline.

Reflecting what was mentioned during the interviews, the app release cycle (i.e. the deployment and test pipeline for apps) is mentioned by several respondents in the survey as one of the biggest bottlenecks, as well as time-consuming assistance to Customer Support causing time fragmentation, and the lack of automated testing.

An additional aspect mentioned in the survey data as a bottleneck is the differing definition of 'Done' in different Jira projects (briefly mentioned in Section *5.3.1 Obstacles in Four Key Metrics estimation*). 'Done' is either implying that the coding is completed on the ticket and it is waiting for release, or 'Done' is implying that the ticket has been released to production. This is interesting from a communication perspective, as teams may have very differing routines due to their autonomy, which can lead to confusion and failure to agree. Connected to this, several respondents mention communication explicitly as a bottleneck, and an even larger group of respondents state that team interdependencies and unclear ownership is a major bottleneck. These aspects can also be related to responses about the increasing size of the Tech Department and teams as being hindrances. Additionally, communication with stakeholders is also mentioned by several respondents. Among these comments are unclear ownership and priorities, misalignment between vision and mission and the work, and unclear direction stated as bottlenecks. Furthermore, a couple of respondents state that requests directly from management can be interrupting, and stakeholder input can interfere with the roadmap and planning. Overall, these communication issues likely affect Delivery Lead Time and Deployment Frequency at Storytel, as they can detract from efficiency by causing delays in decision-making or cause unnecessary work.

Finally, unclear and unrefined requirements for features is mentioned by a large number of respondents as a recurring issue. This could be partly related to survey statements commenting on how there are too few Product Managers, and the growing pains connected to an increasing Product Manager organization. A couple of respondents state that too little time is spent on planning and designing, which results in changes during development that requires development having to start over, which would directly affect the Delivery Lead Time metric.

9. Conclusions

The following conclusion wraps this thesis up by outlining the major findings. Suggestions are made on potential angles for future research.

9.1 Research Questions

Where does Storytel rank in the software development performance categories based on the Four Key Metrics?

Based on this study, Storytel cannot be conclusively categorised into each of the Four Key Metrics. However, estimates with varying degrees of certainty can be made across all of them.

In the Tempo metrics, Delivery Lead Time and Deployment Frequency, Storytel is overall a medium to high performer. For the metric Delivery Lead Time, performance is conclusively classified as medium at Storytel with an average lead time between one week and one month. In the metric Deployment Frequency, Storytel corresponds to both high and medium performers across different parts of the organisation. The deployments to the apps are made every three weeks (medium performance) while other services generally deploy between once per day and once per week (high performance).

Apart from the respective category classification, the main takeaways when it comes to Tempo metrics is that the Delivery Lead Time has increased within the limits of the current Deployment Frequency. Deployment Frequency routines are currently being introduced to more services than the apps. For these actions, it could be valuable to be aware of the potential adaptation to a certain Delivery Lead Time that comes along with a fixed and scheduled Deployment Frequency. Since the chosen Deployment Frequency inherently equals a choice of batch size and theory suggests that smaller batch sizes are better - as they accelerate feedback, enable faster cycle times and reduce overhead and risk - one should adapt Deployment Frequency to maximize these effects.

The stability metrics are a bit more inconclusive in comparison to the tempo metrics. In particular the Mean Time To Restore metric for which we were unable to find appropriate data to give conclusive findings on where Storytel places in this aspect of delivery performance. However, the majority of estimates in our survey data indicate that Storytel's Mean Time To Restore is within the scope of a medium to a high performer (less than one day). Change Fail Rate, while no data source could be found indicating an exact percentage, can be conclusively categorised in the category of 0-15% due to what can be inferred from other broader findings. Since this percentage spans all three top categories, the Change Fail Rate metric at Storytel classifies them as a medium to elite performer.

While the Stability metrics were ambiguous to determine through raw system data, the measuring attempts were not unfruitful in terms of insights and takeaways. When looking at the lead time for critical and blockers bugs, the Mean Time To Restore was found to be too

slow to rank in the performance category table (worse than low performer, i.e. outside the range of ‘1 week to 1 month’). Our conclusion is that inflation has likely occurred along with the growth of the testing organizations that make the internal documentation outdated, meaning that the majority of critical and blocker bugs are in fact not severe enough to require immediate remediation, such as a hotfix. The findings from this attempt could also be misleading due to fallacy in the assumption that most bugs in Jira are in production.

Our suggestions for future routines that would facilitate measurements is to clear out outdated bugs in Jira, introduce a way to separate bugs that are currently in production from those that are not, and to update the bug priority guidelines. We also see benefits of streamlining cooperation with Customer Support, and find a better solution to the current procedure that continually interrupts developers through Slack channels with requests, especially since there is a wide range of urgency among these errands. Lastly, the Change Fail Rate is low, hypothetically because of the skill and commitment of the people rather than the infallibility of current process routines. There's a risk that this rate could increase along with an continuously growing workforce and a consequent decreasing sense of individual responsibility among employees.

Overall, Storytel spans different categories of performer in the different metrics depending on where in the Tech Department you look. However, there are indications for each of the Four Key Metrics that Storytel generally can be minimally classified as a ‘medium’ performer.

What human and technical organisational factors have an impact on Storytel’s software development performance?

The factors that our statistical measures through factor analysis found to have an impact on the *Four Key Metrics* were ‘Generative Culture’, ‘Efficiency (Automation and Shared Responsibility)’ and ‘Number of Projects’.

While ‘Generative Culture’ is the factor we found to have the largest impact on Delivery Lead Time and it is the metric that changed the most between 2019 and 2020, none of our follow-up findings indicate that Storytel’s culture has changed significantly during this period. Findings from HR survey data for example, corroborate that the culture at Storyel’s Tech Department is generative. Looking more closely at one of the items which has a significantly larger factor loading than the other items, on the subject of learning from failures, we find that lack of a well-functioning feedback culture might be a growing source of error as the Tech Department increases in size and that the need of more reflective and retrospective procedures could increase.

The ‘Efficiency’ factor, spanning the theoretical constructs team cohesion, automation and lean management, are indicated to have an impact on the Deployment Frequency. While automation and lean management practices impact on Deployment Frequency were expected, items covering shared responsibilities within teams and whether a team put effort on facilitating work

for other teams, was less anticipated. But HR survey questions regarding responsibility distribution, communication about delays, and visibility into projects within the team are in fact scoring significantly lower than other team identity questions. Moreover, interview data suggests that employees' level of insight into other teams have significantly decreased as the organization has grown, affecting the possibilities to facilitate work for other teams. Determinedly, these items are not unreasonable to have had a negative impact on the Deployment Frequency in the Tech Department.

The negative correlation that was found between 'Number of Projects' and the metric Mean Time To Restore was initially found to be counter-intuitive due to the assumed negative effects of time fragmentation, brought on by having many active projects, on performance. But from our survey the calculated E-Factor is above the suggested benchmark which indicates an environment that allows employees to get into an uninterrupted flow, supporting that task switches might be of a positive character in relation to Mean Time To Restore. Further supporting this correlation is that the restoring process is likely facilitated if the involved employees have recent experience of working on the affected code, which is more probable if they are involved in many projects. Lastly, that they can prioritize solving issues easier than developers working on only one project, as they have a more holistic view.

We found no factors to significantly correlate with the Change Fail Rate metric. Consequently, we are left to analyze material from interviews. One idea is that Change Fail Rate is kept low because of the skill of people at Storytel's Tech Department rather than a fail-safe process. The underlying reasons for this can be high job satisfaction, good mood and intrinsic meaningfulness among other things. According to HR Survey data, the eNPS score reflects a high and increasing job satisfaction at Storytels Tech Department (49 in the latest measurement from late 2020), close to the benchmark for 'excellent' (50). In fact, job satisfaction, mood and perceived meaningfulness of work are found to be of high levels at the Tech Department, which could influence the employees to feel motivated and responsible to ensure good quality of work and in turn keep the Change Fail Rate low.

Among a lot of factors, the factors with a statistically supported impact on the *Four Key Metrics* were expected to be more than three. Other factors might have been dismissed due to design choices and lack of a larger response base. However, in line with the theory on influencing factors on productivity, our factor analysis does support that human factors have an equally large impact as technical factors.

What bottlenecks exist that hinder Storytel from being a better performer?

In contrast to the factor analysis correlations with the Four Key Metrics, the bottlenecks found in interview data and from our survey spanned a lot of different theoretical constructs. The most frequently mentioned bottlenecks are mainly related to the constructs Architecture, Automation, Time Fragmentation and Communication.

As an effect of an environment that allows teams and developers to freely choose among tools and programming languages and of residual legacy systems, there is a lack of common architecture in the Tech Department. This also leads to irregular and in some cases non-existent documentation and outdated code that is difficult to maintain. Giving architectural considerations a greater focus and a higher priority - by better maintaining and supporting existing codebases and accelerating migration from old legacy systems - can prevent this bottleneck from becoming a larger issue, and help avoid increasing the technical debt.

A large number of people mention test-related issues and lack of automation in testing as hindering Storytel from a better performance, as well as the app release cycle in general. While bugs are not frequently mentioned as a bottleneck, support errands are - mainly relating to how communication with Customer Support regarding bugs and issues in the apps leads to undesired time fragmentation. Further investments in automation in the test pipeline and, as previously mentioned, re-organizing how support errands are communicated to the Tech Department could minimize their respective impact on software delivery performance.

Bottlenecks related to different aspects of communication were also found in both the interview and survey data. Confusion and consequent issues seem to frequently occur due to unclear responsibilities, differing routines, and complex interdependencies between teams. Additionally, a similar lack of communication, unclear responsibilities and vague directions from the Management organization is also reflected in the survey data. Related to this, unclear requirements and inadequate planning which eventually leads to scrapped code and consequently longer cycle times is also voiced as a concern in the survey data. These aspects are arguably expected growing pains of a quickly expanding organization, but there is an increasing need of addressing these issues fast as the Tech Department grows more complex. As supported by Wagner and Ruhe (2018), efforts to increase the communication intensity should be made alongside the increasing number of people in the organization, as this has been found to positively correlate with successful projects. Improving communication channels and clarifying responsibilities on all levels of the Tech Department can therefore be crucial to improve software delivery performance.

In the Theory of Constraints, the focus is on identifying the one constraint that limits the whole organization and suggests companies to restructure the rest of the organization around it, adopting the common idiom "a chain is no stronger than its weakest link" (Goldratt and Cox, 2012). This thesis has not had the approach to find the one major bottleneck constraining the system, rather potential bottlenecks in different parts of the organization, but this could be an interesting future research among the found bottlenecks.

9.2 Limitations

A lot of findings that we initially intended to analyze further were ultimately outside of the limited time scope of 20 weeks, and were excluded from the final report. Some of these abandoned ideas are described below.

There was a section included in the survey specifically intended for those that had been employed for 12 months or longer at Storytel that we called 'Historical Perspective'. On a Likert-scale from 1 to 5, respondents were asked to rank how different factors had changed compared to 12 months ago. Due to time constraints and this section's appraised relevance, we did not analyze these results further and connect them to the other findings. We conclude in retrospect that this could have provided interesting further context. A table with these survey results are included in the Appendix 7). Connected to this limitation, an initial hypothesis regarding impact on factors from a rapidly increasing workforce was not possible to investigate further due to the method design and time constraints.

Additionally, another initial intent was to include a comparison between different teams and roles within the Tech Department in regards to the *Four Key Metrics* and different factors measured in the survey. This would have allowed for a more detailed analysis between why software delivery performance might differ across the Tech Department. Another reason for excluding this was due to the limited number of survey respondents; the number of employees from each team would be too irregular (ranging between one and eleven respondents per team) to be able to generalize survey findings. Lastly, information regarding productivity on team or role level is more sensitive than on organizational level and would have required caution and more confidentiality.

Connected to comparisons between teams, studying a more diverse set of workflow stages were deprioritized. Actionable Agile proved to be an inefficient tool to load the large existing amount of issues that prompted this limitation to be made. This could possibly possess interesting information by for example making it possible to measure the time for each issue in "Waiting For Release" to investigate how much faster Deployment Frequency could be if it was dictated by finished code segments.

9.3 Sources of error

The fact that one team by accident was handling tickets in an unconventional way, and that these issues were able to modify at least one graph as much as shown in the method section *5.3.2 Preprocessing of Raw System data sets* (not as clear in other graphs), we needed to examine if there were several similar kinds of deviations. The further examination was unfruitful, but there could possibly exist similar discrepancies since they can be difficult to find. This however supports the choice of looking into more than one type of graph (checking WIP and throughput demonstrated this anomaly, but not the cycle time scatter plot). The steady increasing amount of WIP, not following the increasing number of employees but rather close to exponential, was an indication that more teams used the same routines as the team that never classified issues as 'Done'. However, after excluding these issues the WIP follows a pattern. The number of WIP is doubled each year starting from 2018. In January each year, the WIP is as follows: 500, 1K, 2K and finally 4K in 2021. Furthermore, it is brought up during interviews

that there is in fact a lot of work in progress going on at the moment. With these analyses being made, we had to make the assumption that no other major discrepancies were to be found.

Another source of error could be that Actionable Agile only allows for showing trends, used to estimate the average and mean values rather than median values. Since this is possible for the Jira reports, we got an indication that the median value is a lot less than the average. Since it is possible to exclude outliers in Actionable agile but not in Jira reports, theoretically, the mean value should not differ much from the median value when outliers are excluded, but it would have been interesting to examine. Unfortunately, this has not been possible to investigate further and might therefore be a source of error. Again, this supports the ambition to look at trends rather than numbers, since these will be sufficient no matter if mean or median values are used.

Finally, the factor analysis on the survey data is an additional source of error due to the small sample size. There are different rules of thumb when it comes to appropriate sample size in factor analysis - some authors state that 100 subjects is minimally sufficient, and that the more the better (Kline, 1994). Other authors base minimal sample size on the ratio between the number of subjects and the number of items - ranging between 2 (Kline, 1994) and 10 (Nunnally, 1978) minimum subjects per factor being recommended. This ratio is approximately 2.5 in our study (29 items, 75 subjects). In summary, compared to most of the general recommendations we found in factor analysis literature, the sample size in this study (75 subjects) is on the verge of being insufficient for a reliable factor analysis with reasonable statistical power. We conclude that the results might not be replicated if the survey was to be repeated.

9.4 Lessons learned

When starting off this project, there was a wish for creating and implementing a measurement framework that could be reused. Soon it became clear that the infrastructure and change in guidelines that would be needed was not possible to achieve during the available time frame. Neither did the background information and sufficient knowledge for such an investment exist. It became clear that this thesis would serve as the investigation for if, or ground work for why, such an investment could be valuable. It would be necessary to make such a decision to cover the whole tech organization, and how to structure work with labels and would not be a change happening from one day to another.

As we have learnt more about Storytel's processes, employees and inner workings during this study, we have also grasped the complexities of trying to generalize values of the *Four Key Metrics* across the Tech Department. In the beginning of the project, we had a simplified view on what this task would entail. Because of the heterogeneous tools, tech-stacks, services and work ethics of teams and crews, the F*our Key Metrics* might not be the most appropriate nor accurate measure to try and place Storytel's software delivery performance on an organizational level. If the metrics were to be implemented in an automated and sustainable

way (the use of Actionable Agile and surveys are a time-consuming process with little recurring possibilities), they could perhaps be more useful to track Storytels performance on team level. If the goal is to measure on an organizational level, some generalizations among teams' work procedures will be needed. However, the *Four Key Metrics* already enable an estimate of Storytel's placing on the global scale of Software development organisations. If Storytel as a company will continue to grow in size and number of markets, it might not only be desirable - but necessary to introduce a measurement framework that can additionally provide insight into how their productivity has changed over time.

9.5 Future research

The above limitations and sources of error all contain ideas that could be further evaluated and investigated in future research. Though it might be sensitive for Storytel to compare teams, we got indications during interviews that productivity differs among teams. To study why this might be the case and to learn from each other is an interesting suggestion that could be followed through with potentially great outcomes for Storytel. Both within and outside Storytel, it would be interesting to perform a similar study with a greater sample size. The survey material could also be used to compare performance between different companies' Tech Departments.

The initial work that was prepared and initiated, but brought out of scope, on analyzing throughput is found in Appendix 8. Although measuring the throughput of issues at Storytel generates interesting insight, it was decided that due to the overlaps between throughput and the tempo metrics (Delivery Lead Time and Deployment Frequency) and the findings being exclusively interesting internally at Storytel, throughput would not be a focal point of this thesis. However, studies investigating resolved issues per employee over time suggest correlations with remote work that would be interesting to examine further. A discussion that tasks requiring concentration may be best undertaken at home, whereas other tasks involving teamwork may be best undertaken in the office is proposed. Potential conclusions that would be valuable for Storytel is that remote work is perhaps especially productive for developers to a higher extent than other roles.

The main operative future research for Storytel is however, as mentioned, to implement an autonomous and sustainable measurement of the *Four Key Metrics.* A Github setup script is provided by Google's DevOps Research and Assessment team (Graves Portman, 2020) for this purpose that could be implemented, but there is a need to invest in the structure to get reliable, comparable and generalized results with this approach.

If this option is not examined further, to complement the findings of this thesis and fill in the gaps, there are changes going on at Storytel that will enable new measuring opportunities. Inside the Customer Engagement Department for example, they are working on a solution to follow up each customer support errand when bugs are fixed. This will likely enable a quick and easy setup to track Mean Time To Restore that was not possible at the point of this thesis.

9.6 Final words

One interviewee said that feeling the company's well-being and productivity, influences individuals to be more productive, meaning that the environment itself helps to motivate its employees. Adding to the observation, the interviewee says that it has an inspiring effect when something is suddenly released of which you had no idea that it was even in progress *(Interview 14: Developer, 2021)*. This further supports the importance that communicative actions regarding things happening in the organization, can be a positive context switch and contributory to employees' productivity. Apart from the hazards and growing pains of a fast growing number of employees, this could be evidence of a synergy effect that has occurred along with the increasing workforce. Furthermore, it is a good (non-statistical) example of human factors having an equally large importance for productivity as technical factors.

With the result of greater understanding on how the Storytel Tech Department software development process works, this thesis has mapped the Tech Departments productivity defined by the *Four Key Metrics* by collecting data from internal systems, interviews and a survey. By measuring these key metrics, divided into measurements of stability and tempo, a classification has been made into one out of four performance categories. Furthermore, factors influencing software delivery performance have been determined through factor analysis. Lastly, bottlenecks potentially hindering performance have been deliberated. By creating a performance baseline, teams can analyze and improve on their work processes. If performed continuously - changes can be tracked and analyzed in order to improve the software development process and ultimately achieve better business outcomes.

References

Actionable Agile (2021). *Actionable Agile.* [online] Available at: https://actionableagile.com/ [Accessed 20 Feb. 2020].

Al-Jabery, K.K., Obafemi-Ajayi, T., Olbricht, G.R. and Wunsch II, D.C. (2020). *Computational learning approaches to data analytics in biomedical applications*. London: Academic Press, An Imprint Of Elsevier.

Ali, S., Farid, F. and Ibrarullah (2016). 'Effect of Transformational Leadership on Job Satisfaction and Organizational Commitment', Humanistic Management Network, Research Paper Series, (2), DOI: 10.2139/ssrn.2713386.

Anderson, D.J. (2004). *Agile Management for Software Engineering : Applying the Theory of Constraints for Business Results*. Upper Saddle River, NJ: Prentice Hall.

Android. (2021). *Android.* [online] Available at: https://www.android.com/ [Accessed 5 Mar. 2021].

Apple. (2021a). *App Store*. [online] Available at: https://www.apple.com/app-store/. [Accessed 5 Mar. 2021].

Apple. (2021b). *iOS*. [online] Available at: https://www.apple.com/iOS/ [Accessed 5 Mar. 2021].

Atlassian (2019). *Jira Cloud*. [online] Atlassian. Available at: https://www.atlassian.com/software/jira [Accessed 15 Feb. 2021].

Atlassian (2021). *Epics*. [online] Atlassian. Available at: https://www.atlassian.com/agile/project-management/epics#:~:text=What%20is%20an%20agile%20epic [Accessed 25 Feb. 2021].

Bellman, L. and Hübler, O. (2020). 'Working from home, job satisfaction and work–life balance – robust or heterogeneous links?', *International Journal of Manpower*, DOI: 10.1108/IJM-10-2019-0458.

Bloom, N., Liang, J., Roberts, J. and Ying, Z.J. (2014). 'Does Working from Home Work? Evidence from a Chinese Experiment'. *The Quarterly Journal of Economics* 130(1), p.165–218, DOI: 10.1093/qje/qju032.

Boktugg (2020). *Bästa ljudbokstjänsterna 2020 – vi testar och jämför appar för att lyssna på ljudböcker i mobilen.* Available at: https://www.boktugg.se/ljudbocker/ [Accessed 26 Mar. 2021].

Brooks, F. (1995). *The Mythical Man-Month: Essays on Software Engineering*. Reading: Addison-Wesley.

Bryman, A. and Nilsson, B. (2011). *Samhällsvetenskapliga metoder*. Stockholm: Liber.

Construx Software (2016). *Measuring Software Development Productivity | Steve McConnell*. *YouTube*. Available at: https://www.youtube.com/watch?v=x4IboMnTdSA&t=3472s [Accessed 13 Nov. 2020].

Conway, M. (1968). 'How Do Committees Invent?' *Datamation* 14(4), p.28–31.

Dalen, M. (2008). *Intervju som metod*. Malmö: Gleerups Utbildning.

Delibr (2020). *Feature Refinement Tool for Product Managers*. [online] Delibr. Available at: https://www.delibr.com/ [Accessed 21 Jan. 2021].

Demarco, T. and Lister, T. (1987). *Peopleware: Productive Projects and Teams*. New York: Dorset House Publishing.

DevOps Research and Assessment (DORA) (2021). *DevOps capabilities*. [online] Google Cloud. Available at: https://cloud.google.com/solutions/devops/capabilities [Accessed 17 Feb. 2021].

Dillman, D.A. (1991). The Design and Administration of Mail Surveys. Annual Review of Sociology, 17(1), p.225–249. DOI: 10.1146/annurev.so.17.080191.001301.

DiStefano, C., Zhu, M. and Mîndrilã, D. (2009). 'Understanding and Using Factor Scores: Considerations for the Applied Researcher'. *Practical Assessment, Research, and Evaluation*, 14(1), p.20. DOI: 10.7275/da8t-4g52.

Djurfeldt G., Larsson, R. and Stjärnhagen, O. (2010). *Statistisk verktygslåda 1 : samhällsvetenskaplig orsaksanalys med kvantitativa metoder*. Lund: Studentlitteratur.

Eriksson, L.T. and Wiedersheim-Paul, F. (2014). *Att utreda, forska och rapportera*. Stockholm: Liber.

Esaiasson, P., Gilljam, M., Oscarsson, H, Towns, A.E. and Wängnerud, L. (2017). *Metodpraktikan : konsten att studera samhälle, individ och marknad*. 5th ed. Stockholm: Wolters Kluwer.

Field, A.P. (2009). *Discovering statistics using SPSS*. 3rd ed. London: Sage.

Forsgren, N., Humble, J. and Kim, G. (2018). *Accelerate: The Science Behind DevOps: Building and scaling high performing technology organizations*. Portland, Oregon: IT Revolution.

GitHub. (2019). *The elusive quest to measure developer productivity - GitHub Universe 2019*. *YouTube*. Available at: https://www.youtube.com/watch?v=cRJZldsHS3c [Accessed 12 Nov. 2020].

Github. (2021). *Github*. Available at: https://github.com/ [Accessed 15 Feb. 2021].

Glen, S. (2021). *P-Value in Statistical Hypothesis Tests: What is it?* Available at: https://www.statisticshowto.com/probability-and-statistics/statistics-definitions/p-value/ [Accessed 26 Mar. 2021].

Goldratt, E.M. and Cox, J. (1984). *The Goal: A Process of Ongoing Improvement: a Process of Ongoing Improvement*. Great Barrington, MA: North River Press.

Google Cloud (2019). *Cloud Computing Services | Google Cloud*. [online] Google Cloud. Available at: https://cloud.google.com/ [Accessed 9 Mar. 2021].

Google Play. (2021). *Google Play* [online] Available at: https://www.play.google.com/ [Accessed 9 Mar. 2021].

Graves Portman, D. (2020). *Using the Four Keys to measure your DevOps performance*. [online] Google Cloud Blog. Available at: https://cloud.google.com/blog/products/devops-sre/using-the-four-keys-to-measure-your-devops-performance [Accessed 14 Oct. 2020].

Harpaz, I. (2002). 'Advantages and disadvantages of telecommuting for the individual, organization and society'. *Work Study*, 51(2), p.74–80. DOI: 10.1108/00438020210418791.

Hinton, P.R., Mcmurray, I. and Brownlow, C. (2014). *SPSS explained*. London; New York: Routledge, Taylor & Francis Group.

IBM (2019). *SPSS Statistics - Overview*. [online] Ibm.com. Available at: https://www.ibm.com/products/spss-statistics. [Accessed 1 Mar. 2021].

IBM (2021). *IBM Knowledge Center*. [online] www.ibm.com. Available at: https://www.ibm.com/support/knowledgecenter/SSLVMB_23.0.0/spss/tutorials/fac_telco_kmo_01.html. [Accessed 1 Mar. 2021].

Kline, P. (1994). *An easy guide to factor analysis*. East Sussex, United Kingdom: Psychology Press.

Knekta, E., Runyon, C. and Eddy, S. (2019). 'One Size Doesn't Fit All: Using Factor Analysis to Gather Validity Evidence When Using Surveys in Your Research.' *CBE—Life Sciences Education* 18(1). DOI: 10.1187/cbe.18-04-0064.

Lindstedt, I. (2017). *Forskningens hantverk*. Lund: Studentlitteratur AB.

Litwin, M.S. (1999). *How to Measure Survey Reliability and Validity*. Thousand Oaks: Sage.

Mabel, O.A. and Olayemi, O.S. (2020). 'A Comparison of Principal Component Analysis, Maximum Likelihood and the Principal Axis in Factor Analysis.' *American Journal of Mathematics and Statistics*, 10(2), p.44–54. DOI: 10.5923/j.ajms.20201002.03.

Madhavan, S. (2019). *Employee Net Promoter Score: A Good Measure of Engagement?* [online] www.hrtechnologist.com. Available at: https://www.hrtechnologist.com/articles/employee-engagement/employee-net-promoter-score-a-good-measure-of-engagement/#:~:text=An%20eNPS%20score%20can%20range [Accessed 1 Mar. 2021].

Mathew, J. (2007). 'The relationship of organisational culture with productivity and quality: A study of Indian software organisations.' *Employee Relations*, 29(6), p.677–695. DOI: 10.1108/01425450710826140.

Meyer, A.N., Fritz, T., Murphy, G.C. and Zimmermann, T. (2014). 'Software developers' perceptions of productivity.' *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering - FSE 2014*. p.19-29. DOI: 10.1145/2635868.2635892.

Miro (2021). *Miro*. Available at: https://miro.com [Accessed 15 Feb. 2021].

Nunnally, J.C. (1978). *Psychometric theory : 2nd ed.* New York: Mcgraw-Hill.

Øredev Conference (2015). *Troy Magennis - AGILE METRICS - BEYOND BURN UP/DOWN'S ONTO METRIC DRIVEN COACHING | Øredev 2015*. [online] Vimeo. Available at: https://vimeo.com/144824390 [Accessed 11 Nov. 2020].

Osborne, J.W. (2014). *Best practices in exploratory factor analysis*. Charleston, SC: CreateSpace.

Oswald, A.J., Proto, E. and Sgroi, D. (2009). 'Happiness and Productivity'. *Journal of Labor Economics*, 33(4), p.789–822. DOI: 10.1086/681096.

Pakdamanian, E., Shiyamsunthar, N. and Claudio, D. (2016) 'Simulating the effect of workers' mood on the productivity of assembly lines.' *2016 Winter Simulation Conference (WSC)*, Washington, DC, USA, 2016, p. 3440-3451. DOI: 10.1109/WSC.2016.7822374.

Petersen, K. and Wohlin, C. (2011). Measuring the flow in lean software development. *Software: Practice and Experience*, 41(9), p.975–996. DOI: 10.1002/spe.975.

Rafferty, A.E. and Griffin, M.A. (2004). 'Dimensions of transformational leadership: Conceptual and empirical extensions.' *The Leadership Quarterly*, 15(3), p.329–354. DOI: 10.1016/j.leaqua.2004.02.009.

Schwarz, J. (2011). *Research Methodology: Tools Applied Data Analysis (with SPSS) Lecture 03: Factor Analysis Outline. 8 Concepts of Factor Analysis. 12 Factor Analysis with SPSS: A detailed example. 16*. Lecture notes, Lucerne University of Applied Sciences and Art, Lucerne [online] . Available at: http://www.schwarzpartners.ch/Applied_Data_Analysis_2011/Lect%2003_EN.pdf [Accessed 1 Mar. 2021].

Sedlak, P. (2020). 'Employee Net Promoter Score (eNPS) as a Single-item Measure of Employee Work Satisfaction. An Empirical Evidence from Companies Operating in Poland.' *Contemporary organisation and management. Challenges and trends*, p.347–357. DOI: 10.18778/8220-333-2.21.

Slack (2021). *Welcome to your new HQ*. [online] Slack. Available at: https://slack.com/intl/en-se/ [Accessed 1 Mar. 2021].

Spada, M.M., Hiou, K. and Nikcevic, A.V. (2006). 'Metacognitions, Emotions, and Procrastination.' *Journal of Cognitive Psychotherapy*, 20(3), p.319–326. DOI: 10.1891/jcop.20.3.319.

Statistics Solutions (2013). *Confirmatory Factor Analysis*. [online] Statistics Solutions. Available at: https://www.statisticssolutions.com/confirmatory-factor-analysis/. [Accessed 23 Feb. 2021].

Stevens, J. (2002). *Applied multivariate statistics for the social sciences*. Mahwah, N.J.: Lawrence Erlbaum Associates.

Storytel Service Status Log. (2021) *Storytel*. Available at: https://status.storytel.com/pages/history/54febea209f0ba2f5b000009 [Accessed 21 Jan. 2021].

Storytel AB. (2019a). *Annual Report 2019.* Available at: https://investors.storytel.com/en/annual-report-2019-storytel-ab-publ/ [Accessed 15 Nov. 2020].

Storytel AB. (2019b) *Sustainability Report 2019.* Available at: https://investors.storytel.com/en/wpcontent/uploads/sites/2/2020/02/storytel-annual-report-2019-storytel-ab-publ-200402.pdf [Accessed 15 Nov. 2020].

Straub, D., Gefen, D. and Boudreau, M.-C. (2004). Validation Guidelines for IS Positivist Research. *Communications of the Association for Information Systems*, 3(1), p.380-427. DOI: 10.17705/1CAIS.01324.

SurveyMonkey (2018). *Likert Scale: What It Is & How to Use It | SurveyMonkey*. [online] SurveyMonkey. Available at: https://www.surveymonkey.com/mp/likert-scale/. [Accessed 6 Dec. 2020].

The Agilist. (2014). *Little's Law.* Available at: *https://theagileist.wordpress.com/2014/10/15/littles-law/* [Accessed 19 Feb. 2021].

Trost, J. (2012). *Enkätboken*. 4th edition. Lund: Studentlitteratur AB.

Wagner, S. and Ruhe, M. (2018). 'A Systematic Review of Productivity Factors in Software Development.' *Cornell University arXiv*. DOI: 1801.06475.

Watad, M.M. Will, P. (2003). 'Telecommuting and organizational change: a middle-managers' perspective.' *Business Process Management Journal.* 9(4), p.459-472. DOI: 10.1108/14637150310484517.

Westrum, R. (2004). 'A typology of organisational cultures.' *Quality and Safety in Health Care*, 13(2), p.ii22–ii27. DOI: 10.1136/qshc.2003.009522.

Yong, A.G. and Pearce, S. (2013). 'A Beginner's Guide to Factor Analysis: Focusing on Exploratory Factor Analysis.' *Tutorials in Quantitative Methods for Psychology*, 9(2), p.79–94. DOI: 10.20982/tqmp.09.2.p079.

Internal documents (unavailable without a Storytel-account):

Storytel (2021a). *Tech Recruitment Tracking.* Internal document. [Last Accessed 4 Mar. 2021]. Unpublished.

Storytel (2021b). *Global Core Metrics Explanation Document.* Internal document. [Last Accessed 4 Mar. 2021]. Unpublished.

Storytel (2021c). *Onboarding Day - tech team.* Internal document. [Last Accessed 4 Mar. 2021]. Unpublished.

Storytel (2021d). *App Release Retrospective*. Internal document. [Last Accessed 5 Mar. 2021]. Unpublished.

Storytel (2021e). *App Train Release Strategy*. Internal document. [Last Accessed 5 Mar. 2021]. Unpublished.

Storytel (2021f). *App Release Train Schema.* Internal document. [Last Accessed 5 Mar. 2021]. Unpublished.

Interviews:

No.	Role	Date
1	Product Manager	Nov 27, 2020
2	Crew Coach	Dec 1, 2020
3	Developer	Dec 2, 2020
4	Tech Manager	Dec 7, 2020
5	Customer Support	Dec 15, 2020
6	Crew Coach	Jan 8, 2021
7	Developer	Jan 8, 2021
8	Test Lead	Jan 11, 2021
9	HR	Jan 12, 2021
10	Tech Manager	Jan 18, 2021
11	Customer Support	Jan 18, 2021
12	Developer	Jan 25, 2021
13	Crew Coach	Jan 25, 2021
14	Developer	Jan 27, 2021
15	Tech Manager	Feb 1, 2021
16	Tech Manager	Mar 7, 2021

Appendix

Appendix 1. Questions in questionnaire

Demographics

- Which team do you belong to?
- What is your primary role?
- How long have you been at Storytel in total?
- How many projects are you working on right now simultaneously?
- How many projects have you worked on during the last three months?
- How many projects has your team been involved in during the last three months?

Historical perspective

- How has communication changed within your team? It is…
- How has communication changed between teams? It is…
- How has your job satisfaction changed? It has…
- How has the responsibility distribution changed in your team? It is…
- How has the responsibility distribution changed in the Tech Department? It is…
- How has your team's ability to deploy features independently from other applications or services changed? It has…
- How has the ability to independently test an application without requiring an integrated environment changed? It has…
- In your opinion, how has Storytel's Tech Department's productivity changed? It has…
- How has your individual productivity changed? It has…
- [Optional Comment] - If you want to further comment any of your answers above, please do so here.

Culture

- In my team, information is actively sought.
- In my team, messengers are not punished when they deliver bad news.
- In my team, responsibilities are shared.
- In my team, cross-functional collaboration is encouraged and rewarded.
- In my team, failure causes inquiry so that we learn from the experience.
- In my team, new ideas are welcomed.
- In my team, we put effort into facilitating work for other teams.

Teams

- I have a good insight into what other teams are doing.
- I wish I had more insight into what other teams are doing.

Job Satisfaction

- I would recommend my organization as a place to work.
- I would recommend my team as a place to work.

Leadership

- My manager challenges me to see problems from new perspectives.
- My manager notices me.
- My manager regularly gives actionable feedback that helps me improve my performance.

Team Identity

- I know the reason for all features we develop in my team.
- My team is collectively working towards the same goals.
- I am proud of being a part of my team.

Communication

- Communication is efficient in my team.
- How often do you interact with members from other teams for inspiration and/or assistance for a task you are working on?

Time

- How many hours is your average work day?
- How many complete/full hours without interruptions do you have on an average workday to spend on your main tasks?
- Switching between tasks can be good in terms of being productive.
- How many consecutive uninterrupted hours would I prefer to have on a regular working day?

Architecture

- Features developed in my team can be tested and deployed without being dependent on other teams.
- Security testing is generally done during the early phases of development.

Automation

- What is, in your estimate, the percentage of tasks related to deployment that are automated in your team?
- What is, in your estimate, the percentage of tasks related to testing that are automated in your team?

Flow

- I have access to visual displays showing the status and/or flow of work within my team by some metrics.
- What metrics are available in your team?
- What metrics are not available today, but you would like to have access to in your team?
- In my team, the ambition is to keep the number of WIP (work in progress) to a minimum.
- In your opinion, what is the biggest bottleneck slowing down the work in the Tech Department?

Productivity

- In your estimation, how often does your team deploy code to production (not necessarily to end users)?
- In your estimation, how long does it take for your team to go from code committed to code successfully running in production (not necessarily reaching end customers)?
- In your estimation, how long does it generally take for your team to restore service when a service incident or a defect that impacts users occur?
- In my team, what percentage of changes deployed to production result in degraded service (e.g. lead to service impairment or service outage) and in turn require immediate remediation (e.g., require a hotfix, rollback, fix forward, patch)?

Appendix 2. Final list of the 29 survey items for factor analysis

Category	Theoretical Construct	Items
Culture	Generative Culture (GC)	GC1 - In my team, information is actively sought.
		GC2 - In my team, messengers are not punished when they deliver bad news.
		GC3 - In my team, responsibilities are shared.
		GC4 - In my team, cross-functional collaboration is encouraged and rewarded.
		GC5 - In my team, failure causes inquiry so that we learn from the experience.
		GC6 - In my team, new ideas are welcomed.
	Team Cohesion (TC)	TC1 - In my team, we put effort into facilitating work for other teams.
		TC2 - I have a good insight into what other teams are doing.
		TC3 - I wish I had more insight into what other teams are doing.
	Job Satisfaction (JS)	JS1 - I would recommend my organization as a place to work.
		JS2 - I would recommend my team as a place to work.
	Transformational Leadership (TL)	TL1 - My manager challenges me to see problems from new perspectives.
		TL2 - My manager notices me.
		TL3 - My manager regularly gives actionable feedback that helps me improve my performance.
	Team Identity (TI)	TI1 - I know the reason for all features we develop in my team.
		TI2 - My team is collectively working towards the same goals.
		TI3 - I am proud of being a part of my team.
	Communication (C)	C1 - Communication is efficient in my team.
		C2 - How often do you interact with members from other teams for inspiration and/or assistance for a task you are working on?

Environment	Number of Projects (NP)	NP1 - How many projects are you working on right now simultaneously? NP2 - How many projects have you worked on during the last three months? NP3 - How many projects has your team been involved in during the last three months?
	E-Factor & Time Fragmentation (EF)	EF1 - E-Factor EF2 - Switching between tasks can be good in terms of being productive. EF3 - Fraction desired uninterrupted hours
Process	Architecture (AR)	AR1 - Features developed in my team can be tested and deployed without being dependent on other teams. AR2 - Security testing is generally done during the early phases of development.
	Automation (AU)	AU1 - What is, in your estimate, the percentage of tasks related to deployment that are automated in your team? AU2 - What is, in your estimate, the percentage of tasks related to testing that are automated in your team?
	Lean Management (LM)	LM1 - I have access to visual displays showing the status and/or flow of work within my team by some metrics. LM2 - In my team, the ambition is to keep the number of WIP (work in progress) to a minimum.

Table 13. Factors and corresponding items included in the survey.

Appendix 3. Obstacles in Four Key Metrics Estimation

This is a comprehensive explanation of the obstacles in estimating the *Four Key Metrics*. The tempo metrics are Delivery Lead Time and Deployment Frequency, and the stability metrics are Mean Time To Restore and Change Fail Rate.

Obstacles related to Tempo Metrics

- Autonomous teams: At the Storytel Tech Department, teams have a very high degree of autonomy. As a consequence of this, the way different teams work often differ significantly from each other. As a consequence, there were difficulties in trying to extract values of the *Four Key Metrics*. There are few routines, procedures, and standards that span the entire tech organization, which makes generalization difficult.

- Organizational restructurings: Another obstructing factor in finding reliable data from a longer period of time was due to Storytel's multiple organizational restructurings and refactorings happening because of a very rapid growth of the organization. Some teams are only a few months old and cover new focus areas, some were previously part of a larger team that was split into different focus areas.

- Workflow stages: Determining what workflow stages from Jira to include when measuring Delivery Lead Time proved to be challenging. As there are no standardized guidelines at Storytel regarding what workflow stages a ticket should pass in Jira, data gathered from Jira is not to be treated as exact nor completely reflective of the truth as there are inconsistencies in how data is handled. Rather, the system data should be viewed as an indication of what the reality is at Storytel, and as a way to analyze trends over time. Multiple examples of different workflow procedures exist. For example 'Waiting For Release' is not part of all teams workflow-stages. This is the result of letting all teams decide upon their own flow themselves.

- Consideration of batch size: The metric Delivery Lead Time is, in this report, measured from the time a programmer initiates development on a ticket until the ticket is marked as 'Done'. There was an attempt to only measure the time until development is marked as finished, i.e. excluding 'Testing' and 'Waiting For Release'. Since the fixed app releases happen every third week regardless of when an app-related ticket is completed, involving the part when a ticket is 'Waiting For Release' could be considered uninteresting. Especially, since web releases can happen at any point in time, or at least in shorter intervals, the comparison between these numbers would be ambiguous. However, the decision made was to not adjust the metric due to strategic decisions regarding batch size in the organization. Based on the theory, Delivery Lead Time should be measured until code is in production. To be able to use the metrics for historical comparison and to find and map improvement areas, it is reasonable to not adapt the measure for special arrangements.

- Defining the end: Furthermore, the most interesting comparison is not the exact numbers over time, but the overall trend. Looking at the Delivery Lead Time that for example involves time-consuming testing is still relevant for developers, since it involves significant and meaningful information that affects the entire team delivery. The responsibility for a developed feature should not end when development is marked as finished, as a part of the *shift left* approach and team identity (see Section *3.3.1 Culture Factors* and *3.3.3 Process Factors)*. Measurement is supposed to serve as a benchmark for improvement, and attempts to shrink Delivery Lead Times should be a joint effort, therefore it would not make sense to exclude the bottlenecks, e.g. the test pipeline.

- Definition of 'Done': Measurements in Jira include a significant amount of error due to differences in what the workflow stage 'Done' implies in different Jira projects. Some teams define 'Done' as development having finished on that ticket, but that it has not necessarily been deployed into production. Other teams use it to mark a ticket that has successfully been released to production, however not necessarily reached by end-customers because of the stagewise rollout and use of the feature flag system. A feature flag can indicate that a feature should be accessible to all users in all markets, some markets or none. Furthermore, the workflow stages and their meaning for each team, have also changed over time *(Interview 6: Crew Coach, 2021).*

- Ticket size: Another difficulty in measuring Delivery Lead Time is the difference between teams when it comes to the size of a ticket in Jira. The most explicit guideline described regarding this is one Crew Coach describing that if a ticket is estimated to take more than two workdays, it should be split into smaller subtasks *(Interview 6: Crew Coach, 2021).* The tickets are structured differently between teams, and especially between different tech stacks. Similar estimates are given by several interviewees, however there is no guiding documentation covering several different teams.

- Generalization between services: When discussing the two key metrics that indicate *tempo* - Delivery Lead Time and Deployment Frequency - these vary among on the different services at Storytel. For defining Deployment Frequency, the biggest issue was that it was not possible to give one single estimate that generalizes the entire department. The different services within Storytel have a lot of different routines and procedures when it comes to deployment, and this had to be presented separately.

Obstacles related to Stability Metrics

- Definition of a failure: When discussing the two key metrics that indicate *stability* - Mean Time To Restore and Change Fail Rate - the numbers are heavily dependent on defining what constitutes a failure. A failure could be a bug that the customer rarely notices (or believes is a design choice), a bug that actually impacts the user but has a work-around, a bug that impacts the user that does *not* have a workaround, or it could be related to degraded performance, downtime, partial or whole system disruption.

Several options were considered but none of the attempts were sufficient in estimating the Mean Time to Restore. The options are presented below.

- Hotfixes: According to Forsgren, Humble and Kim (2018), a failure in the primary service or application is something that results in either degraded service or a need for remediation such as a patch, roll-back or a hotfix. A hotfix is a software patch that is delivered to a system as an urgent measure (Storytel, 2021e), and Storytel tries to perform hotfixes as rarely as possible. This is closely connected to the use of their release schedule and a wish to use the regular workflow in order to assure quality.

- Requests for hotfixes: Hotfixes are described within Storytel as very disruptive to the work of developers and testers, and that they can cause additional work for other departments as well. Even requests for hotfixes could therefore be viewed as indications of a degraded service with required remediation, but these are more difficult to find reliable data on. Communication regarding whether a hotfix should be performed or not is carried out in a particular channel on Slack between developers and it is not explicitly logged in Jira or anywhere else. Frequently, there are internal requests for hotfixes after a release that are mutually dismissed after discussion due to the preference in fixing them during the next scheduled release, if the matter is not deemed sufficiently urgent.

- Critical and Blocker bugs: According to internal guidelines, the need for a hotfix is dependent on if there is a bug causing enough impact for customers. They have benchmarks that suggest that a large impact could equal for example more than 200 tickets in one day for Customer Support, or two specific bug *priorities*. To prioritize bugs, Storytel uses a prioritizing schedule in Jira that includes (in order): Irrelevant, Trivial, Minor, Major, Critical and Blocker. Critical and Blockers are issues that are said to be candidates for hotfixes, while others in general can wait to be fixed until the next release. (Storytel, 2021e) Bugs can be found during 'Freeze'-time, in the external review process at Google Store and App Store, in Beta testing, or when the release is deployed to customers.

- Bugs in production: An issue we came across was that Storytel had no option to separate bugs that are in production. It was concluded based on employee estimates, that the large majority of bugs are in production. Some employees mentioned that there are likely some open bug tickets in Jira that are connected to functionalities that are no longer active, and therefore should not be regarded as active bugs. However, these bugs are estimated to be in a small minority of all of the thousands of bugs logged in Jira and therefore unlikely to significantly skew the data.

- External disruption log: Another option was to look at system disruption qualified into the status.storytel.com page which is updated continuously for internal and external usage. This page covers the system status for the website, Android and iOS apps and 'other apps' and reports whether the degradation in service is operational, partially disrupted or full service disruption. The history can easily be analyzed using the updates

from when an issue is found, and measured until it is marked as resolved. This is not countable as failure in the sense that 'bad' code was released. These disruptions more often happen as a result of indexing, unforseen client calls or the size of databases. For example, the last couple of disruptions has been connected to the usage of the local server platform, which could be classified as a result of technical debt. Technical debt refers to the costs of additional rework caused by choices of temporary easier solutions instead of more time consuming but better approaches.

Option 1: Measure the lead time in Jira for bugs with priority critical or blocker
Option 2: Measure the lead time for the disruptions qualified into status.storytel.com
Option 3: Create a timeline for hotfixes based on information from slack.
Option 4: Create a timeline for requests of hotfixes based on information from slack

The first option was eligible due to its reasonable simplicity, and to avoid the time-consuming action to read through and manually evaluate Slack channels for information that would be required for option 3 and 4. This was carried out, but proved to not indicate failures that caused disruption of enough size. The second option was thereafter tested, but concluded faulty. Option 3 and 4 are still unexplored.

The Change Fail Rate is, similar to the Mean Time To Restore metric, particularly difficult to define as the central question becomes what constitutes a failure. If failure is defined as those deployments that require hotfixes, then the Change Fail Rate for Storytel's app services could be obtained by taking the total number of hotfixes performed divided by the total number of releases over the same period of time. A broader definition of failure would include not only performed hotfixes, but additionally requested and discussed hotfixes that were not released. As previously discussed, the difficulty in obtaining this data is that it is logged in different places - mainly Slack channels. Similar to the definition and method used for Mean Time to Restore, the Change Fail Rate was calculated by counting the number of critical and blocker bugs, then divided with the total number of tickets in Jira. Since we came to be mostly interested in a comparison between 2019 and 2020, the division of monthly throughput for all issues vs bugs was used instead of looking at the total number of issues in Jira.

Appendix 4. HR surveys

Survey	Sent 2019	Sent 2020
Diagnose and engagement drivers	week. 6, 14, 23, 32, 41, 49	-
Diversity	week. 20	week. 22
Vision & Mission	week. 18	week. 6
Wellbeing Tech	-	week. 8, 39, 48
Friday Package Tech	once a week	once a week
NPS	-	week. 10, 41
Personal development	week. 12, 45	week. 26
Leadership at Storytel	week. 8, 47	-
Leadership	-	week. 20
Feedback	-	week. 43
Team Efficiency Tech	week. 47	-
Tech work scope retro	week. 5, 9, 13, 18, 23, 32, 41	-
Working remotely	-	week. 14
Working remotely 2.0	-	week. 15, 16
Working remotely 2.2	-	week. 18, 21, 24, 26
Remote work at Storytel	-	week. 24

Appendix 5. Attempts of measuring Mean Time To Restore

First attempt

As discussed in section *6.2.1 Mean Time To Restore*, the first attempt to define a failure was at first in this thesis a bug labeled critical or blocker. From interviews, the expectation was that there was an upper limit to how long restore time was for Storytel's app related bugs. The limit was the length of one routine deployment cycle, e.g. three weeks. For example, when a request for a hotfix has been denied, the fix is instead released in the next scheduled deployment. Looking at the numbers from the system data (first attempt), this does not seem to cohere.

The Cycle Time Scatter plot was used, looking at the dataset with bugs, filtering out critical and blockers (420 bugs). Unlike the Delivery Lead Time metric and regular issues, labels (not referring to prioritization) are frequently used. This allows the analysis to be performed separately for app related bugs, labeled iOS or Android, and non-app related bugs. Reading the green line in figure 19 showing the average, the findings reveal that app related bugs are solved quickly in terms of coding (between one and two days, previously more than five days during 2019).

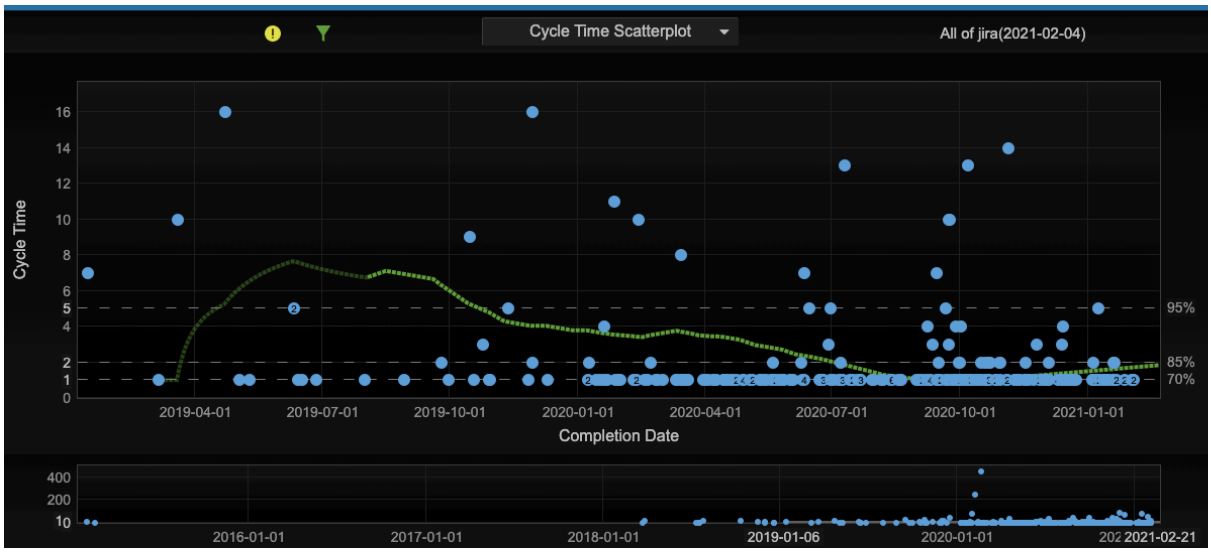

Figure 19. Cycle time scatterplot for critical and blocker app related bugs, workflow stages 'To Do' and 'In Progress' included.

From the corresponding graph over non-app related bugs, perhaps related to backend, it is found to take on average four days to solve bugs codewise. Reasons could be that non-app related bugs may be more complex and therefore more time consuming, or that there is less stress when the affected users are internal and the outage is less costly (in terms of money or customer satisfaction, for example through bad app ratings). However, when a solution exists, it takes less time for non-app related bugs to be deployed into production than for app related bugs. This gives an average cycle time for critical and blocker bugs of 44 days for finishing 85% of the bugs, see the green average line in Figure 20. These numbers involve the workflow stages from 'Backlog' to 'Done'. 'Backlog' and 'To Do' is added unlike in the Delivery Lead

Time metric, since theoretically a bug with high priority should not have to wait long before being picked up. Surprisingly, this is a lot more than one release interval.

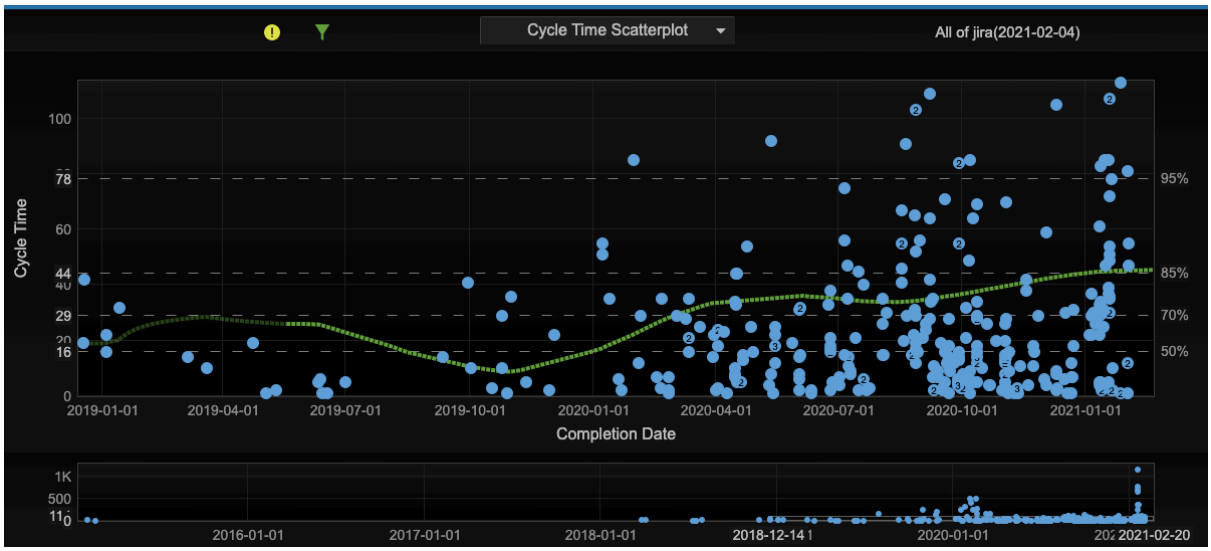

Figure 20. Cycle time scatterplot for critical and blockers, all bugs.

Furthermore, the trend has increased. The time bugs are in 'Backlog' has decreased (from about two weeks to one week), which could be due to the fact that the test club has a motivating influence on developers to fix bugs. However, the time for deployment has increased from slightly more than two weeks (steady during 2019) to almost one month (starting to increase in January 2020, see the green line advocating the average in Figure 21), resulting in a cumulative increase in cycle time. On one hand, the test club has made the number of filed bugs grow. One the other hand, there are a lot more developers available. Why especially the time to deploy bugs has increased has not been possible to investigate further due to time constraints.

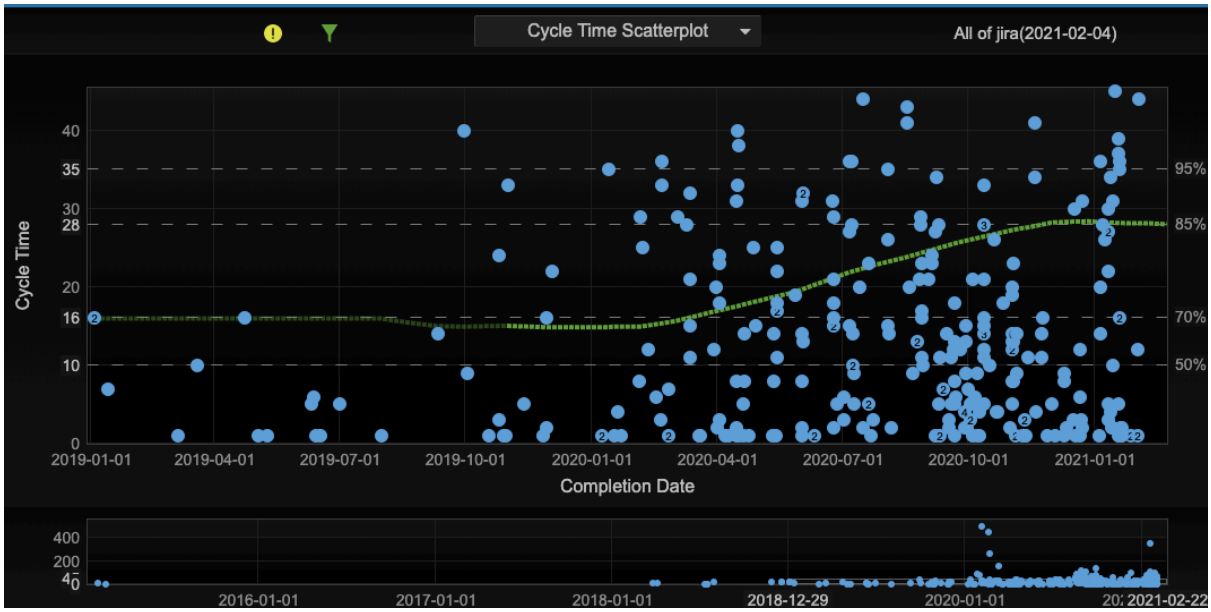

Figure 21. Cycle time scatterplot for critical and blocker app related bugs, not including 'Backlog'.

Since the Mean Time To Restore is much longer than expected, there are reasons to believe that the definition of a failure is faulty. This definition was chosen based on internal documentation stating that Blockers and Critical priority bugs are candidates for hotfixes, while others can wait for the next release train. According to theory, Mean Time To Restore should reflect 'how long it generally takes for a team to restore service when a service incident or a defect that impacts users occurs'. A challenging aspect of this metric is to evaluate what 'impacts a user'. It does not make sense that the Mean Time To Restore is 44 days while codewise fixing the bug generally does not exceed 5 days, if the majority of the critical and blocker bugs had a large user impact (by expectations, at least not longer than one release interval of three weeks). If this number was correct, Storytel would not even qualify as a Low Performer. Consequently, we conclude that the majority of critical and blocker bugs do not have a large enough user impact to serve as the definition of failure in the context of the Mean Time to Restore metric. We believe an inflation has occurred within the prioritization among bugs and that there are only some of the critical and blocker bugs that would have been interesting for us to analyse, however there are no labels that exist to seperate them from the others. Performing the same analysis and only including the blocker bugs is unfortunately still unexplored.

Second attempt

Since the scanning of slack channels to find the critical and blocker bugs that results in a request for hotfix or actual hotfix were not possible due to time constraints, the second attempt was to look into data where we knew the disruption and failure was large. However, this data does only involve the app-related services and were also found inefficient to make an estimation of Mean Time To Restore.

When there is a system disruption affecting a lot of users, it rarely has to do with a new app release - it is usually connected to server issues. If this happens it could be the case that even the customer support is not able to reach their systems, which is why there exists an external system outage log to use both internally and externally (Storytel Service Status Log, 2021). When looking at this history from 2016, see Figure 22, one can say that the average downtime is around 5h before restoration. Confirming that major system disruptions are generally much shorter, the median average downtime is 1h. Based on this data, the Storytel Tech Department would be a high performer if looking at average Mean Time To Restore, and close to elite if looking at the median.

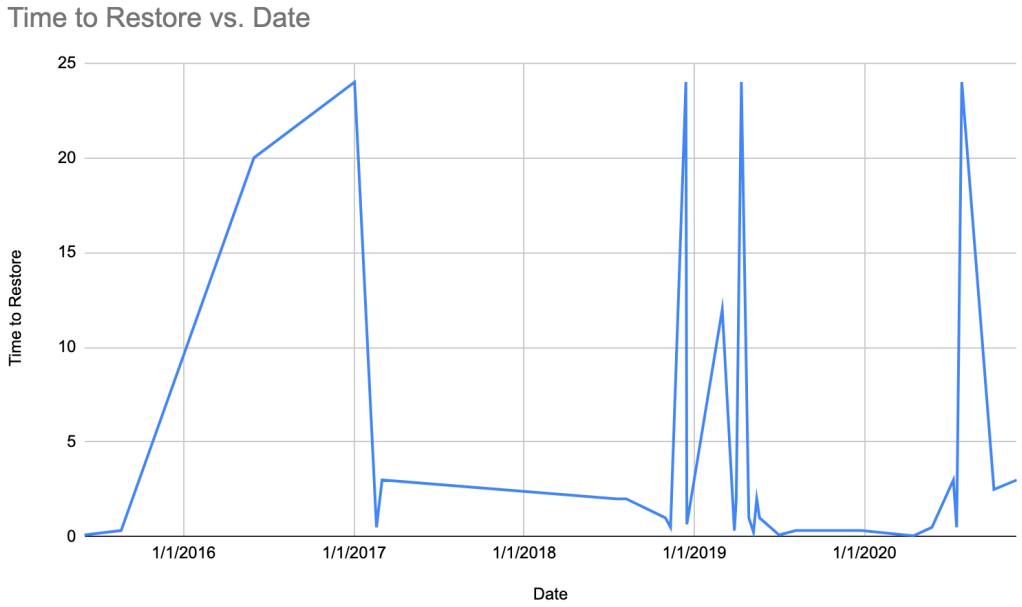

Figure 22. Mean Time To Restore according to disruption history at status.storytel.com

Even if this data source gives a result closer to the estimates in the survey data, it was concluded after discussion that these outages are almost always related to infrastructure outages outside of Storytel, for example connected to the server platform. In these situations the developers usually work together with the vendors to take care of the issues. But more often, issues are fixed by Storytels own developers - why this disruption data does not serve as a proper source to measure the Mean Time To Restore metric, and was also rejected.

If relying on interviews and internal documentation, that Mean Time To Restore would never exceed one release cycle of three weeks, and the Storytel Tech Department would therefore be a low performer.

Appendix 6. WIP per team

Discussions on what effects a high WIP can have on cycle time is interesting. As can be seen in Table 11, two teams have an average number of WIP items per employee exceeding 10. One of them has the second longest cycle time, while the other one has the second shortest cycle time. The average number of WIP items per employee is 5.85. This can confirm the statements from interviews that there are a lot of projects running in parallel. However, we can not confirm that this has a direct effect on cycle time. According to the literature review, the WIP should be kept low to increase Cycle Time but it is also affected by throughput (The Agileist, 2014). The table is sorted on Cycle Time in ascending order, and no correlation can be found towards the WIP/employee column.

Team	Cycle Time (days)	WIP (items)	Average number of WIP items per employee
Team A	9	168	11,2
Team B	11	18	1,8
Team C	12	106	7,5
Team D	14	52	3,25
Team E	15	71	3,9
Team F	16	26	3,25
Team G	16	18	1,2
Team H	21	79	4,9
Team I	21	172	7,8
Team J	21	254	17
Team K	29	13	2,6

Table 14. Overviewing Jira delivery cycle times, WIP, number of employees and the average number of WIP per employee.

Appendix 7. Historical perspective

During the last 12 months...

Question	N	Min	Max	Mean	Std. Deviation	Adjectives on Likert scale
How has communication changed within your team?	40	3	5	4.13	.563	*Worse/Better*
How has communication changed between teams?	40	2	4	2.93	.656	*Worse/Better*
How has your job satisfaction changed?	40	2	5	3.53	.816	*Decreased/Increased*
How has the responsibility distribution changed in your team?	39	2	5	3.62	.847	*Less clear/More clear*
How has the responsibility distribution changed in the Tech Department?	39	1	4	3.05	.916	*Less clear/More clear*
How has your team's ability to deploy features independently from other applications or services changed?	39	2	5	3.59	.993	*Decreased/Increased*
How has the ability to independently test an application without requiring an integrated environment changed?	38	1	5	3.21	.811	*Decreased/Increased*
In your opinion, how has Storytel's Tech Department's productivity changed?	39	1	5	3.56	.912	*Decreased/Increased*
How has your individual productivity changed?	39	1	5	3.56	.882	*Decreased/Increased*

Table 15. Summary of survey answers given in the 'Historical Perspective' section, answered by employees who have been employed at Storytel for 12 months or longer. The answers were given on a five-point Likert scale.

Appendix 8. Initiated analysis of throughput

Throughput as complementary data

The metric with the highest rating to assess productivity among developers in a study by Meyers et al. (2014) is "The number of work items (tasks, bugs) closed". This supports that complementary to the *Four Key Metrics,* it can be valuable to look at throughput. While the *Four Key Metrics* with tempo and stability categories are chosen to avoid measuring quantity in terms of lines of code or number of features, throughput still serves as an interesting foundation for discussion. Furthermore, why WIP and throughput are not metrics themselves is because cycle time can be directly derived from WIP and Throughput (The Agileist, 2014). Several measurements on the same aspect of the process would cause imbalance, why throughput is used as a reference, but not as a measurement. From looking at the throughput data in an exploratory point of view, it was found that monthly throughput has significantly increased over the last years for all types of issues.

Looking at the throughput run chart in Figure 23 based on all issues from all teams in Jira, it can be seen that throughput is steady during 2019 (the green line indicates an average on approximately 200 resolved issues per month). In 2020, this number increases rapidly to a new average at around 1200 resolved issues per month. The throughput decreases during the summer and winter holidays but is during 2020 never below 900 issues per month.

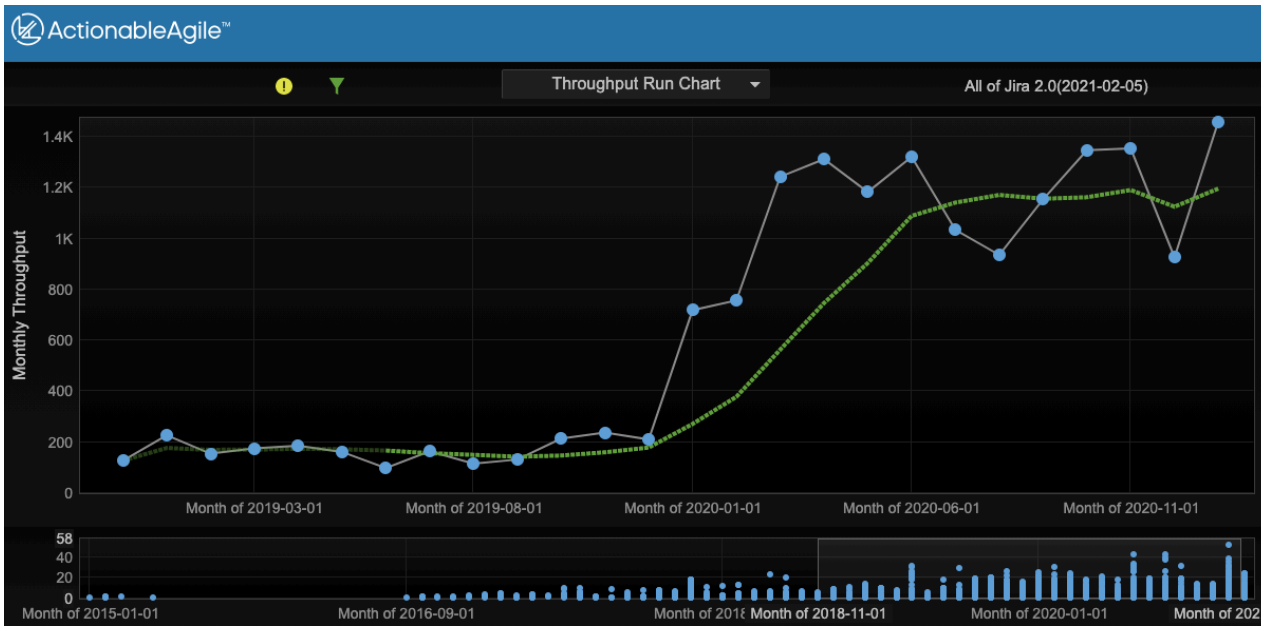

Figure 23. Graph showing the throughput run chart for all issues, for 2019-2020.

In Figure 24. the same type of data is presented but now it includes 2018. A similar but smaller increasing trend is shown between the year 2018 and 2019. The increases are not gradual, but steep. According to interviews, in 2018 a lot of focus was placed on maintenance and strategic development, rather than on developing new features. Meanwhile, competitors caught up with Storytel when it came to features. Therefore, in the next 1.5-2 years there was an increased focus on launching new markets and developing new features *(Interview 14: Developer, 2021).* These accounts seem to be aligned with what can be seen in the data from 2018-2019, assuming

that there are fewer tickets connected to strategic development. However, it seems insufficient to explain the large increase in throughput happening instantly in the beginning of 2020.

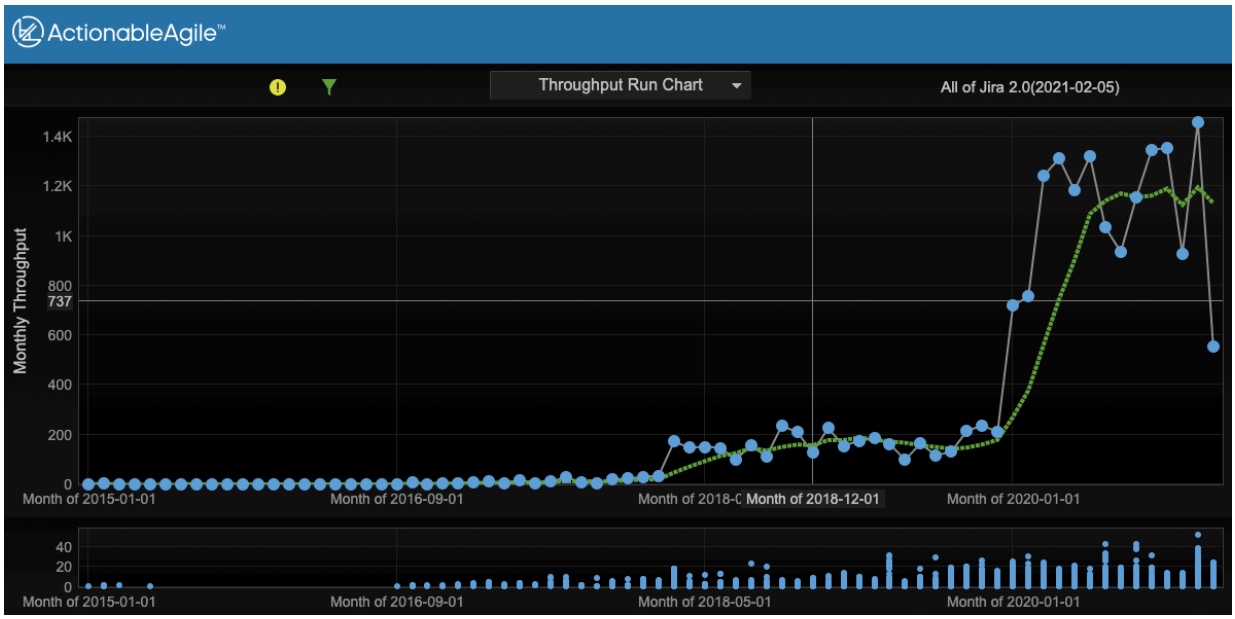

Figure 24. Graph showing the throughput run chart for all issues, including 2018-2020.

That the increase in resolved issues has only to do with the amount of recruited employees is dismissed, as the workforce is growing at a significantly slower pace than the amount of resolved issues, visualized in Figure 25. The number of employees are plotted in the same graph as the throughput for 2018 and 2020.

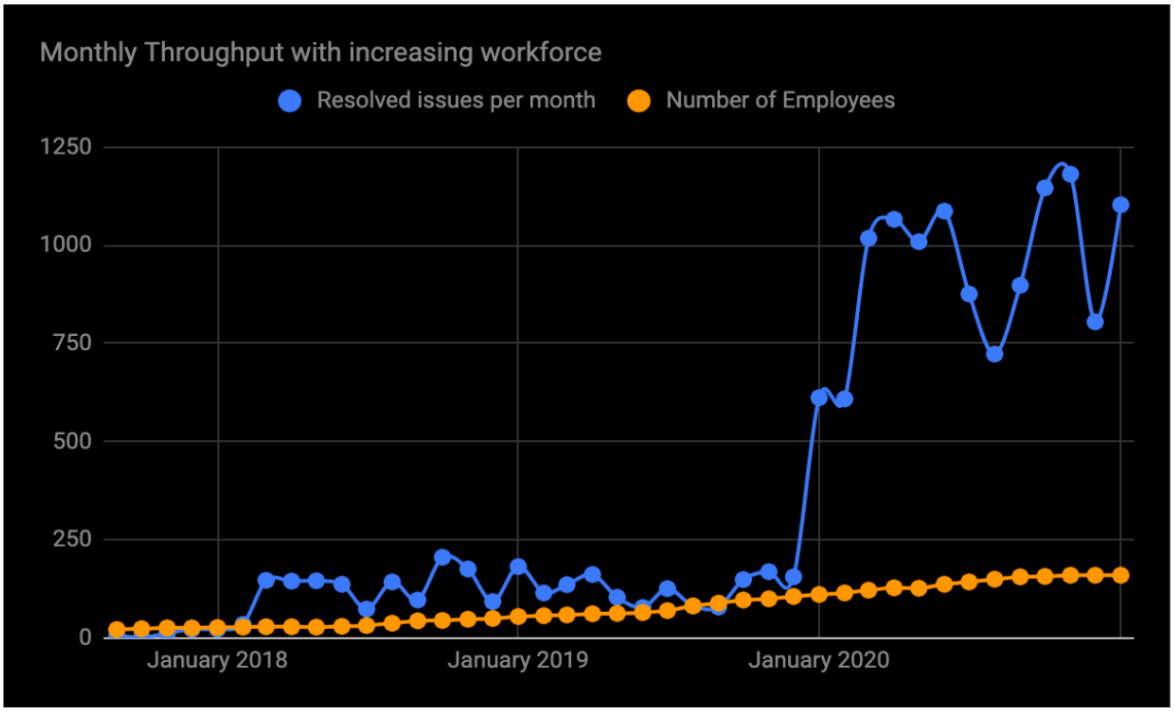

Figure 25. Resolved issues over time (excluding epics) plotted in the same graph as tech tracking, number of employees.

In Figure 26, 2020 is looked into in more detail, plotting the number of resolved issues per employee. Starting off low after winter holidays, in January and February the average is 5 issues per employee. It does not reflect the idea that a larger pile of issues are marked 'Done' and rolled out in deployment after winter holidays. In March 2020, the number of resolved issues reached an all time high, and this number held steady until summer at 8 issues per person. During summer, the number resolved issues are naturally lower due to employee vacations, but not lower than during January and February. After summer, September is starting off less steep before October when the number grows to 7, however not as high as during spring. During the winter holidays, the number of resolved issues per employee decreased again.

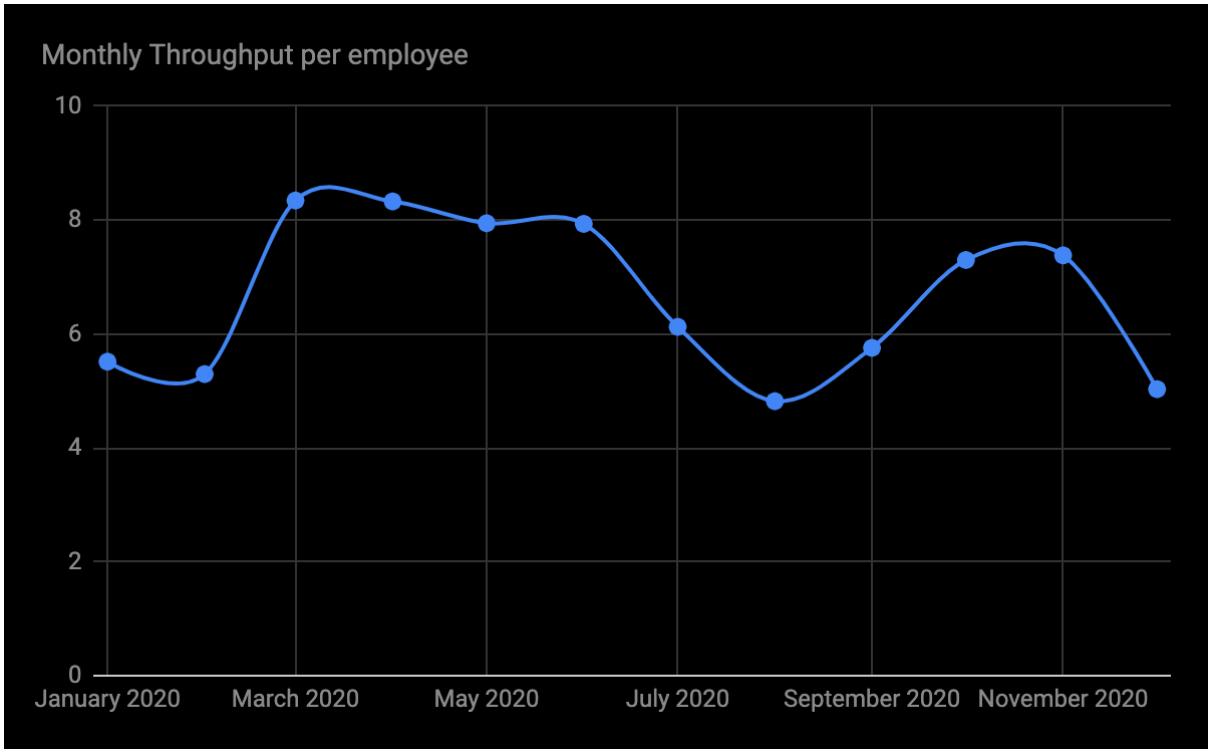

Figure 26. Resolved issues per employee during 2020

September is probably scoring lower since almost a whole week is spent on the yearly conference attended by all employees, and naturally fewer issues are resolved during this time. In January and February people were working at the office as usual. Interestingly, coinciding with the highest number of resolved issues per employee during 2020 in March, the global pandemic forces new routines to be implemented and all employees start working from home. The guidelines become less strict during summer, and during the autumn the approach is to let employees work at the office every other week according to schedule. Simultaneously, employees answer HR surveys about remote work, and almost half of the respondents estimate that their efficiency has improved while working from home, while the second largest majority indicate that it has remained the same. Only a relatively small percentage indicate that their efficiency has decreased while working from home. (Remote work at Storytel - HR Survey Data, 2020).

However, the major increase in resolved issues per person during 2020 is likely not solely caused by the implementation of remote work; the increase in resolved issues from January and February 2020 (before remote work was initialized), compared to 2019, supports that other reasons are involved, see Figure 27. One potential contributing factor is that the amount of teams have increased incrementally during 2019 and 2020. Going from being 3 teams to 11 could reasonably increase the amount of issues. Hypothetically, there is a larger need for transparency coming with an increased workforce and lots of new employees, that prompts smaller and more detailed tickets, i.e. more issues. Especially when working remotely and onboarding of new employees is performed in parallel. Investments in launching new markets and new features could also explain an overall increase, but not the big step between 2019 and 2020. These investments happened incrementally already during 2019.

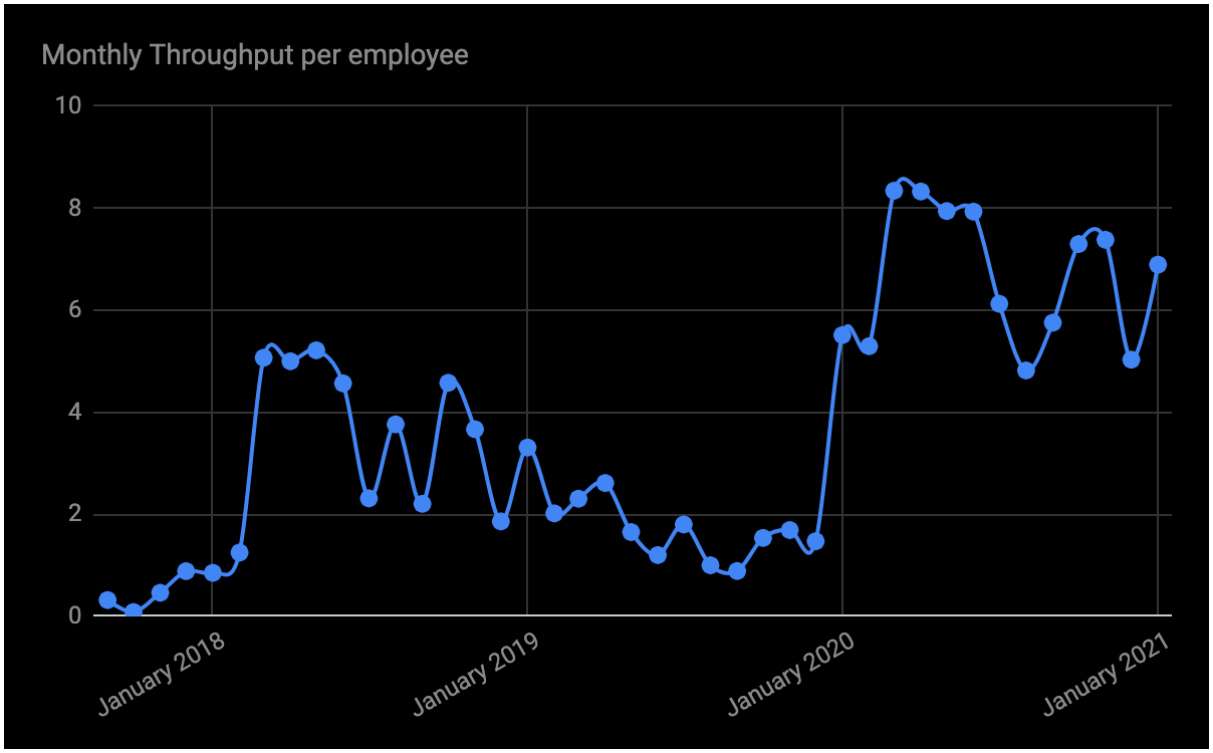

Figure 27. Resolved issues per employee during 2019-2020.

What can moreover be seen in Figure 27 is that the expectation that resolved issues per employee would have decreased along with the rapidly increasing workforce, is not supported. This was theorized due to studies showing that adding more people to software development projects generally makes them more time-consuming, because of the time it takes for new recruits to learn about the project and the complexity of the involved tasks, as well as the increased communication overhead (Brooks, 1995). The number of resolved issues per employee shows the opposite of this, with an increase from 2-4 issues per person during 2019 to between 6-8 during 2020. One interviewee mentioned that this might be due to synergy effects connected to the large increase in workforce, which seems to be supported *(Interview 14: Developer, 2021).*

Apart from a large WIP per person (see *Appendix 6. WIP per team)* among other things showing that Storytel has a lot of projects running in parallel, the high throughput shows that the amount of finished issues each month have also increased. From Little's law (The Agilist, 2014) we define the relation between WIP, throughput and cycle time (see section *3.5 Throughput and finding bottlenecks)*. Conclusively, at Storytel the throughput per employee has increased significantly between 2019 and 2020, but the amount of WIP per employee has grown at an even faster pace - resulting in a longer cycle time and in turn an increased Delivery Lead Time.

Factors influencing Throughput

We have only been able to hypothesize what factors might have influenced the increased level of throughput at Storytel, without investigating these hypotheses further through additional data collection due to time constraints. These hypotheses are described below.

Working remotely

The increased degree of remote work due to the pandemic is hypothesised by us to have had a positive impact on the level of throughput at Storytel. As described in Section *3.3.2 Environment Factors*, introducing remote work can increase performance in some task areas, especially in individual tasks that require a larger degree of concentration (Bloom et. al., 2014). This hypothesis could be further investigated through HR Survey Data, as several surveys were sent out to Storytel employees in 2020 regarding working remotely during the COVID-19 pandemic.

Along with the pandemic, guidelines for almost all Storytel employees were to work from home during March until August. Then a few months followed where employees were allowed at the offices every other week according to schedule. In November 2020 the guidelines were tightened again and remote work was the only option for the employees of the Tech Department. Several surveys were sent out to the employees along with the transition to remote work. This data shows that stress levels decreased in level in the beginning of when employees started working remotely; however the stress level increase again later during the year. The top five feelings reported by employees about working remotely are, in order: productive, focused, lonely, efficient and relaxed. Furthermore, the findings show that while a small percentage of employees were working remotely already before the pandemic, a majority of respondents want to continue working from home in the future, to some degree. As previously mentioned, compared to office work, almost half of the respondents report that they experience an increase in efficiency. (Remote work - HR Survey Data, 2020).

In summary, the remote transition has been rather smooth for Storytel since a lot of teams were distributed among different offices already, and a large percentage of employees report that they feel more efficient and productive. Respondents mention the ease of booking meetings and speedier decision making due to not being limited by booking physical meeting rooms, the positive aspects of reduced commute time, fewer distractions, and easier time management. However, the most frequently mentioned downsides are the negative aspects of less social interaction with co-workers, difficulties in limiting work hours, as well as some people reporting being more distracted rather than less when working from home. (Remote work- HR

Survey Data, 2020). Overall, working remotely may be a contributing factor to the increase in throughput, however it is difficult to measure.

E-factor

A potential reason as to why remote work has had an impact on throughput, is because it offers more uninterrupted hours and allows employees to get into a 'flow'. It is argued that when there is a low number of uninterrupted hours in proportion to total hours, approximately below 40%, this can imply reduced effectiveness and frustration among employees. A number above 40% indicates an environment that allows employees to get into a flow they need to (Demarco and Lister, 1987). Looking at the data from Storytels Tech Department, the calculated E-factor equals 46%. Unfortunately, there are no numbers outside of our survey to compare the E-factor historically. It would have been interesting to know whether or not the current E-factor of 46% is now above the benchmark due to remote work, and that it was below 40% before working remotely - or if it has never been below the benchmark. The survey question we do have regarding preferred uninterrupted hours compared with actual uninterrupted hours, still implies that employees generally desire more uninterrupted hours than they currently have. However, if the E-factor passed the benchmark of more uninterrupted hours during working remotely, it would be a potential explanation for the increased throughput.

There is a balance in regarding what facilitates communication and what can be classified as a distraction. Several interviews confirm that the need of flow is important. Different solutions for how to achieve flow are reported - including scheduling focus hours, reading emails on scheduled times (as seldom as twice a week) and approval to occasionally skip daily standups if it disturbs flow *(Interview 3: Developer, 2020; Interview 12: Developer, 2021; Interview 13: Crew Coach, 2020; Interview 14: Developer, 2021).* However, they mention the difficulties in how to manage Slack; to keep it from becoming a distraction, while also being reachable. It is easier for management to encourage always being up to date with the latest posted information (in more than one place) rather than to limit distractions. While there is an expectation to be approachable, it is left for each employee's own responsibility to for example actively pause and mute notifications from Slack *(Interview 14: Developer, 2021).*

Appendix 9. Results from Factor Analysis

Factor 1: ***Team Identity***
TI1 - My team is collectively working toward the same goal. (.806)
TI2 - I know the reason for all features developed in my team. (.483)
C1 - Communication is efficient in my team. (.391)
Factor 2: ***Number of projects***
NP2 - How many projects have you worked on during the last three months? (.911)
NP3 - How many projects have your team been involved in during the last three months? (.654)
Factor 3: ***Transformational Leadership***
TL3 - My manager regularly gives med actionable feedback. (-.910)
TL1 - My manager challenges me to see problems from new perspectives. (-.849)
TL2 - My manager notices me. (-.696)
Factor 4: ***Generative Culture***
GC5 - In my team, failure causes inquiry so that we can learn from the experience. (-.775)
GC4 - In my team, cross-functional collaboration is encouraged and rewarded. (-.480)
GC6 - In my team, new ideas are welcome. (-.451)
GC1 - In my team, information is actively sought. (-.443)
Factor 5: ***Job Satisfaction***
JS2 - I would recommend my team as a place to work. (-.872)
JS1 - I would recommend my workplace as a place to work. (-.651)
TI3 - I am proud to be a part of my team. (-.495)
Factor 6: ***Efficiency*** **(*****Automation and Shared Responsibility)***
TC1 - In my team we put effort into facilitating work for other teams. (.759)
AU1 - What is, in your estimate, the percentage of tasks related to deployment that are automated in your team? (.649)
AU2 - What is, in your estimate, the percentage of tasks related to testing that are automated in your team? (.529)
GC2 - In my team, responsibilities are shared. (.519)
LM2 - In my team the ambition is to keep the number of WIP to a minimum. (.376)
Factor 8: ***E-factor and Time Fragmentation***
EF3 - Fraction Desired Uninterrupted Hours (-.558)
EF1 - E-Factor (-.367)
Factor 9: ***Team Cohesion***
TC3 - I wish I had more insight into what other teams are doing. (.725)
TC2 - I have a good insight into what other teams are doing. (-.352)

Appendix 10. Overview of Survey Responses

Item	Item code	N	Min	Max	Mean	Std. Deviation	Skewness	Kurtosis
In my team information is actively sought.	GC1	75	2	5	4.13	.777	-.593	-.072
In my team responsibilities are shared.	GC2	75	2	5	4.27	.875	-1.176	.823
In my team cross functional collaboration is encouraged and rewarded.	GC4	73	1	5	4.15	.953	-1.004	.586
In my team failure causes inquiry so that we learn from the experience.	GC5	75	2	5	4.16	.839	-.736	-.108
In my team new ideas are welcomed.	GC6	75	2	5	4.61	.655	-1.765	3.026
In my team we put effort into facilitating work for other teams.	TC1	75	1	5	3.61	.943	-.138	-.393
I have a good insight into what other teams are doing.	TC2	75	1	5	2.68	1.080	.345	-.322
I wish I had more insight into what other teams are doing.	TC3	75	2	5	3.96	.845	-.475	-.336
I would recommend my organization as a place to work.	JS1	75	3	5	4.64	.584	-1.404	1.018
I would recommend my team as a place to work.	JS2	75	1	5	4.59	.773	-2.374	6.700
My manager challenges me to see problems from new perspectives.	TL1	74	1	5	3.50	1.010	.000	-.377
My manager notices me.	TL2	75	1	5	3.99	1.121	-1.156	.797
My manager regularly gives actionable feedback that helps me improve my performance.	TL3	75	1	5	3.20	1.241	-.261	-.839
I know the reason for all features we develop in my team.	TI1	75	1	5	4.13	1.107	-1.253	.893

My team is collectively working towards the same goals.	TI2	75	1	5	4.33	.875	-1.584	2.778
I am proud of being a part of my team.	TI3	75	2	5	4.65	.688	1.985	3.249
Communication is efficient in my team.	C1	75	1	5	3.80	.959	-.813	.645
How often do you interact with members from other teams for inspiration and/or assistance for a task you are working on?	C2	75	1	5	3.25	1.152	-.789	-.155
How many projects have you worked on during the last three months?	NP2	73	1	6	2.25	1.310	1.432	1.813
How many projects has your team been involved in during the last three months?	NP3	68	1	6	2.66	1.410	.794	.199
E-Factor	EF1	75	.10	.88	.4613	.22446	.100	-1.002
Switching between tasks can be good in terms of being productive.	EF2	75	1	5	2.15	1.023	.632	-.333
Fraction Desired Uninterrupted Hours	EF3	73	.13	4.00	.9225	.58981	2.372	9.371
Features developed in my team can be tested and deployed without being dependent on other teams.	AR1	73	1	5	3.86	1.032	-.967	.771
Security testing is generally done during the early phases of development.	AR2	72	1	5	2.33	.949	-.011	-.541
What is, in your estimate, the percentage of tasks related to deployment that are automated in your team?	AU1	59	1	5	2.93	1.244	-.257	-.893
What is, in your estimate, the percentage of tasks related to testing that are automated in your team?	AU2	58	1	5	2.55	1.216	.027	-1.378

I have access to visual displays showing the status and/or flow of work within my team by some metrics.	LM1	70	1	5	3.01	1.257	-.208	-.914
In my team, the ambition is to keep the number of WIP (work in progress) to a minimum.	LM2	71	1	5	3.27	1.095	.048	-.664