2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C)

Exploring the Dependency Relationships between Software Qualities

Celia Chen[*†], Michael Shoga[*], Barry Boehm[*]
[*]Center for Systems and Software Engineering, University of Southern California, Los Angeles, USA
Email:{qianqiac, mshoga, boehm}@usc.edu
[†]Occidental College, Los Angeles, CA
Email: {qchen2}@oxy.edu

***Abstract*—Software-intensive systems are expected to meet a variety of non-functional quality objectives essential for proper system functioning. However, there is still a lack of an effective estimation method for overall quality. Previous work has shown that it is possible to evaluate software systems' qualities based on issue summaries; however, this requires large amounts of manual effort to employ. There is a need for automated methods which can evaluate software systems as they evolve. Moreover, system qualities do not always change independently of each other. There are various dependencies which need to be taken into consideration when looking to make quality improvements. In this paper we look to explore the dependencies between bugs and the qualities they express. We have a fuzzy classifier that uses issue summaries to identify and summarize the quality of software with respect to maintainability, thus we propose a road map which will use this classifier to explore patterns for these relations and identify potential conflicts and synergies between these means-end qualities.**

***Index Terms*—software maintainability, software quality, dependency relationship, bugs**

I. Introduction

Non-functional quality objectives are an integral part of the software system development process. In particular, maintainability significantly affects costs over the development lifecycle [1]. Approaches have been developed to explore software qualities individually [2] [3] [4] [5], but there is a lack of understanding regarding the relationships between these software qualities (SQs) as the software evolves.

Current work has focused on conflicts that arise while defining non functional requirements [6]. For example, deciding whether a requirement for response time can be met while incorporating authentication steps to meet security requirements. However, SQs are not determined by the requirements that mandate them. They dynamically change over time as design decisions and changes are made to the software.

While ideally improvement of one SQ would either improve or at least not affect the other SQs, there are dependencies between these SQs which cannot be ignored. For example, when data or resources are modified to fix an accessibility issue, this can lead to inconsistencies with documentation that do not accurately reflect the structure, thus decreasing understandability. Understanding such relationships between SQs is essential for organizations to make more optimal decisions during maintenance.

Previous work has shown the potential for bug reports to be used as a means of measuring software maintainability using a software maintainability ontology [7]. As part of the analysis, we extracted bugs ("dependent bugs" that depended on other bugs "source bugs") and investigated the characteristics of these different bugs such as domain and number of unique dependency relationships. However, this method involved extensive manual effort to map bug reports to the different SQs defined in the ontology. We have since developed an automated method using a fuzzy classifier to tag issue summaries (including both bug reports and feature requests) with the appropriate SQs, which has been validated for identifying issue summaries expressing understandability concerns [2].

In this paper, we highlight a potential solution and propose a roadmap that uses the tag results provided by the classifier to identify and analyze the relationships among SQ concerns expressed in issue summaries.

The remainder of the paper is organized as follows: Section II elaborates how bugs are used to evaluate the software quality in open source communities and presents a preliminary study on bug dependency characteristics. Section III briefly explains a fuzzy classifier used to identify SQ concerns expressed in issue summaries and provides examples of such dependencies found in several open source software. Section IV details the planned approach. Section V summarizes the related work. Section VI concludes the proposed study.

II. Bug Dependency Characteristics

In Open Source Software (OSS) communities, bug reports are often used as a proxy metric for the quality of the software. Projects whose number of fixed bugs outnumber the number of introduced bugs are considered to have higher software quality.

Previous work has shown that more information can be extracted from issue summaries as SQ concerns can be identified from these summaries [7]. Thus it is possible to map issues to one or more SQs they are expressing. This alone does not establish any relationships between SQs. Fortunately, most of the issue tracking systems use dependency trees or graphs to represent the relationships between issues. An issue may depend on and/or block other issues. Figure 1, generated using Bugzilla Dependency Graph, shows an example of how these issues can depend on each other. Each number represents

978-1-7281-3925-8/19/$31.00 ©2019 IEEE
DOI 10.1109/QRS-C.2019.00032

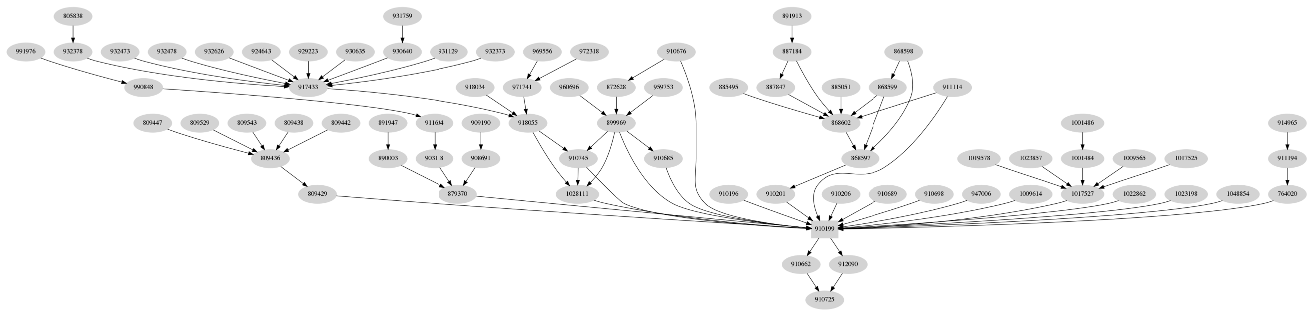


Fig. 1: Bug Dependency Graph

the bug ID and the arrow points from the source bug to the dependent bug.

An empirical study on Firefox OS was conducted to examine the characteristics of bug dependency relationships. We used data provided from its issue tracking system to illustrate such dependency relationships. Out of 10000 bugs, 3111 bugs either block or/and depend on other bugs. 2521 bugs blocks other bugs while 1183 bugs depend on other bugs in the system. This indicates that 1/3 of the bugs in the system do not exist alone. As shown in Figure 2, bugs that have high severity levels contains more bugs that either block or/and depend on other bugs. We use the severity label of each bug report found on the issue tracking systems and group them into one of the four severity groups: blocker, critical, major and others in descending order of severity. The others group includes bugs that are labeled normal, minor, trivia, and others. Almost 90% of blocker bugs are part of some bug dependency relationships. To sum up, the number of bugs that are in dependency relationships shows the potential for additional insight into relationships between expressed SQ concerns.

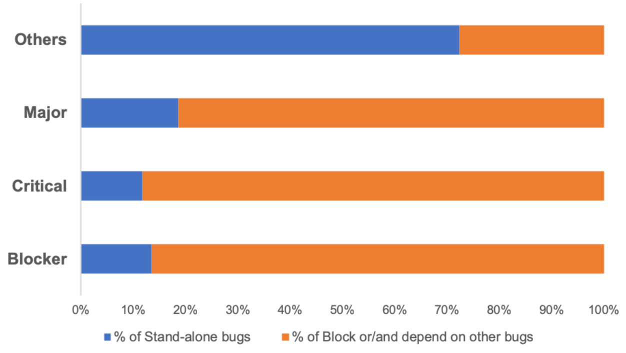


Fig. 2: Distribution of Bug Dependency per Severity Level

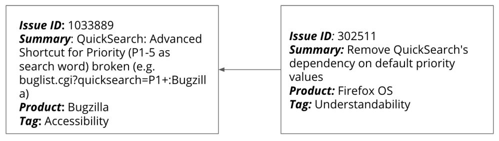
III. SQ Dependency Characteristics

A fuzzy classifier was developed to automatically identify maintainability subgroup SQ concerns expressed in issue summaries [8]. We utilized various natural language processing techniques to extract linguistic patterns presented in these summaries to incrementally discover fuzzy rules. The fuzzy classifier was evaluated on over 6000 issue summaries and the result indicated its high performance in correctly identifying the expressed SQ concerns.

Using the results from the classifier, we explored SQ dependency relationships found in several OSS projects and extracted the quality dependency relationships. The most frequent relationship originates with understandability bugs, which lead to other understandability bugs.

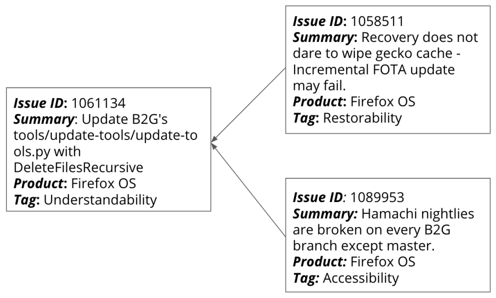
Figure 3 shows several examples of the extracted dependency relationships. The bug *1033889* found in Bugzilla received the tag "Accessibility". There is only one source bug in this case, which is found in Firefox OS with the tag of "Understandability". Therefore, the quality dependency relationship generated from this example is: "Accessibility" - "Understandability". The bug *1061134* found in Firefox OS received the tag "Understandability". There are two source bugs, which are both found in the same product, received "Restorability" and "Accessibility" respectively. In this case, the quality dependency is : "Understandability" - "Restorability, Accessibility".

Fig. 3: Dependency Relationship Examples


Issue ID: 1033889
Summary: QuickSearch: Advanced Shortcut for Priority (P1-5 as search word) broken (e.g. buglist.cgi?quicksearch=P1+:Bugzilla)
Product: Bugzilla
Tag: Accessibility

Issue ID: 302511
Summary: Remove QuickSearch's dependency on default priority values
Product: Firefox OS
Tag: Understandability

(a) One Source Bug

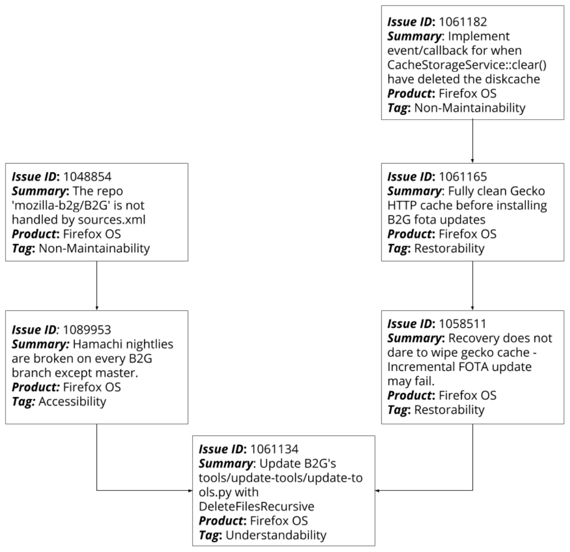

Issue ID: 1058511
Summary: Recovery does not dare to wipe gecko cache - Incremental FOTA update may fail.
Product: Firefox OS
Tag: Restorability

Issue ID: 1061134
Summary: Update B2G's tools/update-tools/update-tools.py with DeleteFilesRecursive
Product: Firefox OS
Tag: Understandability

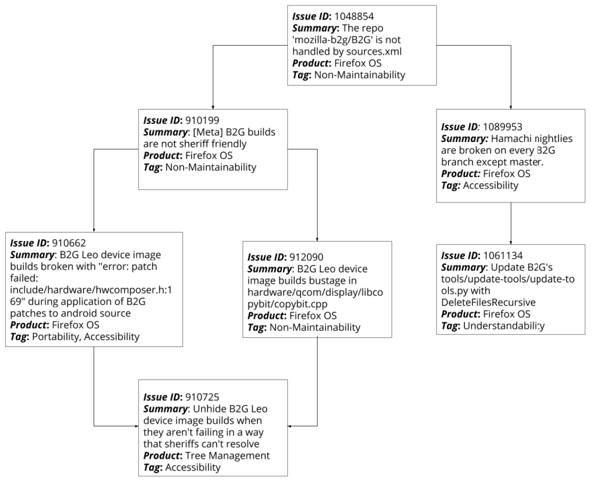
Issue ID: 1089953
Summary: Hamachi nightlies are broken on every B2G branch except master.
Product: Firefox OS
Tag: Accessibility

(b) Two Dependent Bugs

Fig. 4: Extended Dependency Relationship Examples


Issue ID: 1061182
Summary: Implement event/callback for when CacheStorageService::clear() have deleted the diskcache
Product: Firefox OS
Tag: Non-Maintainability
Issue ID: 1048854
Summary: The repo 'mozilla-b2g/B2G' is not handled by sources.xml
Product: Firefox OS
Tag: Non-Maintainability
Issue ID: 1061165
Summary: Fully clean Gecko HTTP cache before installing B2G fota updates
Product: Firefox OS
Tag: Restorability
Issue ID: 1089953
Summary: Hamachi nightlies are broken on every B2G branch except master.
Product: Firefox OS
Tag: Accessibility
Issue ID: 1058511
Summary: Recovery does not dare to wipe gecko cache - Incremental FOTA update may fail.
Product: Firefox OS
Tag: Restorability
Issue ID: 1061134
Summary: Update B2G's tools/update-tools/update-tools.py with DeleteFilesRecursive
Product: Firefox OS
Tag: Understandability

(a) Bug 1061134


Issue ID: 1048854
Summary: The repo 'mozilla-b2g/B2G' is not handled by sources.xml
Product: Firefox OS
Tag: Non-Maintainability
Issue ID: 910199
Summary: [Meta] B2G builds are not sheriff friendly
Product: Firefox OS
Tag: Non-Maintainability
Issue ID: 1089953
Summary: Hamachi nightlies are broken on every B2G branch except master.
Product: Firefox OS
Tag: Accessibility
Issue ID: 910662
Summary: B2G Leo device image builds broken with "error: patch failed: include/hardware/hwcomposer.h:169" during application of B2G patches to android source
Product: Firefox OS
Tag: Portability, Accessibility
Issue ID: 912090
Summary: B2G Leo device image builds bustage in hardware/qcom/display/libcopybit/copybit.cpp
Product: Firefox OS
Tag: Non-Maintainability
Issue ID: 1061134
Summary: Update B2G's tools/update-tools/update-tools.py with DeleteFilesRecursive
Product: Firefox OS
Tag: Understandability
Issue ID: 910725
Summary: Unhide B2G Leo device image builds when they aren't failing in a way that sheriffs can't resolve
Product: Tree Management
Tag: Accessibility

(b) Bug 1048854

Although our preliminary analysis is limited to the dependency relationship between pairs of bugs, the dependency relationships can extend much further as seen in Figure 1. Figure 4 illustrates some examples of this extended dependency relationship found in Firefox OS. Some bugs may appear only in one chain of dependency relationship, whereas others may appear in multiple chains of dependency.

These results provide insights on how bugs and their expressed SQ concerns related to each other within the scope of the investigated projects. More specifically, it shows how bug dependency could be used to reflect the relationships between expressed SQ concerns, thus validating the possibility of such proposed approach.

IV. Roadmap

This section lays out the planned approach for completing this research.

1) Exploring the approach with maintainability:
   - Empirical study: We plan to use our fuzzy classifier to tag issue summaries from a larger set of OSS projects. Once the issue summaries have been tagged, we will extract the issue summaries that depend on and/or block other issues and perform an analysis on their common characteristics (*e.g.*, time of reporting, life-cycle phase status, issue severity, fix-time, etc).
   - Define a relationship framework for the maintainability ontology: Taking the extracted issue summaries along with their SQ concerns, we will define a framework that establishes the different types of relationships among the expressed means-end maintainability quality concerns based on identified attributes such as whether they are blockers or causal.
2) Extending the scope and the classifier:
   - This approach can be extended to include other software qualities by expanding the performance of the classifier to automatically tag issue summaries with quality concerns beyond maintainability.
   - Another direction would be to look at the root causes of the issue summaries as was defined in the prior work [7] to understand the reasons why certain dependency relationships among qualities exist. In this case, the classifier would be expanded to include identification of root causes such as syntax/semantic errors, unexpected interactions, test cases and more.

V. Related Work

Saadatmand et al [9] provided a UML-based approach for conducting tradeoff analysis of non-functional requirements which required users to provide information detailing relationships between features and the non-functional requirements. They have extended that work to incorporate fuzzy logic and TOPSIS decision making. With these extentions, their approach can evaluate different design alternatives and identify those which best satisfy the non-functional requirements.

Roh et al. [10] examined conflicts between security and usability within the context of mobile applications. They proposed an approach for predicting the conflicts during the requirements engineering process by constructing an ontological knowledge base. They provided two case studies in which they compared their approach to a survey of graduate students and found they could derive more than twice the number of conflicts with their approach.

Kobayashi et al. [11] introduced an approach for quantitative evaluation of non-functional requirements using a softgoal weight extension of Softgoal Interdepencency Graphs. Softgoals are decomposed into subsoftgoals and their weights are compared to determine which alternatives should be chosen.

Prifti et al. [12] investigated bug report duplication trends in the open source project, Firefox and proposed an approach for detecting duplicate reports. As part of their duplication trend analysis, they found that dependencies have effects on the total number of duplicates per group and distribution of group sizes. One of their goals is for users to be able to add to existing reports rather than increasing the number of duplicates that need to be examined.

Sandusky et al. [13] conducted an empirical study on how one F/OSS community used a bug report network (BRN) as part of its bug report repository. These BRNs contain relationship information between bug reports provided by community members, including duplication, dependency, and additional informal relationships. Of their sample of bug reports, they found that 65% were part of some kind of BRN.

VI. Conclusions

In this paper, we emphasized the lack of research that has been done to explore the dependency relationships among SQs and presented the benefits of understanding such relationships. Previous work has shown that issues can be mapped to expressed SQ concerns. A fuzzy classifier has been constructed to automatically produce these mappings. Thus, we proposed an idea that utilizes the existing bug dependency relationships with these mapping results to identify potential dependency relationships among expressed SQ concerns. A preliminary analysis on several open source software identified the top dependency relationships between maintainability subgroups SQs. With the proposed roadmap, we plan to conduct further research to examine those relationships in a large empirical study. We believe that such approach can provide a more thorough high level understanding of software qualities.

References

[1] C. Chen, R. Alfayez, K. Srisopha, B. Boehm, and L. Shi, "Why Is It Important to Measure Maintainability and What Are the Best Ways to Do It?" in *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, May 2017, pp. 377–378.

[2] C. Chen, M. Shoga, B. Li, and B. Boehm, "Assessing Software Understandability in Systems by Leveraging Fuzzy Method and Linguistic Analysis," in *2019 Conference on Systems Engineering Research (CSER) (2019 CSER)*, Washington, USA, Apr. 2019.

[3] A. L. Goel, "Software reliability models: Assumptions, limitations, and applicability," *IEEE Transactions on software engineering*, no. 12, pp. 1411–1423, 1985.

[4] R. P. Buse and W. R. Weimer, "A metric for software readability," in *Proceedings of the 2008 international symposium on Software testing and analysis*. ACM, 2008, pp. 121–130.

[5] J. L. Taaffe, *Software security method and systems*. Google Patents, May 1988.

[6] B. Boehm and X. Franch, "Conflicts and Synergies among Quality Requirements," in *2017 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, Jul. 2017, pp. 507–508.

[7] C. Chen, S. Lin, M. Shoga, Q. Wang, and B. Boehm, "How Do Defects Hurt Qualities? An Empirical Study on Characterizing a Software Maintainability Ontology in Open Source Software," in *2018 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, Jul. 2018, pp. 226–237.

[8] C. Chen, M. Shoga, and B. Boehm, "Characterizing software maintainability in issue summaries using a fuzzy classifier," (in press).

[9] M. Saadatmand and S. Tahvili, "A Fuzzy Decision Support Approach for Model-Based Tradeoff Analysis of Non-functional Requirements," in *2015 12th International Conference on Information Technology - New Generations*, Apr. 2015, pp. 112–121.

[10] W. Roh and S. Lee, "An Ontological Approach to Predict Trade-Offs between Security and Usability for Mobile Application Requirements Engineering," in *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*, Sep. 2017, pp. 69–75.

[11] N. Kobayashi, S. Morisaki, N. Atsumi, and S. Yamamoto, "Quantitative Non Functional Requirements evaluation using softgoal weight," *J. Internet Serv. Inf. Secur.*, vol. 6, pp. 37–46, 2016.

[12] T. Prifti, S. Banerjee, and B. Cukic, "Detecting Bug Duplicate Reports Through Local References," in *Proceedings of the 7th International Conference on Predictive Models in Software Engineering*, ser. Promise '11. New York, NY, USA: ACM, 2011, pp. 8:1–8:9, event-place: Banff, Alberta, Canada. [Online]. Available: http://doi.acm.org/10.1145/2020390.2020398

[13] R. Sandusky, "Bug report networks: varieties, strategies, and impacts in a F/OSS development community," in *"International Workshop on Mining Software Repositories (MSR 2004)" W17S Workshop - 26th International Conference on Software Engineering*, vol. 2004. Edinburgh, Scotland, UK: IEE, 2004, pp. 80–84. [Online]. Available: https://digital-library.theiet.org/content/conferences/10.1049/ic$_2$0040481