2021 International Conference on Software Engineering & Computer Systems and 4th International Conference on Computational Science and Information Management (ICSECS-ICOCSIM) | 978-1-6654-1407-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICSECS52883.2021.00060

Enhancing High-Quality User Stories with AQUSA: An Overview Study of Data Cleaning Process

Siti Nur Fathin Najwa Binti Mustaffa
Faculty of Computing
Universiti Malaysia Pahang (UMP)
Pekan, Pahang
fatinnajwa22@gmail.com

Dr Jamaludin Bin Sallim
Faculty of Computing
Universiti Malaysia Pahang (UMP)
Pekan, Pahang
jamal@ump.edu.my

Dr Rozlina Binti Mohamed
Faculty of Computing
Universiti Malaysia Pahang (UMP)
Pekan, Pahang
rozlina@ump.edu.my

Agile software development has ended up profoundly prevalent over the final two decades. In conjunction with the increment of these methodology, amount of scientific research on this topic has also increased. This research concentrates on one component from Agile software development which is User Stories (US). In a recent paper, quality framework for User Stories was proposed together with a tool implementing which is still minimal finding to achieve high quality of user story document. It is closely related to the data requirement as part of the success factor in development project. The main goal is to analyze US requirement written and identification of potential errors in datasets that bring the effectiveness in forecasting the quality of User Stories for monitoring purpose. One of the steps require to identify the quality User Stories is by the route passing through data cleaning process. The research analysis considers performing data cleaning and pre-processing towards existing dataset of user story requirement from open-source agile software projects. Results of analyzing the User Stories will reveal the possibility step of data cleaning as initial step to extend the AQUSA tools and forecast the quality of User Stories. Thus, it can be related to the requirements quality based on the number of issues report in the dataset where highest number of issues report can be categorized as poor requirement statement.

Index Terms— User Story, Agile Software Development, Data Cleaning, Quality User Story

I. INTRODUCTION

The fundamental of good system is starting from clean requirement statement before the development started. However, it is struggled to define a complete set of requirements with minimal error in a project which often proves to be counterproductive, restrictive, and wasteful. This is due to the business environment might changes occasionally which bring the new opportunity of requirement changes. Relevant to the current practice, it indicates that the Requirement Engineering (RE) process is critically leverage the success of the system development life cycle.

Requirement increasingly expressed as user stories for practitioners in agile development [1]. User stories were the ultimate used requirements documentation method among requirements analyst in an agile environment [1]. Despite this popularity, the number of approaches for evaluating and improving user story quality is limited, particularly when it comes to identifying duplications amongst user stories [2]. Existing approaches to user story quality employ highly qualitative metrics and give generic guidelines for ensuring quality in agile [3].

In user story form, requirements are written in short text with format structure describes some functionality of the software, the stakeholder who requires it, and the benefit of having that functionality. Mike Cohn [3] popularized the most extensively used template for writing User Stories with proposes following syntax: *"As a (role), I want (goal), [so that (benefit)]"*.

Widespread interest and reality of agile research are highlighted by the consistently expanding sum of scientific publications tending to the Agile software development [4]. Since User Stories were presented, they have gotten to be a well-known strategy for capturing requirements in agile software projects [5, 6, 7]. A well-known study, VersionOne from 2019 [8] found that, 97% of respondents' organizations utilized agile development strategies. An prior study [4] found that the utilization of agile development strategies has nearly multiplied from 2008 to 2013 getting to be the foremost prevalent software development life cycle. With the increment of notoriety in agile development strategies the popularity of composing requirements within the form of User Stories has too developed [4]. Another study [9] about the use and effectiveness of User Stories states that most of the software professionals who reacted were positive around using User Stories and found them useful.

In spite of the fact that User Stories have a restricted structure, they are still composed utilizing common dialect and so their quality can shift. A Quality User Story (QUS) system [10] and Artisan Quality User Story (AQUSA) open-source software tool [10] are consequently assess the quality of User Stories. The utilize of quality rules, such as the INVEST criteria had appeared to have a momentous impact on respondents' states of mind towards User Stories[5]. Respondents who utilized INVEST quality rules believed that utilizing User Stories incorporates a much more positive impact on efficiency and quality than others. Although this may indicate that utilizing quality rules for User Stories, but there's a need of experimental prove supporting that claim to realize way better efficiency [10].

Apart from that, to define the quality user story, it is closely related to the set of data to define. Common information quality issue such inconsistencies incorporates conflicting information conventions among sources such as diverse truncations or equivalent words, data entry errors such as spelling mistakes, conflicting data format, missing, inadequate, obsolete, information duplication, insignificant object, or any other erroneous attributes values. Information that is fragmented or wrong is known as dirty information. These findings are strongly connected with the purpose of this analyze user story requirement written and identification of potential errors in datasets that bring the effectiveness in forecasting the quality of User Stories for monitoring purposes.

In particular, the theme of this research is to analyze the quality of the User Stories, based on the Quality User Story (QUS) framework, Quality User Story Artisan (AQUSA) tool [10] and online sources dataset significant [10] on data from real-life agile software projects. This intentionally to understand how the quality of User Stories change over time, find if it is possible to forecast the change of User Story quality

978-1-6654-1407-4/21/$31.00 ©2021 IEEE
DOI 10.1109/ICSECS52883.2021.00060

Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2025 at 08:35:25 UTC from IEEE Xplore. Restrictions apply.

over time. The preliminary introduction of quality user story and existing tools will be described in Section II (Related Works). The relevant methodology on the data cleaning analyzing will be discussed in Section III (Methodology).

II. RELATED WORKS

Various approached to requirement documentation have been reported in the literature, each of which has its own characteristics and deals with certain challenges.

A. Determining the Causing Factors of US statement

Requirement and user story is inter-correlated as it is presented from the standpoint of an end-user purpose. User stories are also known as Epics, Themes, or Features, but almost require same format. The format has becoming increasingly popular among Agile practitioners. It is a clearly stated criterion with a defined aim. Within the user story description, the business value of each demand is considered. This is where the user story is including role perspective who will use or be impacted by the solution, defining the business requirement, and clarifying the actual reason for the requirement in high level.

It promotes discussion and learning, which aids project team collaboration. It is not always simple, however there are several issues that have been identified as an impact on the effectiveness of user stories. These issues concept is typically due to execution issues rather than structural issues. It will fail if the required specification unable to deliver in a holistic manner. One of the issues found in requirement written is too technical details. In most circumstances, user stories will include the main function of components as to achieve the value described in the story. Purely technical stories may or may not provide a benefit that can be prioritized by a stakeholder or Product Owner.

Next, the expressed criterion is vague and not concrete. One of the reasons for adopting Agile and user stories is to avoid rely on a requirement document. To various people, words have varied meanings. As previously stated, user stories are all about the dialogue. We will not be able to test and review code until we turn our memories and notes into a design. Difficulty to engage a generic persona is one of the common issues. A group of users is represented by a persona. When you use overly generic user personas in user stories, it is difficult to verify you are delivering what is really needed. Similarly, when teams bypass the difficult job of identifying personas, they might have different perception on the actual purpose of that persona.

Therefore, a user stories may not be recognized as a piece of functionality that needs to be developed by the user. Prioritize tasks are based on the user story function which each of that point will give an impact. A comprehensive result of quality user story can be producing if the user is able to define the function and benefit. Finally, there are stories that have no acceptance requirements. Acceptance criteria are an important aspect of the user narrative definition. This purposely to meet the requirements and can be tested before the story is declared as complete. Not specifying acceptance criteria is a complete lack of discipline, and it may be the most serious of the issues, since it can lead to a series of errors such as misinterpret the feature purpose, delay the development activities and incorrect project estimation.

B. Structure of User Stories

Assessing the quality of User Stories using the QUS framework, the definition of each part of the User Story is provided. User Story incorporate four parts: Role, Means, Ends, and Format [10]. Role is a stakeholder who needs functionality stated in the User Story. Example of Role: *"As a Stock Manager"*. Role can also be a persona or main actor which is an abstract named character representing some group of users [10]. Example of persona in User Story *"As a Bukhari"* Means captures the aim of the software functionality, an action that software needs to perform or object to which the action is made [10]. Example of mean: *"I want to hover the stacked bar graph"*.

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 [10] 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 view the overall physical inventory volume of diesel product"*. Format means that the User Story must follow some certain User Story template in well-formed [10].

C. The Quality User Story framework and tool

INVEST [11] quality guidelines are the most popular approaches used for assessing User Stories. It is described the good user stories should be independent, negotiable, valuable, estimable, small, and testable. These characteristics are needful but difficult to measure. Agile Software Development has some management related weaknesses [12]. For instance, User Stories without incomplete requirement can brings to errors in effort estimation. Authors discussed further in the result section where they applied the INVEST quality guidelines in agile software projects. The objective of this paper is to highlight the importance of applying INVEST guidelines and to reduce errors in effort estimation that caused by the quality issues.

In another recent paper, research revises and defines a QUS framework [10,13] and AQUSA open-source software tool to automatically assess the quality of User Stories. QUS framework was originally proposed by the same authors in their previous paper [14]. 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 [10] The framework is presented in Figure 1.

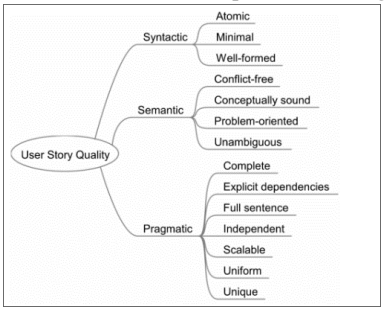

Figure 1: Quality User Story framework [14]

D. Definition of QUS framework 13 quality criteria

Table 1 presented 13 QUS quality criteria and a description of each criterion. QUS framework categorizes

Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2025 at 08:35:25 UTC from IEEE Xplore. Restrictions apply.

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 [10].

TABLE 1: QUS FRAMEWORK CRITERIA DEFINITIONS [10]

No	Criteria	Sub Criteria	Descriptions	Individual / Set
1.	Syntatic	Well-formed	Include at least a role and mean	Individual
		Atomic	Express exactly one feature	Individual
		Minimal	Contains nothing more than one role, mean and ends	Individual
2.	Semantic	Conceptually sound	Express a feature and rationale	Individual
		Problem Oriented	Specify the problem, not the solution	Individual
		Unambiguous	Avoid terms or abstractions that lead to multipe interpretations	Individual
		Conflict-free	Inconsistent with any other user story	Set
3.	Pragmatic	Full Sentence	Well-formed full sentence	Individual
		Estimable	Not denote a coarse-grained requirement that is difficult to plan and prioritize	Individual
		Unique	Unique, duplicated are avoided	Set
		Uniform	Employ the same template	Set
		Complete	Complete feature implemention 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.

E. The Automated Quality User Story Artisan (AQUSA) tool

The architecture of the AQUSA software tool that implements the previously described framework to assess the quality of User Stories [10]. 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 analyze the meaning of User Stories which authors found not possible to implement while keeping the recall close to 100% [10]. Therefore, AQUSA tool only considers following criteria of Well-formed, Atomic, Minimal, Independent (not fully implemented in AQUSA), Uniform and Unique from the QUS framework. This proposed research will extend the criteria (unambiguous) as purpose to achieve high quality user story. The User Stories are inserted and analyzed 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 [10].

III. METHODOLOGY

Most of the Agile project use open-source tools to store the requirement (user stories or feature request). In that case, the dataset from the open-source agile software project is review for the issue reports and change logs of 10 different projects. It was constructed by augmenting an existing dataset that has been used in previous research studies [15,16]. Jira extractor is a tool for connecting with Jira server and downloading issue reports with their changelogs. The dataset can be collected from open-source agile software projects as well as available online requirement in cloud where significant amount of data exploration is needed.

All datasets will incorporate different projects with diverse space, extend scenarios, number of issues, engineer encounter, and improvement period. The taking after list appears all the projects considered within the analysis. Each project within the list contains a brief description, the development period, and its project acronym. These acronyms are indicated behind each extend title in brackets and will be utilized in encourage analysis to allude for project. Each of investigation will be considering improvement period from year 2016 onwards.

The starting dataset comprised of issue reports of 10 projects. The collected information was extricated utilizing Jiraextractor2. Jira extractor could be a device for interfacing with Jira server and downloading issue reports with their changelogs. The code sample shows the usage of Jira extractor to download all the issue reports and their changelogs for the project “Spring Slices”. python jiraextractor.py -s https://jira.spring.io --project SLICE

```
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. Table 2.0 shows the list of 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 to refer to specific project.

TABLE 2.0: PROJECT NAME AND DESCRIPTIONS

No.	Project Name	Descriptions
1.	Spring XD3 (XD)	A system for data ingestion, real time analytics, batch processing, and data export. System is targeted for big data applications.
2.	DNN Platform 4 (DNN)	A content management system designed for .NET developers.
3.	MongoDB Compass5 (COMPASS)	A graphical user interface (GUI) for MongoDB database.

Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2025 at 08:35:25 UTC from IEEE Xplore. Restrictions apply.

No.	Project Name	Descriptions
4.	Aptana Studio 6 (APSTUD)	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)	A cluster management system
6.	Mule8 (MULE)	Enterprise service bus (ESB) and integration platform.
7.	Nexus by Sonatype 9 (NEXUS)	A repository manager for software artefacts.
8.	Titanium SDK10 (TIMOB)	A JavaScript based software development kit (SDK) and command line tool (CLI)
9.	Spring Slice12 (SLICE)	Part of Spring software development framework.
10.	Appcelerator Studio11 (TISTUD)	An Integrated Development Environment (IDE) for building cross-platform mobile applications.

Original dataset consisted of tables in a csv format containing issue reports of all kinds including, bugs, epics, stories, tasks, sub-tasks, and others defined category. The highest and lowest number of issue reports per project is analyzed respectively according to the number of issues report. Continuously, to performed pre-processing activities that require execution from the dataset purposely to provide input for data cleaning activities. The number of issue reports per project is visualized in Figure 2.

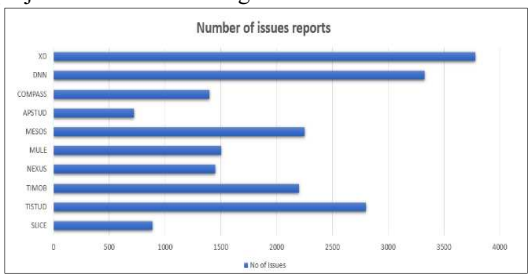

Figure 2: No of Issues Reports per project

As we can see from Figure 2, the datasets of projects under review were quite different in size. XD and DNN had the highest number of issues, 3782 and 3328, respectively. APSTUD and SLICE had the fewest issues, 721 and 885, respectively. The total number of issues under review was 20136. Understanding the data about all projects required a considerable amount of data exploration to reveal all the problems with data and make it possible to perform data pre-processing and data cleaning described in the following paragraphs.

The most extensive part of pre-processing related to User Stories. This part keeping only issues with issue type "Story." Columns as "fields.issuetype.name" categorize the story as bug, improvement, story, or technical task. Next, only issues with statuses "Closed," "Done," "Resolved," and "Complete" will be keep.

Next, columns containing User Stories were chosen for further cleaning. This activity turned out to be challenging as most of the projects form in inconsistent format. This led to a situation where some User Stories is divided into "fields.summary" column and others in "fields.description" column. For next further step, the several approaches to analyse the user story is drafted either concatenate or separated column of "fields.summary" and "fields.description" field to assess the quality with AQUSA.

Assessing the quality of stories concatenate will define either giving high- or low-quality scores for even correct User Stories because AQUSA was not able to identify User Stories correctly. For combine column of "fields.summary" or "fields.description" columns will give greater number of User Stories. The negative side of this method is that it caused a loss of considerable amount of data. Next, both "fields.summary" and "fields.description" columns is keep, and identifier will be 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 is set for "fields.summary" and "fields.description" columns, then merged into one column, followingly referred as "Story" column.

After assessing "Story" column only the story with better quality score will be keep for further analysis. Selecting the best approach included calculating quality scores and inspecting the results manually. Selected method made it possible to identify a greater number of User Stories and ensured more reliable results.

Based on the data consolidation, we observe that most of the user story set is included with unnecessary statement written which will bring to different perception and unclear on the actual requirement. Table 3.0 below shows the consolidate issues observe in the dataset.

TABLE 3: OVERALL ISSUES ITEM BASED ON THE DATA ANALYSIS:

Step.	Issues	Descriptions
1.	User Story include unwanted observation	Irrelevant statements such as including extra information such file extension, sample code reference, and unnecessary path or web link
2.	User Story contain duplicate statement	Use of word is duplicate in one written statement of user story set and duplicate user story set in requirement statement
3.	User stories contain structural errors such inconsistent capitalization	Occurs during measurement and data transfer. After replacing the structure error and inconsistent capitalization, the user story description turns cleaner: As example: 'composition' is the same as 'Composition', 'asphalt' should be 'Asphalt' and 'shake-shingle' should be 'Shake Shingle.
4.	User Story contain blank or missing data statement	Incomplete statement with blank space without ending node. Missing numeric data for example to perform any calculation formula
5.	Filter unwanted outliers	Outliers might cause problems with specific types of models; therefore, it is a good idea to get rid of them. It will help with the model's performance during the analysis.

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

Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2025 at 08:35:25 UTC from IEEE Xplore. Restrictions apply.

for AQUSA. Data cleaning activity will be improved several times during the analysis. General description of data cleaning steps for User Stories is presented in Table 4.

TABLE 4: DATA CLEANING STEPS FOR USER STORIES

No.	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 any reference linked 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.
9.	Removing words longer than 19 characters.
10.	Removing consecutive exclamation marks and everything between them. Such structures were used for adding images. For example: *"!KnoweledgrProp.png!"*
11.	Removing square brackets and everything in them.
12.	Removing non-ASCII characters to make AQUSA able to parse the cleaned input data.
13.	Removing other special characters to make AQUSA able to parse the input data. For example, characters double quote, number if sign(has), dollar sign "", "_", "$", "#" etc.
14.	Removing all different kinds of whitespaces (tabs, " " etc) and replacing them with single whitespace.
15.	Removing duplicate User Stories.
16.	Removing upper outliers (abnormally long User Stories).
17.	Removing lower outliers (User Stories with less than 3 words)

Next, table below is the sample data users story set after applying the data cleaning step rules.

TABLE 5: EXAMPLE OF REMOVING CURLY BRACKET IN USER STORY DESCRIPTIONS.

Key	User Story	Status
XD-3595	As a s-c-d developer, I would like to add test coverage for {{StreamController}}, so I can verify API contracts at build time.	Before Cleaning
	As a s-c-d developer, I would like to add test coverage for StreamController, so I can verify API contracts at build time.	After Cleaning
XD-3641	As a developer, I would like to review and refactor {{JobLaunchingTasklet}}, so I can improve performance characteristics.	Before Cleaning
	As a developer, I would like to review and refactor, so that I can improve performance characteristics.	After Cleaning
XD-3425	As a s-c-d developer, I would like to have {{module info}}, {{module list}}, {{module register}}, and {{module unregister}} commands, so I can interact with {{ModuleRegistry}}.	Before Cleaning
	As a s-c-d developer, I would like to have module info, list, register, and unregister commands, so I can interact with ModuleRegistry.	After Cleaning

A.High-Level Design

The study of research design based on the research steps shown in Figure 3. Continuously, several data pre-processing and data cleaning steps need to apply as to preparation data analysis. After the dataset is clean and pre-process, the User Stories input is select from the dataset and process with AQUSA tool. As a result, a list of defects related to the User Stories can be obtained. This list of defects can be exported for further analysis.

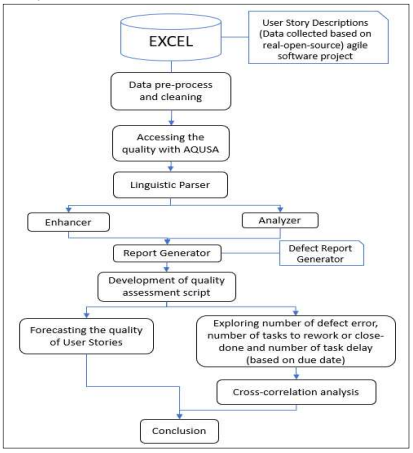

Figure 3: Proposed Research Design

A formula to quantify the list of defects generated by AQUSA was proposed to examine the list of defects generated by AQUSA. This formula allowed for the calculating and representation of the quality of User Stories as numerical values, as well as time series analysis.

Forecasting the quality of User Stories and analysis of number of bugs, rework, and delays can be performed simultaneously as these activities is not connected to each other. Analysis of the number of bugs, rework, and delays first required data exploration to understand the available data. Continuously, cross-correlation analysis will be 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 will obtain via validating through testing in controlled environment setting. 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.

B. Threats to validty

Current research only used 10 existing online project to fetch the dataset of user stories. The date range created for the dataset user stories is taken from 2016 onwards. Further analysis on the latest project date issues as well as additional open-source project can be analyzed to compare the quality requirement. However, the project is incoming in different type of issues such as bug, improvement, epic, story, and technical tasks to get the overall analysis on the discrepancy requirement written in set of user story. We also tested our

Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2025 at 08:35:25 UTC from IEEE Xplore. Restrictions apply.

framework against current literature in the areas of requirements quality and agile requirements.

IV. CONCLUSION

To review of AQUSA tools, it started with the bucket of dataset as main input to measure the quality itself. This research focus on the perspective of the initial step towards the dataset of user stories itself. It is where the data cleaning activity is performed to observe the number of issues report. Although the issues with the data may not be completely solved, reducing it to a minimum will have a significant effect on efficiency. Poor requirements quality is determined from the number of issues report which bring the highest number of issues report can be determine as poor requirement statement. Further work is continuing and studying after the data cleaning step stages. It is where clean user story set will extend with AQUSA output and quantified to forecast the quality of User Stories for monitoring purposes. In addition, the relationship between quantified User Story quality score and the number of issues, delays, and rework will be investigated to see if there is a link between User Story quality and other areas of software development.

V. ACKNOWLEDGMENT

This research work is supported by Universiti Malaysia Pahang Grant: PGRS2003101.

REFERENCES

[1] X. Wang, L. Zhao, Y. Wang, and J. Sun, "The Role of Requirements Engineering Practices in Agile Development: An Empirical Study," in Proc. of the Asia Pacific Requirements Engineering Symposium, ser. CCIS. Springer, 2014, vol. 432, pp. 195–209.

[2] R. Barbosa, A. E. A. Silva, and R. Moraes, ''Use of similarity measure to suggest the existence of duplicate user stories in the srum process,'' in Proc. 46th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. Workshop (DSN-W), Jun. 2016, pp. 2–5, doi: 10.1109/DSN-W.2016.27.

[3] M. Cohn, User Stories Applied: for Agile Software Development. Redwood City, CA, USA: Addison Wesley, 2004.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

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

[9] 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.

[10] 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.

[11] 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].

[12] 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.

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

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

[15] 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.

[16] 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.

Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2025 at 08:35:25 UTC from IEEE Xplore. Restrictions apply.