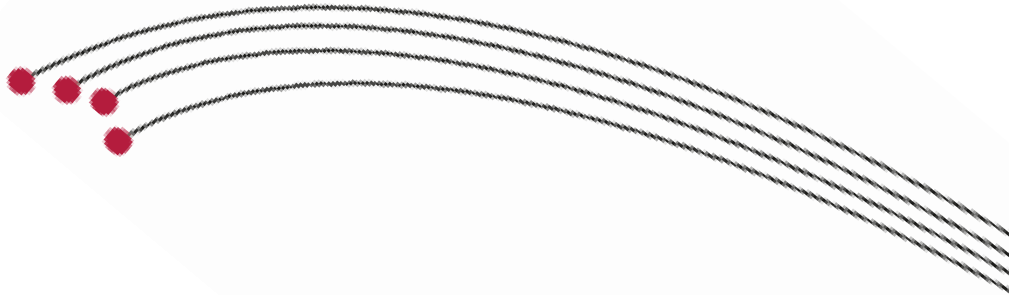
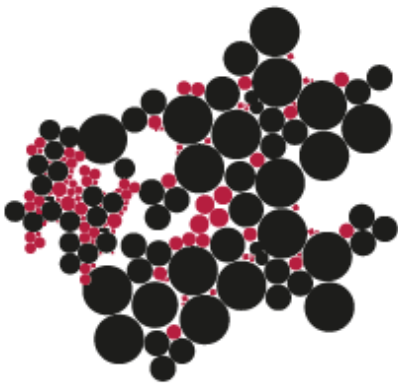


**Analysing and visualising data to improve the productivity level
of an Agile organised company**



Bachelor thesis

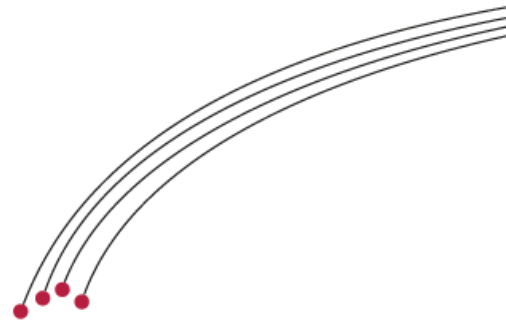
Ceret van der Vegt

Industrial Engineering & Management

Bachelor

University of Twente

02-07-2021



A.S. Watson Group

A member of CK Hutchison Holdings



Analysing and visualising data to improve the productivity level of an Agile organised company

Author

Ceret van der Vegt
S2145677
BSc Industrial Engineering & Management

A.S. Watson Group

Nijborg 17
3927 DA Renswoude
The Netherlands

Supervisor A.S. Watson Group

Mr. B. Jekel
A.S. Watson Group IT Europe
Product owner

University of Twente

Drienerlolaan 5
7522 NB Enschede
Netherlands

First supervisor

prof. dr. J. van Hillegersberg
Faculty of BMS, IEBIS

Second supervisor

dr. G. Sedrakyan
Faculty of BMS, IEBIS



**UNIVERSITY
OF TWENTE.**

Preface

This document contains my bachelor thesis on “how to analyse and visualise data to improve the productivity level of an Agile organised company”, to complete the bachelor Industrial Engineering and Management at the University of Twente.

I would like to thank my company supervisor Bram Jekel for supporting me every Friday morning during my research. I would like to thank Laurens Priemis for giving me the opportunity to do my graduation at the CRM Tribe of A.S. Watson. Also a special thanks to Menno Noorloos, who supported me during my research on the IT systems and different coding parts.

Furthermore, I would like to thank my supervisor Jos van Hillegersberg from the University of Twente, for sharing his enthusiasm about the subject and providing me feedback. Moreover, I would like to thank Gayane Sedrakyan for providing me feedback on my concept version to bring my research to a higher level. Besides my supervisors, a special thanks to my fellow student Lisa Nonhof, for stimulating each other and for her suggestions on improving my research.

Ceret van der Vegt

Management summary

Introduction

AS Watson Group is the world's largest international health and beauty retailer. The Group is a member of CK Hutchison Holdings, also known as Hutchison Whampoa. A.S. Watson operates in four different sectors: health & beauty, luxury perfumeries & cosmetics, food, electronics & wine and last beverages. The CRM department, part of Group IT Europe, is responsible to deal with the software related changing needs of its customers to put a smile on its customers' faces.

Research is conducted within this CRM department. The CRM department is continuously improving the process to ensure their services are matching the changing techniques and needs of their internal as well as external customers as closely as possible. Having an optimal productivity level is important to ensure high quality of their services in this dynamic environment. To be able to respond adequately to those changing needs, a clear overview of different processes is necessary. However, A.S. Watson is not aware of which problems are affecting the productivity level. Currently, decisions to react on changing needs are made based on feelings instead of on data. Therefore, a dashboard with a set of metrics suitable for the CRM Tribe is necessary to identify the problems affecting the productivity level to be able to ensure high quality solutions.

Research approach

To identify a proper solution for the company, different research methods were used. A company analysis is conducted by doing a bottleneck analysis with process mining and by having conversations with employees, a survey is held, literature reviews are conducted, data is analysed, and the own expertise is used.

The solution

During the research, the following is found to solve the problem:

- 1) Important bottlenecks to monitor
- 2) Desired improvements by employees
- 3) Suggestions on increasing productivity by literature
- 4) Visualisation suggestions on tools and charts
- 5) Optimal set of metrics with explanations
- 6) Recommended dashboards

Conclusions & recommendations

A combination of the findings above results in a recommendation for a CRM Tribe (department) dashboard with concerning CRM Squads (teams) dashboards. The bottleneck analysis provides the input which is important to focus on, together with the desires from the employees and the suggestions from literature. As a result, the recommended set of metrics focus on quality, productivity, and performance. Five tools are selected as suitable and several recommendations are provided on how to visualise the data on the dashboard. A demo dashboard is provided to visualise the solution.

By increasing the number of available indicators to a well-determined set of metrics and by increasing the awareness of the need for data visualisation, the core problem has been solved.

Table of Contents

Reader's guide	7
Definitions	8
List of figures	9
List of tables	10
1. Introduction	11
1.1. Introduction to A.S. Watson Group	11
1.1.1. Scope of the research.....	12
1.2. Problem identification	12
1.2.1. Reason for research	13
1.2.2. Problem statement	13
1.2.3. Core problem.....	13
1.3. Problem solving approach	14
1.3.1. Research methodology.....	14
1.3.2. Research goal	14
1.3.3. Research questions	15
1.4. Problem quantification	17
2. Company analysis	18
2.1. The current work process	18
2.1.1. CRM in Group IT of A.S. Watson	18
2.1.2. The Agile principles	19
2.1.3. The A.S. Watson model	22
2.2. Problem analysis	26
2.2.1. The bottlenecks	26
2.2.2. The productivity level	37
2.2.3. The bottlenecks and the actors involved.....	38
2.3. Desired situation.....	38
2.3.1. Desired metrics.....	38
2.3.2. Survey	39
3. Theory on improving productivity level of an Agile organised company	40
3.1. Agile optimisations on improving productivity suggested in research	40
3.1.1. Systematic Literature Review	40
3.1.2. Suggestions on the framework.....	47
3.2. Suggested indicators by research	48
3.2.1. Suggestions from literature review.....	48
3.2.2. Suggestions by experts	50

3.2.3. Results	50
3.3. Data visualisation	51
3.3.1. Analysis and visualisation	51
3.3.2. The design	54
3.3.3. Implementation	55
4. Improving the productivity level of the CRM Tribe by using suggested metrics	56
4.1. The metrics.....	56
4.2. The dashboard	58
4.3. Evaluation	59
5. Conclusion, recommendations & limitations.....	60
5.1. Conclusion.....	60
5.2. Recommendations	61
5.3. Limitations.....	63
References	64
Appendix	67
Appendix A Agile Manifesto	67
Appendix B Data analysis	68
Jira report cumulative flow diagram.....	68
Jira report control chart	68
Jira report average age	69
Jira report resolution time.....	70
Script.....	71
Disco analysis	72
Appendix C Survey.....	73
Appendix D Systematic Literature Review	77
SLR – 1	77
SLR – 2	81
Appendix E Data visualisation.....	84
Jira.....	84
Python	85
Appendix F Dashboard	89

Reader's guide

The research on analysing and visualising data to improve the productivity of an Agile organised company is structured in five chapters. To explain the structure of the report, a brief introduction per chapter is provided below.

Chapter 1. Introduction

Chapter 1 provides all details to understand the company, the problems there are, the reason why it needs to be researched and the research strategy. This chapter provides the approach taken to solve the core problem, and research questions are provided to achieve the research goal.

Chapter 2. Company analysis

For analysis and visualisation, it is important to have an overview of the bottlenecks. Chapter 2 provides a clear explanation of the principles of the department in which research takes place. An insight into the bottlenecks is given by analysing available reports and by applying the technique process mining. The needs of the company are investigated in this chapter as well. The results from the bottleneck analysis and the needs addressed by the employees serve as important input for the solution.

Chapter 3. Theory on improving productivity level of an Agile organised company

Chapter 3 provides the suggestions and theory from literature needing to improve the productivity level. More important, it indicates which indicators are important for data visualisation. The way data should be visualised within a company using the Agile principle is addressed as well within this chapter.

Chapter 4. Improving the productivity level of the CRM Tribe by using suggested metrics

This chapter provides the solution on how to solve the core problem together with a demo dashboard.

Chapter 5. Conclusion, recommendations & limitations

This chapter concludes whether the solution solves the core problem and to what extent the norm set by the company has been achieved. Recommendations are given and limitations are explained.

Definitions

Definition	Explanation
CRM	Customer Relationship Management
CRM Tribe	Name of CRM department in Agile terms
Business unit (BU)	For A.S. Watson a group of customers (e.g. Kruidvat) doing a request
Ticket	Another term for the work item that must be addressed
Issue	Agile term which could represent a story, bug, or task in a project
Topdesk	Software service desk
GIC Project	All tickets related to CRM software changes
Native integration	A pair of applications provide direct integration with one other via APIs
Real-time dashboard	Type of visualisation automatically updating the most current data available
Metric	A quantifiable measure for comparing and tracking performance

List of figures

- Figure 1: Markets in which A.S. Watson operates (A.S. Watson, 2020)11
- Figure 2: A.S. Watson’s brands11
- Figure 3: Organogram12
- Figure 4: Problem cluster13
- Figure 5: Problem solving approach14
- Figure 6: Overview of the research questions16
- Figure 7: Reality vs Norm17
- Figure 8: Scrum framework20
- Figure 9: Scaling Agile at Spotify (Cruth, 2021)21
- Figure 10: Nexus framework (Scrum.org, 2021)21
- Figure 11: Roots of Agile22
- Figure 12: Organisation of CRM Tribe22
- Figure 13: Main roles22
- Figure 14: Scrum process23
- Figure 15: Incoming request24
- Figure 16: Process structure24
- Figure 17: Change management process flow25
- Figure 18: Incoming request flow25
- Figure 19: Cumulative flow diagram (CFD)26
- Figure 20: Control chart27
- Figure 21: Average age report28
- Figure 22: Resolution time report28
- Figure 23: Input in Advanced Report29
- Figure 24: Process flow Jira30
- Figure 25: Animation of the flow generated by Disco (100% activities, 0% paths)31
- Figure 26: Control-flow perspective (left) and performance perspective (right) of the process model generated by Disco32
- Figure 27: Case duration generated by Disco for tickets between 2016-202133
- Figure 28: Events per case generated by Disco (10 events) for tickets between 2016-202133
- Figure 29: Case duration generated by Disco between 2019-202134
- Figure 30: Events per case generated by Disco (10 events) for tickets between 2019-202134
- Figure 31: Example of data uncertainty35
- Figure 32: Problems to visualise36
- Figure 33: Currently available metrics37
- Figure 34: Desired metrics39
- Figure 35: DevOps cycle (Atlassian, 2019)47
- Figure 36: Result of exporting data from Jira51
- Figure 37: Format of exported data with python52
- Figure 38: Dashboard design guidelines54
- Figure 39: Overview of charts55
- Figure 40: Recommended metrics56
- Figure 41: Metrics for CRM Tribe dashboard58
- Figure 42: Metrics for CRM Tribe dashboard58
- Figure 43: Demo dashboard59
- Figure 44: Cumulative flow diagram68
- Figure 45: Control chart68
- Figure 46: Average age report69
- Figure 47: Resolution time70
- Figure 48: Jira Script for data extraction71
- Figure 49: Control-flow perspective generated by Disco (100% activities, 100% path)72
- Figure 50: Overview of tickets in Jira)84
- Figure 51: Anaconda platform85
- Figure 52: Installing jira library85
- Figure 53: The dashboard89
- Figure 54: Release health metric90
- Figure 55: High priority issues91
- Figure 56: Days in status metric91
- Figure 57: Lead time metric91
- Figure 58: Average age metric92
- Figure 59: Cumulative flow metric92

List of tables

- Table 1: Factors affecting productivity (Fate ma & Sakib, 2018)41
- Table 2: Metrics48
- Table 3: Metrics classified in four perspectives49
- Table 4: Set of Metrics50
- Table 5: Balanced scorecard for dashboard tools.....53
- Table 6: Results of balanced scorecard53
- Table 7: Inclusion and Exclusion criteria77
- Table 8: Search terms78
- Table 9: Initial search results78
- Table 10: Results79
- Table 11: Inclusion and Exclusion criteria81
- Table 12: Search terms82
- Table 13: Initial results82
- Table 14: Final results83

1. Introduction

In the first chapter, an introduction to the company and its problem is provided. The approach for solving the problem is given by explaining the research questions to be solved.

1.1. Introduction to A.S. Watson Group

A.S. Watson Group is the world's largest international health and beauty retailer. The Group is a member of CK Hutchison Holdings, also known as Hutchison Whampoa. The holding has four core businesses in 50 countries: ports and related services, telecommunications, infrastructure and retail (Atlassian, 2021a). This research is focused on the retail sector, represented by A.S. Watson Group. The group has over 16,000 stores located in Asia & Europe, visualised in **Figure 1** (Watson, 2020).



Figure 1: Markets in which A.S. Watson operates (A.S. Watson, 2020)

A.S. Watson operates in four different sectors: health & beauty, luxury perfumeries & cosmetics, food, electronics & wine, and last beverages. The different sectors are divided into different brands, also called business units. An example of one of the business units is Kruidvat, which is probably familiar to almost everyone in the Netherlands. To understand the Group, the different business units are visualised in **Figure 2** (Watson, 2020).

Health & Beauty



Luxury Perfumeries & Cosmetics



Food, Electronics & Wine



Beverages



Figure 2: A.S. Watson's brands

1.1.1. Scope of the research

Since A.S. Watson is a large company, research is conducted within one department of the Group. With the help of **Figure 3**, the scope of the research becomes clear. As depicted, A.S. Watson Group consists of an E lab, ASW Benelux and Group IT. The latter is divided into two continents; Asia and Europe. Research is focused on Group IT Europe, specifically within the Customer Relationship Management (CRM) department.

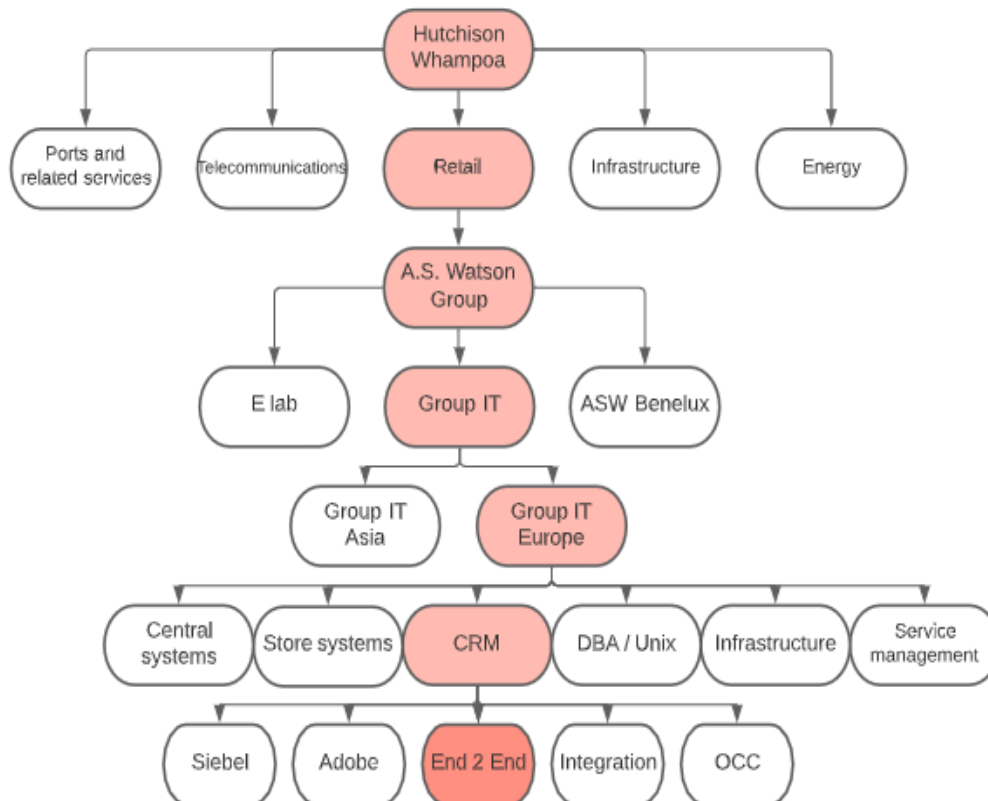


Figure 3: Organogram

1.2. Problem identification

The problem occurs in the CRM department, organised based on the Agile principle. Therefore, the CRM department is further called the CRM Tribe. The CRM Tribe is responsible for handling software related issues and requests from the Business Units. The request or issue, also called 'ticket', comes across several phases within the CRM Tribe. Because the CRM Tribe is Agile organised, all phases are related to each other. One problem in a specific phase could therefore cause another problem in a different phase. There are two types of problems, the knowns and the unknowns. On the one hand, A.S. Watson is aware of the fact that some factors or actions are having a consequence on the productivity level. These actions are the known problems. An example is the lack of communication between the different development teams concerning a cross-team ticket. On the other hand, there are actions affecting the productivity level the company is not aware of. Those are the unknown problems. The consequences of both the known and unknown problems are for example long lead times and missed agreed deadlines, all having an impact on the productivity level.

1.2.1. Reason for research

The main characteristic of Agile is the iterative approach. Terms like continuous improvement and responding quick and easy describe the approach. The CRM Tribe is continuously improving the process to ensure their services are matching the changing techniques and needs of their internal as well as external customers as closely as possible. Having an optimal productivity level is important to ensure high quality of their services in this dynamic environment. Therefore, it is important to identify the problems affecting the productivity level first. Besides, it is of importance to identify indicators to keep track of the problems. When having a clear picture of the indicators affecting the productivity level, adequate response to changes is possible.

1.2.2. Problem statement

Within the software industry, it is important to meet the changing needs of the customers. To be able to respond adequately to those changing needs, the productivity of the process should be optimal. However, A.S. Watson is not aware of which known and unknown problems are affecting the productivity level. Therefore, the action problem is: “the productivity level is lower than desired.” To deal with the action problem, it is important to identify both the known and unknown problems to increase productivity. Currently, there is no insight into those important factors. When having no insight, it is hard to make decisions to react to problems or changes. Since the CRM Tribe is Agile organised, all phases are related. One problem arising in a specific phase is causing other problems in different phases. To visualise the problems, a problem cluster is created which is depicted in **Figure 4**.

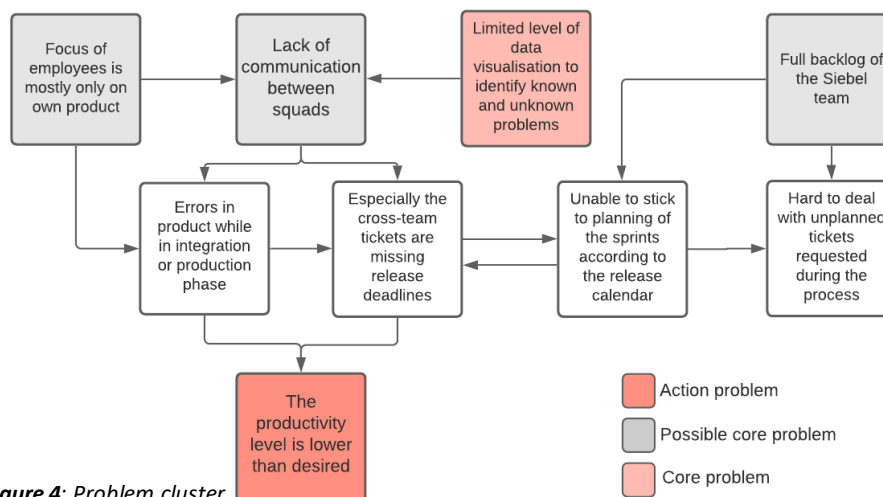


Figure 4: Problem cluster

1.2.3. Core problem

From the problem cluster can be concluded that there are several problems all causing a lower productivity level than desired. However, it is unclear to what extent the problems are having an impact on the process, which is a problem to be able to increase productivity. It is proven in literature that data visualisation, instead of using all the raw data which is available, provides a quick insight into the problems to explore, discover, summarise and present, such that adequate response is possible (Sedrakyan et al., 2019). Not having visualisations can cause a lack of overview. To solve the action problem and thus improve the productivity level, a clear insight into the extent of the problems is necessary, which is currently lacking because of the few visualisations available. Following the problem cluster, the limited level of data visualisation is a problem that needs to be solved, leading to the following core problem:

“Limited level of data visualisation to identify known and unknown problems”

1.3. Problem solving approach

A.S. Watson is already using Jira to manage their projects. Jira is the tool for Agile project management and provides possibilities for using metrics. These metrics help to identify known and unknown problems. Therefore, it is chosen, together with the company, to design a dashboard as a solution to solve the core problem. This section explains what steps are taken to solve the problem.

1.3.1. Research methodology

To solve the core problem of having a limited level of data visualisation to identify known and unknown problems, the Managerial Problem-Solving Method (MPSM) is used. The MPSM is a systematic approach by Heerkens and Winden (2017) to solve business problems handled in their organisational context. The method consists of seven phases, of which the first one “defining the problem” has already been conducted in the second paragraph.

Phase 2, formulating the approach

A scheme is created and visualised in **Figure 5** to show the approach of this research.

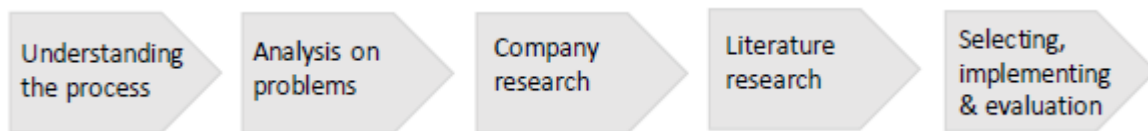


Figure 5: Problem solving approach

Phase 3, analysing the problem

After having an understanding of the process, an extensive analysis of the problem is conducted to completely analyse the process and its problems.

Phase 4, formulating alternative solutions

Key in this research is to identify indicators to track productivity. However, alternative solutions will be researched as well by doing literature research on suggestions from the literature on Agile.

Phase 5, choosing a solution

After doing research, the most suitable solutions will be chosen using the balanced scorecard method to determine about the tool. Besides, the most suitable options are chosen based on the research which will serve as input for data visualisation.

Phase 6, implementing the solution

Implementation of the solutions in the company takes place during this phase of the MPSM.

Phase 7, evaluation of solution

To determine whether the productivity level increased and to determine if the company is able to respond better to the changing needs of their customers.

1.3.2. Research goal

The goal of this research is to get insight into the most suitable indicators for Agile organised companies to identify known and more important, unknown problems. The goal is to increase productivity by solving this problem, so the company can respond better to problems and changing needs. When the company is aware of the problems, an adequate response is possible. After selecting the most important indicators, this information can be visualised on a dashboard, so the company is able to track the process by focusing on important indicators.

1.3.3. Research questions

To solve the core problem of having a limited level of data visualisation to solve known and unknown problems, the following research question is formulated.

“How to analyse and visualise data to improve the productivity level of an Agile organised company?”

The following set of sub-research questions is defined to answer the main research question.

1) How is the work process of the CRM Tribe organised?

- a. What is the meaning of CRM within Group IT Europe of A.S. Watson?
- b. What are the Agile principles on which the Tribe is operating?
- c. How is the process of the A.S. Watson model organised?

First, insight into the environment in which research takes place is necessary to understand the process. The problem analysis can only be conducted correctly if the context is properly understood. It is important here to understand the main tasks of the CRM tribe, how Agile works and last the different flows of the process. Research is conducted with the use of sources, available on the Confluence page of the company. Besides, information is gathered by communicating with employees and by observing the processes.

2) Where in the process are the bottlenecks?

- a. Which bottlenecks are there?
- b. What is the current productivity level?
- c. How can these bottlenecks be explained by the actors involved?

Second, it is important to have a view on the current situation of the productivity level. Therefore, it is important to identify the bottlenecks affecting it to make a problem analysis. By researching the problems, it will become clear what factors are affecting the productivity level. This is an important indication of where to focus on while designing the dashboard. Research is done by doing an observation of the process to identify bottlenecks, communicating with employees about problems and by process mining. This current situation will serve as the initial situation to measure the improved situation.

3) What are the needs of the company for important indicators to track?

- a. What are the norms for those needs?

Third, the preferences and goals of the employees play an important role while designing the dashboard. This will indicate what is seen as important according to the employees. Research is done by communicating with a few employees and with survey research, to include all opinions of all employees. The information is used to measure the difference between the initial and the improved situation.

4) What optimisations on Agile software development teams to improve productivity are suggested in research?

Fourth, a literature study is conducted to identify possible solutions suggested by research to improve the productivity of an Agile organised company. The productivity level is not optimal at this moment. By creating a clear visibility of the data to identify known and unknown problems, the expectation is to solve the core problem. However, it could be that only a data visualisation is not enough to solve the entire problem. Therefore this literature study is conducted. Based on the findings, a recommendation is written to be provided to the company.

5) What indicators are suggested for agile software development companies?

Fifth, a second literature study is important to identify important indicators suggested by research for the company. To control productivity, it is important to identify which indicators should be tracked to be able to do so. Already a lot of research has been conducted on this topic, therefore it is chosen to use the method of a literature study for this knowledge question.

6) What constitutes a dashboard to improve productivity of an Agile process and how to implement this?

Sixth, literature research is conducted to understand how to design and how to implement a dashboard in an Agile process. The company is pragmatic, so it is important to introduce the solutions pragmatically. Implementing a dashboard in an Agile process is already a widely investigated question, so therefore it is chosen to use literature. The results will be provided to the company.

In **Figure 6**, an overview of the research questions is provided. As described, both research question 1 and 2 contributes to the third phase of MPSM, analysing the problem. The goal of the other research questions is to formulate solutions, phase 4 of MPSM. After doing research, a solution is chosen which is phase 5 of the MPSM. In this phase, the design of the dashboard takes place. Implementing the dashboard is part of phase 6 and evaluation is phase 7 of the MPSM.

Research question	MPSM step	Section
<i>Research question 1</i>	<i>Phase 3</i>	<i>Section 2.1.</i>
<i>Research question 2</i>	<i>Phase 3</i>	<i>Section 2.2.</i>
<i>Research question 3</i>	<i>Phase 4</i>	<i>Section 2.3.</i>
<i>Research question 4</i>	<i>Phase 4</i>	<i>Section 3.1.</i>
<i>Research question 5</i>	<i>Phase 4</i>	<i>Section 3.2.</i>
<i>Research question 6</i>	<i>Phase 4</i>	<i>Section 3.3.</i>

1.4. Problem quantification

A problem is often described as a difference between norm and reality. The desired productivity level is lower than desired, resulting in different problems. Some of the problems, the company is aware of, others not. After a quick problem analysis, the underlying problem, the core problem, causing the lower productivity level became clear: there is a limited of data visualisation. Currently, the company is making decisions to react to those problems based on their gut feeling, instead of on data, visualising the problems.

To determine whether the solution solved the core problem or not, it is necessary to set a norm on what the company strives for. The company prefers to have the input for an overview visualising all data affecting productivity. By keeping track of important indicators problems can be detected in an early stage. **Figure 7** represents the current situation and the desired situation.

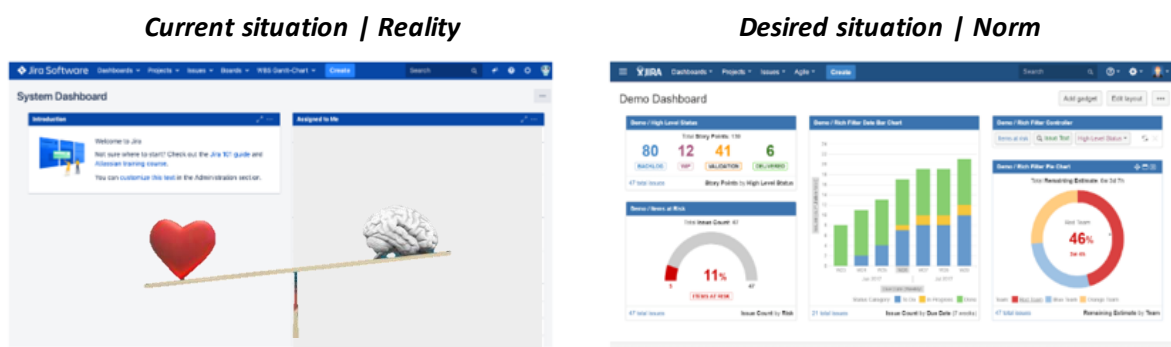


Figure 7: Reality vs Norm

After implementing the solution, a comparison can be made between norm and reality to see whether the core problem and thus the action problem, has been solved. Measuring the increase in productivity is beyond the scope of this research, since the period after the dashboard is out of the research period. To measure the comparison between norm and reality, two variables are assessed:

- 1) Available indicators
This variable expresses the number of indicators currently available to keep track of the problems. By having clear what this number is, the current insight into the problems can be measured. After implementing the solution, the available indicators are measured again to see whether the solution improved the situation.
- 2) Insight into the problems affecting productivity
The second variable is the level of insight into the problems affecting productivity. This variable assesses whether the awareness of the problems affecting productivity is improved. The current insight into the problems has been determined in section 2.2.1.3. To measure norm and reality, the improved insight into the problems affecting the productivity level is determined after implementing the solution.

2. Company analysis

In this chapter, research questions 1, 2 and 3 are answered in section 2.1, 2.2, 2.3, respectively. The answers to these questions are used to select and design the right solution, to create the most suitable dashboard for the CRM Tribe of A.S. Watson.

2.1. The current work process

To have a complete understanding of the work process, this section is divided into subsections to have a complete overview of the whole process. First, an explanation is provided of what CRM actually is and what their tasks and responsibilities are. Second, the principle of Agile is explained. This is important since Agile plays a major role in the way the process is organised. A.S. Watson created their Agile method, which is explained in subsection 3. In this section are all details of the process described.

Research question 1: how is the work process of the CRM Tribe organised?

2.1.1. CRM in Group IT of A.S. Watson

CRM is a combination of people, processes and technology that seeks to understand a company's customers. It is an integrated approach to managing relationships by focusing on customer retention and relationship development (Chen & Popovich, 2003). Customer Relationship Management has a key role in the software industry. While the software world is changing continuously, CRM applications take full advantage of this. Their ability of collecting and analysing data is valuable to respond timely and efficiently to everything related to online activities. Some examples of functionalities are being able to manage marketing campaigns, manage sales and many more.

Within A.S. Watson, the CRM Tribe is responsible to manage the CRM software for their business units. The tools used are Siebel, Adobe and OSB Middleware. The Oracle-powered Siebel CRM is a package of CRM solutions that can easily be modified to the business requirements. It supports major aspects such as marketing, sales, services etc. Adobe Campaign is a powerful campaign marketing solution with a wide variety of options. Personalised deals can be offered to the customers by using this tool. OSB Middleware is software functioning as a transition layer to enable different applications to work together.

There are several processes to manage everything around the software tools the CRM Tribe is dealing with. This varies from the customer, the store and the online environment to the offers and the campaigns. To manage all these different aspects, seven processes are used and shortly explained:

- 1) Test management: this is simply the procedure around the testing process of the product.
- 2) User management: this process is about dealing with important aspects for the user. Examples are GDPR, privacy issues and access to several systems.
- 3) Change management: this process involves managing changes to the systems, such as improvements, new requests, or changes to codes.
- 4) Release management: the process of deploying changes in the software.
- 5) Incident management: incident management plays a role when there is a problem with a direct impact on the functional processes.
- 6) Problem management: when a problem is arising with an indirect impact on the processes, problem management plays a role. This has less priority than incident management.
- 7) Escalation management: when there is an escalation (often a problem with a high financial impact), escalation management is needed.

2.1.2. The Agile principles

Agile is an iterative approach, a methodology, to project management and software development that helps teams deliver value to their customers faster (Atlassian, 2021). Different from regular project management tools are the continuous, small launches instead of one big bang launch. Open communication, collaboration, adaptation, and trust amongst team members are at the heart of Agile. Although the project lead or product owner typically prioritises the work to be delivered, the team takes the lead on deciding how the work will get done, self-organizing around granular tasks and assignments (Atlassian, 2021). The origin of Agile goes back to 2001 when 17 developers met and wrote the Agile Manifesto. Agile is based on twelve principles, included in Appendix A.

The main reason for companies to choose Agile is the ability to respond quickly to changes in the marketplace or feedback from customers, without derailing a year's worth of plans. "Just enough" planning and shipping in small, frequent increments lets your team gather feedback on each change and integrate it into future plans at minimal cost (Atlassian, 2021). According to the Agile Manifesto of Beck et al. (2001), the main focus is on people. Authentic human interactions are more important than rigid processes.

Since Agile is a methodology, many Agile frameworks have emerged over the last couple of years. The most famous framework is Scrum, but Kanban, Lean, and Extreme Programming (XP) are also frequently used frameworks. Each framework embodies the core principles of Agile, such as continuous improvements and frequent iterations. Many Agile teams combine concepts of the different frameworks today to create their own unique framework. This is also how A.S. Watson is organised. The Agile method of the CRM Tribe of A.S. Watson is based on principles from Scrum, the Spotify model, and the Nexus model. First, a clarification of those different methods is given to understand the unique method of the company. Second, the Agile framework within A.S. Watson is discussed.

Scrum

The first framework is Scrum, which is a framework that helps teams to work together. The framework describes a set of roles, meetings, and tools to help structure and organise the team. In this subsection, the main principles are explained. Within Scrum, there are three artifacts. An artifact is an object made by humans, comparable with a tool to solve a problem. Those artifacts in Scrum are defined as a product backlog, a sprint backlog, and an increment which contains the definition of done according to the team.

The product backlog

The following description is the definition of the product backlog according to Drumond (2018). A product backlog is the master list of work that needs to get done maintained by the product owner or product manager. This is a dynamic list of features, requirements, enhancements, and fixes that acts as the input for the sprint backlog. It is, essentially, the team's "To Do" list. The product backlog is constantly revisited, re-prioritised and maintained by the Product Owner because, as we learn more or as the market changes, items may no longer be relevant, or problems may get solved in other ways.

The sprint backlog

According to (Drumond, 2018), the sprint backlog is the list of items, user stories, or bug fixes, selected by the development team for implementation in the current sprint cycle. Before each sprint, in the sprint planning meeting, the team chooses which items it will work on for the sprint from the product backlog. A sprint backlog may be flexible and can evolve during a sprint.

Increment

The increment is the usable end-product from a sprint, often referred to as the team's definition of "Done" (Drumond, 2018).

There are three important roles: product owner, Scrum Master, and the development team. Here, the product owner is the key person between the customer and the Scrum team. It is of high importance that they understand the business, the customer, and the requirements so they are able to prioritise the work that needs to be done. The Scrum master is the person who deeply understands the work and coaches the team through the Scrum process. The development team consist of people who getting the work done.

The Scrum process consists of a set of events and meetings that are performed regularly. The key ceremonies are: 1) organise the backlog, 2) sprint planning, 3) sprint, 4) daily Scrum, 5) sprint review and 6) sprint retrospective. Organising the backlog is the responsibility of the product owner. The sprint planning is a meeting in which the work to be performed is planned by the entire development team. The sprint itself is the actual time period when the Scrum team is working on an increment, mostly two weeks. The daily Scrum is a daily meeting, super-short, to make sure the whole team is on the same page. The sprint review is the demo of an increment. Last, the sprint retrospective is an evaluation to look back at the process and evaluate what went well during a sprint and what did not. A clear overview of the work structure is visualised in **Figure 8**.

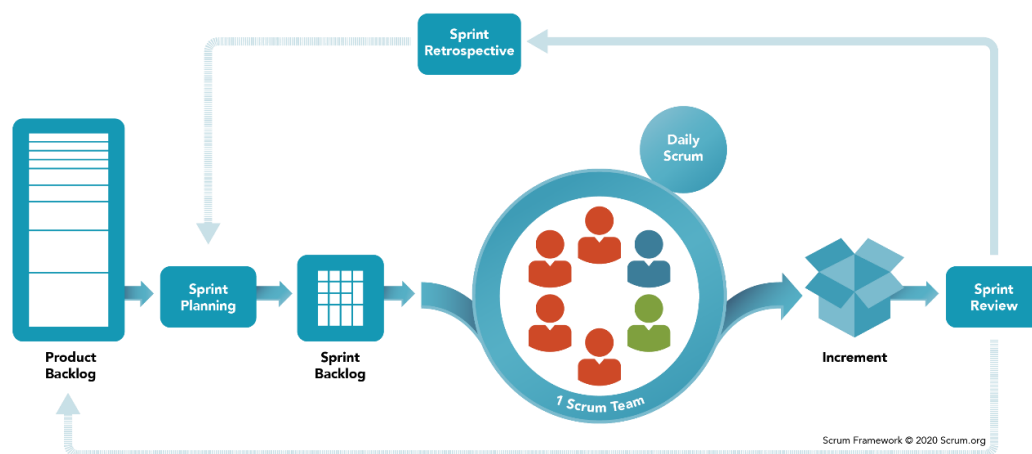


Figure 8: Scrum framework

The Spotify model

The Spotify model is a people-driven, autonomous approach for scaling Agile that emphasises the importance of culture and network (Mark Cruth, 2021). The model is derived from the Agile mindset and was introduced to the world for the first time in 2012. The Spotify model focuses on how business can structure an organisation in order to enable agility. Key focus areas are autonomy, communication, accountability, and quality. It is important to emphasise the fact that the Spotify model is not a framework, but it represents Spotify’s view on Agile. There are some key elements that are important elements for the Spotify model. A department within a company is called a “Tribe” and specific disciplines within the teams are described as “Squads”. There is a product owner and an Agile coach within the squad. Within the Spotify model, there are chapters, which is the family that each specialist has, helping to keep engineering standards in place across a discipline usually led by a senior technology lead (Mark Cruth, 2021). The whole Tribe is led by a Tribe lead. Other groups are a guild, a trio, and an alliance. First mentioned is a group of team members, also called a community with people with the same interest. A trio is formed by the tribe lead, a product lead, and a design lead. This group is formed to ensure there is continuous alignment between the different perspectives. Last mentioned is a group formed by combinations of tribe trios that work together to accomplish a goal involving multiple Tribes. The different terms are visualised in **Figure 9**.

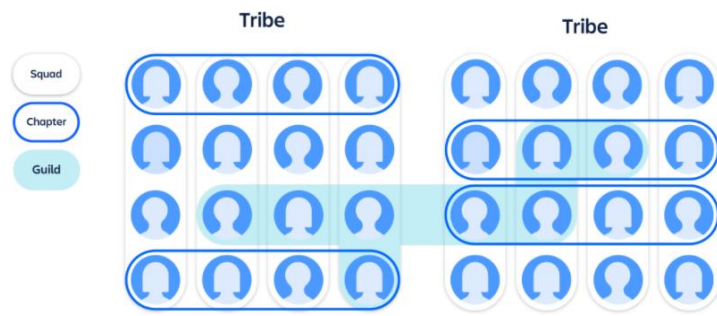


Figure 9: Scaling Agile at Spotify (Cruth, 2021)

The Nexus model

The Nexus framework builds on the Scrum foundation. A Nexus is a group of approximately three to nine Scrum (development) teams that work together to deliver a single product; it is a connection between people and things (Schwaber & Scrum.org, 2021). The Nexus guide describes that Nexus seeks to preserve and enhance Scrum’s foundational bottom-up intelligence and empiricism while enabling a group of Scrum Teams to deliver more value than can be achieved by a single team. Within Scrum, a development team is using one product backlog. The difference with the Nexus framework is the use of multiple so-called Nexus sprint backlogs. Each team collect all the details of tickets on their own backlog. The cross-team refinement sessions are there to prevent delay and to identify dependencies among different development teams. The Nexus framework is visualised in **Figure 10**.

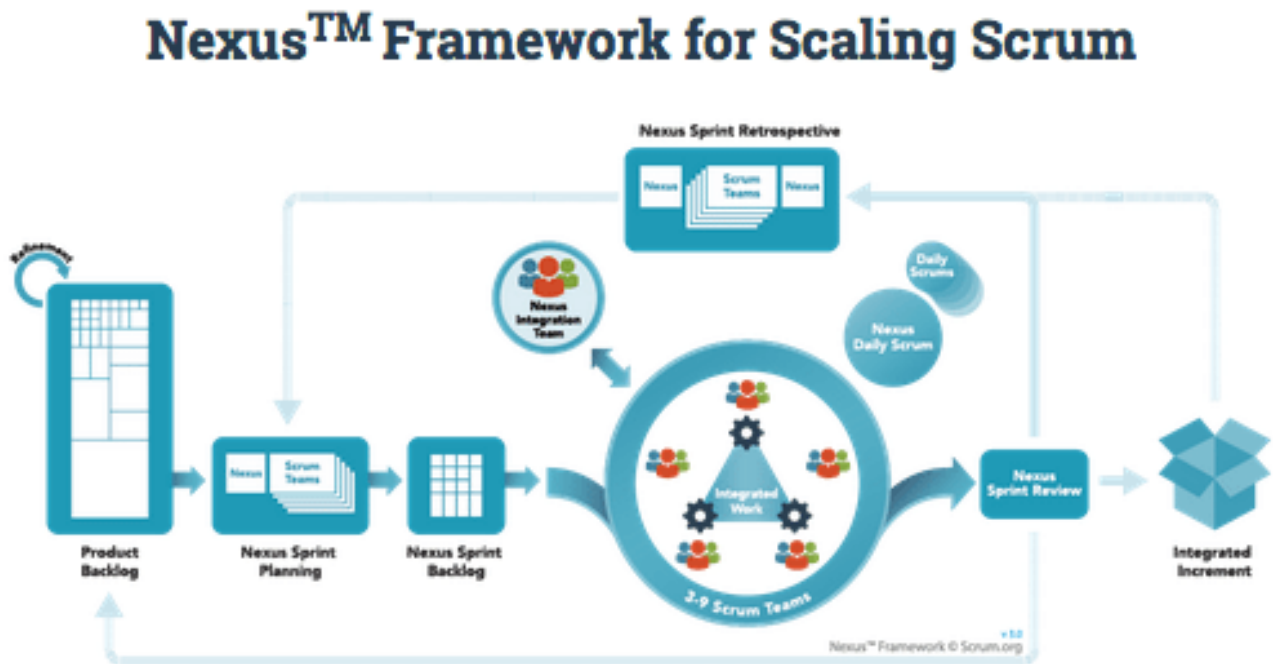


Figure 10: Nexus framework (Scrum.org, 2021)

2.1.3. The A.S. Watson model

Within the CRM Tribe, a self-developed working method has been implemented several years ago based on principles of the three methods described above. The main idea was to implement an Agile method based on several aspects from different frameworks. **Figure 11** visualises the roots of Agile. As can be seen, Agile is a composition of many concepts with different principles. Where for example Sociotechnics focuses on people, theory on constraints (TOC) focuses on processes. Combining principles of all these concepts leads to the model of the CRM Tribe of A.S. Watson.

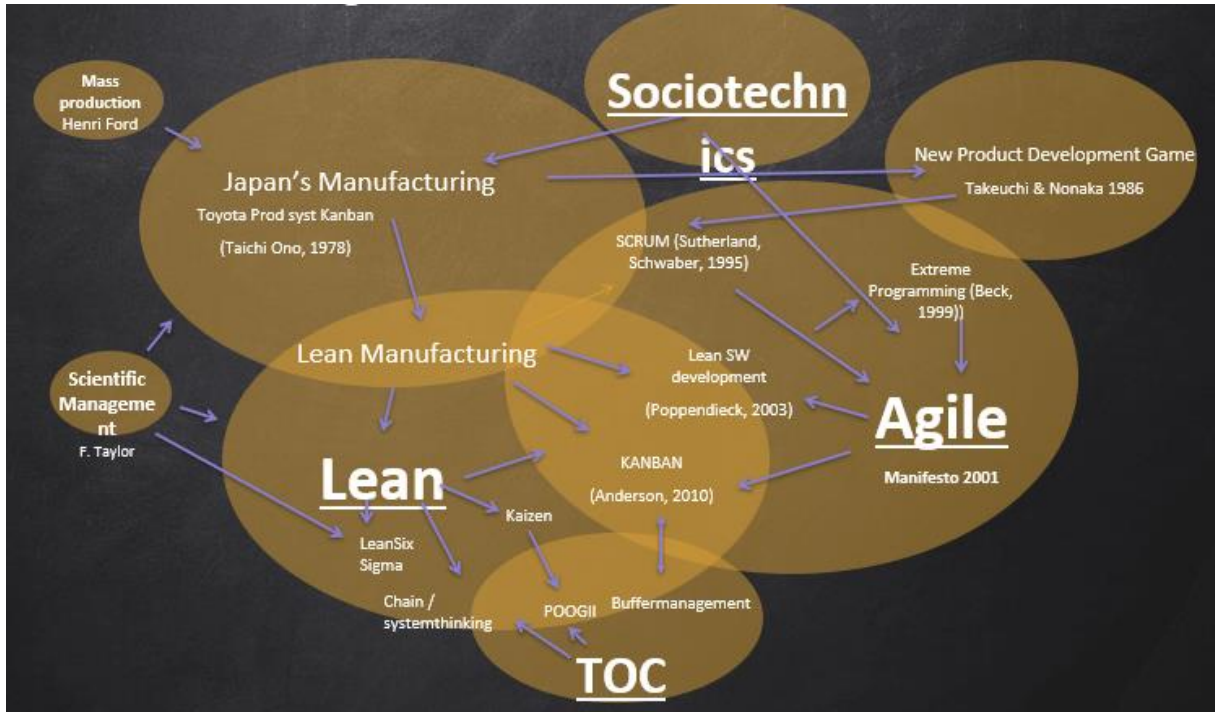


Figure 11: Roots of Agile

The structure of the department is organised based on the Spotify model, visualised in **Figure 12**. The CRM department is called the CRM Tribe, the disciplines within the team are called a Squad. Besides, there are also guilds and chapters. When there is a big request concerning multiple squads, guilds are formed. Testers of the different squads are in a chapter. Scrum describes three important roles: product owner, scrum master and the development team. The method also contains the role of a Tribe lead, as prescribed by the Spotify model. The main roles are depicted in **Figure 13**.

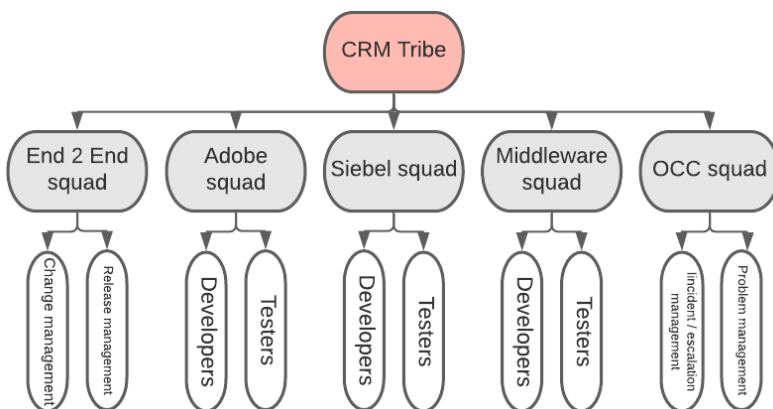


Figure 12: Organisation of CRM Tribe



Figure 13: Main roles

The organisation of the several processes is based on the Scrum framework with the following key ceremonies: organise the backlog, sprint planning, sprint, sprint review and sprint retrospective. In these ceremonies, the artefacts as described by Scrum are also used. To understand the Scrum process, a visualisation is given in **Figure 14**.

The start of the process is at the left side, where inputs are coming into the process via the product owner. The product owner prioritises the requests and is setting up a list of requirements. At A.S. Watson, the information analyst is setting up a first concept of how the solution should look like. The tasks enter the *product backlog*, also called the “to do list” of the concerning squad. The next step is for the teams. Instead of using one single team as described according to Scrum, A.S. Watson is using three teams, as prescribed by the Nexus model. Each team has their own backlog. The teams select requests for the *sprint backlog* to deliver by the end of the Sprint during the sprint planning meeting. Then a period of two weeks follows in which the team is working on the requests standing on the sprint backlog for the specific period. Every 24 hours during the sprint, a daily stand-up meeting takes place to check whether everything is going fine. After the sprint is finished, a sprint review is held and there will be determined whether the product is finished or not. This is also called the *increment*, the definition of done. At the end, the sprint retrospective takes place. The overview of the work to be done can be found on the Kanban board. All tickets are placed on the Kanban board that provides a visual overview of the status of the tickets.

The project management tool the CRM Tribe is using is Jira, in combination with Confluence. Jira is a software application used for the software developing cycle to manage all the work. Confluence is a collaboration tool supporting the processes in Jira for collaborating and sharing knowledge efficiently.

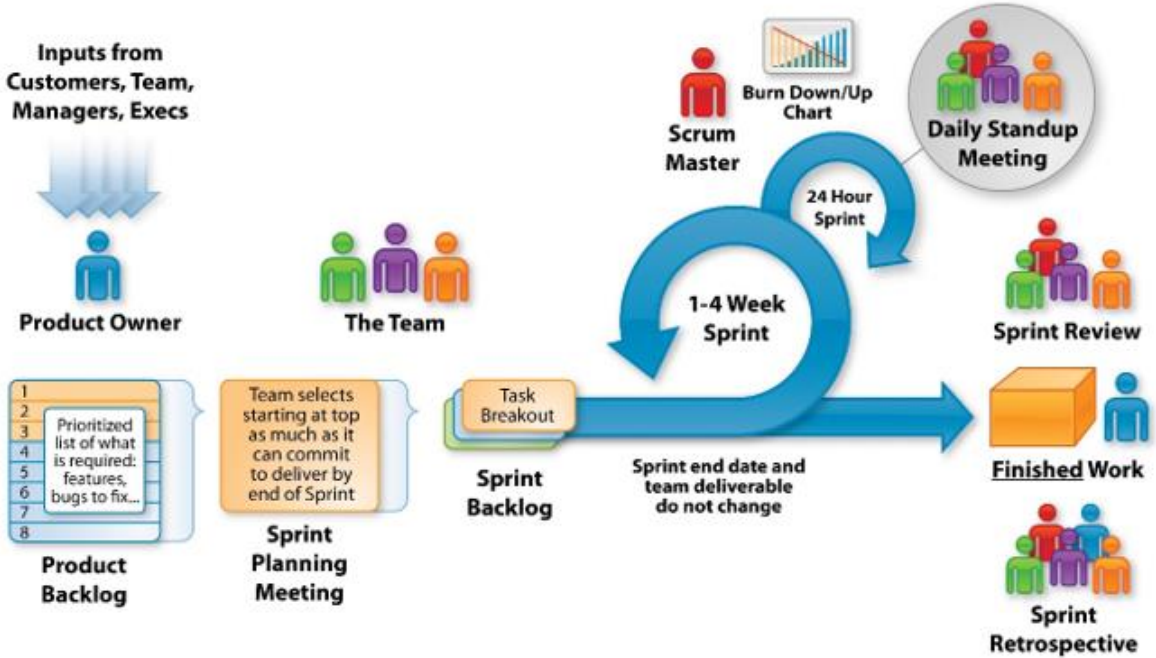


Figure 14: Scrum process

Now all elements of the different frameworks used in the A.S. Watson model are clear. The structure and the roles of the CRM Tribe are described, and the standard process is explained. To be able to understand the problems on which this research focuses, it is of importance to understand the approach of the seven processes described in 2.1.1. within this Agile method.

The inputs/requests are coming into the process in two ways, explained in **Figure 15**.

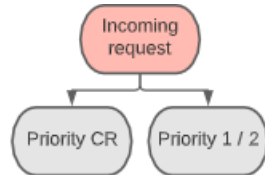


Figure 15: Incoming request

Figure 16 visualises the structure of the processes the CRM Tribe uses. First, entering the process via the product owner is discussed, which is change management (priority CR). An epic planning is made for the coming three months over a set of sprints. An epic serves to manage tasks. It is a defined body of work that is segmented into specific tasks based on the needs/requests of customers or end-users (Rehkopf, n.d.). The idea is to break down large tasks into doable pieces so that large projects can get done. In this epic planning, there will be determined in which release period a request will be addressed.

Every six weeks, a release is planned. In the change planning is determined which request is developed in which sprint. One release period consists of three sprints, each two weeks. In the two weeks, development takes place. After development, the processes release management and test management play a role. First regression tests and integration tests are conducted to identify whether the change integrates within the systems. The next step is the User Acceptance Test (UAT). This is the last phase of the testing procedure. During the UAT, software users of the Business Unit test the software to make sure it meets the requirements. When the testing procedure is completed, the software is ready for production to be deployed to the end-user.

The last phase visualised at the right side is operations (OPS). Within OPS user management, problem management, incident management and escalation management occur. This is the second way a request can come into the process. When there is an error in one of the systems for example, a topdesk ticket can be made which will be addressed by the operations centre control (OCC) squad. If the problem is of priority 1 or 2, the problem will be solved immediately. Otherwise it will start as a new change request.

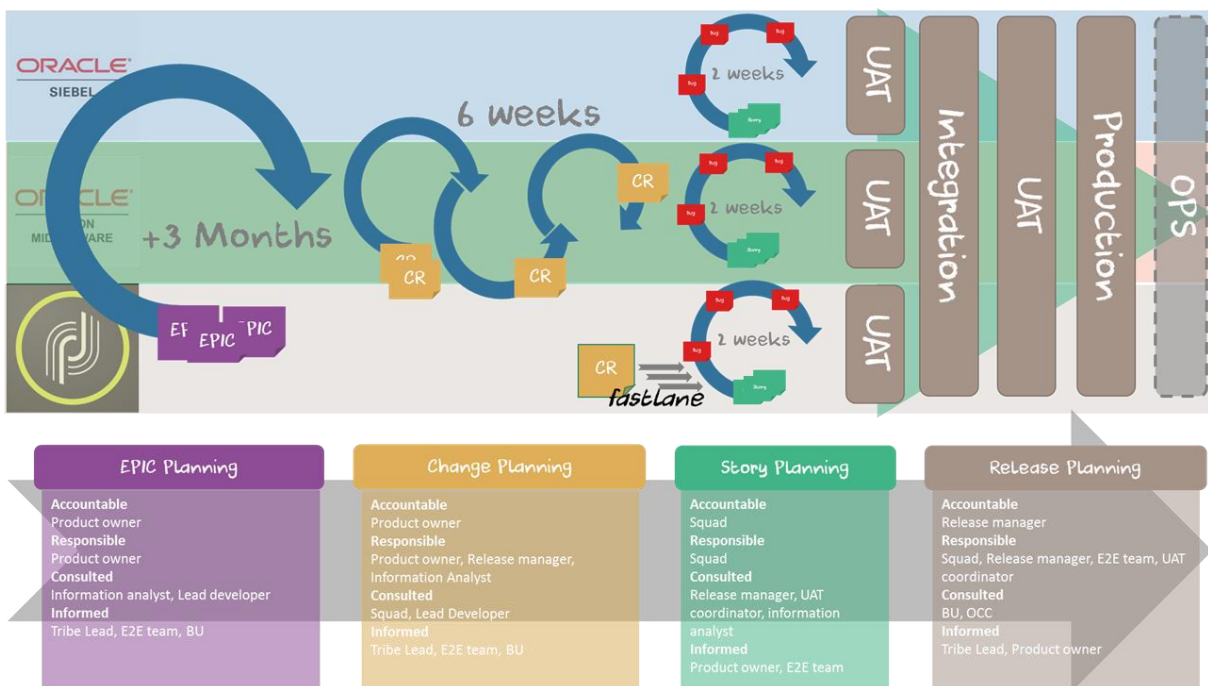


Figure 16: Process structure

For a better understanding of the process in which the request is coming into the change management process flow, the workflow is discussed. When there is a new change request coming in at the product owner, the ticket is going through five phases. The first phase is specification, which involves planning and a quick setup of the solution. The second phase is refinement, in which the request is refined. During the sprint, the actual development phase takes place. After development, the request is ready to get tested. When the BU approves after their UAT, the request is ready for release. The process of the five phases is visualised in **Figure 17**.

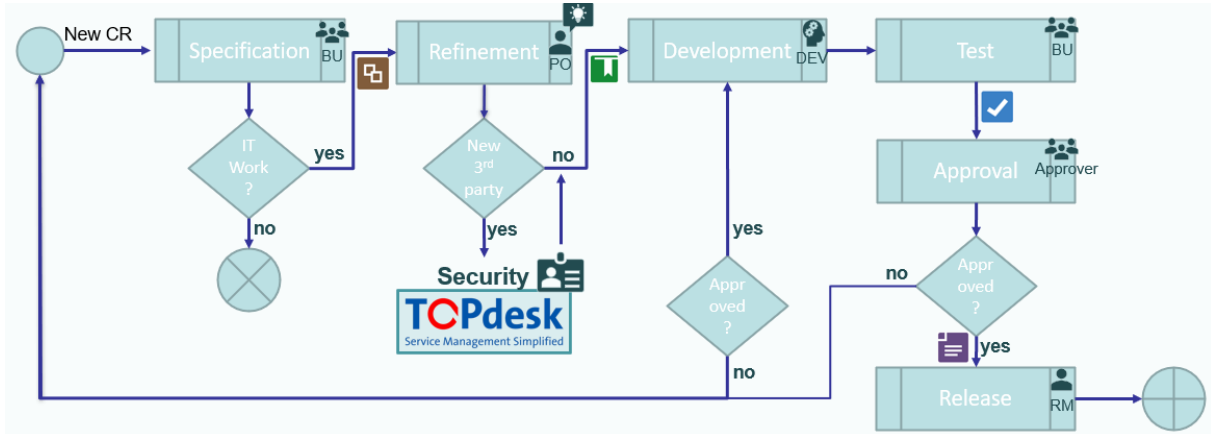


Figure 17: Change management process flow

When the requests are coming into the process by a topdesk ticket, the level of priority will be determined first. If the request concerns a priority level 1 or 2, it will be solved immediately. On the other hand, a ticket can be prioritised as a new change request. If this is the case, the request will be solved based on the change management procedure managed in Jira. **Figure 18** shows a visualisation of this process.

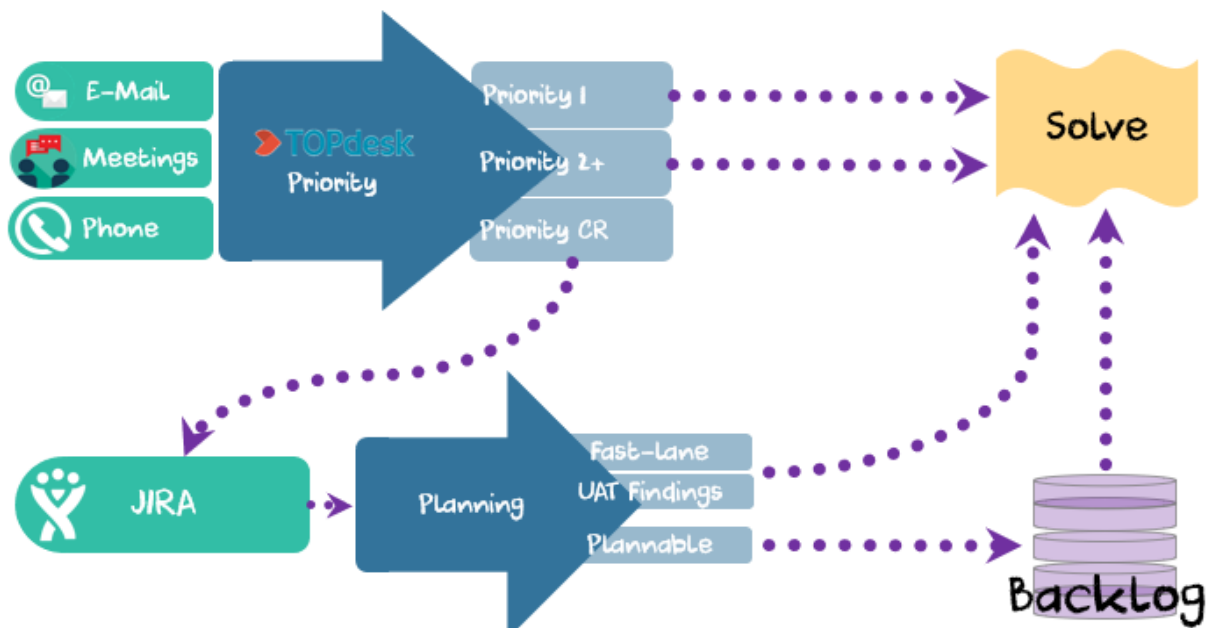


Figure 18: Incoming request flow

2.2. Problem analysis

The second section of chapter 2 is providing an extensive analysis of the, lower than desired, productivity level of the GIC project. An explanation is provided in Definitions . It is clear that there are problems affecting productivity, however, it is unknown what exactly all the problems are since the limited level of data visualisation. To determine which data to visualise on the dashboard, an insight into the problems is necessary. First, to identify the bottlenecks, the process is researched by analysing available reports in Jira and the technique process mining. Important to notice is that these Jira reports are not used at the moment by the company to analyse the process. Next, the influence on the productivity level is analysed and last the productivity level is explained by the actors involved.

Research question 2: where in the process are the bottlenecks?

2.2.1. The bottlenecks

Jira contains a lot of data and information that can help to provide insights into the productivity of a team. Besides, the data can help to identify and resolve bottlenecks to accelerate and improve performance. However, the biggest part of this data is either hidden or hard to retrieve. There are currently six useful reports available within Jira to review the process. Reports help teams to analyse the progress made on a project. This is the first part of the analysis, to have a quick indication of potential bottlenecks, provided in 2.2.1.1. A zoomed-in version of the reports is provided in appendix B. Second, the process is analysed in more detail by using the tool process mining, provided in 2.2.1.2. A conclusion of the problems and the matching indicators is provided in 2.2.1.3.

2.2.1.1. Jira reports

Cumulative flow diagram

The first report is the Cumulative Flow Diagram (CFD), visualised in **Figure 19**. The diagram is an area chart that shows the various statuses of work items for a sprint. The aim of the CFD is to show the stability of a process over time. The horizontal x-axis indicates time, the vertical y-axis indicates issues (tickets). Each coloured area equates to a workflow status (Atlassian, 2020).

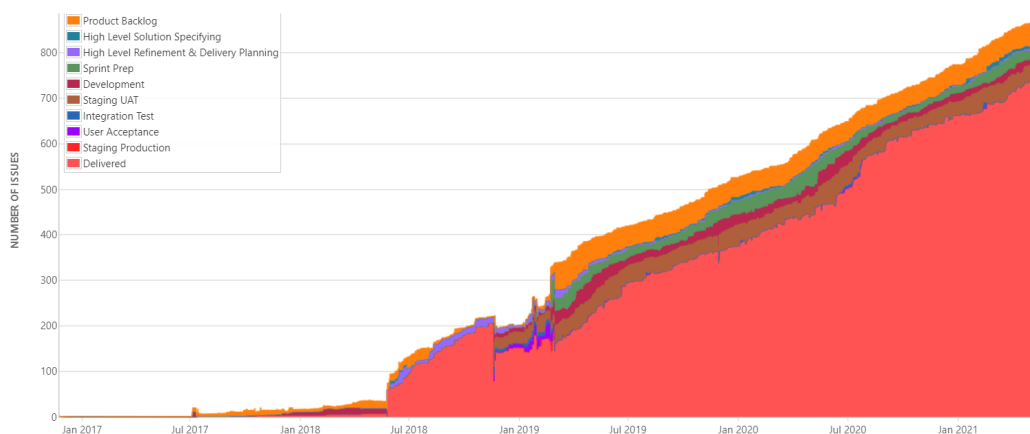


Figure 19: Cumulative flow diagram (CFD)

The CFD is useful for addressing potential bottlenecks. If the chart contains an area widening significantly more vertically than during other periods, the widening column is generally a bottleneck. When zooming in, there are some periods with a widening integration test column.

Control chart

The control chart in **Figure 20** shows the Cycle Time or Lead time for a sprint. It takes the time spent by each issue in a particular status and maps it over a specified period of time (Atlassian, 2021b). The rolling average is shown by the blue line and is issue-based. It is calculated by taking the issue, X issues before and after the issue and averaging their cycle times. The advantage of this method is that it produces a steady average line showing the outliers. The blue shaded area represents the amount of variation from the rolling average, the standard deviation. Each open dot represents a single issue, a solid dot represents a group of issues. The vertical y-axis indicates the elapsed time, which is the cycle time of a single issue and the average cycle time for a group of issues. The horizontal x-axis represents when the issue passed the last status selected.

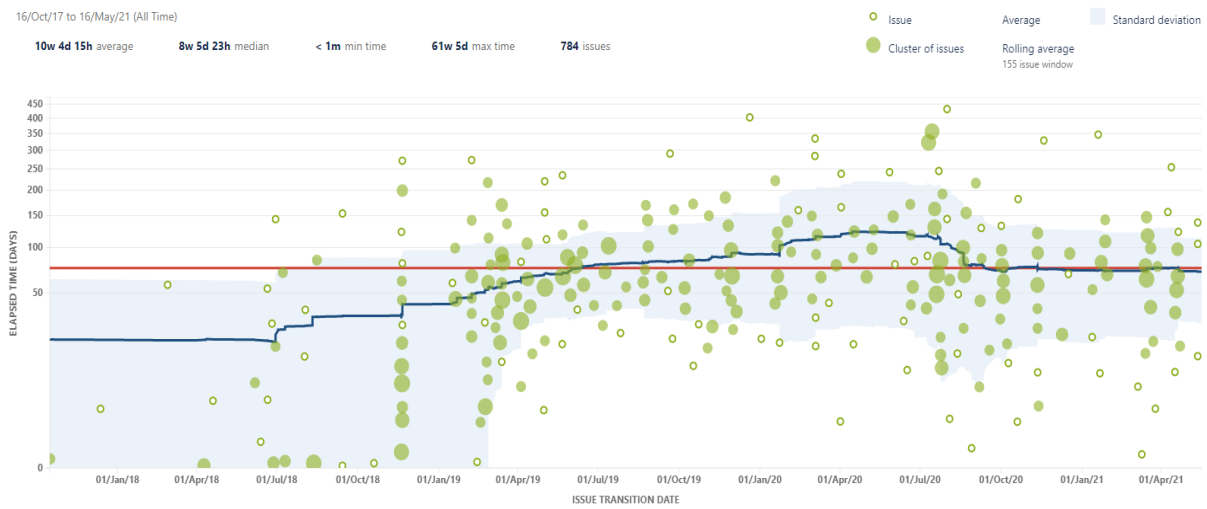


Figure 20: Control chart

When the blue line of the rolling average is below the average, it indicates the team is working efficiently. Higher values may indicate bottlenecks of the process. Besides, the control charts are showing a lot of outliers. Especially the ones above the average could be an indication of a bottleneck somewhere in the process.

Average age report

The average age report in **Figure 21** shows the average age of unresolved issues for a period of two years. The purpose of this report is to identify whether the backlog is kept up to date. The vertical y-axis represents the number of days an issue is unresolved. The horizontal x-axis the period of time. From the period of May 2019, the average age of unresolved issues is rapidly increasing. This report could be an indication of a bottleneck around the process of the backlog.

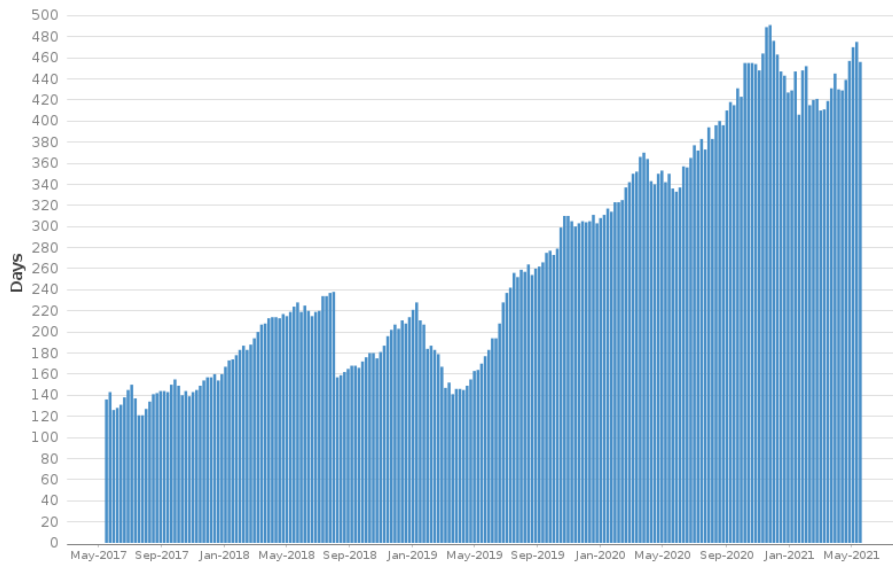


Figure 21: Average age report

Resolution time

Figure 22 shows the resolution time report. This report represents the length of time needed to resolve an issue. It starts counting from the moment the customer, the BU, reaches out to the CRM Tribe and stops when the BU receives an answer they consider as complete. This is important for the satisfaction level of the customer.

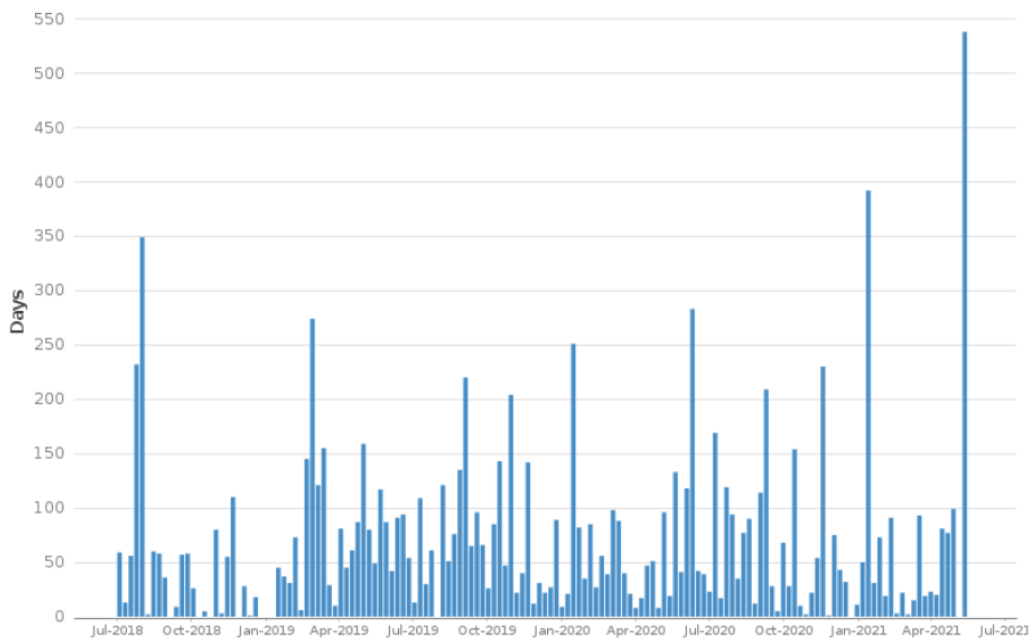


Figure 22: Resolution time report

It stands out that there are several peaks, indicating a high average resolution time. Since the resolution time is indicating the time from starting the request till the end, this means the issue is spending a long time in one or more statuses. This is potentially a bottleneck for a suboptimal productivity level.

2.2.1.2. Process mining

Process mining provides a set of techniques to automatically extract process behaviour from event logs (Marques et al., 2018). The approach has already been applied successfully in certain fields. Jira records a large amount of run-time event data which can be used for process mining to discover the patterns and analyse the process. Every single event contains at least information about the case, the activity (status), the resource and time.

When mining a process, it is important to apply a process mining methodology. The one recommended by Marques et al. (2018) is the PM² process mining methodology since it is a general approach supporting the analysis of both structured and unstructured processes. According to the methodology, there are six stages: (1) planning, (2) extraction, (3) data processing, (4) mining & analysis, (5) evaluation and (6) process improvement & support (Van Eck et al., n.d.)

(1) planning

The goal of applying process mining to this research is to determine the efficiency of the process to identify bottlenecks. The research question corresponding to this is “where in the process are the bottlenecks?” For this purpose, data is exported from the Jira Cloud in which all the events are logged.

(2) extraction

As mentioned, it is hard to easily retrieve data from Jira. The necessary aspects are a key, activity, resource, and time. The challenging part here is time. To eventually analyse the process, it is important to have an insight into the amount of time it took for each ticket to go through each status. Directly exporting the history of an issue is impossible from Jira to Excel, so first a script has been created to extract the desired data. After exporting the data from the GIC project, it became clear that the script worked out, however not in an efficient way. To show the challenge of retrieving data, the script, written by M. Noorloos, Scrum Master of the Adobe squad within the CRM Tribe, is included in **Figure 48** in the appendix B. The solution to export the data is eventually found by using the add-on advanced export, by Atlassian, who also developed Jira. **Figure 23** shows the retrieved format of the data.

The screenshot shows the 'Advanced Export' interface in Jira. At the top, the 'Export issues' filter is set to 'project = GIC and issuetype = Change and status = Delivered', with 793 issues selected. Under 'Fields to export', 'Status' is selected. The 'History export' option is checked, with a note that it adds columns for 'As Of Date', 'Changed Fields', and 'Change Author'. The 'Date/Time Format' is set to '18/05/2021 14:42:47' and the 'CSV Separator' is 'Comma'. Below the settings is a preview table for 5 out of 793 issues.

Key	As Of Date	Changed Fields	Change Author	Created	Status
GIC-3820	18/05/2021 14:42:47			06/04/2021 10:36:29	Delivered
GIC-3820	17/05/2021 15:24:30	Status		06/04/2021 10:36:29	Delivered
GIC-3820	19/04/2021 13:47:03	Status		06/04/2021 10:36:29	Staging UAT

Figure 23: Input in Advanced Report

(3) data processing

To prepare the data for the analysis in Disco, some data is removed. All issues are completed issues, open issues were already filtered out. Cancelled issues are removed because in this research it is unnecessary to improve the process of cancelled issues.

Figure 24 shows the process flow of a change issue within Jira. As can be seen, the ticket starts with the status “inno backlog”. When following the flow, the tickets can go through “specification” and “refinement” before coming into the “sprint preparation” status. After sprint preparation, the ticket can flow through “in development”, “staging UAT”, “integration test”, “user acceptance test”, “staging production” and finally the ticket will be “delivered”. Several different statuses were appearing just a few times. Those were removed or merged with identical statuses since the work statuses showed below are of interest.

It is interesting to analyse the frequency and performance of the workflow. When there is a high frequency in a specific status, it could indicate a bottleneck. Performance is showing the time elapsed between different states. It is for example interesting to analyse the time it takes between “inno backlog” and “delivered”, to identify the complete time it takes from starting the issue till ending the issue. What could also be interesting is the time between sprint preparation and in development, to identify how long it takes before the ticket is in development from the moment it is planned.

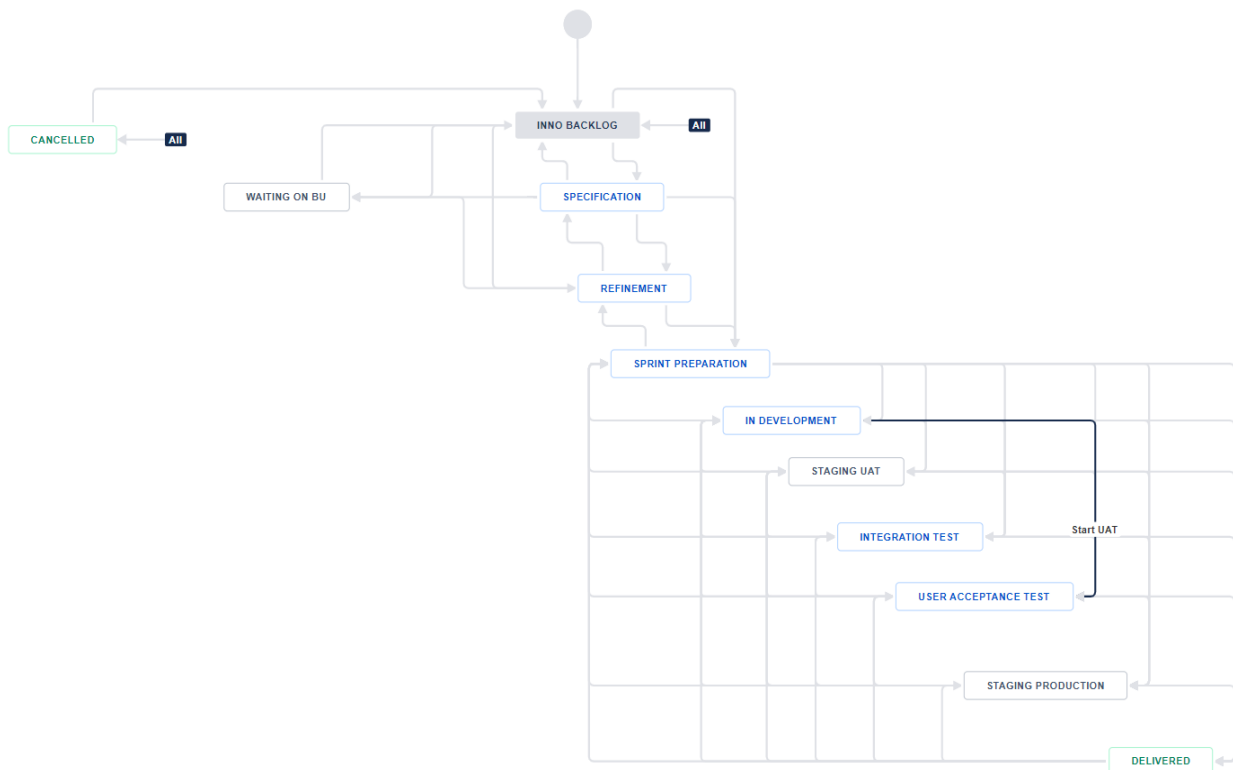


Figure 24: Process flow Jira

(4a) mining and analysis - flows

The analysis is divided into two parts. First, the project is filtered on all completed tickets available of which the flows are analysed. Second, statistics from the period of 2016-2021 are provided. For an accurate insight into the current process, statistics are also provided from 2019-2021.

The project comprises 765 cases and 7438 events performed between 13/10/2016 and 28/04/2021 after applying the filters mentioned above. One single case, also called the issue/ticket, consists of multiple events, which are the statuses. There are 513 variants, with as most common variant the path “Inno backlog → specify and refine → sprint preparation → in development → delivered”.

Figure 25 shows a quick visualisation of how the tickets flow through the different statuses. The path on the left side shows the flow of the tickets in January 2018. The path in the middle represents the process exactly one year later, and the path on the right side visualises the process another year later, in January 2020.

What immediately stands out is the thickness of the yellow dots, highlighted by the grey rectangles. The statuses involved are “specify and refine”, “sprint preparation”, and “in development”. When a ticket is in the first stage, the status “inno backlog”, it takes a long period before entering the development phase. This can be seen when playing the animation, since the yellow dots are moving slowly through the phases “inno backlog”, “specify and refine”, “sprint preparation”, and “in development”. The first phases, before a ticket starts in the development phase, clearly indicate a bottleneck. A lot of tickets are pushed into the process flow before they can be developed. When observing the process, this bottleneck could be declared by the fact that entering tickets are scheduled mostly one or even two release deadlines ahead. One release period consists of six weeks. Therefore, it takes a long period before the tickets can enter the “in development” stage.

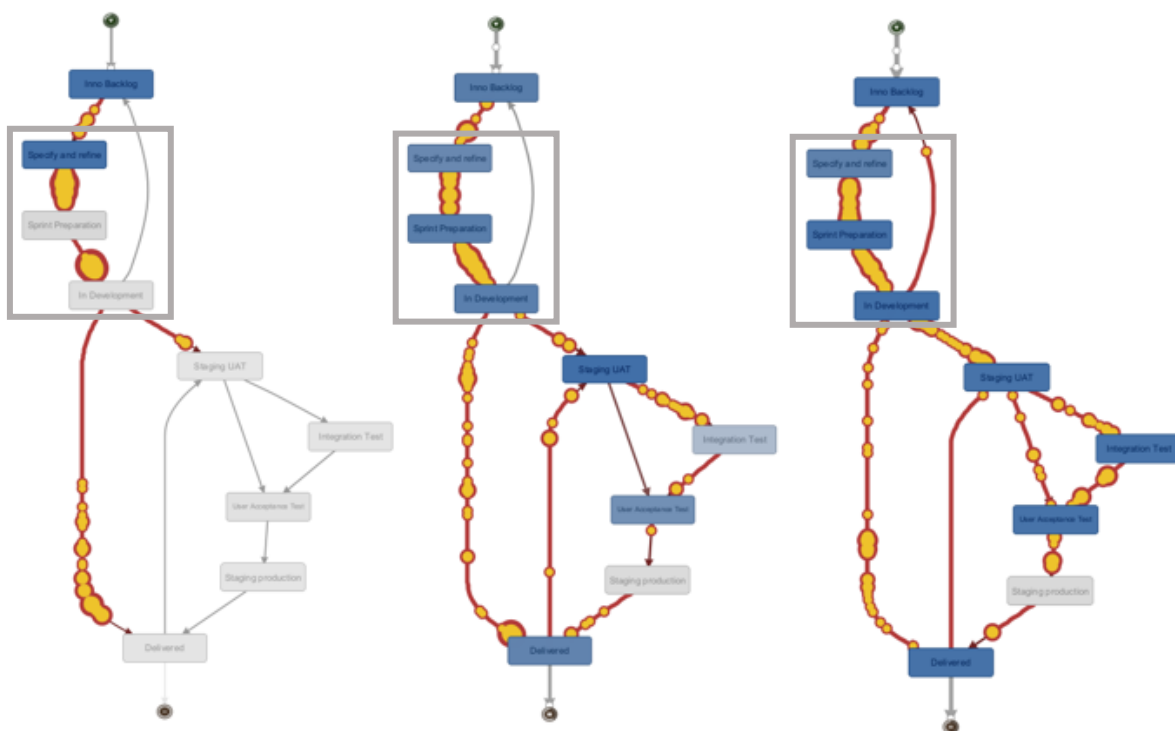


Figure 25: Animation of the flow generated by Disco (100% activities, 0% paths)

When analysing the process from a control-flow perspective, the patterns with the frequency per status of the tickets can be analysed. This can be seen on the left side in **Figure 26**. Remarkable is that a lot of tickets skip the testing phase, which is an important stage. What becomes clear from the analysis is that the data is not going smoothly through the process at all. The number of tickets in the development phase is 1056. However, 659 tickets are continuing the process to either “Staging UAT” or “delivered”. The path of the other tickets can be seen when increasing the number of paths to 100%. This is visualised in Appendix B. The tickets are going to “Integration Test”, again “in development”, “sprint preparation” and back in “inno backlog”.

When analysing the process from a performance perspective visualised on the right side, the mean duration times from stage to stage are given. What becomes clear from this analysis is the time it takes between entering the process and leaving the process. The biggest bottleneck is between the stages “sprint preparation” and “in development”, with a mean duration of 32.1 days. The tickets spend the longest time in the first period of becoming in the development phase. Another remarkable point is the time it takes when a ticket needs to re-enter the process. It takes 24.1 days to go to the “staging UAT” stage after the “delivered” stage, and then the testing phase did not even start.

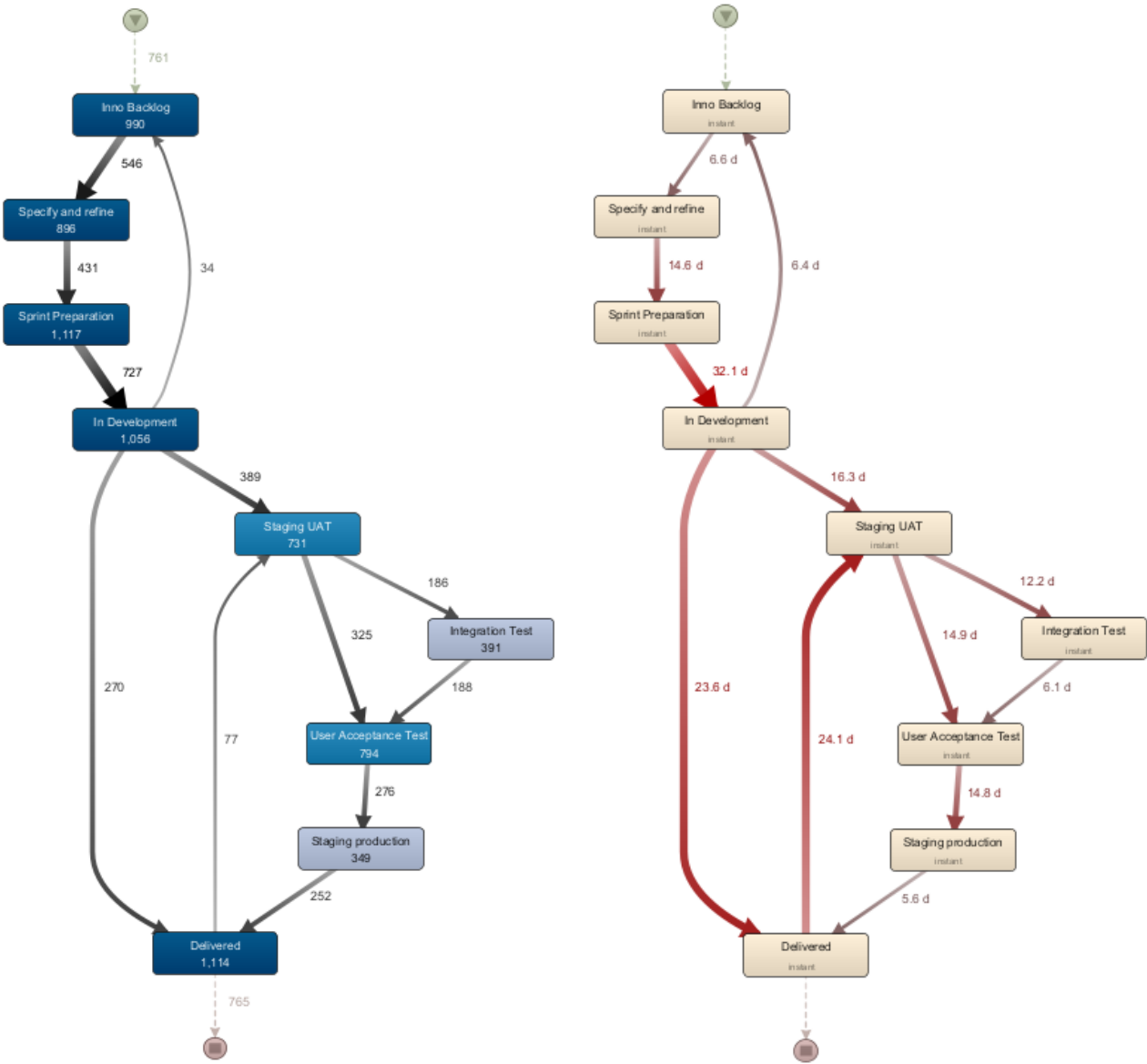


Figure 26: Control-flow perspective (left) and performance perspective (right) of the process model generated by Disco (100% activities, 0% path)

(4b) mining and analysis - statistics

The statistics give a good insight into some potential bottlenecks as well. The median case duration is 18.1 weeks, meaning that most tickets have a duration of 18.1 weeks, which is equal to 126.7 days. The mean case duration is 25.1 weeks, equal to 175.7 days. As visualised in **Figure 27**, many tickets are having a duration of over 180 days.

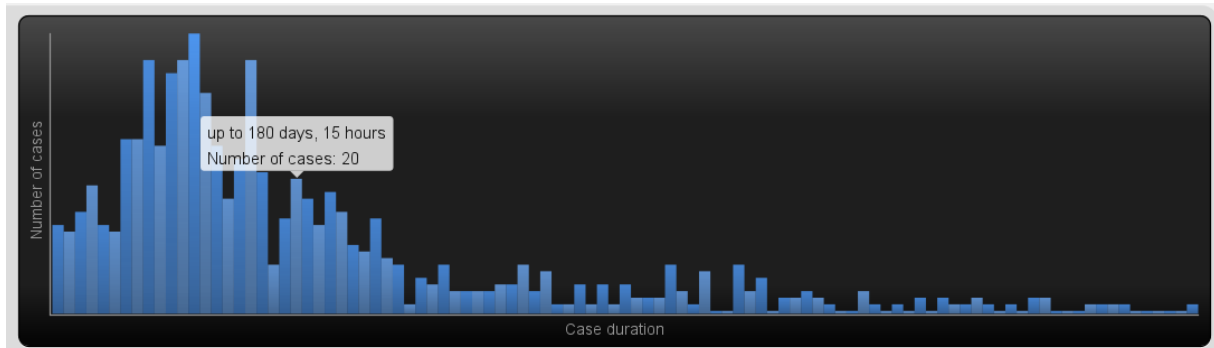


Figure 27: Case duration generated by Disco for tickets between 2016-2021

This could be declared when looking at the events per case. Most tickets are going through seven stages, also called events. As visualised in **Figure 28**, there are many tickets having 10 events or even more. When a ticket goes through all stages, it should have nine events (figure 26). There are even tickets having 20 events or more. This is an important bottleneck that should be optimised to improve the productivity level.

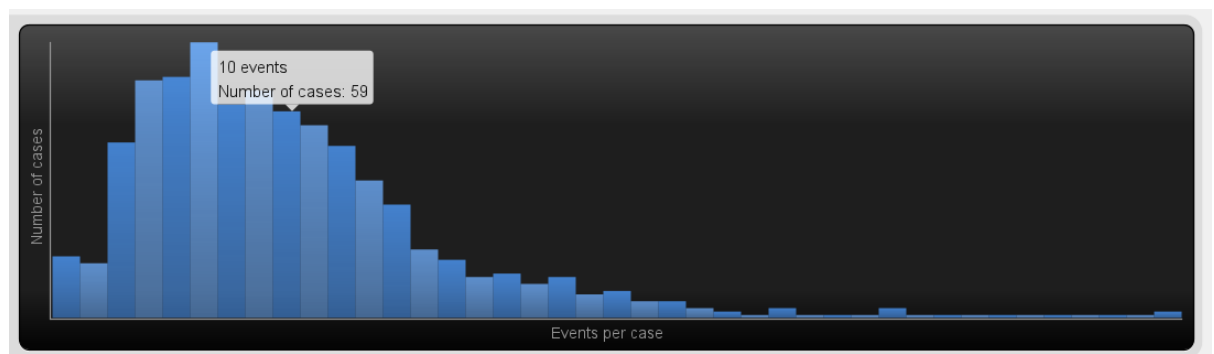


Figure 28: Events per case generated by Disco (10 events) for tickets between 2016-2021

Between 07/01/2019 and 28/04/2021, the project comprises 540 cases and 5499 events performed after applying the filters mentioned in the third step, data processing. There are 360 variants, with as most common variant the path “Inno backlog → specify and refine → sprint preparation → in development → delivered”. The median case duration is 14.3 weeks, equal to 100.1 days. The mean case duration is 17.7 weeks, equal to 123.9 days. As visualised in **Figure 29**, there are many tickets having a case duration over 123.9 days.

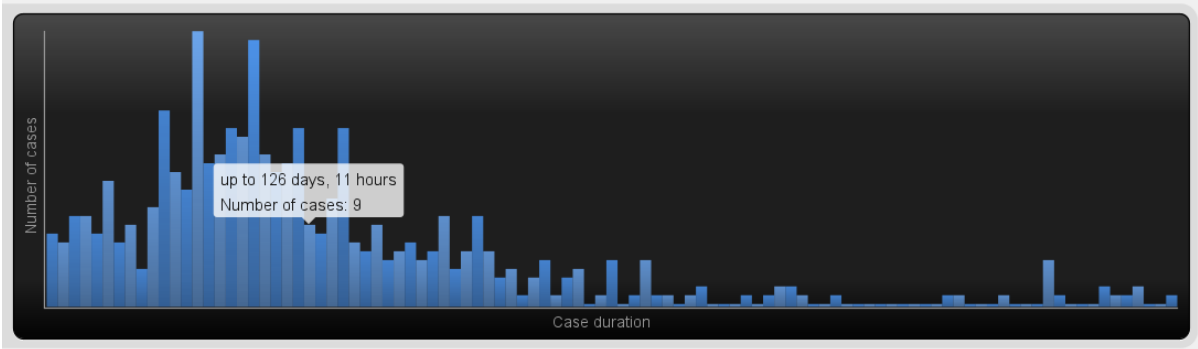


Figure 29: Case duration generated by Disco between 2019-2021

Also for the tickets between 2019-2021 holds that it could be declared when looking at the events per case. Many tickets are having more than nine events.

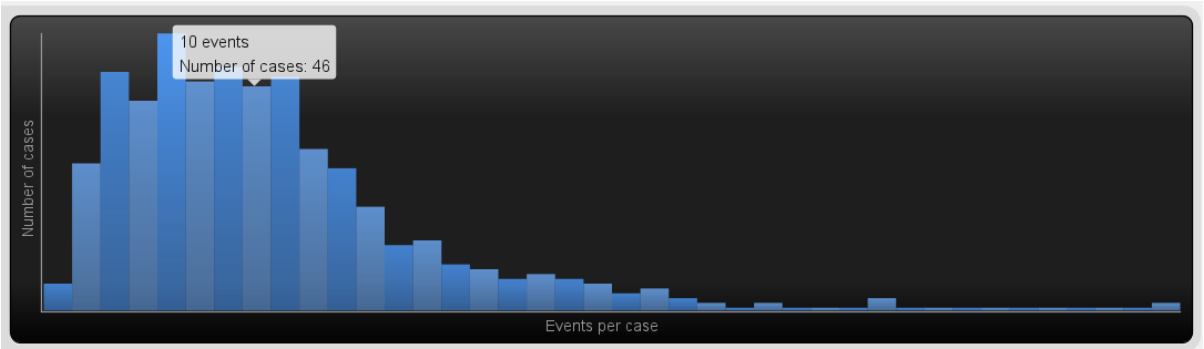


Figure 30: Events per case generated by Disco (10 events) for tickets between 2019-2021

(5) evaluation

When evaluating the findings of conducting the Disco analysis, several bottlenecks that could have a negative effect on the productivity level stand out.

First of all, there are clear bottlenecks visible in the process flow. Between the “inno backlog” stage and entering the “in development” stage, more tickets are pushed very slowly into the process than can be processed. It takes a long period before a ticket can be delivered since the tickets are waiting a long time in the first phases. From a performance perspective, the big red line between “sprint preparation” and “in development” indicates a bottleneck with a mean duration of 32.1 days. This is also shown by the statistics. Many tickets are having a longer duration than the mean duration. Another bottleneck is visualised between “delivered” and “staging UAT”. When a ticket needs to re-enter the testing phase, it already takes a lot of days before actually getting tested. Long waiting times do not contribute to an optimal productivity level.

What is also remarkable is that the data is not going smoothly through the process at all. Many tickets are re-entering similar stages or are starting from the beginning when they were already in the “delivered” stage. This is visualised in appendix B, where the flow is depicted with all possible paths. The statistics confirm this since many tickets having 10 events or even more. Re-entering similar stages take time and could therefore causing a decrease in the productivity level. Also remarkable is that a lot of tickets skip the testing phase. Since testing is of importance in the software developing industry, this is important to further investigate.

The most important conclusion of this analysis is the uncertainty about the correctness of the data. As mentioned during the process of the data extraction, it is really hard to even export the desired data from Jira to a CSV file. When observing the exported data, there were several tickets from which the data is uncertain. To explain this uncertainty, an example is given in **Figure 31**. The rows represent the history of each status the ticket went through. In a time period of 16 minutes, the ticket came across 5 statuses, which is highlighted by the red rectangle. It could be possible that the ticket followed this path in the system when for example several employees push the ticket forward to the new status accidentally. However, in practice, it is not likely to finish these five statuses within 16 minutes. There are more tickets in the dataset with the same uncertainty. Therefore, the findings above cannot be concluded with a high level of certainty.

1	Key	As Of Date	Changed Fields	Change Author	Created	Status
220	GIC-3575	19/04/2021 14:24:3	Status	ScriptRunner for Jira	14/01/2021 09:41:2	Delivered
221	GIC-3575	07/04/2021 13:40:5	Status	ScriptRunner for Jira	14/01/2021 09:41:2	User Acceptance Test
222	GIC-3575	25/03/2021 08:57:5	Status	ScriptRunner for Jira	14/01/2021 09:41:2	In Development
223	GIC-3575	23/03/2021 10:25:1	Status	ScriptRunner for Jira	14/01/2021 09:41:2	Sprint Preparation
224	GIC-3575	23/03/2021 10:24:2	Status	ScriptRunner for Jira	14/01/2021 09:41:2	User Acceptance Test
225	GIC-3575	23/03/2021 10:23:2	Status	ScriptRunner for Jira	14/01/2021 09:41:2	In Development
226	GIC-3575	23/03/2021 10:10:2	Status	ScriptRunner for Jira	14/01/2021 09:41:2	Sprint Preparation
227	GIC-3575	23/03/2021 10:09:3	Status	ScriptRunner for Jira	14/01/2021 09:41:2	In Development
228	GIC-3575	22/03/2021 13:12:3	Status	ScriptRunner for Jira	14/01/2021 09:41:2	Sprint Preparation
229	GIC-3575	03/03/2021 06:03:1	Status	ScriptRunner for Jira	14/01/2021 09:41:2	In Development
230	GIC-3575	28/01/2021 16:00:3	Status		14/01/2021 09:41:2	Sprint Preparation
231	GIC-3575	21/01/2021 12:57:3	Status		14/01/2021 09:41:2	Specify and refine
232	GIC-3575	14/01/2021 09:41:4	Status		14/01/2021 09:41:2	Specify and refine
233	GIC-3575	14/01/2021 09:41:2			14/01/2021 09:41:2	Inno Backlog

Figure 31: Example of data uncertainty

(6) process improvement and support

To improve the process, the first step is to ensure the data is correct to be able to analyse it. Once can be determined with certainty that the status is the actual status, the results of the analysis can be established correctly. Therefore, improvements should first take place on the data side.

When the data is correct, the bottlenecks can be correctly identified and solved. However, to identify the bottlenecks, a clear insight into the process is necessary. This is currently not the case since the Disco visualisation is performed manually. Therefore, it is important to visualise the process in a way it represents important indicators tracking problems influencing the productivity level. In this way, adequate response to problems is possible. Recommendations on how to deal with the visualisation are given in section 3.3.

2.2.1.3. Tracking the identified problems

In this section, the problems identified by both the analysis of the reports and process mining are listed. To be able to track the problems to ensure a high productivity level, it is important to have them visible. The problems are represented in **Figure 32**. Some possible metrics are linked to the problems.

Problem	Possible metrics
<i>Long time in specific stages</i>	<i>WIP, cycle time, throughput time, Time in status</i>
<i>High average ages</i>	<i>Work item age, Lead time</i>
<i>Long waiting times</i>	<i>WIP, mean time to repair, work in progress</i>
<i>Skip testing</i>	<i>Overview of tickets in testing</i>
<i>Re-entering stages</i>	<i>Defects, failed deployments</i>

Figure 32: Problems to visualise

After the bottleneck analysis, it was asked to the employees if they could mention the problems within the process affecting productivity. A ‘full backlog’ was the only answer. An animation of the process in Disco was shown to several employees showing the bottlenecks. The discussion that arose indicated the level of awareness of the employees about the bottlenecks. There was a vague suspicion around which problems there were, but no numbers or visualisations were confirming the thoughts. There was no awareness about the fact that several tickets were skipping the testing phase. Therefore, it can be concluded that there was no insight into the problems affecting productivity at all, only thoughts.

2.2.2. The productivity level

The goal of the data visualisation is to increase the productivity of the CRM Tribe. To increase productivity, it is important to be aware of how to measure it.

When employees and managers are provided relevant data, it can be used to identify bottlenecks early and effectively to be able to reduce risks and eliminate failures. Ramirez-Mora and Oktaba (2018) describe that the number of defects found in software is a measure of software quality related to effectiveness. Identifying bottlenecks such as the number of defects and solving them with the use of metrics is how productivity is measured in Agile software development teams. Metrics are required for planning and tracking projects, measuring quality, and assessing team performance (Budacu & Pocatilu, 2018). Agile productivity refers to how well a team performs on the metrics. A lot of research is conducted on productivity in Agile software development teams since every team wants to be productive. The emergence of new methods and processes requires relevant measuring methods for better visualization and control of software development (Shah et al., 2015).

Jaspan and Sadowski (2019) encourage the design of a set of metrics, except tracking individual performances (Sadowski & Zimmermann, 2019). The use of metrics for tracking individual performances can create moral issues, which could lead to a decrease in productivity in the end. It is therefore important to focus on team performance.

The conducted analysis on the bottlenecks explains the need for measuring productivity because there are several bottlenecks identified. As described, the way to measure productivity in Agile software development teams is using metrics. Currently, there are three metrics available for the GIC project visualised in **Figure 33**; sprint burndown adobe, sprint burndown Siebel and a pie chart. Productivity cannot be measured by having only three metrics, so, therefore, there is a limited level of data visualisation. The metrics are showing the importance again of improving productivity. There is one day left of the sprint at the moment of including the pictures. Siebel rarely finishes all their story points. Adobe often finishes the sprint since the testing is finished mostly on the last day of the sprint. The pie chart is showing a backlog of 25%.



Figure 33: Currently available metrics

2.2.3. The bottlenecks and the actors involved

To understand the relation between the bottlenecks and the process, it is important to identify the actors who are involved. The focus is not on individual actors, but on the different roles and squads within the process. This is of importance because the results and processes of the different squads are different and therefore several metrics should be visualising the performance of individual squads for a better insight.

As described, there are five squads; End 2 End, Siebel, Adobe, Middleware and OCC. The End 2 End team consists of members with functions that cannot be captured within one of the specialised teams, such as the product owner, the information analyst, the UAT tester, the release manager, the Agile coach, and the Tribe Lead. Siebel, Adobe, and Middleware are the squads named after the three tools used by the CRM Tribe. The three squads are responsible for developing CRM related software, testing it, and keeping track of the integration. This is where the main bottlenecks are identified: re-entering stages, testing errors, long duration times, long cycle times and skipping phases. OCC is responsible for handling request coming in by Topdesk. When there is an issue, OCC determines whether it can be solved quickly by the OCC team or when it concerns a change. If it is a change, the ticket will be addressed by one of the specialised teams.

A ticket enters the process via the End 2 End team. Thereafter the ticket is developed by one of the specialised teams, depending on the type of ticket. When observing the process, there are differences visible between the different squads. The backlog of Siebel is bigger than both Adobe and Middleware. When planning the release period for the ticket, it stands out that Siebel is planning more story points in one period and must plan further ahead. This can already be seen in the metric visualised in **Figure 33**. Siebel has planned almost 100 story points and Adobe almost 40. This is an important indication of the importance of having metrics visualising the process per different squad.

2.3. Desired situation

The norm of the company is to have an overview with recommendations of how to visualise data and what data to visualise. Since it is not clear what bottlenecks exactly cause a lower productivity level as desired, they face challenges in determining which metrics to use for visualisations. The company needs to know what metrics to use for which reason, as well as how to use the dashboard in the process. Therefore, an advice on how to visualise data with a set of recommended metrics and explanations is what they wish, so only setting up the final dashboard is left.

The goal of the metrics is to improve productivity by visualising data. The employees of the CRM Tribe have to work with the metrics, so their lives at work must become easier. After having some conversations with several employees a set of metrics stand out. Besides, a survey had been conducted to provide the employees the possibility to share their needs.

Research question 3: what are the needs of the company for important indicators to track?

2.3.1. Desired metrics

The goal of the visualisation is to develop a solution suitable for the entire CRM Tribe. As described in the previous section, it is important to separate the metrics in the overview for the different squads. All employees are interested in different processes. Where a tester for example is interested in everything around testing, the Product Owner is more interested in an overview of tickets and their release deadlines. Therefore, to recommend a proper set of metrics, all different needs must be included.

The Tribe Lead is responsible for the overall process of the Tribe and is therefore interested in metrics that visualise the most important factors with the biggest impact on the process. What is most important according to the Tribe Lead is the well-being of the employees and quality. It is important for him that the work pressure is not too high, the employees are satisfied and happy with their job. Besides, quality is important to determine how the process is going. He is interested in whether the scheduled number of tickets can be released per Squad, how the story points are divided each Sprint and the quality of both developing and testing.

The scrum masters indicate a preference for having an overview of the quality of the work. Especially the findings per specific stage in peer review, testing, integration, user acceptance testing (UAT) and bugs in production, are of importance to identify bottlenecks. Besides, it is desired to have an overview of the UAT tickets to be able to schedule the work better. What is also desired is having an overview of the lead time of the stories in a functional release, specifically from the stages of creation to peer review so it is clear that the tickets are ready with the development.

Employees of the specific Squads indicate that it is desirable to have an overview of the findings in different stages and to see the progression of a ticket. The Siebel team is having a large backlog, so an overview of the work in specific stages is desired.

2.3.2. Survey

To provide all employees the opportunity to share their needs about the input for the dashboard, a survey is sent. Since the Tribe Lead shares his opinion about the importance of the well-being of the employees, questions about this topic are included in the survey as well. When observing the process, it became clear that the Siebel team is having a large backlog. To identify whether this large backlog has an impact on the work pressure and needs to be solved, this question has been included in the survey. The questions and the results are included in appendix C.

Important to take into account is the result of the satisfaction level about the work pressure. As some of the employees are extremely or moderately satisfied about the work pressure, others are slightly satisfied or even slightly dissatisfied. Since well-being is an important topic to the Tribe Lead to ensure a high productivity level, it is recommended to take the result into account and to take further action on this topic. The result of question 5 can be used to improve on this. Several recommendations are given on where improvements are necessary to make the job of the employees easier.

The desired metrics by the employees provide an insight into the needs for the dashboard and are presented below.

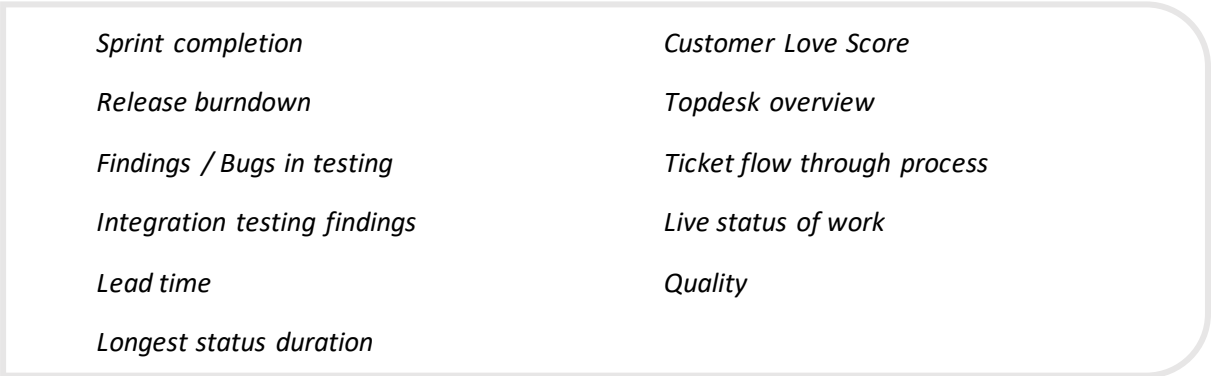


Figure 34: Desired metrics

3. Theory on improving productivity level of an Agile organised company

Chapter 3 provides suggestions on how to improve the productivity level of the CRM Tribe of A.S. Watson. Two different systematic literature reviews (SLRs) are conducted. The SLR is included in appendix D. First, suggestions from literature are given in section 3.1 on what improvements to make within an Agile organisation to improve productivity. Important indicators suggested in research to keep track of important data are provided in section 3.2. Section 3.3 contains research on the design and the implementation of a dashboard in an Agile process.

3.1. Agile optimisations on improving productivity suggested in research

The first paragraph provides suggestions by research on how to improve the productivity of software development teams. The paragraph is separated into two parts; the first part is a SLR with suggestions for software development teams and the second part is about suggestions on the framework.

Research question 4: what optimisations on Agile software development teams to improve productivity are suggested in research?

3.1.1. Systematic Literature Review

Productivity in the software industry is often defined as the amount of output per unit of input used (Kitchenham & Mendes, 2004). Productivity in software is often defined based on the three dimensions velocity, quality, and satisfaction. Productivity refers to the ratio between output and input, where the input side – the cost spent – is relatively easy to measure in software development. The challenge lies in finding a reasonable way to define output as it involves software quantify and quality (Sadowski & Zimmermann, 2019). According to Kitchenham and Mendes (2004), there are no standardized productivity benchmarks in the software industry. Having an insight into productivity is important to measure how the process is going. The reason why this is important is that software development projects often exceed time and budgetary constraints and do not even meet all user requirements (Jiang & Klein, 2000).

Research on productivity in software development has a long history. Several factors affecting productivity are relevant, such as project size, the programming language, and developer experience (Scott et al., 2020). Scott et al (2020) also report the study of Fatema and Sakib, who conducted a survey in several companies to identify important productivity factors. Their study suggests that the effectiveness of an agile team relies on multiple interrelated factors such as excellent communication, leadership, adaptability, motivation, and self-management. Agile means swift, active and responsive and Agile software development is an iterative approach to keep pace with dynamic development environments (Ahmed et al., 2010).

To understand how to improve productivity, it is first important to identify the factors affecting productivity. Wagner and Murphy-Hill (2019) describe that the factors are divided into technical factors and soft factors. The technical factors are categorised into three categories. The first category consists of factors related to the product. Examples are the development flexibility, software size and degree of complexity. The second category is the process. Examples are completion level, project duration and defects solving. The last category of factors impacting productivity is environmental related factors. Examples of this are the programming language, the use of tools and documentation. The soft factors can be divided into corporate culture, team culture, individual skills & experience, work environment and project. Examples are fairness, clear goals, communication, team cohesion, programmer capability, developer happiness, proper workplace, telecommunication facilities and team size. **Table 1** identifies factors in different areas affecting productivity researched by (Fatema & Sakib, 2018).

Table 1: Factors affecting productivity (Fatema & Sakib, 2018)

Factors affecting productivity			
Culture	Project complexity	Team Leadership	Mutual performance monitoring
Staffing	Backup Behavior	Team orientation	Adaptability
Size of team	Mutual trust	Coordination	Communication
Appreciation working long hours	Rewarding for working long hours	Adequate technical training	Adequate team skills training for team
Team member turnover	Key personnel stayed throughout the project	Staff turnover rate in project	Reuse
Software reuse level	Goals	Intra group wage inequality	Team measurement
Self-management	Task variety and innovation	External dependencies	Tools usage
Programming language	Schedule pressure	Impact of pair programming	Resource constraints
Project management	Motivation	External project factors	Dealing with cultural differences
Working environment			

There is a lot of research conducted on how to improve the currently existing frameworks, used in Agile software development teams. When making changes to frameworks, it is of importance to invest in managing, motivating, and coaching the team members involved in software development projects. Especially, because self-managed work teams have now been used for several decades and are popular as a means to make work organisations more effective and to improve productivity. Evidence suggests, however, organisations remain sceptical and even dismissive of self-organised teams. The reasons, it is suspected, is that handing away responsibility for much of the decision-making to team members is perceived as a high risk (Parker et al., 2015).

A study by Kola (2014) researched the thoughts of Scrum Masters on how to improve the productivity of an Agile software development team: measure productivity, find the root causes of possible problems and fix them, eliminate useless work activities in favor of high-value activities, assign the right tasks to the competent people; in other words letting everybody work on what they are best on, motivate the team members to commit on what they are doing, ensure customer contact for the team members, use the agile to improve the team working efficiency and improve the team velocity continuously, involve testing soon and make it as automatic as possible, focus on training people and building competence, focus on improving the productivity of the teams rather than that of specific individuals, organize small activities that make the team happy (sprint celebrations), consider change management.

Several suggestions based on research to improve the productivity of important aspects of an Agile software development team are provided.

Interruptions

The first topic on which can be improved according to the literature is interruptions. In software development projects, agile approaches are often used to manage the project by using daily standups, retrospectives, and other agile tools. Agile software development projects increase collaboration with different stakeholders in software development project teams. This collaboration has many benefits for the agile team, as it ensures that customer needs are met, it fosters knowledge sharing, and it increases employee motivation (Wiesche, 2021). However, Wiesche (2021) describes that the openness toward change and the high degree of collaboration also cause interruptions for the software development team, described as events that impede or delay organizational members during work tasks. Examples of interruptions are being asked for help or feedback by a team member, changing requirements or waiting for input. As far as researched nowadays, recovering from interruptions is a central problem among software development teams. To identify the way a team is affected by interruptions in three different contexts, a study has been conducted. Wiesche (2021), identifies the contexts (1) programming-related work impediments, such as changed requirements, errors, and developers waiting for information; (2) interaction-related interruptions, such as customer requests and formalized meetings; and (3) interruptions imposed by the external environment, such as technology-induced interruptions and interruptions caused by the work environment. All teams participating in this research are using practices embedded in the agile mindset.

When dealing with interruptions, it is important to stick to the rules prescribed by the framework. The results of the study highlight the importance of the roles in agile software development teams. The scrum master needs to possess the resources and power to protect the team from too many interruptions. Further, the product owner needs to channel interruptions during the software development process to help agile software development teams cope with interruptions (Wiesche, 2021). Another possibility to deal with interruptions is to allocate a portion of time to interruptions (Berteig, 2020). To determine whether this is necessary, it is important to track the occurrence of interruptions. It is possible to allocate time to interruptions in two ways, by giving everyone in the team some time or by giving one or two people time to handle interruptions.

Data

Improving the productivity level of software development teams is one of the most important objectives all software companies are dealing with. In this section, several suggestions are provided on how to improve productivity with the use of data.

With the increasing popularity of Big Data, which throughout the years has successfully entered the realm of computer science and business environment alike, software practitioners can now efficiently use it to improve software development processes (Biesialska et al., 2021). When optimising development cycles by using data analytics, it can help to streamline the daily business operations of a company. A study has been conducted by Biesialska et al. (2021), to identify tools to using Big Data Analysis in Agile software development. Many works incorporated additional functionalities to existing tools or integrated with solutions already in use by the studied organizations. In general, Eclipse plug-ins were especially well-represented among the tools, followed by JIRA plug-ins and, to a lesser degree, Azure DevOps extensions (Biesialska et al., 2021). Visual analytics in software engineering mainly provides insights about intangible software product artifacts (such as program runtime behavior or software evolution), while software process visualization is far less popular and deserves more attention (Biesialska et al., 2021). Furthermore, numerous studies proved that visual representation of data often offers the kind of insights that are especially useful for decision-makers, such as dashboards visualising most vital metrics for ASD processes and KPIs. This is essential besides Big Data Analytics.

Retrospective meetings are nowadays a popular practice in software development teams. Activities that take place in these meetings often do not rely on project data, instead depending solely on the perceptions of team members (Matthies et al., 2020). Research by Matthies et al. (2020) proposes new Retrospective activities, based on mining the software repositories of individual teams, to complement existing approaches with more objective, data-informed process views. Most proposed Retrospective exercises focus on gathering the perceptions and experiences of team members and extracting improvement opportunities from them. Another view of the project reality is available through the artifacts that are produced by software developers in the course of their daily work (Matthies et al., 2020). A tool-type suggested for Jira is Issue Tracker, providing the possibility to manage detailed information on work items. Data extracted from such tools are useful for process improvements because of the provided evidence for project problems. A large-scale analysis of this data is the focus of Mining Software Repositories.

Software development teams have usually a lot of data available in different kinds of repositories. The data can be mined as suggested by Biesialka et al. and Matthies et al. The result can be used for software process improvement purposes. As the human mind is powerful in interpreting visual representations, visualizations could help in recognizing problems and areas of improvement in an agile software development process (Lehtonen et al., 2013). Research by Lehtonen et al. on visualization was used to identify problems in sprint length, release cycle, task size, lead time and communication between the development team and the customer. The biggest challenge described by Lehtonen et al. is to export data easily from Jira to be able to analyse it. Jira content was transformed to a visual form as follows. First, a Jira filter was applied to find issues from the past two years. The issue event data consists of state changes, comments, and assignments. However, it turned out to be that there was a better connection needed to access issue event data. This has been done by a custom Jira Web Crawler with robot frameworks and python scripts. The visualization presented contains a lot of information, with tens of tasks and hundreds of events shown in a compact form (Lehtonen et al., 2013).

The employees

Research suggests improving the happiness level/satisfaction rate of employees to improve productivity. A study by Graziotin and Fagerholm (2019) shows that most developers are moderately happy: happy developers are supposedly more productive and, hopefully, also retained. Perks, playground rooms, free breakfast, remote office options, sports facilities near the companies...there are several ways to make software developers happy (Sadowski & Zimmermann, 2019).

A study by Ramirez-Mora and Oktaba (2018) describes that almost all identified factors impacting productivity are team process, attitudes, and affections among team members. It was found that most of the factors are characteristics of "mature teams". Social science exposes that "mature teams" can reach high productivity levels and are characterized by open communication, team cohesiveness, mutual support, mutual trust, shared leadership and effective conflict management (Ramirez-Mora & Oktaba, 2018). To improve productivity, it is suggested to focus on those aspects.

Each developer of a development team feels different about productivity which makes it more challenging to determine which actions could help increase the productivity of a team. To better understand the different feelings, a study has been conducted. Analyzing productivity ratings from hourly self-reports during three workweeks, it is found that developers can roughly be categorized into three groups that are similar to the circadian rhythm: morning person, afternoon person, and low-at-lunch person (Meyer et al., 2019). Besides, the developers can be clustered into six groups based on personality: social, lone, focused, balanced, leading, and goal-oriented. A first step to increasing

productivity is to build self-monitoring tools allowing the developers to increase their awareness about both productive and unproductive behaviours to use the insight to set goals for self-improvements. For the future, it could be possible to build virtual assistants, such as Alexa for Developers, that recommend (or automatically take) actions, depending on the goals of developers or based on the productivity patterns/roles/clusters of developers (e.g. blocking notifications for the lone developer, but allow them for the social developer) (Meyer et al., 2019). By knowing the trends and characteristics, it might be possible to change schedules that best fits the work patterns.

Another important aspect suggested to improve on is team awareness. Currently, there are tools available to provide mostly quantitative information that helps developers making sense of everything that goes on in a project. However, to accurately represent what goes on in a project, awareness tools are important, focusing on summarising the most important information. It is suggested that such tools should categorise the events in a software project according to whether they are expected or unexpected and use natural language processing to provide meaningful summaries rather than numbers and graphs that are likely to be misinterpreted as productivity measures (Treude & Filho, 2019)

Improving the Agile frameworks

As identified, several factors are having an impact on productivity. Some of them are concerning programming languages, coding, etc. To improve on them to increase productivity, it might yield to use existing methods, understanding programmers. Metrics are providing insights into the process, human-computer interaction (HCI) methods can help better understanding programmers. Myers et al. (2019) identified ten possible methods to use for this purpose.

1) *Contextual inquiry*

Used to understand barriers in context. The experimenter observes developers performing their real work where it actually happens and makes special note of breakdowns that occur.

2) *Exploratory lab user studies*

The experimenter assigns specific tasks to developers and observes what happens.

3) *Surveys*

To observe the frequency of an observed barrier, surveys could be used. The survey can be counted how often developers have questions about control flow and how hard those questions are to answer

4) *Data mining (including corpus studies and log analysis)*

As an example, the study by Sadowski and Zimmerman felt that programmers were wasting significant time trying to backtrack while editing code. When analysing code-editing logs the instances could be tracked.

5) *Natural – programming elicitation*

A way to understand how programmers think about a task and what vocabulary and concepts they use so the intervention can be closer to the users' thoughts. A method is to give a blank paper where the desired functionality is described and the programmers design how that functionality should be provided

6) *Rapid prototyping*

Allows quick and simple prototypes of the intervention to be tried, often just drawn on paper, which helps to refine good ideas and eliminate bad ones

7) *Heuristic evaluations*

Ten guidelines to evaluate an interface

- 8) *Cognitive walk-throughs*
Involves carefully going through tasks using the interface and nothing where users will need new knowledge to be able to take the next step
- 9) *Think-aloud usability evaluations*
The participant continuously articulates their goals, confusion and thoughts. This provides the experimenter with rich data about why users perform the way they do so problems can be found and fixed
- 10) *A/B testing*
Statistically valid experiments to demonstrate that one intervention is better than another, or better than the status quo.

Another important element on which can be improved is waste. Software waste refers to project elements (objects, properties, conditions, activities, or processes) that consume resources without producing benefits (Sedano et al., 2019)(Sadowski & Zimmermann, 2019). Wastes can be compared with frictions in the development process. Two steps in waste are important: waste awareness and identification. During a study conducted by Sedano et al. (2019), nine main types of waste in agile software projects were identified:

- 1) *Building the wrong feature or product*
Techniques to avoid or reduce waste include usability testing, feature validation, frequent releases and participatory design
- 2) *Mismanaging the backlog*
Solutions include prioritising backlog several times a week, minimizing work in progress by finishing features before starting new ones, update the backlog with current work in progress
- 3) *Rework*
Solutions include continuous refactoring, reviewing acceptance criteria before beginning a story, and improving testing strategy and root-cause analysing on bugs
- 4) *Unnecessarily complex solutions*
Solutions include simpler designs for user interaction and software code
- 5) *Extraneous cognitive load*
Solutions include refactor code that is difficult to understand, decompose large, complex stories into smaller, simpler stories, replace hard-to-use libraries, work on one task at a time until it is completed; avoid “blocking” tasks (i.e., putting a task on hold to work on something else), improve the development flow including better scripts and tools
- 6) *Psychological distress*
Solutions include simply asking team members “how are things going” and reducing scope or extending deadlines for stress related to deadlines
- 7) *Waiting/multitasking*
Solutions include limiting work in progress and taking breaks instead of task switching
- 8) *Knowledge loss*
Solutions include code review and pair programming
- 9) *Ineffective communication*
Solutions include having face-to-face synchronous communications and conversational turn-taking

Dashboards

There are many kinds of information that developers should be aware of in a software development project. However, with the amount of information available, it is hard and often not necessary to be aware of all details of a project. Important is to use dashboards to focus on quantitative data about the issues. Especially in agile teams are dashboards important for managers. Stephen Few defines a dashboard as “a visual display of the most important information needed to achieve one or more objectives which fit entirely on a single computer screen so it can be monitored at a glance” (Storey & Treude, 2019)(Sadowski & Zimmermann, 2019). It is possible to measure from different perspectives. Storey and Treude (2019) describe the different dimensions as developer activity, team performance, project monitoring and performance, and community health.

Besides tracing productivity by using a dashboard, it is also possible to use a tool for the employees to self-monitor their own work. There is a wide variety of factors that impact success and productivity at work, like interruptions for example. Self-monitoring is proved to be successful to increase awareness about productivity in other domains.

Pair programming

The definition of pair programming is that two programmers are working closely together on a programming task. Zieris and Prechelt (2019) studied the effectiveness of paired programming. They discovered that frequent pair programming is an excellent technique to make sure system knowledge spreads continuously across a team.

Add-ons

A wide variety of add-ons for Jira are developed. To accelerate Jira productivity, some of them are suggested.

1) *Mail Me*

Communication is key, so Mail Me provides the opportunity to send issues to internal and external users while still on Jira.

2) *Checklist*

Provides the possibility to track smaller tasks. It is possible with this add-on to create custom checklists to track issues and manage different statuses.

3) *Structure*

An add-on turning chaos into clarity. With grouping, sorting, filtering, and other rules structures can be built.

3.1.2. Suggestions on the framework

The CRM Tribe of A.S. Watson is using a framework mainly based on the principles of Scrum, a Kanban board, and principles from the Nexus model and the Spotify model. Scrum focuses on the management of development and does not prescribe specific development techniques: therefore, it can easily be combined with other approaches, such as XP (Vogel & Telesko, 2020). A suggestion for improving the current model could therefore be the implementation of some practices of XP. Vogel and Telesko (2020) describe that the following practices can be used together with Scrum:

- 1) Pair programming
- 2) Test-driven development (described as the most useful XP practice)
- 3) Coding standards
- 4) Sustainable pace/energised work

Another framework that deserves to be considered is the combination of Agile with DevOps. DevOps is a set of practices that work to automate and integrate the processes between software development and IT teams, so they can build, test, and release software faster and more reliably (Atlassian, 2019). Atlassian (2019) explains the DevOps lifecycle as a cycle consisting of six phases, representing the processes, capabilities, and tools needed for development on the left side of the loop, and the processes, capabilities, and tools needed for operations on the right side of the loop. Will Kinarad mentions that when it comes to developing software faster and maintain software more efficiently, both agile and DevOps have a big role to play (Ismail, 2018).

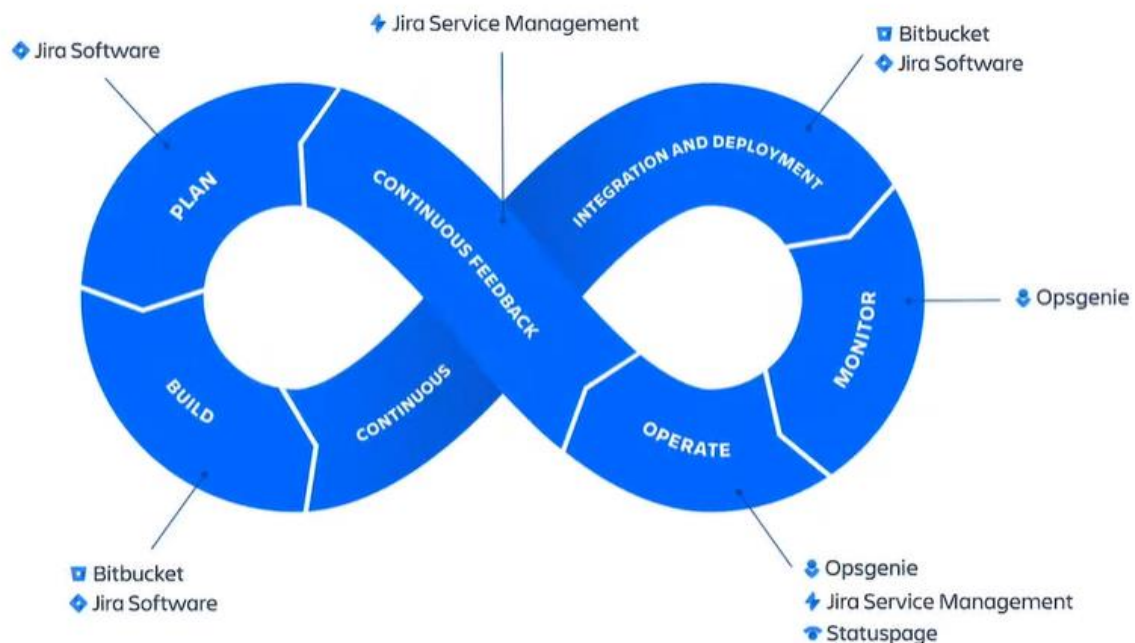


Figure 35: DevOps cycle (Atlassian, 2019)

3.2. Suggested indicators by research

Research question 5: what indicators are suggested for agile software development companies?

To solve the core problem of having a limited level of data visualisation to identify known and unknown problems, a dashboard is needed. To keep track of the process with the dashboard, indicators are needed. The purpose of the data visualisation is to improve the productivity level in the end. The productivity level is tracked with performance indicators. First, a SLR is conducted providing suggestions for indicators to use in software developing companies. Second, suggested metrics are provided based on the opinions of experts.

3.2.1. Suggestions from literature review

An indicator is a common term. In Agile software developing teams, the term metric is used for indicators, which tracks a specific process. Metrics are required for planning and tracking projects, measuring quality, and assessing team performance (Budacu & Pocatilu, 2018). According to Kupiainen et al., (2015), the reasons for and the effects of using metrics are focused on the following areas: sprint planning, progress tracking, software quality measurement, fixing software process problems, and motivating people, which has also been identified by Budacu and Pocatilu, (2018). In **Table 2** an overview of metrics, ranked on importance, is provided based on the literature study conducted by Kupiainen et al., (2015). Metrics were selected as important during the literature study if the author praised the metric if there were signs of continuous use or if there was a positive correlation to important output.

Table 2: Metrics

Metric	
Velocity	Processing time
Effort estimate	Defect trend indicator
Customer satisfaction	Work in progress
Defect count	Number of unit tests
Technical debt	Cost types
Build status	Variance in handovers
Progress as working code	Deferred defects
Lead time	Predicted number of defects in backlog
Story flow percentage	Test coverage
Velocity of elaborating features	Test-growth ration
Story percent complete	Check-ins per day
Number of test cases	Cycle time
Queue time	

Research by Kurnia et al., (2018) suggests that software metrics can improve the development process, produce a better project, and control the quality, and improve the estimation process. Software metrics can be used to provide an objective view and to evaluate the various impacts of software development process changes. In addition, software metrics also provide benefits to review project progress, monitor the product quality, and forecast the project and the project management. Kurnia et al., (2018), provides a set of software metrics, provided in table 3, to measure the performance on each event based on four perspectives of measurement, respectively sprint planning, daily sprint, sprint review and sprint retrospective.

Table 3: Metrics classified in four perspectives

Sprint planning metrics	Daily sprint metrics	Sprint review metrics		Sprint retrospective metrics
Effort estimate	# of an open defect	# of defects found in system test	Focus factor	Earn value management
Story point	Contribution	Bug correction time from new to the close state	Fulfillment of scope	Impression
Task effort	The ratio of work spent and work remaining	Business value delivered	Number of stories	Influence
Task's expected and end date	Standard violation	Customer satisfaction	Open defect severity index	Job satisfaction
Velocity	The release burndown chart	Completed web pages	% of adopted work	Net promoter
	The sprint burndown chart	Defects deferred	% of found work	
		Defect per iteration	Progress chart	
		Delivery time	Unit test coverage for developed code	
		Error density	Work capacity	

The two most popular metrics in the sprint planning stage are story point and velocity. Release burndown and sprint burndown are the two most important metrics for the daily sprint stage. In the sprint review process, customer satisfaction and business value delivered are the most popular. For the last stage, job satisfaction and earn value management (Kurnia et al., 2018). (Padmini et al., 2015) identifies “delivery on time” was the most used metric according to their literature study. The result of recommended metrics based on their literature study is as follows: delivery on time, work capacity, unit test coverage for the developed code, percentage of adopted work, bug correction time from new-to-closed state, sprint-level effort burndown, velocity, percentage of found work, open defect severity index and focus factor.

Agile software development continues to grow. There is a need for empirical evidence on the impact, benefits and drawbacks of an agile transformation (Olszewska et al., 2016). The metrics apply to projects of any size, complexity, and scope and for measuring past and ongoing projects. If the company is shifting towards more agile principles, the following set of metrics is suggested by Olszewska et al., (2016) and by Heidenberg et al., (2013): Customer Service Request (CSR) turnaround time, lead time per feature, functionality/money spent, Number of releases/time period, commit pulse, flow metric, number of external trouble reports and average number of days open external trouble reports.

3.2.2. Suggestions by experts

When reviewing articles and blogs from experts and JIRA users, several metrics are highlighted. It is recommended to divide the metrics into quality, productivity, and performance. Three types of metrics are identified: Scrum metrics, Kanban metrics, and Lean metrics. Since the Tribe is using Scrum and Kanban principles, the focus is on those two categories. Scrum metrics focus on the delivery of working software to customers and Kanban metrics focus on workflow, and the organisation of work and getting it done.

The most common mentioned metrics are:

- 1) Velocity
- 2) Cycle time
- 3) Lead time
- 4) Sprint burndown
- 5) Epic and release burndown
- 6) Issue burndown
- 7) Control chart
- 8) Cumulative flow diagram
- 9) Sprint health gadget
- 10) High priority issues
- 11) Time spent including waiting time
- 12) Defects
- 13) Work Item Age
- 14) Throughput
- 15) Escaped defects
- 16) Failed deployments
- 17) Story points planned
- 18) WIP

3.2.3. Results

“When combining all key recommendations applicable to the Tribe, the following set of metrics arise”

Table 4: Set of Metrics

Metrics		
Velocity	Cycle time	Escaped defects
Story points	Ratio of work spent and remaining	Failed deployments
Number of test cases	Release burndown	Time spent including waiting time
Defect count	Sprint burndown	Work item age
Queue time	Number of stories	Throughput
Build status	Control chart	Processing time
Work in Progress	Cumulative flow diagram	Story percent complete
Lead time	Sprint health gadget	Number of unit tests
Story flow percentage	High priority issues	

3.3. Data visualisation

In this section, all details around data visualisation are provided. First, it is important to export data in a way it is useable for data visualisation and analysis. Suggestions are provided for this. Besides, suitable options for dashboard tools are given. Second, suggestions are given on the type of charts to use and last suggestions are provided on how to implement the use of a dashboard in an Agile process.

Research question 6: what constitutes a dashboard to improve productivity of an agile process and how to implement this?

3.3.1. Analysis and visualisation

Preparing data

The company is using Jira Cloud as project management tool. All tickets and their details can be found here. A picture of how this looks is visualised in **Figure 50** included in appendix E. However, it is hard to easily export data in a way it is usable for further analysis, or for a dashboard that is not connected with Jira. When pressing the export button, data is exported to excel in the following way. To explain why this is a problem, a screenshot is included in **Figure 36**. Many users describe the export of data from Jira as a problem. Therefore, a tool to export data in a structured way is necessary. Below the picture, two suggestions are provided.

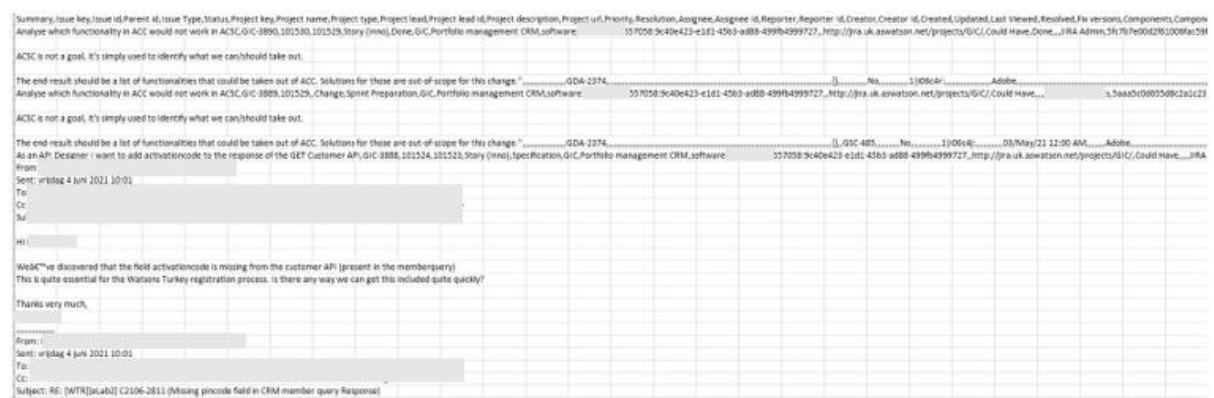


Figure 36: Result of exporting data from Jira

1) Advanced Export (add-on)

For the Disco analysis, data was exported with the free trial of the tool **Advanced export**. With Advanced Export for Jira, unlimited number of issues can be exported easily in a spreadsheet or other tools. This includes the history of changes, which is often the most important input for further analysis. Standard fields can be exported, as well as customised fields. The advantage of this tool is that it is an add-on provided by Atlassian Marketplace. Atlassian is an Australian company that developed the tool Jira. This means the tool can easily be integrated within Jira. When only the CRM Tribe would use it, it will cost between \$25 and \$37.5 per month.

2) Python

There is a lot of information on issues in Jira which is interesting to analyse. This method is based on a Python code to extract the desired data from Jira into a CSV file. Python is a general-purpose programming language. To export data from Jira into Excel, a code is developed based on an existing code (Dmitriev, 2018). The code is written in Spyder; a free and open-source scientific environment written in Python. Besides, the Jira library is needed for the code.

A practical way to start with coding is to use the open-source distribution platform Anaconda. A screenshot of the platform is provided in **Figure 51** which is included in appendix E. After installing Anaconda, CMD.exe.prompt can be launched via Anaconda in which the Jira library can be installed. A visualisation of this is provided in **Figure 52**. Then it is time to launch Spyder. The code is also provided in appendix E. Data is automatically exported to CSV, depicted in **Figure 37**.

The advantage is that it is an open-source platform, and it is possible to program the code in all different ways to export the desired fields. This makes this method also suitable for exporting data for other goals within the company.

N	Issue key	Datetime creation	From Status	To Status	Status Changed@
0	GIC-3873	20-5-2021 14:24	In Development	Delivered	31-5-2021 09:24
1	GIC-3873	20-5-2021 14:24	Sprint Preparation	In Development	20-5-2021 14:30
2	GIC-3873	20-5-2021 14:24	Inno Backlog	Sprint Preparation	20-5-2021 14:27
3	GIC-3856	5-5-2021 08:51	Sprint Preparation	Delivered	20-5-2021 12:07
4	GIC-3856	5-5-2021 08:51	Refinement	Sprint Preparation	5-5-2021 08:53
5	GIC-3856	5-5-2021 08:51	Inno Backlog	Refinement	5-5-2021 08:52
6	GIC-3820	6-4-2021 10:36	Staging UAT	Delivered	17-5-2021 15:24
7	GIC-3820	6-4-2021 10:36	Delivered	Staging UAT	19-4-2021 13:47
8	GIC-3820	6-4-2021 10:36	In Development	Delivered	19-4-2021 10:03
9	GIC-3820	6-4-2021 10:36	Sprint Preparation	In Development	6-4-2021 10:40

Figure 37: Format of exported data with python

Visualisation tools

The set of metrics serve as the input for the dashboard. In this section, several options for dashboards are tested on requirements so a recommendation can be given on what tool to use. The tools and the requirements are represented in **Table 5**. First, the different selected tools are explained.

The first tool is the Jira dashboard, which is a dashboard created in Jira cloud. Personal dashboards for different projects can be created by adding gadgets to keep track of. Geckboard is a cloud-based dashboard tool displaying key business metrics in real-time. The third option, powermetrics, is a tool providing real-time data to visualise data. Klips is a cloud-based platform to make visualisations from important metrics. Screenful is the fifth option and provides the opportunity to visualise and track productivity using data from existing tools. Kanban analytics provides the opportunity to connect with Jira and visualise actionable insights from the data. The seventh option is EazyBI for Jira, which is an add-on available on Atlassian marketplace. It provides the opportunity to analyse & visualise many metrics. Tableau is a popular platform used for dashboards on which all preferred data can be visualised. Kibana is a tool that provides the opportunity to rapidly create views that pull together charts from real-time data. As Excel is probably familiar to everyone, not everyone is familiar with the possibilities to design charts to create a dashboard within Excel. The last tool from the list is an add-on providing a single report; time in status, which serves as a comparison explained below the table.

Table 5: Balanced scorecard for dashboard tools

	Costs	# Users	Real-time	Degree of customisability	Ease to use	Ease to install	Public/private cloud	Integratable with Jira
Weighing factor	2	1	1,5	3,5	3	2,5		
Jira dashboard	10	8	8	3	8	10	Private	Exclusive
Geckoboard	6	4	8	5	9	10	Public	Native integration
Powermetrics	2	6	8	6	6	8	Public	Native integration
Klips	8	6	8	10	3	6	Public	N/A
Screenful	8	10	8	6	8	8	Public	Yes
Kanban analytics	4	4	8	7	8	8	Public	Yes
EazyBI for Jira	8	4	8	7	8	6	Public	Yes / exclusive
Tableau	6	10	6	10	3	3	Private	No
Kibana	6	10	6	9	6	3	Private	No
Excel	10	10	6	8	4	2	N/A	No
Time in status	2	6	8	1	8	8	Public	Yes

The first four requirements speak for themselves. Ease to use means how user-friendly the tool is. Ease to install determines how much effort is needed before having the desired metrics. If the tool is not integratable, a connection needs to be programmed. Each requirement received a score after comparing all tools and a weighing factor. The requirements are weighted based on importance by the Tribe Lead. **Table 6** represents the results of the balanced scorecard method.

Table 6: Results of balanced scorecard

Rank	Tool	Total	Public/private cloud	Integrability
1	Screenful	103	Public	Yes
2	Jira dashboard	99,5	Private	Exclusive
3	Geckoboard	97,5	Public	Native integration
4	EazyBI for Jira	95,5	Public	Yes / exclusive
5	Klips	93	Public	N/A
6	Kanban analytics	92,5	Public	Yes
7	Kibana	88	Private	No
8	Excel	84	N/A	No
9	Tableau	82,5	Private	No
10	Powermetrics	81	Public	Native integration
11	Time in status	69,5	Public	Yes

Instead of choosing a dashboard, it is also possible to use the alternative of single reports such as the add-on “time in status” to visualise a process. However, the low result of the single add-on “time in status” visualises the importance of having a dashboard. Since time in status is an expensive tool, it is really expensive to purchase several single add-ons.

EazyBI is an add-on for the Jira dashboard. This makes the selection the following: Screenful, Jira dashboard with the combination EazyBI for Jira, and Geckoboard are suitable solutions for the dashboard tool, based on a public cloud. Kibana is a suitable option with a private cloud. Tableau has a high level of customisability and is on a private cloud, so could be suitable if this is highly desired. The tools on a public cloud should be checked by the privacy department of the company.




3.3.2. The design

The purpose of the dashboard is to visualise important processes identified by the bottleneck analysis, the needs of the employees, and literature. The goal is, by visualising the data, to improve the productivity level of the CRM Tribe. As analysed, it is important to make a distribution between the different squads. This gives the Tribe Lead the opportunity to analyse the process of the whole Tribe and more specific per Squad and it provides the Scrum Master with the opportunity to discuss the process with the Squad members. A clear layout is of importance to achieve the best results while using a dashboard in your process.

Guidelines for designing a dashboard	Tips
1) Include only the most important metrics	<i>Less is more</i>
2) Make use of size and position to show importance	<i>Big + top left</i>
3) Give context to numbers	<i>Compare with previous data</i>
4) Group metrics related to each other	<i>Quality / Productivity / performance</i>
5) Be consistent with layouts and visualisations	<i>Use same type of charts</i>
6) Clear labels	<i>Use headings</i>
7) Round numbers	<i>Use symbols/abbreviations</i>

Figure 38: Dashboard design guidelines

There are different types of charts with different purposes of visualisations. When choosing the wrong type of chart for a specific metric, the data can be interpreted incorrectly. Therefore, it is important to decide which chart to use for which metric. An overview of the types of charts is provided with an explanation of when to use the specific chart.

<p>1) Line graphs</p> <p><i>Illustrates change and comparisons over time to reflect trends</i></p> <p>Add multiple (different coloured) lines for comparisons and consider shading the areas under the lines</p>	
<p>2) Bar charts</p> <p><i>Illustrates comparisons</i></p> <p>Comparing data across categories and use (different coloured) stacked bars or side-by-side bars</p>	
<p>3) Pie charts</p> <p><i>Illustrates proportions / percentages of information</i></p> <p>Do not use for comparisons, but for proportions and limit it to six categories</p>	

4) Gauges

Illustrates the move towards a goal

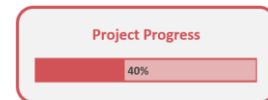
Goal threshold below maximum and set a negative threshold, use red and green to make a distinction between bad and good



5) Progress bars

Illustrates the progress towards a goal

Only use for key, fundamental team goals and provided details about which process the bar is going



6) Colour-coded alerts

Highlight critical metrics

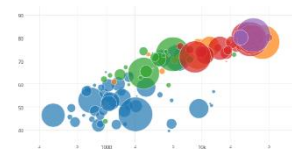
Only use this for critical metrics and consider using an icon to ensure everyone interprets the chart correctly (colour-blind)



7) Scatter plots

Illustrates relationship between different variables

Use to spot outliers that might indicate further investigation. Consider using (different coloured) bubbles as a technique to accentuate data



8) Histogram

Illustrates the distribution across groups

To understand the distribution of the data, use this type of chart. Consider adding a filter to drill down into categories

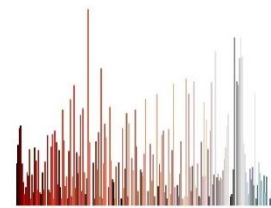


Figure 39: Overview of charts

3.3.3. Implementation

A lot of research suggests using a dashboard during the retrospective meetings with the Scrum master and the entire Squad. In this meeting, it is relevant to have a dashboard visualising the most important processes related to the specific Squad. During a sprint, all employees can regularly check the dashboard themselves. Besides, the Tribe Lead needs to track the process by using the dashboard. To discuss this with the team, the dashboard can be used during the Scrum Master meeting.

4. Improving the productivity level of the CRM Tribe by using suggested metrics

This chapter describes the result of all analyses and provides the solution of how to analyse and visualise the productivity level of the CRM Tribe. Based on bottleneck analyses, personal needs, and literature research, a set of metrics is developed so the Tribe can analyse their productivity by having an overview of the bottlenecks. The overview gives the employees an insight into the problems so that all problems are known to the employees such that adequate response is possible.

4.1. The metrics

The bottleneck analysis showed the importance of visualising the duration of the tickets in the process and between the different stages. Besides, it showed that an important phase, the testing phase, was skipped too often and that several tickets re-entered a specific stage a lot of times. Both bottlenecks are related to quality and are therefore a strong indication to use metrics visualising these processes. As a result of the conversations with the different employees about their needs, they indicate a need for quality and progression. Developing quality and testing quality are both mentioned by the employees. Besides, the need is to track easily whether the process is on track or not, on a general Tribe level and a Squad specific level. Literature research suggests different metrics. The suggestion is to focus the metrics on three categories: quality, productivity, and performance. **Figure 40** shows the suggested set of metrics.

Category	Metric	Chart type
Quality	Escaped defects	Bar chart / colour-coded alert
Quality	Failed Deployments	Bar chart / colour-coded alert
Quality	Findings per stage	Bar chart / colour-coded alerts
Quality	Skipping phases	Colour-coded alert / Pie chart
Quality	Mean time to repair	Colour-coded alert / line graph
Quality	Ticket flow	Bar chart
Productivity	Velocity	Bar chart / line graph
Productivity	Lead time	Line graph
Productivity	Sprint Burndown	Line graph
Productivity	Release burndown	Line graph
Productivity	Days in Status or average age	Histogram / Bar chart / Pie chart
Performance	Cumulative Flow	Line graph with shaded areas
Performance	Release/Sprint health	Gauge / progress bar
Performance	High priority issues	Colour-coded alert
Performance	Progression of testing	Bar chart
Performance	Customer value	Colour-coded alert

Figure 40: Recommended metrics

The different metrics have different purposes. Therefore, a list with both explanations, and some instructions for the metrics is provided.

1. Escaped defects

Helps to identify the number of bugs in a build/release after the ticket enters production. The ideal factor is zero. Since production bugs are often a problem in agile processes, it is an important metric. It could be divided into production findings and change tickets. Currently, it is not possible to visualise this metric because of the way information is logged in Jira. To be able to visualise this metric, the findings should be logged to identify the escaped defects.

2. Failed Deployments

Provides a clear indication of whether tickets in sprints or releases are ready for production, or not. It indicates the quality of the software that will be delivered. Similar to escaped defects, the way of logging in Jira should be changed for this metric as well, to be able to visualise the failed deployments.

3. Findings per stage

Provides an overview of the quality of the different processes, such as developing and testing. It is suggested to monitor the transitions between different stages, especially when a ticket goes back into the process to a stage it already passed. When monitoring for example the transition from a ticket in developing to testing, back to developing, the quality of the development phase can be monitored. By doing this, bottlenecks can be identified quickly.

4. Skipping phases

Helps to identify whether a ticket is skipping an important phase or not. If a ticket is skipping the testing phase for example, it should give a notification so it can be checked whether it is allowed to skip testing. This metric helps to optimise quality of the software delivered.

5. Mean time to repair

The average time it takes to repair a finding. Helps to identify the duration time which seems very long in some cases, as analysed by process mining. Unplanned maintenance time and the total number of failures are needed to calculate this metric.

6. Velocity

A metric allowing to assess the average amount of story points finished in a sprint. It indicates the amount of work the team is able to complete within a timeframe. A decrease in velocity signals a decrease in productivity and vice versa. A consistent velocity can be used to forecast and plan for the amount of work in the backlog.

7. Lead time

The metric provides the opportunity to monitor a ticket from entering the backlog to the moment of release. The lower, the more productive the process is. It is useful to identify tickets with longer lead times than others to prevent long unnecessary duration times.

8. Ticket flow

Visualises the path of the tickets to identify who is responsible for which status. It contributes to the quality of the software since it is clear whether different employees tested a ticket for example.

9. Sprint Burndown

When a sprint starts, a number of story points are scheduled to finish in the sprint. This metric provides the opportunity to identify whether the Squad is on track or not.

10. Release burndown

Instead of monitoring the progression of story points per sprint, this metric allows to track the progression of a release

11. Days in Status or average age

Tracking the time each ticket spends in a specific status. This helps to identify long duration times. Could be divided into the development process and testing process. This is important to identify bottlenecks in specific statuses and it shows the problem of the large backlog.

12. Cumulative Flow

Helps to get a view of the status for each task in a sprint, release and across software squads. Bottlenecks per status can easily be identified.

13. Release/Sprint health

Provides a summary of the progress of the sprint and the story points. A quick indication to see whether everything is on track or not.

14. High priority issues

Preventing missing important tickets by giving them priority by flagging the ticket. Currently, there is a division in must have, should have, and could have. However, it is possible to give priorities to tickets in Jira. This is necessary to visualise this metric.

15. Progression of testing

This metric could help to identify the status of testing. It is currently not visible how far the ticket is with testing, resulting in a lack of overview for the testers. To be able to visualise this metric, the add-on checklist could be an option to make this visible. Another option is to split the status of testing into more sub statuses.

16. Customer value

To identify the value of the work, surveys can be sent to the BUs and the average scores can be added as a metric. The goal of the company is to put a smile on its customers' faces. Therefore, a customer love score is important as a measure to identify the value of the software delivered to the customers. This could work as a stimulation to the employees as well since they are not directly in contact with the customers. By seeing this customer value score, they see the value of their work to their customers.

4.2. The dashboard

To start with using dashboards, it is important to make a distinction between a CRM Tribe Dashboard and Squad dashboards. From the set of metrics, a recommendation is given on the most important metrics suitable to the CRM Tribe to use on which dashboard, represented in **Figure 41** and **Figure 42**.

<i>CRM Tribe dashboard</i>		
Functional release health	Release burndown	Velocity
Escaped defects	Customer value	Findings per stage
Failed deployments	Mean time to repair	Lead time
Ticket flow	Cumulative flow	Days in status

Figure 41: Metrics for CRM Tribe dashboard

<i>Squad dashboards</i>		
Sprint health	Skipping phases	High priority issues
Sprint burndown	Progression of testing	Days in status

Figure 42: Metrics for CRM Tribe dashboard

To realise the dashboards, the first step for the company is to determine which tool to use and to purchase a license. Based on the analysis, there are five possibilities selected following from the balanced scorecard method. Most important considerations are costs, customer friendliness and public/private cloud. The second step is to make changes in the way of logging. Some important information is currently missing to be able to construct several metrics. Priority should be logged to visualise high priority tickets and all findings should be clearly logged as well. After the tool is purchased and the way of logging is changed for specific processes, the dashboard can be constructed. The tool EazyBI is used to visualise how a dashboard would look like, by showing a demo in **Figure 43**. In this demo, only the metrics currently possible to construct with the given data are visualised. A zoomed-in version of the dashboard is provided in appendix F.

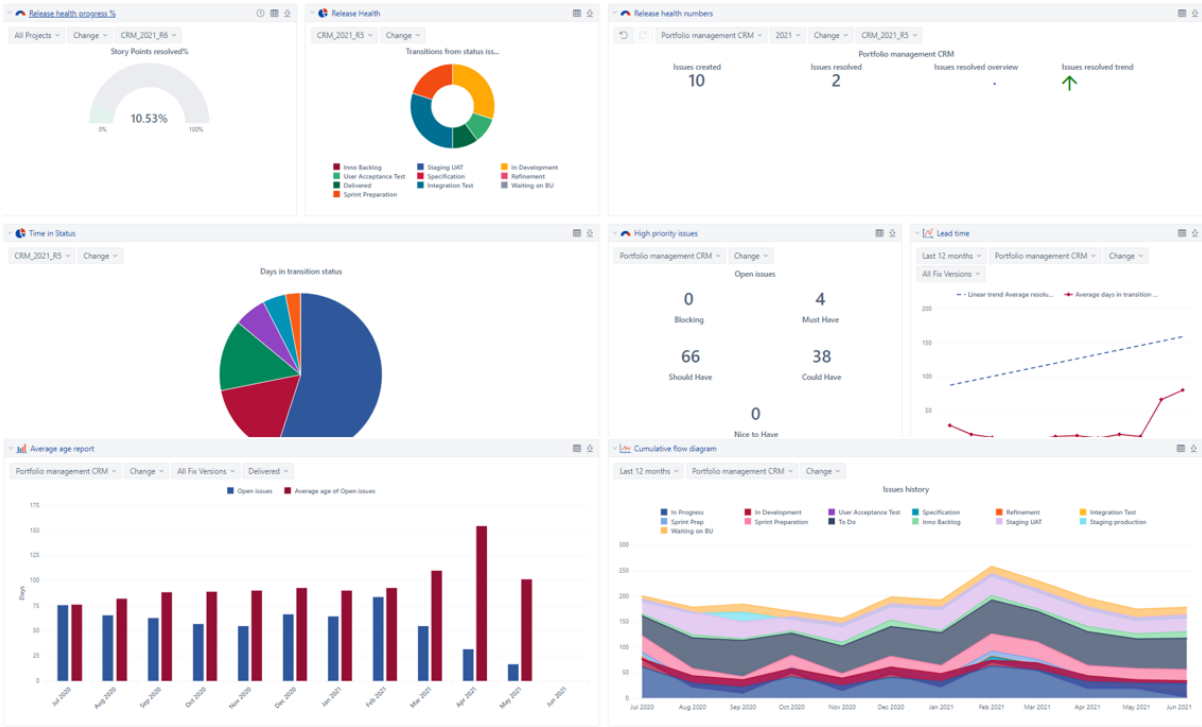


Figure 43: Demo dashboard

4.3. Evaluation

Evaluating the dashboard is beyond the scope of this project. As described above, the company has an advice now on how to analyse and visualise data, so they only need to decide on which tool and realise the dashboard based on the given recommendations. However, a great result is already achieved by creating awareness about the importance of data visualisation. Several discussions took place among the employees following from the bottleneck analysis with process mining. The bottleneck analysis showed important bottlenecks and the uncertainty of the data, so some employees already started with cleaning old data. The importance of having a suitable set with metrics became an important topic in discussions and employees were willing to share their thoughts and desires about the dashboard. Increasing the awareness of the urgency of the dashboard among employees is a first important step in implementing a dashboard.

5. Conclusion, recommendations & limitations

The goal of this research was to solve the core problem of having a limited level of data visualisation identifying known and unknown problems to improve the productivity level. By answering the research question *“how to analyse and visualise data to identify problems so that productivity of an Agile organised company can be improved”* the core problem is solved.

The research was conducted on behalf of the CRM Tribe of Group IT Europe of A.S. Watson. In this chapter, first a conclusion on the extent of which the solution reached the norms of the company is provided. It is concluded whether the solution solved the core problem. Afterwards, recommendations are provided and the limitations of the research are taken into account.

5.1. Conclusion

After a quick problem analysis, it was clear that there were multiple problems of which the impact on productivity was unclear. The underlying problem was having a limited level of data visualisation to track the impact of these problems. A lack of communication was causing errors in testing phases, tickets missing release deadlines, having difficulties with sticking to the sprint planning and there was a full backlog of the Siebel team. The core problem would be solved if the norm set by the company is reached.

An elaborate analysis explained the importance of data visualisation even more. On the one hand, data related bottlenecks, such as long duration times and long waiting times, became clear. These were indications for the importance of visualising productivity metrics. Skipping testing phases and re-entering stages are important indications for having quality metrics. On the other hand, it explained the current level of awareness about the process and its bottlenecks. Disco, the process mining tool, has the option to show an animation of the process. When playing the animation, the ticket flow is clearly visualised so that bottlenecks can be seen easily. During a Teams session, the animation was shown to the Scrum Master resulting in a discussion about the process. It became clear that the limited level of data visualisation resulted in the lack of awareness about important factors within the process.

The company wanted to have an advice on how to visualise the data to have a better overview of the data. The company already used Jira to manage their projects which provides the possibility to use metrics. After the discussion on the visualisations of the Disco animation, the conclusion was that visualisations help to understand the data.

During the research, several research methods were used to identify a suitable set of metrics for the CRM Tribe. The end result is a list with recommended metrics that can be visualised to improve the number of metrics from three to a complete dashboard. The metrics are carefully chosen based on the needs of A.S. Watson. With the use of the metrics, the problems of having a lack of communication and focusing on the own problem will be solved as well. Only a set of metrics is not enough for data visualisation. First of all, it is necessary to determine which visualisation tool to use, so therefore, a set of tools had been chosen to evaluate. Eight requirements were set up to evaluate the tools on. To determine which tool to use, the balanced scorecard method was used. After applying scores on each requirement, weights based on the importance of the requirement were added, to be able to recommend what tool to use. When the tool is determined and when the set of metrics is known, data needs to be visualised. To make optimal use of a dashboard, the right type of chart needs to be chosen. Therefore, the different types of charts were analysed and for each metric, a suggestion on which chart to use is provided. Productivity can also be improved by other improvements besides using a dashboard. For this purpose literature research is conducted to general suggestions on improving the productivity level by using add-ons for example.

To decide whether the problem has been solved, the situation before and after research is compared based on two measures. The first measure is the number of indicators available to keep track of the process. The second measure is the level of insight into the problems affecting productivity, indicating the importance of data visualisation. Both measures show that can be concluded that the core problem has been solved.

Norm Goal of the company	New reality Improved situation
<p>1) <u>Available indicators</u> Improve the amount of metrics from three single metrics to a set of well-determined metrics as input for a dashboard</p>	<p>A set with 16 recommended metrics in total of which 8 metrics are visualised as a demo, all metrics are well-determined based on bottleneck analysis, literature, and input from employees</p>
<p>2) <u>Insight into the problems</u> Instead of making decisions based on feelings about bottlenecks affecting productivity, awareness should be created about the bottlenecks to understand the importance of a dashboard</p>	<p>A clear list with bottlenecks affecting productivity, resulting in employees having conversations about the bottlenecks and the importance of data visualisation</p>

5.2. Recommendations

In this section, recommendations are provided on how to improve the productivity level by analysing and visualising data.

As identified during the bottleneck analysis, there are a lot of tickets going through stages within a short period which seems unrealistic in practice. When visualising data and making decisions based on the visualisations, it is important to ensure the data is correct. Otherwise, the conclusions might be inaccurate. Therefore, it is recommended to check the quality of the data by hiring a data analyst for example, or an audit. When there is certainty about the correctness of the data, the dashboard can be developed.

If the data is accurate, the dashboard can be developed. It is recommended to consider the privacy issues at the privacy department for the tools Screenful, Geckoboard, and EazyBI for Jira, to use in combination with Jira dashboards. Screenful scores high on ease to use and ease to install. The costs are relatively low, and the number of users is unlimited. It is integrable with Jira to provide a real-time dashboard. The only limitation is that it scores average on the degree of customisability. Geckoboard scores high on user friendliness, is real-time and has an average level of customisability. However, it scores lower on costs and number of users. Jira dashboard scores high on all requirements, except on the degree of customisability. The advantage of Jira dashboard is that it is a private cloud. To improve on the degree of customisability, Jira dashboard can be used in combination with the add-on EazyBI for Jira. The main advantage of this add-on is the possibility to customise your own charts to use in combination with Jira dashboards. However, this add-on is a public cloud. Another recommendation is Kibana. Kibana works well with the software Kafka, which provides a framework for storing, reading, and analysing streaming data. Kibana scores really good on the degree of customisability and it is on a private cloud. When there are changing needs in the future, Kibana can easily be programmed in a way it provides the desired charts.

The code provided in appendix E can be used to export data from Jira to Kafka, and from Kafka to Kibana. Since other departments are using Jira as well, they are also having issues with the export of data from Jira. Some want to make analyses and others would like to use it for different goals. The script can be adapted to achieve the desired output, so therefore it is recommended to use the script for data exportation also for different goals. A last option for a tool based on a private cloud is Tableau. The degree of customisability is really high, however it is hard to easily create the charts.

The set of metrics is provided in chapter 4 and it is recommended to use the metrics as input for the dashboards. The set of metrics require some recommended changes in the way of logging. To be able to visualise priority issues, the Tribe should start with assessing priority to tickets. To visualise the defects, the findings should be logged. It is hard to determine how good the software actually is, which is known as software quality intelligence. It is recommended to start with sending short surveys to the customer to identify the value and the quality of the delivered product. This could serve as an indication of the quality of the product so a customer love score can be determined. There are tools combining data about code changes, production uses, and test execution, to provide an insight into this. An example is SeaLights. However, this is more expensive than sending surveys. Therefore, it is recommended to implement the process of sending surveys in the process. To identify the progress of the testing phase for example, changes are required. A great suggestion from literature is the add-on “checklist”. The add-on provides the opportunity to split tasks into clear steps. In this way, it is possible to provide steps to a testing phase so the progression can be identified. This could be of importance for visualising the progression of tickets since it is not always clear what the progression is of testing for example. It is recommended to consider this add-on. In first case it is recommended to realise a CRM Tribe dashboard and Squad dashboards. After the implementation of both, it is suggested to consider a Topdesk dashboard, a Test guild dashboard and an Audit dashboard. Based on the research and the survey, there is a desire to visualise data concerning these specific topics.

Besides using the dashboard to improve productivity by increasing the level of data visualisation, another problem that needs to be solved is unreliability in the Siebel Squad at software delivering level. After observing different processes and conducting conversations with several employees, it became clear that the process was not structured at all. Every sprint again, the scheduled story points were not finished, which could be seen in the burndown charts. When planning the tickets during high-level refinement sessions, the sprints of the Siebel squad are scheduled completely full, resulting in not achieving the scheduled story points. This always results in a bad news conversation. It is recommended to start with decreasing the backlog by scheduling a percentage of the total story points in each sprint. When scheduling 70% of the points for example, 30% remains to decrease the backlog. The use of a dashboard will help to achieve this goal. Using a dashboard should be the starting point of conversations about the process and bottlenecks, so it stimulates people to improve their process. A dashboard should be voluntarily used by the team, so it needs to be seen as a tool that makes their job easier. When the process runs more smoothly, especially the process of the Siebel Squad, employees are happier with their job. Employee satisfaction is key in achieving a high productivity level.

Research suggests improving on the happiness level / satisfaction rate of employees to improve productivity. Regularly not reaching the number of agreed story points is not contributing to a happy feeling. Therefore, it is strongly recommended to improve on this to ensure a high level of satisfaction among the employees. To further increase productivity, the results of the survey can play a role. The work pressure is something which should deserve some attention. Some are satisfied, others are dissatisfied. Since well-being of the employees is priority of the Tribe Lead, it is recommended to spend some time on this topic. It is recommended to take into account their suggestions.

5.3. Limitations

This section provides several limitations that could influence the results of this research.

For the bottleneck analysis, by mining the process, only the tickets from the GIC project were used as input. As explained, the tickets can enter the process in two ways. When it is change-related, it will enter via the product owner. When there is a problem in CRM software, the tickets are entering the process via OCC. OCC either solve the problem themselves or when it is change related it will go into the change project as a GIC ticket. The focus is on the change process of the CRM Tribe since this is the main activity, so therefore it was chosen to only focus on the change tickets and not on the CRM software related tickets solved by OCC. However, those tickets immediately solved by OCC are part of the process, since the tickets can be products developed by the CRM Tribe. This could be a limitation.

A limitation of the research is the uncertainty of the correctness of the data. The uncertainty was explained during the bottleneck analysis and was discovered while doing the exportation of the data from Jira. Because of this uncertainty, it could be that there are errors in the set of recommended metrics. On the one hand, there might be problems that did not show up during the analysis, because of incorrect administration in Jira. On the other hand, it could be possible that the identified problems are no problems at all and thus metrics are wrongly recommended. To make this limitation as small as possible, literature research was conducted and conversations with employees were held.

Another limitation in the set of metrics is the situation of working from home. All employees of the company do work from home because of the pandemic. This made it hard to observe a process to determine a suitable set of metrics. In first place, it was important to get familiar with the company to have an insight into all different projects. This was only possible by conducting conversations and by joining Teams sessions. Next, the bottlenecks should be observed from a distance. This is a limitation so it could be possible that some important processes are missed. However, to limit the impact of this situation, a survey had been constructed to reach all employees.

The last limitation is that it is not possible to test the effect of a dashboard on the process. All literature suggests a dashboard with metrics as the solution, however, due to time constraints, it is not possible to evaluate this within the company.

References

- Ahmed, A., Ahmad, S., Ehsan, N., Mirza, E., & Sarwar, S. Z. (2010). Agile software development: Impact on productivity and quality. *5th IEEE International Conference on Management of Innovation and Technology, ICMIT2010*, 287–291. <https://doi.org/10.1109/ICMIT.2010.5492703>
- Atlassian. (2019). *What is DevOps? | Atlassian*. Atlassian. <https://www.atlassian.com/devops>
- Atlassian. (2020). *View and understand the cumulative flow diagram | Jira Software Cloud | Atlassian Support*. <https://support.atlassian.com/jira-software-cloud/docs/view-and-understand-the-cumulative-flow-diagram/>
- Atlassian. (2021a). *Our business*. <http://www.vireoenergy.se/our-business>
- Atlassian. (2021b). *View and understand the control chart | Jira Software Cloud | Atlassian Support*. <https://support.atlassian.com/jira-software-cloud/docs/view-and-understand-the-control-chart/>
- Atlassian. (2021c). *What is Agile?* <https://www.atlassian.com/agile>
- Beck, K., Beedle, M., Bennekum, A. Van, Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., & Thomas, D. (2001). *Manifesto for Agile Software Development*. The Agile Alliance. <http://agilemanifesto.org/>
- Berteig, M. (2020). *Seven Options for Handling Interruptions in Scrum and Other Agile Methods - Berteig Consulting and Training*. <https://berteig.com/how-to-apply-agile/seven-options-for-handling-interruptions-in-scrum-and-other-agile-methods-3/>
- Biesialska, K., Franch, X., & Muntés-Mulero, V. (2021). Big Data analytics in Agile software development: A systematic mapping study. *Information and Software Technology*, 132, 106448. <https://doi.org/10.1016/j.infsof.2020.106448>
- Budacu, E., & Pocatilu, P. (2018). Real Time Agile Metrics for Measuring Team Performance. *Informatica Economica*, 22(4/2018), 70–79. <https://doi.org/10.12948/issn14531305/22.4.2018.06>
- Chen, I. J., & Popovich, K. (2003). Understanding customer relationship management (CRM): People, process and technology. *Business Process Management Journal*, 9(5), 672–688. <https://doi.org/10.1108/14637150310496758>
- Dmitriev, S. (2018, November 21). *Reading and visualizing data from Jira by python | by Sergei Dmitriev | Towards Data Science*. <https://towardsdatascience.com/communication-story-from-an-issue-tracking-software-efbbf29736ff>
- Drumond, C. (2018). *Scrum - what it is, how it works, and why it's awesome*. Atlassian. <https://www.atlassian.com/agile/scrum>
- Fatema, I., & Sakib, K. (2018). Factors Influencing Productivity of Agile Software Development Teamwork: A Qualitative System Dynamics Approach. *Proceedings - Asia-Pacific Software Engineering Conference, APSEC, 2017-Decem*, 737–742. <https://doi.org/10.1109/APSEC.2017.95>
- Heerkens, H., & van Winden, A. (2021). Solving Managerial Problems Systematically. In *Solving Managerial Problems Systematically*. Noordhoff uitgevers. <https://doi.org/10.4324/9781003186038>
- Heidenberg, J., Weijola, M., Mikkonen, K., & Porres, I. (2013). A metrics model to measure the impact of an agile transformation in large software development organizations. *Lecture Notes in*

- Business Information Processing*, 149, 165–179. https://doi.org/10.1007/978-3-642-38314-4_12
- Ismail, K. (2018). *Agile vs DevOps: What's the Difference?* <https://www.cmswire.com/information-management/agile-vs-devops-whats-the-difference/>
- Jiang, J., & Klein, G. (2000). Software development risks to project effectiveness. In *Journal of Systems and Software* (Vol. 52, Issue 1). [https://doi.org/10.1016/S0164-1212\(99\)00128-4](https://doi.org/10.1016/S0164-1212(99)00128-4)
- Kitchenham, B., & Mendes, E. (2004). Software productivity measurement using multiple size measures. *IEEE Transactions on Software Engineering*, 30(12), 1023–1035. <https://doi.org/10.1109/TSE.2004.104>
- Kola, B. (2014). *Thinking Lean in Agile Software Development Projects*.
- Kupiainen, E., Mäntylä, M. V., & Itkonen, J. (2015). *Using metrics in Agile and Lean Software Development-A systematic literature review of industrial studies*. <https://doi.org/10.1016/j.infsof.2015.02.005>
- Kurnia, R., Ferdiana, R., & Wibirama, S. (2018). *Software metrics classification for agile scrum process: A literature review*. 2018 International Seminar on Research of Information Technology and Intelligent Systems, ISRITI 2018. <https://doi.org/10.1109/ISRITI.2018.8864244>
- Lehtonen, T., Eloranta, V. P., Leppänen, M., & Isohanni, E. (2013). Visualizations as a basis for agile software process improvement. *Proceedings - Asia-Pacific Software Engineering Conference, APSEC, 1*, 495–502. <https://doi.org/10.1109/APSEC.2013.71>
- Mark Cruth. (2021). *The Spotify model | Atlassian | Atlassian*. <https://www.atlassian.com/agile/agile-at-scale/spotify>
- Marques, R., Da Silva, M. M., & Ferreira, D. R. (2018). Assessing agile software development processes with process mining: A case study. In *Proceeding - 2018 20th IEEE International Conference on Business Informatics, CBI 2018* (Vol. 1). <https://doi.org/10.1109/CBI.2018.00021>
- Matthies, C., Dobrigkeit, F., & Hesse, G. (2020). Mining for Process Improvements: Analyzing Software Repositories in Agile Retrospectives. *Proceedings - 2020 IEEE/ACM 42nd International Conference on Software Engineering Workshops, ICSEW 2020*, 189–190. <https://doi.org/10.1145/3387940.3392168>
- Meyer, A. N., Murphy, G. C., Fritz, T., & Zimmermann, T. (2019). Rethinking Productivity in Software Engineering. In C. Sadowski & T. Zimmermann (Eds.), *Rethinking Productivity in Software Engineering* (pp. 137–146). <https://doi.org/10.1007/978-1-4842-4221-6>
- Myers, B. A., Ko, A. J., LaToza, T. D., & Yoon, Y. (2019). Human-Centered Methods to Boost Productivity. In Caitlin Sadowski & T. Zimmermann (Eds.), *Rethinking Productivity in Software Engineering* (pp. 147–157). https://doi.org/10.1007/978-1-4842-4221-6_13
- Olszewska, M., Heidenberg, J., Weijola, M., Mikkonen, K., & Porres, I. (2016). Quantitatively measuring a large-scale agile transformation. *Journal of Systems and Software*, 117, 258–273. <https://doi.org/10.1016/j.jss.2016.03.029>
- Padmini, K. V. J., Dilum Bandara, H. M. N., & Perera, I. (2015). *Use of software metrics in agile software development process*. MERCon 2015 - Moratuwa Engineering Research Conference. <https://doi.org/10.1109/MERCon.2015.7112365>
- Parker, D. W., Holesgrove, M., & Pathak, R. (2015). *Improving productivity with self-organised teams and agile leadership*. International Journal of Productivity and Performance Management. <https://doi.org/10.1108/IJPPM-10-2013-0178>

- Ramirez-Mora, S. L., & Oktaba, H. (2018). Team maturity in agile software development: The impact on productivity. *Proceedings - 2018 IEEE International Conference on Software Maintenance and Evolution, ICSME 2018*, 732–736. <https://doi.org/10.1109/ICSME.2018.00091>
- Rehkopf, M. (n.d.). *Epics Atlassian*. Retrieved May 7, 2021, from <https://www.atlassian.com/agile/project-management/epics>
- Sadowski, Caitlin, & Zimmermann, T. (2019). Rethinking Productivity in Software Engineering. In *Rethinking Productivity in Software Engineering*. <https://doi.org/10.1007/978-1-4842-4221-6>
- Schwaber, K., & Scrum.org. (2021). Online Nexus Guide: The Definitive Guide to Scaling Scrum with Nexus. In *Scrum.org*. <https://www.scrum.org/resources/online-nexus-guide>
- Scott, E., Charkie, K. N., & Pfahl, D. (2020). Productivity, Turnover, and Team Stability of Agile Teams in Open-Source Software Projects. *Proceedings - 46th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2020*, 124–131. <https://doi.org/10.1109/SEAA51224.2020.00029>
- Sedano, T., Ralph, P., & Péraire, C. (2019). Removing Software Development Waste to Improve Productivity. In Caitlin Sadowski & T. Zimmermann (Eds.), *Rethinking Productivity in Software Engineering* (pp. 221–240). https://doi.org/10.1007/978-1-4842-4221-6_19
- Sedrakyán, G., Mannens, E., & Verbert, K. (2019). Guiding the choice of learning dashboard visualizations: Linking dashboard design and data visualization concepts. *Journal of Visual Languages and Computing*, 50, 19–38. <https://doi.org/10.1016/j.jvlc.2018.11.002>
- Shah, S. M. A., Papatheocharous, E., & Nyfjord, J. (2015). Measuring productivity in agile software development process: A scoping study. *ACM International Conference Proceeding Series*, 24-26-Aug, 102–106. <https://doi.org/10.1145/2785592.2785618>
- Storey, M.-A., & Treude, C. (2019). Software Engineering Dashboards: Types, Risks, and Future. In Caitlin Sadowski & T. Zimmermann (Eds.), *Rethinking Productivity in Software Engineering* (pp. 179–190). https://doi.org/10.1007/978-1-4842-4221-6_16
- Treude, C., & Filho, F. F. (2019). How Team Awareness Influences Perceptions of Developer Productivity. In Caitlin Sadowski & T. Zimmermann (Eds.), *Rethinking Productivity in Software Engineering* (pp. 169–178). https://doi.org/10.1007/978-1-4842-4221-6_15
- Van Eck, M. L., Lu, X., Leemans, S. J. J., & Van Der Aalst, W. M. P. (n.d.). *PM 2 : a Process Mining Project Methodology*.
- Vogel, J., & Telesko, R. (2020). Derivation of an agile method construction set to optimize the software development process. *Journal of Cases on Information Technology*, 22(3), 19–34. <https://doi.org/10.4018/JCIT.2020070102>
- Wagner, S., & Murphy-Hill, E. (2019). Rethinking Productivity in Software Engineering. In C. Sadowski & T. Zimmermann (Eds.), *Rethinking Productivity in Software Engineering* (pp. 69–84). <https://doi.org/10.1007/978-1-4842-4221-6>
- Watson, A. S. (2020). *Our Company*. <https://www.aswatson.com/our-company/o-and-o-strategy/#.Ylu9MegzZPY>
- Wiesche, M. (2021). Interruptions in Agile Software Development Teams. In *Project Management Journal* (Vol. 52, Issue 2). <https://doi.org/10.1177/8756972821991365>
- Zieris, F., & Prechelt, L. (2019). Does Pair Programming Pay Off? In Caitlin Sadowski & T. Zimmermann (Eds.), *Rethinking Productivity in Software Engineering* (pp. 251–259). https://doi.org/10.1007/978-1-4842-4221-6_21

Appendix

Appendix A Agile Manifesto

The twelve principles of the Agile Manifesto (Beck et al., 2001)

- 1) Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- 2) Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- 3) Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- 4) Business people and developers must work together daily throughout the project.
- 5) Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- 6) The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- 7) Working software is the primary measure of progress.
- 8) Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- 9) Continuous attention to technical excellence and good design enhances agility.
- 10) Simplicity--the art of maximizing the amount of work not done--is essential.
- 11) The best architectures, requirements, and designs emerge from self-organizing teams.
- 12) At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Appendix B Data analysis

Jira report cumulative flow diagram

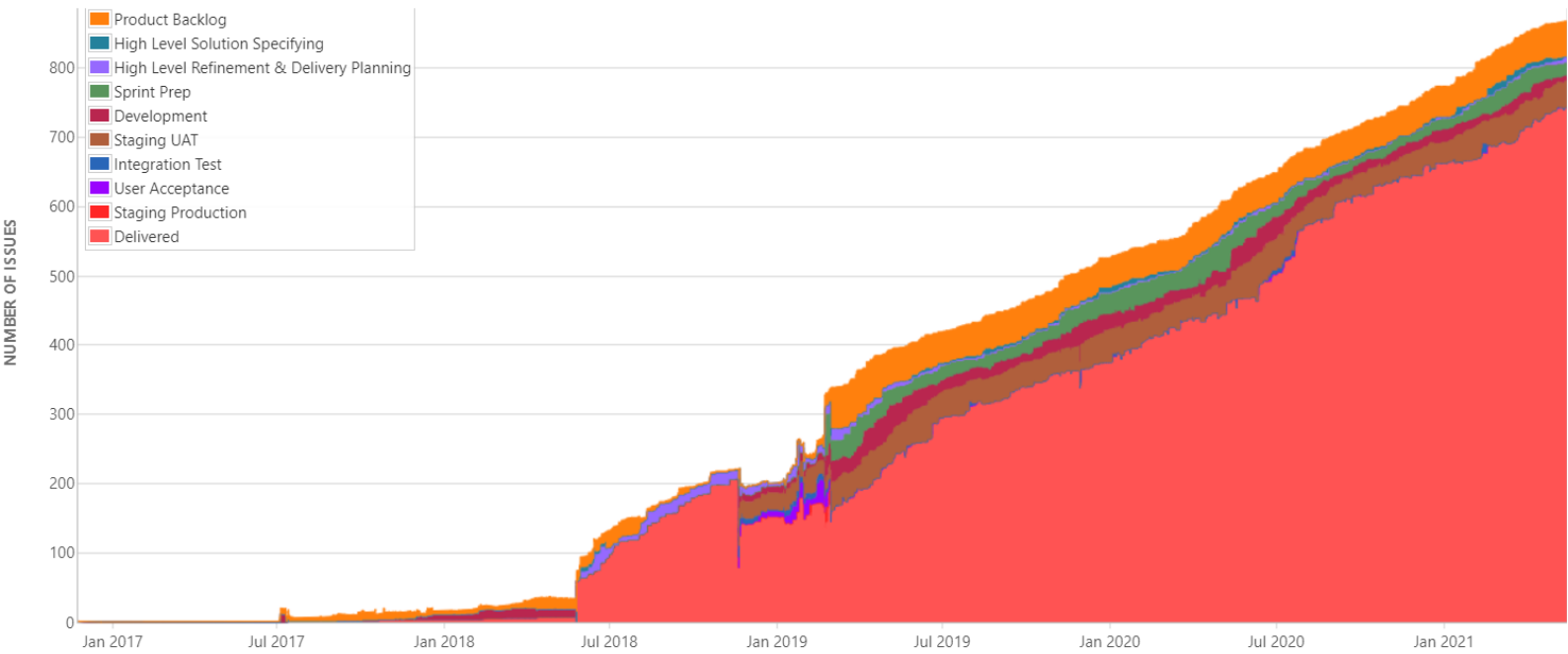


Figure 44: Cumulative flow diagram

Jira report control chart

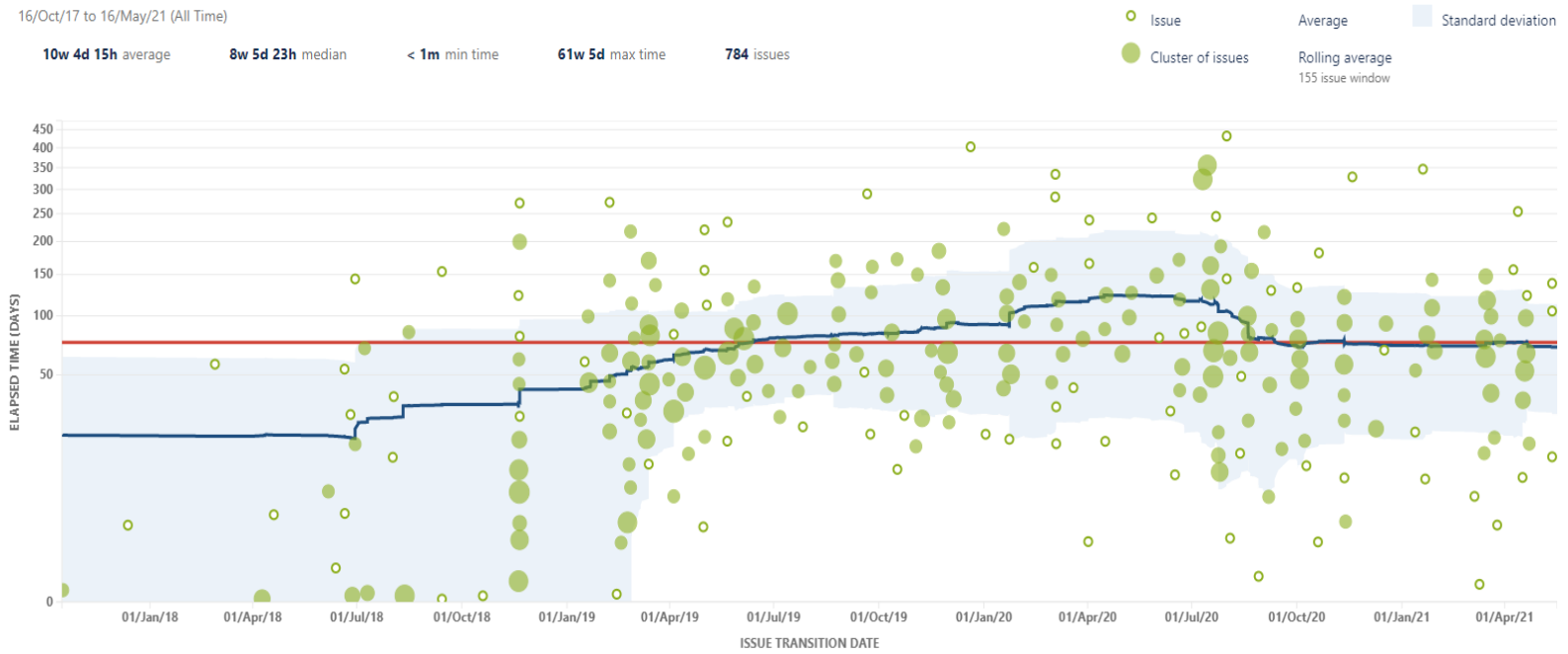


Figure 45: Control chart

Jira report average age

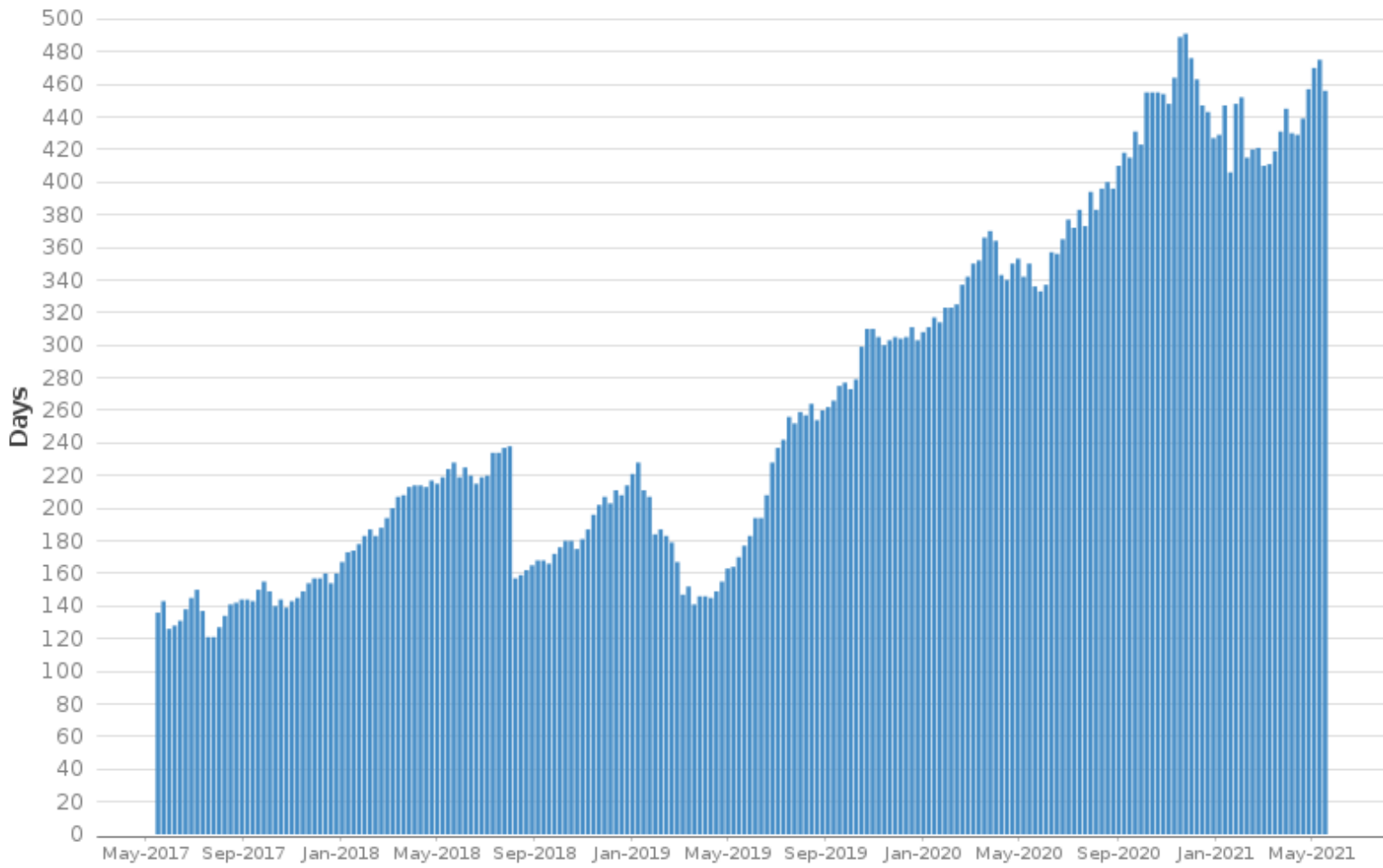


Figure 46: Average age report

Jira report resolution time

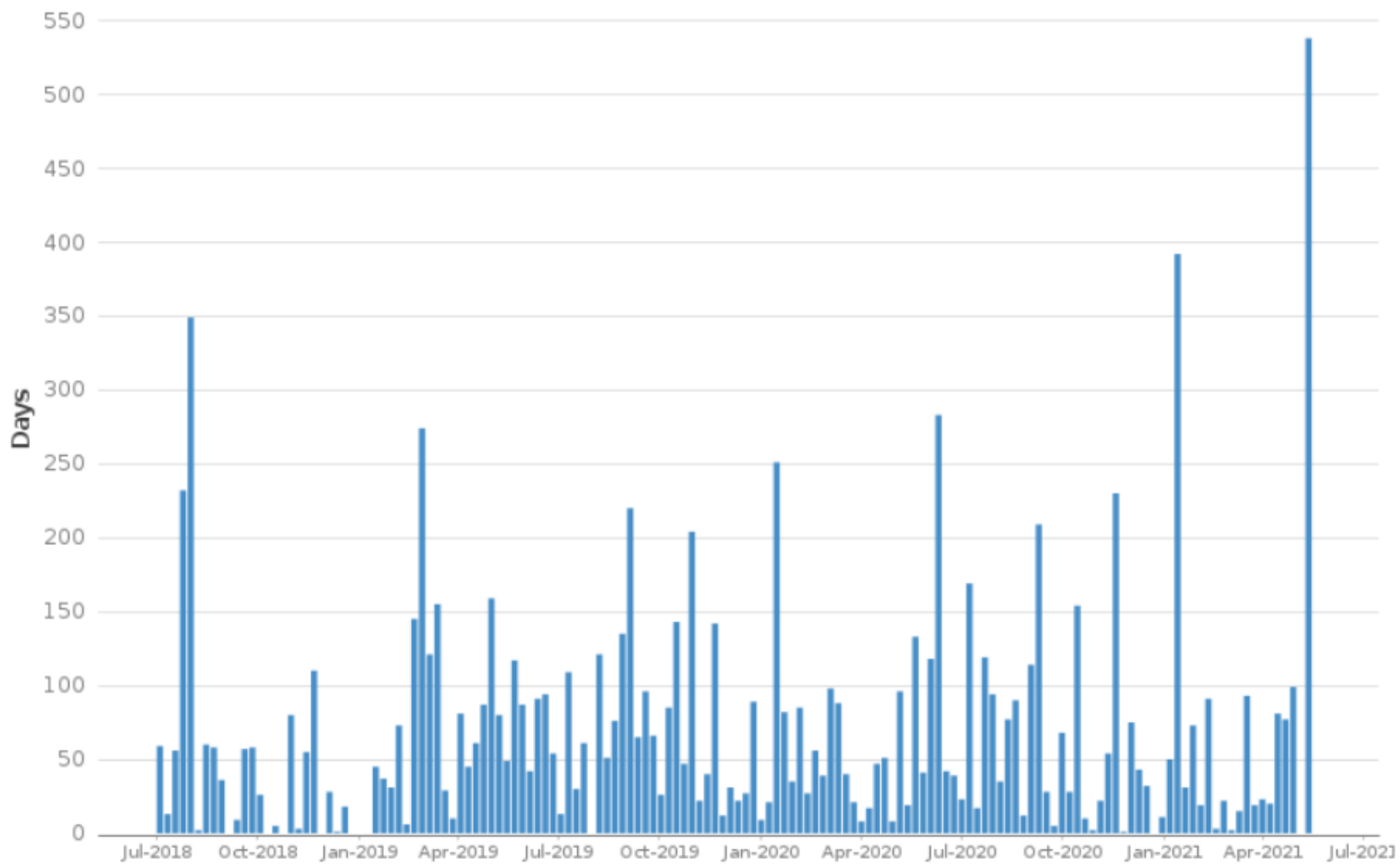


Figure 47: Resolution time

Script

```
1 //General Libs
2 import org.apache.log4j.Category
3 import org.apache.log4j.Level
4 import org.apache.log4j.Logger
5
6 //This logger must be turned on in Scriptrunner. When running in the Groovy Script Fester, this must be disabled.
7 def clog = Logger.getLogger( 'Transition CHANGE (extensive)' );
8 clog.setLevel(Level.DEBUG);
9 //clog.setLevel(Level.WARN);
10 clog.debug( '---- Start Script ----' );
11
12 // Define a JQL query to search for the issues on which you want to set the impediment flag
13 def query = 'project = GIC and issuetype = Change and status = Delivered and "TransactionHistory[Paragraph]" is EMPTY'
14
15 // Search for the issues we want to update
16 def searchReq = get("/rest/api/2/search")
17     .queryString("jql", query)
18     .queryString("fields", "Flagged")
19     .asObject(Map)
20
21 // Verify the search completed successfully
22 assert searchReq.status == 200
23
24 // Save the search results as a Map
25 Map searchResult = searchReq.body
26
27 // Iterate through the search results and set the Impediment flag for each issue returned
28 searchResult.issues.each { Map issueRes ->
29
30     def issueKey = issueRes.key;
31     def issueR = get("/rest/api/2/issue/${issueKey}?expand=changelog.histories")
32         .asObject(Map)
33         issueB = issueR.body as Map
34     def returnString = null;
35
36     for( history in issueB.changelog.histories){
37         //return 'Author='+history.author.displayName+' Field='+history.items.field+' From='+history.items.fromString+' To='+history.items.toString;
38         //clog.debug('Author='+history.author.displayName+' Field='+history.items.field+' From='+history.items.fromString+' To='+history.items.toString);
39         if( history.items.field.contains('status')) {
40             if( !returnString){
41                 returnString = '# From: '+history.items.fromString+' to: '+history.items.toString+' @ '+history.created;
42             } else{
43                 returnString += '# From: '+history.items.fromString+' to: '+history.items.toString+' @ '+history.created;
44             }
45             //clog.debug('# From: '+history.items.fromString+' to: '+history.items.toString+' @ '+history.created);
46         }
47     }
48
49     def result = put("/rest/api/2/issue/${issueKey}")
50         .queryString("overrideScreenSecurity", Boolean.TRUE)
51         .header('Content-Type', 'application/json')
52         .body([
53             fields:[
54                 customfield_12758: returnString
55             ]
56         ]).asString()
57 }
58 }
```

Figure 48: Jira Script for data extraction

Disco analysis

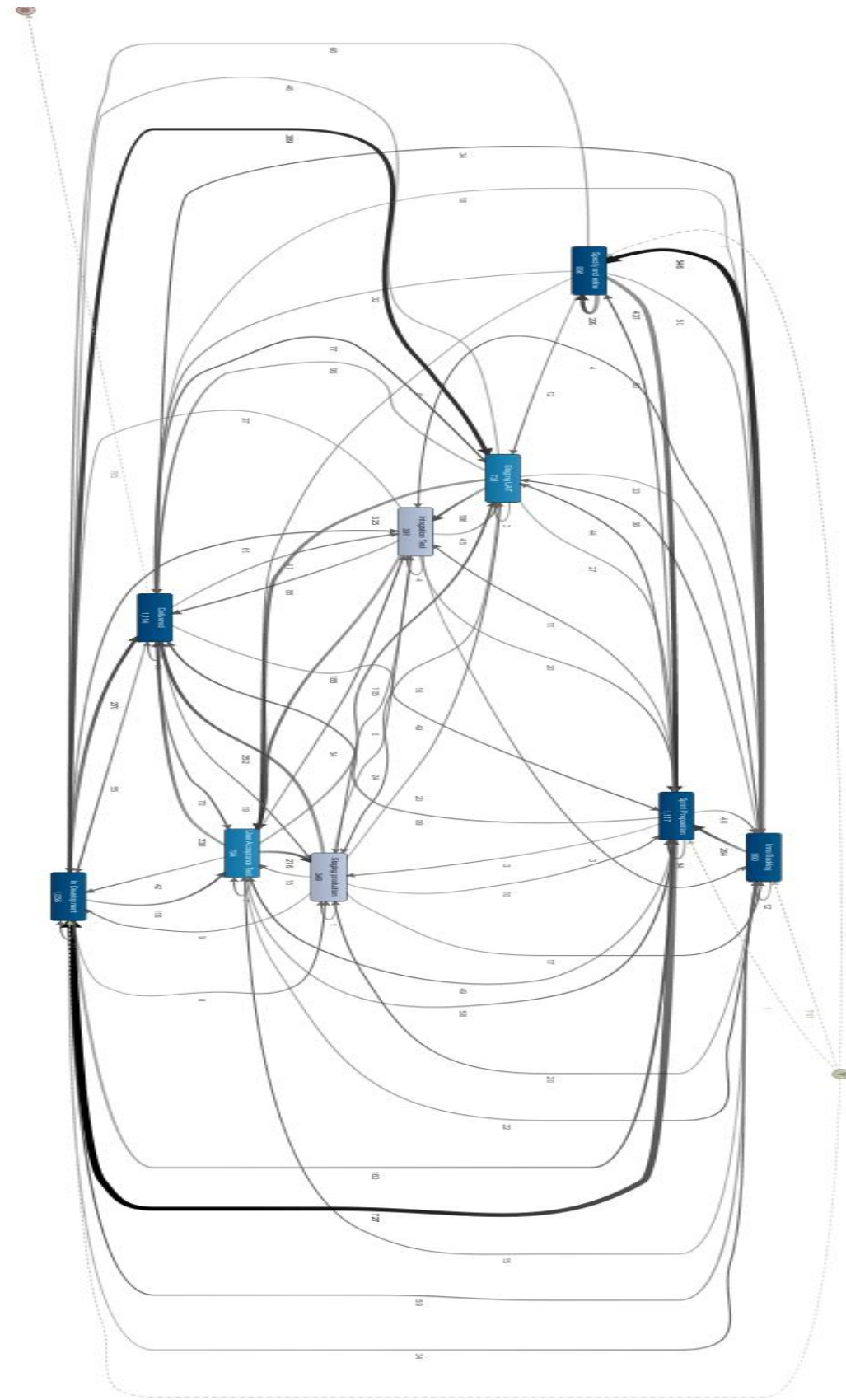


Figure 49: Control-flow perspective generated by Disco (100% activities, 100% path)

Appendix C Survey



Q1.

You are being invited to participate in a research study titled “Analysing and visualising data to improve the productivity level of an Agile organised company”. This study is being done by Ceret van der Vegt from the Faculty of Behavioural, Management and Social Sciences at the University of Twente.

The purpose of this research study is to identify which indicators (further called as metrics) are seen as important by you to add to the dashboard for the CRM Tribe, and will take you approximately 5 minutes to complete. The data will be used for deciding which metrics will be added to the final dashboard.

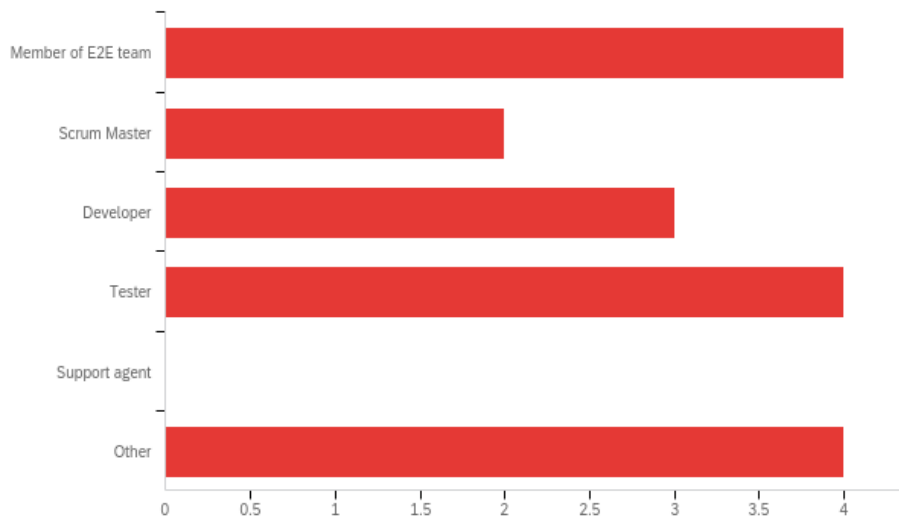
Your participation in this study is entirely voluntary.

There are no known risks associated with this research study. To the best of the ability your answers in this study will remain confidential. The study is anonymised to prevent any risks with personal data. This research has been approved by the Ethical Committee of the University of Twente.

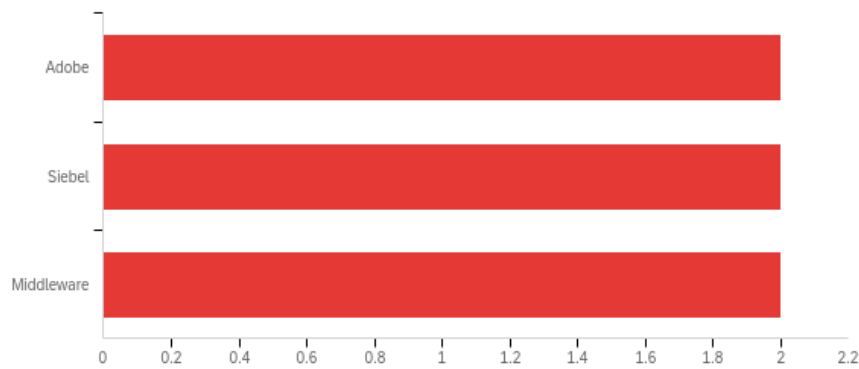
For further questions, do not hesitate to send me an email via c.r.vandervegt@student.utwente.nl

Your help is appreciated. Thank you.

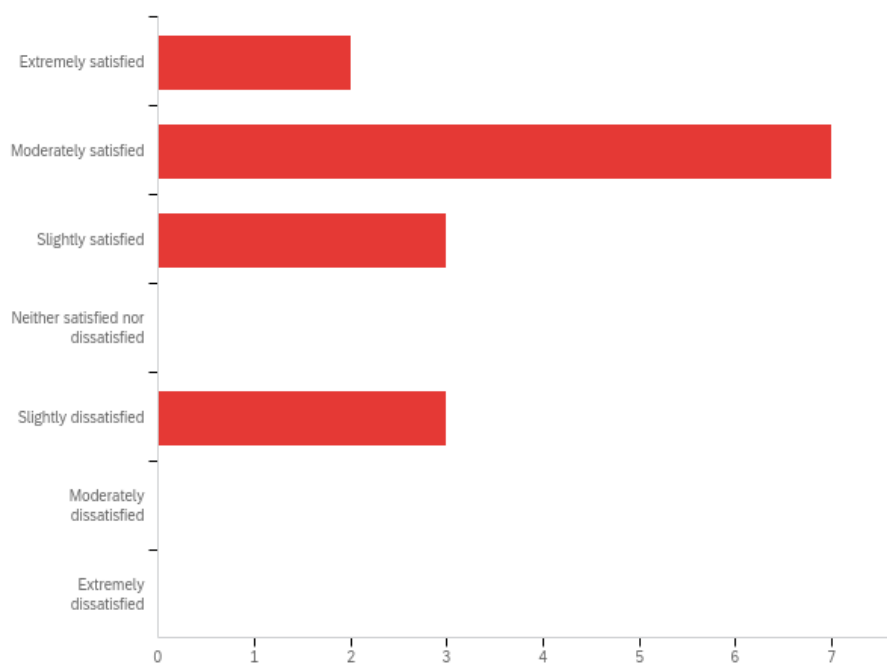
Q2 - What is your function?



Q3 - From which Squad are you?



Q4 - How satisfied are you about the work pressure?



Q5 - What process should be improved to make your job easier?

Ticket routing to second line support. Clear guidelines need to be followed by the person raising Topdesk tickets from Productions Support to DEV

E2E proces and focus of tribe to improve E2E on a regular basis

The period before and during integration testing

There should be a better planning across domains and bus on projects that are started and running. This to prevent an overload of change requests at the same time.

Planning for Siebel and ETL

Developers should do support on their developments to get feedback & learn. Look further than the length of your nose ;-)

In simple terms I do not wish to shuffle between development, analysis, helping testers, solving issue in test and production environments and giving status to different people and chase people to get dependencies resolved. I want to have clear set of things to be done with clear priorities to do my work easier. Now some management language ;), There is also need to improve the way we measure productivity A. Not to confuse between OKRs and KPIs and setup correct KPIs. B. Productivity is measured with same unit for all type of work which creates lack of transparency.

The demand of CR's and project coming to us more spread over the year instead of everything at the same moment in time. I know: wishful thinking ;)

streamline jira statuses to a clear definition to have common understanding what a status means.

Deliver according to agreed release roadmap.

Live tracking of work to be done

Expanding the team.

Less questions about working on other stuff. (But also a point of critique on myself is that I am not able to say no)

1. All instances are aligned so it will reduce time in comparing all 9 instances before starting development. 2. Data availability to test development.

Q6 - Which metric(s) would make your job easier / better?

Sprint completion metrics inclusive of Topdesk activities.

Release burndown related metrics E2E improvements

Efficiency, Quality, predicatability

Findings/Bugs

Number of findings related to the number of changes Number of change requests scheduled for a certain release

Lead time of stories. Knowing when my team is the bottleneck. The status at which a story is the longest. Test finding counts on GIC's

I prefer quality over quantity however even if a quality gauge could be set-up it wouldn't help

That needs to be decided based upon, as a tribe and as a squad, how we wish to measure our KPIs. What bottlenecks, quality gaps and efficiencies we would like to monitor. May be on high level 1. The way quality assurance is achieved. 2. The way squad supports production and test env issues. 3. Development velocity of the squad. 4. Time wasted on manual routine activities 5. Dependencies

No of findings for each squad in integration testing

Customer Love Score: I think it is most important to know how our (internal) customer is scoring is. This because I believe that will lead to higher quality, on all levels.

topdesk overview of statuses, closed ,etc

Lead times, cycle times, know if a change was moves from one release to the next

live status of the work to be done / in progress

A metric which takes into account time, cost and quality.

All metrics should be fine because you can create test data based on most important metrics

Not Sure, Every time needs to work with different data or tables.

Q7 - Do you have any further suggestions for the dashboard?

No

One dashboard for whole CRM Tribe with some general topics and some highlights per squad. Touchpoints to use it within our Agile / Scrum rhythm so it becomes natural to use it. Not too much detail on it.

Would be nice if it is as realtime as possible so that it triggers people and motivates to improve on those area's. Also the question is how do you implement it so it is actively used.

No

Showing status of a functional release with dials / traffic lights per squad / per change request

I don't have a lot of time to figure out how it works nor create my own, so a app that is intuitive and has out-of-the-box reports (or reports are easily share-able) would be great.

Good luck

no

The more you can visualize data, the more it will bring

For my team I do not miss any dashboard item, however from tribe perspective I think it makes sense to have more overview over teams.

Easy, quick, intuitive visuals

Colorful?

Appendix D Systematic Literature Review

SLR – 1

The research question to be answered with this SLR is “What optimisations on Agile software development teams to improve productivity are suggested in research?” The aim of this SLR is to create a recommendation with suitable optimisations on how to improve productivity of Agile organised software development teams suggested by research. In this appendix, step 2 to step 6 from the seven-step approach are included.

Step 2 - Defining inclusion and exclusion criteria

Table 7: Inclusion and Exclusion criteria

Inclusion criteria	Reason
Research on Scrum	The main principles of Scrum are used in the A.S. Watson model, therefore optimisations on the framework Scrum instead of on Agile itself are included.
Improvements / changes / development	Synonyms for optimisations are included.
Optimisations process in general of Agile company	Research on the process in general is included since productivity is one part of the general process.
Exclusion criteria	Reason
Research that is not peer-reviewed	An important aspect of the validity of research is that it is peer-reviewed. Therefore, non-peer-reviewed literature is excluded since valid research is important.
Research that is not cited	When research has been cited by others, it is an indication of quality. Since quality is important, non-cited papers will be excluded.
Papers in a different language than English or Dutch	Translating research to a language I understand is hard and sensitive for errors. Only qualitative research is included so therefore research in another language will be excluded.
Research not suggesting optimisations	The aim is to identify which optimisations are relevant. If the article is not addressing this, the article is irrelevant to the research.
Inaccessible research	If an article is not accessible it will be excluded.

Step 3 - Defining the databases

The databases used for the SLR are Business Source Elite, Google Scholar, IEEE, and Scopus. All databases provide both cited research and peer-reviewed research. The first database, business source elite, is chosen because research takes place within business. Google Scholar contains many literature, also research which is hard to find while using the other databases. IEEE is chosen because it is a technical database. Research takes place in a technical environment, which is the reason for choosing IEEE. Scopus is a reliable database providing high quality literature.

Step 4 - Search terms and strategy

The research question to be answered with this SLR consist of four main parts. Therefore, the search term is separated in four concepts. For each concept, synonyms, broader terms, and narrower terms are listed in **Table 8**. The applied strategy is to insert the different terms and find the most specific results. The concepts/terms with the most specific results are used.

Table 8: Search terms

Key concepts	Related terms / synonyms	Broader terms	Narrower terms
Agile	Agile department	Agile company	Scrum, Spotify model, Nexus model
Optimisations	Improvement, development, enhance, increase	Changes, suggestions, influences	Better results
Productivity	Productiveness	Performance, Process	Capacity, yield, throughput
Software development team	Software engineering, CRM software	IT	Software

Step 5 - Results

Table 9: Initial search results

Date	Database	Search string	Number of hits
20-05-2021	Business Source Elite	(Agile) AND (Process) AND (optimi* OR improv*) AND ("software develop* team*")	11
04-05-2021	Google Scholar	allintitle: (productivity) AND (software) AND Agile	24
04-05-2021	Google Scholar	allintitle: (productivity) AND (software) AND (scrum)	2
04-05-2021	IEEE	(Improve* OR optimi*) AND (productivity) AND (Agile)	118
04-05-2021	IEEE	(factor*) AND (influence*) AND (productivity) AND ("software development* team*")	5
04-05-2021	Scopus	TITLE (Agile) AND (process) AND (improve* OR optimi*)	72
04-05-2021	Scopus	TITLE ("Agile improv*) AND (productivity)	7
Initial number of results			239
Removing after applying inclusion and exclusion criteria			-224
Removing duplicates			-5
Included from references			+3
Final selection			13

Step 6 - Conceptual matrix

In **Table 10**, the final set consisting of 13 articles are defined. All articles provide factors affecting productivity and suggestions on optimisations on the productivity level of an Agile organised company.

Table 10: Results

Journal	Article title	Authors	Research topic	Application to own research
<i>Information and Software Technology</i>	Big Data analytics in agile software development: A systematic mapping study	K. Biesialska, X. Franch, V. Muntés-Mulero	Providing an overview of Big Data analytics in combination with Agile Software development	Suggestions on how to use Big Data
<i>Project Management Journal</i>	Interruptions in Agile Software Development Teams	M. Wiesche	Providing an overview of the type of interruptions in Agile Software Development Teams	Identifying which interruptions there are to take into account and to provide suggestions on how to deal with it
<i>Report</i>	Thinking Lean in Agile Software Development Projects	B. Kola	A qualitative study aiming at providing ways on how to improve productivity	Suggestions based on survey results on how to improve productivity
<i>Proceedings – 46th Euromicro conference on Software Engineering and Advance Applications (SEAA)</i>	Productivity, Turnover, and Team Stability of Agile Teams in Open-Source Software Projects	E. Scott K. N. Charkie D. Pfahl	Researching productivity in open-source projects in agile software development teams	Factors impacting productivity
<i>5th IEEE International conference on management of innovation and Technology</i>	Agile Software Development: Impact on Productivity and Quality	A. Ahmed S. Ahmad Dr. N. Ehsan E. Mirza S.Z. Sarwar	Analysis of Agile methodologies in practice	Impact on productivity
<i>2018 IEEE International conference on software maintenance and evolution</i>	Team maturity in Agile Software Development: The impact on productivity	S.L. Ramirez-Mora H. Oktaba	Identifying factors affecting productivity in Agile Software Development	Identifying factors impacting productivity
<i>2017 24th Asia-pacific Software Engineering Conference</i>	Factors Influencing Productivity of Agile Software Development Teamwork: A	I. Fatema K. Sakib	Identifying the factors influencing productivity of an Agile	Identifying the factors influencing productivity

	Qualitative System Dynamics Approach		Software Development Team	
<i>2020 IEEE/ACM 42nd international conference on software engineering workshops (ICSEW)</i>	Mining for Process Improvements: Analyzing Software Repositories in Agile Retrospectives	C. Matthies F. Dobrigkeit G. Hesse	Proposing new Retrospective activities based on mining the software repositories	Suggesting the use of mining in retrospectives
<i>2013 20th Asia-Pacific Software engineering conference</i>	Visualizations as a Basis for Agile Software Process Improvements	T. Lehtonen V. Eloranta M. Leppänen E. Isohanni	Identifying how to improve software processes	Suggesting to use visualisations based on this research
<i>International Journal of Productivity and Performance Management</i>	Improving productivity with self-organised teams and agile leadership	D.W. Parker M. Holesgrove R. Pathak	Describing the development around leadership of self-organised teams	Describing the importance of leadership in self-organised teams
<i>Book</i>	Rethinking Productivity in Software Engineering	C. Sadowski T. Zimmermann	Exploring what productivity means for modern software development	Suggesting several options to improve productivity
<i>IEEE Transactions on Software Engineering</i>	Software productivity measurement using multiple size measures	B. Kitchenham, E. Mendes	Identifying how to measure productivity	Describing how to measure productivity
<i>Journal of Systems and Software</i>	Software development risks to project management	J. Jiang G. Klein	Indicating common aspects of project effectiveness	Describing why insight in productivity is important

SLR – 2

The research question to be answered with this SLR is “What indicators are suggested for Agile software development companies?” The aim of this SLR is to create a framework which explains suitable indicators for software developing suggested by research. In this appendix, step 2 to step 6 from the seven-step approach are included.

Step 2 - Defining inclusion and exclusion criteria

Table 11: Inclusion and Exclusion criteria

Inclusion criteria	Reason
Research on indicators specifically in <i>software developing companies</i>	For all different types of companies, different indicators are important. Since the research is on a software developing department, this is an inclusion criteria. Otherwise, the suggested indicators cannot be applied.
The term indicator as well as metrics and KPI can be used	The term “metrics” is a common term used for indicators in Agile organised companies. In other companies, KPIs are used. The company is Agile organised, so the term is allowed to use.
Performance / data visualisation	The research is on suggested indicators with the goal to visualise data. Therefore, research on performance and data visualisation are included.
Exclusion criteria	Reason
Research that is not peer-reviewed	An important aspect of the validity of research is that it is peer-reviewed. Therefore, non-peer-reviewed literature is excluded since valid research is important.
Research that is not cited	When research has been cited by others, it is an indication of quality. Since quality is important, non-cited papers will be excluded.
Papers in a different language than English or Dutch	Translating research to a language I understand is hard and sensitive for errors. Only qualitative research is included so therefore research in another language will be excluded.
Research related to non-software companies	Performance is measured in a different way in other types of companies than software related. Results from research on other types of companies cannot be used in my research.
Research not suggesting metrics / indicators / KPIs	The aim is to identify which metrics / indicators / kpis are relevant. If the article is not addressing this, the article is irrelevant to the research.
Research only applicable to SMEs	SMEs are small and medium-sized enterprises. Since the research will be conducted at a very large company all over the world with a lot of employees, it should be applicable to a large company. If only applicable to a SME, the research cannot be applied.
Should be accessible	If an article is not accessible it will be excluded.

Step 3 - Defining the databases

The three databases used for the SLR are Business Source Elite, Scopus, and Web of Science. All databases provide both cited research and peer-reviewed research. The first database, business source elite, is chosen because research takes place within business. After searching on Google for “database for research in business”, this was the first suggestion. Both Scopus and Web of Science are reliable databases providing high quality literature. Therefore, they are also chosen as databases to use.

Step 4 - Search terms and strategy

The research question to be answered with this SLR consist of three main parts. Therefore, the search term is separated in three concepts. For each concept, synonyms, broader terms, and narrower terms are listed in **Table 12**. The applied strategy is to insert the different terms and find the most specific results. The concepts/terms with the most specific results are used.

Table 12: Search terms

Key concepts	Related terms / synonyms	Broader terms	Narrower terms
Indicator	Metric, key performance indicator, lead indicator	Performance indicator, data visualisation	KPI Productivity
Software development	Software engineering, CRM software	IT	Agile
Company	Organisation, enterprise, multinational	Business	Software development team

Step 5 - Results

When using the term “metric*” only, one specific database yields over 50.000 results. Therefore, only three different databases are used to limit the search results. To further narrow down the results, the scope of the search is limited to only title or title-abs-key. From the search terms included in table 12, only the most relevant results are used in the end. Otherwise, there were too many unusable results. The results are visualised in **Table 13**.

Table 13: Initial results

Date	Database	Search string	Number of hits
02-04-2021	Business source elite	(indicator OR metric*) AND (“software development*”) AND (Agile)	66
02-04-2021	Business source elite	(indicator OR metric*) AND (Agile) AND (productivity)	19
01-04-2021	Scopus	TITLE (KPI* OR metric*) AND (“agile”) AND (software)	30
01-04-2021	Scopus	TITLE-ABS-KEY (kpi*) AND (“software development*”) AND (Agile)	5
01-04-2021	Web of Science	("Software Metric*") AND ("Agile Software Development")	10
Initial number of results			130
Removing after applying inclusion and exclusion criteria			-120
Removing duplicates			-4
Final selection			6

Step 6 - Conceptual matrix

In **Table 14**, the final set consisting of 6 articles are defined. All articles provide suggestions on which metrics to use.

Table 14: Final results

Journal	Article title	Authors	Research topic	Application to own research
<i>Journal of systems and software</i>	Quantitatively measuring a large-scale agile transformation	Olszewska, M. Heidenberg, J. Weijola, M. Mikkonen, K. Porres, I.	Measuring changes of an agile transformation by metrics	9 metrics are suggested, those can be used
<i>Journal of Information and Software Technology</i>	Using metrics in Agile and Lean software development – a systematic literature review of industrial studies	Kupiainen, E. Mäntylä, M.V. Itkonen, J.	Increase knowledge on which metrics are used in Agile	List of suggested metrics is provided which are relevant to the research
<i>Informatica Economica</i>	Real Time Agile Metrics for Measuring Team Performance	<i>Budacu, E.N. Pocatilu, P.</i>	Identifying most important metrics, indicators, measures, and tools for Agile	<i>Key agile methods are identified, useful to own research</i>
<i>International Seminar on Research of Information Technology and Intelligent Systems</i>	Software metrics classification for agile scrum process: a literature review	<i>Kurnia, R. Ferdiana, R. Wibirama, S.</i>	Reviewing software metrics and their fundamental role	Software metrics are classified in four groups which are each a phase of the Agile process. These suggestions are useful to my research
<i>2015 Moratuwa Engineering Research Conference</i>	Use of software metrics in agile software development process	Padmini, K. J., Bandara, H. D., & Perera, I.	Metrics applicable in Agile Software Development	Recommended metrics that can be applied to my research
<i>International conference on Agile software development</i>	A metrics model to measure the impact of an agile transformation in large software development organizations	Heidenberg, J. Weijola, M. Mikkonen, K. Porres, I.	Measuring the impact of an Agile transformation by metrics	Suggested metrics to use to measure transformations, these suggestions serve as relevant input

Appendix E Data visualisation

Jira

Projects / Portfolio management CRM

Issues

Search issues Assignee Reporter Status Type Status Category

Export issues Go to advanced search

Switch to detail view

Type	Key	Summary	Assignee	Reporter	P	Status	Created
	GIC-1890	Analyse which functionality in ACC would not work in ACSC	Unassigned			DONE	Jun 4, 2021
	GIC-1889	Analyse which functionality in ACC would not work in ACSC	Unassigned			SPRINT PREPARATION	Jun 4, 2021
	GIC-1888	As an API Designer I want to add activationcode to the response of the GET Customer API	Unassigned			SPECIFICATION	Jun 4, 2021
	GIC-1887	As an API Designer I want to add activationcode to the response of the GET Customer API	Unassigned			SPECIFICATION	Jun 4, 2021
	GIC-1886	Employees as Diamond tier members in the main loyalty programmes for Marionnaud (exc. MFR)	Unassigned			SPECIFICATION	Jun 3, 2021
	GIC-1885	Employees as Diamond tier members in the main loyalty programmes for Marionnaud (exc. MFR)	Unassigned			SPECIFICATION	Jun 3, 2021
	GIC-1881	As a business user I want to be able to approve files so that files are processed afterwards - Housekeeping	Unassigned			IN DEVELOPMENT	May 28, 2021
	GIC-1880	Performance improvement voucher process	Unassigned			DONE	May 26, 2021
	GIC-1879	Performance improvement voucher process - loyalty transactions	Unassigned			SPRINT PREPARATION	May 26, 2021
	GIC-1878	C2010-223DBSAT-AlertSiebel Recon job-failures for ICI/DR BU	Unassigned			INNO BACKLOG	May 25, 2021
	GIC-1877	Performance improvement voucher process - loyalty transactions	Unassigned			IN DEVELOPMENT	May 25, 2021

Figure 50: Overview of tickets in Jira)

Python

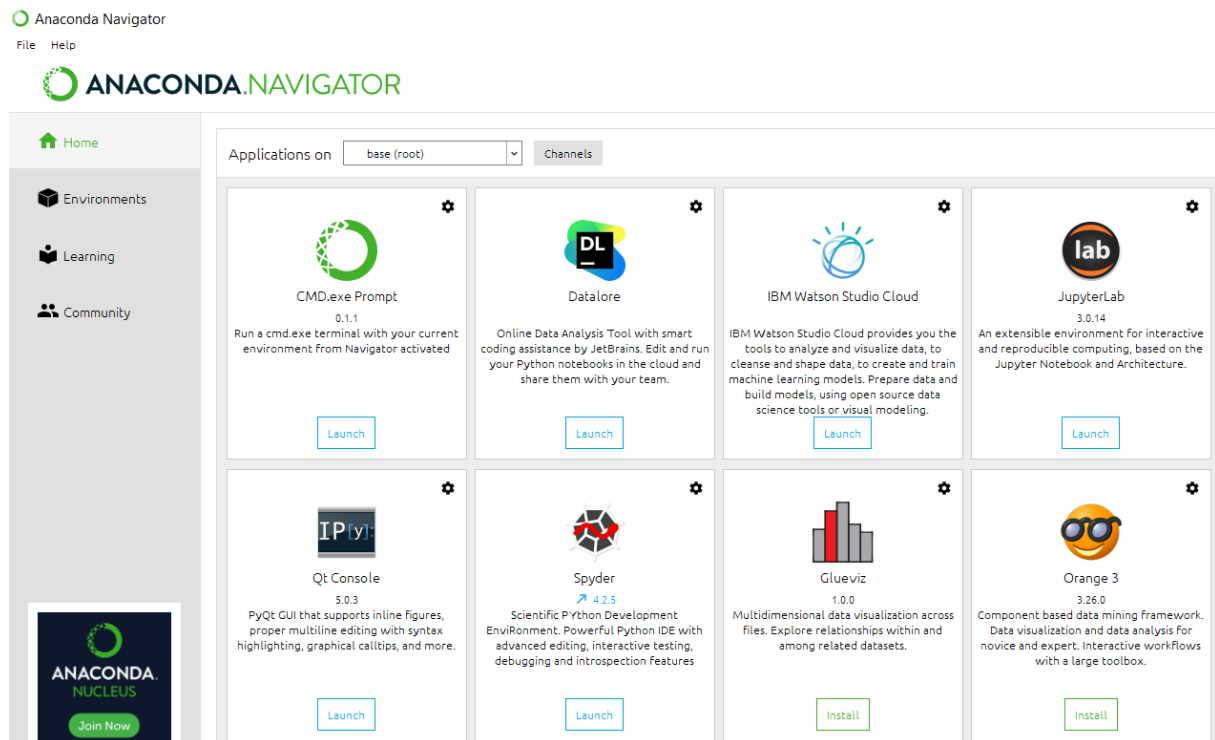


Figure 51: Anaconda platform

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.19042.985]
(c) Microsoft Corporation. Alle rechten voorbehouden.

(base) C:\Users\ceret>pip install jira
Requirement already satisfied: jira in c:\users\ceret\anaconda3\lib\site-packages (3.0.1)
Requirement already satisfied: defusedxml in c:\users\ceret\anaconda3\lib\site-packages (from jira) (0.7.1)
Requirement already satisfied: requests-oauthlib>=1.1.0 in c:\users\ceret\anaconda3\lib\site-packages (from jira) (1.3.0)
Requirement already satisfied: setuptools>=20.10.1 in c:\users\ceret\anaconda3\lib\site-packages (from jira) (52.0.0.post20210125)
Requirement already satisfied: requests>=2.10.0 in c:\users\ceret\anaconda3\lib\site-packages (from jira) (2.25.1)
Requirement already satisfied: keyring in c:\users\ceret\anaconda3\lib\site-packages (from jira) (22.3.0)
Requirement already satisfied: requests-toolbelt in c:\users\ceret\anaconda3\lib\site-packages (from jira) (0.9.1)
Requirement already satisfied: chardet<5,>=3.0.2 in c:\users\ceret\anaconda3\lib\site-packages (from requests>=2.10.0->jira) (4.0.0)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\ceret\anaconda3\lib\site-packages (from requests>=2.10.0->jira) (2020.12.5)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\ceret\anaconda3\lib\site-packages (from requests>=2.10.0->jira) (1.26.4)
Requirement already satisfied: idna<3,>=2.5 in c:\users\ceret\anaconda3\lib\site-packages (from requests>=2.10.0->jira) (2.10)
Requirement already satisfied: oauthlib>=3.0.0 in c:\users\ceret\anaconda3\lib\site-packages (from requests-oauthlib>=1.1.0->jira) (3.1.0)
Requirement already satisfied: pywin32-ctypes!=0.1.0,!0.1.1 in c:\users\ceret\anaconda3\lib\site-packages (from keyring->jira) (0.2.0)

(base) C:\Users\ceret>
```

Figure 52: Installing jira library

```

#-*- coding: utf-8 -*-
"""
Created on Thu May 20 16:24:50 2021

@author: ceret
"""

# Import dependencies
from jira import JIRA , JIRAError

# Create instance for interacting with Jira
jira = JIRA(options={'server':'https://asw-git.atlassian.net'},
basic_auth=('x.xxxx@eu.aswatson.com','{API_TOKEN}'))

# Read issue resources
jql = "project in ('GIC') AND issuetype='Change' and status='Delivered' and created > -730"
jira_search = jira.search_issues(jql)

# As an example: Extract status name of the first issue
jira_search[0].fields.status.name

# Import dependencies
from jira import JIRA, JIRAError
from collections import Counter, defaultdict
from datetime import datetime
from time import sleep

import numpy as np
import pandas as pd
import networkx as nx

# Read data from Jira
try:
    # Search issues
    block_size = 1000
    block_num = 0
    jira_search = jira.search_issues(jql, startAt=block_num*block_size, maxResults=block_size,
fields="issuetype, created, resolutiondate, reporter, assignee, status",
expand='changelog')

# Define parameters for writing data
index_beg = 0
header = True
mode = 'w'

# Iteratively read data
while bool(jira_search):
    # Container for Jira's data
    data_jira = []

    for issue in jira_search:
        # Get issue key

```



```

issue_key = issue.key
# Get request type
request_type = str(issue.fields.issuetype)

# Get datetime creation
datetime_creation = issue.fields.created
if datetime_creation is not None:
    # Interested in only seconds precision, so slice unnecessary part
    datetime_creation = datetime.strptime(datetime_creation[:19], "%Y-%m-%dT%H:%M:%S")

# Get datetime resolution
datetime_resolution = issue.fields.resolutiondate
if datetime_resolution is not None:
    # Interested in only seconds precision, so slice unnecessary part
    datetime_resolution = datetime.strptime(datetime_resolution[:19], "%Y-%m-%dT%H:%M:%S")

# Get reporter's login and name
reporter_login = None
reporter_name = None
reporter = issue.raw['fields'].get('reporter', None)
if reporter is not None:
    reporter_login = reporter.get('key', None)
    reporter_name = reporter.get('displayName', None)

# Get assignee's login and name
assignee_login = None
assignee_name = None
assignee = issue.raw['fields'].get('assignee', None)
if assignee is not None:
    assignee_login = assignee.get('key', None)
    assignee_name = assignee.get('displayName', None)

# Get status
status = None
st = issue.fields.status
if st is not None:
    status = st.name

# Get information from changelog
history_data = []
histories = issue.raw['changelog'].get('histories', None)

if histories is not None:
    for history in histories:
        for item in history['items']:
            if item['field'] == 'status':
                # Get history status, previous status, new status
                history_status = history.get('status', None)

                # Add data to data_jira
                data_jira.append((issue_key, datetime_creation, item['fromString'], item['toString'],
                datetime.strptime(history['created'][:19], "%Y-%m-%dT%H:%M:%S")))

```

```

# Write data read from Jira
index_end = index_beg + len(data_jira)
data_jira = pd.DataFrame(data_jira, index=range(index_beg, index_end),
                        columns=['Issue key', 'Datetime creation', 'From Status', 'To Status', 'Status
                                Changed@'])
data_jira.to_csv(path_or_buf='data_jira.csv', sep=';', header=header, index=True,
                 index_label='N', mode=mode)

# Update for the next iteration
block_num = block_num + 1
index_beg = index_end
header = False
mode = 'a'

# Print how many issues were read
if block_num % 50 == 0:
    print(block_num * block_size)

# Pause before next reading – it's optional, just to be sure we will not overload Jira's server
sleep(1)
print(data_jira)
jira_search = []

jira.close()

except (JIRAError, AttributeError):
    jira.close()
    print('Error')

```

Appendix F Dashboard



Figure 53: The dashboard

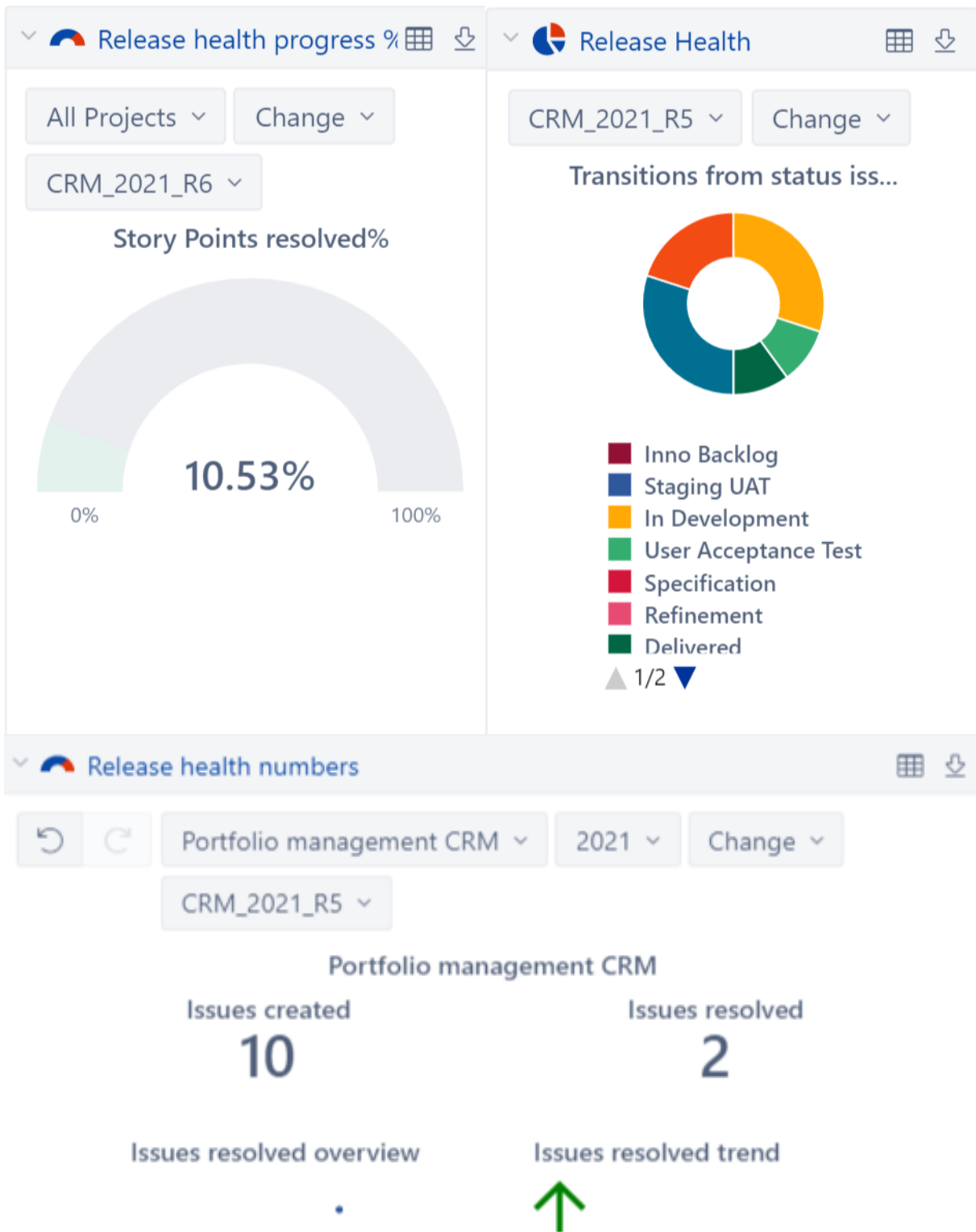


Figure 54: Release health metric

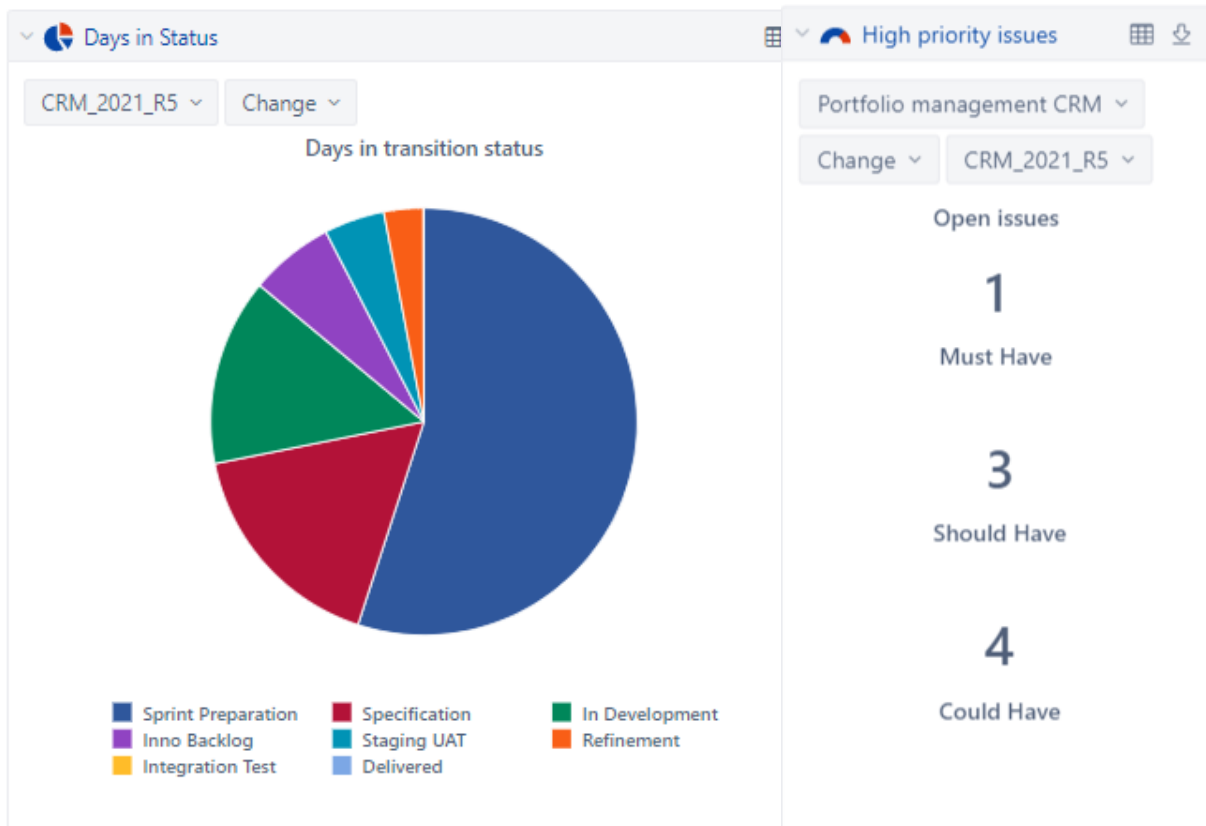


Figure 56: Days in status metric

Figure 55: High priority issues

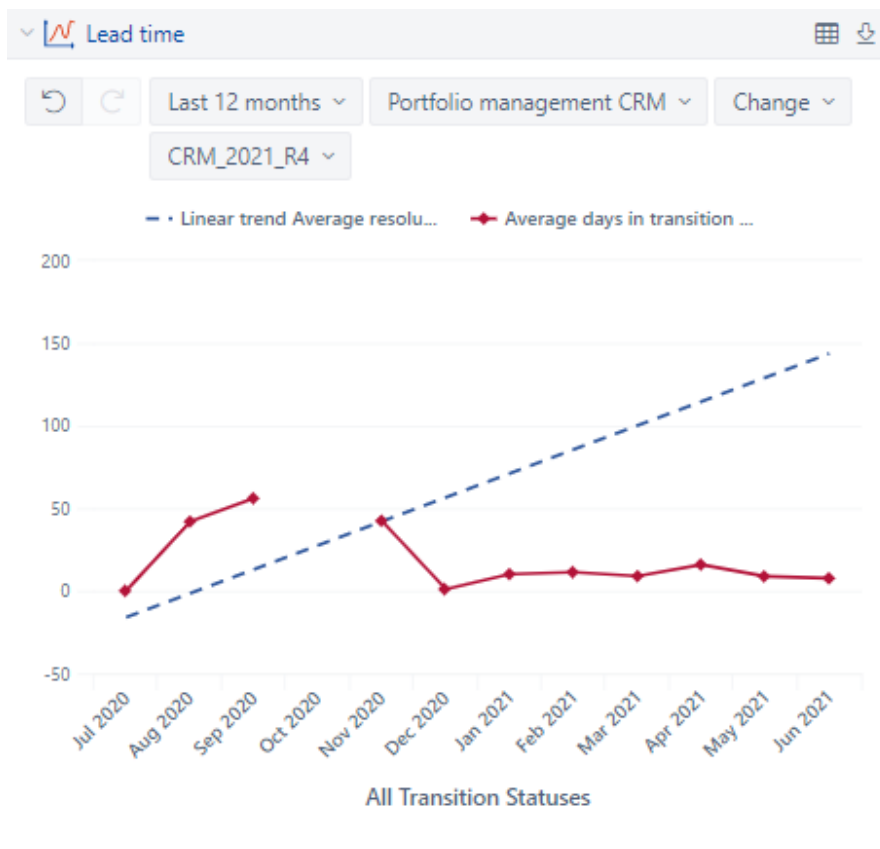


Figure 57: Lead time metric



Figure 58: Average age metric

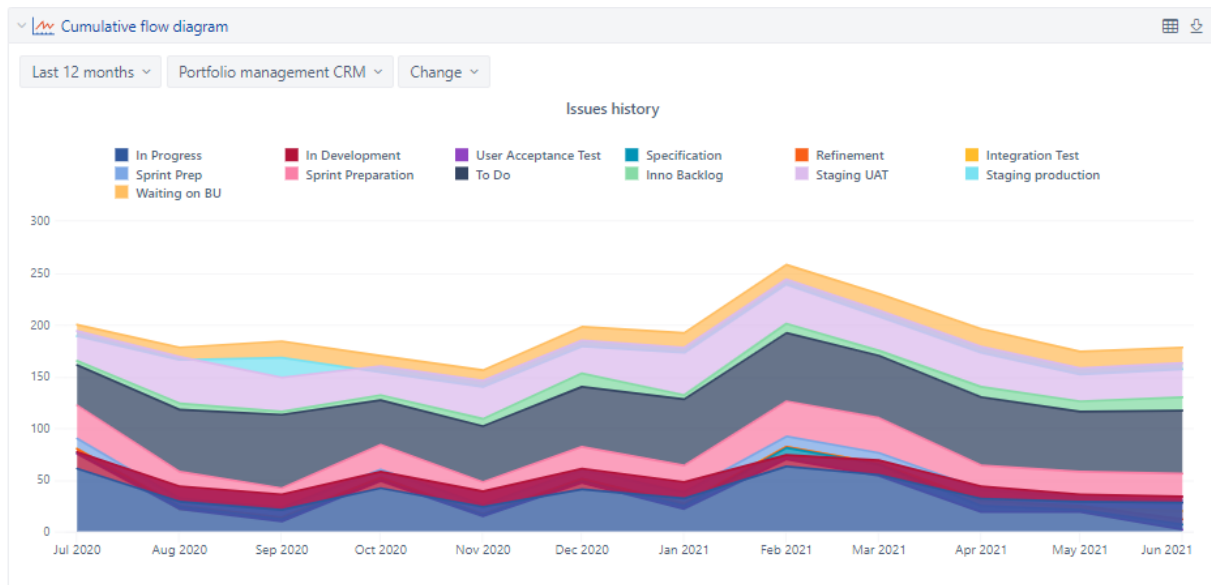


Figure 59: Cumulative flow metric