Induction of Co-existing Items Available in Distributed Version Control Systems for Software Development

Sibel Özyer
Department of Computer Engineering, Ankara Medipol University, Turkey
sbltariyan@gmail.com

Abstract: *Software development in Open-Source Software systems (OSS) allow developers to share their code and modify other developers' code. That leads to collaboration in the development. They can either discuss on the items to be developed, including the errors and technical problems that were faced. One popular OSS platform is github which already has a large number of developers and projects. The data residing in the issues part of github is sufficiently large, complex and unstructured. It could be processed to find novel discoveries. This work concentrates on one selected project to be analyzed systematically. Routine Extract, Transform and Load (ETL) steps have been identified to clean the data before applying natural language processing for prioritizing and taking actions for the requirements. In a collaborative environment. Our work uses terms and guides developers for tracking the co-occurrence of the terms used together to help them focus on the important issues.*

Keywords: *Version control systems, requirement analysis, co-occurrence analysis, natural language processing.*

Received August 31, 2022; accepted July 20, 2022
https://doi.org/10.34028/iajit/20/6/4

1. Introduction

The Open-Source Systems (OSS) domain is prevalent in software development. Distribution policy with a licence allows developers to access the codes and redistribute them for their derivations and use. It includes its structure, ownership, membership, and contribution policy [1]. The developers had the motivation to participate in OSS organizations such as altruism, identification of their community, marketing themselves, increasing their skill base, selling their similar products and their personal needs [5]. Resource management and teamwork require version control systems.

In OSS ecosystem, stakeholders are persons and firms who influence the system or are impacted by the system [4]. Requirement Engineering (RE) is concerned with the discovery of requirements imposed by stakeholders and documentation for the development activities. Stakeholders form a social network having core developers and passive users [1].

RE predominantly involves the discovery of stakeholders' needs, which may be ambiguously described; this may need more elaboration for the resolution of the conflicting requirements [11].

Requirement negotiation needs to elicit the problems that exist in the current system and are not identified in the development phase. Here, the requirements are refined further and validated. Meantime, quick development of software applications with minimal cost and shorter amount of time is another target that causes reusing existing products in a distributed development environment [7].

Version Control System (VCS) is used for managing the evolution and development. It facilitates the management of the source code and keeps past versions. Developers collaborate in a coordinated and effective way [19].

According to the description in [19], VCS is categorized in two classes: Centralized and distributed. Git is a typical example of Distributed VCS. It has been incorporated with the github web site.

The popularity of git and Github is increasing due to their robustness, predisposition to teamwork from the aspects of collaboration, communication and carrying out the necessary part of the software engineering principles. As of 2022, GitHub reported over 83 million developers, 4 million organizations, and more than 327 million repositories. 44 million of which are public repositories.

During the software development, the developers and users in the community report errors using natural language. In general, issues are used to report bugs, or suggest new features [16]. Requirements are expressed in issues. Issues are transformed into relational information between the phrases to structure text data. Semi-automated detection of requirements would be feasible [17]. However, a full automation should be abstained [18]. Finally, there are attempts to extract requirement related data from text [17].

This study addresses the issues in software development by concentrating on a particular project from Github, namely, the MS Typescript project. It is a syntactical superset of javascript. The target is to analyze and describe the issues and prioritize frequent items as requirements. Issues have been manually categorized into four requirement types, namely, performance, documentation, installation, and information. These types have been inspected individually. After part of speech tagging of the data, nouns and adjectives have been particularly used for co-occurrence analysis by employing association rule mining [6].

An issue may be related to one or more classified requirement types. It will help us get sensitized to the problem. An accurate forecast of the resources includes budget, manpower and project work package identification with time planning. Critical parts can be overly tested and improved.

The rest of the paper is organized as follows. The scope of work is covered in section 2. The developed methodology is described in section 3. The experiments and results are included in section 4. Section 5 is conclusions and future work.

2. Scope of Work

Issues that have been collected as text can be processed for knowledge discovery. Co-occurrence of words in a particular domain reveals the strength of an important role for better analysis. Natural language processing has been employed from different perspectives to extract important terms, topics, and entities, optionally including their sentimental polarity with their relationships [9].

Requirement engineering may also need to be processed for the elicitation step. Because of the previously mentioned reasons, a semi-automated system would be preferred.

Part Of Speech (POS) Tagging has been used to link the words with tags to connect them to different types, such as noun phrase, adjective, verb, proposition, interjection, etc., [12]. Semantic Tagging is another direction for extracting elements and annotate them to ontological terms [15]. A framework for stakeholder profiling has been proposed by clustering the needs. Needs in a cluster are prioritized [2].

Described in [8] is another study that proposes a deep learning-based approach for recommending a new repository. Topic modeling also plays an important role in RE [13].

3. System Overview

Open source systems data, especially github data, is resourceful for RE which contains issues for the projects under development. Our system first, collects the issues. After they are collected from the github site, they are categorized as documentation, informative, performance and installation issues. Collected data is cleansed; a pos tagger is employed following the stop word removal process. Next, the co-occurrence analysis, the association rule mining approach was applied on the processed data. It uncovers the association rules that lead to identifying significant results; they are ranked. In the following steps, actions that were taken have been described.

- Issue Collection: the issues have been collected from MS Typescript (github.com/Microsoft/TypeScript) project which is a superset of the popular JavaScript. The reason for the selection of this project is its continuous support and development by a large number of developers and users. The common approach is that the breakdown of the requirements can be under four types, including documentation, informative, installation and performance type issues. In this study, the collected data have 143 issues in total. They have been inspected manually, and have been categorized thoroughly. After categorizing the data, further pre-processing steps have been taken. For better analysis, descriptions which had more user comments were given more preference. The collected data has been categorized, and its distribution has been given as the total number of issues, which are distributed as follows: 20 are installation, 28 are performance, 36 are documentation, and 50 are informative.
- Cleansing the Content: the content can intertwine since it is in unstructured and natural form. The data has been cleansed with regular expressions; the mark up content, emotions, and icons have been eliminated. After cleansing the data, stop word elimination is applied at the next step.
- Eliminating Insignificant Words: textual content can be disorganized; it may contain the same word repeated at several locations. Besides, commonly used insignificant words, the so called stop words must be eliminated. A list of 421 stop words has been compiled [3].
- Applying the POS Tagger: in this study, after POS tagging, only nouns and adjectives have been considered and kept for further use. Others have been discarded. Empirical results indicate that nouns and adjectives give promising results. For this purpose, the Stanford CoreNLP POS Tagger has been used.
- Employing Market Basket Analysis: market basket analysis of data mining has been utilized in several areas, including software engineering such as defect analysis, bug analysis, false test alarming, and change in source code analysis with impact analysis [10, 14].
- Scoring the Extracted Words: after getting the words as frequent patterns, words co-occurring will lead to the rules. At this step, the top k rules are selected based on their confidence values.

The rules related to the issues have been represented with the frequency count of each item. Items stand for the words. For each item t_i its support value $freq_i$ is assessed. Then items are ranked based on the occurrence number. Here, term t_1 exists with a support value of $freq_1$; other terms that coexist with t_1 are detected. When t_2 occurs with t_1 with a support value of $freq_2$, then t_2 contributes with a support value of $freq_2$. The higher the value of $freq_2$ is, the more influence t_2 has on t_1. The ranked results of item combinations with the frequency proportion can be presented to the decision makers for better interpretation of the issue types in four directions. Experiments and discussions are given in section 4.

4. Experiments and Discussion

Four issue types, namely documentation, installation, performance and informative have been considered. Each of these issue types has been analyzed separately in the following sections. The terms have been assessed and their analysis have been conducted for each individual issue type. In our study association rule mining has been run with minimum support value specified as minsup=.08 and minimum confidence value specified as minconf=.8. We have taken top k=10 rules out of the list of all rules. Rules containing similar items have been removed.

- Informative Type Issues: the methodology we have described in section 3 has been applied for the informative type issues. The association rules mining algorithm has been run with the abovementioned parameter values. Tables 1 and 2 list the nine rules after eliminating one duplicate; they are ranked.

Table 1. Frequent item sets of informative type issues.

Items	Count	Confidence
value, variable	7	1
code, type, variable	5	
string, type	5	
function, variable	10	0.91
statement, variable	6	0.86
behaviour, variable	5	0.83
behaviour, typescript	5	
line, typescript	5	
fine, variable	5	

The calculations have been derived from the rules that were extracted from the data, mainly the data related to the informative type issues. Here, Table 2 displays the final results in ranking the terms. It relies on the narrowed list in Table 1. Here the two components, initial term and the accompanying terms which define the influence to the initial term.

Table 2. Analysis of frequent item sets of informative issue type.

Initial Term	Accompanying Terms	Count/Over
variable	function	10/38
variable	value	7/38
variable	statement	6/38
variable	behaviour	5/38
variable	fine	
variable	code, type	
type	string	5/10
type	code, variable	
behaviour	typescript	
behaviour	variable	
typescript	behaviour	
typescript	line	

The study on the information type issues takes the proportion of frequencies into account. It requires the observations of 50 issues containing these terms. The prominent initial terms in Table 2 are type, behaviour, typescript and variable. Using these terms with the accompanying terms, Informative issues are analyzed manually. For the informative issues type, among the issues, it has been realized that variable declarations were inconsistently used in the code for some values and data types. Here, the data type string stands out. It is considered as problematic, in particular for lines existing under function declarations in the code.

After pointing out the problems in the software, test scenarios should be created for debugging purposes. These findings pave the way for the detailed analysis of relevant descriptions. Test cases are constructed after the analysis by looking at the influence of the terms. Test cases are investigated and the main reason can be figured out.

An accurate identification of the problem, will diminish the effort to solve. Problems can be prioritized and they can be assessed for scheduling and for determining the number of developers needed. This knowledge will help us predict when the next release would be possible.

- Installation Type Issues: the methodology we have described in section 3 has been applied for the informative type issues. Association rule mining algorithm has been run with the abovementioned parameter values. Tables 3 and 4 show the ranked nine rules after removing duplication.

The results have been obtained by looking at the rules that are the by-product of the installation type issues. According to the results, it can be clearly stated that visual and studio terms coexist together as one phrase. It appears that it is the subpart of the whole expression, MS Visual Studio IDE. Table 4 displays the finalized list. The list is ranked. Initial terms are taken and the accompanying terms are the co-occurring terms. They point to the contribution of each accompanying term to the initial term.

Table 3. Frequent item sets of installation issue types.

Items	Count	Confidence
studio, typescript, visual	8	1
file, typescript	7	
studio, typescript	6	
typescript, compiler	6	
visual, studio	6	
file, project, typescript	5	
version, visual, studio	5	
file, version, typescript	5	
fine, variable	5	

The study on the installation type issues takes the proportions of frequencies into account. It requires the observations of 29 issues containing these terms.

Initial terms that are notable are the terms: version, typescript, file, visual-studio, and studio. By looking at these initial terms and accompanying terms manually, it is inferred that the typescript project points to some discrepancies for a particular version while compiling under MS Visual Studio IDE.

Table 4. Analysis of Frequent item sets of the installation issue types.

Initial Term	Accompanying Terms	Count/Over
typescript	file	6/40
typescript	compiler	
typescript	studio	
typescript	visual-studio	5/40
typescript	file, project	
typescript	file, version	
typescript	studio, version	
studio	typescript	6/11
studio	typescript, version	5/11
visual-studio	type	6/19
visual-studio	version	5/19
file	typescript, project	5/17
file	version, typescript	
file	typescript	7/17
version	line	5/15

Another inference from Table 4 is that the terms relating to the aspects mentioned above often exist. Installation issues form critical criteria for using any scripting language. Ease in practical use and compatibility come first. These aspects turn it into a popular language.

Varying IDEs may run under differing platforms. They must be compatible. Developers are sensitive to IDE's they have been using.

Domain experts can easily figure out the problems with the version by checking the relevant issues. This may also reduce the time for detecting the problem with the version.

A clear explanation for the issue of the installation, this may be any problem varying from the compilation within a IDE to platform specific installation. The team responsible for that particular task is notified with the detailed requirement description.

This will also require the preparation of the testing stage and maintaining the code.

Finally, actions that are taken are expected to be finished timely. Addressing the issues will be a good start for better planning.

- Documentation Type Issues: the methodology we have described in section 3 has been applied for the documentation issues. The association rule mining algorithm has been run with the abovementioned parameter values. Tables 5 and 6 show the ranked eight rules after removing the duplication. Below, there are items that are considered to be important; they have been ranked. The calculations have been done by using the issues of type documentation. Based on the pre-specified support and confidence parameter values, the rules that have been discovered present the items in Table 6. During the frequency computation, it has been concluded that the two terms, visual and studio occur together, so they are taken as one phrase.

Table 5. Frequent item sets of documentation type.

Items	Count	Confidence
new, typescript	4	1
issue, documentation	3	
good, typescript	3	
topic, language	3	
node, typescript	3	
visual, studio	3	
way, typescript	3	
language, typescript	4	0.8

Table 6 resorts the terms. The initial term stays under inspection, and the accompanying terms represent their impact on the initial term.

The study on the documentation type issues takes the proportions of the frequencies into account. It requires the observations of 36 issues containing these terms. Initial terms that are notable are the terms: language and typescript. It can be stated that there are limited number of issues of type documentation. The subject of the concerns are more common among the issues. In parallel, this is also supported when its content is inspected. One direction is the fact that there is a need to clarify the language of the documentation due to its ambiguity.

Initial and accompanying terms together reveal the fact that the layout and the way it is disseminated are the main concerns. These concerns have been supported when the documents are elaborated. The issues expose the need for introducing features not existing or undetailed.

Table 6. Analysis of frequent item sets of documentation issue type.

Initial term	Accompanying terms	Count/Over
typescript	language	4/17
typescript	new	
typescript	good	3/17
typescript	node	
typescript	way	
language	typescript	4/7
language	topic	3/7

All of these findings lead us to the corrections in the document. It should be imminent to revise the document. Apart from the software itself, one of its supportive parts is documentation which gives the hints on understanding and using the language. Insufficient documentation of a software may hinder the use of a software in development. Detection of weak points besides its ambiguity can be addressed easily after focusing on the necessary parts. Selected developers who are in charge of the development may give more information or examples on particular parts of it.

- Performance Type Issues: the methodology we have described in section 3 has been applied for the performance issues. Association rule mining algorithm is run with the abovementioned parameter values. Table 7 show the initial results. Here, performance is common across all the rows. Here, that shows us that terms other than the performance will be the accompanying terms. The ranking will not change.

Table 7. Frequent item sets of performance type.

Item sets	Count	Confidence
way, performance	7	1
output, performance	5	
compilation, performance	5	
great, performance	5	
possible, performance	4	
class, performance	4	
module, performance	4	
Tsc (compiler), performance	4	
performance	4	
compilation, file	4	
useful, performance	4	

The study on the performance type issues takes the proportion of frequencies into account. It requires the observations of 28 issues containing these terms.

Performance is the sole term. By taking into account both the initial and accompanying terms, when the issues having these terms are analysed, some classes and modules are hard to use. Because of trials and errors, the expected results according to the issues may not be giving. This also causes hard time in compiling the code snippets in terms of the performance; translating it to another script in shorter amount of time.

These issues can be addressed easily by the team. They may be included in the to do list with some priorities. Furthermore, next versions will be arranged so that critical portions of the software can be improved (using typescript) to the developers satisfaction.

5. Conclusions

We have studied the open source software development environments. One of the important OSS was git and github together. They alleviate the development of a software in a distributed environment. It may be public or private. Especially when the development environment is public, developers and users may together serve to the improvement of the software with the issues they have created. They are expressed with natural language.

These issues are helpful for grasping the needs of the software. A system requiring human involvement has been proposed, with the help of managers, an affirmation of the needs is necessary because the absolute accuracy must exist to prefer a fully automated system. Differences in software would also cause the shift in concepts. However, any missing understanding in the requirement or priorities would cause the software to fail or at least hamper the next versions. It also will affect the schedule and resource plan of the development in terms of money, time and human.

Natural language processing yet presents us an opportunity for the reuse of the issues in github to elaborate the needs for the software. A categorization of the requirements has been identified. A manual categorization of issues has been performed to robustly work on the terms. A pre-processing step cleanses the date, removes the outliers and only considers nouns and adjectives selectively. A market basket data analysis has been performed on the terms. Frequently occurring terms in sets have been retained and ranked for prioritization. This facilitates to understand the needs. An efficient resource planning can be possible.

6. Future Work

There is still room to be developed. A semi-automated system can be improved. Manually categorizing issues also can be performed automatically while transitioning to another software with transfer learning methods. At least, a metric can be devised to reduce the time spent for manual categorization.

We have studied a limited amount of time for limited number of issues. It may also be scaled out for larger projects. Sentiment analysis in this work can also be incorporated to fully understand positive and negative aspects. It would also help us reviewing the expectations against the feedbacks.

In the future one more direction would be to overlook the evolution of the software throughout the versions to understand how the software has improved and what needs are realistic with a lookback option. Our work will also be validated by the existing methods in the future.

References

[1] Androutsellis-Theotokis S., Spinellis D., Kechagia M., and Gousios G., “Open Source Software: A Survey from 10,000 Feet,” *Foundations and Trends® in Technology, Information and Operations Management*, vol. 4, no. 3-4, pp. 187-347, 2011. DOI:10.1561/0200000026

[2] Castro-Herrera C., Cleland-Huang J., and Mobasher B., “Enhancing Stakeholder Profiles to Improve Recommendations in Online Requirements Elicitation,” *in Proceeding of the 17th IEEE International Requirements Engineering Conference*, Atlanta, pp. 37-46, 2009. doi: 10.1109/RE.2009.20.

[3] Fox C., “A Stop List for General Text,” *ACM SIGIR Forum*, vol. 24, no. 1-2, pp. 19-21, 1989. https://doi.org/10.1145/378881.378888

[4] Glinz M. and Wieringa R., “Guest Editors' Introduction: Stakeholders in Requirements Engineering,” *IEEE Software*, vol. 24, no. 2, pp. 18-20, 2007.

[5] Hars A. and Ou S., “Working for Free? Motivations for Participating in Open-Source Projects,” *International Journal of Electronic Commerce*, vol. 6, no. 3, pp. 25-39, 2002. doi: 10.1109/HICSS.2001.927045.

[6] Kaushik M., Sharma R., Peious S., Shahin M., Yahia SB., and Draheim D., “A Systematic Assessment of Numerical Association Rule Mining Methods,” *SN Computer Science*, vol. 2

no. 5. pp. 348, 2021. https://doi.org/10.1007/s42979-021-00725-2

[7] Khan H., Niazi M., El-Attar M., Ikram N., Khan S., and Gill A., "Empirical Investigation of Critical Requirements Engineering Practices for Global Software Development," *IEEE Access*, vol. 9, pp. 93593-613, 2021. doi: 10.1109/ACCESS.2021.3092679.

[8] Kim J., Wi J., and Kim Y., "Sequential Recommendations on GitHub Repository," *Applied Sciences*, vol. 11, no. 4, pp. 1585, 2021. https://doi.org/10.3390/app11041585

[9] Mahmoud A. and Zrigui M., "Semantic Similarity Analysis for Corpus Development and Paraphrase Detection," *The International Arab Journal of Information Technology*, vol. 18, no. 1, pp. 1-7, 2021. https://doi.org/10.34028/iajit/18/1/1

[10] Morisaki S., Monden A., Matsumura T., Tamada H., and Matsumoto KI., "Defect Data Analysis Based on Extended Association Rule Mining," *in Proceedings of the 4th International Workshop on Mining Software Repositories*, Minneapolis, pp. 3-3, 2007. doi: 10.1109/MSR.2007.5.

[11] Nawaz S., Zai A., Imtiaz S., and Ashraf H., "Systematic Literature Review: Causes of Rework in GSD," *The International Arab Journal of Information Technology*, vol. 19, no. 1, pp. 97-109, 2022. https://doi.org/10.34028/iajit/19/1/12

[12] Portugal R., Li T., Silva L., Almentero E., Leite J., "Nfrfinder: A Knowledge Based Strategy for Mining Non-Functional Requirements," *in Proceedings of the XXXII Brazilian Symposium on Software Engineering*, Sao Carlos Brazil, pp. 102-111, 2018. https://doi.org/10.1145/3266237.3266269

[13] Ray B., Posnett D., Filkov V., and Devanbu P., "A Large Scale Study of Programming Languages and Code Quality in Github," *in Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, Hong Kong, pp. 155-165, 2014. https://doi.org/10.1145/3126905

[14] Sharma M., Kumari M., and Singh V., "Bug Assignee Prediction Using Association Rule Mining," *in Proceedings of the International Conference on Computational Science and Its Applications*, Banff, pp. 444-457, 2015. https://doi.org/10.1007/978-3-319-21410-8_35

[15] Sonbol R., Rebdawi G., and Ghneim N., "Towards a Semantic Representation for Functional Software Requirements," *in Proceedings of the IEEE 7th International Workshop on Artificial Intelligence for Requirements Engineering*, Zurich, pp. 1-8, 2020. doi: 10.1109/AIRE51212.2020.00007.

[16] Xiao W., He H., Xu W., Tan X., Dong J., and Zhou M., "Recommending Good First Issues in GitHub OSS Projects," *in Proceedings of the 44th International Conference on Software Engineering*, pp. 1830-1842, 2022. https://doi.org/10.1145/3510003.3510196

[17] Yang Y., Xia X., Lo D., Bi T., Grundy J., and Yang X., "Predictive Models in Software Engineering: Challenges and Opportunities," *ACM Transactions on Software Engineering and Methodology*, vol. 31, no. 3, pp. 1-72, 2022. https://doi.org/10.1145/3503509

[18] Ziora L., "Natural Language Processing in the Support of Business Organization Management," *in Proceedings of SAI Intelligent Systems Conference*, Amsterdam, pp. 76-83, 2021. https://doi.org/10.1007/978-3-030-82199-9_6

[19] Zolkifli N., Ngah A., and Deraman A., "Version Control System: A Review," *Procedia Computer Science*, vol. 135, pp. 408-15, 2018. DOI:10.1016/j.procs.2018.08.191

Sibel Özyer received her BSc and MSc from Cankaya University, and PhD degree from Atilim University. She is currently assistant professor at Ankara Medipol University. Her research interests are social networks, computer networks, data mining, internet of things and cloud computing.