# Evaluating and Improving Software Quality Using Text Analysis Techniques - A Mapping Study

Faiz Shah, Dietmar Pfahl

Institute of Computer Science, University of Tartu
J. Liivi 2, Tartu 50490, Estonia
{shah, dietmar.pfahl}@ut.ee

**Abstract**:
Improvement and evaluation of software quality is a recurring and crucial activity in the software development life-cycle. During software development, software artifacts such as requirement documents, comments in source code, design documents, and change requests are created containing natural language text. For analyzing natural text, specialized text analysis techniques are available. However, a consolidated body of knowledge about research using text analysis techniques to improve and evaluate software quality still needs to be established.
To contribute to the establishment of such a body of knowledge, we aimed at extracting relevant information from the scientific literature about data sources, research contributions, and the usage of text analysis techniques for the improvement and evaluation of software quality.
We conducted a mapping study by performing the following steps: define research questions, prepare search string and find relevant literature, apply screening process, classify, and extract data.
Bug classification and bug severity assignment activities within defect management are frequently improved using the text analysis techniques of classification and concept extraction. Non-functional requirements elicitation activity of requirements engineering is mostly improved using the technique of concept extraction. The quality characteristic which is frequently evaluated for the product quality model is operability. The most commonly used data sources are: bug report, requirement documents, and software reviews. The dominant type of research contributions are solution proposals and validation research.
In our mapping study we identified 81 relevant primary studies. We pointed out research directions to improve and evaluate software quality and future research directions for using natural language text analysis techniques in the context of software quality improvement.

**Keywords.** Mapping study; software quality; text analytics; unstructured data;

## 1 Introduction

Since natural language text doesn't have an explicit data model, it is commonly referred to as "unstructured data" [1]. It is estimated that 80 to 85 percent of data in

software repositories is unstructured [1], because most of the software artifacts, such as requirement documents, architecture documents, test cases, commit messages, change requests, developer's mailing list, and source code comments, contain unstructured data [2]. Therefore, utilizing unstructured data is important to derive useful information from software data [3].

Apart from analyzing software artifacts created by developers, testers, and architects, textual data available in the internet, such as reviews, tweets, and discussion forums, have also been analyzed for requirement engineering [4], software evolution [5], and bug prediction [6]. Over the recent years, interest in analyzing unstructured software data has gained attention among researchers and lead to the creation of many workshops and conferences including NaturaLiSE (International Workshop on Natural Language Analysis in Software Engineering), TEFSE MUD (Mining Unstructured Data) and MSR (Working Conference on Mining Software Repositories) [1].

We were interested in understanding how the analysis of unstructured data generated and used during software development could be beneficial to improve and evaluate software quality. To understand the existing body of knowledge, we conducted the mapping study with the aim to review the literature in which text analysis techniques has been applied for improvement and evaluation of software quality. We found few mapping studies [7][8][9] that review literature on evaluation of software quality. However, these studies focused on structured data or have different scope and objective than our mapping study. By this mapping study, we addressed the following research goals: **1)** To discover the data sources which have been used; **2)** To know the research contributions that have been made; **3)** To investigate text analysis techniques which have been employed; **4)** To understand how does text analysis help in improving and evaluating software quality.

This paper is organized as follows. In Section 2 we discuss the related literature. In Section 3 we present the research methodology used to filter out the primary studies relevant for our mapping study. In Section 4 we provide details about the classification scheme used to categorize the primary studies. Results of the study with detailed analysis are presented in Section 5. In Section 6, we present conclusions and provide pointers for future research.

## 2 Related Work

Recently, Sofia et al. conducted a mapping study [7] on evaluation of software product quality. However, this study didn't cover studies that used text analysis techniques for improvement and evaluation of software quality. A mapping study carried out by Misirli et al. [8] is also different from our study, because it only includes studies that used Bayesian network for evaluation of software quality. In another mapping study [9], only those studies were selected that evaluated the software quality of gamified applications. In contrast to the aforementioned studies, our mapping study covered the studies which attempted to improve and evaluate software quality using the techniques of text analysis.

Casamayor el al. performed a literature review [10] on applications of text mining in software architectural design. Similarly, Borg et al. [11] conducted a systematic mapping study (SMS) of information retrieval approaches to software tractability.

Contrary to these studies, our mapping study reviewed the literature that focuses on using text analysis techniques for improvement and evaluation of software quality.

## 3 Research Methodology

To review the literature that applied text analysis techniques for improvement and evaluation of software quality, we conducted a mapping study following the guidelines proposed by Petersen in [12].

### 3.1 Research Questions

We tackled the following research question and sub-questions in this mapping study.

RQ 1: To what data sources have text analysis techniques been applied?
RQ 2: What types of research papers have been published?
RQ 3: What techniques of text analysis have been used?
RQ 4: How does text analytics help in improving and evaluating software quality?

### 3.2 Strategy

**Data Sources.** As our study focuses on research aiming at improving and evaluating software quality using text analysis techniques, we looked for relevant literature in the most popular digital libraries of computer science (CS) and software engineering (SE) which are as follow: IEEE, ACM Digital, Scopus, Science Direct/Elsevier, and Web of Knowledge.

**Keywords.** Keeping in mind the research questions, suitable keywords and their synonyms were determined based on our general background knowledge related to software quality and text analytics. The following keywords were used to build the search strings:

- *Software quality* is a broad term and also referred to using other terms, such as software quality analysis, software quality assurance, software testing, software product quality, software bugs, software defects, bug reports.
- *Text analysis techniques.* Alternative keywords are: text mining, text analytics, text analysis, sentiment analysis, opinion mining

The keywords and alternative keywords identified in software quality and text-analysis techniques were joined by using the Boolean operator OR. Software quality and text analysis techniques were joined with each other by using the Boolean operator AND. The query designed to search the relevant literature is as follows:

*(("software quality" OR "software testing" OR "software product quality" OR "software quality analysis" OR "software bugs" OR "software defects" OR "bug reports") AND ("opinion mining" OR "text mining" OR "sentiment analysis" OR "text analysis" OR "text analytics"))*

The search in the databases was limited to title, abstract, and keywords. Only articles published during the period from 2008 to 2015 were targeted. We executed the search query on the mentioned databases on 26th October 2015. Fig. 1 provides the overview of search results and selection methodology that we used to filter out the primary studies. Section 3.3 describes our selection methodology in detail.

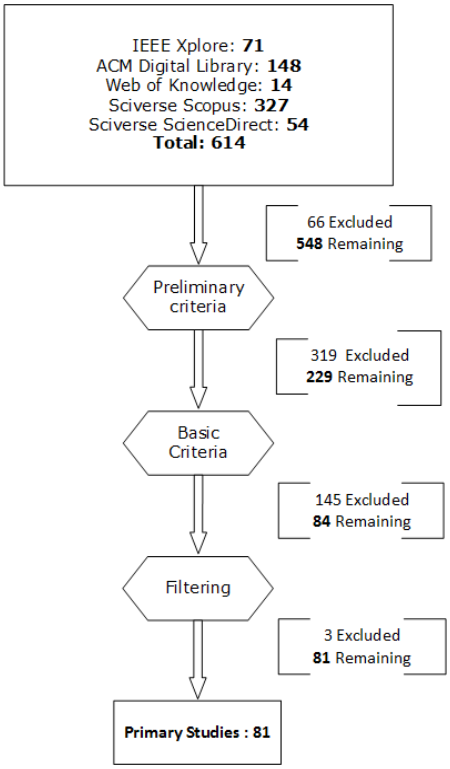


**Fig. 1.** Overview of the steps in the selection of the primary studies

### 3.3 Criteria

Preliminary criteria were applied on the titles of the publications returned by the search query. If the title of an article had not provided sufficient information to make an exclusion or inclusion decision, then the abstract of an article was read for further clarification. If a decision still could not be made, the article was included in the next step where the introduction section or the full text was read to make a decision.

- We excluded any article that is not relevant to improvement and evaluation of software quality.
- We excluded the articles which don't use the techniques of text mining.

**Basic Criteria.** Basic criteria were applied to further evaluate the relevance of articles to the research questions by reading titles and abstracts:

- We included the articles that analyzed natural language text and applied text analysis techniques to improve and evaluate software quality.
- We excluded the articles that applied the techniques of text mining on the source code or structured data.

Each article was labeled as relevant, irrelevant, or uncertain. Table 1 summarizes the results of applying the aforementioned inclusion/exclusion criteria.

**Table 1**. Results of applying the inclusion/exclusion criteria

| Category | No of Articles |
|---|---|
| Relevant | 65 |
| Uncertain | 34 |
| Irrelevant | 130 |

**Adaptive Reading.** Sometimes, it is difficult to decide about the relevance of a study only based on information available in the title and abstract. Therefore, we decided to apply the following two-step process of inclusion and exclusion to gather more information about the articles with "uncertain" label:

- Step 1: Read the introduction of the article to make the decision.
- Step 2: If the decision cannot yet be made, read the conclusion of the article.

After reading the introduction sections of articles, we added 19 articles from the "uncertain" list to our set of primary studies. Table 2 summarizes the results after applying the adaptive reading of articles with "uncertain" status.

**Table 2**. Results after resolving the uncertainty

| Category | No of Articles |
|---|---|
| Relevant | 84 |
| Irrelevant | 145 |

### 3.4 Filtering During Data Extraction

During data extraction, we read the full-texts of articles. This time, we filtered the articles based on the criteria laid out in previous sections (see sections 3.3 and 3.4) but after reading the full text of the articles. We found 3 journals articles that were the extension of conference publications, therefore, conference publications were considered as duplicates and removed from the list of primary studies.

### 3.5 Primary Studies

Following the methodology presented in Section 3.3 to 3.4, we found 81 relevant articles published in peer reviewed conferences, workshops, and journals.

## 4 Classification Scheme

We used systematic mapping as proposed by Peterson [12] to visualize the research corresponding to the research questions (RQ 1 to RQ 4). In our study, each dimension of a systematic map represents a research question. The classification schemes for each dimension of the systematic map are as follows:

**Improvement and Evaluation Aspect.** Similar to Peterson [12], we applied keywording of abstracts to identify the underlying high-level concepts dealt with in the analyzed primary studies. With regards to software quality improvement, we focused on the improvement of the software development process, such as software design, software testing, defect management, and so on. We identified the following improvement aspects in our primary studies: **a)** *software defect management*, **b)** *software testing*, **c)** *requirements engineering*, and **d)** *development process improvement*. These aspects cover primary studies that use textual data to improve respective activities of the software development process. With regards to software quality evaluation, we view software quality as related to the software as a final product of the development process. *Software quality model* is the only evaluation aspect we found. The software quality model aspect relates to primary studies that analyze textual data to evaluate quality characteristics of software based on the ISO/IEC 25010 quality standard.

**Data Source.** We identified 14 data sources in our primary studies as follows: **a)** *software reviews*, **b)** *app reviews*, **c)** *bug reports*, **d)** *discussion forums*, **e)** *API documentation*, **f)** *tweets*, **g)** *comments in source code*, **h)** *test cases*, **i)** *web service response,* **j)** *vulnerability database*, **k)** *project issues*, **l)** *requirement documents*, **m)** *use cases*, and **n)** *user manuals.* Mobile app marketplaces, such as Google Play Store and App-Store, have centralized and consistent mechanism for app distribution and feedback submission, therefore, we treated mobile app reviews as a separate category rather than merging it with the software reviews.

**Research Type.** The classification of research type utilizes the schema proposed by Wieringa et al. in [14]. Their scheme classifies primary studies into six categories: **a)** *solution proposals*, **b)** *validation research*, **b)** *evaluation research*, **d)** *philosophical papers*, **e)** *opinion papers*, and **f)** *experience papers*.

**Text Analysis Technique (TAT).** The classification of text analysis techniques (TATs) is adopted from the classification scheme presented by Gary Miner [15] with some minor modifications. Based on this classification scheme, we categorized TATs as follows: **a)** *classification (T1)* , **b)** *clustering (T2)*, **c)** *concept extraction (T3)*, **d)** *sentiment analysis (T4),* and **e)** *information extraction (T5)*. In [15], sentiment analysis (T4) is treated as a part of the concept extraction and natural language processing

techniques but we treated sentiment analysis as a separate category to get insight about primary studies that particularly uses the sentiment analysis techniques.

Once having the classification scheme in place, we sorted our primary studies into the scheme and started the actual data extraction. The complete set of extracted data per primary study, as well as the bibliographic information of all primary studies can be found in the Appendix provided at https://goo.gl/Q2I980.

## 5 Results and Analysis

Fig. 2 presents a map of research over improvement and evaluation aspect, distributed over the dimensions of data source and research type. Similarly, Fig. 3 presents a map of research over improvement and evaluation aspect, however, distributed over the dimensions of text analysis technique and research type.

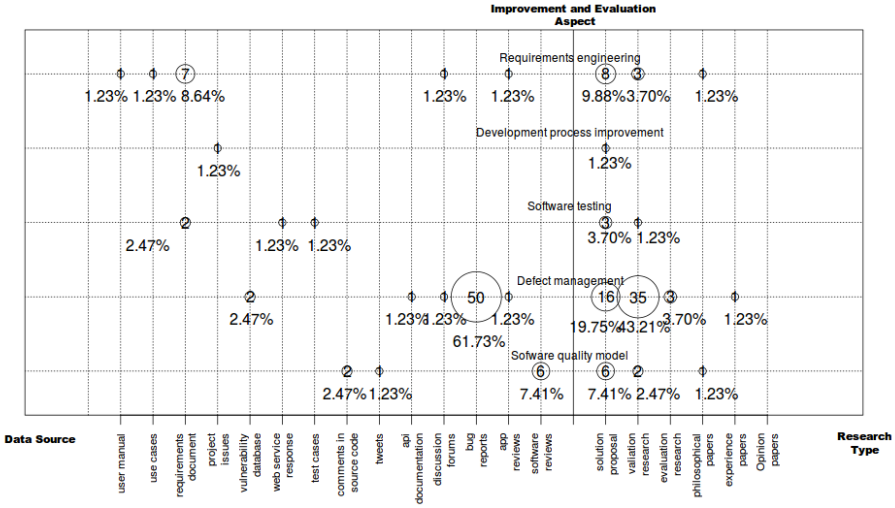


**Fig. 2.** Map of research aimed at improving and evaluating different aspects of software quality using text analysis techniques. Improvement and evaluation aspect dimension is on the y-axis. Data source dimension is on the left side of the x-axis and research type on the right side.

Overall, the defect management aspect is the most frequently mentioned among the primary studies (55 studies, 67.9% of all studies, see Fig. 2). The requirements engineering aspect is the second most frequently mentioned among the primary studies (12 studies, 14.8% of all studies). We found nine primary studies (11.1% of all studies) related to the aspect software quality model. Software testing aspect receives little research as we found just four studies related to this aspect. There is only one study which belongs to development process improvement aspect. We now answer each research question (RQ 1 to RQ 4) one by one.

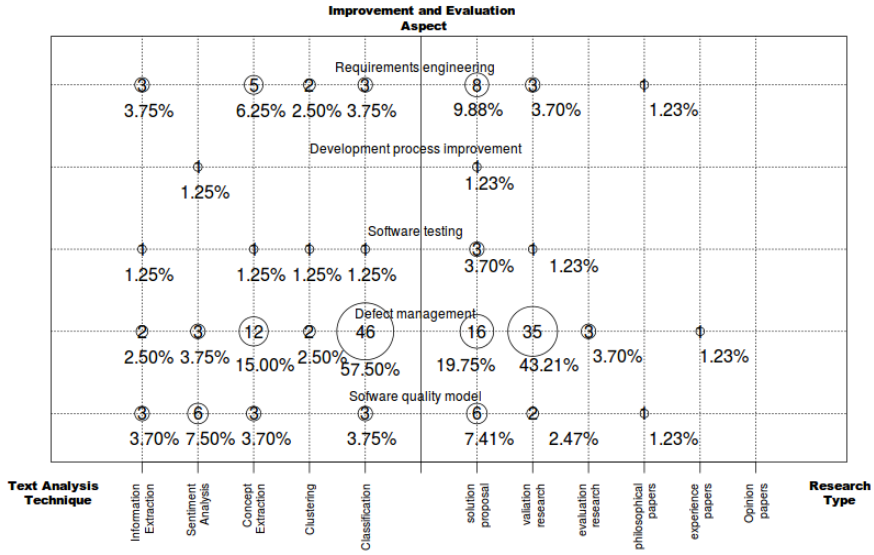


**Fig. 3.** Map of research aimed at improving and evaluating different aspects of software quality using text analysis techniques. Improvement and evaluation aspect dimension is on the y-axis. Text analysis technique dimension is on the left side of the x-axis and research type dimension is on the right side.

### 5.1 Data Sources (RQ 1)

We found each improvement and evaluation aspect used distinct set of data sources identified in the primary studies (see Section 4). Another observation is that overlapping of data sources among aspects is very rare (Fig. 2). The only data sources used by multiple aspects are app reviews, discussion forums and requirements document. We found very little diversity in data sources. By far the most frequently mentioned data source is bug report (50 studies) for defect management aspect. Similarly, in requirements engineering aspect, half of the studies used requirement documents as a data source (6 studies). Two thirds of the studies used software reviews as a data source (6 studies).

None of the studies used app reviews as a data source for software quality modal aspect. This seems to indicate that apps and the aspect software quality model have not been researched in combination. Given the ubiquity of apps we believe this is an omission that should be addressed in the future. For the research of apps under the aspect software quality model, app reviews would provide a perfect data source as they are by nature very similar to software reviews. The data sources used for aspect software testing are as follows: test cases, web service response and requirement documents. However, the bug and issues related information embedded in app and software reviews could also be exploited for test case prioritization. Development process improvement aspect used the project issues in GitHub as a data source.

### 5.2 Research Type (RQ 2)

We found no overlapping research types in our primary studies (see Fig. 2 or Fig. 3). The two most frequently mentioned research types are validation research (35 studies) and solution proposals (16 studies) for the defect management aspect. The amount of evaluation and experience papers is very small (4 studies) for the aspect defect management. For the aspect requirements engineering, the most frequently mentioned research type is solution proposal (8 studies), followed by validation research (3 studies). As for the aspect software quality modal the most frequently mentioned research type is solution proposal (6 studies). For aspect software testing, three studies are solution proposals, and one is a validation research study. Development process improvement aspect only has 1 solution proposal study.

### 5.3 Text Analysis Techniques (RQ 3)

The text analysis techniques (TATs) used for each improvement and evaluation aspect (see Fig.3) are presented as follow:

**Defect Management.** The two most frequently mentioned techniques are classification (46 studies) and concept extraction (12 studies). The commonly employed classification algorithms are as follow: Naïve Bayes, SVM, TF-IDF and logistic regression. We found it surprising that information extraction is the least explored technique, because we believe that significant gain in classification accuracy could be attained by using this technique for defect management aspect.

**Requirements Engineering.** Concept extraction is the frequently used text analysis technique (5 studies) for requirements engineering. The commonly employed concept extraction techniques are as follow: Latent Dirichlet Analysis (LDA), Latent Semantic Indexing (LSI) and Hierarchical Dirichlet Process (HDP).

**Software Quality Model.** The two most frequently used techniques of text analysis are sentiment analysis (6 studies) and concept extraction (4 studies).

**Software Testing.** Each text analysis technique is used except sentiment analysis. However, we believe sentiment analysis technique can be used to determine severity of bugs and then for bug prioritization according to their severity levels.

**Development Process Improvement.** Sentiment analysis technique is applied to determine the satisfaction level of the developers. However, there is a potential to leverage other text analysis techniques such as concept extraction, clustering, classification, and information extraction for improving software development processes in organizations.

### 5.4 Improvement and Evaluation (RQ 4)

In Section 4, we identified four software quality improvement aspects (defect management, requirements engineering, software testing, and development process improvement) and one software quality evaluation aspect (software quality model). In the following, first we discuss the activities improved within the respective improvement aspects followed by a discussion of the quality characteristics evaluated for the aspect software quality model.

**Defect Management.** The activities improved for the aspect defect management are as follows: bug classification (DM-A1), bug severity assignment (DM-A2), bug quality assessment (DM-A3), bug assignment (DM-A4), duplicate bugs (DB-A5) and bug localization (DM-A6). Table 3 presented the text analysis techniques which are used to improve these activities of defect management aspect. This is apparent from Table 3 that TATs are mostly used for the improvement of bug classification (22 studies) and bug severity assessment activities (14 studies). We found it surprising that there was no study applying sentiment analysis (T4) for bug severity assessment. We believe that by determining the strength of opinion words could improve bug severity assessment. The two most frequently used text analysis techniques are classification (T1) (46 studies) and concept extraction (T3) (12 studies).

**Table 3.** Activities improved in the defect management aspect

| TATs | Activities | | | | | |
|---|---|---|---|---|---|---|
| | DM-A1 | DM-A2 | DM-A3 | DM-A4 | DM-A5 | DM-A6 |
| T1 | (P2,P5,P15,P16,P19,P20-P24,P26,P28, P29-P31,P33-P35,P73,P81) | (P4,P36-P48) | (P49) | (P25,P50-P52,P54) | (P55,P57) | (P59-P61) |
| T2 | (P18,P32) | - | - | - | | - |
| T3 | (P17,P19,P26,P27,P32,P34) | - | - | (P25,P53) | (P55) | (P59,P62) |
| T4 | (P28,P29) | - | - | - | | (P62) |
| T5 | - | - | - | - | (P57) | (P58) |

**Requirements Engineering.** The RE activities which are improved by the primary studies using TATs are: functional requirements (RQ-A1), non-functional requirements (RQ-A2), requirements evolution (RQ-A3) and software verification (RE-A4). A compact summery of TATs used for the improvement of these activities is presented in Table 4. The activity which is frequently attempted to improve using TATs is non-functional requirements (7 studies). The most commonly used technique of text analysis is conception extraction (T3). None of the study used sentiment analysis technique for this aspect.

**Table 4.** Activities improved in the requirements engineering aspect

| TATs | Activities | | | |
|---|---|---|---|---|
| | **RE-A1** | **RE-A2** | **RE-A3** | **RE-A4** |
| T1 | (P69) | (P75,P80) | - | - |
| T2 | (P70) | (P70) | (P78) | - |
| T3 | (P71) | (P71,P74,P76,P77) | (P3) | - |
| T4 | - | - | - | - |
| T5 | (P70) | (P70,P74) | - | (P79) |

**Software Testing.** The software testing activities improved in the primary studies are as follow: static black-box test-case prioritization (ST-A1), robustness testing (ST-A2) , test case generation (ST-A3), and test case prioritization (ST-A4). Table 5 presented the text analysis techniques used for the improvement of these activities in the primary studies.

**Table 5.** Activities improved in the software testing aspect

| TATs | Activities | | | |
|---|---|---|---|---|
| | **ST-A1** | **ST-A2** | **ST-A3** | **ST-A4** |
| T1 | - | (P65) | - | - |
| T2 | - | - | - | (P67) |
| T3 | (P64) | - | - | - |
| T4 | - | - | - | - |
| T5 | - | - | (P66) | - |

**Development Process Improvement.** The primary study (P68) uses sentiment analysis technique for the improvement of development process.

**Software Quality Model.** Under software quality model research, we classified the research into the following sub-aspects: **a)** quality in use model and **b)** software product quality model. Related to 'product quality model' we identified 5 relevant studies, and related to 'quality in use model' we identified 4 relevant studies. Each of the two sub-models defines a specific set of quality characteristics as shown in Table 6 (quality in use model, capturing the user's perspective) and Table 7 (product quality model, capturing the developer's perspective). Both tables show how often individual quality characteristics of each sub-model are evaluated in the related primary studies. It is apparent from Table 6 that all quality characteristics defined in the quality in use model are equally well covered by the related primary studies. In Table 7 we see that all primary studies focused on individual or small set of quality characteristics defined in the product quality model. Since we found only five primary studies this yields low coverage of quality characteristics. However quality characteristic operability is addressed in 4 out of 5 primary studies.

**Table 6.** Quality characteristics evaluted the quality in use model aspect

| TATs | Quality characteristics | | | | |
|---|---|---|---|---|---|
| | Effectiveness | Efficiency | Satisfaction | Safety | Usability |
| T1 | (P9) | (P9) | (P9) | (P9) | - |
| T2 | - | - | - | - | - |
| T3 | (P6,P8) | (P6,P8) | (P6) | (P6,P8) | (P6) |
| T4 | (P6,P8,P9) | (P6,P8,P9) | (P6,P9) | (P6,P8,P9) | (P6,P7,P9) |
| T5 | - | - | - | - | - |

**Table 7.** Quality characteristics evaluted in the software product quality model aspect

| TATs | Quality characteristics | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Functional | Reliability | Performance | Operability | Security | Compatibility | Maintainability | Portability |
| T1 | - | (P10) | - | (P12) | (P12) | - | - | - |
| T2 | - | - | - | - | - | - | - | - |
| T3 | - | - | - | - | - | - | - | - |
| T4 | - | (P10) | - | (P12) | (P12) | - | - | - |
| T5 | - | - | - | (P13,P14) | - | - | (P11) | - |

## 6 Conclusions and Future Prospects

Our mapping study found 81 relevant publications on research using text analysis techniques to improve and evaluate software quality. The improvement and evaluation aspects addressed are defect management, requirement engineering, software quality model, software testing, and development process improvement. Our results show that the defect management aspect is in the focus of the research. For the aspect defect management, text analysis is used for the improvement of bug classification and bug severity assignment activities. Other aspects which also gain some attention are: requirements engineering and software quality model. Research in the quality in use model sub-aspect focuses on evaluation of usability and operability quality characteristics, whereas little attention is given to quality characteristics, such as functional, performance, security, portability, compatibility, and maintainability of product quality model sub-aspect. The commonly utilized data sources are: bug report in defect management aspect, requirement documents in requirements engineering aspect, and software reviews in software quality model aspect.

Published research is centered on solution proposals and validation research. All improvement and evaluation aspects focuses more on solution proposals except defect management aspect in which validation research is appeared more than solution proposals. Research in each improvement and evaluation aspect focuses on different text

analysis technique. For instance, bug management aspect mostly used classification technique, requirements engineering aspect mostly used concept extraction technique, and software quality model aspect frequently used sentiment analysis technique. Following are some pointers for future research to improve and evaluate software quality using text analysis techniques:

- Software reviews have been utilized to evaluate software quality characteristics. However, analyzing app reviews to assess quality factors of a mobile app is an interesting area which is required to be explored.
- There is a need to analyze software reviews as a data source to understand users' evolving requirements and planning software next releases.
- Another area of future research could be to identify bugs and their severity by analyzing reviews and use this information for test case prioritization.
- Software engineering research has focused on single repositories to evaluate software quality. Evaluating the software quality through "multi-repositories fusion" could become an intriguing research.
- Given their low coverage by published research, there seems to be a need to evaluate the following software quality characteristics of software product quality modal: portability, functionality, compatibility, performance, security, and maintainability, from different data sources, particularly by utilizing software and app reviews.
- Another exciting research area could be the extraction of "actionable information" helping developers to understand whether degradation of certain software quality characteristic caused by insufficient requirements engineering or insufficient testing.

**Acknowledgement.** This work is supported by the institutional research grant IUT20-55 of the Estonian Research Council.

## References

[1] H. Müller and N. Villegas, "Runtime Evolution of Highly Dynamic Software," in *Evolving Software Systems*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 229–264.

[2] *The Art and Science of Analyzing Software Data*. Elsevier Science, 2015.

[3] T. Menzies and T. Zimmermann, "Software analytics: So what?," *IEEE Softw.*, vol. 30, pp. 31–37, 2013.

[4] L. V. G. Carreno and K. Winbladh, "Analysis of user comments: An approach for software requirements evolution," *Proc. - Int. Conf. Softw. Eng.*, pp. 582–591, 2013.

[5] G. Bavota, R. Oliveto, M. Di Penta, D. Poshyvanyk, and A. De Lucia, "User Reviews Matter ! Tracking Crowdsourced Reviews to Support Evolution of Successful Apps," pp. 1–10.

[6] D. Pagano and W. Maalej, "User feedback in the appstore: An empirical study," *21st IEEE Int. Requir. Eng. Conf.*, pp. 125–134, 2013.

[7] S. Ouhbi, A. Idri, J. L. Fernandez Aleman, and A. Toval, "Evaluating Software Product Quality: A Systematic Mapping Study," *Softw. Meas. Int. Conf. Softw. Process Prod. Meas. (IWSM-MENSURA), 2014 Jt. Conf. Int. Work.*, no. January, pp. 141–151, 2014.

[8] A. T. Misirli and A. B. Bener, "A mapping study on bayesian networks for software

quality prediction," *Proc. 3rd Int. Work. Realiz. Artif. Intell. Synerg. Softw. Eng. - RAISE 2014*, pp. 7–11, 2014.

[9] J. Vargas-Enriquez, L. Garcia-Mundo, M. Genero, and M. Piattini, "A Systematic Mapping Study on Gamified Software Quality," in *2015 7th International Conference on Games and Virtual Worlds for Serious Applications (VS-Games)*, 2015, pp. 1–8.

[10] A. Casamayor, D. Godoy, and M. Campo, "Mining textual requirements to assist architectural software design: a state of the art review," *Artif. Intell. Rev.*, vol. 38, no. 3, pp. 173–191, May 2011.

[11] M. Borg, P. Runeson, and A. Ardö, "Recovering from a decade: a systematic mapping of information retrieval approaches to software traceability," *Empir. Softw. Eng.*, vol. 19, no. 6, pp. 1565–1616, May 2013.

[12] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic Mapping Studies in Software Engineering," *12th Int. Conf. Eval. Assess. Softw. Eng.*, vol. 17, pp. 1–10, 2007.

[13] T. Menzies, L. Minku, and F. Peters, "The Art and Science of Analyzing Software Data; Quantitative Methods," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, 2015, vol. 2, pp. 959–960.

[14] R. Wieringa, N. Maiden, N. Mead, and C. Rolland, "Requirements engineering paper classification and evaluation criteria: A proposal and a discussion," *Requir. Eng.*, vol. 11, no. 1, pp. 102–107, 2006.

[15] G. Miner, *Practical text mining and statistical analysis for non-structured text data applications*. 2012.