UNIVERSITY OF TARTU
Institute of Computer Science
Software Engineering Curriculum

Tanel Tõemets

Analysing the Quality of User Stories in Open Source Projects

Master's Thesis (30 ECTS)

Supervisor:
Ezequiel Scott, PhD

Tartu 2020

Analysing the Quality of User Stories in Open Source Projects

Abstract:

Agile Software Development has become highly popular over the last two decades. Together with the increase of popularity amount of scientific research on this topic has also increased. This research concentrates on one component from Agile Software Development: User Stories. In a recent paper quality framework for User Stories was proposed together with a tool implementing the framework. The proposed framework and tool make it possible to analyse the quality of User Stories from a new viewpoint. The main goal of this master thesis is to find if it is possible to forecast the quality of User Stories for monitoring purposes. Another goal is to find what is the relationship between User Story quality and other aspects of software development. Aspects studied in this thesis are the number of bugs, rework, and delays. The conducted analysis considers 10 open source agile software projects from which 8 are included in the analysis. Results of analysing the User Stories of these 8 projects revealed that it is possible to forecast the quality of User Stories. On average developed forecast models were 94.5% accurate when producing predictions for the next 15 sprints. Regarding User Stories quality relationship to other software development aspects, several interesting patterns were found. In short, a decrease in User Story quality scores reflected in the increase in the number of bugs after approximately 3 to 4 weeks. The same happened for rework but the relationship took longer to reveal (approximately 4 to 6 weeks). Correlation of delays was more spread out meaning that the quality of User Stories was leading the interaction for a longer time.

Keywords:

User Story, Agile Software Development, quality assurance, time series analysis

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

Kasutuslugude kvaliteedianalüüs avatud lähtekoodiga arendusprojektides

Lühikokkuvõte:

Viimase kahe kümnendi jooksul on agiilne tarkvaraarendus palju populaarsust kogunud. Koos agiilse tarkvaraarenduse populaarsuse tõusuga on lisandunud ka teemat käsitlevaid teadustöid. Antud töös keskendutakse kasutuslugudele, mis on üks agiilsest tarkvaraarendusest pärinev komponent. Hiljutises uurimuses pakutakse välja raamistik kasutuslugude hindamiseks. Koos raamistikuga on välja töötatud raamistikku realiseeriv tööriist. Nimetatud raamistik koos tööriistaga võimaldab kasutuslugusid analüüsida uuest küljest. Antud magistritöö põhieesmärgiks on välja selgitada kasutuslugude kvaliteedi ennustamise võimalikkus, et seeläbi jälgida projekti kulgu. Töö teine eesmärk on leida võimalikke seoseid kasutuslugude kvaliteediskooride ning teiste tarkvaraarendusega seotud aspektide vahel. Antud töös uuritavad aspektid on vigade arv, ümbertegemiste arv ning tähtaegade ületamiste arv. Analüüsis on esialgu uuritud 10 agiilset avatud lähtekoodiga tarkvaraarendusprojekti, millest 8 on kaasatud läbiviidud analüüsi. Nende 8 projekti andmete analüüsi tulemuste põhjal saab väita, et kasutuslugude kvaliteedi ennustamine on võimalik. Keskmiselt ennustasid välja töötatud ennustusmudelid õigesti 94,5% järgimise 15 sprindi kvaliteediskooridest. Lisaks leiti kasutuslugude kvaliteedi ja teiste tarkvaraarenduse aspektide vaheliste suhete uurimisel mitmeid huvitavaid mustreid. Kasutuslugude kvaliteedikskooride vähenemine peegeldus vigade arvu tõusus umbes 3 kuni 4 nädala möödudes. Sama muster eksisteeris ka ümbertegemiste arvu puhul, kuid seal võttis seose avaldumine kauem (4 kuni 6 nädalat). Tähtaegade ületamiste puhul oli korrelatsioon laialt jaotunud, mis näitas, et kasutuslugude kvaliteediskoorid olid koostoime puhul juhtivas rollis.

Võtmesõnad:

Kasutuslugu, Agiilne tarkvaraarendus, kvaliteedi tagamine, aegridade analüüs

CERCS: P170 Arvutiteadus, arvanalüüs, süsteemid, kontroll

Contents

1 Introduction

Correctly defining and understanding what software system is supposed to do is vital to any software project's success. Many methods try to minimize the chance of communication errors resulting in project failure. For example, creating use cases or writing detailed requirement documents. The downside of these methods is that there is a big chance of different stakeholders interpreting the requirements differently. Agile Software Development proposes writing requirements in the form of User Stories as a possible solution for this problem. This practice helps to save time by avoiding excessively detailed requirement documents. Also, the use of User Stories supports avoiding the possibility of requirement documents becoming the goal itself that takes the focus away from the actual software development [1].

When User Stories are used, requirements are written in short texts with a strict structure that describes some functionality of the software, the stakeholder who requires it, and the benefit of having that functionality. The most widely known template for writing User Stories was popularized by Mike Cohn [1] and proposes the following syntax: "As a ⟨role⟩, I want ⟨goal⟩, [so that ⟨benefit⟩]".

Widespread interest and actuality of agile research are highlighted by the steadily increasing amount of scientific publications addressing the Agile Software Development [2]. Since User Stories were introduced, they have become a popular method for capturing requirements in agile software projects [3, 4, 5]. A well-known survey, VersionOne from 2019 [6], found that 97% of respondents' organizations used agile development methods. An earlier survey [3] found that the usage of agile development methodologies has almost doubled from 2008 to 2013 becoming the most popular software development life cycle. With the increase of popularity in agile development methodologies the popularity of writing requirements in the form of User Stories has also grown [3]. Another study [7] about the use and effectiveness of User Stories states that most of the software professionals who responded were positive about using User Stories and found them useful.

Although User Stories have a strict structure, they are still written using natural language and therefore their quality can vary. Recent research defined a Quality User Story (QUS) framework [8] and Quality User Story Artisan (AQUSA) open source software tool [8, 10] to automatically assess the quality of User Stories. The use of quality guidelines, such as the INVEST criteria has shown to have a remarkable effect on respondents' attitudes towards User Stories [9]. Respondents who used INVEST quality guidelines believed that using User Stories has a much more positive effect on productivity and quality than others. Although this can indicate that using quality guidelines for User Stories can help to achieve better productivity and deliver better software [7], there is a lack of empirical evidence supporting that claim. These findings are strongly connected with the purpose of the thesis to find if poor quality User Stories are related to other aspects of software development. In particular, the theme of this master thesis is to analyse the quality of the User Stories, based on the QUS framework and AQUSA tool [8, 10], on data from real-life agile software projects to understand how the quality of User Stories change over time, find if it is possible to forecast the change of User Story quality over time, and how the User Story quality is related to the number of bugs, rework done, and delays.

1.1 Problem statement

Poor requirements quality has a severe effect on software projects success. Faulty requirements affect the whole software project. It is widely known that requirement errors found in the later phase of the software development process cost significantly more than faults

found early, during the requirements engineering. Faulty requirements often cause the projects to exceed deadlines, increase the amount of rework and product defects [11]. To make matters worse, ensuring high-quality requirements can be challenging as it is difficult to track and measure automatically [12].

From those challenges comes the main motivation of the research: as User Stories are a popular way for capturing requirements in agile software projects [3, 4, 5], it is valuable to find how the quality of User Stories is related to other aspects of the software development and therefore influence the success of software projects. In particular, it is analysed how the quality of User Stories is related to the number of bugs, rework done, and delays. It is also analysed how the quality of User Stories can change over time and determined if it is possible to forecast the quality of User Stories. These findings will enable software development teams to monitor the quality of their User Stories during agile software projects and allow them to predict User Story quality for the next period. This in turn would enable them to use the predicted quality value and the confidence interval to set standards for quality.

Therefore, the first research question (RQ) concentrates on finding how the quality of User Stories change during real-life agile software projects and discovering whether it is possible to forecast the change of User Story quality for monitoring purposes. The second research question checks how the quality of User Stories is related to different aspects of the software. Hence, the research questions are phrased followingly:

- RQ1: Is it possible to forecast the change of the quality of User Stories for monitoring purposes?
- RQ2: What is the relationship between the quality of User Stories and other aspects of software development? The aspects studied in this thesis are:
    - Number of bugs
    - Rework done (re-opened issues)
    - Delays (based on due date)

1.2 Outline

This thesis is structured as follows. Chapter 2 describes the background of analysed topic. An overview of related works is given in Chapter 3. Chapter 4 describes the study design, including a more thorough overview of the research questions. The data collection procedures section describes data acquisition, data pre-processing, and data cleaning. Next, analysis procedures are described including descriptions of setting up AQUSA and generating defect reports, an overview of how quality was quantified, and activities performed to forecast the quality of User Stories. Next, a description of the analysis of number of bugs, rework and delays is given. The last section in Chapter 4 gives an overview of validity procedures. Chapter 5 gives an overview of the results including an overview of the study population, results regarding User Stories quality, results of forecasting the quality of User Stories, and description of cross-correlation analysis. Chapter 6 answers the research questions, describes the lessons learned and limitations of performed analysis. Finally, Chapter 7 concludes findings and presents relation to existing evidence.

2 Background

To answer the research questions, an overview of the background on given topic is reported. This includes an overview of the framework and software tool that has been chosen for the research as well as a background of time series analysis and its application to give an overview of the concepts used in further steps of the analysis.

2.1 The Quality User Story framework and tool

Recent research revises and defines a Quality User Story (QUS) framework [8] and Quality User Story Artisan (AQUSA) open source software tool [8, 10] to automatically assess the quality of User Stories. Quality User Story (QUS) framework was originally proposed by the same authors in their previous paper [13]. This section gives a short overview of this framework and tool that have been chosen for the research.

QUS proposes a set of 13 criteria to assess the quality of User Stories considering syntax, semantics, and pragmatics [8]. The framework is presented in Figure 1.

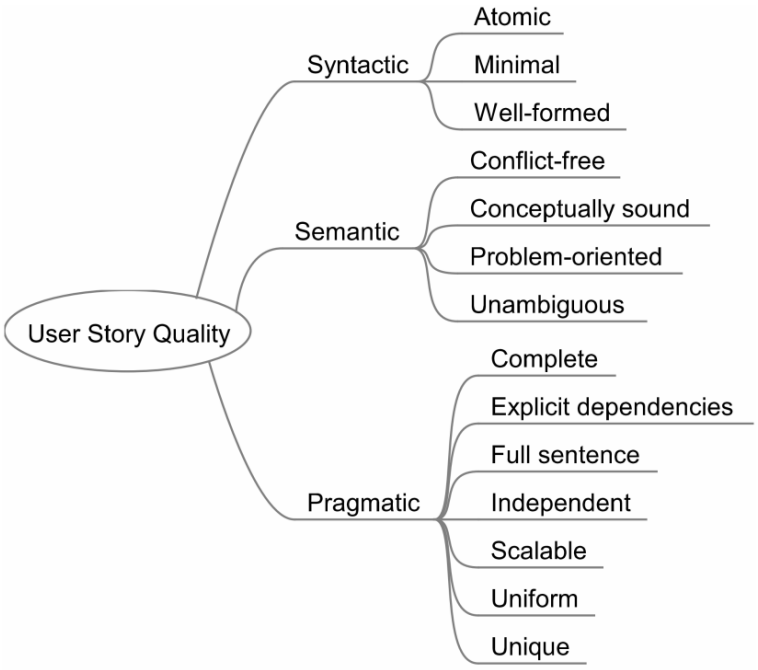


Figure 1. Quality User Story framework [13].

2.1.1 Parts of User Stories

To give a better understanding of what parts of User Stories are analysed while assessing the quality of User Stories using the QUS framework, the definition of each part of the User Story is given. User Story consists of four parts: Role, Means, Ends, and Format [8].

Role is a stakeholder who needs functionality stated in the User Story. Example of Role: "As a customer." Role can also be a persona which is an abstract named character representing some group of users [8]. Example of persona in User Story: "As a Jack."

Means captures the aim of the software functionality, an action that software needs to perform or object to which the action is made [8]. Example of mean: "I want to open a picture gallery."

End parts specify why the mean part is needed to be done but it can also capture other types of information. The article under review [8] proposed three variants for correct ends: "Clarification of means, Dependency on another functionality and Quality requirement." Different variants of ends can also occur concurrently. Example of end: "So that I can look pictures."

Format means that the User Story must follow some certain User Story template [8].

2.1.2 Definition of QUS framework 13 quality criteria

In Table 1 is presented 13 QUS quality criteria and a description of each criterion. QUS framework categorizes each criterion in the context of syntactic, semantic, and pragmatic quality. Syntactic quality considers the structure of the User Story (checks if all needed parts of the User Story are present). Semantic quality considers the meaning of the User Story as well as relations with other User Stories. Pragmatic quality groups other criteria to consider the subjective interpretation important to development project participants [8].

Table 1. QUS framework criteria definitions [8].

Criteria	Description	Individual/set
Syntactic		
Well-formed	A User Story includes at least a role and a means	Individual
Atomic	A User Story expresses a requirement for exactly one feature	Individual
Minimal	A User Story contains nothing more than role, means, and ends	Individual
Semantic		
Conceptually sound	The means expresses a feature and the ends expresses a rationale	Individual
Problem-oriented	A User Story only specifies the problem, not the solution to it	Individual
Unambiguous	A User Story avoids terms or abstractions that lead to multiple interpretations	Individual
Conflict-free	A User Story should not be inconsistent with any other user story	Set
Pragmatic		
Full sentence	A User Story is a well-formed full sentence	Individual
Estimable	A story does not denote a coarse-grained requirement that is difficult to plan and prioritize	Individual

Unique	Every User Story is unique, duplicates are avoided	Set
Uniform	All User Stories in a specification employ the same template	Set
Independent	The User Story is self-contained and has no inherent dependencies on other stories	Set
Complete	Implementing a set of User Stories creates a feature-complete application, no steps are missing	Set

The first column in Table 1 names the specific criterion. The second column gives a short explanation of the criterion and the third column specifies if this criterion considers the quality of only one (individual) or multiple (set) User Stories.

2.1.3 The Automated Quality User Story Artisan (AQUSA) tool

Article [8] also describes the architecture of the Automated Quality User Story Artisan (AQUSA) software tool that implements the previously described framework to assess the quality of User Stories.

The tool is intended to achieve recall close to 100%. This means that true positives will always be found and manual checking for all other requirements is not needed. To achieve this ambitious recall not all criteria are included in the tool. Semantic criteria are excluded completely because they analyse the meaning of User Stories which authors found not possible to implement while keeping the recall close to 100% [8].

Therefore, AQUSA tool only considers following criteria from the QUA framework [8]:

- Well-formed
- Atomic
- Minimal
- Independent (not fully implemented in AQUSA)
- Uniform
- Unique

After User Stories are inserted and analysed by the tool a report is generated presenting warnings, errors, minor issues, false positives, and perfect stories. Report of found defects is stored in the AQUSA database which needs to be set up separately [8].

2.2 Time series analysis

In this section a brief overview of time series, time series analysis, and ARIMA model [14] is given. These are the main concepts used in the further analysis of User Story quality.

A book introducing time series modelling and forecasting [14] defines time series as a series of data points arranged in time order, usually measured over successive times. Time series can contain one or more variables. Time series containing one variable is called univariate and time series with two or more variables multivariate [14].

Time series is continuous if measurements are taken every instance of time. Examples of continuous time series are temperature, pressure, etc. If measurements are taken at discrete points of time, then we have discrete time series. Examples of discrete time series are stock price, retail sales, population, etc [14].

A time series consists of four main components which are following [14]:

- Trend component
- Seasonal component
- Cyclical component
- Irregular component

Trend component shows the time series tendency to increase or decrease during a long period of time. Seasonal and Cyclic components both show periodic fluctuation over time. The difference between these two components is that Seasonal component reveals fluctuations in a shorter period with a maximum length of one year, while Cyclic component reveals fluctuations for longer periods which are longer than a year. For example, weather can cause seasonal variations in the sales of winter clothes or umbrellas. Cyclic variation is caused by longer cycles, for example business cycle. Unpredictable influences that are not regular nether repeat themselves over time are known as Irregular variations. Irregular variations are caused by unforeseen incidences like war or flood and therefore it is not possible to statistically measure the irregular fluctuation [14].

It can be assumed that the described four components of time series can either act independently or affect each other. Consequently, two mathematical models are generally used for time series analysis: Multiplicative Model is shown in Equation 1 where components of time series are assumed to act independently of each other and Additive Model shown in Equation 2 where components are assumed to affect each other [14].

Equation 1. Multiplicative Model for Time Series Analysis [14].

$$Y(t) = T(t) \times S(t) \times C(t) \times I(t)$$

Equation 2. Additive Model for Time Series Analysis [14].

$$Y(t) = T(t) + S(t) + C(t) + I(t)$$

$Y(t)$ is the value of time series at time t. $T(t)$, $S(t)$, $C(t)$, and $I(t)$ are Trend, Seasonal, Cyclical and Irregular components respectively [14].

When considering applications of time series, it can be said that it is used in a large number of various domains like business, science, engineering, etc. Time series can be used for plotting basic patterns of the data. More sophisticated use for time series is performing time series analysis which means fitting time series to a suitable model. This includes models that help to understand the data of time series and assist forecasting and simulation, which are also common uses for time series and time series analysis [14].

Time series forecasting means developing a model by analysing previously observed values. The developed model will be used for predicting future values. Time series forecasting is used in various fields as it can be used for implementing preventative measures and making strategic decisions [14].

An important concept to consider when performing time series forecasting is stationarity. When building a time series model for forecasting it is necessary that mean and variance are not dependent on time. This condition decreases the mathematical complexity of the fitted model. Further, the whole purpose of the stationarity concept is to provide us with a mathematical idea for simplifying stochastic processes. In order to perform sufficiently accurate time series forecasting the time series must be stationary. As data from real life is often non-

stationary methods like differencing and power transformation can be used for solving this problem [14].

Next, a brief overview of the Autoregressive Integrated Moving Average (ARIMA) model is given. ARIMA model is a generalization of the Autoregressive Moving Average (ARMA) model. The advantage of the ARIMA model over ARMA is that it can be used for non-stationary time series. This is important because many time series containing real-life data are non-stationary. For example, it is common for real-life data to contain trend or seasonal patterns which indicate that the time series is non-stationary. This advantage of ARIMA is achieved by making non-stationary time series stationary which in turn is achieved by finite differencing of time series [14]. For example, the differenced time series is analysed instead of the original non-stationary time series. The mathematical model of the ARIMA model is presented in Equation 3.

Equation 3. Formula of ARIMA model [14].

$$\varphi(L)(1-L)^d y_t = \theta(L)\varepsilon_t, i.e$$

$$\left(1-\sum_{i=1}^{p}\varphi_i L^i\right)(1-L)^d y_t = \left(1+\sum_{j=1}^{q}\theta_j L^j\right)$$

ARIMA model is denoted as ARIMA*(p, d, q)* where *p* stands for autoregressive, *d* for integrated, and *q* for moving parts of the model [14].

Although ARIMA can be used for non-stationary data it is not suitable for analysing data with seasonal patterns. For these cases, the SARIMA model can be used which solves the seasonality issue by using seasonal differencing in appropriate order. S in SARIMA acronym stands for seasonality. SARIMA model is denoted as SARIMA(*p, d, q*)×(*P, D, Q*) [14].

3 Related work

This chapter gives an overview of earlier studies related to the topic of the thesis. More precisely, the chapter gives an overview of papers covering the topic of User Stories quality and describes some empirical studies about how requirements can be related to the number of bugs, rework, and delays. Finally, an overview of time series application in the software engineering domain is given.

3.1 Quality of User Stories

A modest number of alternative approaches exist for measuring the quality of User Stories. The number of articles addressing the quality of User Stories is also rather modest. On the other hand, papers covering this field are mostly quite recent which shows the growing interest of the topic and its relevance in the current software engineering domain.

As stated in a recent study [7], INVEST quality guidelines or self-defined guidelines are the most popular approaches used for assessing User Stories. INVEST acronym [9], defined by Bill Wake, states that good User Stories are independent, negotiable, valuable, estimable, small, and testable. These are helpful characteristics but difficult to measure.

L. Buglione and A. Abran [15] state that Agile Software Development has some management related weaknesses. For example, User Stories without enough level of detail or incomplete User Stories can cause errors in effort estimation. Authors have included a discussion section where they cover application of INVEST quality guidelines in agile software projects. The goal of the paper is to highlight the importance of applying INVEST guidelines in order to reduce errors in effort estimation caused by the quality issues of User Stories.

In another recent paper [16], domain ontologies have been applied in order to solve issues originating from the fact that User Stories are written using natural language. Using domain ontologies is expected to improve the quality of User Stories as it removes ambiguity. Proposed approach is demonstrated in a case study where development of educational support system takes place.

Lucassen et al. [8] propose the QUS framework and AQUSA software tool implementing part of the QUS framework. As this approach was used in the thesis more thorough overview of QUS and AQUSA is given in the Background chapter.

In recent research [17], Lai proposes the User Story Quality Measurement (USQM) model. The model groups User Story quality metrics under three layers. Layer “Discussable and Estimable Quality” consists of “Clarity Contents”, “Low Complexity”, and “Modularity”. Layer “Controllable and Manageable Quality” consists of “CM System”, “Version control tools”, and “Cross-reference Table”. The third and last layer “Confirmable quality” consists of “Assured Contents”, “Testability”, and “Verifiability”. As for this thesis, USQM model is not considered mainly because the research does not provide a tool that would implement the USQM model.

3.2 Empirical studies of requirements connections to bugs, rework and delays

In a paper written by D. Firesmith [11], 12 requirement engineering problems and their causes are described. The author has written the paper based on practical experience gathered while working with numerous real-life projects. The article states that an increase in the number of defects and delays can be caused by numerous subproblems, which in turn can be originating from problems like poor requirement quality, requirements volatility, or inadequate requirements process. Another article [18] studies the causes of bugs based on

real life data in order to propose a model of how bugs were introduced. Proposed model groups bugs into two categories. The first category is intrinsic bugs. Bugs in the category of intrinsic bugs were introduced by the changes in the source code. The second category, extrinsic bugs, groups the bugs that were introduced by requirement changes or other issues not recorded in the source code management system. The authors have also stated that the important limitation of proposed model is that it is not useful when bugs were introduced by faulty requirements from the past. Another article [19] about software development waste describes participant-observation study. During the study 8 projects were observed which included conducting numerous interviews. As a result, empirical waste taxonomy was established which listed 9 types of software waste. Possible causes of rework waste by this paper [19] were requirement problems, more precisely User Stories without concrete complete criteria or rejected User Stories. Paper [20] searching connection between requirements quality and project success examined 32 real life projects. One of the findings was that the requirements specification was poor in the projects where significant delays happened. Projects without significant delays had better requirements specification.

The authors of the QUS framework and AQUSA tool [21] report a case study that was conducted testing the QUS framework and AQUSA tool in three companies. The authors defined a package combining the QUS framework and AQUSA tool as Grimm method. The purpose of the case study was to find if Grimm method has an effect on practitioners' work. Thirty practitioners used the QUS framework and AQUSA for two months. Different metrics were evaluated comparing the two months before applying Grimm method and during applying Grimm method. The authors assessed following measures: User Story quality assessed using AQUSA, Perceived Impact on Work Practice (information was collected by survey and interviews), and 5 process metrics. These 5 process metrics were following: amount of Formal communication, Rework, Pre-release defect, Post-release defects, and Team productivity. Although the quality of User Stories in the three companies under review improved, authors were not able to find compelling evidence about the effect on previously stated 5 process metrics. In conclusion, research was able to find some moderate positive consequences, but the authors stated that in order to find statistically proven evidence research with a larger sample group will be necessary. Authors also encouraged others to execute similar studies in order to identify the effect of requirement engineering practices on work practices.

3.3 Time series applications on software engineering

Time series methods have been used for different purposes in software engineering domain. For example, paper [22] describes predicting number of bugs by applying time series analysis and ARIMA model. Another paper [23] describes the usage of time series analysis for identifying normal network traffic behaviour. In another paper [24] time series analysis is used for forecasting the number of changes in Eclipse project. A recent paper [25] explores the possibilities for predicting software faults and software quality by using time series methods. Comparison of Autoregressive Integrated Moving Average (ARIMA), Random walk, and Holt-Winters forecast models revealed that the best accuracy was achieved by ARIMA model. Models were tested with real life data about sprint backlog size, tasks in progress, and delayed tasks. Article [25] also states that automated quality related analysis is necessary for various managerial roles like team leaders and product owners as it would help them in making informed decisions about time scale of the project, bugs predictions, and many other aspects.

To conclude the related work, it can be said that poor requirements quality can be related to various issues like increased number of product defects, rework, and delays. As stated in

recent article [25] automated quality related analysis is important for team leaders, product owners, and other similar roles for monitoring the software development process and for making informed decisions. Regarding User Stories various approaches have been used to improve and measure the quality of User Stories but it has been not possible to forecast the quality of User Stories as it has been difficult to measure. Approach described in the master thesis will address this problem by using QUS framework and AQUSA tool on data from several real life open source agile software projects in order to assess the quality of User Stories. AQUSA output will be quantified in order to forecast the quality of User Stories for monitoring purposes. In addition, connection between the quantified User Stories quality scores and the number of bugs, delays, and rework will be studied in order to discover a possible connection between the quality of User Stories and stated three aspects.

4 Study design

The study design of this thesis is based on the research steps shown in Figure 2. In short, the study started with data acquisition from several publicly available Jira servers. The data was collected from real-life open source agile software projects. In order to understand the data, a significant amount of data exploration was needed. Then, several data pre-processing and data cleaning steps were applied in order to make the data ready for analysis. Once the dataset was cleaned and pre-processed, the User Stories were selected from the dataset and processed with AQUSA tool. As a result, a list of defects related to the User Stories were obtained. This list of defects was exported for further analysis.

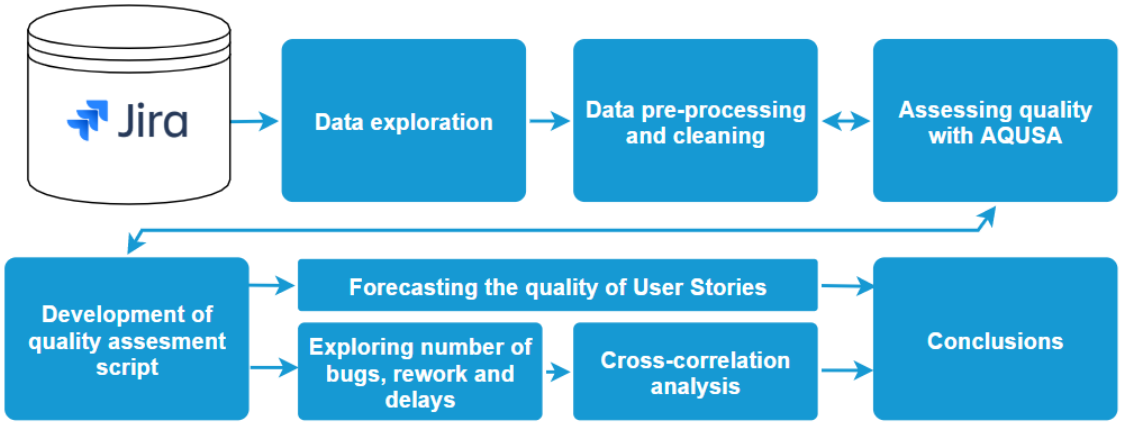


Figure 2. Research steps.

In order to analyse the list of defects generated by AQUSA, a formula to quantify the list of defects was proposed. This formula made it possible to calculate and present the quality of User Stories as numerical values and, thus, enabled the possibility of performing time series analysis. Forecasting the quality of User Stories and analysis of number of bugs, rework, and delays was performed in parallel as these activities were not connected to each other. Analysis of the number of bugs, rework, and delays first required data exploration in order to understand the available data. After this, cross-correlation analysis was performed for User Story quality scores and the stated three software aspects. The goal of this activity was to find possible relationships indicated by repeating patterns between the quality of User Stories and the three software aspects. For example, we searched for repeating patterns between the number of bugs and quality scores of User Stories. As a final step, results were concluded. For data cleaning and analysis, Python programming language was used as well as several libraries needed for data analysis like Pandas, Matplotlib, NumPy, Datetime, Statsmodels, and Pmdarima.

4.1 Research questions

Poor requirements quality can cause significant issues like exceeding deadlines, increase in the amount of rework, and product defects [11]. At the same time ensuring high quality requirements can be challenging as it requires many different steps and activities and due to the informal nature of most requirement documents it is difficult to apply automated tools for quality assessment [12]. RQ1 concentrates on solving this problem by finding how the quality of User Stories can change over time and discovering if it is possible to forecast the quality of User Stories. These findings are important as they would enable software development teams to automatically monitor the quality of User Stories and therefore predict quality for the next period. As a result, software development teams would be able to use

forecasted quality values and confidence intervals for setting the standards for quality. Thus, RQ1 is formulated as follows:

RQ1: Is it possible to forecast the change of the quality of User Stories for monitoring purposes?

To answer RQ1, this study relies on the QUS framework along with AQUSA tool implementing the QUS framework [8]. For forecasting the quality of User Stories SARIMA model [14] was used. During the analysis the quality of User Stories will be assessed by using AQUSA tool, formula will be proposed in order to represent User Story quality scores as numerical values, the change of User Story quality over time will be plotted and SARIMA model will be applied in order to forecast the quality of User Stories for monitoring purposes. In order to validate and understand the accuracy of the forecast, common metrics for evaluating forecasting will be used, for example Mean Absolute Percentage Error (MAPE) [14].

In order to stress the importance of controlling the quality of User Stories, we studied the relationship between the quality and three software development aspects: number of bugs, rework done (re-opened issues), and delays (based on due date). There are some expectations for comparing syntactic and pragmatic User Story quality with those aspects.

The first expectation is related to the relationship between User Story quality and number of bugs. This expectation is based on the assumption that poorly written User Stories can affect the understanding of the requirements and therefore increase the number of bugs found in testing or by software users. This assumption is supported by several papers [11, 18] stating that faulty requirements can cause and increase the amount of product defects.

The second expectation is that poor User Story quality can increase the amount of rework. This assumption is also favoured by several papers. Rework is often defined as a type of software waste. Paper describing different types of software waste [19] stated that rejected stories and stories without concrete complete criteria are one of the possible causes of this kind of waste. Another paper [26] states that a considerable amount of rework can possibly indicate to requirement errors.

The third expectation is that poor User Story quality can cause delays (exceeding due dates). Like previous assumptions, this expectation is also supported by several papers [11, 20] which state that requirement problems can cause numerous issues like product defects, rework, and consequently significant exceeding of due dates. Thus, RQ2 is formulated as follows:

RQ2: What is the relationship between the quality of User Stories and other aspects of software development? The aspects studied are number of bugs, rework done (re-opened issues), and delays (based on due date).

In order to answer RQ2, time series data of User Story quality, number of bugs, rework and delays was prepared. For missing values imputing methods were used. In order to find possible correlation, Windowed Time Lagged Cross Correlation (WTLCC) [27] method was applied and results were visualized in four different time window settings (4 time window, quarters, months and sprints). Visualized results were inspected in order to find possible patterns that could support the previously described expectations for number of bugs, rework, and delays.

4.2 Data collection procedures

In this section, an overview of data acquisition activities and a list of the studied projects are given. In addition, the data pre-processing and cleaning steps are described.

4.2.1 Data acquisition

All the data have been collected from open-source projects that use Jira[1]. Jira is a software development work management tool that helps agile teams to perform issue tracking, progress reports, work estimation, and work logging, among other activities.

The dataset used in this thesis consists of issue reports and changelogs of 10 different open-source projects. It was constructed by augmenting an existing dataset that has been used in previous research studies [28, 29]. The initial dataset consisted of issue reports of 8 projects. Two additional projects were collected during this thesis. The collected data was extracted using Jiraextractor[2]. Jiraextractor is a tool for connecting with Jira server and downloading issue reports with their changelogs. The code sample below shows the usage of Jiraextractor to download all the issue reports and their changelogs for the project "Spring Slices".

```
python jiraextractor.py -s https://jira.spring.io --project SLICE
```

The resulting dataset contains various projects with different domain, project scenarios, number of issues, developer experience, and development period. The following list shows all the projects considered in the analysis. Every project in the list has a short description, the development period, and its project acronym. These acronyms are specified behind every project name in brackets and will be used in further analysis in order to refer to specific project.

1. **Spring XD**[3] **(XD)** is a system for data ingestion, real time analytics, batch processing, and data export. System is targeted for big data applications. In the analysis we are considering the development period from June 2013 to March 2016.
2. **DNN Platform**[4] **(DNN)** is a content management system designed for .NET developers. In the analysis we are considering the development period from August 2013 to April 2016.
3. **MongoDB Compass**[5] **(COMPASS)** is a graphical user interface (GUI) for MongoDB database. In the analysis we are considering the development period from March 2016 to December 2019.
4. **Aptana Studio**[6] **(APSTUD)** is a web development Integrated Development Environment (IDE). In the analysis we are considering the development period from January 2012 to March 2016.
5. **Apache Mesos**[7] **(MESOS)** is a cluster management system. In the analysis we are considering the development period from May 2014 to May 2016.

[1]https://www.atlassian.com/software/jira/guides/use-cases/what-is-jira-used-for
[2]https://github.com/ezequielscott/jiraextractor
[3]https://projects.spring.io/spring-xd/
[4]https://www.dnnsoftware.com/
[5]https://www.mongodb.com/products/compass
[6]http://www.aptana.com/
[7]http://mesos.apache.org/

6. **Mule**[8] **(MULE)** is an enterprise service bus (ESB) and integration platform. In the analysis we are considering the development period from March 2013 to May 2016.
7. **Nexus by Sonatype**[9] **(NEXUS)** is a repository manager for software artefacts. In the analysis we are considering the development period from November 2010 to May 2016.
8. **Titanium SDK**[10] **(TIMOB)** is a JavaScript based software development kit (SDK) and command line tool (CLI). In the analysis we are considering the development period from February 2012 to April 2016.
9. **Appcelerator Studio**[11] **(TISTUD)** is an Integrated Development Environment (IDE) for building cross-platform mobile applications. In the analysis we are considering the development period from January 2012 to March 2016.
10. **Spring Slice**[12] **(SLICE)** is a part of Spring software development framework. In the analysis we are considering the development period from February 2010 to January 2015.

Original dataset consisted of tables in a csv format containing issue reports of all kinds. For example, bugs, epics, stories, tasks, sub-tasks, etc. The number of issue reports per project is visualized in Figure 3.

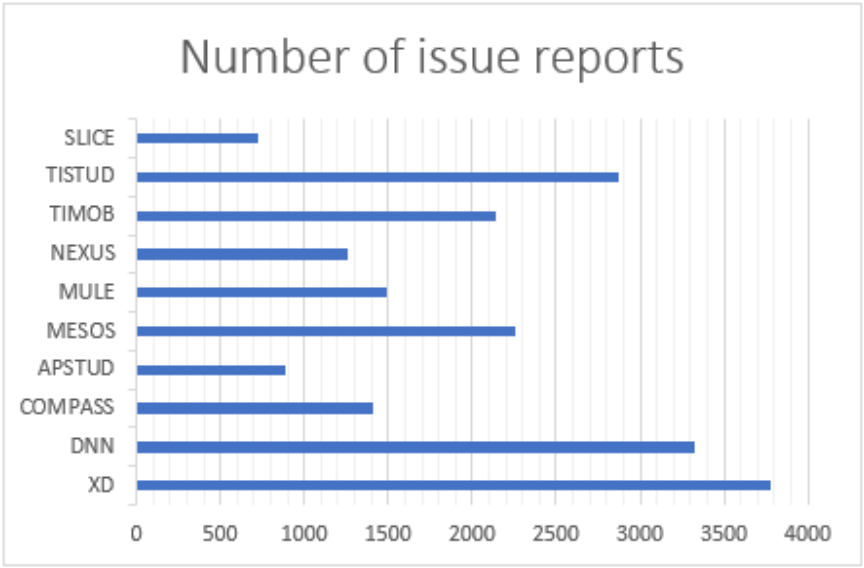


Figure 3. Size of original dataset.

As we can see from Figure 3 the datasets of projects under review were quite different in size. XD and DNN had the highest amount of issues, 3782 and 3328, respectively. APSTUD

[8]https://www.mulesoft.com/resources/esb/what-mule-esb
[9]https://www.sonatype.com/product-nexus-repository
[10]https://www.appcelerator.com/titanium/titanium-sdk/4/
[11]https://docs.axway.com/bundle/Appcelerator_Studio_allOS_en/page/axway_appcelerator_studio_getting_started.html
[12]https://jira.spring.io/projects/SLICE/issues/

and SLICE had the fewest issues, 721 and 885, respectively. The total number of issues under review was 20151. All initial data is available in GitHub repository[13].

Understanding the data about all projects required a considerable amount of data exploration in order to reveal all the problems with data and make it possible to perform data pre-processing and data cleaning described in the following paragraphs.

4.2.2 Data pre-processing

This section gives an overview of performed pre-processing activities that were executed in order to provide input for data cleaning activities.

The most extensive part of pre-processing was related to User Stories. This part included keeping only issues with issue type "Story." Projects had issue types stored in different columns. This problem was solved by renaming all columns that were holding issue type as "fields.issuetype.name". Next, only issues with statuses Closed," "Done," "Resolved," and "Complete" were kept. This was done in order to keep only finished stories because if the story is not finished yet it is not possible to find correlation between the User Story and 3 aspects defined under RQ2. Next, columns containing User Stories were chosen for further cleaning. This activity turned out to be challenging as most of the projects were not consistent about the field where they stored User Stories. This led to a situation where some User Stories were stored in "fields.summary" column and others in "fields.description" column. In order to fix this issue, four possible approaches were tested.

The first approach was to concatenate "fields.summary" and "fields.description" field and assess the quality of concatenated column with AQUSA. Assessing the quality of stories concatenated this way resulted with low quality scores for even correct User Stories because AQUSA was not able to identify User Stories correctly.

The second approach was to keep only "fields.summary" or "fields.description" columns depending on which held a greater number of User Stories. The negative side of this method was that it caused a loss of considerable amount of data.

Finally, an approach was tested where "fields.summary" and "fields.description" columns were both kept, and identifier was set in order to track the source afterwards. Field "identif" was added for both columns. Identifier 0 was set for "fields.description" fields and identifier 1 for "fields.summary." After identifiers were set "fields.summary" and "fields.description" columns were merged into one column, followingly referred as "Story" column. After assessing "Story" column only the story with better quality score was kept for further analysis.

Selecting the best approach included calculating quality scores for all mentioned three approaches and inspecting the results manually. As a result, the third approach was selected. This meant that the quality for "fields.summary" as well as "fields.description" was assessed with AQUSA and then kept the one with better quality score. Selected method made it possible to identify a greater number of User Stories and ensured more reliable results.

4.2.3 Calculation of bugs, rework, and delays

Next, a brief description of required calculation steps for 3 aspects defined under RQ2 is given. These aspects were number of bugs, rework, and delays. Pre-processing for the number of bugs consisted of selecting relevant project by project identifier (whether by "project"

[13]https://github.com/TanelToemets/Analysing-The-Quality-Of-User-Stories-In-Agile-Software-Projects/tree/master/data/datasets

or “project.key” column), keeping only issues with issue type “Bug” by selecting only columns with issue type “Bug” from “issuetype.name” or “fields.issuetype.name” column depending on the project and taking only the bugs that were completed as the intent was to analyse past data. This was done by taking only the bugs with statuses “Closed,” “Done,” “Resolved,” “Complete.” Listed statuses were held in “fields.status.statusCategory.name” or “status.statusCategory.name” column depending on the project. As the next step additional column named “bugnr” was added and evaluated with value 1 for all rows in the bugs dataset. The purpose of doing so was making it possible to sum the number of bugs over different time periods during further steps of the analysis. Result of these activities for each project was exported to csv format and are available at “WTLCC_input_data” folder in GitHub repository[14]. Additionally, creation time was formatted and indexed at “fields.created” or “created” column depending on the project. This was done in order to plot the change of the number of bugs over time in further steps of the analysis.

Pre-processing for rework started with reading changelog files from the initial dataset containing records of rework (whether from “jiradataset_changelog.csv” or “compass_changelog_extracted.csv” depending on the project). Status changes of the issues were captured in the rows where “field” column had a value “status.” Therefore, only the changelog rows where “field” column had the value “status” were kept for further analysis. Next, a function for catching all possible status changes that were indicating rework was built. Status changes were captured in “fromString” and “toString” columns. For most of the projects it was possible to identify rework if status in “toString” column had a value “Reopened.” In order to catch all possible rework cases status changes from “Closed,” “Done,” “Resolved” to “Open,” “In Progress,” “Ready to Work,” “Opened” or “Reopened” were additionally considered. Next, the creation time of status change in “created” column was formatted and indexed in order to be able to use this data in further analysis. Also, additional column named “rework_nr” was added and evaluated with value 1 for all rows in the rework dataset. The purpose of doing so was making it possible to sum the rework cases over different time periods during further steps of the analysis. Results of rework pre-processing activities were also exported to csv format and are available at “WTLCC_input_data” folder in GitHub repository.[15]

Pre-processing for delays started with reading sprint files (“jiradataset_sprints.csv”) from the initial dataset. In order to identify delays, resolution dates of the issues were needed. Resolution dates were stored in initial issues dataset (whether in jiradataset_issues.csv or compass_issues_extracted.csv depending on the project). Initial issues dataset was merged with sprints dataset in order to add resolution date to sprits data. Merging was done by issue keys. Next, storing of null values was made consistent for all projects and resolution dates (“fields.resolutiondate” column), as well as end dates of sprints (“sprint.endDate” column), were indexed and formatted. Formatting the dates made it possible to compare the values of resolution date and sprint end date in order to keep only values where the resolution date of issue was later than the sprint end date. Delays were identified by selecting only the fields where end date value of the sprint in “sprint.endDate” column (in “jiradataset_sprints.csv” file) was smaller than resolution date column “fields.resolutiondate” of the User Story (from whether “jiradataset_issues.csv” or “compass_issues_extracted.csv” depending on the project.) Finally, an additional column named “delay_nr” was added and evaluated with value

[14]https://github.com/TanelToemets/Analysing-The-Quality-Of-User-Stories-In-Agile-Software-Projects/tree/master/data/WTLCC_input_data
[15]https://github.com/TanelToemets/Analysing-The-Quality-Of-User-Stories-In-Agile-Software-Projects/tree/master/data/WTLCC_input_data

1 for all rows in the delays dataset for summarizing the number of delays over time in further steps of the analysis. Result of delays pre-processing was exported to csv format and is available at "WTLCC_input_data" folder in GitHub repository.[16]

4.2.4 Data cleaning

In order to proceed with the analysis, the User Story data gathered from previously described projects had to be cleaned. Before cleaning some pre-processing activities were done, for example keeping only issues with issue type "Story". These activities are described in the previous section. The purpose of the data cleaning was to prepare input for AQUSA. Data cleaning activity was improved several times during the analysis. This was done if new shortcomings of data were identified after assessing User Stories with AQUSA or after analysing the AQUSA output. Data cleaning scripts are available in the thesis repository "data_cleaning" folder.[17]

General description of data cleaning steps for User Stories is presented in Table 2.

Table 2. Data cleaning steps for User Stories.

Step	Description of data cleaning steps for User Stories
1	Removing empty rows from "Story" column.
2	Removing several project specific headers from "Story" column (if they exist). For example, removing "h2. Back story" and everything following this heading.
3	Removing links to web.
4	Removing ".jar" file extensions to clean some manually identified special cases.
5	Removing code examples.
6	Removing different types of curly brackets combinations.
7	Removing paths.
8	Removing words longer than 19 characters. This decision is based on the paper [30] analysing word length, sentence length and frequency. The results presented in the paper provided the knowledge that words with more than 19 characters are very rare in English language (less than 0.1%). This knowledge provided a reason to remove unwanted parts of User Stories as this kind on long strings are usually part of program code added to the User Stories. For example, string like "TriggerSourceOptionsMetadata."
9	Removing consecutive exclamation marks and everything between them. Such structures were used for adding images. For example: "!GettingStarted.png!"

[16]https://github.com/TanelToemets/Analysing-The-Quality-Of-User-Stories-In-Agile-Software-Projects/tree/master/data/WTLCC_input_data
[17]https://github.com/TanelToemets/Analysing-The-Quality-Of-User-Stories-In-Agile-Software-Projects/tree/master/data_cleaning

10	Removing square brackets and everything in them.
11	Removing non-ASCII characters to make AQUSA able to parse the cleaned input data.
12	Removing other special characters to make AQUSA able to parse the input data. For example, characters like "<" ," >", "_", "$" etc.
13	Removing all different kinds of whitespaces (tabs, " " etc) and replacing them with single whitespace.
14	Removing duplicate User Stories. This was done in order to remove a special kind of outliers. Some User Stories were written in an identical way with only one word distinguishing them. Cleaning process required to make AQUSA able to parse the input data. In order to fix parsing issues code examples in brackets were removed. This created outliers with negative quality scores because data cleaning had made a small number of User Stories identical. An example of the described issue is presented in Table 3. In order to fix this problem such User Stories were excluded from our analysis.
15	Removing upper outliers (abnormally long User Stories). Upper outliers were removed using standard deviation formula in Equation 4. Equation 4. Formula for removing outliers $condition = mean - N \times \text{standart\_deviation}$ Column named "description_length" was created to store the number of characters in the column "Story." Next, mean story length and standard deviation were calculated. "*N*" in Equation 4 stands for coefficient selected separately. This was done in order to achieve a more precise outlier detection formula for each project. All User Stories that did not satisfy the condition presented in Equation 4 got excluded.
16	Removing lower outliers (User Stories with less than 3 words). These were excluded because they created outliers in quality scores and therefore influenced the analysis. For example, some User Stories consisted of only "See:" and some webpage URL. Since web page links were removed during the cleaning process only one word "See" was present in the User Story which lead to several outliers with very low (negative) quality scores.

Table 3 presents an example of the issue described in step 14 of the cleaning steps in Table 2. From the example, it can be observed that the content of two different User Stories was made identical after performing previously described cleaning steps.

After applying the pre-processing and cleaning steps, the resulting dataset was saved in a file named "jira-PROJECT_ACRONYM-allus-DS.csv" for further use. Files containing cleaned User Stories are available in the thesis repository.[18]

[18]https://github.com/TanelToemets/Analysing-The-Quality-Of-User-Stories-In-Agile-Software-Projects/tree/master/data/cleaned_input_data

Table 3. Example of outlier User Stories.

Key	User story	Status
XD-3531	As a Spring XD developer, I'd like to move {{tcp}} module from XD to s-c-s repo, so I can use it as source to build streaming pipeline.	Before cleaning
XD-3530	As a Spring XD developer, I'd like to move {{mail}} module from XD to s-c-s repo, so I can use it as source to build streaming pipeline.	Before cleaning
XD-3531	As a Spring XD developer, I'd like to move module from XD to s c s repo, so I can use it as source to build streaming pipeline.	After cleaning
XD-3530	As a Spring XD developer, I'd like to move module from XD to s c s repo, so I can use it as source to build streaming pipeline.	After cleaning

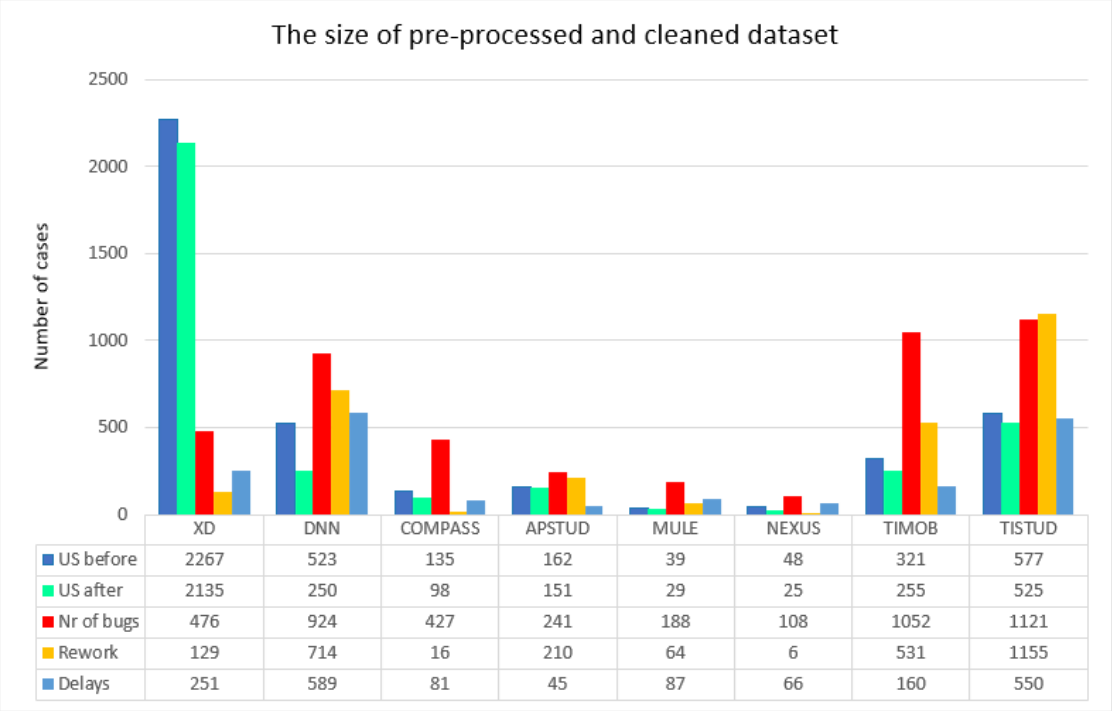
Data cleaning also revealed that projects MESOS and SLICE had only 17 and 14 User Stories, respectively. This was not enough for making generalizations in further steps of the analysis. Therefore, a rule was set that any project with less than 20 User Stories will be excluded from further analysis. As a result, MESOS and SLICE were excluded, meaning that the remaining dataset contained 8 projects.

Additional data cleaning step was added after inspection of initial visualizations of AQUSA output revealed that several projects had inactive development periods at the start or end of the project. Initial plots revealed that 6 of 8 projects (DNN, COMPASS, APSTUD, MULE, NEXUS, and TISTUD) had very few User Stories at the beginning and end of the viewed time period. This was caused by the fact that development at the start and end of the projects was inactive and therefore very few new User Stories were produced. To not let these low activity periods affect the results, only active development periods were included in further analysis. Table 4 lists all the projects used for analysis with their development periods.

Table 4. Active development periods.

Project	Start of active development	End of active development
XD	2013/06/11	2016/02/23
DNN	2013/07/01	2016/01/01
COMPASS	2017/09/20	2018/09/01
APSTUD	2011/06/08	2012/06/20
MULE	2013/01/01	2014/09/01
NEXUS	2014/09/01	2015/04/01
TIMOB	2012/02/15	2016/04/20
TISTUD	2011/01/01	2014/08/01

Figure 4 shows the size of the pre-processed and cleaned dataset. In the bar chart, "US before" indicates the number of User Stories before data cleaning and after performing pre-processing activities. "US after" shows the number of User Stories after performing all data cleaning activities. In addition, the number of bugs, number of rework cases, and number of delays are plotted in Figure 4 for all projects under review.



	XD	DNN	COMPASS	APSTUD	MULE	NEXUS	TIMOB	TISTUD
US before	2267	523	135	162	39	48	321	577
US after	2135	250	98	151	29	25	255	525
Nr of bugs	476	924	427	241	188	108	1052	1121
Rework	129	714	16	210	64	6	531	1155
Delays	251	589	81	45	87	66	160	550

Figure 4. The size of pre-processed and cleaned dataset.

4.3 Analysis procedures

This section gives an overview of all analysis procedures. The analysis relies on the QUS framework [8] and AQUSA tool [8, 10]. This section describes the steps: assessing the quality of User Stories using AQUSA, quantifying the quality using generated defect report (AQUSA output), applying ARIMA model for User Story forecasting (RQ1), exploring 3 software development aspects, and performing cross-correlation analysis (RQ2).

4.3.1 Setting up AQUSA and generating defect reports

AQUSA is available for download in a public GitHub repository[19]. As stated in the repository readme file current tool is a prototype and was rather challenging to understand and use due to missing documentation. Regarding this analysis, AQUSA was installed on Ubuntu 18.04 and the usage of the tool also involved installing Python 3, Flask, all required libraries (listed in AQUSA requirements.txt file) and creating a database (PostgreSQL was chosen). It is possible to use AQUSA by implementing a specific connector or by using the user interface of the AQUSA-GUI component [8]. In case of this analysis GUI component was used. There were some minor bugs in the publicly available source code that, before fixing them, prevented proper working of the tool. These were mainly naming issues, for example "text" and "title" were both used as variables for storing User Stories. In addition, it was not possible to install some libraries specified in AQUSA requirements.txt file. In order to overcome this issue, problematic libraries were replaced with newer versions. Understanding the

[19] https://github.com/gglucass/AQUSA

architecture of AQUSA tool was necessary in order to fix all issues encountered during setup.

Cleaned User Stories of all 8 projects under review were imported to AQUSA to generate defect reports. AQUSA stores the defect reports in its internal database, so they were exported by using a simple SQL query[20]. Exported defect reports (AQUSA output) for all 8 projects are available in the thesis repository[21].

Table 5 shows one row of the defect report (AQUSA output) showing one example defect of project XD. Row "Title" contains original User Story, "Key" is unique identifier of the story, "Role," "Means," and "Ends" show separate parts of User Story, "Highlight" shows the concrete location of the issue, "Kind" marks the kind of issues, "Sub kind" specifies the issue and "Severity" states the severity of the issue. Possible severity values are high, medium, and minor. The last row "False positive" is for marking false positives by the user (from the GUI component). Value "FALSE" in "False positive" row states that the story has not been marked as a false positive by the AQUSA user.

Table 5. Example row from project XD defect report (AQUSA output).

Title	As a user, I'd like Flo Graphs as screenshots while referring to the batch DSL, so it will be easy for me to relate to concepts.
Key	XD-3669
Role	As a user
Means	, I'd like Flo Graphs as screenshots while referring to the batch DSL,
Ends	so it will be easy for me to relate to concepts.
Highlight	Use the most common template: As a, I want to, So
Kind	uniform
Sub kind	uniform
Severity	medium
False positives	FALSE

4.3.2 Quantifying the quality

In order to quantify defect reports and provide each User Story with a quality score, two formulas were proposed. The first formula defines the total quality of User Story whereas the second formula defines the total penalty calculated by the results from defect report. The definition of total quality of User Story is presented in Equation 5.

[20]https://github.com/TanelToemets/Analysing-The-Quality-Of-User-Stories-In-Agile-Software-Projects/blob/master/user_story_analysis/extracting_defects.sql
[21]https://github.com/TanelToemets/Analysing-The-Quality-Of-User-Stories-In-Agile-Software-Projects/tree/master/data/analyzed_output_data

Equation 5. Total quality of User Story.

$$Q = 1 - P$$

Q is defined as the total quality of the User Story. Quality of the perfect story is 1. *P* is the total penalty that is obtained by summarising all defects.

The penalty for each defect detected by AQUSA depends on the severity of the issue. A higher penalty is granted to the issues with higher severity. Multiplier 1/6 in Equation 6 was selected because AQUSA can give a maximum of 6 different defects in the category high. Multiplier 1/9 was selected because AQUSA can give a maximum on 9 different defects in category medium and 1/12 because AQUSA can give 12 different defects in category minor. The formula for calculating the total penalty is presented in Equation 6, where variables "*high*," "*medium*," and "*minor*" are defined as the number of defects with high, medium, or minor severity, respectively.

Equation 6. Total penalty.

$$P = \frac{1}{6} \times high + \frac{1}{9} \times medium + \frac{1}{12} \times minor$$

The proposed formulas enable us to calculate numerical quality scores for each User Story which will be followingly used for time series analysis. Table 6 shows an example result for three User Stories from project DNN. For every User Story (identified by the "Key") total quality score is calculated.

Table 6. Example result of summarizing the AQUSA quality scores.

Key	User story	Quality score
DNN-6927	As an administrator, I want to have a single module that allows me to manage and edit Dynamic Content Types and the associated data types, validators and templates.	0.83
DNN-3624	As a user I want to be able to see images I have uploaded.	1.00
DNN-4881	We need to improve the performance of the Login Scenario. We will follow the overall guidelines for this scenario as described in the Epic.	0.67

Scripts used for quantifying the quality are available in the project repository "user_story_analysis" folder[22].

4.3.3 Forecasting the quality of User Stories

In order to forecast the quality values, the quality was modelled as time series. The first step for forecasting the quality of User Stories was calculating quality scores for all projects as defined in Equation 5 and Equation 6. Calculated quality scores were merged with initial issues dataset in order to catch the corresponding creation time of each User Story. Next, the data was resampled over the frequency of semi-month (15th and end of month). This was done to mimic the regular 2-week cycle of sprints. Then, mean quality of User Stories was

[22]https://github.com/TanelToemets/Analysing-The-Quality-Of-User-Stories-In-Agile-Software-Projects/tree/master/user_story_analysis

calculated for each two-week period. Missing values were filled by propagating the last valid value forward until the next existing value. The result of these activities was time series data containing creation times of User Stories as well as numerical quality scores.

Followingly, the first check of time series stationarity was performed. In order to check if time series of the specific project was stationary or not quality distribution was visualized as histograms. These distribution histograms were visually checked for each project.

Next step in the analysis was performing time series decomposition [14] to receive better insights of the data. More precisely, it was done in order to decompose time series into components of trend, seasonality, and residual (or irregularity). For decomposition Statsmodels library for Python was used. To be more exact, Statsmodels seasonal_decompose[23] function with the model set to "additive". Time series was considered as additive [14] since the components of time series (quality scores) were expected to affect each other. Decomposition plots are presented in Appendix IV.

The existence of seasonality discovered during the decomposition supported the use of SARIMA model for forecasting the quality of User Stories. The advantage of SARIMA model was that it can be used with seasonal data. SARIMA as well as ARIMA can also make non-stationary time series stationary by performing finite differencing of the time series [14].

At this stage, more convenient stationarity check was performed. Stationarity of time series is usually tested by statistical tests like ADF (Augmented Dickey-Fuller test) [31, 14] or KPSS (Kwiatkowski Phillips Schmidt Shin test) [31]. In case of this thesis, Pmdarima library for Python was used. More precisely, auto_arima[24] function which provides the user with "test" parameter which can be used for setting unit root tests like previously mentioned ADF or KPSS. For this analysis auto_arima "test" parameter was set to use ADF test. Test results are described in the results section of the thesis.

The same auto_arima function was used for performing a grid search in order to find the best parameters for all models. Previously performed decomposition had revealed that the data of all projects had a seasonal component. As a result of this, *m* parameter of auto_arima function was set for all projects. As stated in the documentation[25] of auto_arima function, *m* parameter indicates the number of observations per seasonal cycle. In order to set this parameter decomposition plots, available in Appendix IV, were inspected to identify approximate cycle length from seasonal components. For longer projects (XD, DNN, TIMOB, TISTUD) seasonal cycle was set to 12 two-week periods (*m*=12), which is approximately half a year. For shorter projects, seasonal cycle was set to 6 two week periods (*m*=6) meaning that one cycle was expected to last approximately 3 months. For the shortest project NEXUS, seasonal cycle was set to 3 two-week periods meaning that one cycle was expected to last approximately 6 weeks. From the output of performing grid search the model with the lowest AIC (Akaike information criterion) [32] value was selected as lowest AIC value indicates to best model. This approach was used for all 8 projects in order to identify best model for each project.

After identifying best models, models were fitted by using SARIMAX[26] function from Statsmodels library. It would have been possible to use auto_arima, but instead Statsmodels

[23]https://www.statsmodels.org/stable/generated/statsmodels.tsa.seasonal.seasonal_decompose.html
[24]https://alkaline-ml.com/pmdarima/modules/generated/pmdarima.arima.auto_arima.html
[25]http://alkaline-ml.com/pmdarima/0.9.0/tips_and_tricks.html#period
[26]https://www.statsmodels.org/dev/generated/statsmodels.tsa.statespace.sarimax.SARIMAX.html

solution was preferred because of more convenient usability for plotting. Selected SARIMAX function also took care of making the data of 7 non-stationary projects stationary.

4.3.4 Analysis of number of bugs, rework and delays

In order to get first insights of number of bugs, number of rework cases, and number of delays all these aspects were plotted. Before plotting the data was resampled over the semi-month period (15th and end of month) in order to mimic the regular 2-week cycle of sprints. Plots for number of bugs are available in Appendix I, plots for number of rework cases in appendix II and number of delayed cases in appendix III. Code for producing these plots is available in thesis GitHub repository[27] in "bugs_analysis", "rework_analysis" and "delays_analysis" folders.

A cross-correlation analysis was conducted in order to determine the relationships since no evident patterns were observed from the plots.

First step in doing so was preparing needed time series data. Time series data for cross-correlation analysis was resampled over business days, meaning that weekends were excluded. Missing values were managed by using "time" interpolation. This method was suggested in a practical article[28] written by M. El-Nesr where different imputing methods on data similar to the dataset of this thesis were compared.

For further analysis Windowed Time Lagged Cross Correlation (WTLCC) method described in [27] was used. Method also provided great instructions for visualising the results. WTLCC analysis scripts for bugs, delays, and rework available in thesis repository[29].

Expectations described under RQ2 were following:

- Poor User Story quality can increase the number of bugs [11, 18].
- Poor User Story quality can increase the amount of rework [19, 26].
- Poor User Story quality can cause increase of delays [11, 20].

In order to find evidence of the expectations WTLCC was performed in 4 settings for all 8 projects. Used settings are presented in Table 7.

Table 7. WTLCC time window settings.

Setting	Size of time window
Setting 1	4 windows (start, 2 middle parts, end)
Setting 2	Quarters
Setting 3	Months
Setting 4	Sprints

In setting 1 in Table 7 time series data was split into four windows. The first window was showing the starting period of active development, second and third window showed the middle period of active period and last window closing part of active development. In setting

[27]https://github.com/TanelToemets/Analysing-The-Quality-Of-User-Stories-In-Agile-Software-Projects
[28]https://medium.com/@drnesr/filling-gaps-of-a-time-series-using-python-d4bfddd8c460
[29]https://github.com/TanelToemets/Analysing-The-Quality-Of-User-Stories-In-Agile-Software-Projects/tree/master/cross_correlation_analysis

2 each window represented quarters, in setting 3 months and in setting 4 sprints. This resulted with 96 measurements as 4 settings, 8 projects and 3 aspects under review were analysed. Settings 3 and 4 were excluded from further analysis because the size of these time windows were too small and amount of data not enough in order to generalize the results.

Inspection of the results revealed that Setting 1 was best for shorter projects, more precisely for COMPASS, APSTUD, MULE, and APSTUD. For these projects Setting 1 meant that one time window corresponded to approximately 1 quarter of a year. In order to better compare shorter and longer projects XD, DNN, TIMOB, and TISTUD were analysed in setting 2. For longer projects setting 2 meant that one time window corresponded to a one quarter of a year. In order to answer the research question, a search for repeating patterns was performed while also considering previously listed expectations. An overview of the findings with explanations is given in the results chapter of the thesis.

4.4 Validity procedures

This section gives an overview of the methods and procedures applied in order to ensure the validity of results. At first, validity procedures regarding RQ1 are described followed by the validity procedures regarding RQ2.

Regarding RQ1, SARIMA model was used. The validity of this decision is supported by the use of seasonal_decompose[30] function from Statsmodels Python library, which revealed seasonal patterns for all projects. Choice of forecast model is also supported by papers [22, 25] where similar approach was successfully applied.

In order to identify any unusual behaviours of the selected forecast model, plot_diagnostics[31] function from Statsmodels library was used. This function produces 4 plots showing standardized residual, histogram together with estimated density, normal Q-Q plot with normal reference line as well as correlogram. Using the diagnostics tool did not reveal any significant shortcomings for any of the projects. Residual for all projects was normally distributed, all histograms suggested normal distribution, no autocorrelation in residual errors were identified and no unwanted behaviours of residual errors were present. Regarding forecasting 2/3 of data was used for training and 1/3 for testing for all projects under review.

Many measures aim to assess the performance of forecasting models [14]. In this work, three popular metrics were chosen to evaluate the model, namely, the Mean Absolute Percentage Error (MAPE) [14], the correlation coefficient [33], and the Min-Max error [32]. These metrics were chosen since they allow us to compare the forecast results among the projects separately [32]. Followingly, a short overview of each of these metrics is given.

The Mean Absolute Percentage Error (MAPE) [14] is a percentage error showing the percentage of average absolute error. The scale of measurement does not influence MAPE and it is therefore a good choice for comparing different models. Values of MAPE range from 0 to 1. It can also be presented as a percentage if the formula is multiplied by 100 [14]. Mathematical formula of MAPE is presented in Equation 7, where n indicates the number of observations, e_t indicates the difference between actual and forecasted value, and y_t indicates the actual value.

[30]https://www.statsmodels.org/stable/generated/statsmodels.tsa.seasonal.seasonal_decompose.html
[31]https://www.statsmodels.org/stable/generated/statsmodels.tsa.statespace.sarimax.SARIMAXResults.plot_diagnostics.html

Equation 7. MAPE [14].

$$MAPE = \frac{1}{n} \times \sum_{t=1}^{n} \left| \frac{e_t}{y_t} \right|$$

As a second validity measure the Pearson product-moment correlation coefficient was used. It is a widely used statistical measure for comparing two observations, more precisely for finding linear correlation between two observations. Values of correlation coefficient range between -1 and 1. Negative values show negative correlation, positive values positive correlation and 0 indicates that no linear correlation exists [33]. For calculating Pearson product-moment correlation coefficient corrcoef[32] function from NumPy's Python library was used. Mathematical formula for calculating the Pearson product-moment correlation is presented in Equation 8.

Equation 8. The Pearson product-moment correlation coefficient [33].

$$r = \frac{\sum(X_i - \bar{X})(Y_i - \bar{Y})}{[\sum(X_i - \bar{X})^2 \ \sum(Y_i - \bar{Y})^2 \ ]^{\frac{1}{2}}}$$

Min-Max error is a simple way to propagate errors. In this case it is calculated by taking minimums and maximums from forecasted and actual values. For this amin[33] and amax[34] functions from NumPy's Python library were used. Those functions return minimum and maximum values from the array, respectively. After this, Min-Max error was calculated by the formula presented in Equation 9. As the ranges of MAPE, Min-Max error values are in a range of 0 to 1 [32].

Equation 9. Min-Max error [32].

$$MinMax = \ 1 - \frac{1}{n} \times \sum_{t=1}^{n} \left(\frac{a_{mins}}{a_{maxs}} \right)_t$$

Followingly, a description of validity procedures regarding the relationship between User Story quality and 3 aspects stated under RQ2 are covered.

As a result of performing previously described analysis procedures time series data of quantified User Story quality, number of bugs, rework, and delays were produced. Quantitative data about number of bugs, rework, and delays was compared to User Story quality scores using a statistically valid method called Windowed Time Lagged Cross Correlation (WTLCC) [27].

This comparison will make it possible to discover if the specific issue like high number of bugs is in a correlation with the change of User Story quality over time. The results are valid if similar patterns for several projects are found which is also the reason of involving several projects to the analysis. Another indicator of valid results is the value of correlation coefficient. The higher value suggests a larger correlation between two observations. The search of possible repeating patterns relies on the expectations described under RQ2. The trustworthiness of the results is supported by the fact that the expectations and therefore all possible patterns that were searched are supported by various papers [11, 18, 19, 26, 20].

[32]https://docs.scipy.org/doc/numpy/reference/generated/numpy.corrcoef.html
[33]https://docs.scipy.org/doc/numpy/reference/generated/numpy.amin.html
[34]https://docs.scipy.org/doc/numpy/reference/generated/numpy.amax.html

To determine the effect size of the relationships found, the Spearman r coefficient values were interpret by following Cohen's guidelines [34, 35]: small, medium, and large effect sizes are respectively 0.1, 0.3, and 0.5.

Therefore, it can be concluded that the validity of RQ2 is ensured by following validity procedures:

- Involvement of 8 projects in order to generalize the results.
- Expectations of findings are supported by several papers [11, 18, 19, 26, 20].
- A possible correlation is searched by using statistically valid method (WTLCC).

5 Results

In this section, an overview of final study population together with descriptive statistics is given. Then, the results of forecasting activities are described (RQ1). Finally, the results of how User Story quality can be related to number of bugs, rework and delays are described (RQ2).

5.1 Study population

The final dataset contains 9308 issue reports and 2825 changelog entries of 8 different open source projects. The size of final dataset by initial variables is presented in Table 8.

Table 8. The size of final dataset.

Project	User Stories	Bugs	Rework	Delays
XD	2135	476	129	251
DNN	250	924	714	589
COMPASS	98	427	16	81
APSTUD	151	241	210	45
MULE	29	188	64	87
NEXUS	25	108	6	66
TIMOB	255	1052	531	160
TISTUD	525	1121	1155	550
Total:	3468	4537	2825	1829

From Table 8 we can see that the size of analysed projects was very different. In total 3468 User Stories, 4537 bugs, 2825 rework cases, and 1829 cases of delays were analysed. Better insights of initial variables are presented in Table 9.

Table 9. Descriptive statistics of initial variables.

Metric	User Stories	Bugs	Rework	Delays
Mean	434	567	353	229
Median	201	452	170	124
Std	706	407	413	221
Min	25	108	6	45
Max	2135	1121	1155	589

From Table 8 and Table 9 good overview of analysed projects can be observed. NEXUS had the smallest number of User Stories, Bugs, and Rework while APSTUD had the smallest number of delays. By far most User Stories were written in project XD (2135) while median of User Stories was just 201. When looking at the values of mean it can be said that on average projects had 434 User Stories, 567 bugs, 353 rework cases, and 229 cases of delays.

Regarding the calculated values overview of the descriptive statistics of User Story quality scores (as the most important calculated value in this analysis) are given in Table 10.

Table 10. Descriptive statistics of User Story quality scores.

Project	**Nr of quality scores (*Q*)**	**Mean**	**Median**	**Std**	**Min**	**Max**
XD	2135	0.812	0.833	0.070	0.389	1.000
DNN	250	0.812	0.833	0.118	0.389	1.000
COMPASS	98	0.928	1.000	0.083	0.667	1.000
APSTUD	151	0.800	0.833	0.074	0.389	0.889
MULE	29	0.781	0.833	0.076	0.667	0.889
NEXUS	25	0.733	0.833	0.131	0.389	0.833
TIMOB	255	0.768	0.833	0.083	0.500	0.889
TISTUD	525	0.806	0.833	0.066	0.500	0.889

Additionally, aggregated statistics of User Story quality scores across all projects are presented in Table 11.

Table 11. Aggregated statistics of User Story quality scores across all the projects.

Metric	**Mean**	**Median**	**Std**	**Min**	**Max**
Mean	0.805	0.854	0.088	0.486	0.924
Median	0.803	0.833	0.079	0.444	0.889
Std	0.057	0.059	0.024	0.122	0.066
Min	0.733	0.833	0.066	0.389	0.833
Max	0.928	1.000	0.131	0.667	1.000

From Table 10 and Table 11 good overview of User Story quality scores can be observed. These two tables give valuable information for comparing different projects regarding their User Story quality. Followingly, the most interesting findings are described.

Worst quality User Stories were in project NEXUS where the maximum quality score was lowest (0.833) compared to all other projects. NEXUS also had the lowest mean value of User Story quality (0.733). Quality scores of project TIMOB were also very low with mean of 0.768. Best User Story quality was in project COMPASS with mean of 0.928 and median of 1. Projects XD and DNN also had good quality User Stories as average quality value for both projects was 0.812. Perfect User Stories were found in XD, DNN, and COMPASS. The highest variation of User Story quality scores was found in project DNN and NEXUS indicated by the highest standard deviation values. TISTUD had the lowest standard deviation value (0.066). This indicates that quality scores of project TISTUD were the most stable, which in turn means that quality scores were closest to the mean value of 0.806.

In conclusion, it can be said that worst quality User Stories were found in NEXUS followed by TIMOB. Best quality User Stories were identified in COMPASS followed by XD and DNN. DNN and NEXUS showed most variation in their User Story quality scores. Most stable quality scores were found in TISTUD.

5.1.1 User Story quality

Followingly, the change of User Story quality over time is presented in Figure 5. In order to produce these visualizations, data was resampled over the frequency of semi-month (15^{th} and end of month). This was done to mimic the regular 2-week cycle of sprints. Then, mean quality of User Stories was calculated for each two-week period. Missing values were filled by propagating last valid value forward until the next existing value.

From plots presented in Figure 5, it can be observed that XD has an upward trend while COMPASS shows a downward trend. No other obvious trends can be identified. For COMPASS it is also possible to identify some seasonal fluctuation.

The constant quality value of 0.84 for DNN during period from April 2014 to December 2014 suggests that the development of DNN was very inactive for almost a year. MULE and TIMOB also have some shorter inactive periods during which no new User Stories was produced. All other projects show continuous and active development during the observed time period.

Plots in Figure 5 also confirmed some of the previous findings, like TISTUD being the project with most stable User Story quality scores. Quality scores of TISTUD show modest fluctuation with most values staying in a range of 0.75 to 0.85. Although, some outliers can also be observed. A wide range of quality scores and a numerous fluctuations in the plots of TIMOB and nexus also confirm the previous findings that these are the projects with most fluctuation in User Story quality scores. Without further activities it is difficult to identify any other obvious behaviour like trend or seasonality in the change of User Stories quality scores.

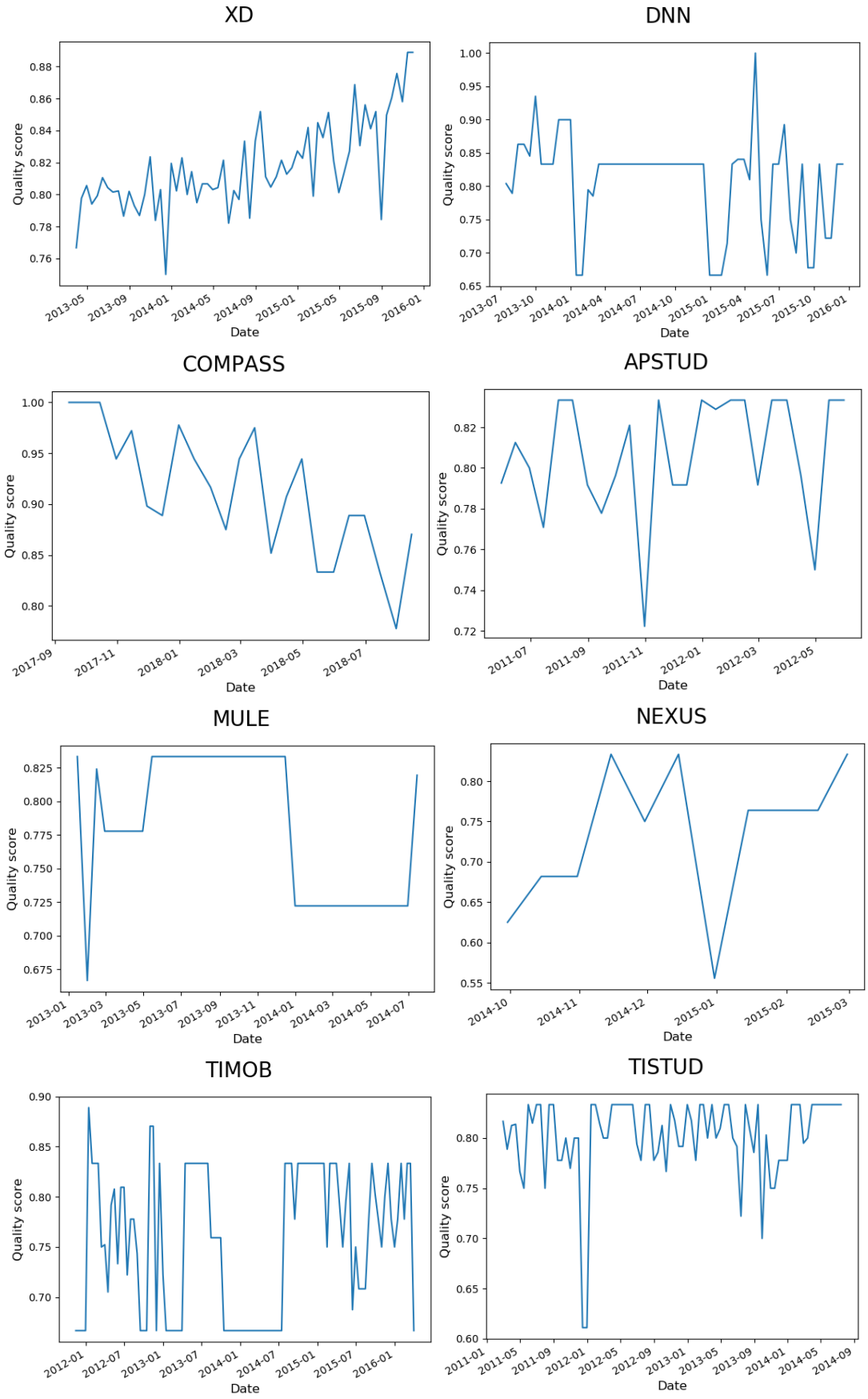


Figure 5. The change of User Story quality over time for all 8 projects.

To get first insights and to describe the quality of all 8 projects under review the quality distribution of User Stories was visualized as histograms presented in Figure 6. y axis shows the percentage of all User Stories while x axis shows defined quality intervals.

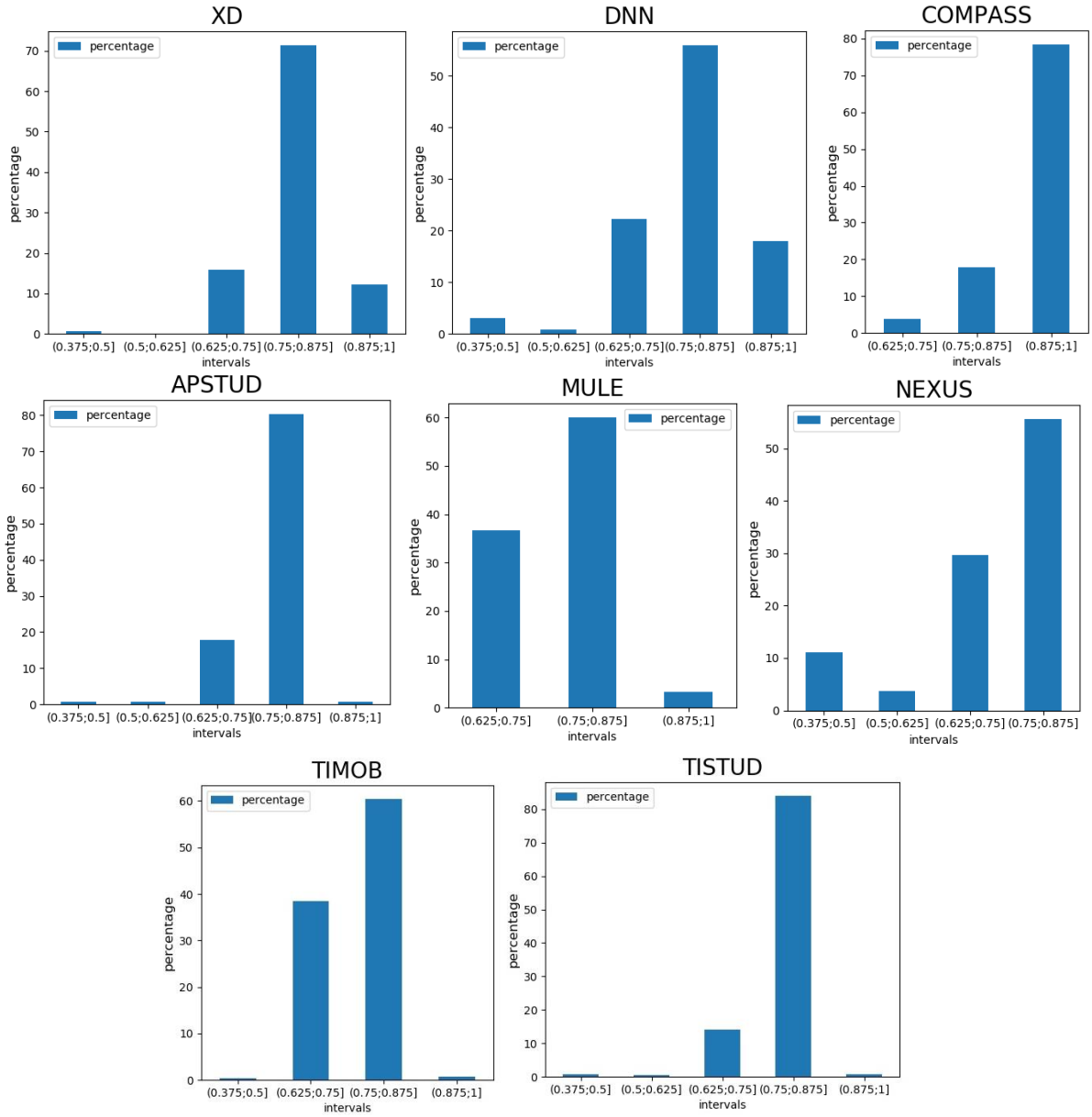


Figure 6. Distribution of User Story quality for all 8 projects.

From the histograms in Figure 6 it can again be seen that the User Story quality in different projects differs a lot. This visualization confirms that COMPASS has the best User Story quality with almost 80% of all User Stories falling into the category of highest quality scores in the range of 0.875 to 1. Quality scores for XD and DNN are also quite high. 10% of XD and 20% of DNN quality scores are falling into the category of highest quality scores. Quality scores for other projects are lower. Again, it is confirmed that NEXUS is the project with worst User Story quality with 10% of all User Stories falling into the worst category ranging from 0.375 to 0.5. NEXUS is also the only project without any User Stories in the category of highest quality scores.

5.2 Forecasting the quality of User Stories

Another important observation that can be made for forecasting purposes when observing Figure 6 is that most of the quality values seem to confirm a normal distribution with either longer right or left tail. This is the first hint that the time series data might be non-stationary. Fact that projects XD and COMPASS were showing some obvious trend in Figure 5 also suggests that the time series for these two projects might be non-stationary [14].

Followingly, results gathered by observing decomposition plots are described. An example of decomposition plot for project XD is presented in Figure 7.

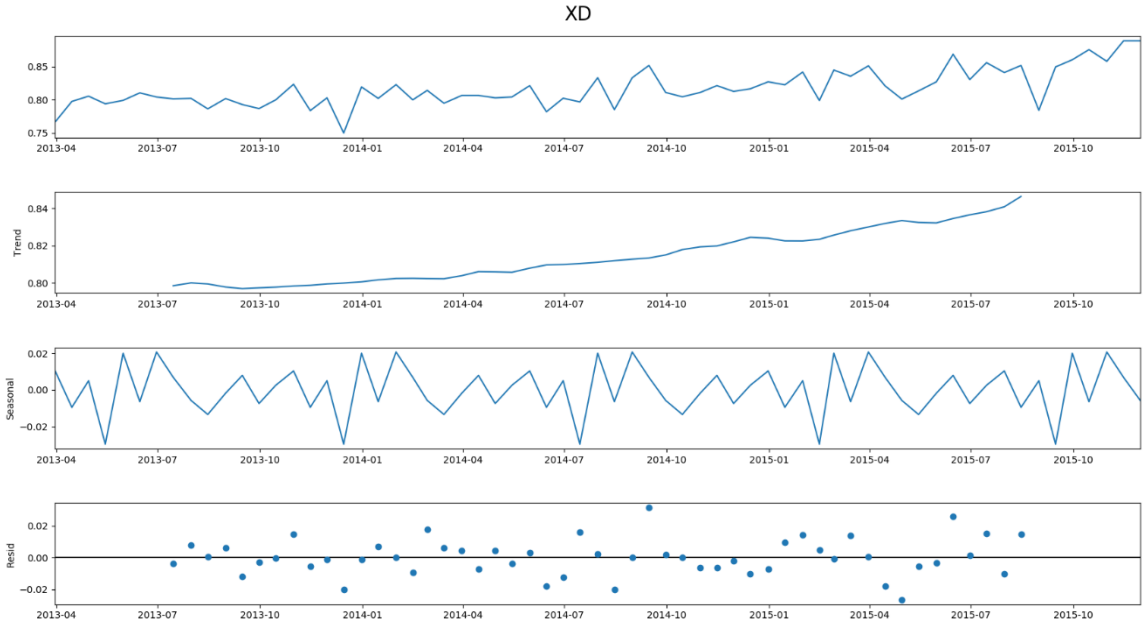


Figure 7. Decomposition of project XD.

Decomposition plots for all other projects are presented in Appendix IV. From these the most important observation is that all projects have a clear seasonality component. In addition, some of the projects, like XD and COMPASS, also have obvious trend.

The existence of seasonality component for all projects as well as trend for 2 projects indicate that the data might be non-stationary [14]. As a result of these findings SARIMA model was selected for forecasting the quality of User Stories.

At this stage, a more precise stationarity check was performed. More precisely, ADF [31, 14] tests were executed. Test results revealed that the data of 7 projects out of 8 was non-stationary and needed differencing in order to make data stationary. DNN was the only project in which data was already stationary and therefore did not require differencing.

As a result of executing procedures and search process described in analysis procedures section of the thesis best forecast models were identified. Best SARIMA forecast model for each project is presented in Table 12.

Table 12. Best SARIMA forecast models for all 8 projects.

Project	Model
XD	SARIMAX(0, 1, 1)x(0, 1, 1, 12)
DNN	SARIMAX(1, 0, 1)x(1, 1, [], 12)
COMPASS	SARIMAX(2, 1, 1)x(0, 1, 1, 6)
APSTUD	SARIMAX(0, 2, 2)x(2, 1, [], 6)
MULE	SARIMAX(1, 2, 1)x(0, 1, 1, 6)
NEXUS	SARIMAX(1, 2, 0)x(0, 1, 0, 3)
TIMOB	SARIMAX(0, 1, 1)x(0, 1, 1, 12)
TISTUD	SARIMAX(0, 1, 2)x(0, 1, [1], 12)

As described in a paper [14] SARIMA model is denoted as SARIMA(p, d, q)×(P, D, Q) where p stands for autoregressive, d for integrated, and q for moving average part of the model. P, D, and Q describe the same for seasonal component of the model. Last parameter, which is whether 3, 6 or 12, shows the frequency of the time series [32]. A more detailed description of setting the last parameter is given in the analysis process section of the thesis. As a next step forecasting was performed. In order to illustrate the comparison of forecasted values and actual values, graphical representation for project XD is shown in Figure 8

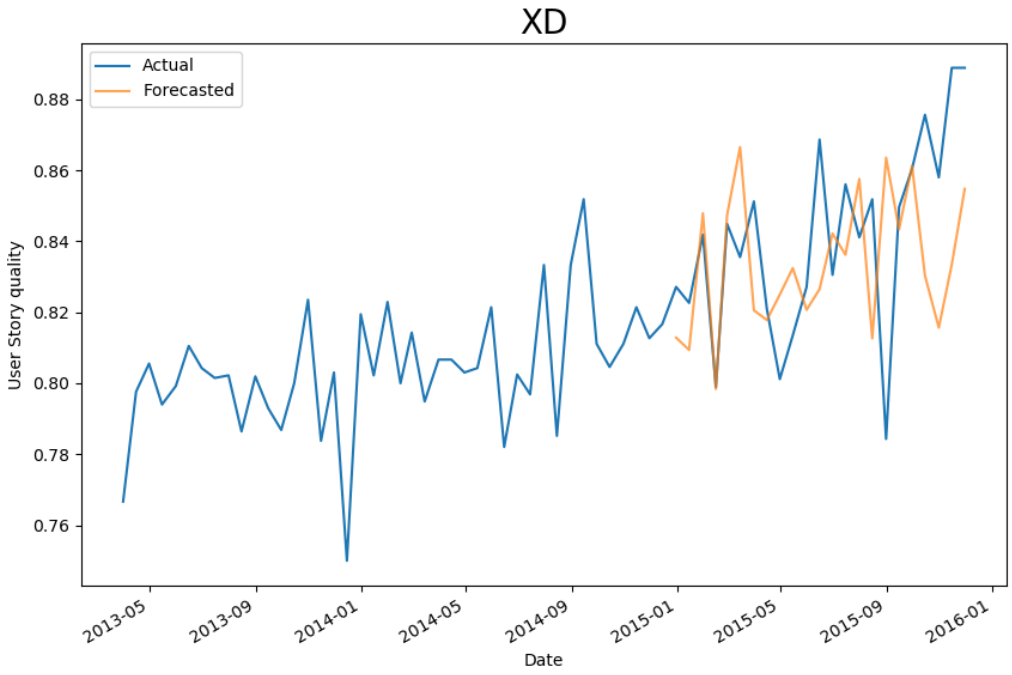


Figure 8. Graphical representation of forecasting results for project XD.

From Figure 8 it can be observed that when comparing the prediction with actual values, the prediction for XD is quite accurate as prediction line is imitating the change of actual values very well.

Table 13. Accuracy metrics for User Story quality forecasts.

Project	MAPE	Correlation	Min-Max
XD	0.028	0.163	0.027
DNN	0.080	0.376	0.072
COMPASS	0.035	0.530	0.034
APSTUD	0.054	0.324	0.053
MULE	0.033	0.242	0.033
NEXUS	0.203	0.791	0.201
TIMOB	0.053	0.494	0.050
TISTUD	0.030	0.648	0.030
Mean	0.0645	0.446	0.0625

The accuracy metrics of the best models for all 8 projects are presented in Table 13, as can be seen, the predictions for all projects are fairly accurate. Average MAPE of 0.0645 shows that models of all 8 projects are about 94.5% accurate when forecasting 15 next sprints [32]. In order to mimic the length of regular sprints average User Story quality of 2-week periods was used. NEXUS is having the worst accuracy which can be caused by the rather small amount of data for training when compared to other projects. Best accuracy is achieved by project XD with most User Stories. This suggests that more data for training the model results in a more accurate model, as can be expected.

5.3 Cross-correlation analysis

As stated in RQ2, one of the purposes of the analysis was to find if User Story quality can be related to number of bugs, rework, and delays. No obvious relationships or interesting patterns can be identified by only observing and comparing the plots of the three aspects and the change of User Story quality scores plots presented in Figure 5. Plots for number of bugs are available in Appendix I, plots for number of rework cases in Appendix II, and number of delayed cases in Appendix III.

Since the plots did not reveal any evident patterns, Windowed Time Lagged Cross Correlation method (WTLCC) [27] was applied. The following sections describe the results obtained.

5.3.1 Number of bugs

This section describes the results of applying WTLCC on User Story quality and number of bugs. To visualize the results, heatmaps are used. To illustrate the idea, Figure 9 shows a heatmap of project COMPASS. In the figure, blue colour indicates that User Story quality was leading the interaction while red indicates the number of bugs.

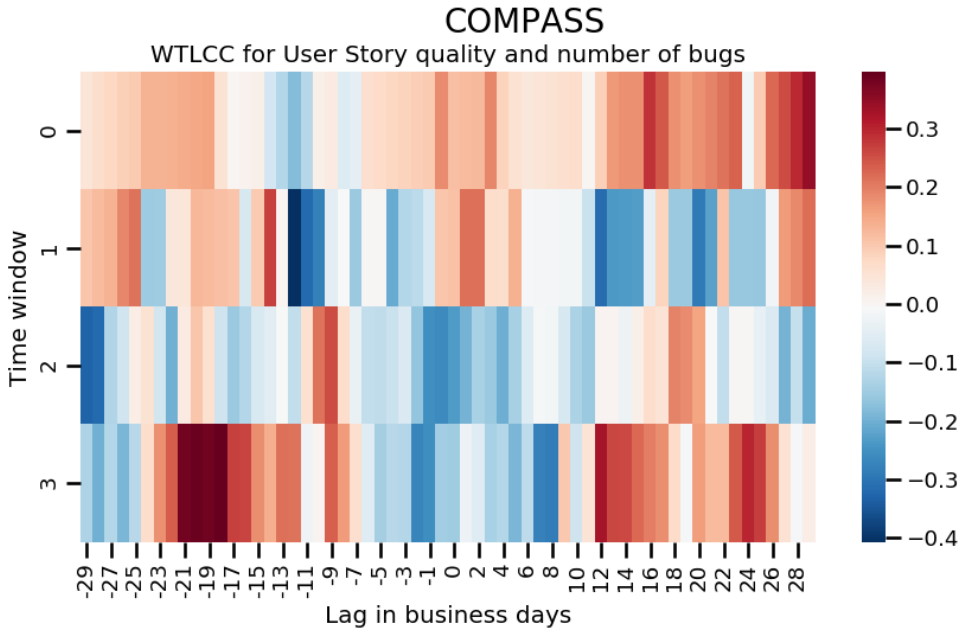


Figure 9. COMPASS WTLCC heatmap for User Story quality and number of bugs.

In Figure 9 WTLCC results are visualized in setting 1, which means that a possible correlation is searched in 4 time windows (y axis on the plots). Window 0 represents the starting phase of active development, windows 1 and 2 are middle parts when development was most active and window 3 represents the ending of active development. Lag in business days (x axis on the plots) indicates how many business days it took for one interaction to mimic the other. Correlation value can be identified from the colour bar on the right. Other analysed heatmaps are presented in Appendix V to X, which correspond to the 2 different setting (4-window and quarters). In this analysis, most emphasis is on windows 1 and 2 in order to find possible patterns during the most active part of the development.

For all projects the correlation of User Story quality and number of bugs ranges from -0.6 to 0.6. During the analysis some interesting patterns were identified for 6 out of the 8 projects. For COMPASS, large negative correlation can be seen (visualized as dark blue on the plot) in window 1 with a negative lag of -13 to –11 business days (r = -0.4). High negative correlation shows that first User Story quality decreased. After 13 to 11 business days number of bugs increased similarly. A similar pattern of large correlation is recognizable for MULE in window 2 with a negative lag of -23 to -14 business days (r = -0.6). NEXUS also shows a similar pattern of large correlation with a negative lag of -3 business days (r = -0.6) in window 2. When looking at windows 1 and 2 of APSTUD and checking the results at lag of -17 to -14 (r = -0.2) small increase in correlation similar to COMPASS, NEXUS, and MULE wan be observed. As said before, such pattern with negative lag shows that when the User Story quality decreased, after about 3-4 weeks (about 1 week in case of NEXUS), the number of bugs increased correspondingly.

All projects where such patterns were identified had a rather short active development period meaning that splitting these projects into 4 time windows resulted with windows with approximate length of one quarter of year. In order to get similar insight of projects with longer development period XD, DNN, TIMOB, and TISTUD are also visualised in setting 2 (available in Appendix VI) where one window represents one quarter of a year. Project XD, despite having a large amount of User Stories, shows no interesting patterns. This can be affected by the fact that XD had very few bugs (476) compared to the number of User Stories (2135). For DNN such pattern with medium correlation is identified in window 2

with a negative lag of about -14 business days (r = -0.3). TISTUD is showing the same pattern with large correlation in window 6 with a negative lag of -20 (r = -0.4) and in window 8 with a negative lag of -20 business days. (r = -0.4). TIMOB shows evidence of such patterns with medium correlations in windows 2, 14 and 15 with negative lag of -18 (r = -0.2), -19 (r = -0.4) and -14 (r = -0.2) business days, respectively. This means that longer projects show evidence of the same pattern found in shorter projects. In conclusion, it can be said that for 6 projects out of 8 similar patterns were found. These patterns show that approximately 3-4 weeks after the decrease of User Story quality number of bugs increased correspondingly.

5.3.2 Rework

This section shows the results of applying WTLCC on User Story quality and number of rework cases. In Appendix VII, WTLCC results plotted in setting 1 can be observed. When looking at the results of COMPASS, medium correlation at window 2 with lag of -12 business days (r = -0.3) can be identified. Another medium correlation can be spotted in window 3 with lag of -27 business days (r = -0.3). Similar pattern with a slightly larger correlation is identifiable for APSTUD in window 1 at the lag of -18 business days (r = -0.4). For MULE it can be seen that in window 2 User Story quality is strongly leading the interaction at lag of approximately -30 business days (r = -0.4). XD, DNN, TIMOB, and TISTUD are additionally plotted in setting 2 (available in Appendix VIII) in order to make them more comparable with shorter projects. For XD previously described patterns with large correlation can be identified in window 1 with lag of -27 (r = -0.6) and in window 8 with lag of -32 (r = -0.6). For DNN such patterns with medium correlation can be identified in window 8 with lag of -30 business days (r = -0.4). TIMOB shows such patterns with medium correlation at windows 13 and 16 with lag of -21 (r = -0.4) and -27 (r = -0.3) respectively. For TISTUD it can be seen that User Story quality is leading the interaction with strong correlation in several time windows: 6, 8, 9 and 10 with lag of -28 (r = -0.5), -26 (r = -0.5), -32 (r = -0.5) and -29 (r = -0.5) respectively.

From these observations it is possible to generalize that similar patterns as described for number of bugs were found for User Story quality and number of rework cases. The main difference is that lag values are larger (ranging from -12 to -32) meaning that it takes longer for User Story quality to show a relation with rework. After a decrease in User Story quality, it takes roughly 4-6 weeks until increase in rework happens which seems valid as the rework is often discovered by the end-user after new functionality has been developed and released.

5.3.3 Delays

This section describes applying WTLCC on User Story quality and number of delayed cases. In Appendix IX WTLCC results plotted in setting 1 can be observed. COMPASS, APSTUD, MULE, and NEXUS are analysed in setting 1. COMPASS shows some medium negative correlation in window 1 with lag of -9 business days (r = -0.2). Another medium negative correlation can be spotted in window 2 with lag of -11 business day (r = -0.4). The same happens in window 3 at lag of -8 (r = -0.4). An important aspect to notice is that correlation is more spread out and for all three windows User Story quality is strongly leading the interaction. For APSTUD large negative correlation in window 1 with a negative lag of -7 (r = -0.6) can be identified. The same pattern of large negative correlation is noticeable in window 3 at lag of -7 (r = -0.6) and again correlation is more spread out and User Story quality is leading the interaction. For MULE lack of data is affecting the results but in window 2 it is again clearly visible that correlation is more spread out and interaction is led by User Story quality (r = -0.1). When looking at windows 1 and 2 of NEXUS, it can again be

seen that a similar pattern with medium correlation exists and User Story quality is again leading the interaction (r = -0.2 to -0.4).

As before, longer projects were observed in setting 2 (available in Appendix X). For XD previously described pattern is not noticeable. For DNN spread out medium negative correlation in window 2 with lag ranging between -30 and -27 (r = -0.2 to -0.4) can be identified. Additionally, a similar pattern can be observed in window 7 with lag ranging between -30 and -24 (r = -0.4 to -0.5). For TIMOB such spread out pattern can be seen in window 1 with medium negative lag between -14 and -8 (r = -0.2 to -0.4). TISTUD also shows such patterns with medium correlation in window 2 with negative lag between -22 and -16 (r = -0.2 to -0.4), in window 4 with negative lag of -27 to -22 (r = -0.4) and in window 5 with negative lag of -25 to -22 (r = -0.2 to -0.4).

When we compare these results with the results of previously described number of bugs and rework, we can see that correlation is more spread out meaning that User Story quality is leading the interaction for a longer time and has therefore a wider range of lag. This can be caused by the fact that delays can be caused by an increased number of bugs, increased amounts of rework as well as other factors. As previous observations revealed a relationship between User Story quality and number of bugs is revealed faster (3-4 weeks) while relationship with rework takes longer (4-6 weeks) to reveal. Delays are the consequence of high number of bugs, rework cases, and other factors that justifies the more spread out correlation with a wider range of lags.

6 Discussion

This section answers the research questions and gives an overview of the lessons learned and limitations of the used approach.

6.1 Answering the research questions

The first research question (RQ1) asked if it is possible to forecast the change of User Story quality for monitoring purposes. Accuracy metrics presented in Table 13 revealed that the accuracy of forecasting models for forecasting User Story quality was good. The best accuracy was achieved for project XD, which had the largest number of User Stories. NEXUS forecast had the worst accuracy when compared to other projects. This could be caused by the fact that NEXUS also had the smallest number of User Stories when compared to other projects. From aggregated mean MAPE value, it was possible to conclude that on average 94.5% of forecasted quality values were accurate when forecasting 15 next sprints. Here, one sprint refers to 2 week period as it is the regular length of sprints in agile software projects. This means that it is possible to forecast the quality of User Stories with good accuracy for more than half a year, more precisely for 6 to 7 months. This is an important finding as it would enable team leaders, product owners and other similar roles to monitor the change of User Story quality during the development process. Problems with User Stories syntactic and some pragmatic criterion would be identified as early as possible. As a result, team leaders, product owners, and other stakeholders in similar roles would be able to execute changes in order to fix sources of bad quality User Stories. For example, notifying developers and analysts about the issue.

The second research question asked about the relationship between User Story quality and other aspects of software development. The aspects studied in this thesis were the number of bugs, rework done (re-opened issues) and delays (based on due date). The results revealed that for most of the projects it was possible to identify similar patterns. Cross-correlation analysis revealed that if the quality of User Stories decreased similar increase in number of bugs happened approximately 3 to 4 weeks after the decrease in the quality of User Stories. Similar relationship was identified when comparing User Story quality and number of rework cases but, in this case, it took longer until increase in rework cases happened. If User Story quality decreased number of Rework cases increased similarly after about 4 to 6 weeks.

Delays are a consequence of high number of bugs, high number of rework cases, and other aspects. This can be the reason why more spread out correlation was discovered for delays. This means that User Story quality was leading the interaction with delays for a longer time when compared to the results of number of bugs and rework.

It is worth noting that one of the projects showed different behaviour. For project XD the above described patterns either did not apply or were rare when compared to other projects. This can be caused by the fact that XD was also the only project showing an upward trend in the change of User Story quality over time. This in turn, indicates that the quality of User Stories was increasing rather than decreasing for XD. During the analysis, situations where User Stories quality decreased were analysed. As XD showed upward trend there were less such situations for XD which can be the reason why fewer interesting patterns were identified for this project when compared to other projects.

6.2 Lesson learned

Although it was possible to produce accurate prediction models, the process itself was complicated containing numerous steps like data pre-processing, data cleaning, setting up

AQUSA tool for User Story quality assessment, proposing formulas for quantifying the quality scores, and performing forecasting. Therefore, it can be said that while it is possible to forecast the quality of User Stories more convenient solutions should be developed in order to make monitoring of User Stories simple for development teams. As stated in [8] AQUSA provides an API that makes it possible to integrate AQUSA with work management tools like Jira. In order to automate forecasting and make monitoring of User Story quality possible additional functionalities should be automated like quantifying the quality of User Stories as well as producing the forecast models and forecast plots.

In order to accurately forecast the quality of User Stories a considerable amount of training data was needed. This could be an issue when using described approach for monitoring purposes as forecasts would be inaccurate during the early phase of development.

6.3 Limitations

Results of the thesis are interesting but limited to the data sample. Larger data sample could have been used in order to produce better possibilities for generalizing the results, but it was proved difficult to find open-source projects with usable data for this kind of analysis. Used dataset required an extensive amount of data cleaning and it was not possible to discover all existing issues during the data cleaning which could also have affected the results. Missing values are also a threat to validity. Although imputing methods were used missing data can still cause inaccuracies.

Missing data points could also have affected the procedures performed for finding relationships between the quality of User Stories and other aspects listed in RQ2. Although many similar patterns were discovered, used approach required manual interpretation of visualized results which could have introduced errors.

Another limitation is related to the selected tool. Three software aspects from RQ2 are analysed considering limitations of AQUSA tool and current state of technological knowledge. Trustworthily assessing sematic criterion is not possible considering the current technology and knowledge as it would require expert domain or using artificial intelligence. Consequently, authors of AQUSA tool [8] found it not possible to assess sematic criterion while keeping recall close to 100%. But it was possible to measure syntactic and some pragmatic criterion of User Stories.

7 Conclusion and future work

This chapter gives a summary of findings and highlights relations with existing evidence.

7.1 Summary of findings

The results revealed that, on average, produced models were 94.5% accurate when forecasting the quality of User Stories for the 15 next sprints (with a length of 2 weeks). As a result, it can be said that it is possible to predict the quality of User Story for the next 6 to 7 months with good accuracy.

Regarding individual projects, XD showed best accuracy predictions while NEXUS showed worst accuracy. As expected, project with most data achieved best accuracy and project with least amount of data points worst accuracy.

The correlation analysis showed several interesting relationships between User Story quality and other software development aspects such as the number of bugs, rework, and delays. The results show that, when the quality of User Stories decreased, the number of bugs increased correspondingly after approximately 3 to 4 weeks. Regarding the number of rework cases, a similar pattern was found after 4 to 6 weeks. Therefore, it can be said that the relationship with rework takes longer to reveal. This finding can be explained by the fact that bugs are usually found in a testing phase while rework usually happens after new functionality has been developed and released. Regarding delays, the correlations were more spread out indicating that User Story quality was leading interaction for a longer time. This finding can be explained by the fact that delays are usually caused by various factors including a high number of bugs and increased amounts of rework.

7.2 Relation to existing evidence

The need for predicting and monitoring software development metrics is highlighted in recent paper [25]. This thesis addresses the stated issue by proposing a way for quantifying the quality of User Stories as well as producing accurate forecast models of User Story quality for monitoring purposes.

Analysis is also related to a previous study [21] executed by the authors of QUS framework and AQUSA tool. Longer description of the content of this analysis is given in the Earlier work chapter of this thesis. In this previous analysis [21] authors were able to find mildly positive effects of applying QUS framework and AQUSA tool but no significant changes in the number of issues, defects, rework, velocity and comments. Authors also encouraged others to execute similar studies. In our work larger sample group of was used. User Stories quality was analysed in 8 different open source agile software projects. Although, discovered relationships were not strong some interesting patterns were discovered indicating that User Story quality scores are related to the number of bugs, rework, and delays.

References

[1] M. Cohn, User Stories Applied: for agile software development, Redwood City, CA, USA: Addison Wesley, 2009.

[2] T. Dingsøyr, S. P. Nerur, V. Balijepally and N. B. Moe, "A decade of agile methodologies: Towards explaining agile software development," *Journal of Systems and Software,* vol. 85, no. 6, pp. 1213-1221, 2012.

[3] M. Kassab, "The changing landscape of requirements engineering practices over the past decade," in *2015 IEEE Fifth International Workshop on Empirical Requirements Engineering (EmpiRE)*, 2015.

[4] X. Wang, L. Zhao, Y. Wang and J. Sun, "The Role of Requirements Engineering Practices in Agile Development: An Empirical Study," *In: D. Zowghi and Z. Jin, editor(s). Requirements Engineering . Springer; 2014. p. 195-209.*, pp. 195-209, 2014.

[5] M. Kassab, "An Empirical Study on the Requirements Engineering Practices for Agile Software Development," in *2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications*, 2014.

[6] CollabNet VersionOne, "The 13th Annual State of Agile Report," CollabNet VersionOne, Alpharetta, GA, USA, 2019.

[7] G. Lucassen, F. Dalpiaz, J. M. Werf and S. Brinkkemper, "The Use and Effectiveness of User Stories in Practice," in *REFSQ 2016 Proceedings of the 22nd International Working Conference on Requirements Engineering: Foundation for Software Quality - Volume 9619*, 2016.

[8] G. Lucassen, F. Dalpiaz, J. M. V. D. Werf and S. Brinkkemper, "Improving agile requirements: the Quality User Story framework and tool," *Requirements Engineering,* vol. 21, no. 3, pp. 383-403, 2016.

[9] B. Wake, "INVEST in Good Stories, and SMART Tasks," 17 8 2003. [Online]. Available: https://xp123.com/articles/invest-in-good-stories-and-smart-tasks/. [Accessed 14 11 2019].

[10] "Automatic Quality User Story Artisan Prototype," Utrecht University, 2016. [Online]. Available: https://github.com/gglucass/AQUSA. [Accessed 14 11 2019].

[11] D. Firesmith, "Common Requirements Problems, Their Negative Consequences, and the Industry Best Practices to Help Solve Them," *The Journal of Object Technology,* vol. 6, no. 1, pp. 17-33, 2007.

[12] A. Aurum and C. Wohlin, Engineering and Managing Software Requirements, 2005.

[13] G. Lucassen, F. Dalpiaz, J. M. E. v. d. Werf and S. Brinkkemper, "Forging high-quality User Stories: Towards a discipline for Agile Requirements," in *2015 IEEE 23rd International Requirements Engineering Conference (RE)*, 2015.

[14] R. Adhikari and R. K. Agrawal, An Introductory Study on Time Series Modeling and Forecasting, 2013.

[15] L. Buglione and A. Abran, "Improving the User Story Agile Technique Using the INVEST Criteria," in *2013 Joint Conference of the 23rd International Workshop on Software Measurement and the 8th International Conference on Software Process and Product Measurement*, 2013.

[16] P. L. d. Souza, A. F. d. Prado, W. L. d. Souza, S. M. d. S. F. Pereira and L. F. Pires, "Improving Agile Software Development with Domain Ontologies," in *15th International Conference on Information Technology : New Generations, ITNG 2018*, 2018.

[17] S.-T. Lai, "A User Story Quality Measurement Model for Reducing Agile Software Development Risk," *International Journal of Software Engineering & Applications,* vol. 8, no. 2, pp. 75-86, 2017.

[18] G. Rodríguez-Pérez, G. Robles, A. Serebrenik, A. Zaidman, D. M. Germán and J. M. Gonzalez-Barahona, "How bugs are born: a model to identify how bugs are introduced in software components," *Empirical Software Engineering,* vol. 25, no. 2, pp. 1294-1340, 2020.

[19] T. Sedano, P. Ralph and C. Peraire, "Software development waste," in *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*, 2017.

[20] M. I. Kamata and T. Tamai, "How Does Requirements Quality Relate to Project Success or Failure," in *15th IEEE International Requirements Engineering Conference (RE 2007)*, 2007.

[21] G. Lucassen, F. Dalpiaz, J. M. E. M. v. d. Werf and S. Brinkkemper, "Improving User Story Practice with the Grimm Method: A Multiple Case Study in the Software Industry," in *Proceedings of the 23rd International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ'17)*, 2017.

[22] W. Wu, W. Zhang, Y. Yang and Q. Wang, "Time series analysis for bug number prediction," in *The 2nd International Conference on Software Engineering and Data Mining*, 2010.

[23] H. Kai, Q. Zhengwei and L. Bo, "Network Anomaly Detection Based on Statistical Approach and Time Series Analysis," in *2009 International Conference on Advanced Information Networking and Applications Workshops*, 2009.

[24] I. Herraiz, J. M. Gonzalez-Barahona and G. Robles, "Forecasting the Number of Changes in Eclipse Using Time Series Analysis," in *Fourth International Workshop on Mining Software Repositories (MSR'07:ICSE Workshops 2007)*, 2007.

[25] M. Choraś, R. Kozik, M. Pawlicki, W. Hołubowicz and X. Franch, "Software Development Metrics Prediction Using Time Series Methods," in *Computer Information Systems and Industrial Management, 18th International Conference, CISIM 2019: Belgrade, Serbia, September 19–21, 2019: proceedings*, 2019.

[26] R. Fairley and M. Willshire, "Iterative rework: the good, the bad, and the ugly," *IEEE Computer,* vol. 38, no. 9, pp. 34-41, 2005.

[27] J. H. Cheong, "Four ways to quantify synchrony between time series data," Towards Data Sience, 13 May 2019. [Online]. Available: https://towardsdatascience.com/four-ways-to-quantify-synchrony-between-time-series-data-b99136c4a9c9. [Accessed 14 March 2020].

[28] S. Porru, A. Murgia, S. Demeyer, M. Marchesi and R. Tonelli, "Estimating Story Points from Issue Reports," in *Proceedings of the The 12th International Conference on Predictive Models and Data Analytics in Software Engineering*, 2016.

[29] E. Scott and D. Pfahl, "Using developers' features to estimate story points," in *Proceedings of the 2018 International Conference on Software and System Process*, 2018.

[30] B. Sigurd, M. Eeg-Olofsson and J. v. d. Weijer, "Word length, sentence length and frequency – Zipf revisited," *Studia Linguistica,* vol. 58, no. 1, pp. 37-52, 2004.

[31] E. E. Holmes, E. J. Ward and M. D. Scheuerell, Applied time series analysis for fisheries and environmental data, Seattle: Northwest Fisheries Science Center, 2020.

[32] S. Prabhakaran, "ARIMA Model – Complete Guide to Time Series Forecasting in Python," Machine Learning Plus, [Online]. Available: https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/. [Accessed 29 March 2020].

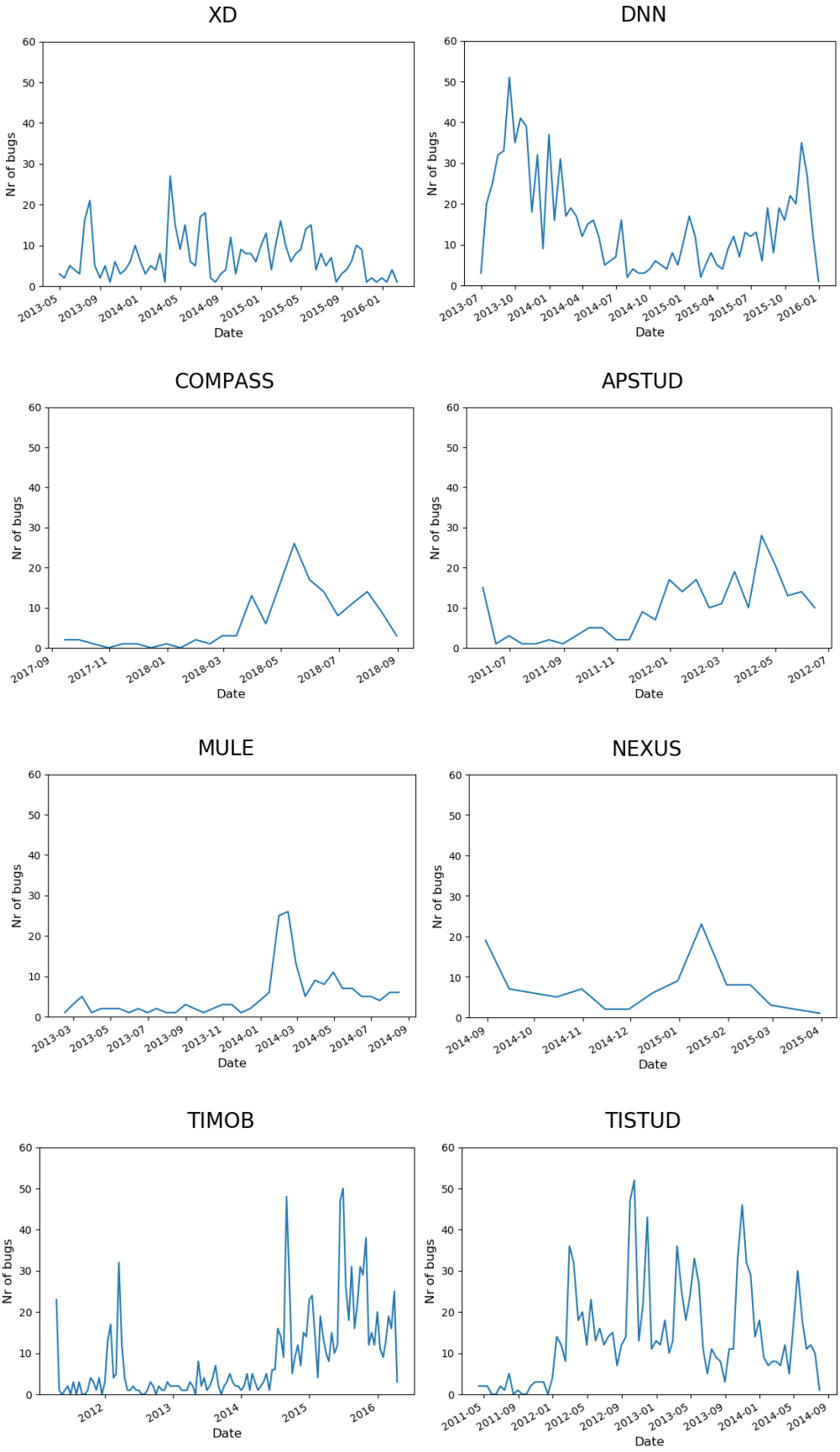
[33] J. L. Rodgers and W. A. Nicewander, "Thirteen Ways to Look at the Correlation Coefficient," *The American Statistician,* vol. 42, no. 1, pp. 59-66, 1988.

[34] J. Cohen, Statistical Power Analysis for the Behavioral Sciences, 1969.

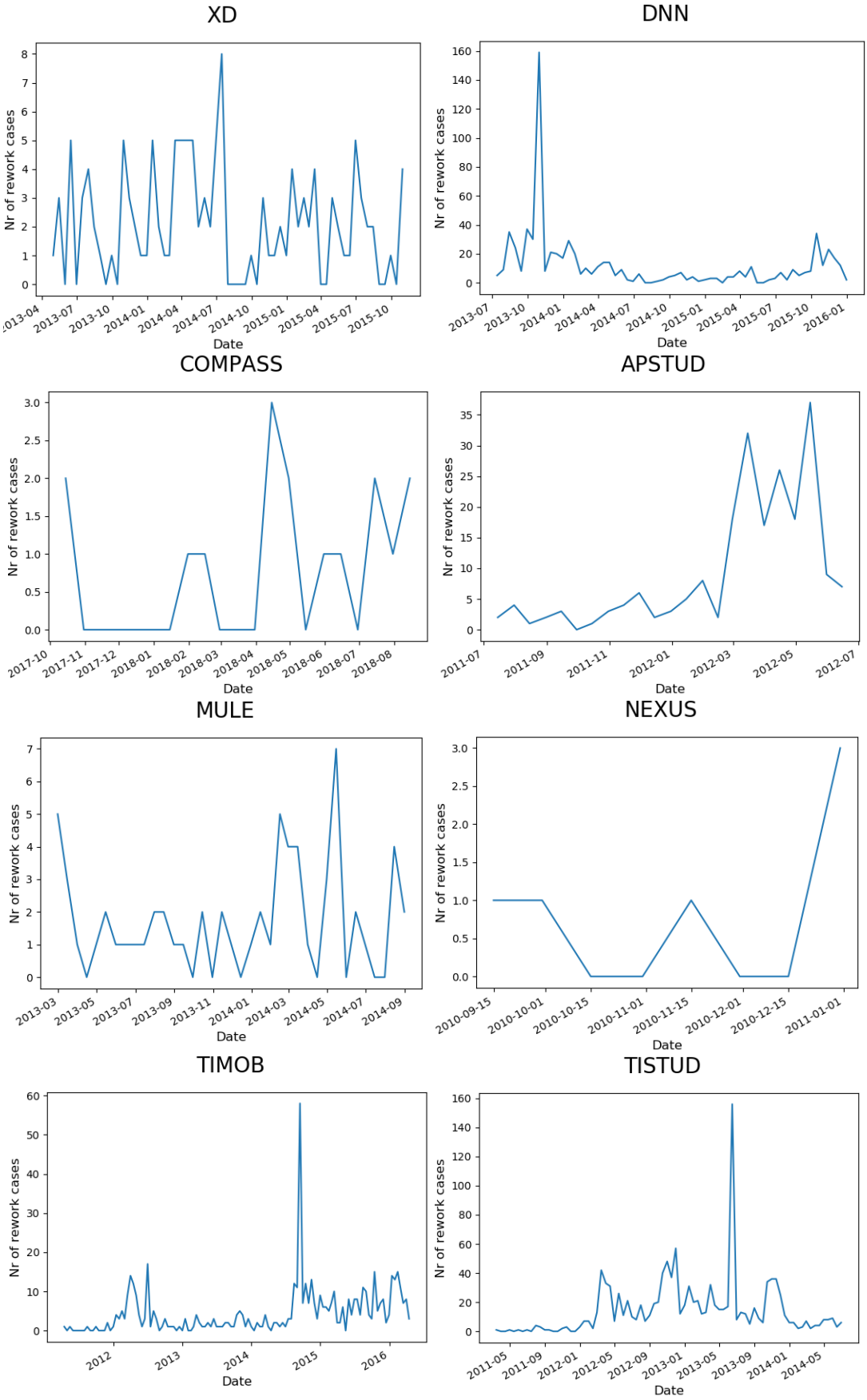
[35] J. I. Cohen, "A power primer.," *Psychological Bulletin,* vol. 112, no. 1, pp. 155-159, 1992.

Appendix

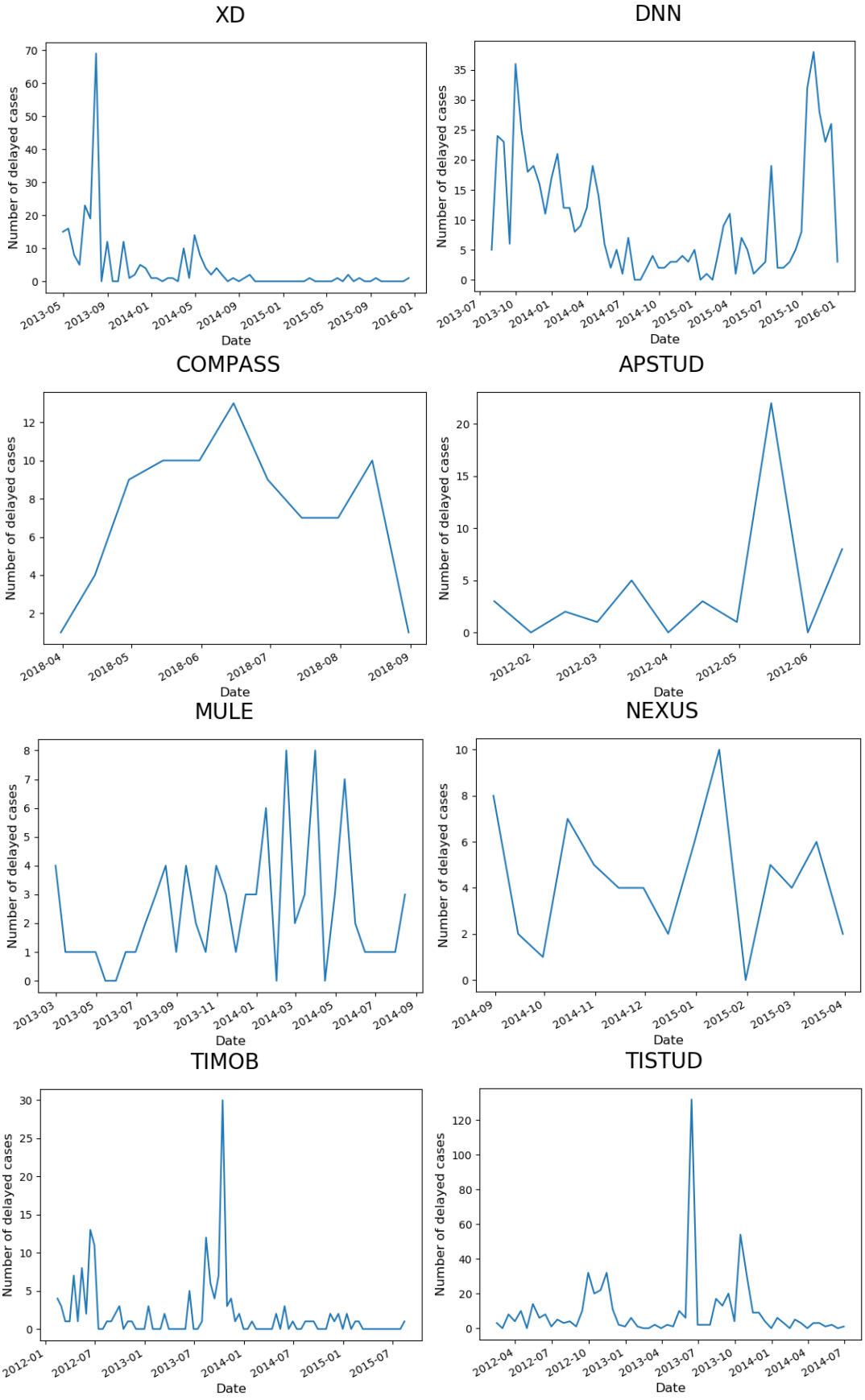
I. Number of bugs for all 8 projects under review



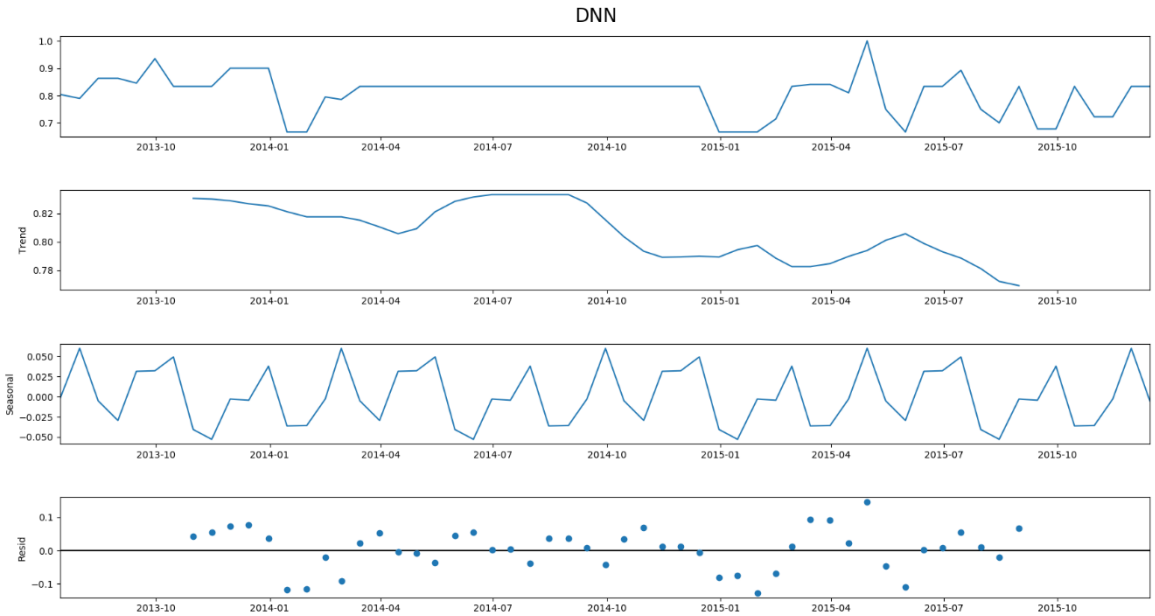
II. Number of rework cases for all 8 projects under review

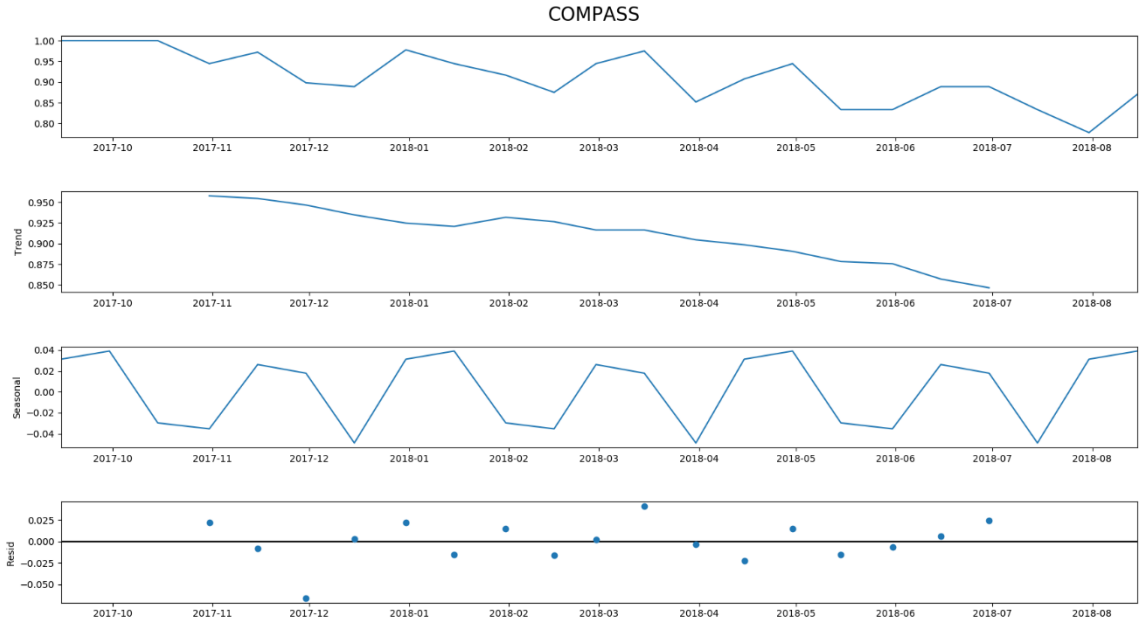


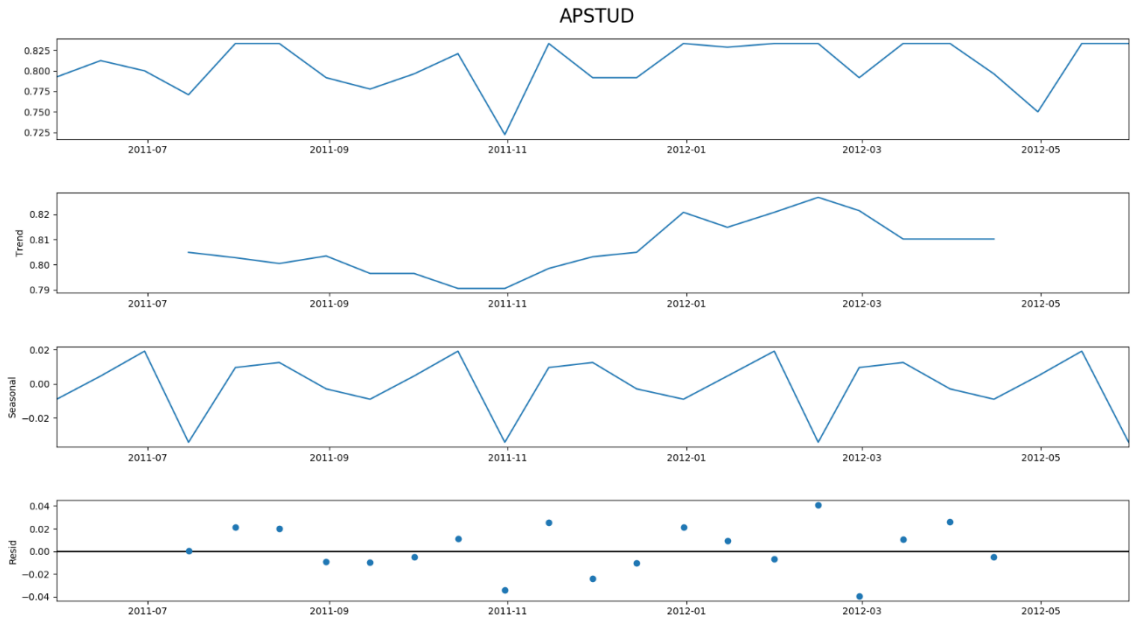
III. Delays for all 8 projects under review

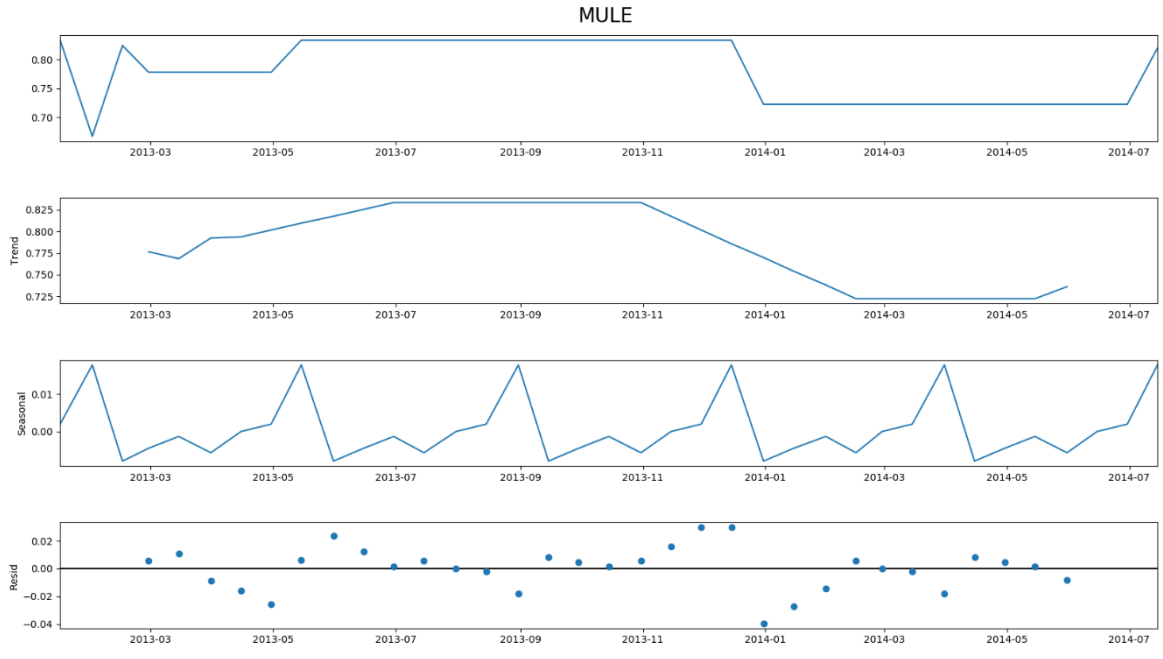


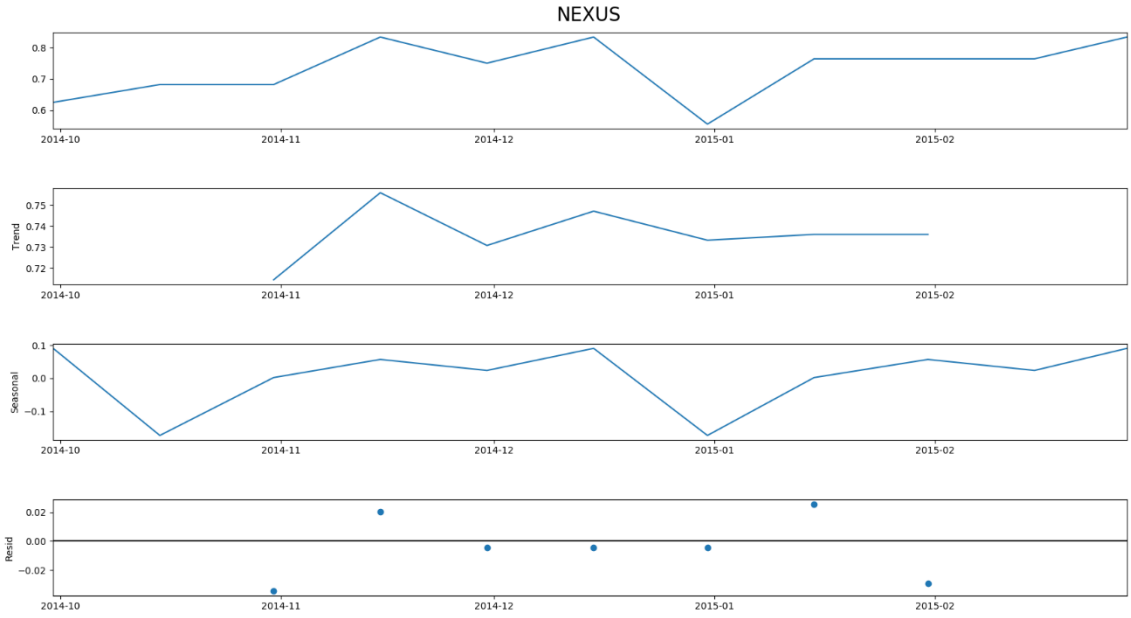
IV. Decomposition plots for all 8 projects under review

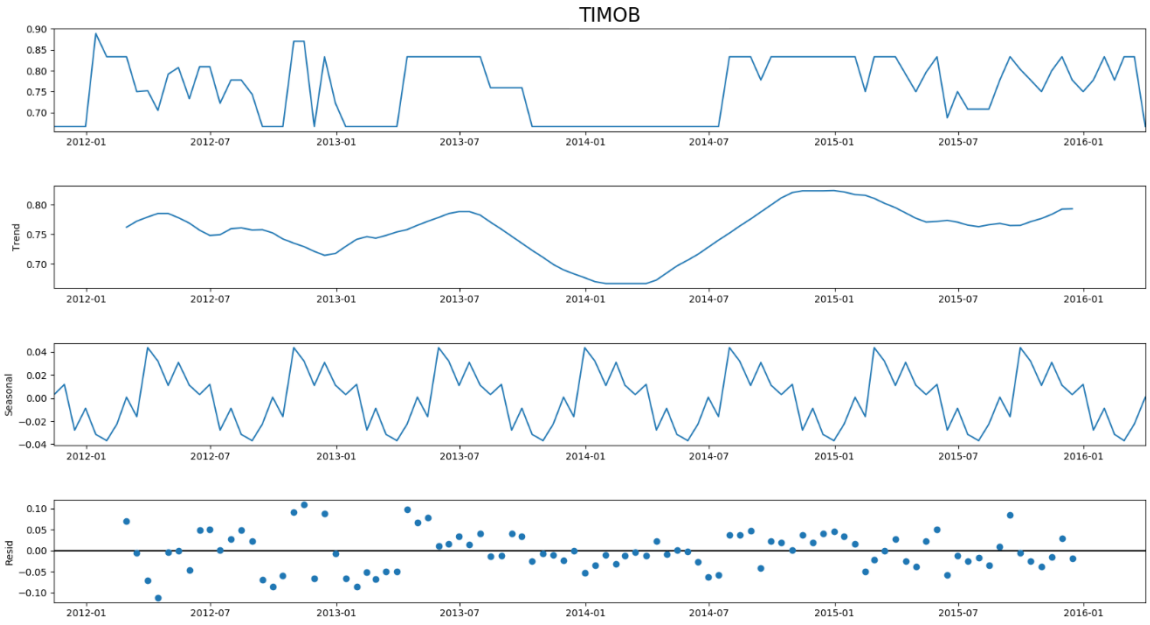


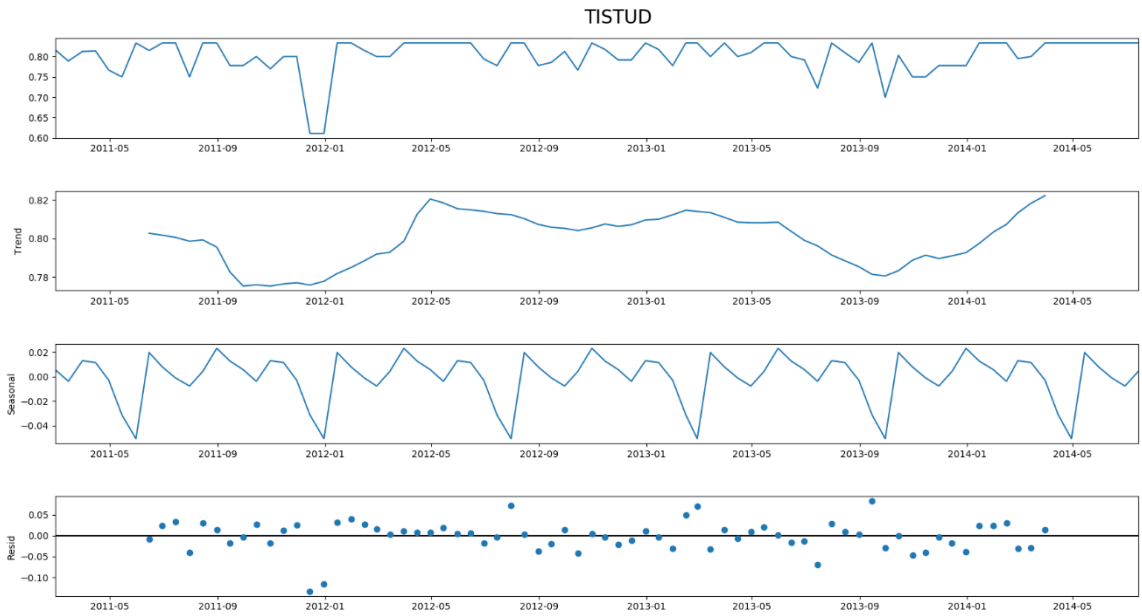



APSTUD
Trend
Seasonal
Resid
2011-07
2011-09
2011-11
2012-01
2012-03
2012-05

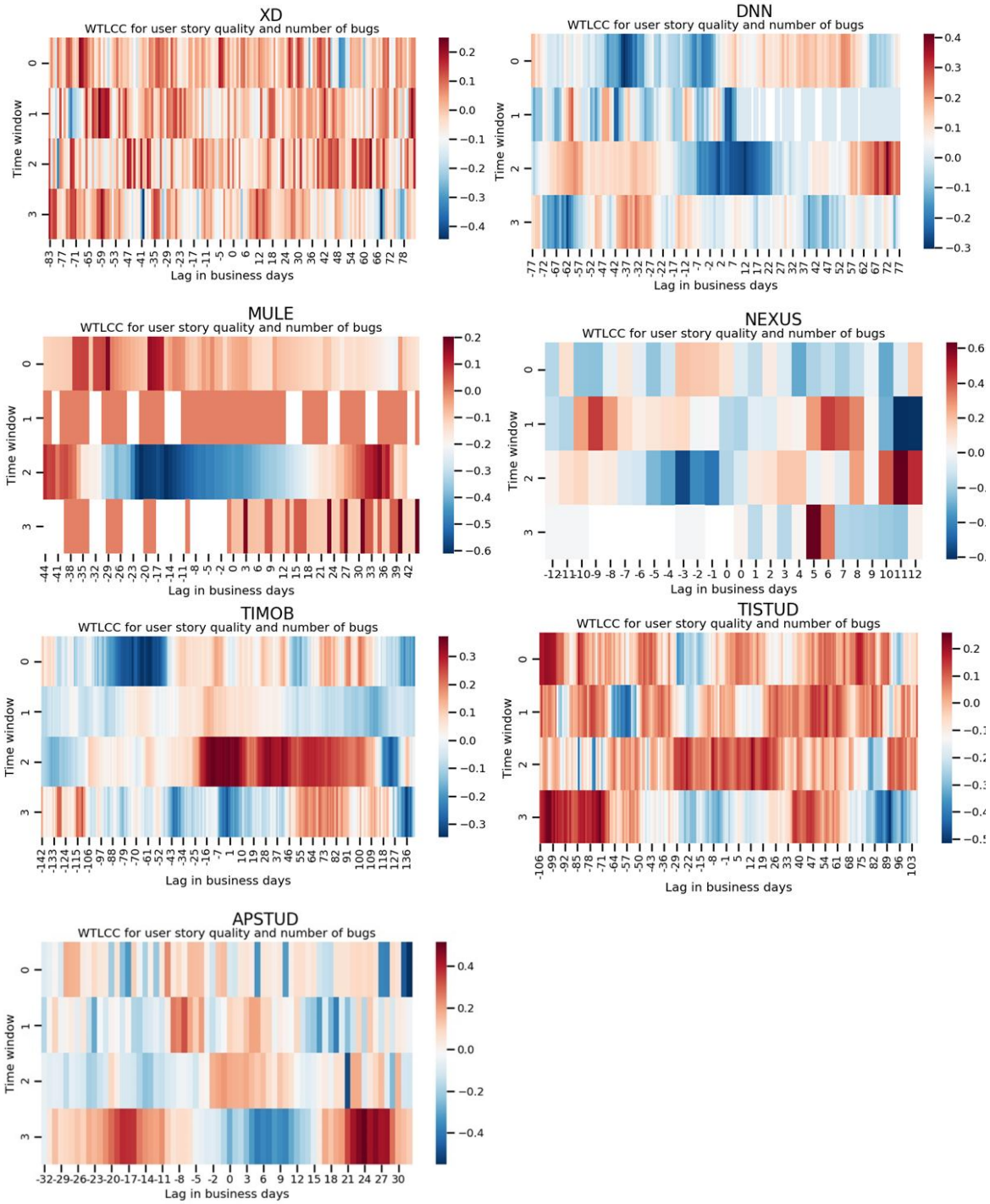

MULE
Trend
Seasonal
Resid
2013-03
2013-05
2013-07
2013-09
2013-11
2014-01
2014-03
2014-05
2014-07


NEXUS
Trend
Seasonal
Resid
2014-10
2014-11
2014-12
2015-01
2015-02

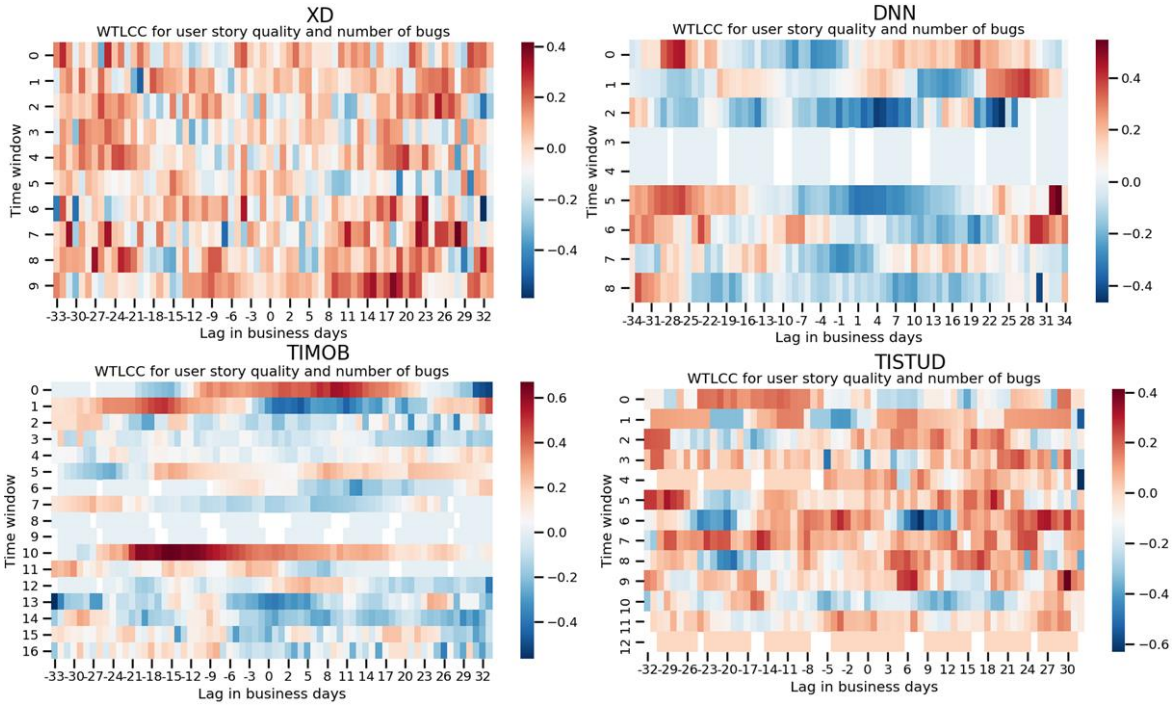

TIMOB
Trend
Seasonal
Resid
2012-01
2012-07
2013-01
2013-07
2014-01
2014-07
2015-01
2015-07
2016-01


TISTUD
Trend
Seasonal
Resid
2011-05
2011-09
2012-01
2012-05
2012-09
2013-01
2013-05
2013-09
2014-01
2014-05

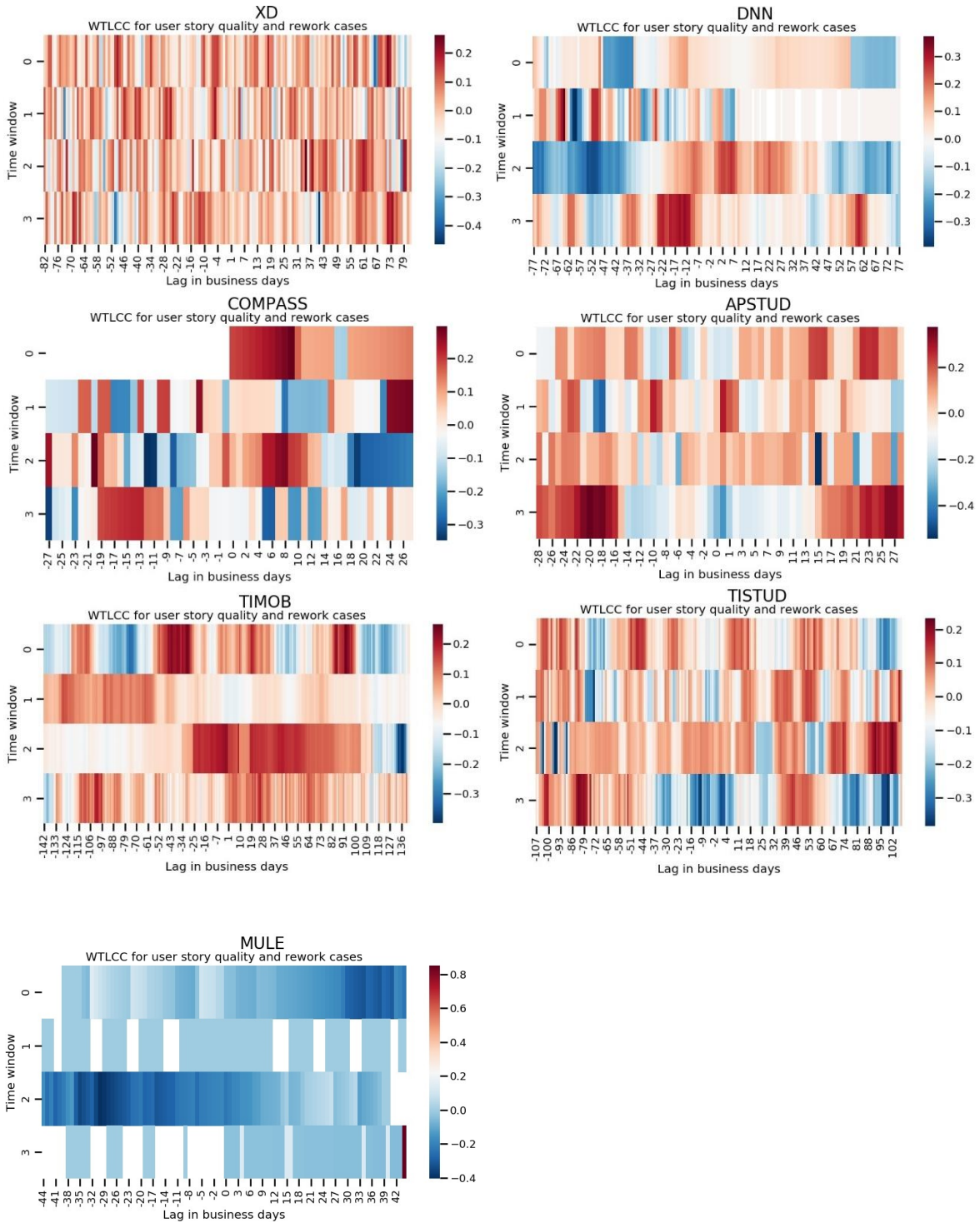
V. WTLCC plots for User Story quality and bugs in setting 1



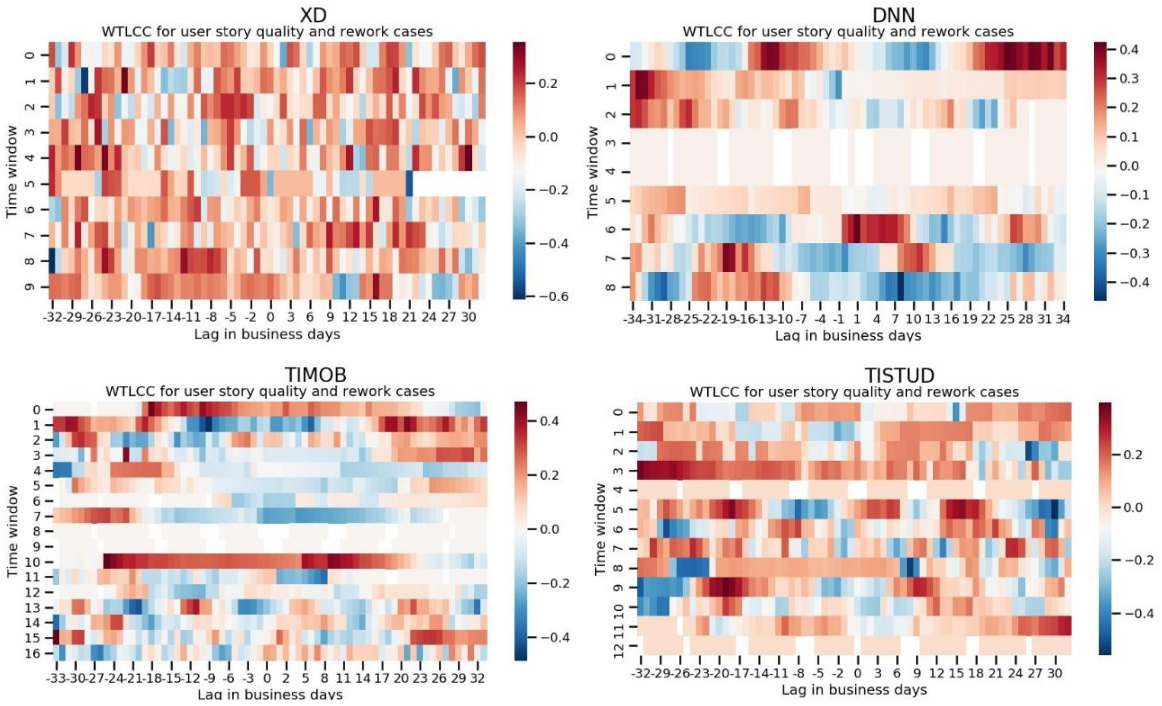
VI. WTLCC plots for User Story quality and bugs in setting 2



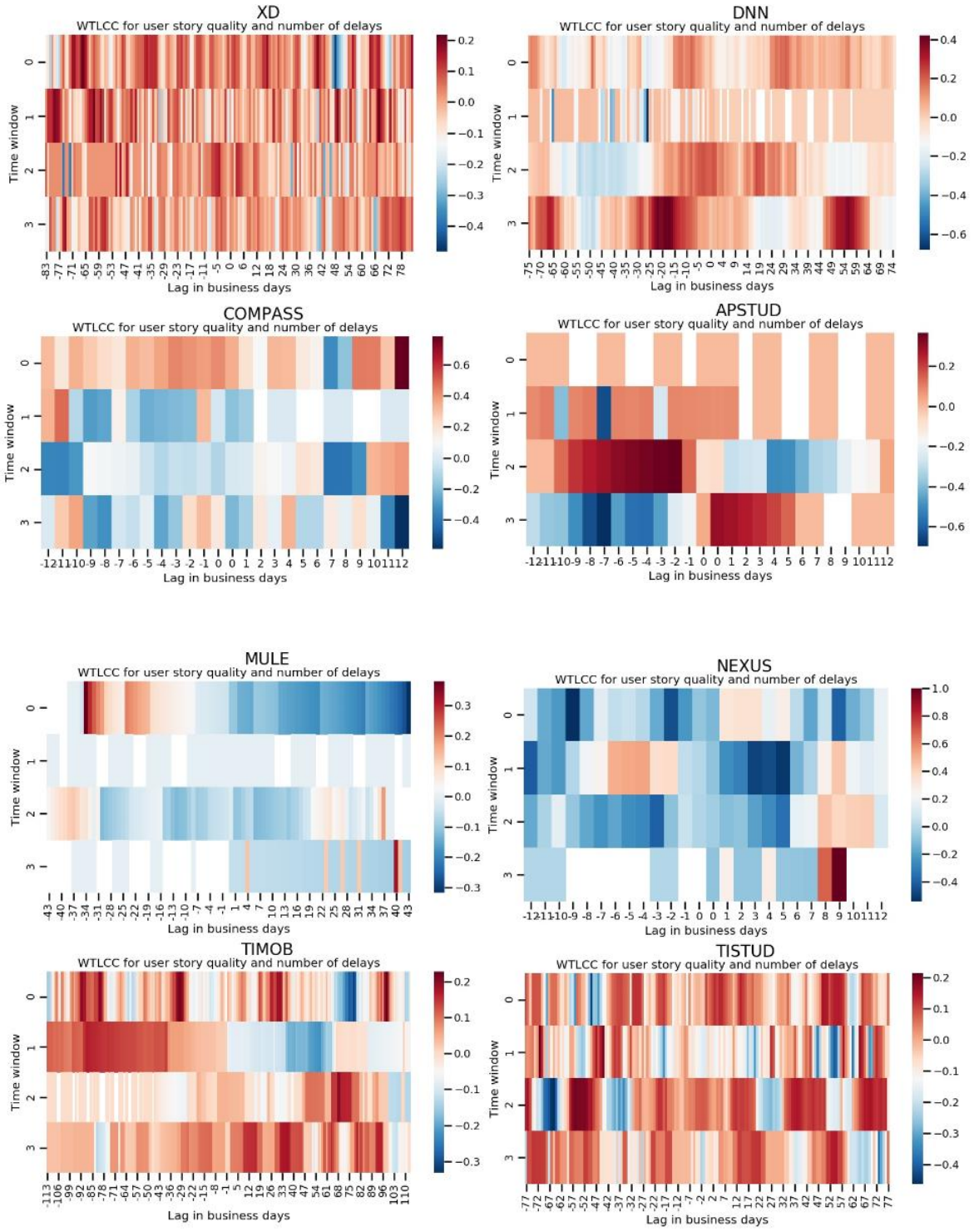
VII. WTLCC plots for User Story quality and rework cases in setting 1



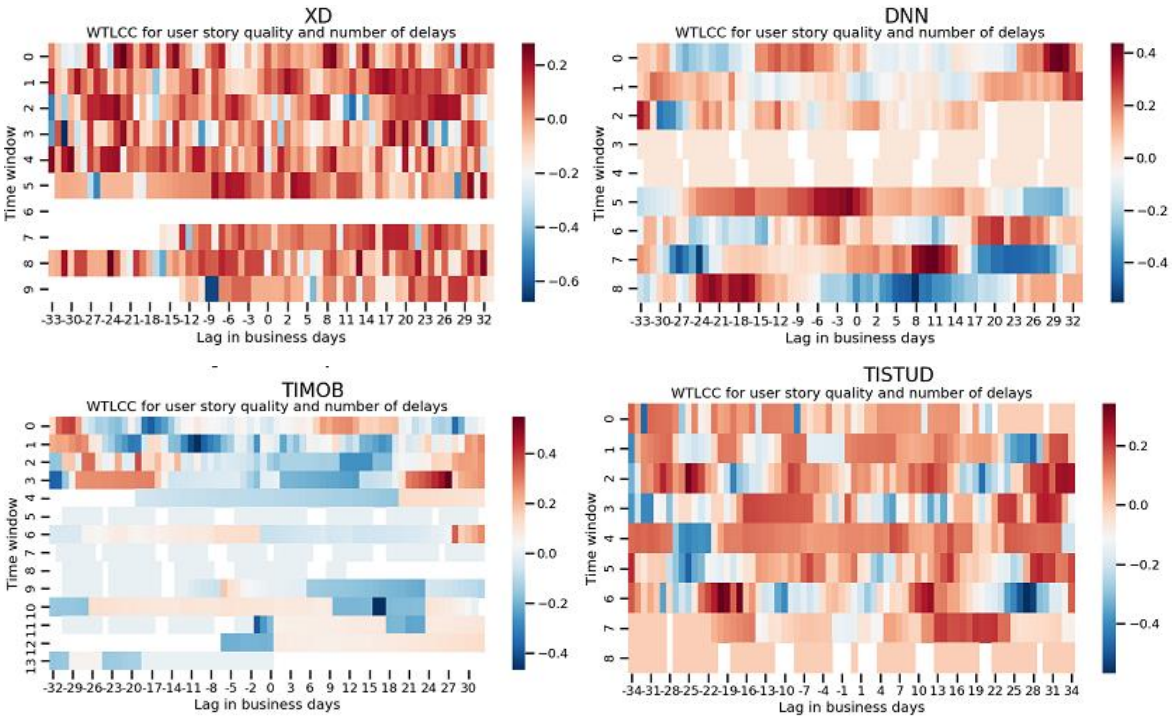
VIII. WTLCC plots for User Story quality and rework cases in setting 2



IX. WTLCC plots for User Story quality and delayed cases in setting 1



X. WTLCC plots for User Story quality and delayed cases in setting 2



XI. License

Non-exclusive licence to reproduce thesis and make thesis public

I, Tanel Tõemets,

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,

   Analysing the Quality of User Stories in Open Source Projects,

   supervised by **Ezequiel Scott.**

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons 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.

Tanel Tõemets

13/05/2020