UNIVERSITY OF TARTU

Institute of Computer Science

      Curriculum

admin

Title

Master’s Thesis (30 ECTS)

Supervisor(s):      

Tartu 2019

Title

Abstract:

This paper gives a template to theses. It provides the styles, automatisation examples, and instructions on how to write effectively using Microsoft Word.

Keywords:

Layout, formatting, template

**CERCS:**

Pealkiri eesti keeles

Lühikokkuvõte:

Selles mallis kirjeldatakse ingliskeelse lõputöö mall, stiilid ja antakse soovitusi töö teostamisel kasutades Microsoft Wordi.

Võtmesõnad:

Kujundus, paigutus, mall

**CERCS:**

Table of Contents

[1 Introduction 5](#_Toc27258651)

[1.1 Aim of the paper 7](#_Toc27258652)

[2 Terms and Notations 8](#_Toc27258653)

[3 Background 9](#_Toc27258654)

[3.1 Personality type assessment model 9](#_Toc27258655)

[3.2 Researchers preferred model of Personality type assessment 10](#_Toc27258656)

[3.3 Agile software development 11](#_Toc27258657)

[3.3.1 Agile methodologies 13](#_Toc27258658)

[3.3.1.1 SCRUM 13](#_Toc27258659)

[3.3.1.2 XP 13](#_Toc27258660)

[3.3.1.3 FDD 14](#_Toc27258661)

[3.3.2 Agile Metrics 14](#_Toc27258662)

[4 Methodology 17](#_Toc27258663)

[4.1 Technical environment 17](#_Toc27258664)

[4.2 Datasets 17](#_Toc27258665)

[4.3 Metrics 18](#_Toc27258666)

[4.3.1 Actual development time 18](#_Toc27258667)

[4.3.2 Prioritization 18](#_Toc27258668)

[4.3.3 Story estimate 19](#_Toc27258669)

[4.4 Personality traits model 19](#_Toc27258670)

[4.5 Personality traits assessment for software developers 19](#_Toc27258671)

[4.5.1 Text cleaning 20](#_Toc27258672)

[4.5.2 Retrieving developers` personality traits 26](#_Toc27258673)

[4.6 Computation of metrics 28](#_Toc27258674)

[4.6.1 Actual development time metric 28](#_Toc27258675)

[4.6.2 Task status metric 31](#_Toc27258676)

[4.6.3 Prioritization metric 32](#_Toc27258677)

[4.6.4 Story estimate metric 32](#_Toc27258678)

[4.6.5 Metrics from the research of E.Scott 33](#_Toc27258679)

[4.7 Association rules mining 35](#_Toc27258680)

[4.8 Research results 38](#_Toc27258681)

[4.8.1 Correlation 38](#_Toc27258682)

[4.8.2 Association rules results 39](#_Toc27258683)

[4.8.3 FSLM / Big 5 / Agile validation 44](#_Toc27258684)

[4.9 48](#_Toc27258685)

[4.10 General Use 48](#_Toc27258686)

[5 Discussions 49](#_Toc27258687)

[6 Conclusions 50](#_Toc27258688)

[7 References 51](#_Toc27258689)

[Appendix 52](#_Toc27258690)

[I. Glossary 52](#_Toc27258691)

[II. License 53](#_Toc27258692)

[**Non-exclusive licence to reproduce thesis and make thesis public** 53](#_Toc27258693)

[I, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, 53](#_Toc27258694)

[(*author’s name*) 53](#_Toc27258695)

[(*title of thesis*) 53](#_Toc27258696)

[(*supervisor’s name*) 53](#_Toc27258697)

# Introduction

Agile software development has already become the most common software development methodology in the industry. Number of previous researches 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 general, the right diversity of personalities within the team has a significant effect on the team performance [2], in this paper, we will research the personality traits more specifically within the Agile software development teams.

One of the first values of the Agile Manifesto [4] is that Individuals and Interactions are over the processes and tools. Furthermore, it states that developers should work together with business people during the whole project on a daily basis; Developers need to stay motivated and be trusted to get the job done; Face-to-face conversation is the most efficient method of communication and information conveying within the development team; Developers should keep the constant pace indefinitely; The best performance emerges from self-organized teams; Team needs to reflect on effectiveness and adjust behaviour accordingly. Coming from the principles and values of Agile manifesto, the high priority is set on the interaction of the software developers within the development team, and outside the team to communicate with the business people.

Personality traits of the developers are not only the subject of study of personal performance and preferences, but also for the team results. Having homogenous personality and mixture of personality types within the team reported higher cohesion and performance in the research experiments of Karl et al [5]. Therefore, the personality types of the development team are important indicators of the team performance as well.

The model of the personality traits assessment varies in different researches. The formerly mentioned research is using Myers-Briggs Type Indicator (MBTI) to assess personality traits. The same MBTI model was very popular tool during the last decades of past century, but the position has been changed lately. Big Five Personality Traits is another personality traits assessment model, that is currently a dominant choice of psychologists as the preferred consensual model of assessing the personality traits - Openness, Conscientiousness, Extraversion, Agreeableness, Neuroticism. It is also known to have more suitability in assessing the Agile team members personality traits, comparing to the traditional models, including the Myers-Briggs Type Indicator, Big 5 not only provides the 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].

Although the recent studies about personality in software development have actively switched to use Big 5 model, there is still a lack of developers` personality traits relation to the Agile software development. it makes problematic to link it with the studies that using Big 5.

To address this issue, we started with literature review. There are studies that have been done solely on Agile, and solely on Big 5 personality traits model, that have a link in common. In his paper, E. Scott has studied the Agile software development within the context of learning styles using the Felder–Silverman Learning Style Model (FSLSM), on the other hand, the research of N.L. Siddiquei and R. Khalid [8] examined and found out the significant correlation between Learning styles using the above mentioned Felder–Silverman Learning Style Model (FSLSM) and Personality traits of individuals using the Big 5 Personality Traits model. The papers are identifying relation of 1) Agile and FLSM and 2) FLSM and Big 5, notably, both the papers are linked and use FLSM model. 2

The experiment and analysis in these two papers are done on the research of the educational institution students, however, there are no scientific analysis of the industry beyond the academia. Coming from these researches, we will check if there is the scientific evidence of the logical link between these two papers. We will examine How the Agile metrics studied in E. Scotts work (Prioritization, Estimation time, time of completion, user-story state) are related to the Personality traits model (Big 5 Personality Traits) studied in the work of N.L. Siddiquei and R. Khalid, Therefore, our second research question is as follows:

- RQ 1) How Big 5 Personality Traits model is related to Agile metrics according to the previous studies?

For this research question, we use the dataset of open source JIRA logs of eight Agile software development teams. For the first step of our research, we study the personality traits of the software developers from this dataset using IBM Watson API. This tool uses the LIWC (Linguistic Inquiry and Word Count) to analyse the textual data and predict the personality of the individual. The result of IBM Watson is an assessment of personality with Big 5 Personality Traits model, therefore, developers will be assigned the score for each of the Big 5 traits – Openness, Conscientiousness, Extraversion, Agreeableness, Neuroticism. LIWC has been used for detecting personality traits of Facebook users, Stackoverflow community users [7], Apache ecosystem and Github users [2]. Then, analyse the logs of JIRA and calculate the same metrics of Agile from the dataset as in Scott`s work- Prioritization, Estimation time, time of completion, user-story state. Afterwards, we make association rules of the Big five Personality traits and the Agile metrics, and finally, we will examine significance of the association rules, that concludes the relationship between the two papers.

The above mentioned studies have the limited number of metrics used in their research, which makes our work incomplete unless we inspect the more used and frequent metrics of the industry. Therefore, we dig into the topics of Agile metrics. E.Kupainen, M.V.Mantyla and J.Itkonen have studied highly influential metrics in Agile software development by the systematic literature review. High number of occurrences of a metric in the researched literature considered as a sign of high importance. [9]. We study the personality traits of the Agile software developers within the context of these Agile performance metrics, Accordingly, our following research question is:

- RQ2) How Big 5 Personality Traits model is related to the most frequently used metrics in Agile?

We use the same dataset of JIRA logs for this question. We have already performed the assessment of the personality types of the developers with Big 5 model for the previous question, therefore, we move forward to analyse and evaluate the most important Agile metrics according to the studies of E. Kupiainen, M.V. Mäntylä and J. Itkonen. Finally, we analyse and point out the relationship between the big 5 personality traits of the developers and the Agile metrics.

The dataset of JIRA logs, contains the logs of 8 different agile teams, which brings out the possibility to assess not only personal, but also a team performance. We evaluate the combination of the different personality traits as the factors of successful or unsuccessful team performance. Accordingly, our last research question is:

- RQ3: How personality traits are important for agile teams?

Answering the research questions will be beneficial mainly for the software development industry. Knowledge of the relationships of the personality traits and the specific Agile metrics can be useful for the team managers, to form and adjust the teams accordingly. They would be able to predict and 3

know in advance the possible outcomes of having the developers of the specific personalities in their teams.

For the developers, knowing their own personality traits can be a note for the possible positive or negative results in their work. Considering the fact, that the personalities of the developers can be changed over the short time [2], they can work on their own specific personality traits to improve the personal and team productivity.

For the Researchers, answer of the questions of this paper can be useful for the further researches on the Agile teams and Personalities. Furthermore, answering to the second question can add more scientific proof and strengthen the research findings of the linked researches. Having the positive results in relationship of Big 5 Personality traits and Agile preference metrics can make a solid bridge that will make a solid and fact-based triangular relationship of Felder–Silverman Learning Style Model, Big 5 Personality Traits model and Agile metrics.

## Aim of the paper

# Terms and Notations

# Background

## Personality type assessment 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 others needs.
* Neuroticism describes the level of negative emotions expression and personal tolerance and to the stress [2].

Personality trait factors and their definer adjectives, scales and sort items, as described by McCrae and John, are given in Table 1 [3].

The claim of five-factor theorists is that these factors, singly or in combination, can be found in virtually all personality instruments [McCrae and John, 1992]. 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. *[4]*.

|  |  |  |  |
| --- | --- | --- | --- |
| **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  Trusting | Not critical, sceptical  Behaves in a giving way  Sympathetic, considerate  Arouses liking  Warm, compassionate  Basically trustful | Trust  Straightforwardness  Altruism  Compliance  Modesty  Tender-Mindedness |
| Conscientiousness | Efficient  Organized  Planful  Reliable  Responsible  Thorough | Dependable, responsible  Productive  Able to delay gratification  Not self-indulgent  Behaves ethically  Has high aspiration level | Competence  Order  Dutifulness  Achievement Striving  Self-Discipline  Deliberation |
| Neuroticism | Anxious  Self-pitying  Tense  Touchy  Unstable  Worrying | Thin-skinned  Brittle ego defenses  Self-defeating  Basically anxious  Concerned with adequacy  Fluctuating moods | Anxiety  Hostility  Depression  Self-Consciousness  Impulsiveness  Vulnerability |
| Openness | Artistic  Curious  Imaginative  Insightful  Original  Wide interests | Wide range of interests  Introspective  Unusual thought processes  Values intellectual matters  Judges in unconventional terms  Aesthetically reactive | Fantasy  Aesthetics  Feelings  Actions  Ideas  Values |

*Table 1: Personality trait factors and definers*

## Researchers preferred model of Personality type assessment

A systematic literature review on Information systems researches show 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 focused more on individual personality rather than the team of individuals. [2].

Researches on the personality traits of Information System personnel with Jungian Instruments (MBTI) conclude that they tend to be Feeling and Thinking types. Some studies reveal that Sensing is the most dominant trait characteristic, while the studies with Non-Jungian instruments (Big five personality traits model) reveal that Information systems personnel are conscientious, confident, persistent, self-assured, assertive, dominant and abstract thinking.

Authors` systematic Literature review is concluded by the statement 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. [2]

Several researchers have complied the tables that show standard personality measures to the big five personality traits, also the deep understanding and comparability of the five factors. Researchers may interpret and convert various factors into the big five personality traits model using these tables. Table 2 shows empirically relation of some of such scales to the big five [4].

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Instrument** | **Factor** | | | | |
|  | Extraversion | Agreeableness | Conscientiousness | Neuroticism | Openness |
| Hogan Personality Inventory | Sociability | Likeability | Prudence | Adjustment (-) | Intellectance |
| California Psychological Inventory | Sociability | Femininity | Norm-Favoring | Well-Being (-) | Achievement via independence |
| Multidimensional Personality Questionnaire | Social Closeness | Aggression (-) | Control (-) | Stress Reaction | Absorption |
| Adjective Check List | Self-Confidence | Critical Parent (-) | Military Leadership | Ideal Self (-) | Creative Personality |
| MMPI Personality Disorder Scales | Histrionic | Paranoid (-) | Compulsive | Borderline | Schizotypal |

*Table 2: Big five factors personality traits relation to the other instruments.*

## 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 the high-quality software development. Agile values individuals and interactions, working software, customer collaboration and fast responding to the changes. [5]

Main characteristics of the agile projects are [8]:

* 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 (2002) has compared and analysed process-oriented methodologies vs the agile software development methodology. In his work, Open source software is included as a multifaceted variant method of Agile. Table 3 shows the comparative analysis according to the author [8]:

|  |  |  |
| --- | --- | --- |
| **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

## Agile methodologies

## SCRUM

SCRUM is a commonly used Agile software development method. Its full product development phase is represented by the sprints that release the incremental development states of the needed product. All of these task items are gathered in backlog, each item has estimated velocity and complexity, and for each sprint, the items are taken that do not exceed the available resources. During the sprints, the team has short stand-up meetings where the team members talk about the status and the progress of their tasks and discuss the daily work topics. After the sprint is over, the team has the meeting for the past sprint review. The team gathers once more time to have a retrospective analysis of the past sprints. [6]

Software development approaches consider the development processes to be unpredictable and risky, while SCRUM on the high level offers reliable, agile and responsive results, that is considered as a mechanism that reduces the risk and unpredictability. [6]

## XP

Extreme Programming (XP) has been actively used in software development after 2000. The method has been derived from the traditional development methodologies, as the development lifecycles have been lengthy causing various issues and dissatisfaction. [8]

The typical lifecycle of extreme programming is made of five stages: [8]

* Exploration: The customers of the project make user stories of the features they want to be in the scope of the first release. They write a short description of each feature on the stories. At the same time, the system prototype is built to test the architectural design and possibilities by the programmers. The first phase takes from a couple of weeks up to several months.
* Planning: The user stories are agreed and prioritized, the programmers estimate the efforts needed for the user stories to be released and the schedule is agreed, then the first short release is done.
* Interactions: on this stage, the schedule is split to the iterations, that will take up to several weeks to implement. During the iterations, the system architecture is created for the whole system. Customers assign the stories selected for each iteration and create the functional tests. At the end of the iterations, the tests that are run, and at the end of all the iterations, the system goes for production.
* Productionizing: Before the product will be released to the end customer, the system gets additional tests, performance is checked and if the changes are needed, the decision is agreed if the lacking features will be included or not in that release.

Death: the stage is current when the customers` requirements have been fulfilled and no more user stories are left to be implemented. At this phase, the customer is satisfied with the whole system, it’s reliability and performance. Other alternatives to the death stage to occur is when features become impossible or expensive for development.

## FDD

The feature-driven development methodology is described as the adaptive and agile application of systems development. Generally, FDD has the focus on the building and designing stages, rather than reaching the whole software development process, also the method is designed to be compatible with the other tasks of the project. FDD prioritizes the quality of the project with careful monitoring and the persistent deliveries of the processes. [8]

The process of FDD consists of five sequential phases: First, overall model development, when the system context and the requirements are built, use cases and the functional specifications are set and the domain area specialists know the scope. Second, building a list of the features. the development team builds and presents the list of the functions grouped for each of the domain area, that makes the feature sets. The third phase is Planning by features – all the features are sequenced and prioritized and assigned to the expert programmers. The fourth stage is Designing by feature when the set of the features and the respective development teams are selected. Finally, the fifth stage is Building by feature – the building of the features is divided into the iterations, within each of the iteration there are selected feature sets to be built. The iterations generally take up to two weeks to complete. [8]

## Agile Metrics

E.Kupainen, M.V.Mantyla and J.Itkonen have studied 30 researches and 36 case studies in the systematic literature review of Agile Software Development methodologies and it’s 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. [7]

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. Metrics massively used in primary studies of Agile development are presented in Table 4. In most of the measurements, internal attributes show Product class entities, while under external is listed also Resource class entities. Additionally, some of the metrics can be grouped in more than one category. [7]

|  |  |  |
| --- | --- | --- |
| **Entities** | **Attributes** | |
| ***Internal*** | ***External*** |
| ***Products*** | | |
| *Products* | Running tested features, build status | Customer satisfaction, progress as working code |
| *Test plans* | Number of test cases |  |
| *Code* | Technical debt in categories, technical debt in an effort, violations of static code analysis |  |
| *Features* | Task’s expected end date, the task done, effort estimate, story completion percentage | Business value delivered |
| *Requirements* | Requirement’s cost types, percentage of stories prepared for sprint |  |
| Defects |  | Defect trend indicator, predicted the number of defects |
|  | | |
| ***Processes*** |  |  |
| *Testing* | Defect count, test success rate, test failure rate, defects deferred, test coverage, test growth ratio | Number of bounce backs, fault slips |
| *Implementation* | Velocity, number of unit tests, completed web pages, cost performance index, schedule performance index, planned velocity, common tempo time, check-ins per day, fix time of failed build | story flow percentage |
| *Requirements Engineering* | Velocity of elaborating features |  |
| *Whole* | Development cycle | Cycle time, lead time, processing time, queue time, maintenance effort, work in progress, variance in handovers, through put, queue, implemented vs wasted requirements |
|  | | |
| ***Resources*** |  |  |
| *Team* |  | Team effectiveness |
| *Customer* | Revenue per customer |  |

Table 4: Agile metrics in primary studies.

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 development time, Load factor, Work in Progress and Lead time. [7]

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, Velocity and Effort Estimate are the most important, for sprint progress tracking, completed work, number of automated tasks, burn-down check-ins and defects are the main indicators. For understanding and Improving quality, Number of change requests, maintenance effort, Net Promoter Score and Defects are the most influential. For Fixing Software Process Problems, major indicators are time-related metrics: Lead time, processing time, Queue time. For Motivation of the people defects, defect trend, fix times of failed build and build status are among the ones that matter the most. [7]

# Methodology

## Technical environment

Majority of the research analysis is done in a local version of jupyter notebook by Anaconda. Languages used: Python and SQL.

Libraries from Python: Jupyter notebook (environment), Pandas, Numpy, Matplotlib, Seaborn, Re (regular expression), Collections, Pandasql (SQLite local), Apyori (apriori algorithm for association rules)

## Datasets

For the research we used open-source JIRA issue tracker datasets that are published by the project `Spring` framework.

Four datasets were used: issues, changelog, users and sprints. Figure 1 describes the ER diagram of these datasets and shows attributes of each of them.

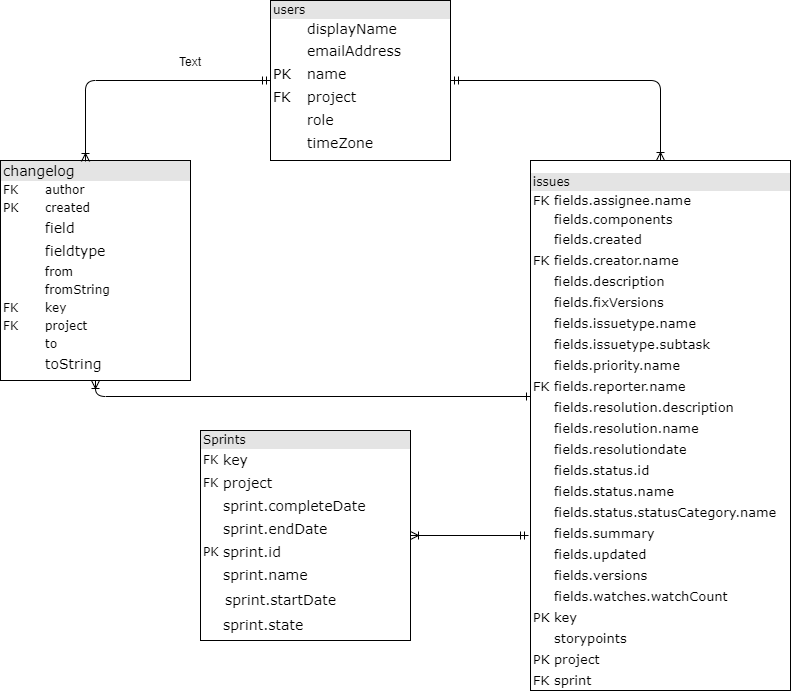


Figure 1. Datasets ER diagram

Issues dataset

Jira Issues dataset: 35234 Kb, 18090 rows.

Issues dataset is contains all the necessary attributes about the specific Jira issue. Dataset consists of 18090 rows, out of which there are 15155 unique issues. Dataset is linked to Sprints by `project` and `sprint` foreign keys, and to users by reporter name and creator foreign keys. Column `key` is the primary key attribute of the dataset alongside with `project`.

Changelog dataset

Jira Changelog dataset: 63592 Kb, 332690 rows.

Dataset for tracking the changes, that are made by 1104 unique users to the 15066 unique Jira issues. Key attributes are project code (`project`) and Jira Issue code (`key`), log creation date (created), username of the author of change (`author`), field that has been changed (`field`), value of the field that was set before the change (`fromString`) and value of the field that was set (`toString`).

Jira Users dataset: 245 Kb, 3124 rows.

Dataset of the 1533 unique users across all the projects. User name stored as `name` column is a primary key and used by the other datasets for linking their foreign key. Dataset has additional attribtues - `displayName` optional, denoting the full name of the user, `emailAddress` for email address of the user and `role` with the values 'assignee', 'creator', 'reporter'.

Jira Sprints dataset: 2646 Kb, 22891 rows.

Dataset that stores information about the sprints of each project, sprint start and end dates and sprint status.

## Metrics

### Actual development time

`Actual development time` denotes the time spent on the specific issue by a developer. Unlike the more common metric `Elapsed time`, actual development time can be more useful to distinguish each developers’ work separately, since `Elapsed time` would simply measure the time from registering the issue until marking it done, whereas there can be many developers working on each task. Actual development time will catch only the times that was actually being worked, excluding the waiting times.

### Prioritization

From the metrics of sprint and project planning, there are three types of activities that are commonly used according to (Agile Metrics, Eetu Kupiainen a, Mika V. Mantyla a,b,⇑, Juha Itkonen a ) – prioritization, scoping and resourcing. Prioritization metric denotes how the prioritize their tasks. Various companies use their own preferred approach for prioritizing depending on the sprints and projects, but if considering the issue-level and taking into account the datasets that have been used in this research, prioritization metric can be retrieved from the `priority` field.

### **Story estimate**

The other most used activity of planning according to the authors are scoping and resourcing. In this paper, estimation of scope metrics are also used. Although there is no doubt that Velocity is the most widely used metric for the sprint estimation, it still is more a measurement of team/sprint effort, that does not take into account the single develoeprs perspective for a given issue. For having more detailed understanding on the personal level, story estimation was used by the individual developers for each separate issue. In general, Story estimate is the metric that denotes the efforts that the developer is marking to the task and the estimation usually is assessed with the Fibonnaci sequence numbers.

* + 1. **Task status**

There are progress tracking metrics that are used for monitoring the progress. Depending on the project, the metrics can be, in a majority of the researched cases, project progress or increasing visibility and achieving goals. These metrics are commonly accepted measurement for the project standards. Task status was the choice of metric for the individual developers’ progress tracking. Task status metric is assessing how the developers tend to assign the status to the task – close them, put in progress or in to do status list.

## Personality traits model

As this research is on one hand, measuring the metrics of the Agile software development on the individual developer`s level, and on the other hand, it studies the personality traits of these developers, choosing personality type assessment model was a matter discussion.

As described in the background chapter, Authors have done the systematic literature review of the papers to define the extensively used personality assesment methodology models.

The research is concluding, that the five factor model, also known as the Big Five Personality Traits model is better in terms of measuring it`s five factors, and additionally contains a trait – Neuroticism, that is only slightly examined by Myers-Briggs Type Indicator.

Furthermore, there is a significant ease of use of Big 5 Personality traits model and additionally, datasets that are used in this research are applicable for big 5. These arguments supported the idea of choosing big 5 personality traits model.

## Personality traits assessment for software developers

Within this research, IBM Watson Personality Insight was used for getting the Developers personality traits assessment.

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 API (Application Programming interface) that extracts the personality insights depending on the social media, digital communications or the individuals` data. IBM uses linguistic analytics to determine the personality characteristics within the Big 5 personality traits model.

When using the IBM Watson Personality Insights API, 600 words is the minimum threshold of the text input, that can be used to perform statistical analysis. Accordingly, the first task is to retrieve the texts written by each developer.

## Text cleaning

IBM Watson offers the Big 5 personality traits assessment for the input of 600 or more words written by a person textually. Therefore, the first task towards getting personality profiles is to retrieve the texts written by each developer.

Changelog dataset has the history log of changes in the textual and coded values, therefore, it was commonly used for getting the texts of developers. There are number of fields in Jira that are potentially filled by the developers, adn these texts need to be retrieved. However, there`s an issue that comes up in this step: considering, that the main users of these Jira software projects repositories are software developers, the texts that developers have inputted into the various fields of Jira are quite technical, in many cases – just a copy of the code snippet of a certain programming language, error code and syntax, system logs, stack traces, technical commands, other technically formatted texts etc. To resolve the former issue, a complex text-cleaning activities should be performed and got only the actual textual input that was handwritten by the developer.

To go back to the changelog dataset, at first, need to filter only the necessary fields. The columns that were used from changelog dataset:

* `author`, author of the change, can be taken as-is.
* `project`, code of the project, can be taken as-is.
* `key`, the code of the Jira issue, can be taken as-is.
* `created`, the date and time when the change was performed. Can be taken as is.
* `field` that denotes the field of Jira software that has been changed.
  + Need to detect the `field` values that represent the textual value columns, and filter them.
* `toString` column stores the values that were set to the given field.
  + Need to retrieve the textual values that have actually been written by developers

Then, need to detect the fields where the developers input the textual information, as mentioned before. After the initial check of the `field` column values, the main ones that contain manually written texts are: 'summary', 'description', 'Acceptance Criteria', 'Comment', 'Epic Name', 'Out of Scope', 'QA Test Plan', 'Epic/Theme', 'Migration Impact', 'Business Value'

|  |  |
| --- | --- |
| **Text Field** | **Occurrences** |
| 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 |

After a thorough check of these fields, it was revealed that 'summary', 'description', 'Comment', 'Acceptance Criteria', 'Migration Impact', 'QA Test Plan', 'Out of Scope' were the ones that were surely holding the useful textual data, so this research will be focused on these. In total, there are 8578 rows where these fields are filled.

Additionally, an Jira user may submit a change of the textual value into one column multiple times. For example, one can write the half of the description in the description text. Then, after a while when one realizes the data was partial, they can input the full description text. In such case, there will be two records of `description` field change in the changelog dataset, whle the last edit is the only complete and thus sufficient for this research, while the very first edit is the partial copy of the last one, which is on one hand, a duplicate, and on the other hand, an incorrect/insufficient edit of the data. Possibility of such duplicates/incorrect data has to be avoided. 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 user, and the latest one will be calcualted from the `created` column value.

After taking the latest edits, detecting the textual fields, and filtering the changelog respectively, analysis of the set text values has shown, that there is a need of text cleaning from technical terms and automatically generated snippets copied form the various software logs or editors, as it is described in the beginning of this chapter.

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 argumetns that backed it`s usage, hence it was the text-cleaning library of choice within this research as well. Specifically, regular expression library and its substitution functions (re.sub) alongside with simple string substring (str().replace) functions were used in the environment of Jupyter notebook. Parts of the 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;

After applying the text cleaning techniques, it was necessary to calculate the length of the text for each of these field text values and created the dataset of the textual values with the following structure of columns: `key`, `project`, `author`, `date`, `field`, `text`, `textlength`.

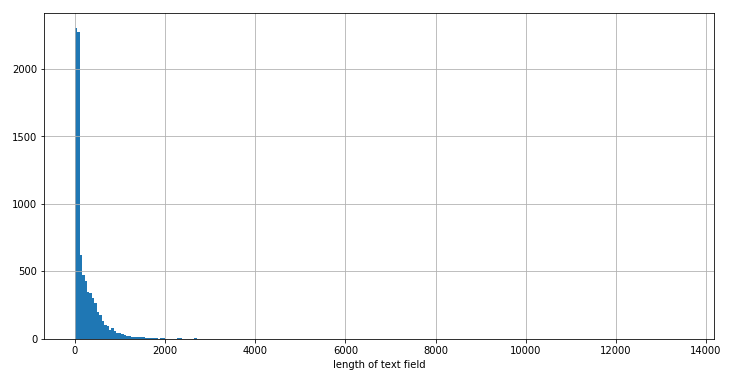
Each row in the text values dataset represents authors written text on a single task. In the end, the results should be one text row per user, hence all the texts written by the given developer were combined on all the Jira issues in all the fields.

Length of the text in words is an important variable, since the tool that will be used for personality traits assessments – `IBM Watson Personality Insights` demands at least 600 words as an input.

Before combining all the texts of each developer, it`s important to check the basic statistical describtion of textLength field and distribution of of these length values.Table x shows such statistical description.

|  |  |
| --- | --- |
|  | **text Length** |
| **count** | 8578 |
| **mean** | 240.448 |
| **std** | 383.987 |
| **min** | 0 |
| **25%** | 52 |
| **50%** | 92 |
| **75%** | 322 |
| **max** | 13503 |

Table x: description of textLength variable from the texts dataset.



Picture x: Distributino of text field lengths.

Picture x shows the histogram of the fields text lengths distribution. It is obvious that there are outliers in the dataset – very few number of text fields, that have suspiciously long texts. To have the better picture of the field text lengths, distribution from the subsets of the datasets are plotted as well:

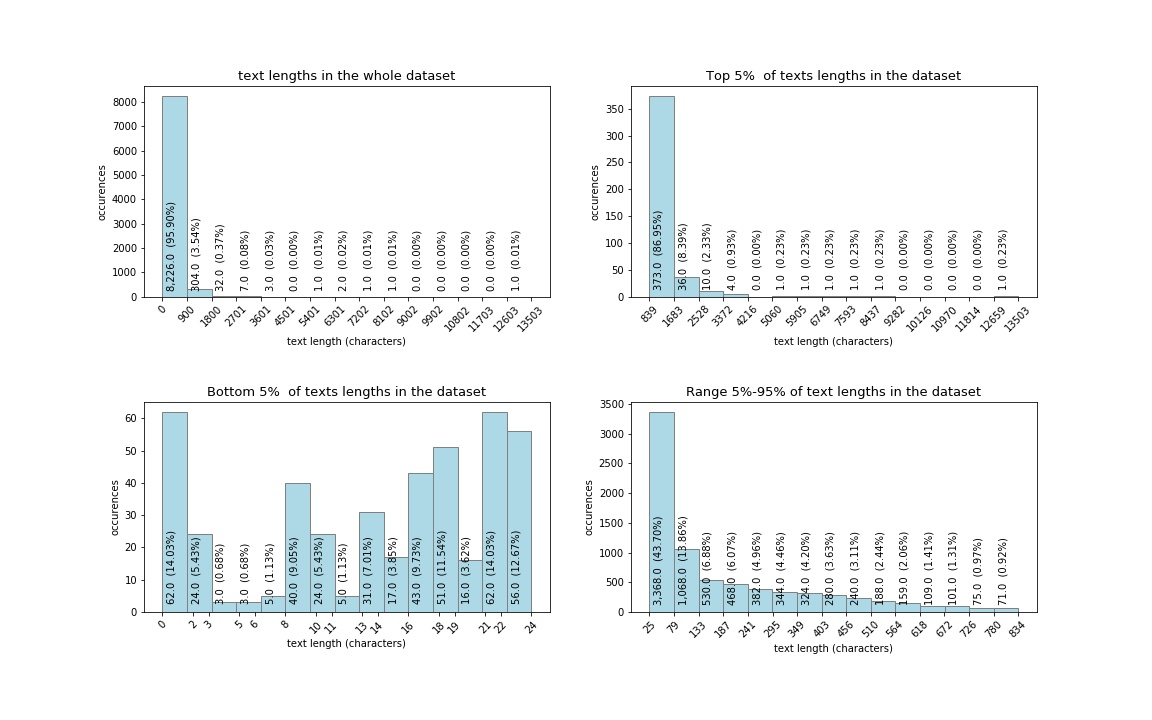


Figure 1. distribution of text lengths.

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

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

Shortest 5% texts rows 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 **\*\***contains 6-8 characters, and that 15 characters would make only two words, it is easy to understand that the texts with two words are less likely to contribute into forming of the 600 words threshold, 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 the arguments mentioned above, removed the rows with longest 1% texts (86 records with more than or equal to 1504 characters), and rows with shortest 2% texts (172 records, less than or equal to 13 characters).

After the above mentioned filter, dataset gets 8308 rows from the original 8578 rows.

Next step that is performed is to group the texts written by each developer. Texts from all tasks that was written by the same developer were combined. Using this approach, the result is the dataset of the texts written by 618 developers.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **number\_of\_texts\_combined** | **words\_in\_text** | **texts\_length** |
| **count** | 618 | 618 | 618 |
| **mean** | 13.443 | 514.73 | 2990.117 |
| **std** | 41.068 | 1369.427 | 8003.169 |
| **min** | 1 | 3 | 14 |
| **25%** | 1 | 44 | 253.25 |
| **50%** | 2 | 116 | 653.5 |
| **75%** | 9 | 364 | 2107.75 |
| **max** | 390 | 15004 | 87980 |

Table x. Description of the combined texts written by the developers.

Description of this dataset shows, that the developers have filled from 1 to 390 different text fields of various tasks. Based on the percentiles, it is obvious, that the half of the developers fill 1 or 2 different text fields. But the fact that mean number of number of filled texts (13.44) is greater than the 75th percentile, says that there are several developers, that have significantly higher number of text fields filled (max is 390), unlike the majority of the developers, that have 1 or 2 text fields filled.

## Retrieving developers` personality traits

After cleaning and filtering the dataset, it needs to be grouped by the users. Since the same users could be present into multiple projects, the email address was additionally used to identify the actual users (whenever the email address was present in the users' dataset). As a result, in total, dataset has 100 unique users that have more than 600 words manually written in Jira fields, therefore, these users are capable to get the assessment of personality traits.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***Project*** | tistud | timob | Dnn | xd | mesos | nexus | apstud | mule |
| ***Users*** | 33 | 30 | 20 | 15 | 12 | 11 | 10 | 5 |

Table 1.

Table 1 shows the number of users per group. 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.

The next step performed is retrieving the results from IBM Watson Personality Insights API. The texts written by the developers was given to API in JSON format as an input. API has also returned the results in the JSON format, which then was parsed and stored into pandas dataframe.

Resulting JSON file returned by IBM Watson contains assessment of personality trait with percentile and raw score, significance indicator, percentile and raw score results of the facets that belong to the personality. This research focuses only on Big 5 Personality traits – Openness, Conscientiousness, Extraversion, Agreeableness, Neuroticism, therefore the results of the sub-categories of these traits are ignored.

Furthermore, IBM API has returned the normalized percentiles and the raw scores. 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. Both, normalized and raw score percentiles are the double type numbers in range of 0 to 1.

Distribution of personality traits assessment results are shown on Figure 2. top rows on the graph represent the personality assessment percentiles of the users as normalized by IBM, while bottom row shows the respective raw scores of personality assessments percentiles.

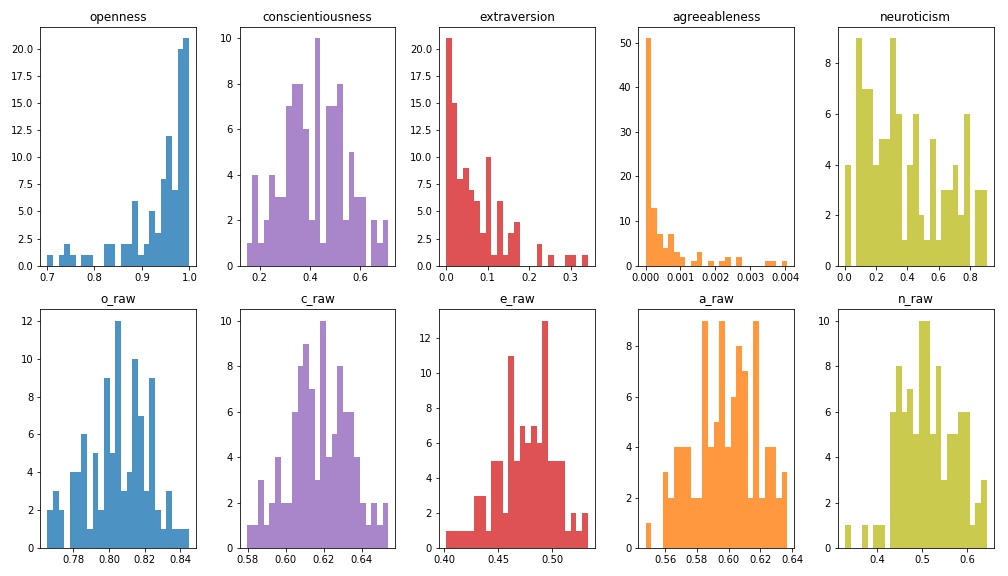
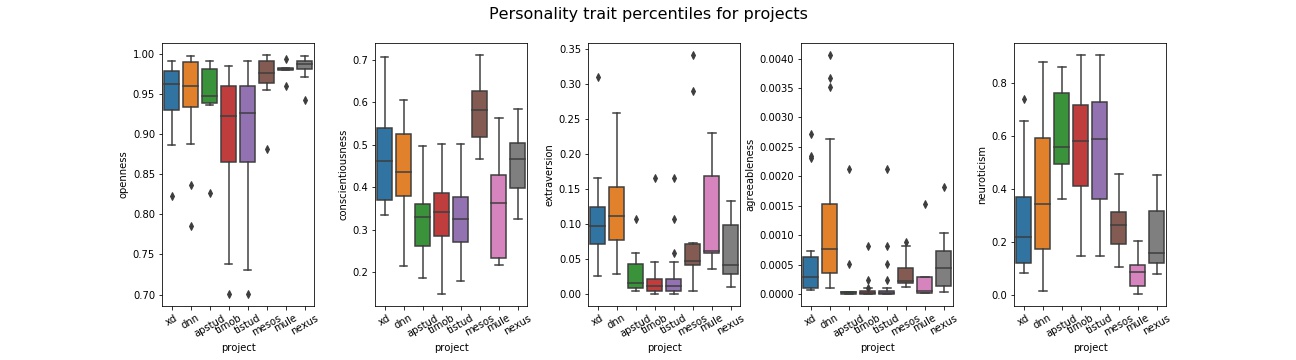


Figure 2: Histograms of users` personality trait raw scores and percentiles

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 within 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. Most distinctive difference between these two scores are returned on agreeableness. 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 on Figure 3. 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, as given in Table 1, 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.



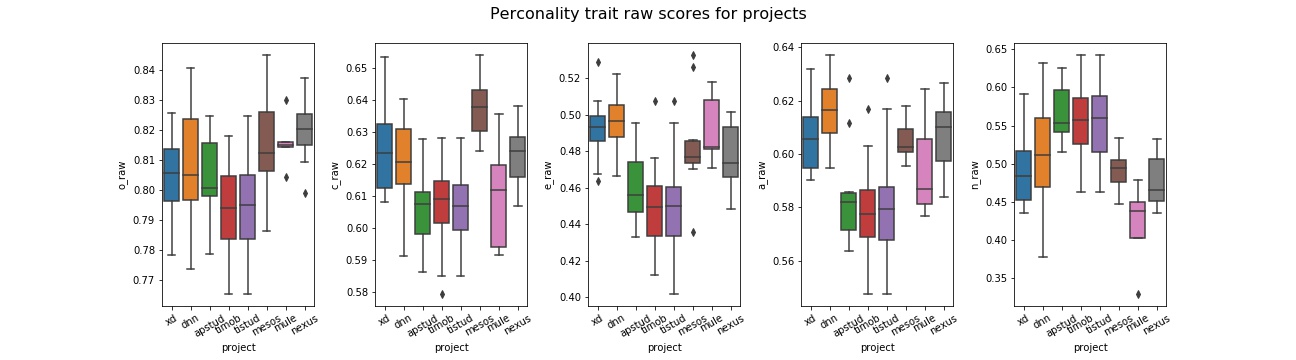


Figure 3. boxplots of users` personality trait raw scores and percentiles per project

When choosing the one measurement to use among these two, raw scores were opted within the research, based on the following advantages:

* IBM uses sample population for normalizing the scores, that may not be applicable and proportionate to the developers`. On the contrary, raw scores can be normalized using the results of the users of inputted dataset, resulting in the relatively more or less open/conscientious/extravert/agreeable/neurotic scores that is comparable within the given dataset.
* Results of raw scores were normally distributed, unlike the normalized scores, that were imbalanced, as shown on Figure 2.

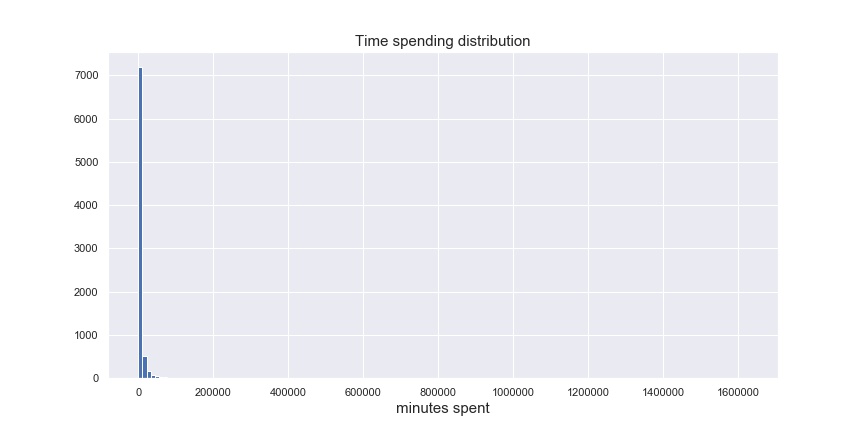
## Computation of metrics

### Actual development time metric

For calculating **Time** metric, the logics were applied on Jira issues change log dataset. First, was defined the time spent per task by a developer: time that is 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 change log, that have 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 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 on Figure 4 shows the distributions of spent minutes by the developers. Noticeably, there are outliers that puts the vast majority of the records into the first few bins.

Figure 4. Histogram of time spending by developers

To have the better picture, need to filter out the records with minutes spent greater than 5000 and less than 1, the result is on the Figure 5. Now it is visible that although the majority of the records are with the 300 minutes or less, there still are considerable amount of records with higher number of minutes spent.

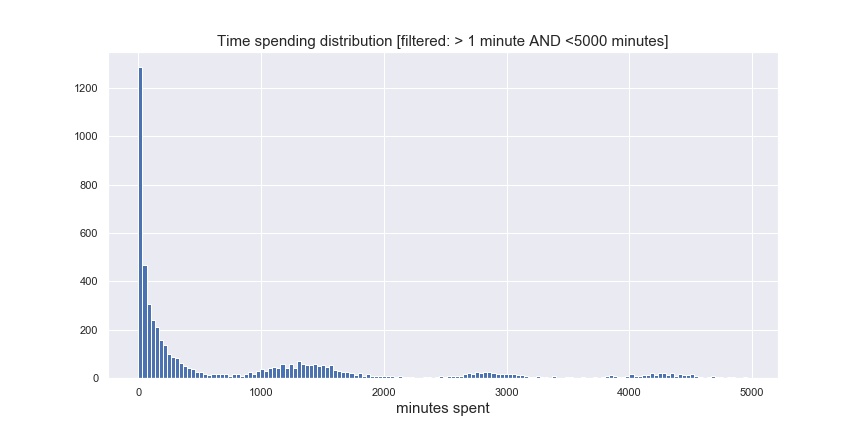


Figure 5. Histogram of time spending by developers on a subset of dataset.

The end-goal of getting the Time measurement is to have the categorical variables. The top and bottom threshold are defined for high, medium and low 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 time spent category: tasks, that took 1 working day or less. Equal to 8 working hours, and equal to 480 minutes spent, respectively.

Medium time spent category: 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 time spent category: tasks, that took more than a working week (2400 minutes spent) to complete.

Out of the whole dataset of time metric calculated previously (8093 rows in total), the low time spent category is 4159 rows (52%), medium time spent category holds 1418 rows (17%) and high time spent category contains 2516 rows (31%), as presented in the Table 3.

|  |  |  |  |
| --- | --- | --- | --- |
| ***time\_spending\_category*** | time\_high | time\_low | time\_medium |
| ***rows*** | 2516 | 4159 | 1418 |

Table 3. Time spent by developers

The rest of the metrics - state, estimation and prioritization are categorical, and the identical approaches are used for working on all of these metrics. For all the categorical metrics the calcularion logic used the change log dataset that is filtered to be done only by the users, that were valid for IBM Watson Personality Insights API and have got the results retrieved. The end goal of this part of data analysis was to get the same three categories for each metric, that was mentioned in Scott`s paper.

### **Task status metric**

For identifying whether the developers tend to put the tasks on hold, or gather in to-do list, or mark them done, filed `status` was ideal to use from changelog. Taking a look at the values that were set to `status`, there were 28 of such values found, that are shown on Table 3 with respective number of records.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***status*** | In Progre ss | Closed | Resolved | Done | Re opened | Open | Accepted | Reviewa ble | In Review | To Be Tested |
| ***rows*** | 11986 | 11236 | 10583 | 3201 | 3171 | 2550 | 1347 | 1174 | 1144 | 836 |
|  |  |  |  |  |  |  |  |  |  |  |
| ***status*** | In Review | Testing In Progr ess | In PR | Waiting for Review | Pull Request Submi tted | To Do | To Be Merged | Pending 3rd-Party | Waiting for Response | Planned Develop ment |
| ***rows*** | 807 | 738 | 675 | 665 | 379 | 233 | 100 | 67 | 41 | 27 |
|  |  |  |  |  |  |  |  |  |  |  |
| ***status*** | Defered | Writing | Inactive - Pending Closure | Scoped | Refine | Triaged | New | Raw |  |  |
| ***rows*** | 11 | 8 | 8 | 6 | 1 | 1 | 1 | 1 |  |  |

Table 3: jira issues statuses

They have grouped into 3 categories according to the goals of this research:

‘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';

‘inprogress’ 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'.

In technical solution, only the latest log record is taken on the task by each user. Meaning, that task and user are unique keys. changelog dataset is filtered with `field` column to be equal to `status`, and the set values are filtered within the column `toString` with the statuses mentioned above in three categories. In the end, filtered dataset contains project code, Jira issue key, user, email address and the status (‘todo’, ‘inprogress’ or ‘done’). Dataset has 21585 records in total, whereas most of them eventually get done (17073), 2315 of them have left to in progress, and 2197 have status todo, as shown on Table 4.

|  |  |  |  |
| --- | --- | --- | --- |
| ***status*** | status\_done | status\_inprogress | status\_todo |
| ***rows*** | 17073 | 2315 | 2197 |

Table 4. Jira issues statuses

### **Prioritization metric**

Priority metric is used to define how the developers are prioritizing their tasks, weather 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 5 shows the priority values within the Jira changelog and the number of rows that have assigned the given priority.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***status*** | High | Medium | Critical | Major | Low | Blocker | Minor | None | Trivial | To be reviewed |
| ***rows*** | 1978 | 1267 | 1031 | 742 | 670 | 254 | 240 | 166 | 89 | 3 |

Table 5. Jira issues priorities

These statuses have to be groupped into high, medium and low categories, 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.

Technical solution was the same as the previous metric, only the latest change log record was taken on the task by each user, making task and user the unique combination. Changelog dataset was filtered with `field` column to be equal to `priority`, and the set values are filtered within the column `toString` with the priority values mentioned above in three categories. Resulting dataset contains project code, Jira issue key, user, email address and the priority (‘high, ‘medium‘, ‘low’). Dataset has 4796 rows in total, where the low estimated tasks by users are 2872, medium estimated tasks are 2780 and high estimated are 1188 tasks, as shown on table 6.

|  |  |  |  |
| --- | --- | --- | --- |
| ***StoryPoints*** | estimation\_high | estimation\_low | estimation\_medium |
| ***rows*** | 1188 | 2872 | 2780 |

Table 6. Jira issues estimations.

### **Story estimate metric**

Estimation metric is used to show the tendency of developers to mark the tasks complexity by assigning the respective number of Story Points. By definition, story point is a measurement of difficulty of the task. In SCRUM teams there is a certain number of numeric ranges of story points that are assigned to the tasks, among them the most common approach is to use the alternative version of Fibonacci sequence. For the initial check of the story points it is necessary to merge the changelog and issues datasets. Results of the initial analysis are shown on Figure 6. On Y axis there are the list of issue types, on the x axis there are story points assigned to these tasks. With the color and the numbers on the suqares are shown the number of the respective records. The figure represents only the users that are in the scope of IBM Watson Personality Insights check.

Generally, the fibonnaci sequence based story points range uses the following numbers for estimation: 0, 0.5, 1, 2, 3, 5, 8, 13, 20, 40, 100. The heatmap on the figure 6 shows that 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.

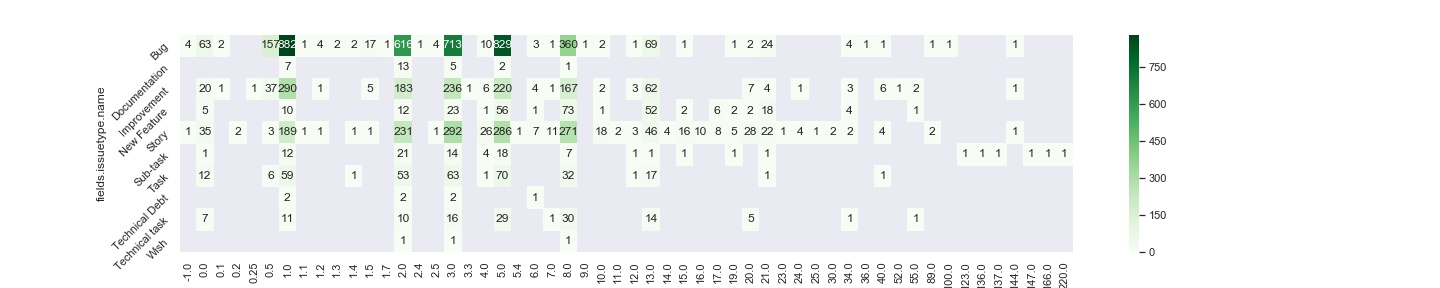


Figure 6. Heatmap of the story points and issue types.

For the consistency, the cases have to be ignored, where the story points are represented with different number than the common approach of using Fibonacci sequence alternative.

Out of these range, story point are divided into the 3 categories – low, medium, and high estimation respectively:

low estimation: 0, 0.5, 1, 2; medium estimation: 3, 5; high estimation = 8, 13, 20, 40, 100.

Following the same technical solution as for the other categorical variable metrics, result is a dataset with 6840 rows in total. Table 6 shows the representation of number of rows each estimation category.

|  |  |  |  |
| --- | --- | --- | --- |
| ***StoryPoints*** | estimation\_high | estimation\_low | estimation\_medium |
| ***Rows*** | 1188 | 2872 | 2780 |

Table 6. Story points

### Metrics from the research of E.Scott

Apart from the software developer`s personality traits research in the accordance of agile metrics, this research is also establishing the bonds with Big 5 and FLSM models, by validating the papers of E.sidiiquei and E.Scott. Scott used Felder-Silverman Learning Styles to explain the SCRUM methodology preferences and metrics. Availability to replicate the same SCRUM metrics is given in the table 2.

|  |  |  |
| --- | --- | --- |
| **Metric** | **Value** | **Comment** |
| State | to do | avalable |
| State | doing | available |
| State | done | available |
| Prioritization | low | available |
| Prioritization | medium | available |
| Prioritization | high | available |
| Estimation | low | available |
| Estimation | medium | available |
| Estimation | high | available |
| Time | low | available |
| Time | medium | available |
| Time | high | available |
| Role | Developer | Not Available in our JIRA dataset |
| Role | Support | Not Available in our JIRA dataset |
| ScrumMaster | Yes | Not Available in our JIRA dataset |
| ScrumMaster | No | Not Available in our JIRA dataset |
| Recommendations | OK | Not Available in our JIRA dataset |
| Recommendations | Ignored | Not Available in our JIRA dataset |

Table 2.

E. Scott has calculated the categorical variables of State, Prioritization, Estimation, Time, Role, ScrumMaster and Recommendations metrics in his paper. Out of these, State, Prioritization, Estimation and Time can be retrieved from the Jira issues and changelog datasets that come from open source, however Role, ScrumMaster and Recommendations are not available.

Firstly, the scope of the metric calculations are shortened to take into account only the users that have been checked on IBM Personality Insights on the previous steps. All the changelog records performed by the other users than the ones mentioned previously, got filtered out.

## Association rules mining

Now that the dataset of Software Developers metrics and their respective developers` personalities is ready, they can be used to proceed to association rules. The end goal of this chapter is to get the association rules with two variables – 1) personality trait and 2) metric.

First step towards 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 mean raw score of all the samples in the dataset of retrieved personality scores of developers.

In overall, out of the 100 developer profiles that has been analysed during this research have the personality trait results as shown on the Table x.

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

Table x. Number of developers falling into each category of the personality traits.

For each trait, separately, it`s needed to make the associations with all the metrics. This way, can be detected the association of one personality trait and the agile metric. First step for that is to get the dataset of each metric calculated in the previous chapters – actual development time, story estimate, prioritization, task status:

|  |  |  |  |
| --- | --- | --- | --- |
| ***dataset of metric:*** | **Nbr of Rows** | **Nbr Of unique users** | **Nbr of unique issues** |
| *Story Estimate* | 6840 | 76 | 6677 |
| *Prioritization* | 4796 | 81 | 4464 |
| *Task status* | 21585 | 97 | 12283 |
| *Actual development time* | 8093 | 80 | 7723 |

And join each of these metric datasets with personality traits dataset, so that it will have corresponding five personality trait result for each row of metric dataset. The resulting dataset for each metric has the following structure:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| user | key | project | Metric value | openness value | conscientiousness value | extraversion value | agreeableness value | neuroticism value |

Goal is to know associations of metric / personality, therefore, user, key and project columns are discarded.

Next, all the four metric datasets are combined. Structure stays the same.

Finally, the subsets of columns for each personality trait is done, meaning that the result is an association rule for each personality trait.

Since there are 4 agile metrics studied within this research and each of them is three dimensional, it gives total of 12 metric values. There are 2 dimensions for each personality trait – Positive and Negative, therefore, there are in total, 24 possible association rules, as show on the case of Neuroticism trait on table x:

|  |  |  |  |
| --- | --- | --- | --- |
| Rule Id | **trait value** | **metric** | **metric value** |
| 1 | No\_neuroticism | estimation | high |
| 2 | low |
| 3 | medium |
| 4 | priority | high |
| 5 | low |
| 6 | medium |
| 7 | status | done |
| 8 | inprogress |
| 9 | todo |
| 10 | time | high |
| 11 | low |
| 12 | medium |
| 13 | Yes\_neuroticism | estimation | high |
| 14 | low |
| 15 | medium |
| 16 | priority | high |
| 17 | low |
| 18 | medium |
| 19 | status | done |
| 20 | inprogress |
| 21 | todo |
| 22 | time | high |
| 23 | low |
| 24 | medium |

Table x: Association table of Jira metrics and Neuroticism.

Now that there is the association rules of personality traits and metrics as predefined in the beginning of this chapter, it is possible to get the actual results of these associations.

In this research two main association rules measurements are used for finding out the relationships of the metrics and personality traits: Support and Confidence. Support is used to measure how frequent the itemset appears in dataset, and the confidence measures the conditional probability of the of occurrence of consequent given the antecedent.

Additionally, lift values are also used to compare the rule confidence with the expected confidence.

Having the association rules and the relationship measurements defined, it is ready to be applied the association rules mining techniques to detect the important patterns.

During the frequent pattern mining and association rules analysis, apriori algorithm from python `apyori` library was chosen to filter the frequent rules and only show the reliable and important rules. 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 dataframe.

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 ∪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 items as the consequent, and 24 inverse rules - 12 metric items as the antecedent associated with 2 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 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 `estimation low` has two possible consequents – ‘Yes neuroticism` and `No Neuroticism`:

* 1. Confidence (`estimation low`=>`Neuroticism Yes`) = Support (`Estimation low`∪ `Neuroticism Yes`) / Support (`Estimation low`)
* 2. Confidence (`Estimation low`=>`Neuroticism No`) = Support (`Estimation low`∪ `Neuroticism No`) / Support (`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 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 Yes` has 12 possible consequents - `Estimation` low/medium/high, Prioritization low/medium/high, status todo/inprogress/done, development time low/medium/high:

* Confidence (` Yes Neuroticism` => `Estimation low`) = Support(`Yes Neuroticism` U `Estimation low`) / Support (`Yes Neuroticism)
* Confidence (` Yes Neuroticism` => `Estimation medium`) = Support(`Yes Neuroticism` U `Estimation medium`) / Support (`Yes Neuroticism)
* Confidence (` Yes Neuroticism` => `Estimation high`) = Support(`Yes Neuroticism` U `Estimation high`) / Support (`Yes Neuroticism)

And the same way for all the 12 metric value. Evidently, all these 12 rules confidence values gives 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 first example, when the antecedent is a jira metric, the reliable rule should have the confidence greater than 0.5, and in the other case, when the antecedent is personality trait, the reliable rule should have confidence greater than 0.08.

## Research results

## Correlation

Before conducting the association rules mining, the correlation between the Big 5 personality trait variables was analyzed (Table 10). Correlation matrix shows several notable results:

First of all, neuroticism is negatively correlated to the other traits, while all the other traits are positively correlated to each other.

Furthermore, correlation scores vary from (+/-) 0.4 to (+/-)0.6 in majority of the cases, there are couple of exceptions here: Extraversion – agreeableness, that has higher 0.723 correlation, that can be considered as reasonable, as more extravert people are – more they agree to other`s opinions and ideas. Another is agreeableness – Neuroticism, -0.271. Negative sign is logical – neurotic people tend to be less agreeable, and lower correlation number shows, that there`s a weak causality in that.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **openness** | **conscietiousness** | **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 10. Big 5 Personality Traits correlation matrix

All the association rules mentioned in the above paragraph have been re-studied with related big five personality traits instead of FSLM model. The results show, that the majority of the associations from Scott`s paper are valid within this research as well.

## Association rules results

The association rule graphs in the next five chapters show only the frequent itemsets` association rules with support, confidence and lift in the context of each personality trait. The filter criteria of the rule significance is applied as descried in the former chapter.

On Important note in the following personality traits association rules graphs is that 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 rows), and moreover, out of that ~22k rows, 17k are specifically with the `done` value.

#### 4.8.2.1 Openness

Graph x show the association rules of the Openness personality trait. Open developers usually give low estimation to their tasks and mark the tasks done (openness:Yes => estimation:low, openness:Yes => status:done), while the less open developers tend to spend low time on actual development and also mark the task status to done (openness:No => development time: low, openness:No=> 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 (estimation:low => openness:Yes), furthermore, the confidence of both, nominal and inversive rule is high enough to conider it reliable, and finally, the lift value is positive and high, meaning that low estimation and openness trait are positively correlated. But in 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:No => development time: low) also has the sufficient support and confidence for the trustworthyness of the rule. Notably, here the lift value is very close to 1, yelding 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:No).

Among the other significant rules, the notable is rule priority:high=>openness:No, meaning that developers that set the task priority to high, are less open people.

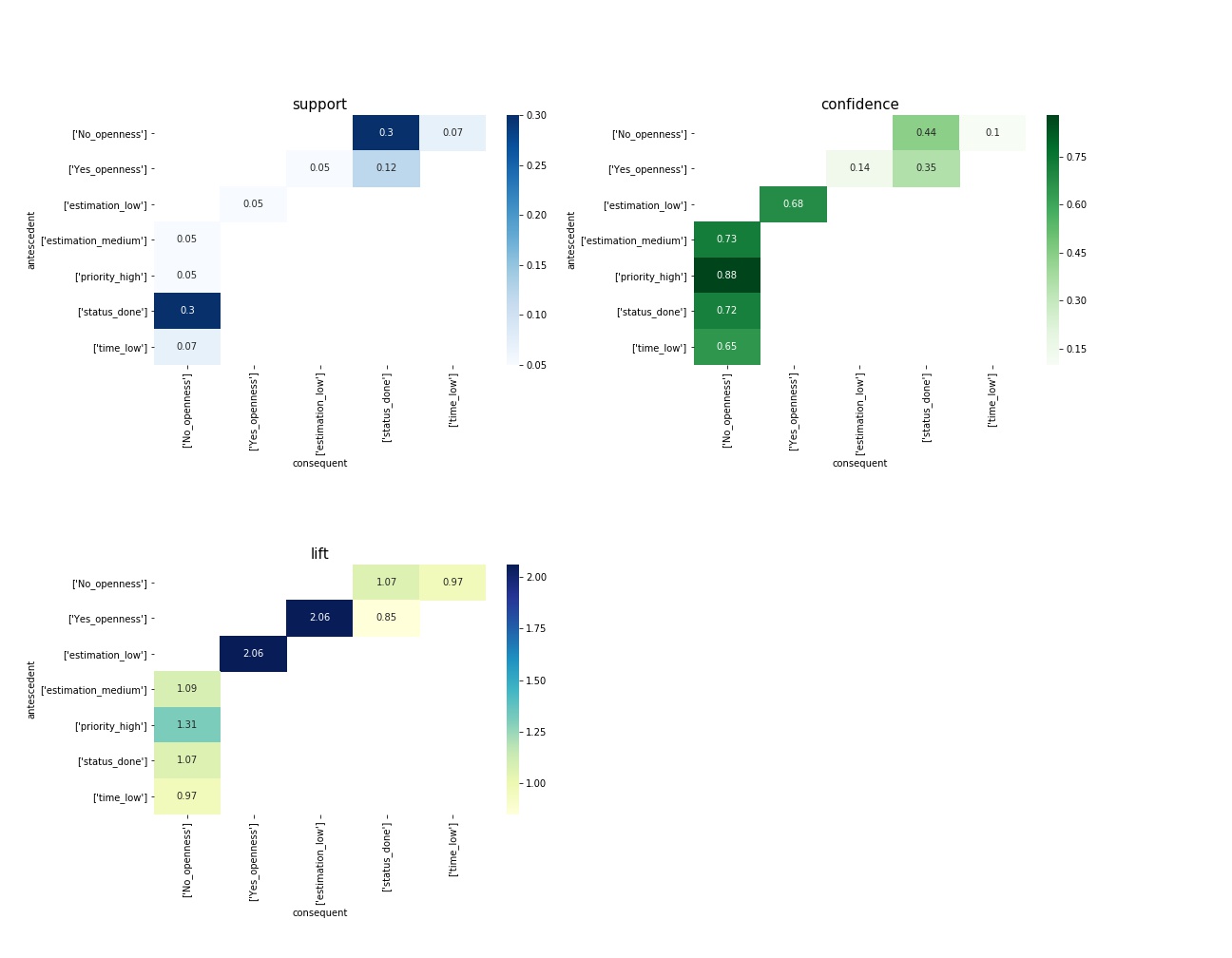


Figure x: Openness personality trait and related assocaition rules.

#### 4.8.2.2 Conscientiousness

Association rules of Conscientiousness personality trait is show on the figure x. Likewise the openness trait, on one hand, conscientiousness is also associated with low estimation and done task status (conscientiousness:Yes => estimation:low, conscientiousness:Yes => status:done), additionally, positive value of this personality trait has significant association with low development time. These relations show, that more conscientious developers assign 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:Yes => 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:Yes => development time:low has not enough confirmation to be considered as reliable.

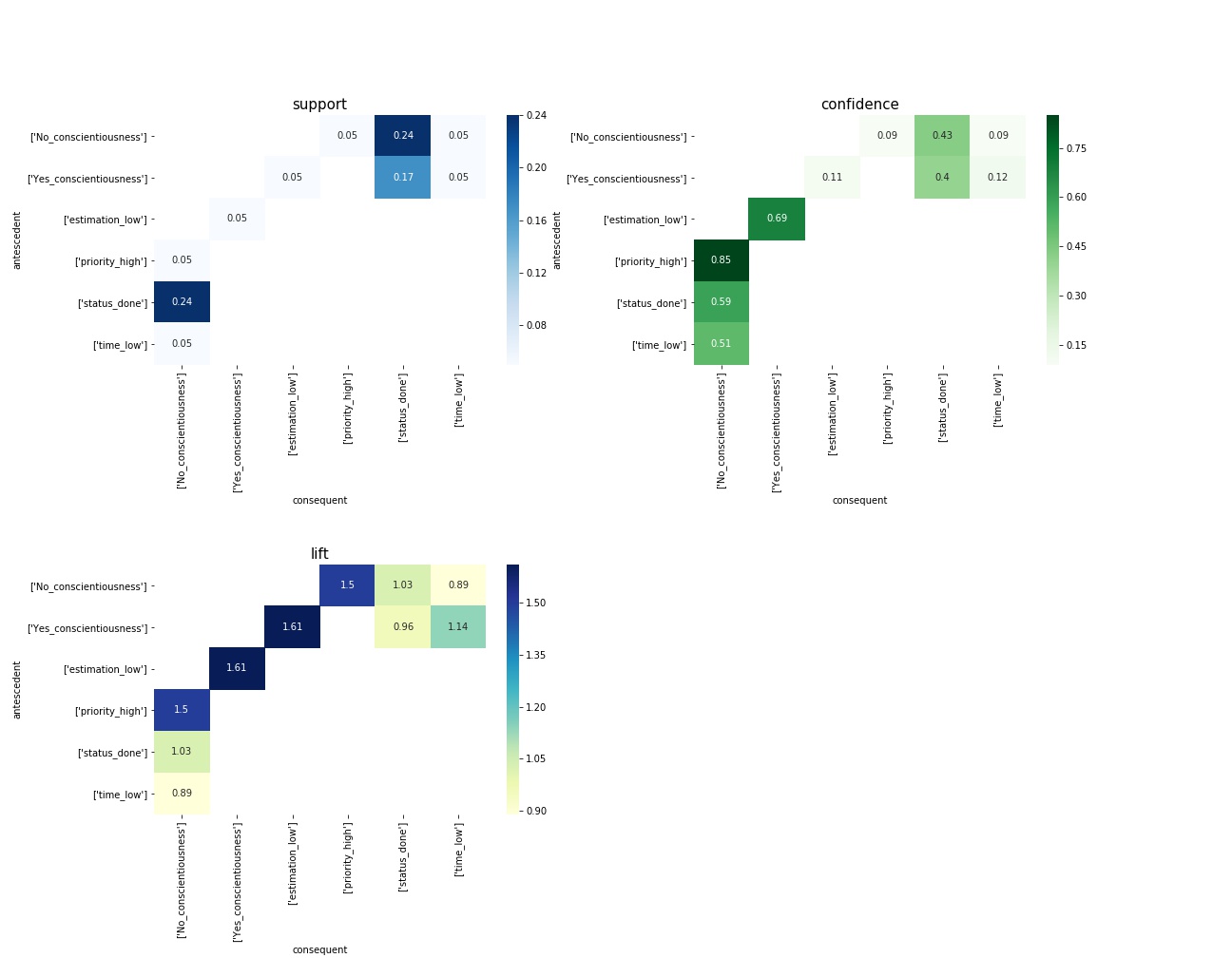
On the other hand, less conscientious developers also set high priority, spend low time and mark done status on the development of the tasks, alike the negative valued variable of former personality trait. All these three rules have the sufficiently supported inversive rule as well.

Figure x: Conscientiousness personality trait and related assocaition rules.

#### 4.8.2.3 Extraversion

As the figure x 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 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 – estimation:medium => extraversion:No, meaning that among the developers that estimate task 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.

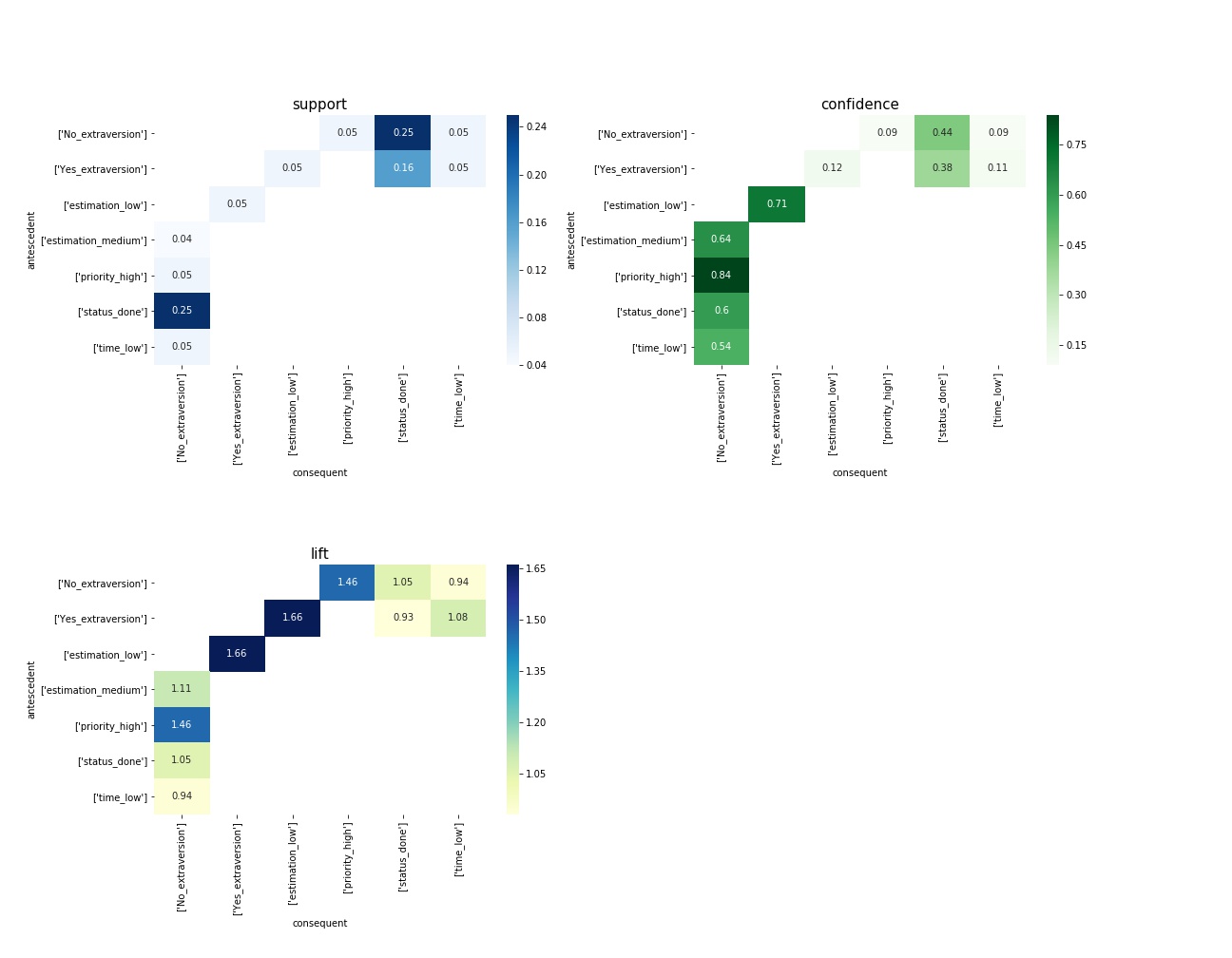


Figure x: Extraversion personality trait and related assocaition rules.

#### 4.8.2.4 Agreeableness

Figure x shows the association rules related to agreeableness personality trait. Agreeable developers tend to set the low story points to their issues (agreeableness:Yes => estimation: low) and mark them done afterwards (agreeableness:Yes => 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 task as done. All three respective inverse rule (priority:high => agreeableness:No, time:low => agreeableness:No, status:done => agreeableness:No) are also significant, moreover there is a notably high confidence of the rule priority:high => agreeableness:No, 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.

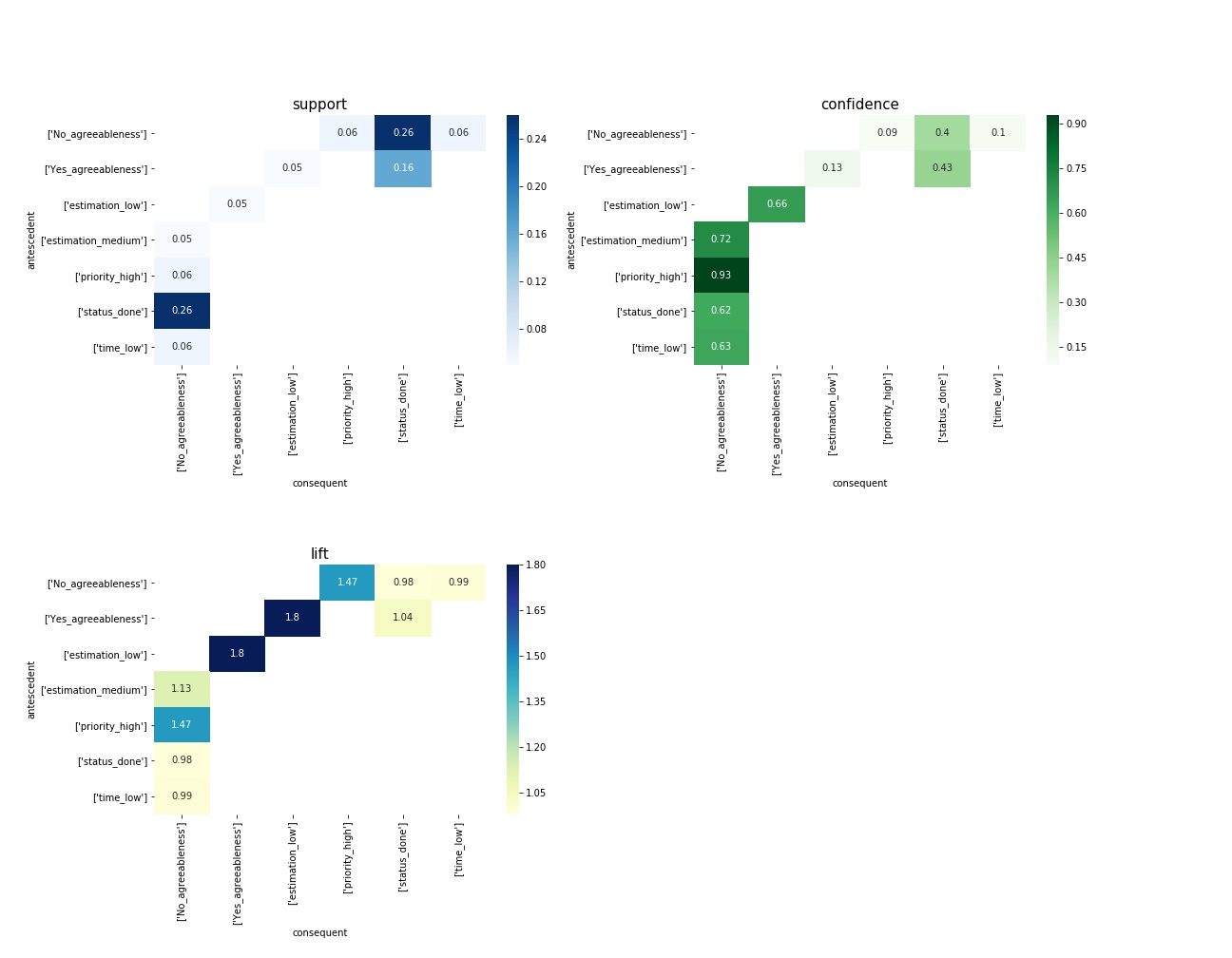


Figure x: Agreeableness personality trait and related assocaition rules.

#### 4.8.2.5 Neuroticism

Figure x shows Neuroticism personality traits associations with jira metrics. According to the plot, neuroticism is associated with high prioritization, low development time and done task status. (neuroticism:Yes => prioritization: high, neuroticism:Yes => development time: low, neuroticism:Yes => status: done). The respective inverse association rules are also supported by the 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 estimation:low => neuroticism:No 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.

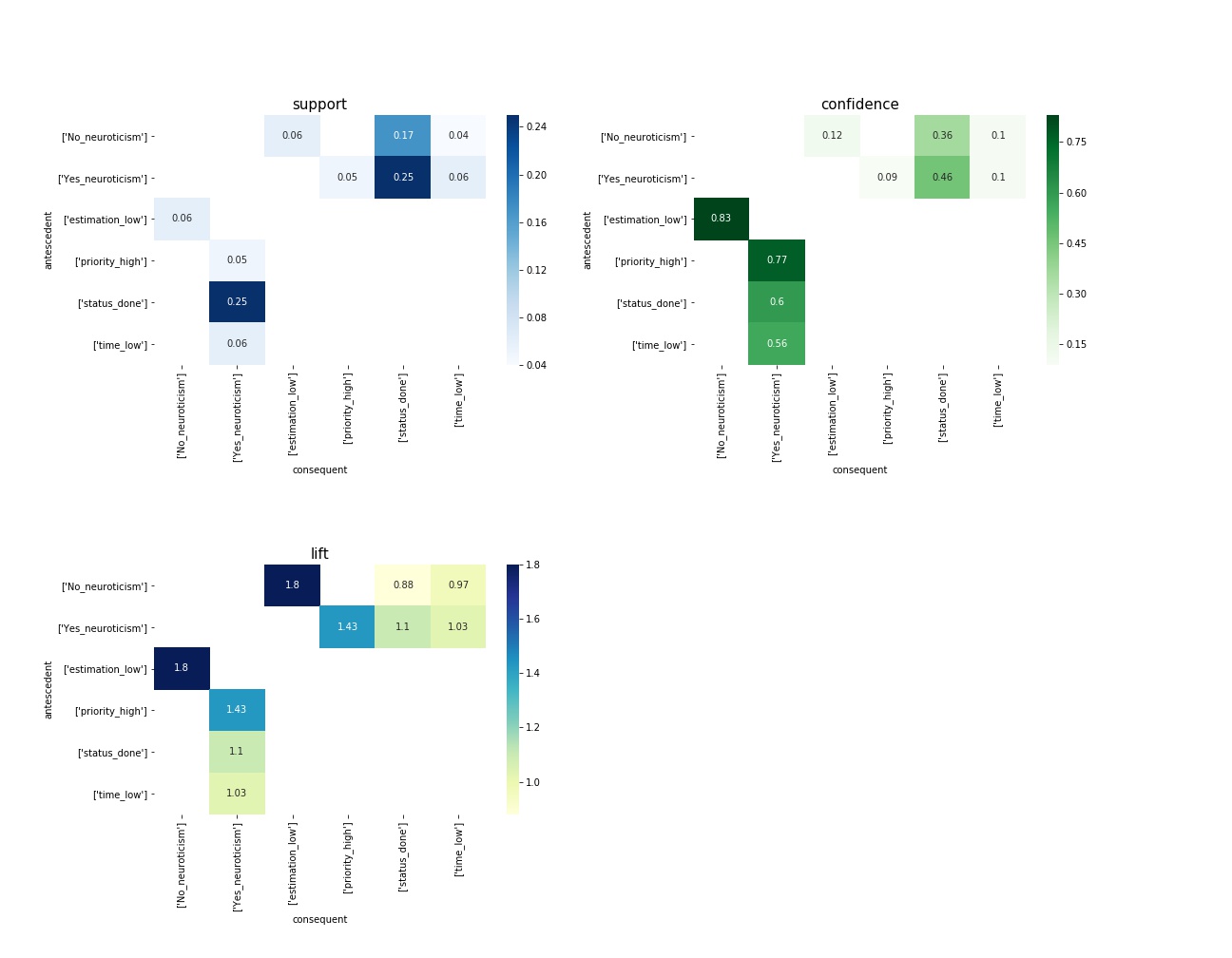


Figure x: Extraversion personality trait and related assocaition rules.

## 4.8.3 FSLM / Big 5 / Agile validation

On one hand, In his research, E.Scott studied the students` preference on SCRUM metrics. He has beforehand studied the learning styles of these students according to the Felder-Silverman Learning Styles Model (FSLM), and after analyzing the students SCRUM performance, he created the association rules of SCRUM metrics and FSLM learning styles.

On the other hand, E.Siddiquei has done the research of the students personalities and learning styles. He used, Felder-Silverman Learning Styles model for the learning styles assessment, and Big 5 Personality traits questionnaire for personality type detection. As a result, he got the correlation matrix of FSLM learning styles and Big 5 Personality traits.

Taking a closer look to these two on Figure 7, there is a mutual variable that both papers share – FSLM learning styles.

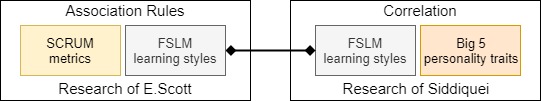


Figure 7.

Both of these papers are done by the research of the students. To add more confidence in the results of these papers, would be beneficial to check these two out of the academia, specifically on the actual teams of the developers. Furthermore, not to replicate the same research, it is possible to use the common link of these two papers, and check the association on the corresponding entities. Visually it can be represented as shown on Figure 8.

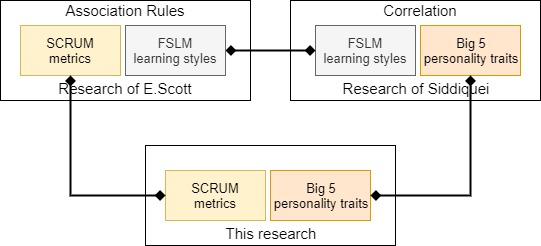


Figure 8.

One of the research question of this paper is to take the association rules of FSLM model and SCRUM metrics, get the correlation of FSLM and Big 5 model by the paper of Siddiquei, and then check the former associations with Big 5. Proving the association rules of SCRUM metrics with big5 personality traits will create triangular relation of FSLM, Big5 and SCRUM, as shown on Figure 8 and will strengthen the link between these models by the real-world team project analysis.

Let`s take a closer look at the association rules that was studied to be significant by E.Scott, given in Table 7. Antecedent and consequent of the association rules are either, Felder-Silverman learning model (FSLM) variables or SCRUM metrics variables.

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

Table 7.

In his paper, Siddiquei studies the correlation between FSLM learning styles and the Big 5 Personality traits. Among the correlated variables, he points out the ones that are significant, given on the table 8.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Variable** | **Value** | **Trait** | **Correlation** | **Correlation value** |
| Perception | intuitive | Agreeableness | Negative | -0.268 |
| Perception | intuitive | Conscientiousness | Positive | 0.247 |
| Perception | sensing | Agreeableness | Positive | 0.261 |
| Perception | sensing | Conscientiousness | 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 | sequential | Neuroticness | Negative | -0.199 |
| Understanding | global | Neuroticness | Positive | 0.199 |

Table 8.

Based on these two researched, there is an opportunity to check the same association rules as Scott has done, but this time, with Big 5 personality traits: each FLSM learning style are replaced with correlated Big 5 Personality trait.

AS the results, wthere are the same 21 association rules on Table 9 consisting of Metrics researched by E.Scott and Big 5 personality traits corelated to FSLM learning styles as researched by Siddiquei:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **N** | **Original Rule by Scott** | | **Correlated big5 trait rule 1** | | **Correlated big5 trait rule 2** | |
| ***antescedent*** | ***consequent*** | ***antescedent*** | ***consequent*** | ***antescedent2*** | ***consequent2*** |
| 1 | priority: high | Perception: sensing | priority: high | Yes: agreeableness | priority: high | Yes: conscientiousness |
| 2 | priority: low | Perception: intuitive | priority: low | No: agreeableness | priority: low | Yes: conscientiousness |
| 3 | time: low | Perception: intuitive | time: low | No: agreeableness | time: low | Yes: conscientiousness |
| 4 | time: high | Perception: sensing | time: high | Yes: agreeableness | time: high | Yes: conscientiousness |
| 5 | status: done | Perception: intuitive | status: done | No: agreeableness | status: done | Yes: conscientiousness |
| 6 | Perception: intuitive | time: low | No: agreeableness | time: low | Yes: conscientiousness | time: low |
| 7 | Perception: intuitive | estimation: high | No: agreeableness | estimation: high | Yes: conscientiousness | estimation: high |
| 8 | estimation: high | Perception: intuitive | estimation: high | No: agreeableness | estimation: high | Yes: conscientiousness |
| 9 | Perception: sensing | time: high | Yes: agreeableness | time: high | Yes: conscientiousness | time: high |
| 10 | Perception: sensing | estimation: high | Yes: agreeableness | estimation: high | Yes: conscientiousness | estimation: high |
| 11 | estimation: high | Processing: active | estimation: high | Yes: extraversion | estimation: high | Yes: openness |
| 12 | time: low | Processing: active | time: low | Yes: extraversion | time: low | Yes: openness |
| 13 | priority: low | Processing: active | priority: low | Yes: extraversion | priority: low | Yes: openness |
| 14 | Processing: active | time: low | Yes: extraversion | time: low | Yes: openness | time: low |
| 15 | Processing: active | estimation: high | Yes: extraversion | estimation: high | Yes: openness | estimation: high |
| 16 | status: done | Processing: active | status: done | Yes: extraversion | status: done | Yes: openness |
| 17 | status: todo | Processing: active | status: todo | Yes: extraversion | status: todo | Yes: openness |
| 18 | status: todo | Processing: reflexive | status: todo | Yes: extraversion | status: todo | No: openness |
| 19 | Processing: reflexive | status: todo | Yes: extraversion | status: todo | No: openness | status: todo |
| 20 | Processing: reflexive | estimation: high | Yes: extraversion | estimation: high | No: openness | estimation: high |
| 21 | status: done | Understanding: global | status: done | Yes: neuroticism |  |  |
| 22 | status: todo | Understanding: sequential | status: todo | No: neuroticism |  |  |
| 23 | time: low | Understanding: sequential | No: neuroticism | sequential |  |  |
| 24 | Understanding: global | status: done | Yes: neuroticism | status: done |  |  |
| 25 | Understanding: global | estimation: high | Yes: neuroticism | estimation: high |  |  |
| 26 | priority: low | Understanding: sequential | priority: low | No: neuroticism |  |  |
| 27 | Understanding: sequential | status: todo | No: neuroticism | status: todo |  |  |
| 28 | Understanding: sequential | time: low | No: neuroticism | time: low |  |  |
| 29 | Understanding: sequential | estimation: high | No: neuroticism | estimation: high |  |  |
| 30 | priority: high | Understanding: sequential | priority: high | No: neuroticism |  |  |

Table 9. Combined association rules of E.Scott and correlated Big 5 Personality traits.

## 

## General Use

# Discussions

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

Firstly, the developers tend to mark their tasks done within all the groups of the personality trait and the result. That means they all share in common the likelihood of getting things done. This should be considered as the positive-stated finding, that the developers do not leave their tasks unfinished.

Secondly, developers within all the traits like estimating the task 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 low number of story points to their tasks.

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 less conscientiousness people are less organized, not fully disciplined and more chaotic, they may not be fully realizing the impact and dependences of the task, and only working on the core idea of the issue without realizing 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 explanation of the behavior of the less agreeable people who have less altruism, morality and modesty.

One more evident association rule that is common within the developers with the less scores of Conscientiousness, Extraversion, Agreeableness and more score of Neuroticism is, that they 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 a 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.

As the majority of the results of the association rules show, it is evident that 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 comparing to the more constructive personality traits.

The former statement is also backed in the correlation matrix of the developers` personalities, as discussed in the respective chapter within this research. As the matrix show, the same four personalities have positive correlation with each other, while neuroticism has the negative correlation with all the other traits.

# Conclusions

# References

x

|  |  |
| --- | --- |
| [1] | Matemaatika-informaatikateaduskonna nõukogu. (2014, jaanuar) Tartu Ülikooli matemaatika-informaatikateaduskond. [Online]. <http://www.math.ut.ee/et/oppimine/loputoode-kirjutamise-kaitsmine-juhend> |
| [2] | Council of the Faculty of Mathematics and Computer Science. (2014, January) University of Tartu Faculty of Mathematics and Computer Science. [Online]. <http://www.math.ut.ee/en/studies-2/guide-writing-and-defense-32-curricula-graduation-thesis> |
| [3] | A. J. Wilkins and M. I. Nimmo-Smith, "The clarity and comfort of printed text," *Ergonomics*, vol. 30, no. 12, pp. 1705-1720, 1987. |
| [4] | Jeremy Foster and Peter Coles, "An Experimental Study of Typographic Cueing in Printed Text," *Ergonomics*, vol. 20, no. 1, pp. 57-66, 1977. |
| [5] | Andrew Dillon, "Reading from paper versus screens: a critical review of the empirical literature," *Ergonomics*, vol. 35, no. 10, pp. 1297-1326, 1992. |
| [6] | Stephen A. Bernhardt, "Seeing the Text," *ACM SIGDOC Asterisk Journal of Computer Documentation*, vol. 16, no. 3, pp. 3-16, September 1992. |
| [7] | Kimberly Martin. (2012, November) Self-pub by Jera Publishing. [Online]. <http://www.self-pub.net/blog/common-mistakes-made-when-writing-a-book-in-microsoft-word/> |
| [8] | Nathan. (2009, April) Office Blogs. [Online]. <http://blogs.office.com/2009/04/29/bibliography-citations-102-building-custom-styles/> |

x

Appendix

1. Glossary

|  |  |
| --- | --- |
| Caret  The bar (or other symbol) marking the active editing point. | Sisestusmärk  Märk, mis märgib teksti sisestamise asukohta. |
| Template  A gauge, pattern, or mold, commonly a thin plate or board, used as a guide to the form of the work to be executed. | Mall  Näidik, muster või valuvorm, mis esitab täitmisele võetava töö struktuuri. |

1. License

**Non-exclusive licence to reproduce thesis and make thesis public**

I, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_,

(*author’s name*)

1. 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,

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_,

(*title of thesis*)

supervised by \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

(*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.

*author’s name*

***dd/mm/yyyy***