UNIVERSITY OF TARTU
Institute of Computer Science
Software Engineering Curriculum

Khaled Nimr Charkie

Focus Factor and Hyper-Productivity of Agile Teams: A Study of 8 Open-Source Projects

Master's Thesis (30 ECTS)

Supervisor: Ezequiel Scott

Tartu 2020

Focus Factor and Hyper-Productivity of Agile Teams: A Study of 8 Open-Source Projects

Abstract:

One of the ways to understand and improve the productivity of agile project teams is through the usage of metrics that monitor productivity. Despite the emergence of new methodologies and metrics in recent studies promoting productivity increase in agile teams, the utilization of these metrics should be accompanied by a thorough understanding of the factors that affect them. This thesis analyzed data extracted from JIRA repositories of 8 open-source projects. Two productivity metrics, focus factor, and velocity increase were calculated. The projects' sprints were classified as hyper-productive and healthy using the values of the calculated metrics. In addition, different project factors such as the team size, the number of new team members, and the sprint length were studied. This thesis aims to describe the differences between the productivity metrics with their corresponding states and the project factors. The results showed that there are only statistically significant differences between the team sizes of hyper-productive and non-hyper-productive sprints, while there are no statistically significant differences between the project factors and the healthy and non-healthy sprints. This thesis also highlights some limitations of these metrics and their corresponding states after applying them to the real open-source projects' data.

Keywords:

Agile, scrum, productivity, open-source, metrics, hypothesis testing, hyper-productivity.

CERCS: P170 - Computer science, numerical analysis, systems, control

Väleda tarkvaraarenduse meeskondade fookusfaktor ja hüperproduktiivsus: kaheksa avatud lähtekoodiga projekti uuring

Lühikokkuvõte:

Väleda tarkvaraarenduse meeskondade produktiivsuse parandamiseks ja mõistmiseks on võimalik kasutada erinevaid vahendeid, mis monitoorivad projekti kasulikkust. Hiljutistest uuringutest selgunud uued metoodikad ja mõõdikud, mis edendavad väleda tarkvararenduse meekondades produktiivsuse kasvu, peaks nende mõõdikute kasutamisega kaasnema neid mõjutavate tegurite põhjalik mõistmine. Käesolevas magistritöös analüüsitakse JIRA andmehoidlatest saadud kaheksa avatud lähtekoodiga projekti andmestikku. Töö käigus kalkuleeritakse välja kaks produktiivsuse mõõdikut, fookustegur ja kiiruse kasv. Nende mõõdikute väärusi kasutades liigitatakse projektide tulemused ülitõhusateks ja tervislikeks. Lisaks uuriti projektitegureid, nagu meeskonna suurus, uute meeskonnaliikmete arv ja sprintide pikkus. Lõputöö eesmärgiks on kirjeldada produktiivsuse mõõdikute, neile vastavate olekute ja projektitegurite vahelisi erinevusi. Tulemused näitasid, et hüperproduktiivsete ja hüpoproduktiivsete meeskondade suuruste vahel esinesid statistiliselt olulised erinevused, samas kui projektitegurite, tervislike ja ebatervislike sprintide vahel ei esine statistiliselt olulisi erinevusi. Lõputöö toob välja ka mõndade mõõdikute piirangud ja neile vastavad olekud pärast nende rakendamist reaalse avatud lähtekoodiga projektide andmetel.

Võtmesõnad:

Väle (agile), scrum, produktiivsus, avatud lähtekood, meetrika, hüpoteesi testimine, hüperproduktiivsus.

CERCS: P170 Arvutiteadus, arvanalüüs, süsteemid, kontroll

Table of Contents

1 Introduction

Productivity in software engineering is a broad concept [1]. With the continuously added complexities to software development from new technologies and methodologies, relying only on delivered work as a productivity measure is not enough to capture the different processes that a team goes through. As most of the work in software development teams are organized and planned using agile frameworks, metrics are not only considered an indicator of a team's performance but as a way to save costs, organize resources, and improve the development lifecycle [2],

Since no single metric can capture productivity, as it is explained by Sadowski and Zimmermann [1]. Teams started to use a wide range of metrics depending on their needs [2], [3]. In fact, velocity is one of the most popular metrics, as shown by Kupiainen et al. [3]. However, metrics involve the use of story points and, which are team-specific, and not suitable for being used in comparisons across multiple teams. A study by Downey and Sutherland [4], [5] was made to solve this issue. This study explained how proper usage of the scrum framework with an agile coach allowed a team to reach a high productivity level known as the "hyper-productive state". In addition, this study addressed the problems faced when using story points for comparison between teams by introducing a group of metrics known as the "hyper-productive metrics".

A drawback in Downey and Sutherland's study is that it did not describe the factors that allowed the teams to reach high levels of productivity or the circumstances they went through. Factors can be faster than some metrics in predicting problems in a team, and the best way to improve productivity is to understand the factors affecting it [1]. Hence, this study tries to utilize the metrics (focus factor and velocity increase), and their corresponding states (healthiness and hyper-productivity) introduced in [4] but on open-source software. What this study aims for is to verify the possibility of open-source projects in reaching hyper-productive and healthy states, and identify the factors that may facilitate a team to reach them.

1.1 Problem Statement

Every team aims for better productivity. However, productivity will always be a mystery for teams if the common factors and reasons behind it are unknown or not adequately explained. The results proved by Downey and Sutherland in [4] showed that the usage of agile frameworks and hyper-productivity metrics build the foundation for teams aiming to increase their general productivity.

Even though hyper-productivity was explained in [4], a minimum explanation was given on the factors that help a team to reach hyper-productivity, especially since this study was performed on closed-source software teams. Our aim is to understand the factor and identify the capabilities of open-source software teams in reaching a hyper-productive state, taking into consideration the metrics and states proposed by [4], such as focus factor and hyper-productivity.

To achieve this, I will use the data from 8 open-source projects to calculate the focus factor and the hyper-productivity states. Then, I propose several hypotheses that cover specific factors affecting hyper-productivity states and the focus factor. This thesis seeks to answer the following research questions:

1. What are the contextual factors that facilitate a team to reach a hyper-productive state?
2. What are the contextual factors that facilitate a team to have a healthy focus factor?

2 Background

Productivity is considered one of the most critical variables in economic production and is often neglected by those who control the production process [6]–[8]. Productivity is a multidimensional term used or viewed differently depending on the area it is utilized in. This reason may explain why several studies gave it different interpretations. However, most of them agree that productivity is simply the ratio between the output and the input, as shown in equation (1):

$$\text{Productivity} = \text{Output} / \text{Input} \tag{1}$$

In this formula, the output can be referred to as the produced goods, while the input is the consumed resources. Nevertheless, this interpretation cannot suit all the fields it is applied to. For instance, measuring the output and input in a factory can be possible. But the situation is different if we want to identify the output and input for authors, scientists, and engineers, also known as knowledge workers [1], [9]. For this reason, Tangen [9] has opted for a different way of explaining and describing productivity and identified five terms closely related to it. Sadowski and Zimmermann [1] have summarized the terms as follows:

1. Profitability: Similar to productivity and often confused with it. Profitability has more factors affecting it compared to productivity and can change without any effect on productivity. It is more monetary oriented and represents the ratio between revenue and cost, which influences price-factors.
2. Performance: Broader than productivity and profitability and cover factors influencing a company's success such as quality, delivery, speed and flexibility
3. Efficiency and Effectiveness: Often confused with each other and with productivity. Efficiency refers to the proper utilization of resources (it affects the input of productivity ratio). Effectiveness aims at the usefulness of the output (it affects the output of productivity ratio).
4. Influence of Quality: Quality is essential when describing the productivity of knowledge workers. It should be at its maximum value and is considered more important than the volume of work, as outlined by Drucker [10].

In software development, the Agile Manifesto [11] introduced several principles that were the basis of the agile methodology. These principles organize the delivery of software, communication between stakeholders, the internal structure of teams, and the quality and effectiveness of work. In a way or another, these principles are connected to what Tangen [9] has mentioned about productivity. This makes the usage of agile a way to boost a team's productivity, which was proven in different studies [12], [13].

In this study, the focus will be on open-source projects, which are open-source software backed by the internet community of programmers. Those projects can be used by anyone for free, the source code these projects are available for the public, participation is voluntary, and contributors do not receive direct compensations. The interesting part of these projects is that they change the economics of traditional software development. In contrast to private software projects which suffer from a scarcity of resources, open-source projects do not suffer from this issue. This gives the study of productivity in these projects an additional value [14].

3 Related Work

Productivity in software development is not a new topic, as research addressing productivity in software development existed for several decades. The systematic review of productivity factors by Wagner and Ruhe [15], which is considered a starting point for understanding factors affecting productivity in software development, has found studies from four different decades (70's, 80's. 90's and 2000's). In their description of these studies, it was clear how did the research studies develop from the '70s till today. Even though some findings in the earlier studies are not relevant for today's use, it is surprising that some factors that affect productivity nowadays, like the project size, the programming language used, and developer experience, were addressed back then.

The vast number of studies included in Wagner and Ruhe's [15] review resulted in an important classification of productivity factors. Moreover, this classification was followed in other studies, such as [1]. The factors are divided into two main categories, technical factors, and soft factors. The technical factors are divided into three categories, while the soft factors are divided into five. The main findings in this study show that it is worth investing in proper team communication despite the increase in team size. Also, the business domain plays a role in productivity and its factors. One notable finding is that the studies favored the capabilities of developers and not their years of experience as a factor of productivity for individuals. It was shown that factors in different categories share common values. Hence, technical and soft factors may overlap.

Other studies have taken different approaches to identify the factors affecting productivity. In a study by Fatema and Sakib [16], data were collected from various companies in the form of surveys and analyzed to find the factors with the most influence among the participants. The study has shown that the effectiveness of an agile team relies on multiple interrelated factors such as excellent communication, leadership, adaptability, motivation, and self-management. Other factors, such as team training and backup behavior, had the lowest influence on productivity, according to the participants.

A study by Rodriguez et al. [17] took a different approach as it focused mainly on one factor, team size. This study used statistical analysis and machine learning methods on data obtained from a project repository. The results showed that there exists a correlation between team size and productivity, as the team size increase to more than nine people, the less productive the team. The study has also shown that programming language has an impact on productivity. One important note the study has identified is that using repositories with large amounts of data has several drawbacks, such as errors in the data and missing values. All that makes the results less reliable when using statistical and data mining techniques.

With the variety of studies explaining factors influencing productivity in software development teams, there was a necessity in organizing these factors in a way that could apply to the teams in realistic conditions.

The necessity of organizing productivity factors according to applicable conditions grew with the increase in the number of studies addressing these factors. For this reason, some studies, such as Dingsøyr [18], have constructed a performance model inspired by multiple factors that are vital for a team's productivity. This model includes five different categories:

1. Team coordination: The importance of having good coordination and management among team members to ensure success.
2. Goal orientation: The importance of having a clear goal.
3. Team cohesion: The importance of having a team sticking together in different situations.
4. Shared mental models: Coordinated knowledge among team members that enable them to solve different tasks.
5. Team learning: The process of actively seeking answers, feedback, and analysis of produced results or errors in the team.

Some studies intended to know if these models are effective in development teams. A study by Dingsøyr and Lindsjørn [19] has focused on finding how effective proposed productivity models are in identifying factors influencing productivity. This study has conducted 18 focus groups to see if the model proposed by Salas et al. [20] reflects how different teams perceive team productivity. The results showed that the model and the participants' perceptions of factors affecting productivity are well fitted. The most factors that teams are aware of can be classified under "team leadership," which can be related to the ability to direct team members, activities, and other resources. The category with the lowest awareness is "backup behavior" which is related to the ability to anticipate and predict team members' needs and achieve workload balance among the team.

The study by Sudhakar et al. [21] gave ten techniques or models extracted from other studies for measuring a team's productivity. Other studies focused solely on metrics used in agile teams with no mention of productivity, such as the study by Kupiainen et al. [3]. The aim of this study was to identify the reasons and effects of using metrics in agile teams through a systematic literature review. The results have been that velocity and effort estimates are the most used metrics, while the reasons behind using them lie on project planning and team motivation.

The studies reviewed showed that there exists a considerable focus on identifying productivity factors and models. On the other side, there is enough work done on agile metrics and their influence on a team's work. However, little research has been done in identifying relations between metrics and productivity factors using statistical differences. Thus, a study that can unite both the usage of metrics and factors seemed a logical starting point for this study, which can add valuable information that improves agile teams' productivity. The summary of the studies in this section is in table 1:

Table 1. Summary of the studied related work.

Study Name	Source	Focus Area	Used methods
A Systematic Review of Productivity Factors in Software Development	[15]	Identification and classification of productivity factors.	Systematic literature review
Factors Influencing Productivity of Agile Software Development Teamwork: A qualitative system dynamics approach	[16]	Identification of productivity factors in software companies.	Interviews, surveys, and data analysis.
Empirical Findings on Team Size and Productivity in Software Development	[17]	Effect of team size on productivity.	Statistical analysis and empirical study
Team Performance in Software Development: Research Results versus Agile Principles	[18]	The proposition of a performance model.	Systematic literature review
Team Performance in Agile Development Teams: Findings from 18 Focus Groups	[19]	Identification of productivity factors through performance models.	Focus groups and empirical study
Measuring productivity of software development teams	[21]	Productivity measuring techniques and factors	Systematic literature review
Using metrics in Agile and Lean Software Development – A systematic literature review of industrial studies	[3]	Reasons and effects of using metrics in agile teams.	Systematic literature review

4 Methodology

This chapter includes the descriptions of the different datasets used and the steps conducted in cleaning and pre-processing the data. In addition, the metrics and their corresponding states in this study were identified, and a brief description of their calculation method was given. The datasets used in this study, together with the scripts of cleaning, calculating, and analysis of data, are in the following repository (https://github.com/chefk5/ThesisKhaled-2020).

4.1 Datasets

The datasets used in this study contain issues belonging to Jira projects for eight different open-source products. These datasets have also been used in previous studies [22], [23], [24]. Table 2 summarizes the products used in this study.

Table 2. Projects used in this study. Details adapted from [22].

Project	Product Name	Product Description	Website
APSTUD	Aptana Studio	Web development IDE	http://www.aptana.com/
NEXUS	Sonatype's Nexus	Repository manager	https://www.sonatype.com/
TISTUD	Appcelerator Studio	Integrated IDE for mobile development	http://www.appcelerator.com/
XD	Spring XD	Runtime Environment for big data applications	https://projects.spring.io/spring-xd/
MESOS	Apache Mesos	A distributed systems cluster manager	http://mesos.apache.org/
DNN	Dnn Platform	Online content management system	https://www.dnnsoftware.com/platform
TIMOB	Titanium SDK/CLI	Node.js-based command-line tool for Titanium projects.	https://jira.appcelerator.org/projects/TIMOB/issues
MULE	Mule	A Java-based integration platform	https://www.mulesoft.com/

In order to calculate the metrics required for this study, several fields from the datasets were used. The list of fields is grouped and summarized below:

- Name fields: These are fields that show the project names, sprint names, and contributors names, including assignees, reporters, and creators. These selected fields include:
  - "project"
  - "sprint"
  - "author"
  - "assignee.name"
  - "key"

- Date fields: These are fields containing dates. In this study, the update field was used for the issues both in the issue list and changelog for determining when the issue was closed. In addition, the sprint start date and end date fields were used to determine the sprint duration. These selected fields include:
    - "updated"
    - "startDate"
    - "endDate"
    - "created"
- Status fields: These are fields containing the status of an issue or a sprint. They include:
    - "status.name"
    - "field"
- Value fields: These are fields containing the story points for each issue. They were sued to calculate the metrics in this study. They include:
    - "storypoints"

The dataset consisted of 18090 issues belonging to 8 projects with 332690 issue logs. The issues were tracked from 632 sprints that included 1268 contributors. The dataset also included 20 unique issues statuses used, which varied per project.

Table 3 describes the number of sprints, issues, developers, and possible statuses of each project in the original dataset.

Table 3. Datasets issues

Project	Number of Sprints	Number of Issues	Number of contributors/developers	Issue status
APSTUD	34	886	171	Closed, Resolved Open, In Review Reopened
NEXUS	72	1268	69	Open, Refine Closed, Done Raw
TISTUD	59	2879	210	Reopened, Closed Resolved, Open In Review, In Progress
XD	66	3784	75	To Do, In PR, Done, In Progress
MESOS	71	2304	128	Open, Reviewable In Progress, Resolved Accepted
DNN	107	3328	135	Closed, Open, Reopened, Pull Request Submitted, Resolved, Planned Development
TIMOB	107	2144	377	In Progress, Open, Resolved, In Review, Closed, Reopened
MULE	116	1497	103	Closed, Resolved, To Do, In Progress Reopened

4.2 Data cleaning and pre-processing steps

In order to ensure that the datasets are in the best quality prior to any calculations or modifications, several data cleaning and pre-processing steps were performed on the data utilized in this study. The cleaning was conducted in iterations and accompanied by the calculation and analysis of data. The steps can be classified into three levels: project, issue, and sprint.

Project:

- Selected fields were analyzed for errors in the data. The analysis resulted in missing sprint end dates in TIMOB and unclear sprint names in MULE affecting all the projects' rows. These two projects were removed from the study.
- Projects like DNN and MESOS seemed to have different sub-projects in their repository. For example, it was better to divide sprints with names like "Content 9.0 - Sprint 1" and "Platform 8.1 - Sprint 2" into separate sub-projects. Thus, MESOS was divided into MESOS1 and MESOS2. DNN was manually divided into five projects DNN1, DNN2, DNN3, DNN4, and DNN5.
- Projects consisting of less than ten sprints were eliminated. Projects in this category are DNN2 and DNN5. After metrics were calculated and organized by project, it was perceived that projects with less than 5% of either hyper-productive or healthy sprints would lead to wrong conclusions. For this reason, projects under this category, such as DNN1, DNN3, and DNN4, were excluded from the study.

Issues:

Issues meeting the following criteria were removed:

- Issues that were not assigned to any sprint.
- Duplicate issues.

Sprints:

Sprints meeting the following criteria were removed:

- Sprints with no sprint start and end dates were excluded.
- Sprints with 0 velocities were removed since they had a negative impact on the metrics and states.
- Eight initial sprints were removed in NEXUS as their velocity values appeared to be incorrect.

After cleaning, the grouping of issues according to sprints and calculation of metrics was done. The final used dataset used consisted of 7163 issues and 326 sprints belonging to 9 projects (including sub-projects) and 852 developers. The cleaned dataset is summarized in table 4.

Table 4. Dataset projects after data cleaning.

Project	Number of Sprints	Number of Issues	Number of Developers
APSTUD	27	397	100
NEXUS	59	548	46
TISTUD	53	1672	150
XD	66	2155	55
MESOS1	30	329	60
MESOS2	26	783	100
DNN1	28	717	108
DNN3	15	45	123
DNN4	22	517	110
All Projects	326	7163	852

4.3 Metrics

One of the ways to view and understand productivity in different teams is through the utilization of metrics. The hyper-productivity metrics introduced by [4] covers a wide range of metrics measuring various aspects of productivity. On the other hand, developers reported that metrics capturing the number of tasks they have done in a specific time period are the best metrics to assess their productivity [25]. For this reason, metrics that serve this purpose were given priority in this study.

4.3.1 Velocity

Velocity is the amount of work that is completed at the end of each sprint [26]. In other words, it is a metric that identifies the amount of product backlog that has been completed in a specific sprint [27]. Velocity is usually measured by adding the sizes of completed product backlog items (PBIs) in a certain sprint.

In this study, the velocity of a sprint represented the sum of story points of issues that were closed - “status” field marked as either “Closed” or “Resolved” - before the end date of the sprint using the “updated” field.

Velocity is also one of the most popular, essential, and relevant measures, as shown by [3]. One of the reasons that make velocity popular is the versatility of its usage [26] as it can be used for planning, estimating work, measuring work done, and improving the delivery of customer value.

Velocity is a measure proper of each team, and it is affected by the current circumstances and conditions that each team is facing. Hence, a velocity value in a particular team can be interpreted differently in another team [28]. This can be one of the significant drawbacks of this metric since it cannot be used for team comparison. As an alternative, Downey and Sutherland [4] introduced two similar metrics to cope with this problem.

4.3.2 Baseline Velocity

Baseline velocity is defined initially as the velocity of the first sprint in a project [4]. In this study, the baseline is considered as the average of the first two sprints as it resulted in more realistic results. This metric does not bring significant value as it is meant to be used in obtaining other metrics, such as the velocity increase.

4.3.3 Percentage of Velocity Increase

The percentage of velocity increase indicates to what extent a team's velocity improved compared to its first sprint (baseline velocity). This metric merges the usage of velocity and baseline velocity into one metric, giving a value that can serve as an indicator of a team's performance. The percentage of velocity increase is calculated as described by Equation (2).

$$\text{Percentage of Velocity Increase} = \frac{\text{Current Sprint Velocity}}{\text{Baseline Velocity}} * 100 \quad (2)$$

4.3.4 Work Capacity

Work capacity is introduced by Downey and Sutherland [4] in order to deal with the drawback of using velocity. Velocity is a metric that does not benefit from partially completed items since it concerns about the size of what is delivered (output) rather than the value of what is delivered (outcome) [26].

Therefore, work capacity ensures the measurement of all the work reported during the sprint, whether it was marked as "Done" or not [4]. This metric is usually used together with velocity in order to derive the total amount of work in a sprint.

In this study, work capacity was calculated as the sum of story points of all the issues that were assigned to a sprint, no matter its status.

4.3.5 Hyper-Productivity state

Downey and Sutherland [4] introduced the term hyper-productivity when they aimed to increase the overall productivity of teams in a case study. One of its main objectives was to enable teams to achieve a velocity of 400% higher than their baseline velocity with the help of a Scrum coach and a proper usage of the Scrum framework [5].

Based on their definition, a team is considered as hyper-productive if it reaches a hyper-productive sprint. A sprint is considered hyper-productive if its percentage of velocity increase surpasses 400%.

4.3.6 Focus Factor

The focus factor shows how focused the team is toward achieving the sprint goal [29]. In other words, this ratio represents how much of the planned work was done. This metric is strongly related to the hyper-productivity state introduced in [4]. Its usage can give a better understanding of the general productivity of a team with respect to the work planned to be done. The focus factor is calculated as described by Equation (3).

$$\text{Focus Factor} = \frac{\text{Velocity}}{\text{Work Capacity}} \quad (3)$$

4.3.7 Healthy state

The study done by Downey and Sutherland [4] mentioned that a "healthy" focus factor should be in the neighborhood of 80%. Also, they state that when the focus factor goes too high, it generally indicates that the team did forecasts to appear perfect. However, their study does not define precise values for the neighboring values and "high" values. For this reason, we consider

a focus factor in between 70% and 90% as healthy; otherwise, we consider the sprint as unhealthy.

Different interpretations have been given by [4] regarding the interpretation of different values of focus factor:

- A focus factor around 80% is referred to as "healthy" (in our study, sprints with a focus factor from 70% to 90% were considered healthy).
- A focus factor below 70% reveals that a team is struggling to accomplish the assigned work planned at the beginning of the sprint, or it is facing some external events affecting their productivity. This focus factor is referred to as "unhealthy".
- A focus factor that is too high (we assume it is 90% or above) indicates that a team is avoiding completing their assigned work or trying to look "perfect" from the metrics perspective. This focus factor is referred to as "unhealthy."

4.4 Contextual Factors

To understand productivity, a set of factors that characterize the teams were defined. Factors including team size, number of new team members, and sprint duration have been shown that have an effect on productivity [1], [15], [17], [30]–[32]. Therefore, this thesis analyses the influence of these factors on the focus factor and hyper-productivity. These factors differ from team to team and from sprint to sprint. Studying these variables will help us in recognizing some factors that influence these metrics and productivity as well. The contextual factors chosen in this study are the following:

- **Number of current developers in a sprint:** Those are the developers who had their names assigned to issue at any time during the sprint. Also, these issues were marked as "Done" before the end of the sprint.
- **Number of New Developers in a Sprint:** Those are the number of developers who were not assigned to any issue in the previous sprint but who have their names assigned to issues in the current sprint. These issues should be marked as "Done" before the end of the current sprint in order to be considered. *(example: Sprint 1 has five developers, while sprint 2 has eight developers. Then the number of new developers in sprint 2 is three developers).*
- **Sprint length:** It is the duration of the whole sprint iteration in days. The start and end dates were already provided in the original JIRA data, but the duration of the sprint in days was calculated by counting the days from the sprint start date towards the sprint end date.
    - In the case of projects having sprints with explicitly defined start and end date, the sprint length was calculated as (end – start)
    - When the start date was not available, we consider the start as the end of the previous sprint (we identified the order of sprints by their name which includes ordinal numbers – for example, "sprint 1", "sprint 2")

4.5 Research Questions and Hypotheses

In this section, we list and explain the research questions formulated around the metrics and factors mentioned in sections 3.3 and 3.4. And based on each research question, a series of hypotheses are defined. This method allows us to approach the research question from its different angles, understand the variables and metrics used, and analyze their overall effect.

4.5.1 Research Question 1 (RQ1): What are the contextual factors that facilitate a team to reach a hyper-productive state?

Motivation: Agile emphasizes the importance of people, such as customers and developers, and organizes their communication and contribution throughout the development lifecycle. One of the agile manifesto principles [11] assures that teams shall be keen on improving their effectiveness. On the other hand, Downey and Sutherland [4] have identified a hyper-productivity state, which is defined as a 400% increase in the baseline velocity. This state can be utilized in identifying and comparing productivity across various teams. Thus, it is considered wise in understanding how we can use one of the Agile's constituents – people – in increasing productivity.

Procedure: To answer this research question, the percentage of the velocity increase was calculated for each project understudy on a sprint basis. The percentage of velocity increase allows for identifying projects that reached hyper-productivity and which did not. Then, we propose hypotheses to identify any linkage among the factors and the hyper-productivity. These hypotheses are verified using statistical tests. After verifying the hypotheses, we interpreted the results accordingly.

Hypothesis 1 (H1): There is a difference in the number of developers between hyper-productive and non-hyper-productive sprints.

Motivation: Behind each sprint lies a team of developers that are responsible for solving the assigned issues. Claudia et al. [30] state that smaller teams have better communication, commitment, and easier conflict management. On the other hand, Agarwal and Mujamdal [27] showed that increasing the team size proportionally increases the complexity of the team by four times.

The studied projects have sprints with teams of different sizes behind them. This may be one of the causes behind the variation of metrics and the overall productivity of the teams. In addition to the identification of hyper-productive sprints, it is essential to find the best team size behind the sprints that we classified as hyper-productive. Hence, we decided to verify if team size affects the productivity metrics in the projects in this study.

Hypotheses:

In hypothesis testing, the null hypothesis H_0 represents the base condition where there is no difference in the parameters of each population. While the alternative hypothesis H_a represents the hypothesis contradicting the null hypothesis, which is the same as H1. The same pattern will be followed in the other hypotheses.

***$H1_0$*:** There is no difference in the number of developers between hyper-productive and non-hyper-productive sprints.

***$H1_a$*:** There is a difference in the number of developers between hyper-productive and non-hyper-productive sprints.

H2: There is a difference in the number of new developers in hyper-productive and non-hyper-productive sprints.

Motivation: Some studies [13] suggest that having new developers in a team may contribute to decreasing the productivity of the team. As these developers are new to the project, and they need some time to mature and give insightful opinions that benefit the project. Also, some new developers may require a massive amount of time to start contributing to the project, which in other words, means that other developers are carrying additional responsibilities in assisting them to finish their assigned tasks. Thus, decreasing the overall productivity. In contrast, [30]

suggests that new developers view the current project from a different perspective, making them able to identify more efficient ways to solve existing problems and increase the overall productivity of the team.

In terms of hyper-productivity, we should see a decrease in the number of new developers in hyper-productive sprints. Hence, proving this hypothesis will assure that hyper-productive teams are also affected by the number of new developers in the team.

Hypotheses:

***$H2_0$*:** There is no difference in the number of new developers between hyper-productive and non-hyper-productive sprints.

***$H2_a$*:** There is a difference in the number of developers between hyper-productive and non-hyper-productive sprints.

H3: There is a difference in the sprint length of hyper-productive and non-hyper-productive sprints.

Motivation: Kaur and Jajoo [33] have found that one of the most common challenges agile teams are facing is having an unfixed sprint length. This was supported in a study by Mohallel and Bass [32], which found that four out of seven companies in their study changed their sprint length according to the amount of workload they have. On the other hand, Rubin in the Essential Scrum [26] has mentioned the necessity of having a consistent sprint length throughout the project. Additionally, he stated that a team should not alter the sprint length in case of the incapability to finish the planned sprint at the required time, addressing this issue as "symptoms of disfunction." Since, in our study, having a fixed sprint length was not common between all the studied projects, we have decided to verify if altering the sprint length may be one of the keys to having a hyper-productive team.

Hypotheses:

***$H3_0$*:** There is no difference in the sprint length between hyper-productive and non-hyper-productive sprints.

***$H3_a$*:** There is a difference in the sprint length between hyper-productive and non-hyper-productive sprints.

4.5.2 RQ2: What are the contextual factors that facilitate a team to have a healthy focus factor?

Motivation: The first principle in the Agile Manifesto [11] explains that continuous and early delivery of valuable software is one of the main goals of Agile. We can identify from this principle three crucial keywords, which are "continuous," "early," and "valuable." While the first two are the goals of any Agile or hyper-productive team, the term "valuable" can be related to the focus factor metric as Downey and Sutherland explain that the focus factor shows the amount of value delivered from the current effort [34]. Even though Downey and Sutherland [4] stated that a healthy focus factor should be around 80%, limited information was given regarding the factors or reasons behind reaching this value. For this reason, studying each sprint may reveal some factors a team could use for obtaining a healthy focus factor.

Procedure: In order to answer this research question, the first step is obtaining the focus factor of each sprint, which will be classified as "unhealthy" (below 70% or above 90) and "healthy" (around 80%). On the other hand, we need to analyze different variables or metrics and identify any relation with respect to the "healthy" and "unhealthy" focus factor. This will be done by conducting some hypotheses which will be verified using U-tests on the means of the variables

used. This will be followed by a more in-depth analysis to understand and identify the aspects that affect the team's focus factor.

H4: There is a difference in the number of developers between healthy and unhealthy sprints

Motivation: Similar to RQ1-H1, the number of developers influences the focus factor of the teams studied. As Downey and Sutherland [4] identified the focus factor as a hyper-productive metric, we suppose that any effect from the number of developers on the hyper-productivity shall be reflected on the focus factor.

Hypotheses:

***$H4_0$*:** There is no difference in the number of developers between healthy and unhealthy sprints.

***$H4_a$*:** There is a difference in the number of developers between healthy and unhealthy sprints.

H5: There is a difference in the number of new developers between healthy and unhealthy sprints.

Motivation: The motivation behind this hypothesis is the same as in H2 of RQ1, but with the focus factor as a metric. Here we propose that the number of new developers influences the value of work delivered in the sprint. Verifying this hypothesis will enhance our chances of knowing the optimum trends in the number of new developers in order to reach a healthy focus factor.

Hypotheses:

***$H5_0$*:** There is no difference in the number of new developers between healthy and unhealthy sprints

***$H5_a$*:** There is a difference in the number of new developers between healthy and unhealthy sprints.

H6: There is a difference in sprint length between healthy and unhealthy sprints.

Motivation: The motivation behind this hypothesis is the same as in H3 of RQ1, but with the focus factor as a metric. This hypothesis will identify if the teams that can deliver 80% of their assigned work tend to have a specified sprint length, which provokes them to reach this value.

Hypotheses:

***$H6_0$*:** There is no difference in the sprint between healthy and unhealthy sprints.

***$H6_a$*:** There is a difference in the sprint length between healthy and unhealthy sprints.

5 Results

In this chapter, the research questions stated previously are answered after the metrics are calculated and portrayed, the contextual variables are analyzed, and the hypotheses are tested.

5.1 Descriptive Statistics

In this section, we perform some descriptive statistics for both the metrics and variables used. Performing this kind of statistics is the first step in understanding the size and characteristics of our data and may serve as a logical way to highlight trends and relations. All the measurements in the tables below can be classified under "measurements of central tendency."

Table 5. Descriptive statistics of velocity for the six projects studied.

Velocity (measured in story points)						
Project	Number of Sprints	Mean	Median	Std. Deviation	Minimum	Maximum
APSTUD	27	76	41	77	8	290
NEXUS	59	7	5	7	1	33
TISTUD	53	85	81	58	3	254
XD	66	95	81	68	1	294
MESOS1	30	14	13	9	1	36
MESOS2	26	44	37	31	1	101

Table 5 shows the velocities of the studied projects. The velocity varied from an average of 7 to 95 story points. This was reflected in the other measurements, such as the median and the standard deviation. The maximum column confirms that velocity is team-specific, and each team has a different way of giving points. For example, NEXUS has a mean of 7 story points and a maximum of 33 story points, while XD has an average of 95 story points and has a maximum of 294 story points. The common characteristic among the projects is their minimum velocity that was less than ten story points across all the projects. Sprints with low story points were located either at the start or the end of the projects.

Table 6. Descriptive statistics of work capacity for the six projects studied.

Work Capacity (measured in story points)						
Project	Number of Sprints	Mean	Median	Std. Deviation	Minimum	Maximum
APSTUD	27	123	148	99	8	352
NEXUS	59	10	8	8	1	38
TISTUD	53	184	168	77	52	384
XD	66	108	94	74	1	294
MESOS1	30	32	30	15	3	67
MESOS2	26	87	83	51	2	182

The work capacity, as shown in table 6, shares a similar trend with velocity, as each team has a different criterion in selecting their planned work. This resulted in the work capacities of different averages across the projects. As the work capacity includes the velocity, it is natural that the work capacity table includes values higher than the ones in the velocity table. The values in the minimum column usually represent the last sprints in a project. Comparing them with the velocity show that assigned and delivered work became significantly less compared to the mean.

Table 7. Descriptive statistics of the percentage of focus factor for the six projects studied.

Focus Factor						
Project	Number of Sprints	Mean	Median	Std. Deviation	Minimum	Maximum
APSTUD	27	0.71	0.72	0.27	0.15	1.00
NEXUS	59	0.73	0.81	0.28	0.10	1.00
TISTUD	53	0.45	0.44	0.25	0.04	0.94
XD	66	0.88	1.00	0.21	0.11	1.00
MESOS1	30	0.50	0.40	0.20	0.10	0.86
MESOS2	26	0.52	0.54	0.23	0.11	1.00

Table 8. Descriptive statistics of the percentage of velocity increase for the six projects studied.

Percentage of Velocity Increase (Hyper-productivity)							
Project	Number of Sprints	Baseline	Mean	Median	Std. Deviation	Minimum	Maximum
APSTUD	27	51	149	80	151	16	569
NEXUS	59	2	337	250	347	25	1650
TISTUD	53	35	242	231	166	9	726
XD	66	7	1361	1150	969	14	4200
MESOS1	30	7	205	179	132	14	514
MESOS2	26	2	2210	1850	1528	50	5050

Tables 7 and 8 consists of the two metrics used in the research questions focus factor and velocity increase. Starting from the focus factor, we can see that the mean varies from 0.45 to 0.88. Three out of the six projects have a mean between 0.7 and 0.9. This shows that half of the projects have a healthy focus factor as an average. Also, three projects have their means and median almost equal. The standard deviation was similar throughout all the projects and showed that the focus factor variation was similar in the six projects. With this similarity, the mean can be considered as the most appropriate factor for identifying healthy focus factors.

As we move to the percentage of velocity increase, the means of the projects vary from 205% to 2210%. This variation becomes more obvious in the remaining columns, such as the median, standard deviation, and maximum. All the values suggest that there exist two types of projects

based on their percentage of velocity increase. Projects with extremely high velocity increase (maximum velocity increase is more than 1000%) and projects with normal velocity increase (maximum velocity increase is less than 1000%). Another important characteristic is the high deviation of projects with extremely high velocity increase. This shows that even though they have a high velocity, it tends to deviate throughout the sprints and is not maintained. Another important point regarding projects with extremely high velocity increase is that their baseline is always low compared to the other projects, this may have an impact on their velocity increase to reach high levels over 1000%.

Table 9. Descriptive statistics of the number of developers for the six projects studied.

Number of Developers						
Project	Total Number of Sprints	Mean	Median	Std. Deviation	Minimum	Maximum
APSTUD	27	12	12	4	4	20
NEXUS	59	10	10	3	2	15
TISTUD	53	19	18	6	11	38
XD	66	13	13	3	3	20
MESOS1	30	13	13	4	7	21
MESOS2	26	26	28	8	7	41

Table 10. Descriptive statistics of the number of new developers for the six projects studied.

Number of New Developers						
Project	Total Number of Sprints	Mean	Median	Std. Deviation	Minimum	Maximum
APSTUD	27	4	3	3	0	16
NEXUS	59	3	2	2	0	12
TISTUD	53	6	5	4	0	18
XD	66	2	2	2	0	8
MESOS1	30	5	4	4	0	15
MESOS2	26	8	7	5	0	17

Table 11. Descriptive statistics of the sprint length (days) for the six projects studied.

Sprint Length (Days)						
Project	Total Number of Sprints	Mean	Median	Std. Deviation	Minimum	Maximum
APSTUD	27	18	13	27	2	153
NEXUS	59	14	14	1	7	14
TISTUD	53	13	13	4	2	30
XD	66	13	12	6	3	28
MESOS1	30	13	14	2	7	14
MESOS2	26	14	14	1	11	15

The contextual factors in this study, the number of developers, number of new developers, and sprint length are shown in tables 9, 10, and 11, respectively. The average number of developers per sprint ranged from 10 to 26 throughout the projects, with a minimum of 2 and a maximum of 41. Since the standard deviation in the different project stayed under ten developers, we can assume that teams had a moderate number of developers (under 20 developers) throughout different sprints except for MESOS1.

When it comes to the number of new developers, the minimum number of new developers ranged from 0 to 18, and the average did not exceed eight new developers per sprint, while the standard deviation was five or fewer new developers. This shows that in most of the sprints, a moderate number of new developers were included in the team.

In table 11, almost all the projects had an average sprint length between 12 and 14 days (except for APSTUD), and this value is also seen in the median. The standard deviation remained minimal with a limit of 6 days (except for APSTUD), despite some odd values in the minimum and maximum columns, we can say that the sprint length remained stable throughout most of the sprints.

Figure 1. The aggregate percentage of the velocity increase of the studied projects.

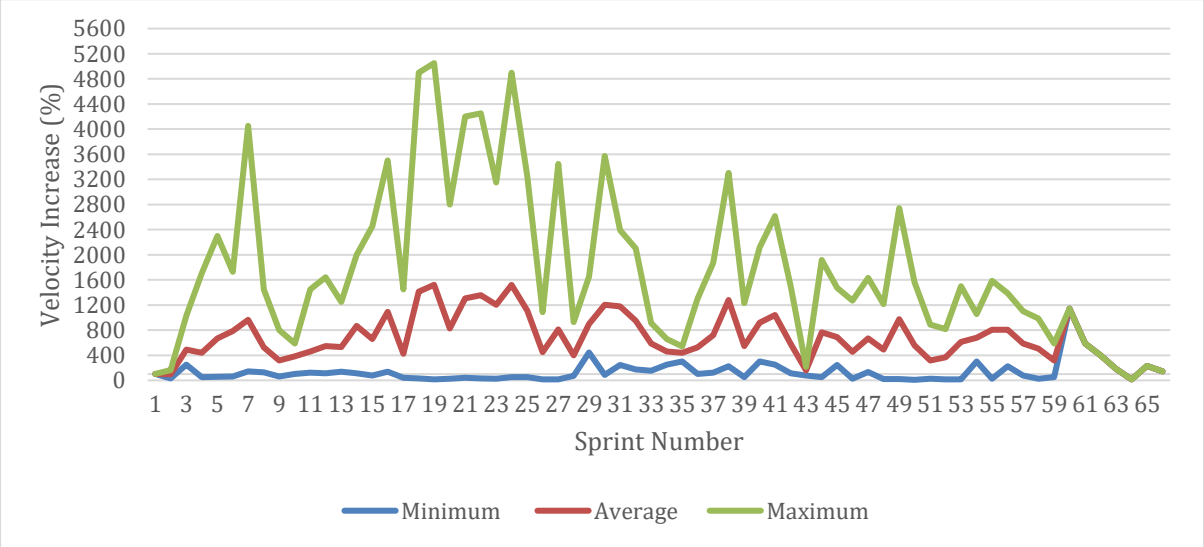

Figure 1 summarizes the percentage of velocity increase across the studied projects by having their values aggregated in one graph. Besides the average value of the percentage of velocity

increase, the minimum and maximum values are also shown. The maximum percentage of velocity increase mainly represents values from XD and MESOS2, which had high values in table 8. This is also contributed to the increase in the average values as the remaining projects did not have an extremely high percentage of velocity increase. Moreover, this figure shows the extreme variation in terms of velocity increase across the projects. This variation makes the hyper-productive state a less favorable productivity level in some projects due to their high values.

Figure 2. The aggregate focus factor of the studied projects.

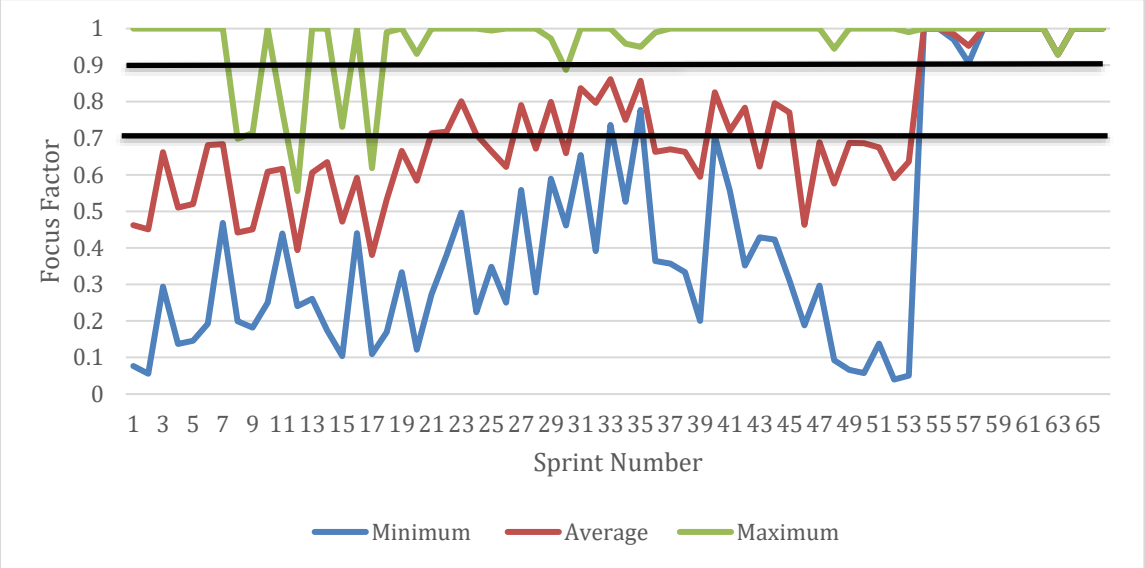

Figure 2 shows the minimum, average, and maximum focus factor values across the studied projects. The area between the two black lines at 0.7 and 0.9 represents the healthy state of focus factor. It is clear that the average focus factor entered the healthy state after almost 20 sprints. This timing is either the last sprints for projects like APSTUD, MESOS1, MESOS2, or the second half of the sprints for the remaining projects.

5.2 RQ1

To answer RQ1 – What are the contextual factors that facilitate a team to reach a hyper-productive state? –, a set of contextual variables are analyzed regarding the hyper-productive states. The motivation behind this analysis is to identify which of these contextual variables influence hyper-productive states.

Each contextual variable (i.e., number of developers, number of new developers, sprint length) is analyzed separately, and three hypotheses are tested to answer RQ1. The following subsections show the results of the hypotheses testing.

The Mann–Whitney U-test was the chosen hypotheses testing method in this research question as the Shapiro-Wilk test indicated deviation from normality in the projects studied.

5.2.1 H1: There is a difference in the number of developers between hyper-productive and non-hyper-productive sprints.

Table 12 classifies each projects' sprints into two groups, hyper-productive (G1) and non-hyper-productive (G2). The means of the sprints in G1 are higher than in G2 except for TISTUD. Also, the same results were obtained when combining all the projects together.

Table 13 shows the results of applying the Mann–Whitney U-test. The results show that 4 out of 6 projects confirmed H1 since their P-value was less than α (α = 0.05). The U-test gave the same positive results when combining the projects in the "All Projects" row. APSTUD was the only project where its low number of sprints in G1 prevented it from having results in the U-test.

Table 12. Statistical results for the "Number of Developers" with respect to Hyper-productivity.

Project	Group	Number of Developers		
		Number of Sprints	Mean	Std. Deviation
APSTUD	G1: Hyper-productive	2	15	0
	G2: Non-hyper-productive	25	11.4	4.5
NEXUS	G1: Hyper-productive	16	11.9	2.4
	G2: Non-hyper-productive	43	9.0	3.3
TISTUD	G1: Hyper-productive	9	18.1	4.0
	G2: Non-hyper-productive	44	18.8	6.0
XD	G1: Hyper-productive	55	13.5	2.2
	G2: Non-hyper-productive	11	8.9	3.3
MESOS1	G1: Hyper-productive	3	18.7	3.2
	G2: Non-hyper-productive	27	12.7	3.3
MESOS2	G1: Hyper-productive	24	27.2	6.8
	G2: Non-hyper-productive	2	7.5	0.7
All Projects	G1: Hyper-productive	109	16.8	6.9
	G2: Non-hyper-productive	152	12.9	5.9

Table 13. Mann–Whitney U-test results of H1 for the six studied projects.

Project	W-value	P-value	Is H1 Confirmed?
APSTUD	∅	∅	Not enough sprints
NEXUS	162	0.002	Yes
TISTUD	194.5	0.943	No
XD	76	< .001	Yes
MESOS1	7	0.022	Yes
MESOS2	0	0.023	Yes
All Projects	5213	< .001	Yes

5.2.2 H2: There is a difference in the number of new developers in hyper-productive and non-hyper-productive sprints.

Table 14 show that the mean of new developers in G1 and G2 varied from project to project. For example, NEXUS and TISTUD had slightly higher means in G2. While other projects had higher means in G1. However, the overall results in the last row of table 10 suggest there is no difference in the means of new developers. Table 15 shows the Mann–Whitney U-test results, which shows that most of the projects failed to confirm H2 except for MESOS1. As a result, the null hypothesis is confirmed in this case, indicating no difference in the number of new developers among G1 and G2.

Table 14. Statistical results for the number of new developers with respect to Hyper-productivity.

Project	Group	Number of New Developers		
		Number of Sprints	Mean	Std. Deviation
APSTUD	G1: Hyper-productive	2	6.0	0.0
	G2: Non-hyper-productive	25	4.0	3.5
NEXUS	G1: Hyper-productive	16	2.4	1.7
	G2: Non-hyper-productive	43	2.7	2.7
TISTUD	G1: Hyper-productive	9	5.9	3.6
	G2: Non-hyper-productive	44	6.4	3.7
XD	G1: Hyper-productive	55	2.3	2.0
	G2: Non-hyper-productive	11	1.5	1.8
MESOS1	G1: Hyper-productive	3	10.7	5.1
	G2: Non-hyper-productive	27	4.1	2.8
MESOS2	G1: Hyper-productive	24	8.0	4.7
	G2: Non-hyper-productive	2	1.5	2.1
All Projects	G1: Hyper-productive	109	4.1	4.0
	G2: Non-hyper-productive	152	4.1	3.5

Table 15. Mann–Whitney U-test results of H2 for the six studied projects.

Project	W-value	P-value	Is H2 Confirmed?
APSTUD	∅	∅	Not enough sprints
NEXUS	333	0.856	No
TISTUD	217.5	0.65	No
XD	227.5	0.191	No
MESOS1	9	0.031	Yes
MESOS2	4.5	0.067	No
All Projects	8604.5	0.592	No

5.2.3 H3: There is a difference in the sprint length of hyper-productive and non-hyper-productive sprints.

The projects' sprint lengths in Table 16 show similar mean values in addition to a standard deviation that indicates little to no variation in the sprint length. This is corroborated by the Mann–Whitney U-test results shown in Table 17 as three projects were disqualified from the U-test and only one project confirming H3.

Table 16. Statistical results for the sprint length with respect to Hyper-productivity.

Project	Group	Sprint Length		
		Number of Sprints	Mean	Std. Deviation
APSTUD	G1: Hyper-productive	2	14.5	2.1
	G2: Non-hyper-productive	25	18.6	28.2
NEXUS	G1: Hyper-productive	16	14.0	0.0
	G2: Non-hyper-productive	43	13.8	1.1
TISTUD	G1: Hyper-productive	9	13.0	0.0
	G2: Non-hyper-productive	44	13.4	3.9
XD	G1: Hyper-productive	55	13.4	5.8
	G2: Non-hyper-productive	11	8.3	3.5
MESOS1	G1: Hyper-productive	3	14.0	0.0
	G2: Non-hyper-productive	27	13.3	1.6
MESOS2	G1: Hyper-productive	24	13.9	0.6
	G2: Non-hyper-productive	2	13.5	2.1
All Projects	G1: Hyper-productive	109	13.6	4.1
	G2: Non-hyper-productive	152	14.0	11.8
	G3: All	261	13.8	9.4

Table 17. Mann–Whitney U-test results of H3 for the six studied projects.

Project	W-value	P-value	Is H3 Confirmed?
APSTUD	14.5	0.172	No
NEXUS	∅	∅	No variation in sprint length
TISTUD	∅	∅	No variation in sprint length
XD	134.5	0.004	Yes
MESOS1	∅	∅	No variation in sprint length
MESOS2	25	0.931	No
All Projects	7174.5	0.052	No

Based on the results obtained, the answer to RQ1 is given as follow:

H1 was the most confirmed hypothesis where the projects showed that hyper-productive sprints need more developers compared to non-hyper-productive sprints. While in H2, most of the projects failed to confirm H2 except for MESOS1, which had more new developers in G1. Logically, this shows that the number of new developers in G1 and G2 does not change, but it may also suggest that hyper-productivity is not affected by the number of new team members. Finally, H3 showed that in case of a variation in sprint length as in XD, hyper-productive sprints would be longer.

5.3 RQ 2

To answer RQ2 – What are the contextual factors that facilitate a team to reach a healthy state? –, the same set of contextual variables are analyzed regarding the healthy focus factor. The motivation behind this analysis is to identify which of these contextual variables influence healthy focus factor states.

Each contextual variable (i.e., number of developers, number of new developers, sprint length) is analyzed separately, and three hypotheses are tested to answer RQ2. The following subsections show the results of the hypotheses testing.

The Mann–Whitney U-test was the chosen hypotheses testing method in this research question as the Shapiro-Wilk test indicated deviation from normality in the projects studied.

5.3.1 H4: There is a difference in the number of developers between healthy and unhealthy sprints.

The mean of the number of developers in Table 18 shows a slight variation between G1 and G2. However, what differs the results of H4 from H1 is that the means in G2 exceed the mean in G1 in some projects as TISTUD and are equal in the case of MESOS1. Despite that, only NEXUS was able to confirm H4 in Table 19. This can be due to the high standard deviation of these projects compared to the slight difference between the means of G1 and G2.

Table 18. Statistical results for the number of developers with respect to the focus factor.

Project	Group	Number of Developers		
		Number of Sprints	Mean	Std. Deviation
APSTUD	G1: Healthy	7	13.1	5.2
	G2: Unhealthy	20	11.2	4.1
NEXUS	G1: Healthy	14	11.3	2.5
	G2: Unhealthy	45	9.3	3.4
TISTUD	G1: Healthy	5	16.0	4.6
	G2: Unhealthy	48	19.0	5.7
XD	G1: Healthy	8	14.1	1.6
	G2: Unhealthy	58	12.5	3.0
MESOS1	G1: Healthy	4	13.3	5.4
	G2: Unhealthy	26	13.3	3.6
MESOS2	G1: Healthy	3	26.0	7.0
	G2: Unhealthy	23	25.7	8.8
All Projects	G1: Healthy	41	14.0	5.2
	G2: Unhealthy	220	14.6	6.9

Table 19. Mann–Whitney U-test results of H4 for the six studied projects.

Project	W-value	P-value	Is H4 Confirmed?
APSTUD	51	0.304	No
NEXUS	198	0.037	Yes
TISTUD	157.5	0.258	No
XD	156	0.133	No
MESOS1	54	0.927	No
MESOS2	35.5	0.968	No
All Projects	4546	0.936	No

5.3.2 H5: There is a difference in the number of new developers between healthy and unhealthy sprints.

The mean values of G1 and G2 in Table 20 have less variation if they are compared to the means in Table 18 in H4. Even though the standard deviation is lower than the values in Table 18, the results of the U-test in Table 21 showed that all the projects rejected H5. This result, combined with the result in H4, shows that having healthy sprints has no significant difference in the number of new developers or the team size as a whole.

Table 20. Statistical results for the "Number of New Developers" with respect to the focus factor.

Project	Group	Number of New Developers		
		Number of Sprints	Mean	Std. Deviation
APSTUD	G1: Healthy	7	4.3	3.8
	G2: Unhealthy	20	4.1	3.4
NEXUS	G1: Healthy	14	3.7	3.1
	G2: Unhealthy	45	2.3	2.2
TISTUD	G1: Healthy	5	6.0	4.4
	G2: Unhealthy	48	6.4	3.7
XD	G1: Healthy	8	2.5	1.8
	G2: Unhealthy	58	2.1	2.0
MESOS1	G1: Healthy	4	5.0	5.1
	G2: Unhealthy	26	4.8	3.4
MESOS2	G1: Healthy	3	5.3	4.2
	G2: Unhealthy	23	7.8	5.0
All Projects	G1: Healthy	41	4.1	3.4
	G2: Unhealthy	220	4.2	3.8

Table 21. Mann–Whitney U-test results of H5 for the six studied projects.

Project	W-value	P-value	Is H5 Confirmed?
APSTUD	68	0.933	No
NEXUS	219	0.084	No
TISTUD	131.5	0.735	No
XD	188	0.383	No
MESOS1	53	0.976	No
MESOS2	46.5	0.354	No
All Projects	4436.5	0.869	No

5.3.3 H6: There is a difference in sprint length between healthy and unhealthy sprints.

The mean values in Table 22 have less variation in the means if they are compared to the values in H3. Adding to this the null standard deviation values, none of the projects confirm H6 in Table 23. This result and follows the previous results in H5 and H4 were the focus factor tends to have no significant differences with the contextual variables in general compared to the hyper-productivity.

Table 22. Statistical results for the "Sprint Length" with respect to focus factor.

Project	Group	Sprint Length		
		Number of Sprints	Mean	Std. Deviation
APSTUD	G1: Healthy	7	15.4	5.2
	G2: Unhealthy	20	19.3	31.6
NEXUS	G1: Healthy	14	14.0	0.0
	G2: Unhealthy	45	13.8	1.0
TISTUD	G1: Healthy	5	13.0	0.0
	G2: Unhealthy	48	13.4	3.7
XD	G1: Healthy	8	13.4	7.0
	G2: Unhealthy	58	12.4	5.6
MESOS1	G1: Healthy	4	13.0	2.0
	G2: Unhealthy	26	13.4	1.5
MESOS2	G1: Healthy	3	14.0	0.0
	G2: Unhealthy	23	13.8	0.8
All Projects	G1: Healthy	41	13.9	3.7
	G2: Unhealthy	220	13.8	10.1

Table 23. Mann–Whitney U-test results of H6 for the six studied projects.

Project	W-value	P-value	Is H6 Confirmed?
APSTUD	49.5	0.103	No
NEXUS	∅	∅	No variation in sprint length
TISTUD	∅	∅	No variation in sprint length
XD	206.5	0.622	No
MESOS1	53.5	0.937	No
MESOS2	∅	∅	No variation in sprint length
All Projects	3715	0.059	No

Based on the results obtained, the answer to RQ2 is given as follow:

In general, the Mann–Whitney U-test results showed that the factors studied have no significant differences in the focus factor. The mean values of G1 and G2 are less diverse when they are sectioned according to the healthy state. Also, the number of sprints in G1 much less. If these results are compared to the ones in RQ1, it will be clear how the means and G1's sprints are way less. However, it is not clear if having projects with a higher number of sprints in G1 will give additional findings regarding the factors studied.

6 Discussion

Table 24 summarizes the results of RQ1 and RQ2. Factors with "Yes" show that there are statically significant differences in the results where 80% of the projects confirm the proposed hypothesis. "No" indicates that there are no statistically significant differences in the results where all the projects fail to confirm the proposed hypothesis. "Non-conclusive" results indicate that the projects confirming the proposed hypothesis are less than 80% of the total number of projects. The results of "All projects" were also included in these criteria.

Table 24 - Summary of results for RQ1 and RQ2

Research question	# developers	# new developers	Sprint length
RQ1: hyper-productive	Yes	Non-conclusive	Non-conclusive
RQ2: Focus factor (healthy)	Non-conclusive	No	No

The main goal of this study is to identify the relevant factors that a team should be aware of in order to either reach a state of hyper-productivity or healthiness. Both hyper-productivity and healthy states are the results of reaching predefined value levels of specific metrics (velocity increase and focus factor), as introduced by Downey and Sutherland [4]. This classification of sprints into states was the primary motivation to formulate the research questions around hyper-productivity and focus factor.

The first research question focuses on hyper-productivity and is sectioned into three hypotheses, each dealing with one contextual variable. The first hypothesis (H1) found significant statistical differences between hyper-productivity and the number of developers, which showed that hyper-productive sprints tend to have more developers compared to non-hyper-productive sprints. This result contradicts the findings of Rodriguez et al. [17] that stated a decrease in productivity when the team size surpasses nine members, which is not the case in the results. Melo et al. have also reported in [30] a decrease in productivity with an increase in team size. Productivity is broad, and metrics fail to capture the full space of developers' productivity [1]. And without specifying the criteria in which productivity was measured in these two studies, it will be inadequate to reach decisive conclusions from this contradiction, and further research on the topic will be required.

The second hypothesis (H2) aims to find statistical differences between hyper-productivity and the number of new developers. The results showed that new developers have fewer significant differences in hyper-productivity compared to the number of developers in H1. A study by Melo et al. [30] described how teams tend to lose focus when members join or leave the team. However, this was not seen when sprints were sectioned according to hyper-productivity. This could be addressed as one possible limitation of hyper-productivity, as the significant differences can only be visible at a team level.

The results of the third hypothesis (H3) showed many significant differences. The fixed length of the sprints makes it difficult to identify any pattern or results except for one project, which its hyper-productive sprints showed an increase in the sprint length. However, with one project, it is not possible to conclude if longer sprint periods increase the overall productivity.

The second research question (RQ2) focused on the focus factor and the healthy state. The hypotheses did not show significant differences when it came to the number of developers compared to the first research question. Similar results were obtained in the number of new

developers were no significant differences were found. This shows that a healthy state should be used as a productivity indicator and not as a way to find external factors influencing the team's productivity.

The last hypothesis results in RQ2 showed no significant results when it came to sprint length. The results could be similar to the one in RQ1. This shows that both the hyper-productivity and the healthy state have no significant results when it comes to the sprint length. Even though some sprints had variations in their lengths, they turned to be insufficient in giving meaningful findings. Thus, the usage of sprint length as a factor should be modified in the following studies.

6.1 Lessons Learnt

Furthermore, the descriptive statistics and the results obtained from running the tests show that there were several issues that didn't allow projects to reach hyper-productive or healthy states, namely:

- **Unstable projects:** The projects studied suffered from instability in different aspects and metrics. For example, metrics such as the work capacity and velocity repeatedly oscillated throughout the project's lifetime. This issue can be related to either the nature of the open-source projects and how they operate, errors in calculating the metrics, or missing data in the original dataset. Moreover, the instability had consequences on the baseline, hyper-productivity state, and the healthy state as well.
- **Baseline velocity problem:** The baseline velocity can be considered as the cornerstone of the hyper-productivity state, which plays a huge role in calculating the velocity increase used in determining whether a sprint is hyper-productive or not. The issue is that the baseline velocity, which is the first velocity of the sprint, was low compared to the latter sprints. This has led projects that have a low baseline and high velocity sprints to reach the hyper-productive state more frequently compared to projects with high baseline velocity.

  The baseline velocity can be considered one of the hyper-productive state flaws. Downey and Sutherland, in their studies [4], [5] did not give any guidance regarding the baseline velocity. In order to have a more realistic baseline velocity, the average of the first two sprints was used as the baseline in this study. Nevertheless, this alternative baseline calculation may not be suitable for all projects. One solution for this issue may be in defining a period, identified by each team, of a limited number of sprints, where the team can adjust their velocity and obtain an average, which may serve as a baseline. However, this problem can only be completely solved with better estimation and usage of story points.

  On the other hand, low baseline velocity can cause an extremely high velocity increase as what occurred to some of the projects in this study, where velocity increase higher than 1000% were obtained, which does not sound reasonable.
- **Levels of hyper-productivity and healthy states:** The levels of hyper-productivity state at 400% of the velocity increase and 0.8 for a healthy focus factor may not be suitable for all the projects. There were no reasons why these levels were chosen by Downey and Sutherland, and further studies should be conducted to find if these levels are the most suitable. Despite that, it may seem that the only logical reason behind this decision is that these values may be hard to reach for some projects, and that was proven in this study. Since some projects either had a low number of hyper-productive or healthy sprints or did not have them at all. The question that arises is whether the hyper-productivity state and healthy state are suitable for measuring and identifying the productivity of different agile projects?

It is still unclear whether the current metrics results are a sign of a problem happening inside the team or if this is due to the state of open source projects. Despite that, one important point which makes the usage of these metrics ambiguous is the duration required for a project to be called hyper-productive or healthy. Till now, the only clarification at hand is that a sprint is required to reach certain values to be called either hyper-productive or healthy, without any additional info regarding the minimum number of sprints or duration. It is considered misleading to classify projects which reached the hyper-productive state for one sprint and other projects which reached the same hyper-productive state for multiple sprints in the same category. Projects that reach the hyper-productive state once, often suffer from an unstable velocity increase due to the rise and drop of the velocity increase. However, this issue is not perceived in sprints that maintain a consistent level of velocity increase. This scenario is similar to what has been found in the projects of this study.

The final question that remains is how the usage of such metrics could benefit different businesses. The only mentioned benefits are the increase in work and faster delivery. Still, there is no mention of the increase in quality, which is the essence of productivity in software development [1]. Should businesses invest in coaches that change a single team, or should these costs be invested in finding universal or common factors (if possible) that can contribute to improving thousands of teams' productivity? As this study has shown that productivity is complex, involving several factors, each is having its own effect on the overall performance, but in the end, if merged together in a proper manner, better control over the general productivity of a team can be possessed.

6.2 Threats to validity

This study has some threats to validity that need to be addressed.

Note: The validity classification and definitions were taken from [17].

- *Construct validity* is to what extent the variables used in the study measure the concepts they are intended to measure. The data used in this study belong to open source projects known to the public. Previous studies such as [22]–[24] have used some of the projects in this study for similar purposes. The main risk remains the variables chosen and how well do they reflect what they suppose to measure.
- *Conclusion validity* is to what extent conclusions can be drawn. One of the main risks which have an impact on the conclusions of this study is the process of data cleaning and filtering. From one side, many issues have been eliminated due to wrong, missed, or repetitive values (sprint names, dates, and keys). And on the other side, the data went to additional filtering and grouping. Adding to these points several assumptions made on the variables which may have an effect on the conclusions drawn
- *External validity* is to what extent the research results could be generalized to the population under study. Even though the data belongs to real open-source projects, the results found in this study cannot represent the whole open-source community due to several reasons. For example, the programming language, tools, and resources used varies from project to project. Several fields in the data show evidence of agile practice; still, it is not known to what extent agile was applied. Since the metrics analyzed in this study were initially tested on private projects, additional research is required to arrive at final conclusions regarding the results achieved in this study.

7 Conclusions and future work

In this study, we have used the data of eight open-source JIRA projects to apply hyper-productive metrics such as focus factor and velocity increase (hyper-productivity) and to identify factors with significant statistical differences. Before and during the calculation of metrics, projects like MULE and TIMOB were excluded from the study, and new subprojects were introduced from DNN and MESOS. Projects' sprints were then divided into two groups per metric, either hyper-productive or non-hyper-productive sprints for hyper-productivity, and either healthy or unhealthy sprints for focus factor.

Different hypotheses were proposed regarding the statistical differences of the contextual factors on both focus factor and hyper-productivity. Hypotheses testing using the Mann–Whitney U-test was used on the means of each project to identify the ones which confirm the proposed hypotheses.

Results showed that the number of developers has a significant statistical difference in the hyper-productivity where hyper-productive sprints had a higher number of developers compared to non-hyper-productive sprints. However, the number of new developers and sprint lengths did not show a huge influence on hyper-productivity except in one project where a higher number of new developers and longer sprint lengths resulted in more hyper-productive sprints. Regarding the focus factor, the hypotheses test results showed that, in general, the contextual factors have no statistical differences in the projects.

This study has also shed light on the limitations found in the metrics which affected the results. Also, it has questioned several points regarding the calculation originally proposed by Downey and Sutherland [4], and the usage of the hyper-productive metrics and whether the idea of hyper-productivity is suitable for different types of projects.

In future work, it will be beneficial to include the rest of the hyper-productive metrics proposed by Downey and Sutherland. In addition to including more open-source projects that may result in a broader range of results. On the other hand, more factors can be added to the study, and a more in-depth statistical analysis can be implemented. It is also possible after acquiring enough results to start building prediction models where we can predict the productivity of a team based on specific variables or inputs.

8 References

[1] C. Sadowski and T. Zimmermann, *Rethinking Productivity in Software Engineering*. Springer, 2019.

[2] D. R. Greening, "Agile Enterprise Metrics," presented at the 2015 48th Hawaii International Conference on System Sciences, 2015, pp. 5038–5044.

[3] E. Kupiainen, M. V. Mäntylä, and J. Itkonen, "Using metrics in Agile and Lean Software Development–A systematic literature review of industrial studies," *Information and Software Technology*, vol. 62, pp. 143–163, 2015.

[4] S. Downey and J. Sutherland, "Scrum metrics for hyperproductive teams: how they fly like fighter aircraft," presented at the 2013 46th Hawaii International Conference on System Sciences, 2013, pp. 4870–4878.

[5] J. Sutherland, S. Downey, and B. Granvik, "Shock therapy: A bootstrap for hyper-productive scrum," presented at the 2009 Agile Conference, 2009, pp. 69–73.

[6] H. Singh, J. Motwani, and A. Kumar, "A review and analysis of the state-of-the-art research on productivity measurement," *Industrial Management & Data Systems*, vol. 100, no. 5, pp. 234–241, 2000.

[7] D. S. Sink and T. C. Tuttle, *Planning and measurement in your organization of the future*. Industrial Engineering And Management, 1989.

[8] M. Broman, "Assessing productivity in assembly systems," *Licentiate thesis, Department of Production Engineering, The Royal Institute of Technology, Stockholm*, 2004.

[9] S. Tangen, "Demystifying productivity and performance," *International Journal of Productivity and performance management*, vol. 54, no. 1, pp. 34–46, 2005.

[10] P. F. Drucker, "Knowledge-worker productivity: The biggest challenge," *California management review*, vol. 41, no. 2, pp. 79–94, 1999.

[11] M. Fowler and J. Highsmith, "The agile manifesto," *Software Development*, vol. 9, no. 8, pp. 28–35, 2001.

[12] A. Ahmed, S. Ahmad, N. Ehsan, E. Mirza, and S. Sarwar, "Agile software development: Impact on productivity and quality," presented at the 2010 IEEE International Conference on Management of Innovation & Technology, 2010, pp. 287–291.

[13] C. D. O. Melo, D. S. Cruzes, F. Kon, and R. Conradi, "Interpretative case studies on agile team productivity and management," *Information and Software Technology*, vol. 55, no. 2, pp. 412–427, 2013.

[14] S. O. Alexander Hars, "Working for free? Motivations for participating in open-source projects," *International Journal of Electronic Commerce*, vol. 6, no. 3, pp. 25–39, 2002.

[15] S. Wagner and M. Ruhe, "A systematic review of productivity factors in software development," *arXiv preprint arXiv:1801.06475*, 2018.

[16] I. Fatema and K. Sakib, "Factors influencing productivity of agile software development teamwork: A qualitative system dynamics approach," presented at the 2017 24th Asia-Pacific Software Engineering Conference (APSEC), 2017, pp. 737–742.

[17] D. Rodríguez, M. Sicilia, E. García, and R. Harrison, “Empirical findings on team size and productivity in software development,” *Journal of Systems and Software*, vol. 85, no. 3, pp. 562–570, 2012.

[18] T. Dingsøyr, T. E. Fægri, T. Dybå, B. Haugset, and Y. Lindsjørn, “Team Performance in Software Development: Research Results versus Agile Principles,” *IEEE Software*, vol. 33, no. 4, pp. 106–110, Jul. 2016, doi: 10.1109/MS.2016.100.

[19] T. Dingsøyr and Y. Lindsjørn, “Team performance in agile development teams: Findings from 18 focus groups,” presented at the International Conference on Agile Software Development, 2013, pp. 46–60.

[20] E. Salas, K. C. Stagl, C. S. Burke, and G. F. Goodwin, “Fostering team effectiveness in organizations: Toward an integrative theoretical framework,” presented at the Nebraska symposium on motivation, 2007, vol. 52, p. 185.

[21] G. Sudhakar, A. Farooq, and S. Patnaik, “Measuring productivity of software development teams,” *Serbian Journal of Management*, vol. 7, no. 1, pp. 65–75, 2012.

[22] E. Scott and D. Pfahl, “Using developers’ features to estimate story points,” presented at the Proceedings of the 2018 International Conference on Software and System Process, 2018, pp. 106–110.

[23] M. Choetkiertikul, H. K. Dam, T. Tran, T. Pham, A. Ghose, and T. Menzies, “A Deep Learning Model for Estimating Story Points,” *IEEE Transactions on Software Engineering*, vol. 45, no. 7, pp. 637–656, Jul. 2019, doi: 10.1109/TSE.2018.2792473.

[24] M. Choetkiertikul, H. K. Dam, T. Tran, and A. Ghose, “Predicting the delay of issues with due dates in software projects,” *Empir Software Eng*, vol. 22, no. 3, pp. 1223–1263, Jun. 2017, doi: 10.1007/s10664-016-9496-7.

[25] A. N. Meyer, T. Fritz, G. C. Murphy, and T. Zimmermann, “Software developers’ perceptions of productivity,” presented at the Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, 2014, pp. 19–29.

[26] K. S. Rubin, *Essential Scrum: A practical guide to the most popular Agile process*. Addison-Wesley, 2012.

[27] M. Agarwal and R. Majumdar, “Tracking scrum projects tools, metrics and myths about agile,” *Int J Emerg Technol Adv Eng*, vol. 2, pp. 97–104, 2012.

[28] W. Hayes, S. Miller, M. A. Lapham, E. Wrubel, and T. Chick, “Agile metrics: Progress monitoring of agile contractors,” CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST, 2014.

[29] H. Kniberg, *Scrum and XP from the Trenches*. Lulu. com, 2015.

[30] C. Melo, D. S. Cruzes, F. Kon, and R. Conradi, “Agile team perceptions of productivity factors,” presented at the 2011 Agile Conference, 2011, pp. 57–66.

[31] B. Boehm and R. Turner, “People factors in software management: lessons from comparing agile and plan-driven methods,” *Crosstalk-The Journal of Defense Software Engineering,(Dec 2003)*, 2003.

[32] A. A. Mohallel and J. M. Bass, “Agile software development practices in Egypt SMEs: a grounded theory investigation,” presented at the International Conference on Social Implications of Computers in Developing Countries, 2019, pp. 355–365.

[33] K. Kaur and A. Jajoo, “Applying agile methodologies in industry projects: Benefits and challenges,” presented at the 2015 International Conference on Computing Communication Control and Automation, 2015, pp. 832–836.

[34] “Hyper-Productive Scrum Metrics,” *Scrum Inc*, 26-Aug-2015. [Online]. Available: https://www.scruminc.com/hyper-productive-metrics/. [Accessed: 12-Mar-2020].

9 Appendix

I. License

Non-exclusive license to reproduce thesis and make thesis public

I, Khaled Nimr Charkie,

1. herewith grant the University of Tartu a free permit (non-exclusive license) to reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Focus Factor and Hyper-Productivity of Agile Teams: A Study of 8 Open-Source Projects,

   supervised by **Ezequiel Scott**

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons license CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive license does not infringe on other persons' intellectual property rights or rights arising from the personal data protection legislation.

Khaled Nimr Charkie

12/03/2020