UNIVERSITY OF TARTU
Institute of Computer Science
Innovation and Technology Management Curriculum

Tedo Gogoladze

The Importance of Personality Traits in Agile Software Development: A Case Study

Master's Thesis (20 ECTS)

Supervisor(s): Ezequiel Scott

Tartu 2020

The Importance of Personality Traits in Agile Software Development: A Case Study

Abstract:

During the last decade, more and more companies have transformed their way of working to agile, making it the most common software development methodology in the industry. The core principle of Agile as given in the original manifesto states, that individuals and their interactions are more important than the tools and processes. This convention leads to the need for the study of the software developer individuals in the psychological level of personality and the performance results associated with the specific personality. There researches that study the personality of the developers and their performances have been set either, in academia, or in the different software development methodology environment, than agile. This study aims to describe the relationship between the software developers' personalities based on the Big Five model and the performance of the developers based on agile software development metrics. Within this research, the logs of 8 open-source projects that use the most common agile issue tracker, JIRA, are used to retrieve the personality trait of the developers involved and to calculate their performance metrics. Finally, association rules are mined using this dataset, and the consistency of the findings are checked against the existing literature. Evidently, the analysis shows interesting relationships between the personality types and metrics, that can favour the work of both, the management of the software development teams and the developers themselves.

Keywords:

Personality, Agile, metrics

CERCS: P170

Isiksuseomaduste olulisus agiilses tarkvaraarenduses: juhtumianalüüs

Lühikokkuvõte:

Viimase kümnendi jooksul on üha enam ettevõtteid võtnud kasutusele agiilse tööviisi, muutes selle tarkvaraarenduse valdkonna levinumaks metoodikaks. Agiilse manifesti põhiprintsiip väidab, et inimesed ja nende tegevused on olulisemad, kui tööriistad ja protsessid. See printsiip tingib vajaduse uurida tarkvaraarendajate isikuomadusi psühholoogilisel tasemel, et selgitada välja seosed töötulemuste ja indiviidi isikuomaduste vahel. Sarnaseid uuringuid on ka tehtud varem, kuid need on keskendunud tarkvaraarendajate isikuomaduste uurimisele akadeemilises või teiste tarkvaraarendus meetodite keskkondades. Antud lõputöö uurib tarkvaraarendajate isikuomaduste ja töötulemuste seost keskkonnas, mis kasutab agiilset töö tegemise meetodit. Lõputöö raames kasutatakse tarkvaraarendajate isiksuseomaduste leidmiseks ja nende jõudlusmõõdikute arvutamiseks, enamlevinud agiilse tööriista Jira logisid, ning lõpuks hinnatakse nende kahe seose vastavust tarkavaraarenduse uuringu reeglitele. Lisaks uuritakse käesolevas lõputöös agiilsete mõõdikute ja isiksuseomaduste seotud uuringuid, et teada saada, kuidas hiljutised uuringud määratlevad sama suhet.

Võtmesõnad:

Isiksuseomaduste, agiil, reeglitele

CERCS: P170

Table of Contents

1 Introduction

Agile software development has already become the most common software development methodology in the industry. Several studies have found out that agile frameworks are the most used in software development due to impressive results of high quality, productivity, and client satisfaction [1]. In addition, the right diversity of personalities within the team has a significant effect on team performance [2]. Personality traits of the developers are not only the subject of study of personal performance and preferences but also for the team results. Having a homogenous personality and mixture of personality types within the team reported higher cohesion and performance in the research experiments [5]. All these reasons make developers' personality traits worth to be studied.

There are different models of personality traits and their use varies according to the context and purpose of the research. The formerly mentioned model used in the research was the Myers-Briggs Type Indicator (MBTI). The model was very popular during the last decades of the last century, but the position has been changed lately. Nowadays, the Big Five Personality Trait is currently the dominant choice of psychologists as the preferred consensual model of assessing personality traits. The Big Five model describes the personality traits in five dimensions: Openness, Conscientiousness, Extraversion, Agreeableness, and Neuroticism. It is also known to have more suitability in assessing the Agile team members personality traits in comparison to other traditional models such as MBTI. Big Five not only provides a better measurement of all the factors measured by MBTI but also it allows to assess Neuroticism, a very important personality factor when working in the teams [6].

Recent studies about personality in software development have investigated the relationship between the Big 5 model and the developers' contribution type [2], that showed that apache project developers with commit access on repositories have the different scores of extraversion and agreeableness, comparing to the developers having no access yet. Additionally, research of Big 5 model in the context of Q&A community has concluded that the users having the same reputation share similar personality traits [7]. However, there is still a lack of studies related to the developers` personality traits in the context of Agile software development and the metrics of agile projects.

To address this issue, this thesis studies performance indicators based on different agile software development metrics that have been used in previous studies [1] and are considered as relevant for agile [9]. Within this research, personality traits of agile software developers are studied in the context of these Agile performance metrics, specifically, Task prioritization, task complexity Estimation, Actual development time, and task state.

In this context, the current thesis aims to answer to following research questions:

RQ1: *What is the relationship between the personality traits of the developers and their performance based on agile software development metrics?*
To answer this question, a dataset of open source JIRA logs of eight agile software development teams is used. The personality traits of the software developers are calculated from the text written by developers using IBM Watson Personality Insights. IBM Watson platform uses the LIWC (Linguistic Inquiry and Word Count) to analyse the textual data and predict the personality trait of the individual who is the author of the former text data. Therefore, the JIRA logs are processed and the performance of the developers based on agile

software development metrics are calculated. Finally, these metrics and the personality traits of the software developers are used to mine association rules and find out the most frequent and interesting patterns between them.

When studying research related to agile software development and personality traits, there are studies that have been done solely on agile, and solely on Big 5 personality traits model. In his paper, Scott et. al [1] has studied agile software development, specifically, SCRUM, within the context of learning styles using another personality model, the Felder–Silverman Learning Style Model (FSLSM). On the other hand, the research of Siddiquei et. al [8] examined and found out the significant correlation between Learning styles using the Felder–Silverman Learning Style Model (FSLSM) and the personality traits of individuals using the Big Five Personality Traits model. Therefore, our second research question is formulated as follows:

RQ 2: *How the relationship between the personality traits of the developers and their performance is related to previous studies?*
For answering this question, a comparison of the results obtained from RQ1 and the results from previous studies were conducted. In particular, the consistency of the association rules found by Scott et al. [1] was analyzed with regards to the correlation of Big 5 and FSLSM models studied by Siddiquei et. al [8]. In their work, Scott et. al [1] have studied several agile performance metrics but using a different model of personality, the Felder-Silverman model, since the research was conducted in an educational context. Furthermore, there is a link between Big 5 and Felder Silverman models defined by Siddiquei et al. [8], that sets up the transformation of Felder-Silverman learning styles to the correlated Big 5 personality traits in the findings of Scott et. al [1] similarly to RQ1 associations, enabling the logical comparison of the rules.

Answering these research questions will be beneficial for the software development industry. Knowledge of the relationships of the personality traits and the specific agile software development metrics is useful for the team and product management staff to analyze, forecast, and handle the performance of the developers and the teams accordingly. They would be able to recognize the certain behaviours in advance and be aware of the possible outcomes of having the developers of the specific personalities in their teams. Furthermore, developers can favour by being aware of the possible positive and negative effects of their personality traits in their work. Considering the fact that the personalities of the developers can change over a short time [2], they can work on their own specific personality traits to improve personal and team productivity. As for researchers, answering the research questions can add more scientific proof and strengthen the research findings of the related work. Having the positive results in a relationship of Big 5 Personality traits and Agile preference metrics can make a solid and fact-based triangular relationship of Felder–Silverman Learning Style Model, Big 5 Personality Traits model and Agile metrics.

This research is organized as follows. In Chapter 2, the background concepts are described. In Chapter 3 the methodology and the technical solutions are defined. The answers to the research questions are presented in Chapter 4. The perspectives and applications of the research results are discussed in Chapter 5. Chapter 6 presents the conclusions of the research.

2 Background

2.1 The Big Five Personality Traits model

The Big Five Personality Traits Model is a five-dimensional personality assessment tool. These five factors are Openness, Conscientiousness, Extraversion, Agreeableness, Neuroticism.

- Openness shows being open to new ideas and experiences, it also indicates how creative they are.
- Conscientiousness indicates the level of goal-orientation, commitment, self-discipline, organization and persistence of the individuals.
- Extraversion denotes the level of gregariousness and sociability of a person, ease of interaction with the others.
- Agreeableness measures degree of trust to the other individuals, cooperation and level of friendliness, adaptive and adjustable to the needs of the others.
- Neuroticism describes the level of negative emotions expression and personal tolerance and to the stress [6].

Personality trait factors and their definer adjectives, scales and sort items, as described by McCrae and John, are given in Table 1 [11].

The claim of five-factor theorists is that these factors, singly or in combination, can be found in virtually all personality instruments [11]. Big Five personality traits model does not imply that the whole of the individuals` personality traits can be divided by these five traits, these traits are more broad definitions and each of these traits summarizes various specific, distinct characteristics of the personality [12].

Factor	**Factor Definers**		
Name	**Adjectives**	**Q-sort items**	**Scales**
Extraversion	Active Assertive Energetic Enthusiastic Outgoing Talkative	Talkative Skilled in play, humor Rapid personal tempo Facially, gesturally expressive Behaves assertively Gregarious	Warmth Gregatiousness Assertiveness Activity Excitement Seeking Positive Emotions
Agreeableness	Appreciative Forgiving Generous Kind Sympathetic	Not critical, sceptical Behaves in a giving way Sympathetic, considerate Arouses liking	Trust Straightforwardness Altruism Compliance Modesty

	Trusting	Warm, compassionate Basically trustful	Tender-Mindedness
Conscientious-ness	Efficient Organized Planful Reliable Responsible Thorough	Dependable, responsi-ble Productive Able to delay gratifica-tion Not self-indulgent Behaves ethically Has high aspiration level	Competence Order Dutifulness Achievement Striv-ing Self-Discipline Deliberation
Neuroticism	Anxious Self-pitying Tense Touchy Unstable Worrying	Thin-skinned Brittle ego defenses Self-defeating Basically anxious Concerned with ade-quacy Fluctuating moods	Anxiety Hostility Depression Self-Consciousness Impulsiveness Vulnerability
Openness	Artistic Curious Imaginative Insightful Original Wide interests	Wide range of interests Introspective Unusual thought pro-cesses Values intellectual matters Judges in unconven-tional terms Aesthetically reactive	Fantasy Aesthetics Feelings Actions Ideas Values

Table 1: Personality trait factors and definers

A recent systematic literature review [6] shows that by the end of the nineties, Myers Briggs Type Indicator (MBTI), a Jungian typology personality assessment model, was the most common approach to measure personality traits, also these researches had set the focus individual personality rather than the team of individuals. The authors of the review conclude that the Big Five Personality Traits model provides better measures for these five factors. Moreover, Big Five includes trait Neuroticism, which is not fully covered by MBTI, while it is a major factor to study in case of a collaborative team of individuals. These make the Big Five model the preferred one by researchers.

2.2 Agile software development

Agile has emerged in recent years as the substituting software development solution of the complex, or plan-driven software development methods. The main focus of Agile is set on customer satisfaction. Goals of Agile are delivered by the continuous process of high-quality software development. Agile values individuals and interactions, working software, customer collaboration and fast responding to the changes [4].

Main characteristics of the agile projects are [13]:

- The software starts with a minimum viable product, goes on with small changes and frequent release cycles, development goes incremental;
- Developers work closely with the customers, both entities have an interest of cooperation and keep close communication;
- The working method is easy to learn and use, straightforward and well-documented;
- The software development process is adaptive, the system is open for the late software changes.

Boehm et. al [15] has compared and analysed process-oriented methodologies vs the agile software development methodology. They have described open-source software as a multi-faceted variant method of Agile. Table 3 shows the comparative analysis according to the author [13]:

Ground Area	**Agile methods**	**Plan-driven methods**
Developers	Agile; Knowledgeable; Collocated; Collaborative;	Plan-oriented; Adequate skills; Access to external knowledge;
Customers	Dedicated; Knowledgeable; Collocated; Collaborative; Representative; Empowered;	Access to knowledge; Collaborative; Representative; Empowered;
Requirements	Largely emergent; Rapid change;	Knowable early; Largely stable;
Architecture	Designed for current;	Designed for current and foreseeable requirements;
Refactoring	Inexpensive	Expensive;
Size	Smaller teams and products	Larger teams and products

Primary objective	Rapid value	High assurance

Table 3: comparison of Agile and Plan-driven methodologies

2.2.1 Agile Metrics

Kupainen et. al [9] studied 30 research papers and 36 case studies in the systematic literature review of Agile Software Development methodologies and their main Metrics. Authors have revealed various domain areas, among those, Information Systems, Telecommunication, web applications were in majority. The results of their research show that SCRUM was the most preferred method of software development, while eXtreme Programming was the second most popular, and Lean and Kanban were following the leaders with less popularity [9].

Researchers have listed the Metrics that were used by the primary works about Agile software development, and they have also listed the ones that were not mentioned in the primary works and the practitioners had to invent them according to the needs [9]. Additionally, the authors have also identified the metrics, that are suggested by agile literature, as a result, Effort estimate, and Velocity are the most popular. Furthermore, quality assurance metrics were also prominent, followed by measures of actual development time, Load factor, Progress tracking, Task Status, Work in Progress and Lead time. Prioritization was the popular metric of controlling the project planning. [9]

The Authors have also grouped five reasons of metric usage and identified metrics for each of these categories. Results yield, that for sprint and project planning, Effort Estimate, Velocity, Task done status are the most important ones, for sprint progress tracking completed work and the number of automated tasks are the main indicators. For understanding and improving quality, work in progress is among the most influential ones. For fixing Software process problems, velocity is one of the major time-related indicators, which in the practitioner`s view can be quite similar to actual development time in this case [9].

2.3 Related work

In the research of Calefato et. al [2], the authors have analysed datasets of Apache Software Foundation projects web pages, project mailing lists and Git repositories. Likewise to our research, their paper has used “Big Five Personality Traits” as the personality traits assessment model of the developers, however, the study was not conducted on Agile software development environment and the focus of the research was set on finding the links of the developers’ personalities with the contributing community team members, the changes in personality traits over the time and hierarchical advancement. According to their results, developers have changed Openness, Agreeableness and Neuroticism over two years with a statistically significant difference. However, there was no difference in the personality traits of the developers before and after becoming the teams’ core member and committing their first git. Moreover, the difference between extraversion and agreeableness traits were found in the core and peripheral developer groups.

Bazeli et. al [7] performed the analysis of personality traits of the software developers on programming community portal - Stackoverflow. Researchers have grouped the community authors into the different categories based on the reputation on the website and analysed questions and answers with regard to the developers’ personality traits according to Big 5

personality traits assessment model. The results yield, that evidently, top authors shared four personality traits out of five, while neuroticism was an exception. Furthermore, the top reputed users have shown to express less neuroticism comparing to the medium and low users. additionally, the researchers have concluded that the authors of the posts with the same tag and authors of the same category posts have similarities in personality traits. Finally, the authors of the down-voted questions and answers were sharing the same personalities and the opposite similarities were found among the authors of the up-voted questions and answers. However, the study was not aiming at the study of software development metrics and there is no specific way of working of the team since it is the online community portal.

Significant analysis and discussion on the subject of software development metrics was presented by Scott et. al [1]. The authors have studied how learning styles can indicate the students` prefered usage of SCRUM. Similarily to our research, Scott et al [1] have performed the analysis of the software development metrics, however, the authors have used the Felder-Silvermann Learning Styles Model (FSLSM) for the assessment of the learning styles and the research was conducted on software engineering students. The results have shown several significant relationships between the students' preferred way of working in SCRUM and their learning styles: intuitive students tend to spend more time on finishing the task, unlike the sensing. Active students are likely to finish the task in a short time and give their tasks the done status, frequently estimating their tasks as high, while the students with the reflective learning style usually keep their tasks on to-do status and estimate their tasks as high. Sequential students found to be keeping their tasks on to-do status, complete them usually in a short time and estimate them as high, however, global students like to move their tasks to done status.

There is an unambiguous relationship between Felder-Silvermann Learning Styles Model (FSLSM) and Big 5 Personality Traits according to Siddiquei et. al [8]. The authors have conducted a questionnaire of the students and analyzed the results using the Pearson product-moment correlation coefficient. Correlation analysis has indicated number of significant relationships, first of all- openness is positively correlated with Active-Reflective learning styles, secondly – conscientiousness is positively correlated with sensing – intuitive learning style, furthermore, agreeableness was found to be positively correlated to active, sensing, visual and sequential learning styles and negatively correlated to the rest of the styles, extraversion was in positive correlation with all the learning styles and finally, there was a negative correlation between neuroticism and all the learning styles.

In a conclusion, some of the related works [2][7] involve the Big 5 personality traits with regard to the reputation and achievements of the developers within the open-source development and Q&A community, whereas the others [1] have studied the software development metrics with respect to Felder-Silverman learning styles of the students and have found the correlations of Felder-Silverman learning styles and Big 5 personality traits [8]. However, the relationship of Big 5 personality traits and software develpoment metrics within Agile environment has not yet been covered by the related works and our research aims to fill this gap.

3 Methodology

In this section, the methodology for answering the research questions are defined, the technical solutions are described, and all the major calculations are presented.

To answer RQ1, the following steps were conducted: obtained the written texts by the software developers from the open-source dataset of software development projects, that required the text cleaning procedures and creating a corpus for each developer. Corpora was used as input for retrieving the big 5 personality traits. Afterwards, metrics were calculated and finally, developers' personality profiles were joined with metrics to create association rules and study the interesting relationships of Big 5 personality traits and metrics, that directly answers the first research question.

To answer RQ2, the association rules found in RQ1 were compared to the association rules from the work of Scott et. al [1]. In order to make the association rules comparable, the correlation between the Big Five model and the FSLSM was used based on the work of Siddiquei et. al [8]. Reliability of the outcome association rules is checked following the same methodology that was conducted for answering RQ1.

3.1 Technical environment

Majority of the research analysis was done in Jupyter notebook using Python and SQL. During the study, we have used the third-party libraries provided by python Anaconda stack, including Pandas, Numpy, Matplotlib, Seaborn, re (regular expression), Collections, Pandasql (SQLite local), Apyori (apriori algorithm for association rules). The datasets and the source code written for the analysis are publicly available in a GitHub repository:

https://github.com/gogoladzetedo/Personalities_In_Agile.

3.2 Datasets

This study analyses data extracted from JIRA issue trackers of 8 open-source projects, that cover the range of scenarios with regards to project domain, number of issues and developer experience The projects are:

- Aptana Studio (APSTUD), a web development IDE;
- Dnn Platform (DNN), a web content management system;
- Apache Mesos (MESOS), a cluster manager;
- Mule (MULE), a lightweight Java-based enterprise service bus and integration platform;
- Sonatype's Nexus (NEXUS), a repository manager for software artefacts required for development;
- Titanium SDK/CLI (TIMOB), an SDK and a Node.js based command-line tool for managing, building, and deploying Titanium projects;
- Appcelerator Studio (TISTUD), an Integrated Development Environment (IDE);
- Spring XD (XD), a unified, distributed, and extensible system for data ingestion, realtime analytics, batch processing, and data export.

The dataset includes data of the developers involved in the projects, the issue reports, the changelog of the issues, and the sprints. Figure 1 describes the entity-relationship (ER) diagram of these datasets and shows attributes of each of them.

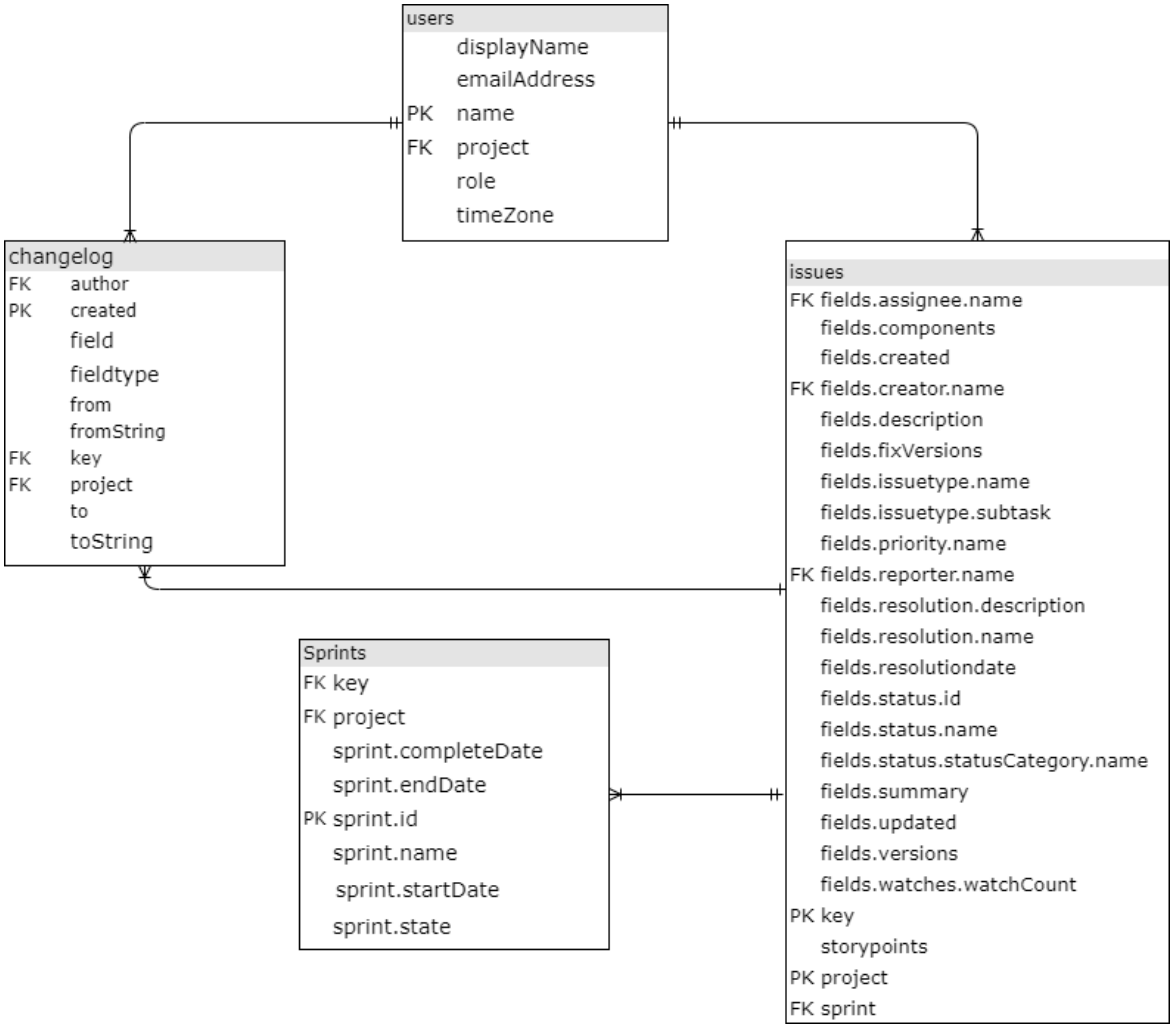

Figure 1. ER diagram of the dataset used.

In total, the dataset consists of 18090 issue reports, out of which there are 15155 unique issues. Each issue report contains references to the sprint in which it was developed, its creator and reporter. A total of 1533 unique users (developers) across all the projects were identified as „creator“ or „reporter“.

In addition, the dataset contains information about every change made to the fields of the issue reports. For example, a change on the status of an issue from „To Do“ to „Done“ is recorded in the changelog along with its timestamp and creator. In total, there are 332690 changes recorded in the original dataset.

3.3 Developers' metrics

In order to characterize the individual performance of the developers, a set of common-used metrics in agile software development were selected. These metrics involved the actual development time, prioritization, effort estimates, and the status of the issues. During the selection of the metrics the following arguments were taken into account:

- Popularity and usage of the metric - selected metrics are popular, commonly used and accepted in the modern agile software development according to the systematic literature review of Agile metrics usage by Kupiainen et al. [9];

- Availability in the current research dataset – these four metrics are retrievable from the open-source dataset used in this research;

- Variety of the metrics – selected metrics can measure various aspects of software development and cover sprint planning (prioritization metric), progress tracking (task status metric), prioritization (prioritization metric), scoping and resourcing (effort estimate metric and actual development time metric) [8];

- Consistency with the related literature – the second research question requires the metrics that were analysed in the related literature. All the selected metrics are used by the authors of the related literature [1], therefore these metrics are applicable for the second research question and consistent with the others.

3.3.1 Actual development time

`Actual development time` denotes the time spent on a specific issue by a developer. Actual development time will catch only the times that was actually being worked, excluding the waiting times.

3.3.2 Prioritization

Prioritization is a type of activities that are commonly studied to monitor and control the project planning [9] – prioritization denotes the urgency of the task from the point of view of the developer. The prioritization values of the issue reports can be easily retrieved from the dataset since they are recorded in a specific field, which makes it worth to be analysed.

3.3.3 Effort estimates

Effort estimation is another important activity of planning related to the scoping and resourcing of an agile software project [9]. In agile projects, effort estimates are given in story points and the number of story points completed during a sprint is known as velocity. Although Velocity is the most widely used metric [9], it does not take into account the single developer's perspective for a given issue.

Since we are interested in having a more detailed understanding on the personal level, the effort estimation of story points by the individual developers was opted as the metric of choice.

3.3.3 Task status

There are progress tracking metrics that are used for monitoring the progress. Depending on the project, the metrics can be divided into project progress or increasing visibility and achieving goals [9].

In this study, the task status was the choice of metric for the individual developers' progress tracking. The task status metric assesses how the developers tend to assign the status to the issue (e.g. close them, put in progress or in to do status list).

3.4 Developers' personality traits

Within this research, IBM Watson Personality Insight was used for getting the developers' personality traits. This approach has been used by related works as well [2]. IBM Watson Personality Insights is the platform that detects the individuals` personality traits based on the writing style of the person. IBM Watson has implemented an API (Application Programming Interface) that extracts the values of each dimension of the Big5 model by analysing written text. IBM Watson uses linguistic analytics to determine the personality characteristics within the Big 5 personality traits model.

Since this study is interested in the developers' personality traits, we have collected every single piece of text in the dataset written by the developers. We have uses the fields which contain information on the issue summary and description, comments, migration and business impact, acceptance criteria and the test plan.

3.5 Data cleaning and pre-processing

3.5.1 Text pre-processing and cleaning

This section describes the pre-processing and cleaning steps performed to make the data suitable for the analysis.

IBM Watson offers the Big 5 personality traits assessment for the input of 600 or more words written by a person. Therefore, we need to get the developers with enough and meaningful texts, where enough refers to the developers having the corpus with 600 or more words, and meaningful refers to the texts that are useful to detect the personality traits. In order to obtain the data of developers with meaningful corpora, it is required to perform the text cleaning procedures, get rid of the system-generated and automatic texts, and create a corpus of the text for each developer. For getting developers with enough texts, the developer corpora must be filtered according to criteria of IBM Watson Personality Insights. The corpus of the developer must contain a minimum of 600 words. Accordingly, the steps performed for getting meaningful and enough developer corpora are described in the following paragraphs.

The first task is to apply text cleaning procedures to get meaningful texts from Jira fields written by the developers.

There are several fields in Jira that are filled by the developers. These fields are: 'summary', 'description', 'Acceptance Criteria', 'Comment', 'Epic Name', 'Out of Scope', 'QA Test Plan', 'Epic/Theme', 'Migration Impact', 'Business Value'. Table 5 shows the number of different written texts in each field. Based on these numbers, we decided to use the following fields: 'summary', 'description', 'Comment', 'Acceptance Criteria', 'Migration Impact', 'QA Test Plan', 'Out of Scope' since they were holding the most written texts. In total, there are 8578 written texts filled in the formerly mentioned fields.

Text Field	**Written texts**
Acceptance Criteria	97

Business Value	1
Comment	522
Epic Name	57
Epic/Theme	414
Migration Impact	6
Out of Scope	10
QA Test Plan	21
description	4227
summary	3695

Table 5. Jira changelog text fields

Additionally, a cleaning step to avoid duplicated text was performed since a JIRA user may submit the change of the textual value into one column multiple times, but it is necessary to include only one edit on each task field for each user. Therefore, the following technique was applied: the calculation logic is defined to take the latest one change of the task field by the user, and the latest one will be calculated based on the `created` timestamp column value.

Considering the fact that this study analyses data from software projects repositories, the text written by developers contain various technical terms, a copy of the code snippet of a certain programming language, error code and syntax, system logs, stack traces, technical commands or other technically formatted texts. For addressing this issue, several text-cleaning activities have been performed to get only the meaningful textual inputs that were handwritten by the developers. The cleaning steps include deleting of the standard code snippets and commands, cleaning the text from special characters and tags, removing the texts with system logs. The full list of cleaning steps is given in Appendix I.

After applying the aforementioned cleaning steps, the texts written by each developer were put together in a single corpus. Then, the corpus represents all the text written by a single developer during the project and it will be used to automatically calculate their personality traits. Furthermore, we introduced a variable number of words in the corpus, in order to meet the criteria of the IBM Watson Personality Insights tool. For having the developer corpora with enough written texts, as stated at the beginning of this paragraph, it is necessary to filter the dataset and keep only the corpora with 600 or more words. Dataset filtered with 600 or more words in the corpora is calculated and the descriptive statistics are shown in Table 6.

	Number of words in corpus
count	100
mean	2568.81
std	3987.884
min	603
25%	942.5
50%	1419.5
75%	2339.5
max	34135

Table 6: Description of number of words in corpus variable from the texts.

Description of this dataset shows, that there are corpora of 100 different developers having enough number of words. Value of the mean number of words (2568.81) is 81% more than the median (1419.5) number of words, which indicates, that there are several corpora with the comparably high number of words than the vast majority of the corpora.

Getting the dataset with the clean written texts corpora and filtering with the given threshold fulfils the goal set at the beginning of the paragraph – having the developers corpora with the meaningful and enough number of words, thus the dataset is ready to retrieve the personality traits of the developers.

3.5.2 Retrieving developers` personality traits

After cleaning and filtering the text in the dataset, a total of 100 unique developers were subject to analysis. Table 8 shows the number of developers per project.

Project	tistud	timob	Dnn	xd	mesos	nexus	apstud	mule
developers	33	30	20	15	12	11	10	5

Table 8. Projects with the number of developers

Notably, the sum of the users in this table is more than the total number users mentioned above (100), because, this table presents unique users per project, meaning that one user could work in multiple projects. As it is shown in Table 9, out of the 100 developers 68 of them work solely on one project, while 26 of them work on two projects and 4 developers are presented on 3 projects.

Number of projects	Number of developers
1	68
2	28
3	4

Table 9. Number of developers on the number of projects.

The corpora of the developers are provided to IBM Watson Personality Insights API in JSON format. The API returns the results in JSON frozensets, which is parsed and stored into the dataset. The resulting JSON file contains an assessment of personality traits with percentile and raw score along with a significance indicator. Percentiles scores are generalized on the whole users` dataset of IBM. It is normalized based on all the users that have been requested to Watson, while raw scores are the plain results based solely on the person`s characteristics. Raw scores are the same as what the personality assessment test would return. Normalized and raw score percentiles are both real numbers ranging from 0 to 1.

The histograms of the personality traits assessment obtained are shown in Figure 4. The first row in the graph represents the personality assessment percentiles of the developers as normalized by IBM Watson, while the second row shows the raw scores. Y-axis shows the number of times that the given personality scored occurred.

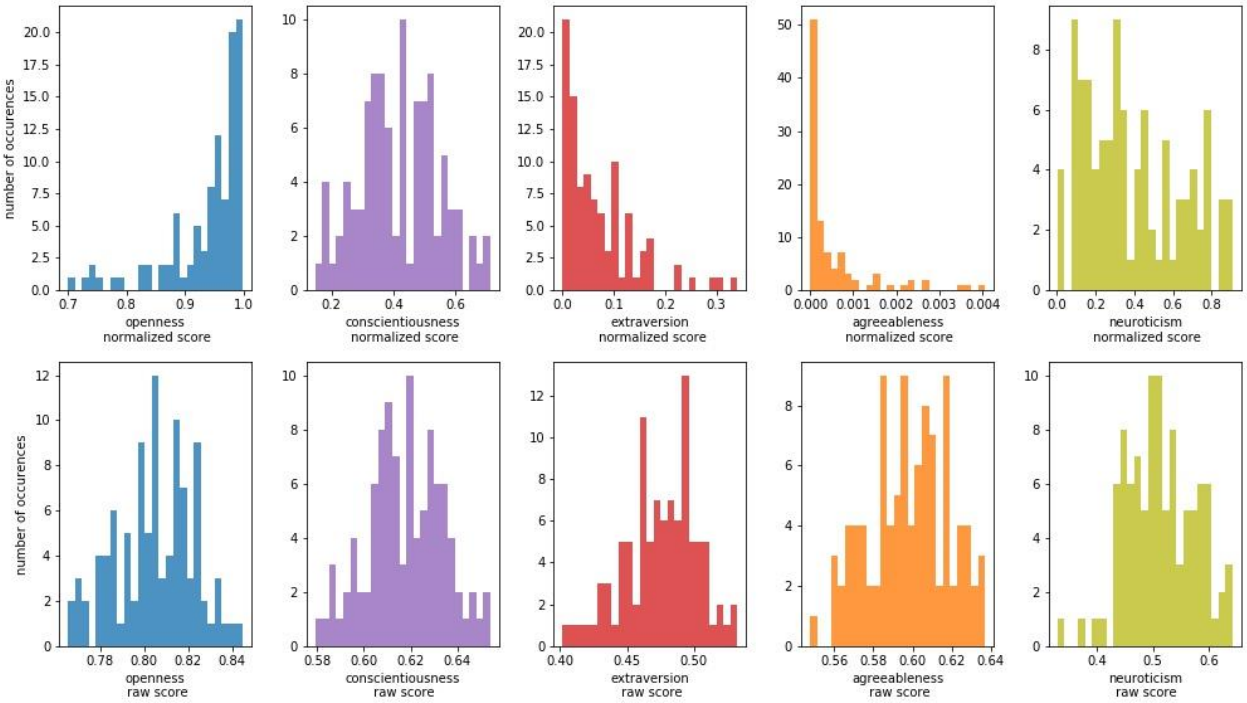

Figure 4: Histograms of developers` personality trait raw scores and percentiles

To compare the result of each trait, first look makes it clear that raw scores are more normally distributed, than the normalized scores.

Openness raw score varies from 0.75 to 0.85, while on the normalized percentile it scores from 0.7 to 1, and notably, the most of the results here are distributed in 0.95 to 1 percentile bin, meaning that the raw scores of the developer`s openness are higher than the sample population scores. This trait is more or less in accordance with the raw and normalized score, similarly to conscientiousness and neuroticism, those have wider percentile range on IBM`s adjusted scores than on raw scores. Extraversion scores are also significantly lower on normalized percentiles, as the developers seem to be less open, than the sample users of IBM. The most distinctive difference between these two scores is returned on agreeableness. The raw score seems reasonable – varying from 0.54 to 0.64, but the normalized scores show extremely low scores - less than the 0.004, meaning that the absolute majority of the IBM sample users have more scores than all the users of this dataset.

Same results summarized within the projects are present in Figure 5. It shows that certain projects users tend to have more polar personalities, than the others, specifically project `mesos` developers are more open and conscientious, project `xd`, `dnn` and `nexus` developers are also conscientious, extravert and agreeable, and finally `timob` and `tistud` project developers share similarities in being less open, conscientious, extravert, agreeable, and more neurotic, than the other project developers. Notably, the last two projects have significantly more users checked for IBM Personality traits than the others, giving more confidence in their results as a group of developers. This can be the reason, why these two projects have similar results in all the personality traits.

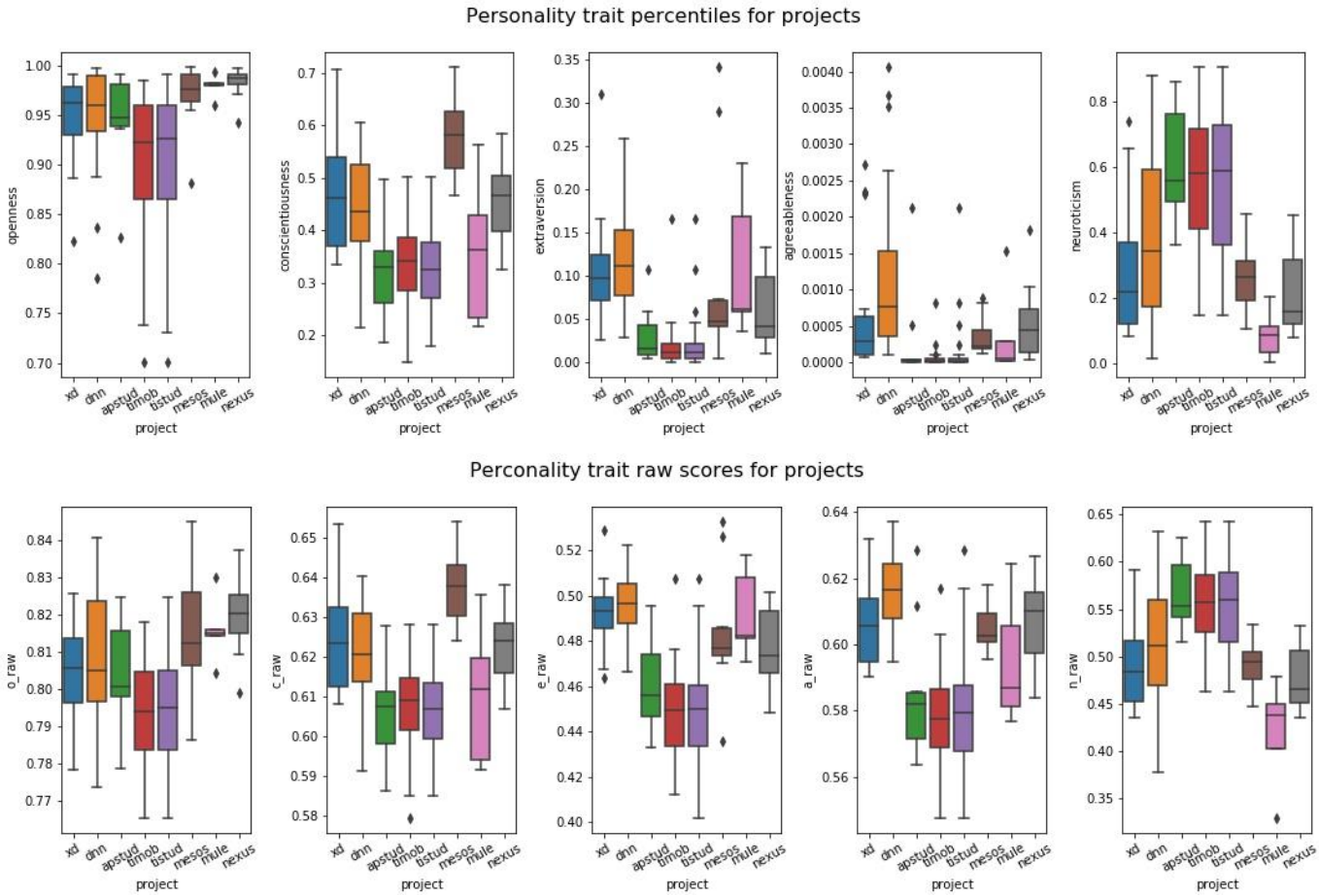

Figure 5. boxplots of users` personality trait raw scores and percentiles per project

Raw scores are used in this study since IBM uses a sample population for normalizing the scores, which may not be representative of a population of developers. In addition, raw are normally distributed, unlike the normalized scores, as shown in Figure 4.

3.5.3 Calculation of metrics

3.5.3.1 Actual development time

For calculating the actual development time, first, we defined the time spent per task by a developer: time that passed while the task was set to 'In Progress' status. In technical terms, this is the time between the two log records when 1) the task status was set to 'In Progress' and 2) task status was changed from 'In Progress'. Therefore, the records were retrieved from Jira changelog, that has the status set from In Progress to any other status, and the records that have set status set to In Progress from any other status.

In case when several developers have worked on one task – calculation occurs on a time for each of them separately. In case when one developer has set the status to 'In Progress' multiple times within one task, the function calculates the sum amount, so that each developer has one number of minutes spent on one task.

The histogram in Figure 6 shows the distribution of actual development time. Noticeably, there are outliers that put the vast majority of the records into the first few bins.

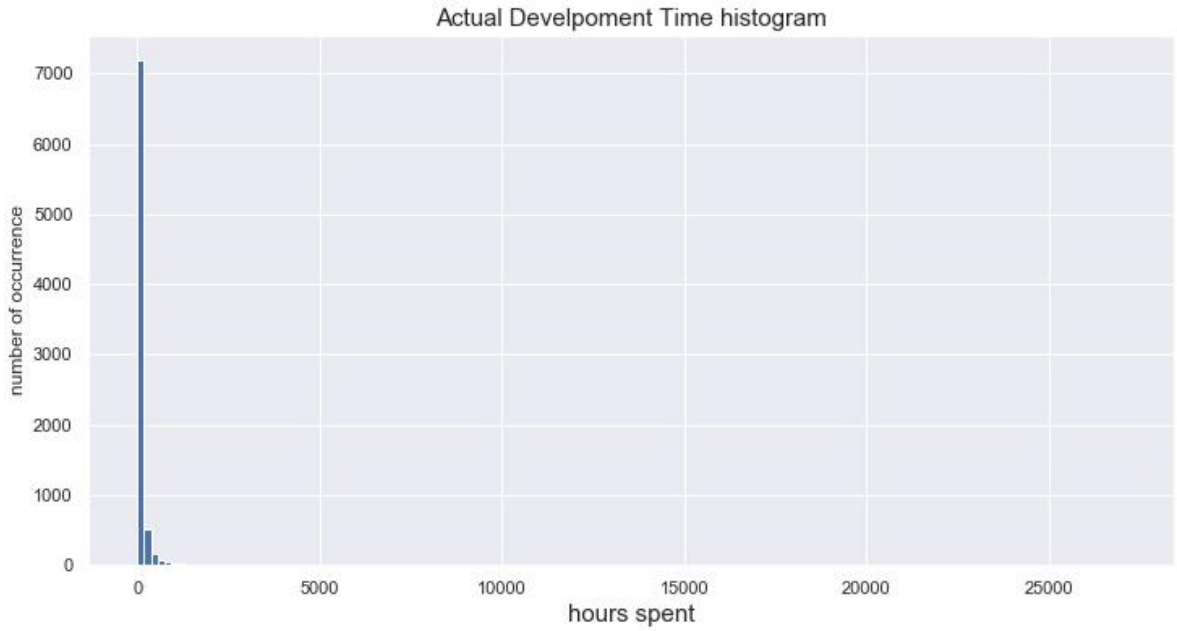

Figure 6. Histogram of actual development time spent by developers

To have a better picture, it is needed to filter out the records with minutes spent greater than 5000 and less than 1, the result is on Figure 7. Now it is visible that although the majority of the records are with the 300 minutes or less, there still are a considerable amount of records with a higher number of minutes spent.

Figure 7. Histogram of actual development time spent by developers on a subset of the dataset.

The end-goal of getting the Actual development time measurement is to have the categorical variables. The top and bottom threshold are defined for high, medium and low development

time spent tasks considering the number of records that would fall into each of these categories, and taking into account the reasonability of each category boundaries:

- Low category of actual development time: tasks, that took 1 working day or less. Equal to 8 working hours, and equal to 480 minutes spent, respectively.
- Medium category of actual development time: tasks, that took more than 1 day and up to 1 week, which is equal to 5 working days, and equal to 2400 minutes spent, respectively.
- High category of actual development time: tasks, that took more than a working week (2400 minutes spent) to complete.

In the sections of metric calculations, we use term transaction that denotes an occurrence of the metric value per one developer on one issue, meaning that for each metric one developer can have one transaction on one jira issue, therefore, in total, the developer can have up to as many transactions, as the number of jira issues he/she has worked on. The whole dataset of actual development time metric contains 8093 transactions in total, whereas the low actual development time category is 4159 transactions (52%), medium actual development time category holds 1418 transactions (17%) and high actual development time category contains 2516 transactions (31%), as presented in Table 9.

Actual development time category	high	low	medium
Number of transactions	2516	4159	1418

Table 9. Actual development time transactions.

The rest of the developers' metrics used in this study (i.e. state, estimation and prioritization) are categorical, and the identical approaches are used for working on all these metrics.

3.5.3.2 Task status

To identify whether the developers tend to put the tasks on hold, or gather in to-do list, or mark them done, the field `status` of the changelog was used. This field can take values from 28 possible statuses. To facilitate the analysis, the possible values were grouped into 3 categories that represents the most common statuses of tasks during agile software development [1]

- 'todo' state values: 'To Do', 'Open', 'Reopened', 'Reviewable', 'To Be Merged', 'Scoped', 'Refine', 'New', 'Raw', 'Waiting for Response', 'To Be Tested', 'Pending 3rd-Party', 'Defered', 'Triaged';
- 'in progress' state values: 'Pull Request Submitted', 'Planned Development', 'In Progress', 'In PR', 'In Review', 'In Review', 'Writing', 'Waiting for Review', 'Testing In Progress';
- 'Done' state values: 'Closed', 'Resolved', 'Done', 'Inactive - Pending Closure', 'Accepted'.

As a result, in total we get 21585 transactions of status metric, out of which majority (17073) is grouped in the category `done`, 2315 transactions are `in progress`, and 2197 transactions are grouped into `todo` status, as shown on Table 11.

Tas status	done	in progress	to-do
number of transactions	17073	2315	2197

Table 11. Task status transactions

3.5.3.3 Prioritization

Priority metric is used to define how the developers are prioritizing their tasks, whether they assign low, medium or high priority.

The changelog dataset is filtered with ‘priority’ field and checked what are the values that developers are assigning to their tasks. Table 12 shows the priority values within the Jira changelog and the number of transactions that have assigned the given priority.

Priority status	High	Medium	Critical	Major	Low	Blocker	Minor	None	Trivial	To be reviewed
number of transactions	1978	1267	1031	742	670	254	240	166	89	3

Table 12. Jira issues priorities

These statuses were grouped into high, medium and low prioritization categories to facilitate the analysis, which is quite intuitive considering by the values in the field: 'High', 'Critical', 'Blocker' tasks were ranked as `high` priority, 'Medium', and 'Major' values were ranked as ‘medium’ priority, and 'Low', 'Minor', 'None', 'Trivial', 'To be reviewed' were categorized under `low` priority, respectively.

The resulting dataset contains 4796 transactions in total, where the low priority transactions are 1412, medium priority transactions are 865 and high priority transactions are 2518, as shown in table 13.

Prioritization	high	low	medium
number of transactions	2518	1412	865

Table 13. Prioritization transactions.

3.5.3.4 Effort estimates

Effort estimation metric is used to show the tendency of developers to mark the complexity of the task by assigning the respective number of Story Points.

In general, story points are based on an adapted version of the Fibonacci sequence: 0, 0.5, 1, 2, 3, 5, 8, 13, 20, 40, 100. In majority of the cases this rule was thoroughly followed, with slight deviation – some issues have assigned the different story point numbers than that, but there are few of such cases. For this reason, the story points that are represented by different numbers than the ones in the adapted Fibonacci sequence were discarded. After this cleaning step, the resulting 6840 transactions were discretized in 3 categories as follows:

- low effort estimation: values 0, 0.5, 1, and 2;
- medium effort estimation: values 3 and5;
- high effort estimation: values 8, 13, 20, 40, and 100.

Table 14 shows the representation of the number of transactions of each effort estimation category.

Effort Estimation	high	low	medium
number of transactions	1188	2872	2780

Table 14. Effort Estimation transactions

3.5.3.5 Cleaned Dataset

The first step towards the creation of the rules is mapping the personality trait scores to the binary variables – weather the particular personality score yields a positive or negative result. The personality trait assessment results are mapped as Positive when the particular developers' raw score is greater than the mean raw score of all the retrieved developers. Accordingly, the personality trait assessment result is mapped as negative when the raw score is less or equal than the mean raw score of all the samples in the dataset of retrieved personality scores of developers.

In overall, out of the 100 analysed developer profiles have the personality trait results as shown in Table 16.

	agreeableness	conscientiousness	extraversion	neuroticism	openness
Negative	50	48	47	54	50
Positive	50	52	53	46	50

Table 16. The number of developers falling into each category of personality traits.

Following the calculations, we get the dataset of metrics, as described in Table 17. In total, there are 41314 transactions. Later, personality trait assessment results are joined and the resulting dataset has the columns: user, issue name, project name and metric, openness, conscientiousness, extraversion, agreeableness, neuroticism. The personality trait variables have only Positive or Negative as the value. The metric variable has all the possible metric values.

Additionally, in order to have the association rules in the context of each personality trait, following steps are performed: Dataset is split horizontally for each personality trait and subsequently gives one dataset for openness, one for conscientiousness, one for extraversion, one dataset for agreeableness and finally, one for neuroticism. These datasets also have 41314 transactions in total and they consist of the following variables: user, issue name, project name, metric and personality trait.

dataset of metric:	Number of transactions	Number of unique users	Number of unique issues
Effort Estimate	6840	76	6677
Prioritization	4796	81	4464
Task status	21585	97	12283
Actual development time	8093	80	7723

Table 17. Available number of transactions of each metric

3.6 Association rule mining

Next step towards the answering of the RQ1 is conducting association rules. Association rule mining is a technique used in machine learning to discover interesting and reliable relation patterns in the large datasets. Association rule consists of two parts: antecedent – an item belonging the dataset, and consequent – an item that belongs to the dataset in the combination with the antecedent item.

In the context of this research, association rules of form X => Y, where X is a personality trait and Y is a software development metric. Since there are 4 agile metrics studied within this research and each of them is three dimensional, it gives a total of 12 metric values. There are 2 dimensions for each personality trait – Positive and Negative, therefore, there are in total, 24 possible combinations of values. Table 18 gives an example of the case of Neuroticism.

Rule Id	Neuroticism	metric	metric value
1	Negative	Effort estimation	high
2			low
3			medium
4		Prioritization	high
5			low
6			**medium**
7		Task status	done
8			In progress
9			todo
10			high
11			low

12		Actual development time	medium
13	Positive	Effort estimation	high
14			low
15			medium
16		Prioritization	high
17			low
18			medium
19		Task status	done
20			In progress
21			todo
22		Actual development time	high
23			low
24			medium

Table 18: Possible association rule values between the metrics and Neuroticism personality trait.

In this research, three measures of interest are used for finding out high-quality association rules between the metrics and the personality traits: Support, Confidence, and Lift. Support is used to measure how frequent the itemset appears in the dataset, and the confidence measures the conditional probability of the occurrence of consequent given the antecedent. Lift values are also used to compare the rule confidence with the expected confidence.

To define important relationships and associations, first, it`s needed to define minimum support. By the definition, support of an itemset is the ratio of transactions where the given itemset exists: support (X => Y) = support (X $\cup$ Y).

In the dataset of one particular personality traits association rules, there are 2 variable items from personality traits (positive and negative) and 12 items from the metrics (4 metrics, and 3 values for each metric). That gives in total 24 rules by 2 personality traits items as antecedent associated with 12 different metric value items as the consequent, and 24 inverse rules - 12 metric value items as the antecedent associated with 2 values of personality traits as the consequent. Considering all these rules information, one association rules dataset was created. The support for one of these rules on average should be one rules portion in the whole rules, 1/24, which is equal ~ 0.04.

The average itemset support value can be used as the threshold for the filtering of the frequent itemsets. All the itemsets, that have higher support than the average support value (0.04) will be labelled as the frequent itemset and be used in the analysis of the personality trait/metric relationship.

Additionally, two more variables are added into the rule parameters - confidence value can be used as a measurement of reliability of the rule, and the lift value for the confidence comparison to the expected confidence of the rule.

Moreover, all the itemsets contain only 1 item, therefore, the support value of a rule and it`s inversive rule will always be the same. However, the confidence variable is different within the inversive rules: On one hand, the rules with metric as an antecedent, for each metric value, there are only 2 possible personality trait consequents, and in total, these two give confidence 1 as the sum. For example, within the neuroticism rules, antecedent `Effort estimation low` has two possible consequents – 'Positive neuroticism` and `Negative Neuroticism`:

- 1. Confidence (`Effort estimation low`=>`Neuroticism Positive`) = Support (`Effort estimation low` ∪ `Neuroticism Positive`) / Support (`Effort estimation low`)
- 2. Confidence (`Effort estimation low`=>`Neuroticism negative`) = Support (`Effort estimation low` ∪ `Neuroticism negative`) / Support (`Effort estimation low`)

Naturally, the sum of these two confidence values give 1 as a result. Hence, the rule with more than 0.5 confidence, in this case, can be trusted more than the other one.

On the other hand, the rules with Personality trait as antecedent, each value of the personality trait (positive/negative) confidences give 1 in total. For example, within the neuroticism rule, antecedent `Neuroticism Positive` has 12 possible consequents - `Effort estimation` low/medium/high, Prioritization low/medium/high, status todo/inprogress/done, development time low/medium/high:

- Confidence (` Positive Neuroticism` => `Effort estimation low`) = Support(`Positive Neuroticism` U `Effort estimation low`) / Support (`Positive Neuroticism)
- Confidence (` Positive Neuroticism` => `Effort estimation medium`) = Support(`Positive Neuroticism` U `Effort estimation medium`) / Support (`Positive Neuroticism)
- Confidence (` Positive Neuroticism` => `Effort estimation high`) = Support(`Positive Neuroticism` U `Effort estimation high`) / Support (`Positive Neuroticism)

And the same way for all the 12 metric values. Evidently, all these 12 rules confidence values give 1 as the sum, and therefore, the mean confidence value for each of these twelve rules is 1/12 = 0.08.

The above-mentioned logic can be used to filter more reliable rules – in the case of the first example when the antecedent is a metric, the reliable rule should have the confidence greater than 0.5, and in the other case, when the antecedent is a personality trait, the reliable rule should have confidence greater than 0.08.

The second research question aims to analyse how the Big 5 personality traits in the context of agile software development are related to previous studies.

In a previous study, Scott et al. [1] found evidence of the relationship between several software development metrics and another personality trait model, the FSLSM. The main approach used for discovering these relationships was association rule mining, and the main results are summarized in Table 20.

antecedent variable	antecedent value	consequent variable	consequent value
priority	high	Perception	sensing
priority	low	Perception	intuitive
time	low	Perception	intuitive

time	high	Perception	sensing
status	done	Perception	intuitive
Perception	intuitive	time	low
Perception	intuitive	Effort estimation	high
Effort estimation	high	Perception	intuitive
Perception	sensing	time	high
Perception	sensing	Effort estimation	high
Effort estimation	high	Processing	active
time	low	Processing	active
priority	low	Processing	active
Processing	active	time	low
Processing	active	Effort estimation	high
status	done	Processing	active
status	todo	Processing	active
status	todo	Processing	reflexive
Processing	reflexive	status	todo
Processing	reflexive	Effort estimation	high
status	done	Understanding	global
status	todo	Understanding	sequential
time	low	Understanding	sequential
Understanding	global	status	done
Understanding	global	Effort estimation	high
priority	low	Understanding	sequential
Understanding	sequential	status	todo
Understanding	sequential	time	low
Understanding	sequential	Effort estimation	high
priority	high	Understanding	sequential

Table 20. Relevant association rules between software development metrics and the FSLSM (adapted from Scott et. Al [1]).

In a different study, Siddiquei et. Al [8] found that the FSLSM and the Big 5 model are related. The authors studied the correlation between these two models and found several correlations, which are shown in Table 21.

Variable	Value	Trait	Correla-tion	Correlation value
Perception	intuitive	Agreeableness	Negative	-0.268

Perception	intuitive	Conscientious-ness	Positive	0.247
Perception	sensing	Agreeableness	Positive	0.261
Perception	sensing	Conscientious-ness	Positive	0.239
Processing	active	Extraversion	Positive	0.228
Processing	active	Openness	Positive	0.234
Processing	reflexive	Extraversion	Positive	0.236
Processing	reflexive	Openness	Negative	-0.243
Understanding	sequen-tial	Neuroticism	Negative	-0.199
Understanding	global	Neuroticism	Positive	0.199

Table 21. Correlated FSLSM and Big 5 personality traits

Based on these two studies, a comparison between the results obtained from RQ1 and the findings from Scott et. al [1] was conducted. In order to make the results comparable, the relationship between the FSLSM and the Big5 model is given by the correlations found by Siddiquei et. al [8]. Therefore, an association rule is consistent with the previous study of it is a logical consequence of the premises given by the previous studies. More formally, the rule $x \rightarrow z$ is consistent with previous studies if it is entailed by the premises $x \rightarrow y$ (a rule taken from the work of Scott et al. [8]) and $y \leftrightarrow z$ (a correlation taken from the work of Siddiquei et al.). (See eq. 1)

Eq. 1 : $x \rightarrow y, y \leftrightarrow z \models x \rightarrow z$

Within the RQ2 validation, the same association rules were used, as discussed for the RQ1. The only difference is, that here, for the proper data representation, minimum frequency threshold value has been decreased, due to the fact that more than half of the dataset records are related to only one specific metric (status). A minimum value of support and confidence have been altered respectively, and then performed the same association rules mining techniques as in the previous chapters. As a result, the association rules function checks the association rules that were generated respectively from the original rules of Scott et. al [1] and Siddiquei et. al [8].

During the association rules mining, a package `apyori` was used to analyze interesting patterns. apyori is an open-source distributed package for Python programming language, that uses apriori algorithm and provides API and command-line interface. Package description is available on the Python Package Index website: https://pypi.org/project/apyori/.

apyori package provides the function `apriori` that requires items, minimum support and confidence as the input, and provides the list of rules on the base of the common apriori algorithm functionality. In a frequent pattern mining, apriori algorithm uses the fact that any subset of a frequent itemset is also frequent, and that way, algorithm excludes the itemsets, whose support is less than the minimum support, and therefore, it excludes all of their supersets as well. Apriori algorithm returns frozenset results formatted as JSON, then JSON output is parsed and stored the antecedent, consequent, support, confidence and lift variables into the working data frame.

4 Results

The following sections in this chapter show the results organized by research question.

4.1 RQ1

What is the relationship between the personality traits of the developers and their performance based on agile software development metrics?

The following sections describe the relationship of the available 100 developers personality traits and four metrics in the context of all big five personality traits separately, therefore, the following five chapters answer the RQ1 in the context of openness, conscientiousness, extraversion, agreeableness, neuroticism separately.

The association rule graphs in the next sections show only the frequent itemsets` association rules with support, confidence and lift in the context of each personality trait, to show only the reliable relationship between the variables of RQ1. The filter criteria of the rule significance are applied as described in the former chapter.

4.1.1 Openness

Figure 9 shows the association rules of the Openness personality trait. Open developers usually give low effort estimation to their tasks and mark the tasks done (openness:Positive => effort estimation:low, openness:Positive => status:done), while the less open developers tend to spend low time on actual development and also mark the task status done (openness:Negative => development time: low, openness:Negative=> status:done). The rules show that open developers tend to take the tasks that are less complex, also this pair has the significant evidence on the inversive as well (effort estimation:low => openness:Positive), furthermore, the confidence of both, nominal and inversive rule is high enough to consider it reliable, and finally, the lift value is positive and high, meaning that low effort estimation and openness trait are positively correlated. But on the contrary, less open developers are the ones that actually spend less time for the development on the tasks, and similarly to the former one, the respective inverse rule (openness:Negative => development time: low) also has the sufficient support and confidence for the trustworthiness of the rule. Notably, here the lift value is very close to 1, yielding the independence of the former two variables.

Both of these developer subgroups – open and not open, fall into the main class of developers that finish the tasks with the done status, however, among the rules with done status as the antecedent, non-open developers are the ones that have sufficient confidence to be on the consequent side (status:done => openness:Negative).

Among the other significant relationships, the notable is rule priority:high=>openness:Negative, meaning that developers that set the task priority to high, are less open people.

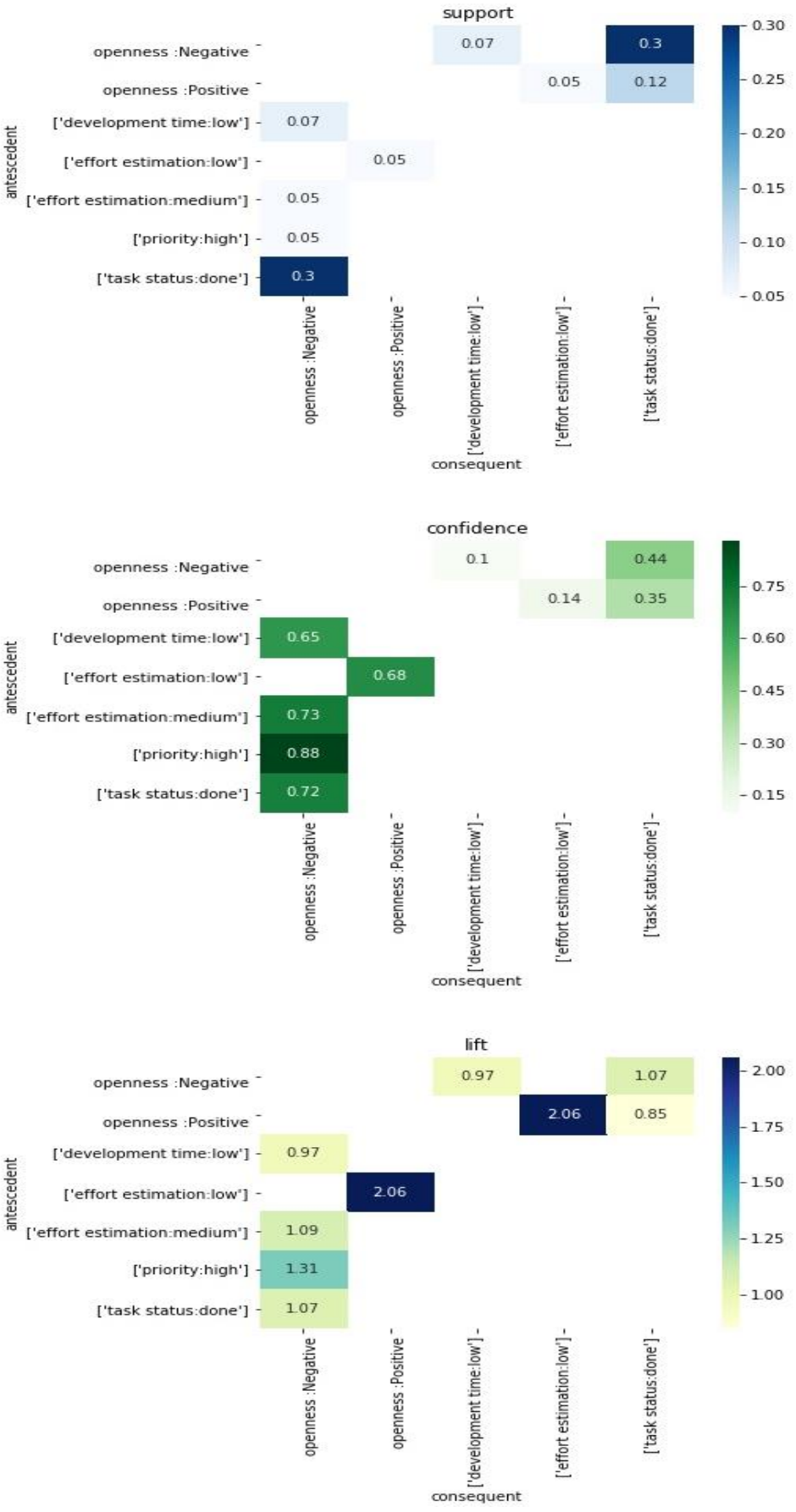

Figure 9: Openness personality trait and related assocaition rules.

4.1.2 Conscientiousness

Association rules related to Conscientiousness is shown in Figure 10. Likewise the openness trait, on one hand, conscientiousness is also associated with low effort estimation and done task status (conscientiousness:Positive => effort estimation:low, conscientiousness:Positive => status:done), additionally, a positive value of this personality trait has a significant association with low development time. These relations show, that more conscientious developers assign the low number of story points to the tasks, spent low time on them, and tend to mark them done at the end, that is quite a valid relation considering the definition of the conscientiousness. The inversive rule of the above-mentioned conscientiousness:Positive => effort estimation:low has also satisfactory support and confidence to be considered as the frequent and reliable, moreover, the lift value shows that there is a positive correlation between the former two variables, while the conscientiousness:Positive => development time:low has not enough confirmation to be considered as reliable.

On the other hand, less conscientious developers also set high priority, spend low time and mark done status on the development of the tasks, like the negatively valued variable of former personality trait. All these three rules have a sufficiently supported inversive rule as well.

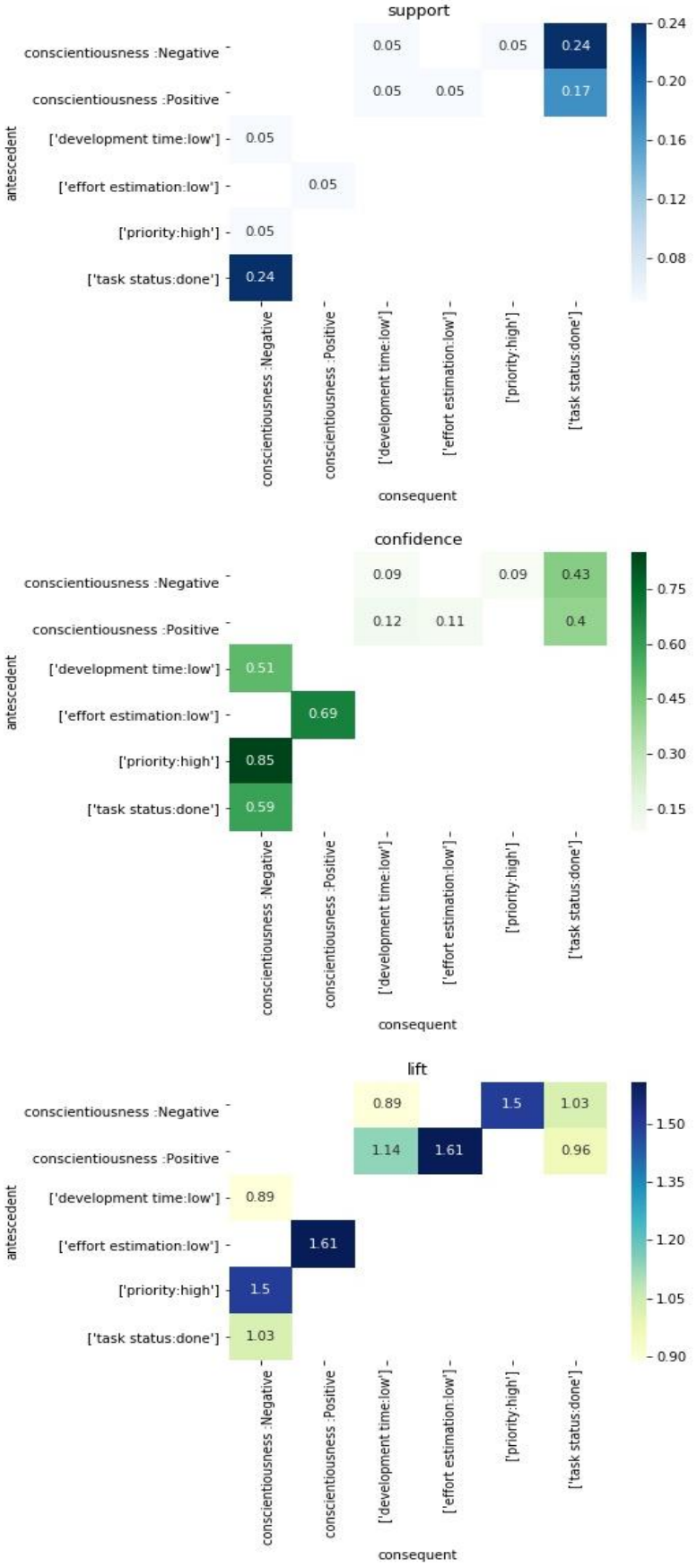

Figure 10: Conscientiousness personality trait and related assocaition rules.

4.1.3 Extraversion

As the figure 11 shows, the association rules of Extraversion personality trait are very similar to the ones within Conscientiousness trait – positive valued variable of the trait Extraversion is associated with low effort estimation, low development time and done task status, while the negative value of the Extraversion trait is related to high prioritization, low development time and done task status. The support, confidence and lift values are also similar to the ones from Conscientiousness personality trait (no more than 0.01 difference in support, no more than 0.05 difference in confidence and lift).

Apart from the similarities, there is one rule regarding the extraversion that is not present within conscientiousness trait – effort estimation:medium => extraversion:Negative, meaning that among the developers that estimate efforts on task as a medium, mainly there are introversive people. This could also mean that introversive developers are trying to keep the balance and choosing to give their tasks the medium-range story points.

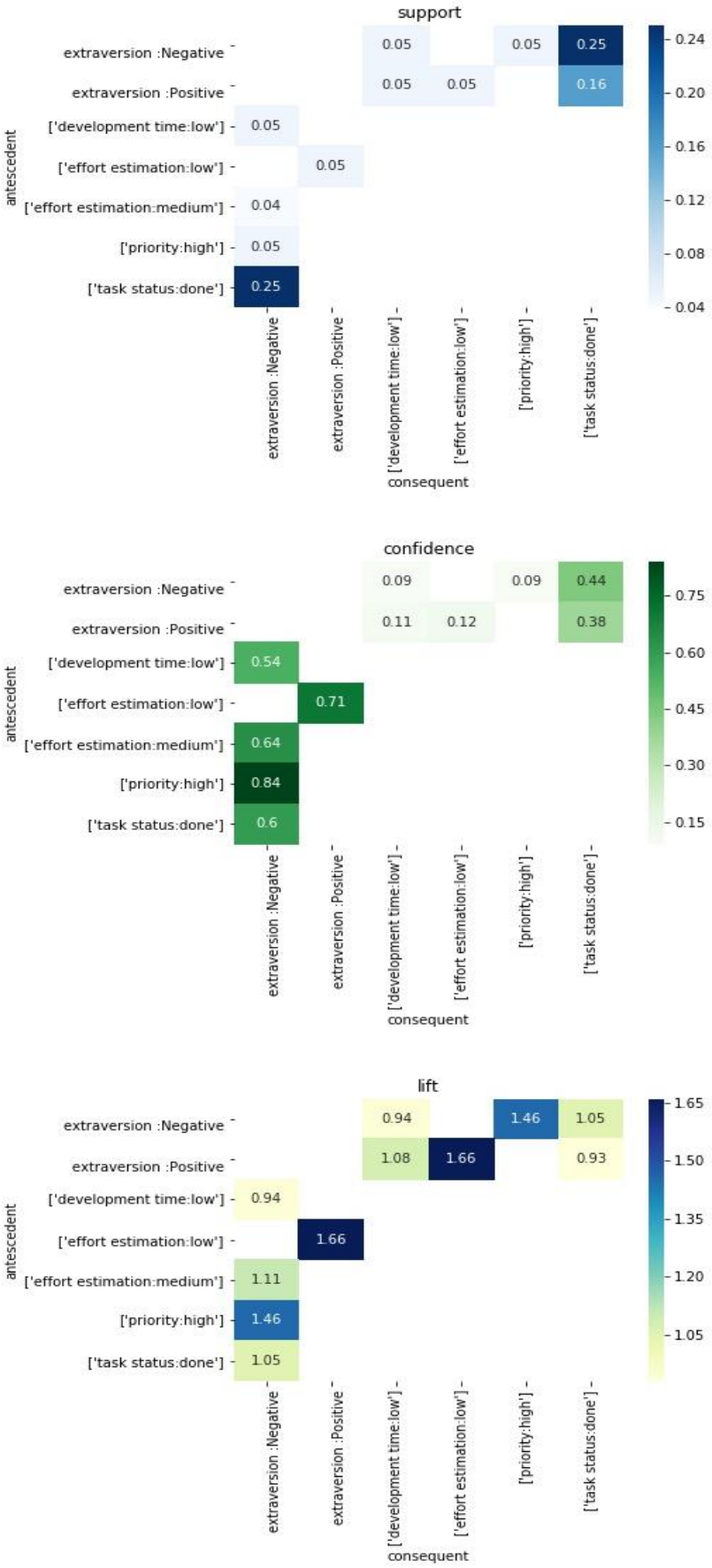

Figure 11: Extraversion personality trait and related assocaition rules.

4.1.4 Agreeableness

Figure 12 shows the association rules related to an agreeableness personality trait. Agreeable developers tend to set the low story points to their issues (agreeableness:Positive => effort estimation: low) and mark them done afterwards (agreeableness:Positive => status:done). The first mentioned rule has also significant inversive rule, while the lift value is high enough to consider these two variables as positively correlated.

Non-agreeable developers, likewise in the formerly described personality traits, also tend to set high priority, spend low time and mark the task as done. All three respective inverse rule (priority:high => agreeableness:Negative, development time:low => agreeableness:Negative, status:done => agreeableness:Negative) are also significant, moreover there is a notably high confidence of the rule priority:high => agreeableness:Negative, that is entirely understandable point since the developers that consider their tasks as the most important, would not be among the most altruistic and modest ones, and these are two of the major facets of agreeableness. The other two association rules have lift value narrowly close to 1, that implies the statistical independence of the two variables in each of these rules.

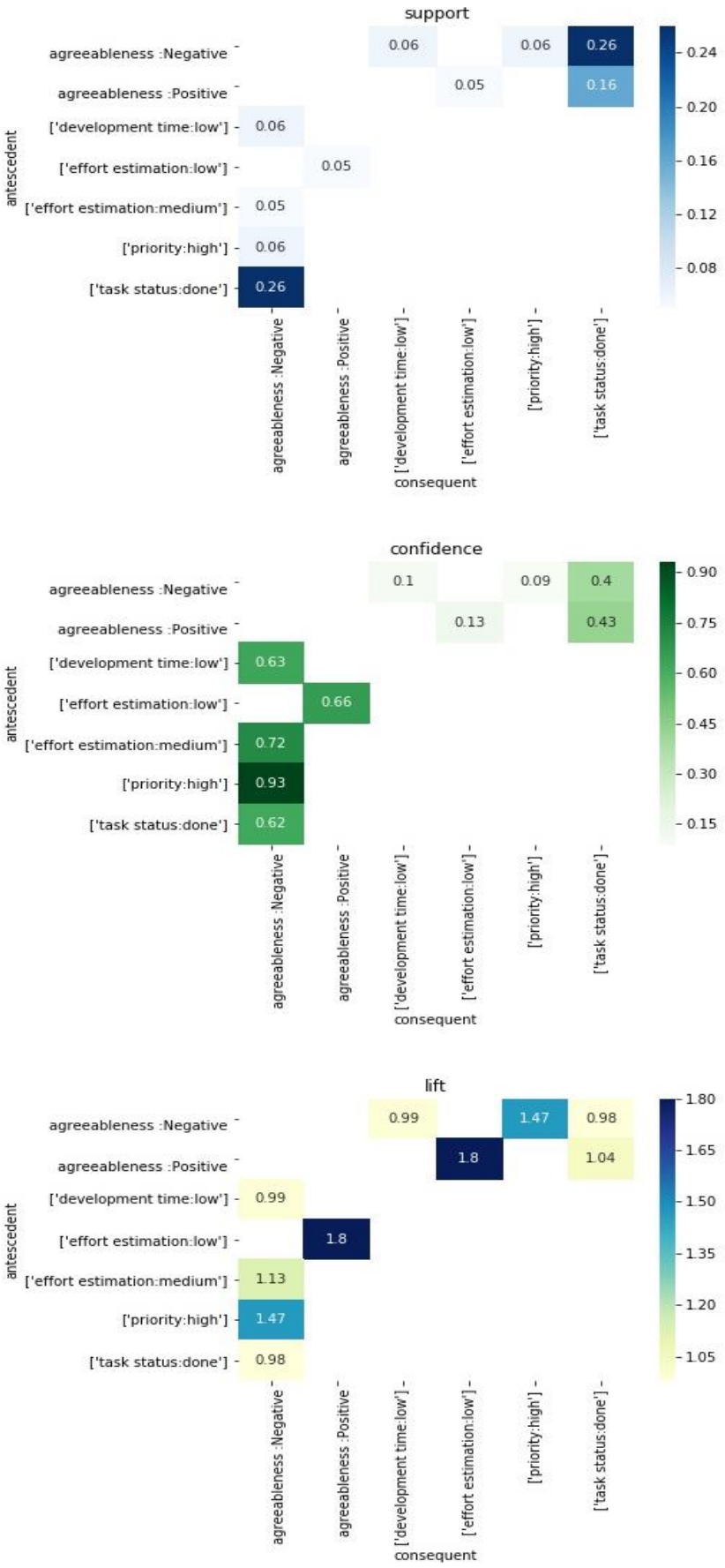

Figure 12: Agreeableness personality trait and related assocaition rules.

4.1.5 Neuroticism

Figure 13 shows the Neuroticism personality traits associations with metrics. According to the plot, neuroticism is associated with high prioritization, low development time and done task status. (neuroticism:Positive => prioritization: high, neuroticism:Positive => development time: low, neuroticism:Positive => status: done). The respective inverse association rules are also supported by sufficient support and confidence. On the other hand, non-neurotic developers are assigning low story points to their tasks, spend low time and mark the tasks done. However, form the inverse rules, only the effort estimation:low => neuroticism:Negative has satisfactory support value to be considered as the valid rule and furthermore, it has the highest lift value among all the neuroticism related rules, which means it is the most correlated (positively) variable to the non-neuroticism personality trait.

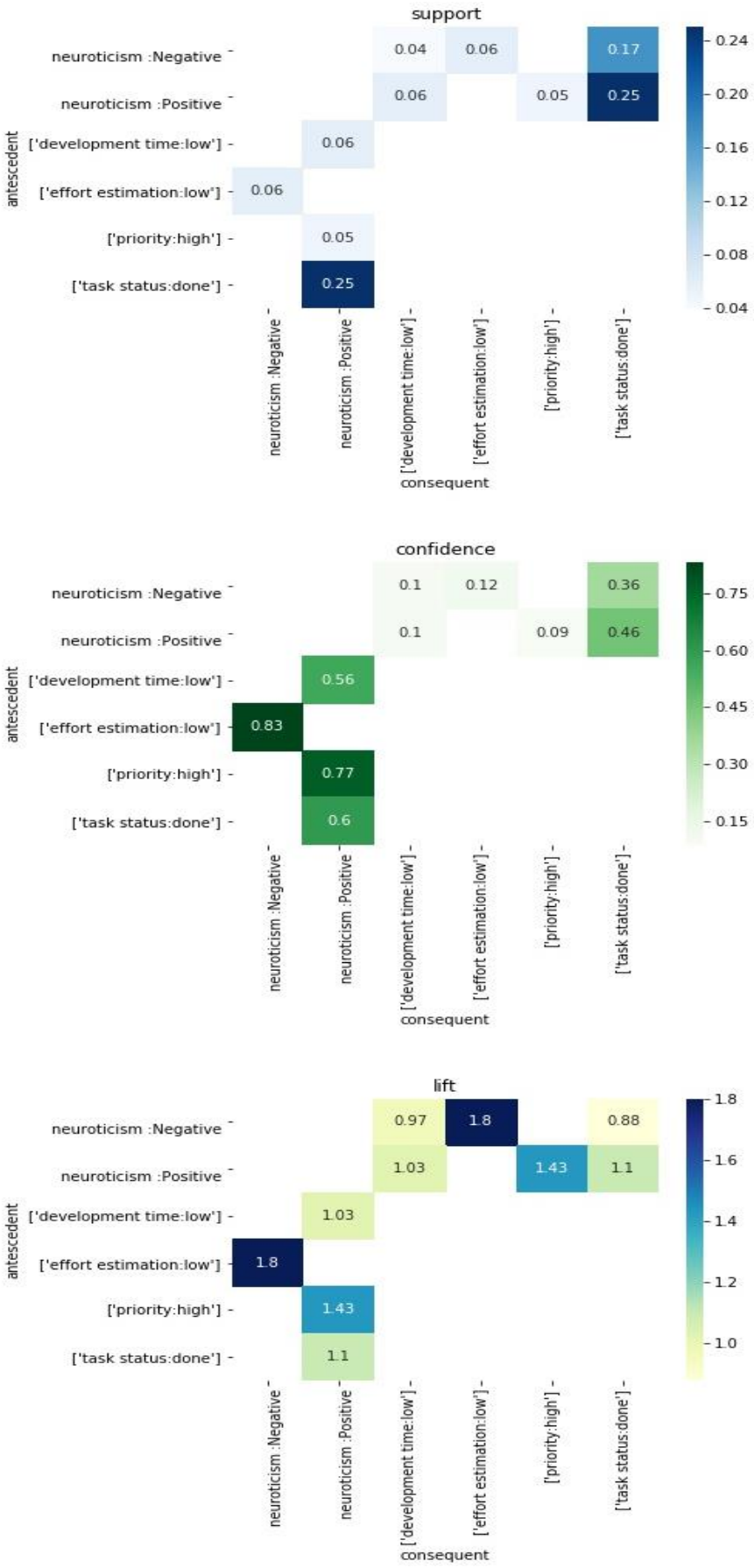

Figure 13: Extraversion personality trait and related assocaition rules.

4.2 RQ2

How the relationship between the personality traits of the developers and their performance is related to previous studies?

RQ2 of this paper is to explore how the relationship between the personality traits of the developers and their performance is related to previous studies, specifically, the researches of Scott et al. [1] and Siddiquei et al. [8]. The methodology of answering the RQ2 is defined in the chapter of Association rules mining method.

Findings related to the FSLSM [Scott et. al]	Big5 and FSLSM correlation [Siddiquei et. al]	Findings related to Big5 (RQ1)	Conclusion
priority: high → Perception: sensing	Perception: sensing ↔ Agreeableness: positive; Perception: sensing ↔ Conscientiousness: positive	priority: high → Agreeableness: positive; priority: high → Conscientiousness: positive	Inconsistent
priority: low → Perception: intuitive	Perception: intuitive ↔ Agreeableness: negative; Perception: intuitive ↔ Conscientiousness: positive	priority: low → Agreeableness: negative; priority: low → Conscientiousness: positive	Consistent
time: low → Perception: intuitive	Perception: intuitive ↔ Agreeableness: negative; Perception: intuitive ↔ Conscientiousness: positive	time: low → Agreeableness: negative; time: low → Conscientiousness: positive	Consistent
time: high → Perception: sensing	Perception: sensing ↔ Agreeableness: positive; Perception: sensing ↔ Conscientiousness: positive	time: high → Agreeableness: positive; time: high → Conscientiousness: positive	Consistent
status: done → Perception: intuitive	Perception: intuitive ↔ Agreeableness: negative; Perception: intuitive ↔ Conscientiousness: positive	status: done → Agreeableness: negative; status: done → Conscientiousness: positive	Consistent
Perception: intuitive → time: low	Perception: intuitive ↔ Agreeableness: negative; Perception: intuitive	Agreeableness: negative → time: low; Conscientiousness:	Consistent

	↔ Conscientiousness: positive	positive → time: low	
Perception: intuitive → effort estimation: high	Perception: intuitive ↔ Agreeableness: negative; Perception: intuitive ↔ Conscientiousness: positive	Agreeableness: negative → effort estimation: high; Conscientiousness: positive → effort estimation: high	Consistent
effort estimation: high → Perception: intuitive	Perception: intuitive ↔ Agreeableness: negative; Perception: intuitive ↔ Conscientiousness: positive	effort estimation: high → Agreeableness: negative; effort estimation: high → Conscientiousness: positive	Consistent
Perception: sensing → time: high	Perception: sensing ↔ Agreeableness: positive; Perception: sensing ↔ Conscientiousness: positive	Agreeableness: positive → Actual development time: high; Conscientiousness: positive → Actual development time: high	Consistent
Perception: sensing → effort estimation: high	Perception: sensing ↔ Agreeableness: positive; Perception: sensing ↔ Conscientiousness: positive	Agreeableness: positive → effort estimation: high; Conscientiousness: positive → effort estimation: high	Inconsistent
effort estimation: high → Processing: active	Processing: active ↔ Extraversion: positive; Processing: active ↔ openness: positive	effort estimation: high → Extraversion: positive; effort estimation: high → openness: positive	Inconsistent
time: low → Processing: active	Processing: active ↔ Extraversion: positive; Processing: active ↔ openness: positive	Actual development time: low → Extraversion: positive; Actual development time: low → openness: positive	Inconsistent
priority: low → Processing: active	Processing: active ↔ Extraversion: positive; Processing: active ↔ openness: positive	priority: low → Extraversion: positive; priority: low → openness: positive	Inconsistent
Processing: active → time: low	Processing: active ↔ Extraversion: positive; Processing: active	Extraversion: positive → Actual development time: low;	Consistent

	↔ Openness: positive	Openness: positive → Actual development time: low	
Processing: active → effort estimation: high	Processing: active ↔ Extraversion: positive; Processing: active ↔ Openness: positive	Extraversion: positive → effort estimation: high; Openness: positive → effort estimation: high	Inconsistent
status: done → Processing: active	Processing: active ↔ Extraversion: positive; Processing: active ↔ openness: positive	status: done → Extraversion: positive; status: done → openness: positive	Inconsistent
status: todo → Processing: active	Processing: active ↔ Extraversion: positive; Processing: active ↔ openness: positive	status: todo → Extraversion: positive; status: todo → openness: positive	Consistent
status: todo → Processing: reflexive	Processing: reflexive ↔ Extraversion: positive; Processing: reflexive ↔ Openness: negative	status: todo → Extraversion: positive; status: todo → Openness: negative	Inconsistent
Processing: reflexive → status: todo	Processing: reflexive ↔ Extraversion: positive; Processing: reflexive ↔ Openness: Negative	Extraversion: positive → status: todo; Openness: Negative → status: todo	Consistent
Processing: reflexive → effort estimation: high	Processing: reflexive ↔ Extraversion: positive; Processing: reflexive ↔ Openness: Negative	Extraversion: positive → effort estimation: high; Openness: Negative → effort estimation: high	Inconsistent
status: done → Understanding: global	Understanding: global ↔ Neuroticism: positive;	status: done → Neuroticism: positive;	Consistent
status: todo → Understanding: sequential	Understanding: sequential ↔ Neuroticism: negative;	status: todo → Neuroticism: negative;	Consistent
time: low → Understanding: sequential	Understanding: sequential ↔ sequential;	Neuroticism: negative → sequential;	Inconsistent

Understanding: global → status: done	Understanding: global ↔ Neuroticism: positive;	Neuroticism: positive → status: done;	Consistent
Understanding: global → effort estimation: high	Understanding: global ↔ Neuroticism: positive;	Neuroticism: positive → effort estimation: high;	Inconsistent
priority: low → Understanding: sequential	Understanding: sequential ↔ Neuroticism: negative;	priority: low → Neuroticism: negative;	Inconsistent
Understanding: sequential → status: todo	Understanding: sequential ↔ Neuroticism: negative;	Neuroticism: negative → status: todo;	Consistent
Understanding: sequential → time: low	Understanding: sequential ↔ Neuroticism: negative;	Neuroticism: negative → Actual development time: low;	Consistent
Understanding: sequential → effort estimation: high	Understanding: sequential ↔ Neuroticism: negative;	Neuroticism: negative → effort estimation: high;	Inconsistent
priority: high → Understanding: sequential	Understanding: sequential ↔ Neuroticism: negative;	priority: high → Neuroticism: negative;	Inconsistent

Table 22. Combined association rules of Scott et al. [1] and correlated Big 5 Personality traits [8].

As table 22 shows, there are 1 or 2 association rules respective to each of the original association rule. In cases where at least one rule is frequent on the base of the newly defined threshold of support and reliable enough based on the confidence value provided, then the rule is considered as consistent. If none of the rules respective of the original rule is frequent or reliable enough, then this rule will be labelled as inconsistent. The results of this association rules analysis are shown in Table 22. Apparently, more than the half of these research rules have been found consistent (16), while the rest of the rules (14) are not frequent or reliable based on the datasets used within this research, therefore, these are counted inconsistent.

5 Discussion

In this chapter the results of the research questions are explained in more detail and the perspectives and the applications of the research are discussed.

This research aimed to study the relation of developers` personality and the metrics of software development. As the results show, there are some consistent patterns.

Notably, the `status done` metric has the highest support due to the high availability of the respective data in the dataset, more than half of the JIRA metrics observations are the `Task status` (~22k transactions), and moreover, 17k transactions are specifically with the `done` value. This can be interpreted as the common habit of the developers – getting things. However, on the other hand, it is interesting to study the personality of the developers that tend to leave the tasks on todo or in progress state, although the number of such habitat-attributed developers and the respective cases is relatively low, it can be influential if such habitat is repetitive, or if the task is critical. Knowing the personalities of the author - developers of unfinished tasks could be crucial information – management should be more observant on the tasks of such developers when the completion of these tasks is vital.

Based on this argument, one more narrow-filtered frequent associations have been analysed - rules within the task status `todo` and `in progress`, as shown in figure 14. Circle size is the indicator of the confidence, more the support – greater the confidence. Similarly, colour variable shows support value, darker the colour – greater the value of support. It appears, that having the status in `to do` status in most frequent within the developers with less neuroticism personality – one way of understanding this association rule is that developers which do not suffer from anxiety or immoderation, have also more patience and can have the multiple tasks assigned themselves without being oppressed to proceed with all of them. Furthermore, the association rules also show that status `in progress` is common among the more extravert and less open developers (status:inprogress => extraversion:Positive, status:inprogress => openness:Negative, status:todo => neuroticism:Positive and their respective inverse rules). The knowledge about this very specific type of metric and the respectively associated personalities can greatly benefit the software company, letting them lower down the scope of the possible causalities of the certain result of the respective metric.

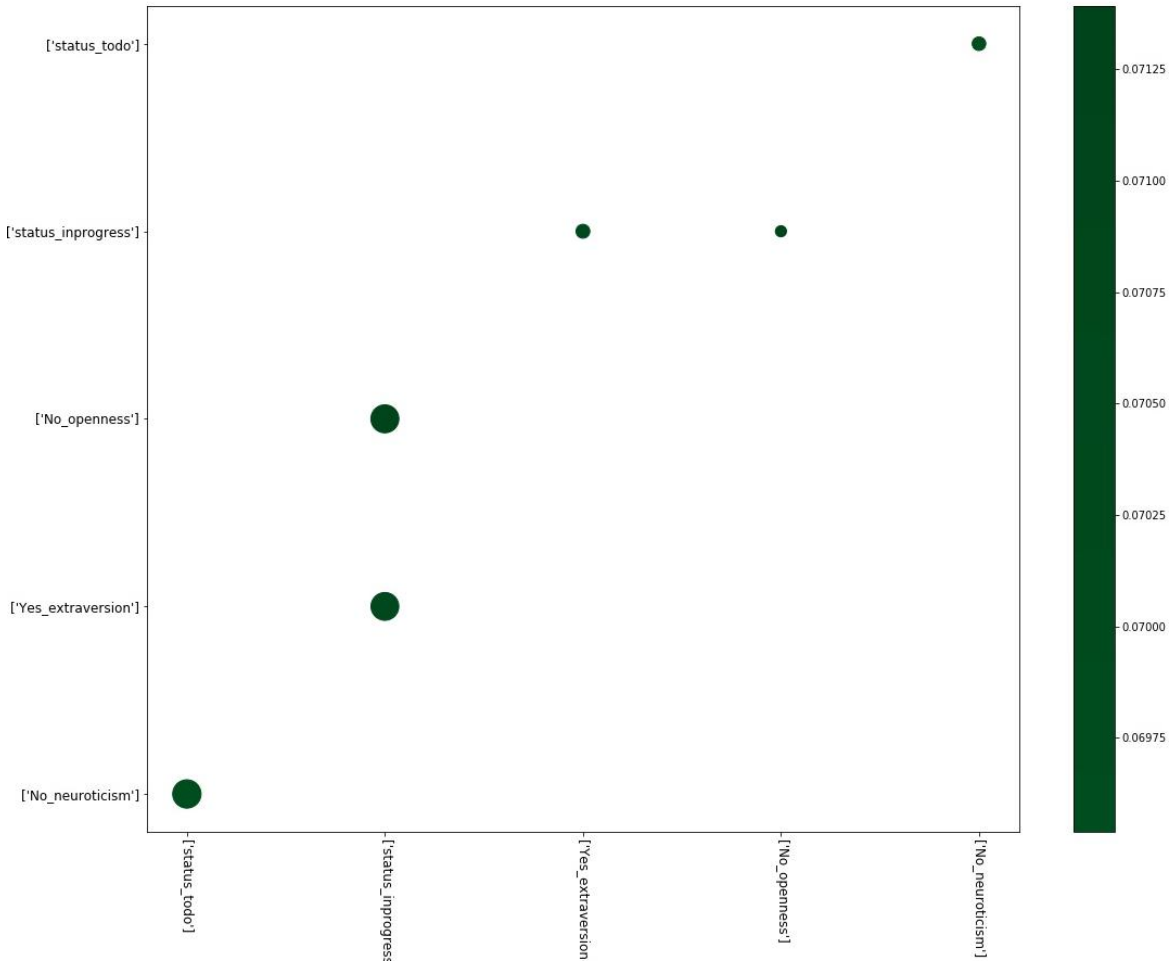

Figure 14: association rules for task status `done` and `in progress`

Secondly, developers within the number of personalities favour estimating the task efforts as `low`. As the association rules were showing, developers with positive values of Openness, Conscientiousness, Extraversion, Agreeableness and the negative value of Neuroticism – share the tendency of assigning the low number of story points to their tasks.

On one hand, developers are assigning story points to the task by themselves to assess the estimation of task complexity. This is a purely predictive process since there can be complications or ease within the tasks that are not possible to foresee in advance. On the other hand, there are an `actual story points` or `actual estimate` term, that describes the actual complexity of the issue, however, it can be measured only after the completion of the task (although, this metric requires existence of the additional attributes that were not possible to obtain within this research dataset), and this leads to the outcome, that the pre-estimation of the efforts can be varied from the actual complexity of the task. The value that the business should pursue regarding this metric is to reduce the difference between the estimated effort and actual complexity of the task, while the result of this research can greatly contribute this idea by providing the group of personalities of developers, that are consistent with task estimation. Seeing that the personality traits, like strong Openness, Conscientiousness, Extraversion, Agreeableness and less Neuroticism are associated with low task effort estimation, can point out the following tendency: these traits are associated with personality facets, like calmness, friendliness, trust, altruism, dutifulness, liberalism etc. These personalities apparently give developers confidence, letting them feel in control of their responsibilities and usually considering their tasks as easily solvable. How accurate they are – this

is another question, and possibly the topic of further research, but even without knowing that – it already lets the software project manager narrow down the analysis and have answers to actions of certain personality developers.

On the opposite side of these traits – the developers with less Openness, Conscientiousness, Extraversion, Agreeableness and the ones with more Neuroticism also share a trend of spending low time on development. This can be explained by the common facets these traits have, that the people with more neuroticism are anxious and try to complete the tasks faster, and fewer conscientiousness people are less organized, not fully disciplined and more chaotic, they may not be fully realizing the impact and dependencies of the task, and only working on the core idea of the issue without considering the other aspects of the task. Similarly, the developers with less openness and less extraversion are the ones that are closed to the experiences, like imagination, adventurousness, excitement-seeking, and likewise the previous trait – they will likely only focus narrowly on the core idea of the task without exploring the further results. The same way can be an explanation of the behaviour of the less agreeable people who lack altruism, morality and modesty.

One more common association rule with the sufficient support and confidence among the developers with the fewer scores of Conscientiousness, Extraversion, Agreeableness and more score of Neuroticism is that they frequently set the task priority to high. Since the trait conscientiousness is associated with the facets – orderliness and dutifulness, this relation can be considered to be logical. It is also notable, that Extraverted people are more friendly, cheerful and active, apparently, less these traits are in people – more likely these people are to consider their tasks as the highest priority. Regarding the low agreeable developers – the rule show quite interesting, yet intuitive pattern: The less agreeable, altruistic and cooperative the person is – more is the likelihood that person will consider their tasks as the most important ones in the sprint and therefore, put the highest priority to them. It is pragmatic to be aware of this tendency when the software product owner or manager is taking care of the prioritization of the tasks assigned to the developers.

As the majority of the results of the association rules show, the developers with the four traits – Openness, Conscientiousness, Extraversion, Agreeableness share the similarities in their performance in Agile software development. There is also sufficient indication of the negative relationship of these personality traits with the Neuroticism trait. Considering the facets of Neuroticism, the anxious traits of the personality are yielding the opposite results compared to the more constructive personality traits. The former statement is also backed in the correlation matrix of the developers` personalities, as presented in Table 19. As the matrix show, the same four personalities have a positive correlation with each other, while neuroticism has a negative correlation with all the other traits. Software project management can take advantage with the formerly mentioned relationship as well – knowing the patterns of the behaviour of the specific personality they can identify in advance some general tendencies within these specific characteristics attributed developer, implement the better forecasting of the performance on the individual or team level.

	openness	conscientiousness	extraversion	agreeableness	neuroticism
openness	------	0.429	0.534	0.365	-0.508

conscietiousness	------	0.337	0.407	-0.519
extraversion		------	0.723	-0.41
agreeableness			------	-0.271
neuroticism				------

Table 19. Big 5 Personality Traits correlation matrix

When comparing these results with the existing literature, in particular, based on the studies of Scott et al. [1] and Siddiquei et al. [8], we must note that both of these studies are done with students as research subjects. In contrast, the current work focuses on software developers that develop their projects in non-academic settings. The findings of this research show that there is enough evidence to conclude that majority of the rules found in the related work [1] are consistent with the current student, being as frequent and reliable as the previous study. Sixteen out of the 30 original rules have had the related association rule with the Big 5 personality trait that has found out to be significant

Seemingly, the new relationship within the FSLSM, Big5 and Agile metrics is a strong bond of the former variables, can be used both, for the other researches in Academia, and within the actual working environment, since it has been proven on the open-source software development project and on the researches of the students. Value for the business is to have the opportunity to analyse personality traits of the developer, learning styles preferences, and the certain agile methodology metrics within the three-dimensional relationship, having more confidence within the analysis, and perform accurate forecast of the certain metric for the given personality/learning style of the developer.

6 Conclusions

This research was conducted in order to understand the relationship between the personality traits of the developers and their performance in Agile software development projects. Previous studies have investigated the personality of developers [2][5][6][7]. These studies have been focused on the developers' personality type changes over time and contribution of the developers within the teams, additionally, these papers were not explicitly focused on the agile software development environment. The goal of this research was the study of developers' personality types within an agile software development environment and, moreover, create the link between developers' personalities and their respective results in an agile environment.

I have achieved the main goal of this research by analysing a dataset of the various software development teams that use JIRA as their issue tracker. I retrieved the big 5 personality types of the developers and calculated the metrics from the developers` contribution logged into the dataset. Next, I performed the association rules mining to study the relationships between the personality traits and the metrics, and finally, I discussed these relationships to address the research questions.

As for the first research question, How Big 5 Personality Traits model is related to the frequently used metrics in Agile, there were several significant relationships between the studied agile metrics and the developers' personality traits. First, all personality traits tend to finish their task and mark them done. Secondly, the developers with the negative score range of the Neuroticism and the positive score of the rest of the traits are favouring the low effort estimation of their tasks. Furthermore, developers with openness, conscientiousness, extraversion, agreeableness, and the ones with high neuroticism are spending low time on the actual development of their tasks. Additionally, the developers with a low score of conscientiousness, extraversion and agreeableness, and the ones with a higher score of neuroticism personality trait frequently set their task priority to high. Considering the bigger picture of all the personality traits, it was evident that the developers with the low level of the neuroticism have the similar preferences of the metrics, as the developers with the high level of the rest of the personality traits, and vice versa. It is reviewed in discussion paragraph, that the results of the research can be beneficial for both, management of the software development teams to improve the performance, meet the set goals and be more precise in the analysis and forecast of the upcoming results, and also for the software developers, to be aware of the possible links of their personality traits to the certain work outcomes.

With regard to the second research question, How the relationship between the personality traits of the developers and their performance is related to previous studies, we have found that the results from this research are consistent with previous studies [1][8]. As a result, more than the half of the association rules found by Scott et al. (16 out of 30) are consistent and reliable within this research as well, that leads to the conclusion, that there is a link of the developers' personality traits and their performances.

7 References

[1] Scott, E., Rodríguez, G., Soria, Á., & Campo, M. (2014). Are learning styles useful indicators to discover how students use Scrum for the first time? Computers in Human Behavior, 36, 56–64
[2] Calefato, F., Iaffaldano, G., Lanubile, F., & Vasilescu, B. (2018). On developers' personality in large-scale distributed projects: The case of the apache ecosystem. In Proceedings - International Conference on Software Engineering (pp. 92–101). IEEE Computer Society.
[3] Schwaber, K., & Beedle, M. (2001). Agile Software Development with Scrum. cdswebcernch (Vol. 18, p. 158)
[4] Beck, K., Beedle, M., Bennekum, A. V., Cockburn, A., Cunningham, W., Fowler, M., … Thomas, D. (2001). Manifesto for Agile Software Development
[5] Karn, J. S., Syed-Abdullah, S., Cowling, A. J., & Holcombe, M. (2007). A study into the effects of personality type and methodology on cohesion in software engineering teams. Behaviour and Information Technology, 26(2), 99–111
[6] Balijepally, V., Mahapatra, R., & Nerur, S. P. (2006). Assessing Personality Profiles of Software Developers in Agile Development Teams. Communications of the Association for Information Systems, 18
[7] Bazelli, B., Hindle, A., & Stroulia, E. (2013). On the personality traits of StackOverflow users. In IEEE International Conference on Software Maintenance, ICSM (pp. 460–463)
[8] Siddiquei, N. L., & Khalid, D. R. (2018). The relationship between Personality Traits, Learning Styles and Academic Performance of E-Learners. Open Praxis, 10(3), 249
[9] Kupiainen, E., Mäntylä, M. V., & Itkonen, J. (2015). Using metrics in Agile and Lean software development - A systematic literature review of industrial studies. Information and Software Technology. Elsevier B.V
[10] Ryckman, R. M. (2013). Theories of personality (10th ed.). Belmont, CA: Wadsworth, Cengage Learning
[11] McCrae, R. R., & John, O. P. (1992). An Introduction to the Five-Factor Model and Its Applications. Journal of Personality, 60(2), 175–215
[12] John, O. P., & Srivastava, S. (1999). Big Five Inventory (Bfi). Handbook of Personality: Theory and Research, 2, 102–138 [13] P. Abrahamsson, O. Salo, J. Ronkainen, J. Warsta. (2002). Agile Software Development Methods: Review and Analysis. Proc. Espoo 2002. 3-107.
[14] Henrick J. (2008), On word-length and dictionary size
[15] Boehm, B. (2002). "Get Ready For The Agile Methods, With Care." Computer 35(1): 64-69

Appendix

I. Text cleaning

Using Regular Expression is the common practice for text cleaning and pattern recognition, simplicity of use and the recognition (and therefore trust) by most of the modern programming languages were the main arguments that backed it`s usage, hence it was the text-cleaning library of choice within this research. Regular expression library and its substitution functions (re.sub) alongside with python string replace (str().replace) functions were used in the environment of Jupyter notebook. Parts of the written texts that have been excluded by the regular expression:

- Texts within not formatted Jira command tags
    - {noformat}(.+?){noformat};
- texts within common Jira code formatting tags
    - {code(.+?){code};
- texts within code tags, squared bracket tags and Html tags
    - <(.+?)>
    - {{(.+?)}}
    - {(.+?)}
    - \[([^[\]{}()]+?)\]
- java commands / JDBC calls
    - "jdbc(.+?)";
- texts of SQL command execution
    - sp_executesql ’’
- texts of command execution
    - exec ’’;
- module calls
    - module(.+?);
- scheduled job calls
    - "job(.+?)";
- texts within MS SQL transaction commands
    - \s*(B|b)(egin|EGIN)\s+. +\s+(E|e)(nd|ND)\s*;
- SQL SELECT, INSERT, DELETE statements
    - (\s*(s|S)(elect|ELECT).+(f|F)(rom|ROM)\s*\S+(\s*(w|W)(here|HERE)\s*\S+\s*\S*\s*\S*\s|))
    - (\s*(I|I)(nsert|NSERT)\s*(I|i)(nto|NTO)\s+.+(V|v)(alues|ALUES)\s*.+\(.+\)\s*)
    - (\s*(d|D)(elete|ELETE)\s*(f|F)(rom|ROM)\s*\S+(\s*(w|W)(here|HERE)\s*\S+\s*\S*\s*\S*\s|);
- texts of System version descriptions
    - [*][*][*]Version(.+?)[*][*][*];
- texts of deployment system technical descriptions
    - [*][*][*]Describe XD Deployment(.+?)[*][*][*];
- texts of system component descriptions
    - [*][*][*]Describe Other Components(.+?)[*][*][*];
- texts within the headers generated by the system
    - '***Description', '***Steps to recreate the problem', '***Error Message:';
- web-links

    - http[s]?://\S+;
- local path links with slashes and backslashes
    - \S+?(?=\/)\/\S*\/\S*
    - r'\S+?(?=\\)\\\S*\\\S*';
- system logs within the asterisks
    - *{50,}(.+?)*{50,}
    - *+(.+?)*+;
- texts of the word that has the length of more than 18 characters
    - .\S{15,}.;
- email addresses
    - \s|\S+(?=@))@\S*;
- SQL command parameters
    - \s|\S+(?=@))@\S*;
- cmd commands
    - --(\s{0,1})\S*
    - ~(\s{0,1})\S* ;
- texts of words with colons
    - \S+\:\S+;
- texts of application version numbers
    - \S+\.\S+;
- texts of command words and versions
    - \S*(_|-|:|\.)\S*(_|-|:|\.)\S+
- non-textual special characters
    - r'(\||~|=|>|_|\[|\]|{|}|--|\/|\\|#)';
- whitespaces
    - \s{2,};
- non-unicode characters
    - r'[^\x00-\x7F]+'
- dates;

Text cleaning techniques removed the most common patterns of the texts generated by the various systems, however, there still are the less frequent cases where the regular expressions could not catch the system log texts. In order to detect and clean these texts with system logs, we introduced text length variable for each written text.

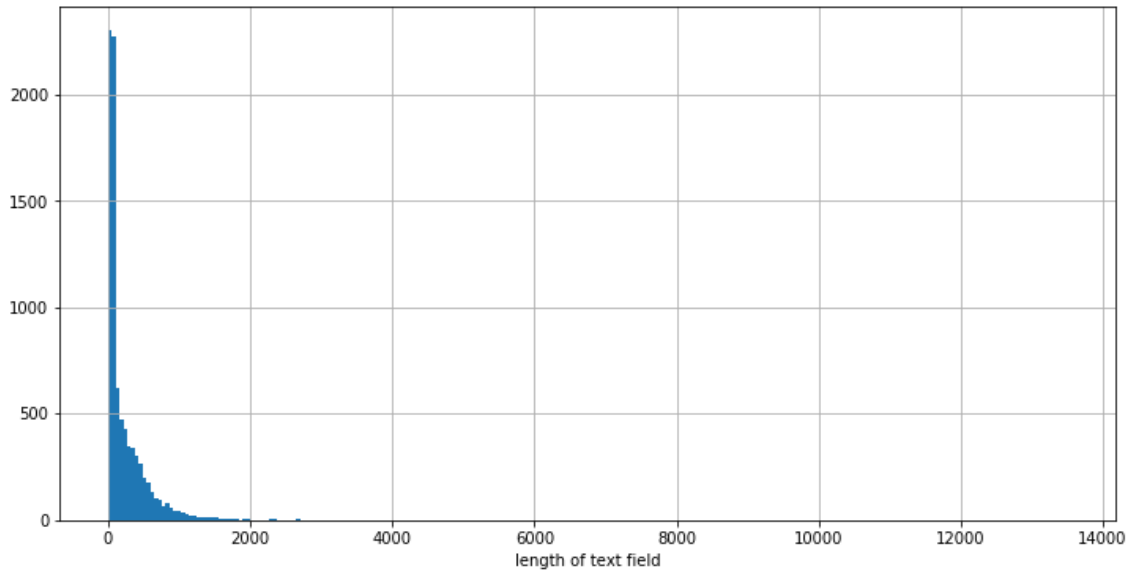

Figure 2: Distribution of word lengths.

Figure 2 shows the histogram of the written text lengths distribution. First, we checked the outliers manually – written texts with more than 5000 characters and these texts were indeed the logs generated by the system. These outliers were removed (6 such written texts). Next, we checked the distribution of the character length of the written texts.

To have a better picture of the lengths of the written text, the distributions are plotted in Figure 3.

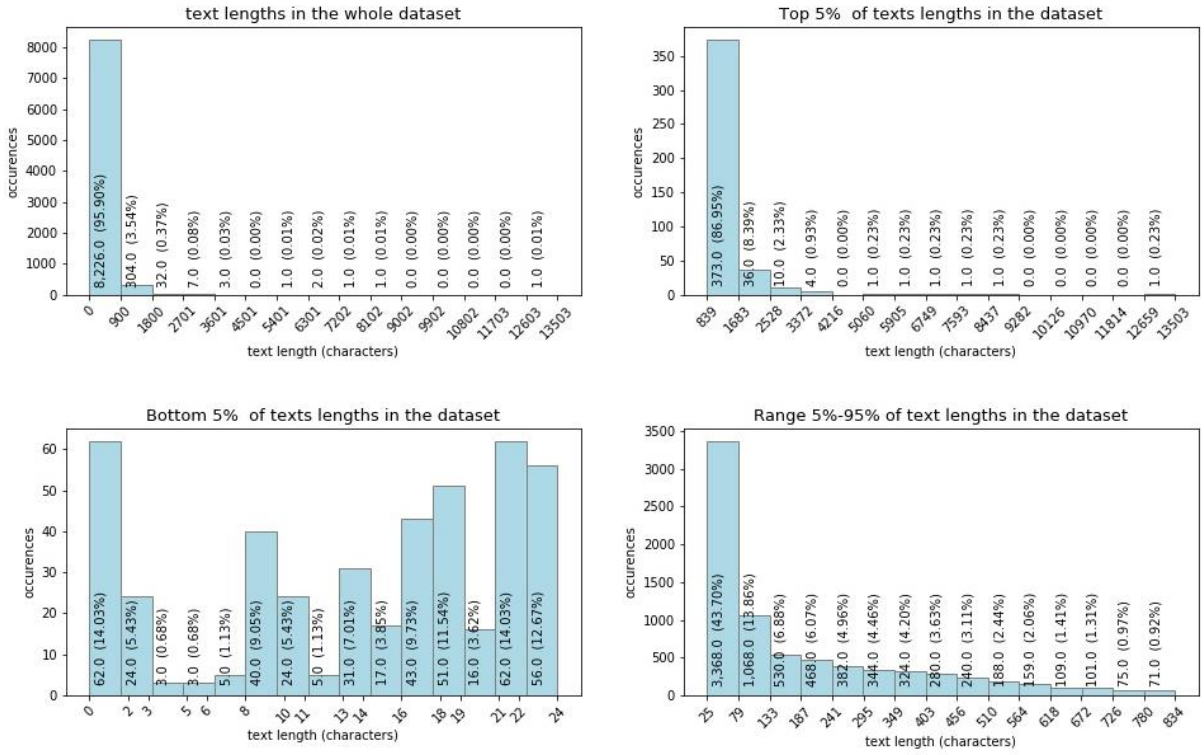

Figure 3. histograms of written text lengths.

The detailed plots of text length distribution show that more than 95% of the written texts have a length of 900 characters or less.

Longest 5% written texts plot shows that majority – more than 86% of these written texts are 1683 characters or less, while the rest of the lengthy written texts, more than 13% (and out of the whole dataset – 0.13*0.05=0.065, roughly 6.5%) are still with an unusually long number of characters. Although the text cleaning procedure excluded the majority of the automatically generated texts and we removed the outliers, there still are the cases of the texts that are likely to be not written by the developers. After a thorough check of the written texts with the highest number of characters, it was found out, that some of them still contain the system logs. Removing the top records with the relatively long written texts would help to make sure that automatically generated texts by the system are presented with down to the minimum in this dataset.

Shortest 5% written texts graph shows that all of them are 24 characters or less, and as many as half of these records are even less than 15 characters long. Considering the following facts, that each English word on average contain 6-8 characters [14], and that 13 characters would make only two words on average, it is easy to understand that the texts with two words are less likely to contribute into forming of the 600 words threshold (The word count threshold is set by the IBM Watson Personality Insights tool that we use in the next chapters for detecting personality traits of the developers), while on the other hand, these two words can be system commands that are useless and even more, can affect the personality traits assessment results.

Based on this reasoning, the transactions with longest 1% written texts (86 written texts in total with more than or equal to 1504 characters length), and shortest 2% written texts (172 written texts in total, less than or equal to 13 characters length) were removed. As a result, the dataset gets 8308 written texts from the original 8578.

II. License

Non-exclusive licence to reproduce thesis and make thesis public

1. I, Tedo Gogoladze,

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

The Importance of Personality Traits in Agile Software Development: A Case Study, (*title of thesis*)

supervised by Ezequiel Scott.

(*supervisor's name*)

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 licence 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 licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Tedo Gogoladze

09/01/2020