An Empirical Study of User Story Quality and its Impact on Open Source Project Performance

Ezequiel Scott, Tanel Tõemets, and Dietmar Pfahl

Institute of Computer Science, University of Tartu, Narva mnt 18, 51009 Tartu, Estonia,
ezequiel.scott@ut.ee, tanel.toemets@gmail.com, dietmar.pfahl@ut.ee

Abstract. When software development teams apply Agile Software Development practices, they commonly express their requirements as User Stories. We aim to study the quality of User Stories and its evolution over time. Firstly, we develop a method to automatically monitor the quality of User Stories. Secondly, we investigate the relationship between User Story quality and project performance measures such as the number of reported bugs and the occurrence of rework and delays. We measure User Story quality with the help of a recently published quality framework and tool, Automatic Quality User Story Artisan (AQUSA). For our empirical work, we use six agile open source software projects. We apply time series analysis and use the Windowed Time Lagged Cross Correlation (WTLCC) method. Our results indicate that automatic User Story quality monitoring is feasible and may result in various distinct dynamic evolution patterns. In addition, we found the following relationship patterns between User Story quality and the software development aspects. A decrease/increase in User Story quality scores is associated with (i) a decrease/increase of the number of bugs after 1-13 weeks in short-medium projects, and 12 weeks in longer ones, (ii) an increase in rework frequency after 18-28, 8-15, and 1-3 weeks for long, medium, and short projects, respectively, and (iii) an increase in delayed issues after 7-20, 8-11, and 1-3 weeks for long, medium, and short duration projects.

Key words: User Story, Agile Software Development, Quality Assurance, Time Series Analysis, AQUSA, QUS, WTLCC

1 Introduction

Correctly defining and understanding what a software system is supposed to do is vital to any software project's success. Poor quality of software requirements has a severe effect on software projects success. 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. Low-quality requirements often cause the projects to exceed deadlines, increase the amount of rework and product defects [1]. In addition, ensuring high-quality requirements can be challenging as it is difficult to track and measure automatically [2].

To minimize the risk of communication errors as a potential threat to project success, requirements may be described using structured natural language (e.g., use case descriptions) or formal specifications. The downside of these methods is their high definition and maintenance cost.

Agile Software Development (ASD) methods recommend writing requirements in the form of User Stories as a possible solution that helps to save time by avoiding excessively detailed requirement documentation and to avoid that the focus is taken away from the actual software development task [3].

User Stories describe requirements as short texts with a strict structure consisting of three elements: (i) a desired function or property of the software, (ii) the stakeholder who requires this function or property, (iii) and the benefit of having this function or property. The most widely known template for writing User Stories was popularized by Mike Cohn [3] and proposes the following syntax: "As a <role>, I want <goal>, [so that <benefit>]".

The high interest in research on ASD [4] as well as the ever growing popularity of ASD as the development method of choice among software developers has been confirmed in several studies [5, 6]. Since User Stories were introduced, they have become a popular method for capturing requirements in ASD projects [5, 7, 8], because software developers find them effective and useful [9].

While User Stories have a well-defined structure, they are still written using natural language and, thus, their quality may vary. The consideration of quality criteria, such as the INVEST criteria (independent, negotiable, valuable, estimable, small, testable) seems to have a positive effect on software developers' attitudes towards User Stories, as they believe that using User Stories increases productivity and quality [10]. However, there exists little empirical evidence confirming the existence of such an effect in practice [9].

Our study aims to shed light on two research questions. Firstly, we analyze the evolution of User Story quality over time for the purpose of monitoring. Secondly, we investigate the relationships between User Story quality and characteristics of the development process, i.e., rework and delay, as well as the product, i.e., software quality. We conduct an empirical study on the data collected from six open-source software projects. To assess the quality of User Stories we apply the Quality User Story (QUS) framework together with the Automatic Quality User Story Artisan (AQUSA) open source software tool [11]. We are not only interested in finding out whether there exists an effect of User Story quality on process and product properties. We also try to find out whether it is possible to predict with how much delay a change of User Story quality in one sprint has an effect on process and product properties in later sprints. Our results indicate that a decrease in User Story quality scores seems to be associated with an increase of the number of bugs after 6 and 12 weeks in short and large projects, respectively; an increase in rework frequency after 1 and 8 weeks in short and large projects, respectively; and an increased number of delayed issues after 1 and 10 days in short and large projects, respectively.

The rest of our paper is structured as follows. In Section 2, we present related work. In Section 3, we introduce the research questions and describe the research design of our study. In Section 4, we present the results. In Section 5, we discuss our results and limitations of the study. In Section 6, we conclude the paper.

2 Related Work

2.1 Quality of User Stories

Several approaches exist for measuring the quality of User Stories. Literature covering this field are quite recent which shows the growing interest of the topic and its relevance in the current software engineering research. As stated in a recent study by Lucassen et al. [9], proprietary guidelines or the application of the INVEST criteria [10] are the most popular approaches used for assessing User Stories. Buglione and Abran [12] point out that User Stories without enough level of detail or incomplete User Stories can be the cause of incorrect effort estimation. Therefore, they highlight the importance of applying INVEST guidelines in order to reduce estimation error caused by low-quality User Stories. While the INVEST characteristics seem to be comprehensive, they are difficult to measure. Therefore, Lucassen et al. [11] developed the QUS framework supported by the AQUSA software. Since we used these instruments in our study, more detail is provided when we describe our research design in Section 3.

Lai proposes the User Story Quality Measurement (USQM) model [13]. This model is more complex than the INVEST model and organizes User Story quality metrics in 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". Since there doesn't seem to exist tool support for automatic measurement of criteria in the USQM model, the model is difficult to apply.

To solve issues originating from the fact that User Stories are written using natural language, de Souza et al. [14] propose the use of domain ontologies. This is expected to improve the quality of User Stories as it removes ambiguity. This approach, however, has not yet been evaluated much in practice.

2.2 Empirical Studies on the Impact of Requirements Quality

Firesmith [1] described twelve requirement engineering problems and their causes based on practical experience gathered while working with numerous real-life projects. He states that an increase in the number of defects and delays can be caused by numerous sub-problems, which in turn might root in problems like poor requirement quality, requirements volatility, or an inadequate requirements engineering process.

Rodríguez-Pérez et al. [15] studied the causes of bugs based on project data. They propose a model that groups bugs into two categories, intrinsic and extrinsic bugs. Intrinsic bugs are introduced by changes in the source code. Extrinsic bugs are introduced by requirement changes or other issues not recorded in the source code management system. The authors also state that an important limitation of their model is that it does not cover bugs caused by faulty requirements in the first place. Therefore, this model is not used in our study.

Sedano et al. [16] conducted a participant-observation study about waste in software development. The authors observed eight projects and conducted interviews with project team members. As a result, they established an empirical waste taxonomy listing nine types of software waste. As a possible cause of rework waste the authors identified requirement problems, more precisely User Stories without concrete complete criteria or rejected User Stories.

Tamai et al. [17] examined 32 projects to explore the connection between requirements quality and project success. A key finding was that requirements specifications were poor in those projects where significant delays happened.

The creators of the QUS framework and AQUSA tool evaluated their framework in a multi-case study involving three companies [11]. The purpose of the study was to explore whether their framework and tool affects the work of software developers positively. Thirty practitioners used the QUS framework and AQUSA tool for two months. With the help of surveys and interviews, the authors collected data for the following metrics at the start and during the case study: User Story quality, perceived impact on work practice, amount of formal communication, rework, pre-release defects, post-release defects, and team productivity. The authors found that the quality of User Stories improved over time after introduction of the QUS framework but they were not able to find clear evidence about an effect on the other metrics. The authors conclude that more data should be collected and they encourage others to conduct similar studies to identify the effects of User Story quality on the software development process and its outcomes.

2.3 Time Series Analysis in Software Engineering

Time series methods have been used for different purposes in software engineering. For example, Jahanshahi et al. [18] conducted time series analysis with the ARIMA (Auto Regressive Integrated Moving Average) model to predict the number of anomalies in software. Kai et al. [19] describe the usage of time series analysis for identifying normal network traffic behaviour. In a paper by Herraiz et al. [20] time series analysis is used for forecasting the number of changes in Eclipse projects. A recent paper by Choras et al. [21] explores the possibilities for predicting software faults and software quality by using time series methods. In their study, the authors compared the ARIMA, Random walk, and Holt-Winters forecast models with ARIMA showing the best performance. The models were tested with data on sprint backlog size, number of tasks in progress, and number of delayed tasks. Choras et al. also state that automated quality analysis is important for managerial roles like team leaders and product owners as it helps them make informed decisions regarding project timing, quality prediction, and other important aspects. For the purpose of correlation analysis between time-series, the Windowed Time Lagged Cross Correlation (WTLCC) method [22] has been proposed. This is the method we use in our study.

2.4 Summary

In summary it can be said that poor quality of requirements quality can be related to various issues such as increased number of defects, and excessive rework and delay. As stated by Choras et al. [21], 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 their quality but these approaches have not yet been applied on larger data sets to forecast User Story quality for the purpose of monitoring. In addition, the relationship between User Stories quality the amount of quality problems, delays, and rework has not yet been studied extensively.

3 Study Design

An overview of our study design is shown in Figure 1. We start with acquiring data from several publicly available JIRA server instances. The data is collected from real-life open source ASD projects. In order to understand the data, a significant amount of data exploration, data pre-processing and data cleaning steps are applied to make the data ready for analysis. Once the dataset has been cleaned, the User Stories are selected and their quality measured with the help of the AQUSA tool. As a result, a list of defects related to the User Stories is obtained and the quality score calculated based on those defects. We also calculate the software development performance measures from the dataset, i.e., number of issues labeled as "bug", frequency of rework occurrence, and frequency of delay occurrence. The quality scores and performance measures are captured as time series and their correlations analyzed using Window Time Lag Cross Correlation (WTLCC).

3.1 Research Questions

Our study tackles two research questions, RQ1 and RQ2. RQ1 focuses on exploring how the quality of User Stories changes over time and whether this can be monitored automatically. RQ1 is formulated as follows:

RQ1: What dynamic patterns can be observed when applying automatic measurement of User Story quality for the purpose of monitoring?

To answer RQ1, we rely on the QUS framework along with the AQUSA tool. In order to express User Story quality with one number, we introduce a formula that maps the data collected by the AQUSA to a rational number in the

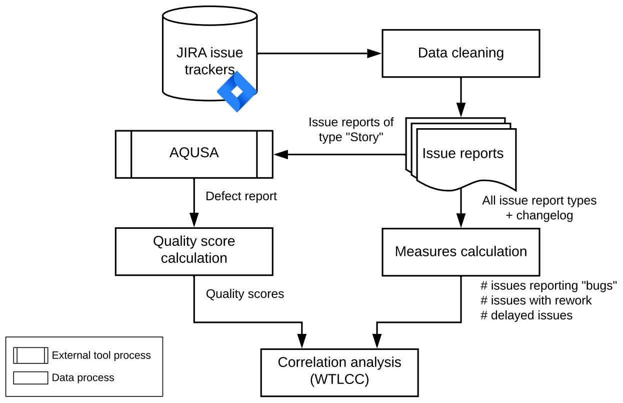

Fig. 1. Conceptual scheme of the study design.

interval [0, 1]. With the automatic monitoring of User Story quality in place, we study the relationship between User Story quality and three external quality characteristics of open-source ASD projects, i.e., number of bugs (issues labeled as "Bug"), rework done (re-opened issues of all types), and delays (issues not closed on due date). 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).

Regarding the relationship between User Story quality and number of bugs, we expect that poorly written User Stories affect the understanding of requirements negatively and, therefore, increase the number of bugs found in testing or by end users. Similarly, we expect that poorly written User Stories increase the amount of rework done and the number of delays.

To answer RQ2, time series data of User Story quality, number of bugs, rework and delays is collected. In the rare case of missing values we use imputing methods. For the correlation analysis, we apply the Windowed Time Lagged Cross Correlation (WTLCC) method [22].

3.2 Initial Dataset

We collected data from open-source projects using JIRA, a popular software development project management tool. Our initial dataset consisted of issue reports and their change logs from ten open-source ASD projects. Of them, eight projects had already been used in previous studies [23, 24, 25]. We identified two additional projects for our study and collected the relevant data.

Our initial dataset contained projects from different domains and with variance regarding the number of issues, developer experience, and development period. The original dataset had more than 20K issue reports.

3.3 Data Cleaning

We applied several steps for cleaning the collected dataset. First, we kept issue reports of type ”Story” for calculating the quality of user stories, issue reports of type ”Bug” for calculating the number of bugs, and issue reports of type ”Task” for calculating the occurrences of rework and delay. We only considered issue reports in complete status, indicated by the tags ”Closed”, ”Done”, ”Resolved”, and ”Complete”.

During the exploratory data analysis, we found that projects used different Jira fields to store the textual description of a User Story. Some projects used the field ”Summary” whereas others used the field ”Description”. After evaluating several alternatives, we opted for keeping both fields and evaluated their quality separately.

We applied several cleaning steps to remove the noise from the dataset and avoid misleading conclusions. In total, we applied 16 cleaning steps including the removal of duplicates, empty data, and the cleaning of textual descriptions by removing hyperlinks, code snippets, among others. The complete list of cleaning steps is given in Appendix 7. After the cleaning, we found that several projects only have few user stories (less than 30) such as the projects MESOS, SLICE, NEXUS, and MULE. We excluded these projects from the analysis since these few data points can not reveal reliable patterns in the data. The resulting dataset consits of six projects. Table 1 describes the projects (after cleaning) considered in the analysis. We consider all projects as completed as the data was collected more than one year after the latest observed activity (24 Aug 2018, project COMPASS). Two of the projects, i.e., APSTUD and COMPASS have a relatively short duration (313 and 338 days, respectively). Two other projects, i.e., TIMOB and TISTUD, have a relatively long duration (1625 and 1295 days, respectively). The projects DNN and XD are inbetween (869 and 726 days, respectively).

Table 1. Descriptive statistics of the projects in the dataset.

Project	Stories	Bugs	Rework	Delays	Quality				Development period	
					Mean	Std	Min	Max	From	To
APSTUD	151	329	160	87	0.90	0.04	0.67	0.92	08.06.2011	14.06.2012
COMPASS	98	427	13	319	0.96	0.05	0.83	1.00	20.09.2017	24.08.2018
DNN	250	1075	524	679	0.90	0.06	0.67	1.00	29.07.2013	15.12.2015
TIMOB	255	1052	399	160	0.88	0.04	0.75	0.92	22.11.2011	05.04.2016
TISTUD	525	1380	792	567	0.90	0.03	0.75	0.92	01.03.2011	21.07.2014
XD	2135	476	124	251	0.90	0.03	0.67	1.00	12.04.2013	30.11.2015
Total	3414	4739	2012	2063	–	–	–	–	–	–
Median	252.5	764	279.50	285	0.90	0.04	0.71	0.96	–	–
Mean	569	789.83	335.33	343.83	0.91	0.04	0.72	0.96	–	–

Several projects had inactive development periods at the start or end of the project. We manually inspected the dataset regarding the number of issue reports created and we kept only the issues created during the active development periods.

3.4 Measurement

To study the variation of the quality of User Stories over time, we define a measure that quantifies the quality. To study the correlation between user story quality and project performance, we measure project performance by counting the number of bugs reported, the number of occurrences of rework, and the number of occurrences of delays to study.

Quality of User Stories (Q): For each issue tagged as "Story" we calculated the quality of the text contained in the fields "Summary" or "Description" based on the defect report generated by the AQUSA Tool. The tool implements the quality model QUS proposed by [11] and is publicly available[1]. The tool analyzes the descriptions of the User Stories and uses a linguistic parser to detect violations. As a result, the AQUSA tool reports each violation as a defect along with its type, i.e., kind and subkind, and its severity. There are three possible severity values, i.e., high, medium, and minor, and 13 possible defects in total. Table 2 shows the different types of defects that AQUSA can report.

Table 2. Possible defects from AQUSA

Kind	Subkind	Severity
well_formed	no_means	high
well_formed	no_role	high
unique	identical	high
minimal	brackets	high
minimal	indicator_repetition	high
atomic	conjunctions	high
well_formed_content	means	medium
well_formed_content	role	medium
well_formed	no_ends	medium
uniform	uniform	medium
well_formed	no_ends_comma	minor
well_formed	no_means_comma	minor
minimal	punctuation	minor

We use a local instance of the AQUSA tool to process the user stories in our dataset. Then, the report generated by AQUSA is processed to quantify the quality of each user story and get a numeric value between 0 and 1. The quality of a user story Q is calculated as $Q = 1 - P$, where P is a penalty value calculated as a function of the number of defects and their severity. Equation 1 defines the formula to calculate the quality score of a given user story, where f_c is the percentage of defects of the user story in a category $c \in C$, $C = \{high, medium, minor\}$,

[1] AQUSA Tool repository – `https://github.com/gglucass/AQUSA`

and w'_c is the normalized weight for category c. To assign weights that correspond to the level of severity, we set $\mathbf{w}' = (0.5, 0.33, 0.16) = (\frac{3}{6}, \frac{2}{6}, \frac{1}{6})$ as a result of using $\mathbf{w} = (3, 2, 1)$ for high, medium, minor severity, respectively. The total number of defects possible in a severity category is 6 (high), 4 (medium), and 3 (minor), respectively.

$$Q = 1 - P = 1 - \sum_{c \in C} w'_c f_c \quad \text{with } w'_c = \frac{w_c}{\sum_{j=1}^{|C|} w_j} \quad \text{and } f_c = \frac{\#defects_c}{\#total\_defects_c} \qquad (1)$$

Number of bugs (*B*): Count of the issue reports of a project where the type is "Bug" and the status is complete (e.g. "Closed", "Done", "Resolved", or "Complete").

Rework (*R*): Count of the issue reports of a project that were re-opened. To calculate this, we analyze the log of changes of each issue. By default, JIRA records every change made to the issues along with a timestamp in a changelog. Therefore, if an issue was in status "Reopened", it is considered as rework.

Delays (*D*): Count of issue reports of a project that were completed after their originally planned date. To calculate this, we compare the issue resolution date with the end date of the sprint to which the issue was assigned to.

3.5 Data Analysis

We first create a time series representation for the quality of the user stories Q_p for each ASD project p in the dataset. We also create time series for bugs (B_p), issues with rework (R_p), and delayed issues (D_p). For indexing the time series, we use the issue creation date for user stories and bugs, the date when the change was made for rework, and the issue resolution date for delays. The data is re-sampled over 14 business days by using the mean and missing values are imputed by interpolation.

RQ1: To study the evolution of the quality of user stories over time, we present each time series Q_p in a separate plot and we describe the evolution of the quality by visual inspection.

RQ2: To study the relationship between the quality of the user stories Q_p and the variables of interest B_p, R_p, and D_p, we use Windowed Time Lag Cross-correlation (WTLCC). WTLCC is a method that allows us to study the association between two time series, $x(t)$ and $y(t)$ with $t = 1, 2, \ldots, T$, at different time lags (τ) and temporal changes in that association along the sample period [26, 27]. Thus, for a window size of $W(W < T)$, a pair of windows $\mathbf{Wx}$ and $\mathbf{Wy}$ can be selected for two data vectors x and y respectively, and the cross-correlation between the windows (WTLCC) at time t for different time lags (τ) is given by Eq. 2, where $\mu(\mathbf{Wx})$, $\mu(\mathbf{Wy})$, $\sigma(\mathbf{Wx})$ and $\sigma(\mathbf{Wy})$ are the means and standard deviations of the windows $\mathbf{Wx}$ and $\mathbf{Wy}$.

$$r_t(\mathbf{Wx}, \mathbf{Wy}, \tau) = \frac{1}{W - \tau} \sum_{i=1}^{W-\tau} \frac{(\mathbf{Wx}_i - \mu(\mathbf{Wx}))(\mathbf{Wy}_{i+\tau} - \mu(\mathbf{Wy})}{\sigma(\mathbf{Wx})\sigma(\mathbf{Wy})} \tag{2}$$

The calculation of WTLCC involves the selection of the window size. To the best of our knowledge, there is no method to determine the window size and, therefore, the window size must be determined based on theoretical considerations or data characteristics. The window size also defines the desired level of granularity during the analysis. We did a preliminary exploration of the results using different window sizes such as monthly and quarterly time periods. We finally opted for splitting the entire development period into four equally-sized windows. This way, the resulting windows are easy to explain. The first window (window 0) may correspond to the set up of the project, the next two windows (window 1 and 2) represent the development phase where most of the features are developed, and the last window (window 3) refers to the project finalization phase.

The results are depicted using heatmaps, a visualization technique that helps us with the inspection of the results at different time lags. We interpret the correlation values (r) by following Cohen's guidelines [28], where small, medium, and large effect sizes are $0.1 \leq r < 0.3$, $0.3 \leq r < 0.5$, and $0.5 \leq r$, respectively.

When analyzing the heatmaps, we are mainly interested in high positive or negative correlations with positive lags. In general, we expect a negative correlation between user story quality and the variables of interest since we assume that an increase (decrease) of user story quality results in a decrease (increase) of the project performance (i.e., bug/rework/delay count) after some lag time.

Positive correlations could be difficult to explain. Why would, for example, an increase in user story quality correspond to an increase in the number of bugs after some delay? A possible explanation could be that the work triggered by the content of the user story is complex or difficult by nature and, thus, more prone to bugs. Another reason could be a technical effect of the choice of window size.

It is also possible to find correlations with negative lags. For example, an increase (decrease) of the number of bugs yields an increase (decrease) of user story quality after some lag time (delayed positive correlation). This could indicate that teams have improved the quality of their user stories as a consequence of a previous increase in the number bugs. Or, in the reaction to less bugs, more time is spent on creating more user stories less carefully. Additional analyses would be needed to clarify this situations.

Finally, there is the possibility that, after some lag time, an increase (decrease) in bug count is followed by a decrease (increase) in user story quality (delayed negative correlation). This could be interpreted, for example, as a situation where, due to an increasing number of bugs, more time has to be spent on bug fixing and, thus, less time is available for writing proper user stories. Conversely, less bugs (and therefore less rework effort) might give more time for thinking about the requirements resulting in better user story quality.

4 Results

4.1 Study Population

As presented in Table 1, our cleaned dataset contains six projects with 3414 user stories, 4739 bug reports, 2012 rework cases, and 2063 delays. The project COMPASS has the smallest number of user stories and rework whereas APSTUD has the smallest number of delays and bug reports. The project XD has the largest number of user stories. Overall, the median number of user stories, bugs, rework, and delays is 252, 764, 279, and 285, respectively.

Table 1 also shows the descriptive statistics of the quality of the user stories. Overall, TIMOB has the lowest quality values whereas COMPASS has the highest. The projects XD, TISTUD, APSTUD, and DNN also have good quality values as their mean quality value is 0.9. The projects have a low standard deviation value regarding their quality values (mean std = 0.04).

4.2 User Stories Quality Monitoring and Evolution Patterns

Figure 2 shows the evolution of the mean quality of user stories over time. These graphs are useful to show how the quality of user stories can be used for monitoring purposes. A quantitative measure of the quality of the user stories of each project can be calculated by applying Equation 1 to the defect report created by AQUSA tool. Although the quality values remain almost stable due to the low standard deviation (see Table 1), it can be seen that the overall quality values vary over time exhibiting different patterns.

Figure 2 shows that project XD is rather stable since it has low variance. On the other hand, projects such as DNN and COMPASS exhibit an erratic behavior. Moreover, both projects show a trend of decreasing user story quality over time as it is shown by the regression line. The remaining projects indicate a slight increase of user story quality over time as their regression lines have a positive slope.

4.3 User Story Quality and Project Performance

Bug Count Figure 3 shows the results of applying WTLCC analysis to the six projects. The heatmaps associate the quality of user stories with the number of bugs. In each heatmap, the values on the y-axis represent the labels of the four windows used in the analysis. The x-axis shows the lag in business days that is applied before matching user story quality with the number of bugs. The correlation values are represented by the color scale. The title shows the name of the project along with the number of business days analyzed (n) on each case since the correlation analysis requires that the series occur simultaneously and in similar lengths.

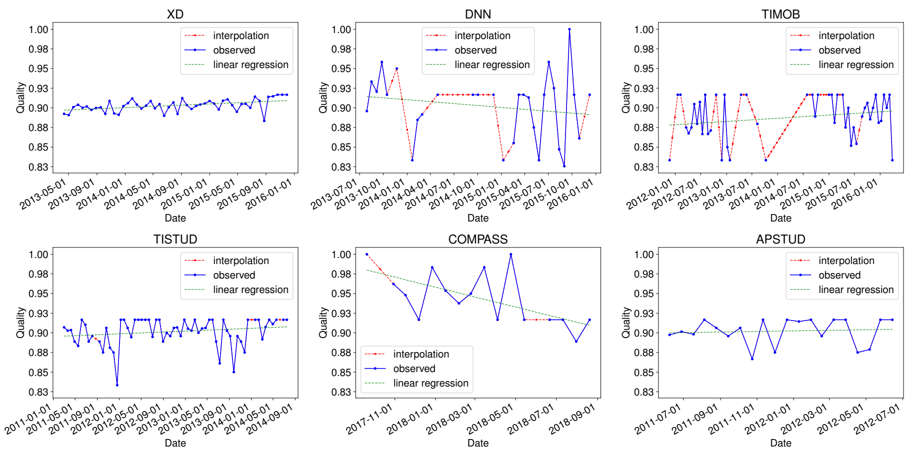

Fig. 2. Evolution of quality of user stories over time. A linear interpolation method was used to impute missing data points (red color).

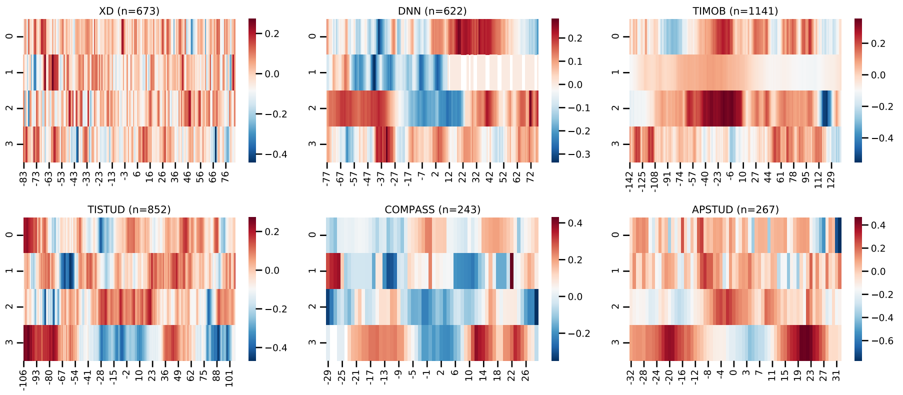

Fig. 3. Heatmaps representing the WTLCC results to compare the quality of user stories with the number of bugs.

Negative correlations with positive lags. The highest negative correlation values with positives lags are in the range $[-0.77, -0.26]$. The highest values are given by APSTUD ($r = -0.77, window = 0, lag = 32$), and TIMOB ($r = -0.55, window = 2, lag = 122$) whereas the remaining projects have correlations with medium effect XD ($r = -0.44, window = 3, lag = 68$), TISTUD ($r = -0.46, window = 3, lag = 89$). The interpretation is that the positive trend in user story quality pays off after 68 (XD), 89 (TISTUD), 122 (TIMOB) business days in the form of a decrease in bug count.

Negative correlations with negative lags. Negative high correlations with negative lags are present in the following projects: XD ($r = -0.33, window =$

$1, lag = -75$), DNN ($r = -0.34, window = 1, lag = -43$), TIMOB ($r = -0.30, window = 0, lag = -85$), TISTUD ($r = -0.47, window = 1, lag = -60$), COMPASS ($r = -0.30, window = 1, lag = -13$), APSTUD ($r = -0.31, window = 2, lag = -18$). A negative correlation with negative lag could indicate that an increase in the number of bugs creates more rework and, thus, leaves less time for conducting proper requirements engineering, which decreases the quality of user stories.

Positive correlations with negative lags. When looking at high positive correlations with negative lags, the results shows correlations in the range $[0.15, 0.27]$: XD ($r = 0.22, window = 3, lag = -76$), TISTUD ($r = 0.24, window = 0, lag = -104$), DNN ($r = 0.15, window = 1, lag = -64$), COMPASS ($r = 0.24, window = 0, lag = -1$), and APSTUD ($r = 0.27, window = 0, lag = -11$). TIMOB does not show a relevant correlation ($r = 0.09, window = 1, lag = -31$). This can be interpreted as follows: an increase in bug count results in an increase in user story quality with a lag of 1 to 104 business days. Possibly, the increase of user story quality was triggered as an attempt to stop a further increase in bug count.

Positive correlations with positive lags. Positive high correlations with positive are present in the following projects: XD ($r = 0.21, window = 2, lag = 47$), DNN ($r = 0.27, window = 0, lag = 19$), TISTUD ($r = 0.21, window = 2, lag = 21$), COMPASS ($r = 0.43, window = 1, lag = 22$), and APSTUD ($r = 0.47, window = 3, lag = 22$). This is difficult to interpret as it seems to suggest that an increase in user story quality yields an increase of bug count after 19 to 47 business days. We can speculate that other factors, e.g., increasing complexity of the system under development, are responsible for the negative effect on bug count. Only additional information could shed light on this.

Rework Count Figure 4 shows the results of the WTLCC analysis. In the following, we present the relevant correlation values. Thse results can be interpreted as we presented in full detail for the correlation between user story quality and bug count.

Negative correlations with negative lags. We found correlations in the range $[-0.69, -0.22]$ for the following projects. XD ($r = -0.33, window = 1, lag = -74$), DNN ($r = -0.54, window = 1, lag = -62$), TIMOB ($r = -0.22, window = 3, lag = -141$), TISTUD ($r = -0.40, window = 2, lag = -98$), COMPASS ($r = -0.37, window = 2, lag = -13$), and APSTUD ($r = -0.69, window = 2, lag = -13$).

Negative correlations with positive lags. These type of correlations are present in 4 out of 6 projects, in the range $[-0.40, -0.51]$: XD ($r = -0.50, window = 3, lag = 41$), TIMOB ($r = -0.51, window = 2, lag = 134$), TISTUD ($r = -0.42, window = 3, lag = 96$), COMPASS ($r = -0.40, window = 3, lag = 5$).

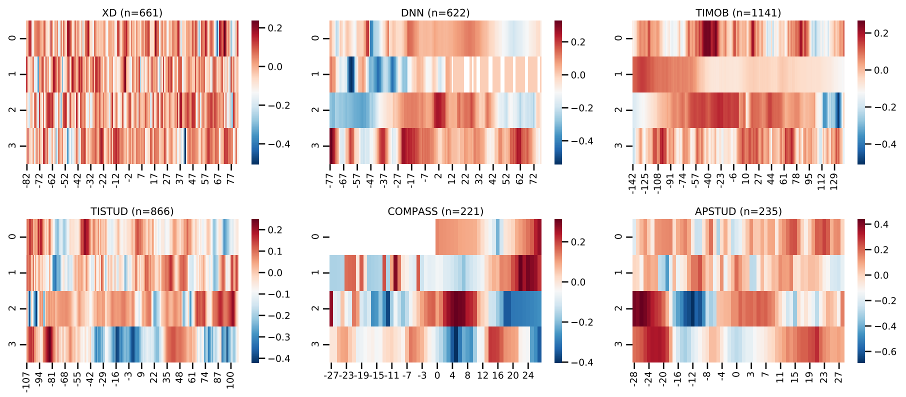

Fig. 4. Heatmaps showing the WTLCC results to compare the quality of user stories with the number of issues with rework done.

Positive correlations with negative lags. We found correlations in the range $[0.22, 0.44]$ for the following projects: XD ($r = 0.22, window = 1, lag = -6$), DNN ($r = 0.33, window = 3, lag = -77$), TIMOB ($r = 0.27, window = 0, lag = -43$), TISTUD ($r = 0.25, window = 3, lag = -85$), and APSTUD ($r = 0.44, window = 2, lag = -27$)

Positive correlations with positive lags. We found correlations in the range $[0.22, 0.31]$ for the following projects: XD ($r = 0.24, window = 0, lag = 72$), DNN ($r = 0.25, window = 2, lag = 2$), TISTUD ($r = 0.22, window = 2, lag = 102$), COMPASS ($r = 0.31, window = 2, lag = 5$), and APSTUD ($r = 0.26, window = 0, lag = 23$).

Delay Count Figure 5 shows the results of the WTLCC analysis regarding delay count. We present the relevant correlation values. These results can be interpreted as we presented in full detail for the correlation between user story quality and bug count.

Negative correlations with negative lags. We found correlations in the range $[-0.73, -0.16]$ for the following projects. XD ($r = -0.38, window = 2, lag = -76$), DNN ($r = -0.73, window = 1, lag = -28$), TIMOB ($r = -0.16, window = 3, lag = -83$), TISTUD ($r = -0.47, window = 0, lag = -50$), COMPASS ($r = -0.45, window = 3, lag = -10$), and APSTUD ($r = -0.69, window = 3, lag = -6$)

Negative correlations with positive lags. These type of correlations are present in 5 out of 6 projects, in the range $[-0.83, -0.37]$: XD ($r = -0.41, window = 0, lag = 48$), TIMOB ($r = -0.37, window = 2, lag = 100$), TISTUD ($r = -0.49, window = 1, lag = 53$), COMPASS ($r = -0.83, window = 2, lag = 8$), and APSTUD ($r = -0.47, window = 2, lag = 2$)

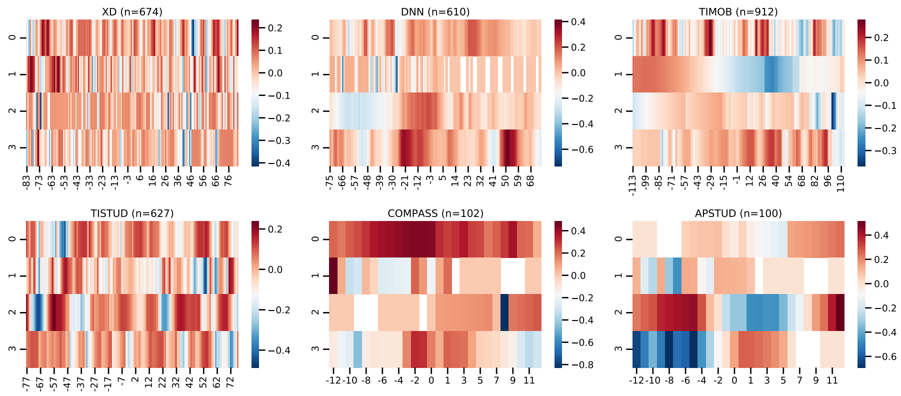

Fig. 5. Heatmaps showing the WTLCC results to compare the quality of user stories with the number of issues with delays.

Positive correlations with negative lags. We found relevant correlations in the range $[0.14, 0.24]$ for the following projects: XD ($r = 0.24, window = 1, lag = -59$), DNN ($r = 0.22, window = 2, lag = -6$), TIMOB ($r = 0.28, window = 0, lag = -31$), TISTUD ($r = 0.24, window = 2, lag = -58$), COMPASS ($r = 0.49, window = 1, lag = -13$), APSTUD ($r = 0.14, window = 1, lag = -7$).

Positive correlations with positive lags. We found correlations in the range $[0.19, 0.51]$ for the following projects: DNN ($r = 0.41, window = 3, lag = 51$), TIMOB ($r = 0.20, window = 3, lag = 94$), TISTUD ($r = 0.19, window = 1, lag = 71$), COMPASS ($r = 0.23, window = 2, lag = 12$), and APSTUD ($r = 0.51, window = 2, lag = 12$). XD has a correlation close to zero ($r = 0.09, window = 2, lag = 65$).

5 Discussion

Regarding the first research question, we observed that the projects exhibit different behaviors in terms of their quality of user stories over time. For example, project XD shows an upward trend in the change of quality. This indicates that the quality was increasing rather than decreasing. On the other hand, COMPASS showed an opposite behavior, where the quality of the user stories decreased as a trend.

The second research question asked about the relationship between User Story quality and the project performance, which is measured by the number of bugs, rework done, and delays. The analysis shows that the projects exhibit an inverse relationship between the quality of user stories and the studied project performance variables. If the quality of user stories increases (decreases), the number of bugs decreases (increase).

Interestingly, our results indicate that the events propagate from one variable to the other at different times (lags), and the lags seem to depend on the whole duration of the project. In short duration projects, the lags where smaller than in long projects. This can be a consequence of the amount of data to analyze but, surprisingly, strong correlations were found even in shorter projects where there are considerable less data points.

Regarding bugs, the effect can take 17-30 weeks to propagate from one variable (user story quality) to the other (bug count) in case of long duration projects whereas it can take 1-13 weeks in the case of medium and short duration ones. The number of delays can be inversely affected by the user story quality after 7-20 weeks for long projects, 8-11 weeks for medium projects, and 1-3 weeks for short ones. The occurrence of rework can also be inversely affected by the user story quality after 18-28, 8-15, and 1-3 weeks for long, medium, and short projects, respectively. Table 3 summarizes the main findings.

Table 3. Summary of results. The cells show the minimum and maximum lags expressed in business weeks (5 business days).

		Long duration		Medium duration		Short duration	
Variable	r	$lag < 0$	$lag > 0$	$lag < 0$	$lag > 0$	$lag < 0$	$lag > 0$
Bugs	$r < 0$	[-17.0, -5.8]	[17.8, 27.8]	[-16.0, -8.0]	[2.2, 13.6]	[-5.6, -2.6]	[0.8, 6.4]
	$r > 0$	[-22.4, -1.0]	[4.2, 9.4]	[-15.2, -1.2]	[3.8, 14.4]	[-2.2, -0.2]	[2.4, 4.4]
Delays	$r < 0$	[-16.6, -10.0]	[7.4, 20.0]	[-15.2, -5.6]	[8.0, 11.6]	[-2.2, -1.2]	[0.4, 2.4]
	$r > 0$	[-22.8, -3.6]	[10.8, 18.8]	[-15.2, -1.2]	[4.8, 13.0]	[-2.6, 0.0]	[2.4, 2.4]
Rework	$r < 0$	[-28.2, -15.8]	[18.6, 28.0]	[-14.8, -9.6]	[8.2, 15.6]	[-4.0, -1.2]	[1.0, 3.2]
	$r > 0$	[-26.0, -5.4]	[20.2, 20.4]	[-15.4, -1.2]	[0.4, 14.4]	[-5.4, -2.2]	[1.0, 5.4]

6 Limitations

The current findings are subject to several limitations that must be considered. It is worth noting that our data-driven approach does not support causal inference and it is mainly based on the discovering of patterns and correlation analysis. Controlled experiments are required to gain insights about causality. Furthermore, the data analysis approach required manual interpretation of visualized results which could have introduced errors. A more systematic approach to interpret the results could improve the accuracy and reliability of the results in further studies. More specifically, the selection of the window size in the WTLCC analyses has a strong influence on the observed results. In our case the window size varied in relation to the overall project duration. Long projects have wide windows and short projects have short windows. The results might change if, for example, uniform window sizes across all analyzed projects are chosen.

Regarding the generalization of the results, they are limited to the data sample. A larger data sample could produce different results, although it is difficult to find open-source projects that use ASD practices and track their process data using publicly available issue trackers. We mitigated this issue by analyzing a heterogeneous set of projects with different characteristics.

Missing values are another threat to validity. The dataset required an extensive amount of data cleaning in order to remove the noise that could have led to misleading conclusions. The projects also show periods of inactivity and we do not know the reasons behind this. To mitigate the impact of these missing data points, we removed the periods of inactivity by manual inspection and we use a linear interpolation method for imputing the remaining missing data points.

Another limitation is introduced by the AQUSA tool. The current development state of the tool is not able to assess the semantics behind the user stories descriptions as it would require expert domain or using advanced artificial intelligence. The AQUSA tool is only able to detect defects related to syntactic and pragmatic aspects of user stories.

In summary, although it is possible to monitor the quality of user stories by using the proposed approach, the process itself is complicated since considerable amount of work has to be done regarding the pre-processing and data cleaning of the data. The visual inspection of the heatmaps can be also prone to errors. therefore, it can be said that while it is possible to analyze the quality of user stories, more convenient solutions should be developed in order to make the monitoring of user stories simple for development teams.

7 Conclusion

The correlation analysis showed several interesting relationships between the quality of user stories and the project performance measured by the number of bugs, rework, and delays. The results show an inverse relationship between the user story quality and the project performance. When the quality of user stories decreased (increased), the number of bugs increased (decreased) correspondingly. This effect propagates from one variable to another at different lag times, and the lags seem to be related to the whole duration of the project. In particular, long-duration projects exhibit longer propagation time than short-duration projects.

We believe our results shed light on the benefits of writing high-quality user stories when managing requirements in agile software development. In particular, we provide empirical evidence that supports one of the most popular agile practices and the general agile mentors' advice of writing good user stories. Furthermore, this paper integrates previous research into an approach that can be easily extended into a monitoring tool (e.g. a dashboard) that allow developers and stockholders to visualize the overall quality of the written requirements in an aggregated way and set quality standards during the software development.

Appendix

During the data cleaning phase, we applied the following steps:

1. Removal of empty rows.
2. Removal of special headings in the description of the user story (e.g., "h2. Back story")
3. Removal of hyperlinks to web sites.
4. Removal of mentions to files with extensions such as ".jar".
5. Removal of code examples.
6. Removal of different types of curly brackets combinations.
7. Removal of paths to files.
8. Removal of word whose length is longer than 19 characters. According to [29], words with more than 19 characters are very rare in English (less than 0.1%). In our case, this usually happens when the bod y of a user story describe part of the program code. For example, the string "TriggerSourceOptions-Metadata"
9. Removal of consecutive exclamation marks and the text between them. This notation is commonly used for adding images (e.g., "!GettingStarted.png!")
10. Removal of square brackets and everything between them.
11. Removal of non-ASCII characters.
12. Removal of special characters such as "¡", "¿", "_", and "$".
13. Removal of different kinds of whitespaces (e.g., tabs, "&nbsp" etc) and replacing them with a single whitespace.
14. Removal of duplicated User Stories.
15. Removal of upper outliers (abnormally long User Stories). Upper outliers are removed based on the description length using Tukey's fences.
16. Removal of lower outliers (User Stories with less than 3 words). For example, some User Stories consisted of only the description "See: http://...".

Acknowledgment

This work was supported by the Estonian Center of Excellence in ICT research (EXCITE), ERF project TK148 IT, and by the team grant PRG 887 of the Estonian Research Council.

References

1. Firesmith, D.: Common requirements problems, their negative consequences, and the industry best practices to help solve them. Journal of Object Technology **6**(1) (2007) 17–33
2. Wohlin, C., et al.: Engineering and managing software requirements. Springer Science & Business Media (2005)
3. Cohn, M.: User stories applied: For agile software development. Addison-Wesley Professional (2004)

4. Dingsøyr, T., Nerur, S., Balijepally, V., Moe, N.B.: A decade of agile methodologies. J. Syst. Softw. **85**(6) (2012) 1213–1221
5. Kassab, M.: The changing landscape of requirements engineering practices over the past decade. In: 2015 IEEE Fifth International Workshop on Empirical Requirements Engineering (EmpiRE), IEEE (2015) 1–8
6. VersionOne, C.: 13th annual state of agile report (2018)
7. Wang, X., Zhao, L., Wang, Y., Sun, J.: The role of requirements engineering practices in agile development: an empirical study. In: Requirements Engineering. Springer (2014) 195–209
8. Kassab, M.: An empirical study on the requirements engineering practices for agile software development. In: 2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications, IEEE (2014) 254–261
9. Lucassen, G., Dalpiaz, F., van der Werf, J.M.E., Brinkkemper, S.: The use and effectiveness of user stories in practice. In: Int. working conference on requirements engineering: Foundation for software quality, Springer (2016) 205–222
10. Wake, B.: Invest in good stories, and smart tasks
11. Lucassen, G., Dalpiaz, F., van der Werf, J.M.E., Brinkkemper, S.: Improving agile requirements: the quality user story framework and tool. Requirements Engineering **21**(3) (2016) 383–403
12. Buglione, L., Abran, A.: 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) 49–53
13. Lai, S.T.: A user story quality measurement model for reducing agile software development risk. International Journal of Software Engineering and Applications **8** (2017) 75–86
14. de Souza, P.L., do Prado, A.F., de Souza, W.L., dos Santos Forghieri Pereira, S.M., Pires, L.F.: Improving agile software development with domain ontologies. In Latifi, S., ed.: Information Technology - New Generations, Cham, Springer International Publishing (2018) 267–274
15. Rodríguez-Pérez, G., Robles, G., Serebrenik, A., Zaidman, A., Germán, D.M., Gonzalez-Barahona, J.M.: How bugs are born: a model to identify how bugs are introduced in software components. Empir Software Eng **25** (2020) 1294–1340
16. Sedano, T., Ralph, P., Péraire, C.: Software development waste. In: Proceedings of the 39th International Conference on Software Engineering. ICSE '17, IEEE Press (2017) 130–140
17. Tamai, T., Kamata, M.I.: Impact of requirements quality on project success or failure. In Lyytinen, K., Loucopoulos, P., Mylopoulos, J., Robinson, B., eds.: Design Requirements Engineering: A Ten-Year Perspective, Berlin, Heidelberg, Springer Berlin Heidelberg (2009) 258–275
18. Jahanshahi, H., Cevik, M., Başar, A.: Predicting the number of reported bugs in a software repository. In: Advances in Artificial Intelligence, Cham, Springer International Publishing (2020) 309–320
19. Kai, H., Zhengwei, Q., Bo, L.: Network anomaly detection based on statistical approach and time series analysis. In: 2009 International Conference on Advanced Information Networking and Applications Workshops. (2009) 205–211
20. Herraiz, I., Gonzalez-Barahona, J.M., Robles, G.: 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) 32–32

21. Choraś, M., Kozik, R., Pawlicki, M., Hołubowicz, W., Franch, X.: Software development metrics prediction using time series methods. In Saeed, K., Chaki, R., Janev, V., eds.: Computer Information Systems and Industrial Management, Cham, Springer International Publishing (2019) 311–323
22. Roume, C., Almurad, Z., Scotti, M., Ezzina, S., Blain, H., Delignières, D.: Windowed detrended cross-correlation analysis of synchronization processes. Physica A: Statistical Mechanics and its Applications **503** (2018) 1131–1150
23. Scott, E., Pfahl, D.: Using developers' features to estimate story points. In: Proceedings of the 2018 International Conference on Software and System Process. (2018) 106–110
24. Scott, E., Charkie, K.N., Pfahl, D.: Productivity, turnover, and team stability of agile software development teams in open-source projects. In: 2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), IEEE (2020)
25. Porru, S., Murgia, A., Demeyer, S., Marchesi, M., Tonelli, R.: Estimating story points from issue reports. In: Proc. of the 12th Int. Conf. on Predictive Models and Data Analytics in Soft. Eng. (2016) 1–10
26. Boker, S.M., Rotondo, J.L., Xu, M., King, K.: Windowed cross-correlation and peak picking for the analysis of variability in the association between behavioral time series. Psychological methods **7**(3) (2002) 338
27. Jammazi, R., Aloui, C.: Environment degradation, economic growth and energy consumption nexus: A wavelet-windowed cross correlation approach. Physica A: Statistical Mechanics and Its Applications **436** (2015) 110–125
28. Cohen, J.: A power primer. Psychological bulletin **112**(1) (1992) 155
29. Sigurd, B., Eeg-Olofsson, M., Van Weijer, J.: Word length, sentence length and frequency–zipf revisited. Studia Linguistica **58**(1) (2004) 37–52