Software Quality Journal (2025) 33:22
https://doi.org/10.1007/s11219-025-09720-9

RESEARCH

Supporting the identification of prevalent quality issues in code changes by analyzing reviewers' feedback

Umar Iftikhar[1] · Jürgen Börstler[1] · Nauman Bin Ali[1] · Oliver Kopp[2]

Accepted: 5 April 2025

Abstract
Context: Code reviewers provide valuable feedback during the code review. Identifying common issues described in the reviewers' feedback can provide input for devising context-specific software development improvements. However, the use of reviewer feedback for this purpose is currently less explored. Objective: In this study, we assess how automation can derive more interpretable and informative themes in reviewers' feedback and whether these themes help to identify recurring quality-related issues in code changes. Method: We conducted a participatory case study using the JabRef system to analyze reviewers' feedback on merged and abandoned code changes. We used two promising topic modeling methods (GSDMM and BERTopic) to identify themes in 5,560 code review comments. The resulting themes were analyzed and named by a domain expert from JabRef. Results: The domain expert considered the identified themes from the two topic models to represent quality-related issues. Different quality issues are pointed out in code reviews for merged and abandoned code changes. While BERTopic provides higher objective coherence, the domain expert considered themes from short-text topic modeling more informative and easy to interpret than BERTopic-based topic modeling. Conclusions: The identified prevalent code quality issues aim to address the maintainability-focused issues. The analysis of code review comments can enhance the current practices for JabRef by improving the guidelines for new developers and focusing discussions in the developer forums. The topic model choice impacts the interpretability of the generated themes, and a higher coherence (based on objective measures) of generated topics did not lead to improved interpretability by a domain expert.

Keywords Modern code review · Software quality improvement · Natural language processing · Open-source systems

✉ Umar Iftikhar
umar.iftikhar@bth.se

Jürgen Börstler
jurgen.borstler@bth.se

Nauman Bin Ali
nauman.ali@bth.se

Oliver Kopp
oliver.kopp@iste.uni-stuttgart.de

[1] Blekinge Institute of Technology, Valhallavägen 1, SE-37179 Karlskrona, Blekinge, Sweden

[2] University of Stuttgart, Universitätsstr. 38, 70593 Stuttgart, Baden-Württemberg, Germany

Published online: 23 April 2025

1 Introduction

In modern code review, experienced developers and architects review code changes and give feedback as code review comments (CRCs) to improve source code quality (McIntosh et al., 2014; Bavota and Russo, 2015; McConnell, 2004; Unterkalmsteiner et al., 2024). The expert's feedback on the submitted code change is cognizant of important factors such as the system architecture (Paixao et al., 2019), domain (Bacchelli and Bird, 2013), technology, the team, and the organization (Bosu et al., 2017). Such feedback is often beyond what static code analyzers can provide today. Moreover, static code analyzers have been criticized for reporting relatively high numbers of false positives (Johnson et al., 2013; Vassallo et al., 2020) or trivial issues. The feedback provided in CRCs is typically only utilized once when developers implement a specific corrective action. As reviewers may point out similar issues to the same or different developers, identifying such prevalent issues can be helpful, e.g., to propose preventive measures and find systematic improvements (Ochodek et al., 2022; Iftikhar et al., 2023). Such preventive measures may include more precise guidelines for project contributors, focused training on a particular technology, or reconfiguring the static analysis tool (Gunawardena et al., 2023), thus helping avoid similar issues in the future. The literature shows that analyzing the communication between developers can lead to the improvement of software documentation (Henß et al., 2012; Souza et al., 2019), identification of software maintenance tasks (Sun et al., 2015), and bug localization (Ahasanuzzaman et al., 2020).

It is infeasible to aggregate quality issues discussed in individual CRCs without qualitatively analyzing individual comments. The large number of review comments (Bosu et al., 2017) makes their manual analysis impractical beyond research studies. Practitioners, therefore, often have to rely on their subjective judgment of what they perceive as prevalent code quality issues, and thus what, in their opinion, are the required improvements. Therefore, automated classification approaches are needed to more objectively assess prevalent code quality issues discussed in large code review repositories. Automated code review comment classification approaches based on machine-learning methods have recently been shown to provide promising results (Arafat and Shamma, 2020; Ochodek et al., 2022; Wen et al., 2022; Iftikhar et al., 2023).

In this study, we investigate how to support practitioners in identifying and profiling prevalent quality issues, as pointed out in code review comments. We used the participatory case study design with an iterative evaluation approach suggested by Kindon et al. (2007) and Baca and Petersen (2013). We performed two iterations using the code review comment analysis approach by Iftikhar et al. (2023). In the first iteration, we used Iftikhar et al.'s (Iftikhar et al., 2023) approach as is, while in the second iteration, we evaluated an alternate topic modeling method based on BERTopic (Grootendorst, 2020). The second iteration was required to further explore improvements in the results of the first iteration. In both iterations, with the help of a domain expert, we evaluated the degree of ease of naming the topics produced (Silva et al., 2024). The approach used for each iteration is summarized in Fig. 1.

To demonstrate the utility of the approach proposed in (Iftikhar et al., 2023), we explore whether it helps identify code quality issues discussed in code review comments for merged or abandoned code changes. The main contribution of this study is:

1. Evaluation by domain expert of two promising approaches to derive interpretable and meaningful themes from code review comments
2. Exploring the effectiveness of large language models to provide appropriate labels to generated topics.

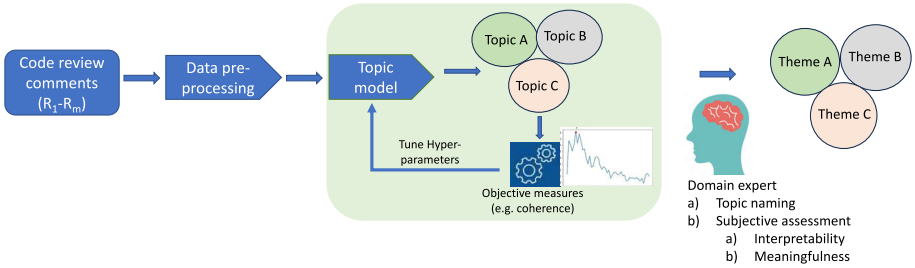

Fig. 1 Overview of the approach utilizing objective and subjective measures to assess the quality of the generated topics in each iteration

We organized the remainder of the paper as follows. We discuss related work in Section 2, and the methodology is covered in Section 3. Section 4 presents our results, followed by a discussion of the results in Section 5 and threats to validity in Section 6. Conclusions can be found in Section 7.

2 Related work

This section summarizes related work on three themes: (1) manual categorization of CRCs, (2) automated analysis of CRCs, and (3) investigations of factors that impact the outcome of submitted code changes.

2.1 Manual categorization of CRCs

Among the existing studies that manually categorized issues in code reviews, Mäntylä and Lassenius (2009) manually analyzed nine industrial and 23 academic systems to categorize defects identified during code review discussions. They further classified the identified defects and observed that 75% of the defects relate to evolvability while 25% of the defects relate to functionality. Beller et al. (2014) adopted a similar approach to analyze 1400 code changes from two open-source systems to investigate issues fixed during code review. Their results corroborated (Mäntylä and Lassenius, 2009), where approximately 75% of the fixed issues were related to maintainability and 20% of the fixed issues were related to functionality. Gunawardena et al. (2023) manually analyzed 417 CRCs to propose a fine-grained taxonomy of 117 defects discussed in the CRCs. They further mapped the proposed defects taxonomy to static analysis tools, where 38% identified defects could be resolved using static analysis tools, thus demonstrating that categorization of issues in CRCs may have implications for practice.

While these studies demonstrate that categorizing CRCs can lead to identifying quality-related issues, which can assist in software improvement tasks, the manual, effort-intensive approach limits their application to new datasets.

2.2 Automation to support CRCs analysis

Given the large code review repositories in practice, existing studies have approached automated categorizing of CRCs with different methodologies. Arafat and Shamma (2020)

categorized and predicted topics in CRCs from six closed-source systems using supervised machine learning algorithms using a manually labeled dataset. They achieved 63% accuracy with the Support Vector Machine (SVM) method. Ochodek et al. (2022) classified 2,672 CRCs from three open-source systems using the Bidirectional Encoder Representation from Transformers (BERT) (Devlin et al., 2018) language model. They achieved an average accuracy of over 80% compared to manually classified CRCs. Other studies (Li et al., 2017; Fregnan et al., 2022; Turzo et al., 2023) have also considered using supervised machine learning to train models using multiple features, e.g., code snippets and source code metrics, in addition to code review comments to categorize code review comments and code changes automatically. The studies have promising results but require labeled datasets, thus limiting the applicability of their approach due to the lack of labeled data in real-world settings.

To investigate how community and personal feedback trends evolve as the community matures, Wen et al. (2022) utilized Latent Dirichlet Allocation (LDA) on CRCs from one open-source system, Nova, and one closed-sourced system. Their results show that as reviewers accumulate experience, the feedback provided to code changes is more context-specific and technical.

Iftikhar et al. (2023) extended the work of Wen et al. (2022) and evaluated several potential improvements in the design by Wen et al. (2022). Among the proposed improvements, they found that short-text topic modeling (Yin and Wang, 2014) leads to more stable topics than traditional topic modeling. Similarly, among the alternative methods for selecting the number of topics, their two-stage topic selection approach slightly improved topic stability over the single-stage topic selection used by Wen et al. (2022). However, both studies (Iftikhar et al., 2023; Wen et al., 2022) do not demonstrate how to derive and profile code quality issues using common themes identified from CRCs. We summarize the main findings from these studies in Table 1. In this study, we address this gap by involving a domain expert from JabRef.

Silva et al. (Silva et al., 2024) used four short text topic models and human-centric metrics to evaluate how comprehensible the topics generated from developer communication on an open-source instant messaging platform. In their study, while no single short text topic model performed best on all datasets considered, the Gibbs Sampling Dirichlet Multinomial Mixture Model (GSDMM) (Yin and Wang, 2014) led to more comprehensible topics for the participants involved. Their study also observed that objective coherence metrics and human-centric metrics, e.g., subjective evaluation of comprehension of the generated topics by developers, did not align. However, their study did not consider BERTopic and only considered four short-text topic models.

BERTopic (Grootendorst, 2020) is a modified BERT-based topic model that uses pre-trained word embeddings and a clustering approach to leverage semantic relationships between words in a document. Recently, Udupa et al. Udupa et al. (2022) demonstrated that BERTopic (Grootendorst, 2020), compared to GSDMM (Yin and Wang, 2014), provides more coherent topics for short-text data. However, the performance of BERTopic is yet to be evaluated in the analysis of code review comments.

2.3 Investigations of factors behind abandoned and merged code changes

Existing studies have investigated the reasons for the rejection of code submissions. Gottigundala et al. (2021) reported that the reasons for the rejection of pull requests include implementing unnecessary functionality, conflicting pull requests, reattempted pull requests, and inactivity in open pull requests. Kononenko et al. (2018) reported that pull requests that are large and do not address a single purpose are likely not to be merged. They further observed

Table 1 Overview of existing studies

Study	Approach	Defined categories	Result
Arafat and Shamma (2020)	Logistic Regression, k-Nearest Neighbors (KNN), Bernoulli Naive Bayes, Multinomial Naive Bayes (MNB), Support Vector Machine, Stochastic Gradient Descent (SGD)	Documentation, Visual Representation, Organization, Solution Approach, Resource, Validation, Logical, Synchronization, Supporting Library, Defect, API Calls, False Positive, Others	Accuracy 63.89%
Ochodek et al. (2022)	BERT	Code_design, Code_style, Code_naming, Code_logic, Code_io, Code_data, Code_API, Code_doc, Compatibility, Rule_def. Config_commit_patch_review, Config_building_installling	Matheews Correlation Coefficient 80%
Li et al. (2017)	SVM based Two-Stage Hybrid Classification	Correctness, Decision, Management, Interaction	F-Measure 82%
Fregnan et al. (2022)	Random Foreset, J48, Naive Bayes	Documentation, Visual Representation, Organization, Solution Approach, Resource, Validation, Logical, Synchronization, Supporting Library, Defect, API Calls, False Positive, Others	AUC-ROC 0.91
Turzo et al. (2023)	BERT, CodeBERT	Functinoal, Refactoring, Documentation, Discussion, False Positives	MCC 0.57
Wen et al. (2022)	LDA	Context Specific, Exception handling, Language Specific, Design, Code review process, Code styling, Unit testing	-
Iftikhar et al. (2023)	GSDMM	Inheritance, Concurrency, Guidelines, Component level logic	-

that the experience of a pull request author is significantly linked with the merge decision for a pull request. Papadakis et al. (2020) found that source code management issues, lack of understanding of project functionality, and poor understanding of reviewer expectations and project guidelines were among the reasons for the rejection of pull requests.

Wang et al. (2019) identified 12 reasons for the abandonment of code changes. Duplicate code changes, i.e., similar to other code changes, and code changes with a lack of reviewer feedback, were among the frequent reasons for abandoned code changes. Researchers have explored various factors influencing the likelihood of a code change being merged or abandoned. However, to the best of our knowledge, no prior work has analyzed CRCs to identify themes in abandoned and merged code changes that might help developers identify recurring quality-related issues.

As discussed in Section 1, code review comments are potentially a very relevant source of information to mine insights regarding code quality issues. Thus, in this study, we explore how CRCs can be analyzed to identify and develop a better profile of quality issues identified in merged or abandoned code changes.

3 Methodology

In this case study, we pose and answer the following research questions:

1. **RQ1:** Which topic modeling method derives more interpretable and meaningful themes from CRCs to identify recurring code quality issues?
2. **RQ2:** To what degree can Large-Language Models support topic naming of themes from CRCs?

The first research question aims to subjectively evaluate the automation method in the two iterations in terms of the ease of interpreting the generated topics by the domain expert. Furthermore, we also subjectively assess the assigned themes in terms of their meaningfulness and degree of information from the perspective of the domain expert. We relied on a domain expert's perceptions regarding the interpretability and meaningfulness of the derived themes. Their extensive system knowledge is required to evaluate whether the themes are meaningful and indicate quality issues. The second research question explores the effectiveness of Large Language Models in supporting the naming of the generated topics from the two iterations. Evaluating LLMs' effectiveness aids in evaluating whether we can fully automate the approach and reduce the dependence on a human expert, which is usually a resource-intensive activity.

The participatory case study method is motivated by the need to incorporate the feedback of domain experts in the iterative development and improvement of research-based interventions (Baca and Petersen, 2013). In this study, the steps of each iteration include implementing the topic model approach and collecting feedback from the domain expert to capture its performance (Fig. 1 shows the steps in which the domain expert participated). In the first iteration, we used (Iftikhar et al., 2023)'s approach to automatically identify topics in code review comments. The approach suggests the use of GSDMM for topic modeling. A domain expert analyzed and evaluated the themes derived through this approach. To address the feedback of the domain expert and to improve the coherence, as measured using objective coherence measures, of the topics in the second iteration, we explored the use of BERTopic instead of GSDMM for topic modeling (Grootendorst, 2020). The resulting themes in the second iteration were also analyzed by the same domain expert.

Table 2 Overview of data from JabRef (PR=pull request)

PR status	Total number of PRs	Number of PRs with CRCs	Total CRCs	Average CRC length in words
Abandoned	717	38	535	26.5
Merged	4,862	388	5,025	22.8
Total	5,579	426	5,560	23.2

In this study, we analyzed code review comments for both abandoned and merged code changes to demonstrate the effectiveness of our approach in profiling code quality issues. Based on the different reasons for abandoning or merging code changes described in the existing literature (Kononenko et al., 2018; Papadakis et al., 2020), we expect differences in the reviewer feedback for the two classes of code changes. The topics identified using our approach and named by the expert are expected to help profile the code quality issues the reviewers are pointing out for merged and abandoned code changes.

3.1 Case project and domain expert

We choose to use JabRef[1] as the case project. JabRef is an open-source, cross-platform, citations and reference management tool developed in Java (Kopp et al., 2023). It aims to help students, academics, and researchers to collect and maintain bibliographic information. It covers more than 15 reference formats and supports 23 languages. At the time of the study, JabRef version 5.12 was released. It is maintained by a core team of researchers and students with 581 active contributors[2]. At least two reviewers, belonging to the core team members and module experts, can review each submitted code change[3]. A proposed code change may undergo several iterations of code review before merging until the code reviewers are satisfied with the code quality. Code changes to specific modules, e.g., JavaFX and BibTeX, are reviewed by module experts, while anyone in the core team can review the generic code changes. The quality of the JabRef source code has previously been analyzed in the literature (Nuñez-Varela et al., 2017; Olsson et al., 2017).

Another reason for selecting JabRef was the availability of a long-term maintainer and a domain expert to assign representative names to identified topics. The domain expert was not part of the research team that conceived, designed, or analyzed the results. The research team collected, analyzed, performed member checking, and completed the first draft of the paper before inviting the domain expert to be a co-author. In the writing phase, the domain expert contributed mainly with a deeper contextual understanding of JabRef, checking the results and reflections in the paper and thoroughly reviewing several versions of the paper.

3.2 Datasets

Table 2 shows an overview of the dataset. We used the REST API provided for GitHub[4] to extract CRCs and code change outcome status. Since we are only interested in code changes

[1] https://www.jabref.org/

[2] https://github.com/jabref/jabref/

[3] https://devdocs.jabref.org/teaching.html

[4] https://docs.github.com/en/rest/overview

that are either abandoned or merged, we did not consider CRCs from code changes that are still open or under discussion. We extracted CRCs from May 2014 till September 2023 as CRCs from earlier were unavailable. We used the *pull request number* to relate CRCs and code change status.

We replicated the pre-processing steps in previous studies (Iftikhar et al., 2023). We converted all CRCs to lowercase and removed all brackets, punctuations, URLs, and words containing numbers. We removed stop words using the standard pre-processing library in the Gensim natural language processing toolkit[5]. We further removed null strings and lemmatized all CRCs to create a document entry for each code review comment, forming two lists of documents, one from abandoned changes and one from merged changes. We preferred lemmatization instead of stemming as lemmatization provides higher objective coherence (Silva et al., 2021).

After data extraction and pre-processing of the CRCs, we got 5,560 CRCs from 426 code changes belonging to three major releases of JabRef. We did not consider code changes without CRCs for the analysis and issue comments. We did not consider 60 CRCs from version 2 as the release only contained CRCs from merged changes. We also removed 124 CRCs with zero length after pre-processing. The extracted CRCs are provided in the replication package online[6].

3.3 First iteration: with GSDMM

3.3.1 Short text topic models

Short text topic models (STTM) have been demonstrated by Qiang et al. (2020) to have superior performance than Latent Dirichlet Allocation (LDA) for short texts in terms of classification accuracy and purity (Yin and Wang, 2014). Short text topic models can handle the sparse word co-occurrence pattern typical of short documents (Qiang et al., 2020; Silva et al., 2024). CRCs are short pieces of text (Li et al., 2017). We selected the Gibbs Sampling-based Dirichlet Multinomial Mixture model (GSDMM) (Yin and Wang, 2014) implementation by Qiang et al. (2020). GSDMM has also improved average topic stability compared to LDA (Iftikhar et al., 2023). Since we are interested in analyzing CRCs from abandoned and merged changes, we generate separate topic models for CRCs from abandoned and merged changes.

3.3.2 Parameter selection

The performance of topic classification depends on the choice of hyperparameters, topic-to-document probability *(alpha)*, word-to-topic probability *(beta)*, and number of topics *(N)* (Iftikhar et al., 2023). Biggers et al. (2014) suggest that lower values for the hyperparameters lead to a more decisive model with less overlap among generated topics.

Topic stability, an adapted measure of cross-run similarity of topics (Agrawal et al., 2018), is defined as the median number of word-terms occurrences in all considered runs for a given topic number while varying the hyperparameters *alpha* and *beta* between 0 and 1 for each considered run. While (Panichella, 2021) observes that no fitness measure consistently leads to superior quality topics in their dataset, we chose topic stability as it has been used in existing studies (Wen et al., 2022; Iftikhar et al., 2023).

[5] https://www.nltk.org/

[6] https://doi.org/10.5281/zenodo.10408930

In the first stage, we considered N=[5..55] (in steps of five) for the number of topics to analyze topic stability. We trained five GSDMM models for each dataset, sorted in a different order for each run, with the choice of hyperparameters varied for each run. We used the ten top words from five runs of GSDMM models to identify the most stable choice of N in this first stage. In the second stage, we iterate in steps of one in the neighborhood of N from the first stage to select the most stable topic. Next, to choose the most suitable values for hyperparameters, we also calculate the objective coherence for the five combinations of *alpha* and *beta* used in the topic stability stages and select the combination that provides the highest objective coherence.

3.4 Second iteration: with BERtopic

BERTopic[7] has been observed to achieve higher coherence, measured using objective coherence measures, than GSDMM on short text classification tasks (Udupa et al., 2022) by incorporating transformer-based clustering and categorical Term Frequency-Inverse Document Frequency (c-TF-IDF) to generate dense clusters leading to interpretable topics (Grootendorst, 2020). BERTopic's design is versatile and provides several methods for dimensionality reduction, clustering, and topic representation. For each of these methods, multiple parameters can be optimized to further improve the objective coherence of the generated topics. However, finding optimal parameters can be challenging. In this regard, we adopted the suggestions provided in the existing literature (Martin Borčin, 2024; Udupa et al., 2022; Schneider et al., 2023).

Borčin et al. (Martin Borčin, 2024) evaluated several pre-trained embedding models, and "all-mpnet-base-v2" provided the highest objective coherence in the considered datasets.

BERTopic supports several dimensionality reduction techniques, including Uniform Manifold Approximation and Projection for Dimension Reduction (UMAP), Principal Component Analysis (PCA), and truncated Singular Value Decomposition (SVD). We used UMAP as suggested by McInnes et al. (2018) for its ability to retain more local and global properties in lower dimensions. We used the default values for the UMAP parameters.

Multiple clustering approaches are supported by BERTopic. We selected Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) (Udupa et al., 2022; Schneider et al., 2023). We used TopicTuner API [8] to select the optimal parameter range for the HDBSCAN (Schneider et al., 2023), specifically, *min cluster size* and *min sample size*, which control the cluster size and the number of outliers. Martin Borčin (2024) observed highest objective coherence while retaining approximately 80% of the input documents. Thus, we did not consider *min sample size* that led to more than 20% CRCs as outliers. Similarly, we did not consider *min cluster size* that led to more than 55 topics to ensure a fair comparison with the approach in the first iteration.

To generate topic representations, we used CounterVectorizer as suggested by Udupa et al. (2022). We used the default values for *max features*, which are the maximum features considered for topic representation. To select an appropriate range of N-grams to consider for CounterVectorizer, we calculated the frequency of the different lengths of the N-grams present in each dataset using the *Ngrams* package available in the NLTK library. We considered all unigrams (Ngram=1) up to the largest N-gram that appeared more than five times in the dataset. To improve the accuracy of the topic representations, we used the BM-25 weighing suggested by Martin Borčin (2024) to reduce the number of stopwords used to formulate topic

[7] https://github.com/MaartenGr/BERTopic

[8] https://github.com/drob-xx/TopicTuner

representations. To further fine-tune topic representation, we used the KeyBERTInspired sub-module, as described in (Iftikhar, 2024). However, not all options were evaluated by the domain expert. Only the most promising combination of parameters that provided high objective coherence was selected for evaluation by the domain expert.

3.5 Objective coherence measures

Röder et al. (2015) systematically evaluated several objective coherence measures, including C_{npmi} (Bouma, 2009), C_v (Röder et al., 2015), and U_{mass} (Aletras and Stevenson, 2013) for their correlations with human judgment on six generic datasets. We chose C_v in our study as it scored the highest average correlation with human judgment in their study (Röder et al., 2015). The C_v objective coherence measure utilizes indirect cosine measure with Normalized Pointwise Mutual Information (NPMI) over a boolean sliding window (Röder et al., 2015).

$$C_V = \frac{1}{|P|} \sum_{(w_i, w_j) \in P} \frac{\vec{v}(w_i) \cdot \vec{v}(w_j)}{\|\vec{v}(w_i)\| \|\vec{v}(w_j)\|}$$

where P is the set of word pairs considered in the topic, $\vec{v}(w_i)$ is the vector representation of the word w_i.

3.6 Data collection

We designed a structured questionnaire[9] that was shared with the domain expert for data interpretation tasks. The structured questionnaire consisted of the top 20 unprocessed CRCs belonging to each topic, the top 20 terms representing the topic (Wen et al., 2022), and meta information such as the PR number, PR URL, and PR title, among other important information that may help to assign a suitable theme to CRCs.

In addition to the structured questionnaire, we provided a separate document containing the study aims and instructions for steps to be performed during the interpretation task. We used the same document to collect the overall reflections on the ease of naming topics, reflections on the naming process, difficulties faced when interpreting topic evolution, and potential implications of the identified themes for process improvements within the project. The same structured questionnaire was used to collect data at the end of both iterations.

3.7 Topic naming & subjective assessment

Current naming approaches can be categorized as manual, automated, and a combination of manual and automated steps (Silva et al., 2021). Manual topic-naming has been used in existing studies (Wen et al., 2022; Haque et al., 2020; Han et al., 2020). As depicted in Fig. 1, after tuning the topic model parameters to achieve high objective coherence, we (a) asked the domain expert to suggest a representative name for each topic in each iteration, (b) to provide their subjective assessment and perception regarding the ease of naming the topics or the degree of interpretability, and (c) to provide their perception regarding the utility of the themes and their meaningfulness in the context of JabRef. For topic naming, the domain expert used 20 unprocessed CRCs and top 20 terms representing a topic.

[9] online_link

Silva et al. (2021) observed that more coherent and cohesive topics are easier to label. While (Silva et al., 2024) utilized the number of topics named and the consistency of the named themes by different practitioners, we adopted the ease of naming the identified theme as the criteria to judge the quality of the topics produced.

3.8 Automated topic naming using LLMs

To explore whether we can automate the topic naming process, we prompted the publically available Large Language Model, ChatGPT (OpenAI, 2023), to suggest a representative name by utilizing each topic's top 20 CRCs along with 20 top terms and compared the results to the names that the domain expert manually assigned. We then compared the names assigned by ChatGPT to the names by the domain expert. The prompt used is provided in the appendix and as part of the replication package. We used GPT3.0 (OpenAI, 2023) based on the "text-davinci-003" model due to models performance and cost-effectiveness. Since we were only interested in exploring the potential of LLMs for topic labeling tasks, we only considered a single model in this regard and chose not to compare other available LLM models.

3.9 Understanding abandoned and merged changes

The generated topics are derived from the analysis of individual CRCs. On average, abandoned code changes in JabRef have 14 CRCs per code change, while merged code changes have 13 CRCs per code change on average. The highest values of CRCs per code change for abandoned and merged changes are 28 and 29, respectively. Since identified themes can belong to different code changes, we aggregated identified themes at the code change using the *pull request number* to better understand how code changes differ regarding the themes of their related CRCs.

4 Results and analysis

In this section, we describe the characteristics of the data before presenting the derived themes in the two iterations. We then present the results for the two research questions.

4.1 Data characteristics

The distribution of unprocessed CRC lengths from abandoned and merged changes shows that both are predominantly short texts and follow a similar trend (see Fig. 2). Of the CRCs in abandoned and merged changes, 75% are at most 28 and 29 words long, respectively, and only 5% and 3%, respectively, are 80 words or longer in abandoned and merged changes. This confirms our choice of topic model in Section 3.3.1.

Figure 3 shows the number of code commits contributed by different groups of reviewers. Approximately 71% (3,937 out of 5,560) of the CRCs are from three reviewers who have each contributed more than 500 PRs. Eight reviewers contributed between 100 and 500 PRs each and provided 18% (1,013 out of 5,560) of the CRCs. The remaining 33 out of 44 reviewers have only contributed approximately 9% (523 out of 5,560) of the CRCs. The above

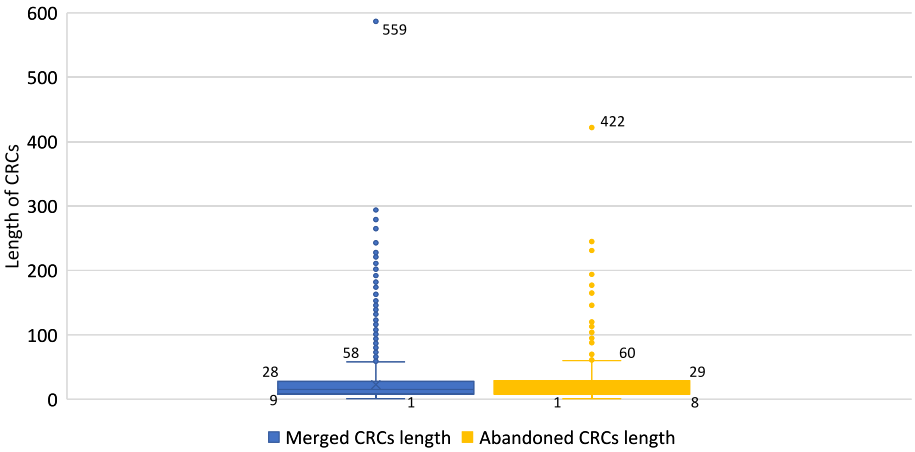

Fig. 2 Distribution of lengths (in the number of words) of CRCs for abandoned and merged changes

patterns show that reviewing is important in JabRef development, as the main contributors also extensively review the code.

Figure 4 shows the distribution of the percentage of total PRs contributed by different developers. The figure further depicts the percentage of each developer's abandoned and merged PRs. Almost 67% (3,780 out of 5,579) of the total PRs are contributed by only 1.7% (11 out of 630) of the developers. Therefore, in Fig. 4, we show the contributions of 11 developers individually and group the rest as 'Others'. The 'Others' in the figure, who individually have contributed to less than one percent of total PRs, have collectively contributed approximately 33% (1,799 out of 5,579) of the total PRs. The top 11 active developers, on average, had approximately 7% PRs abandoned, while other developers, on average, had 30% PRs abandoned.

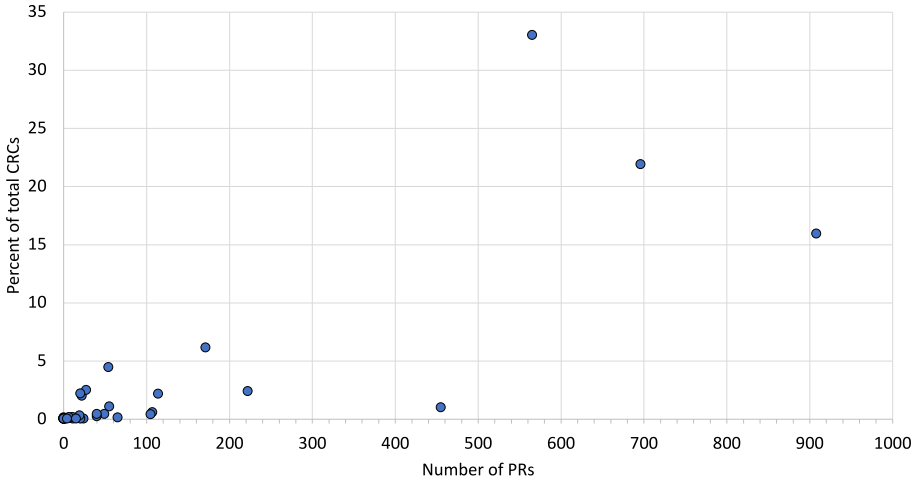

Fig. 3 Distribution of the number of PRs and percentage of CRCs by code reviewers

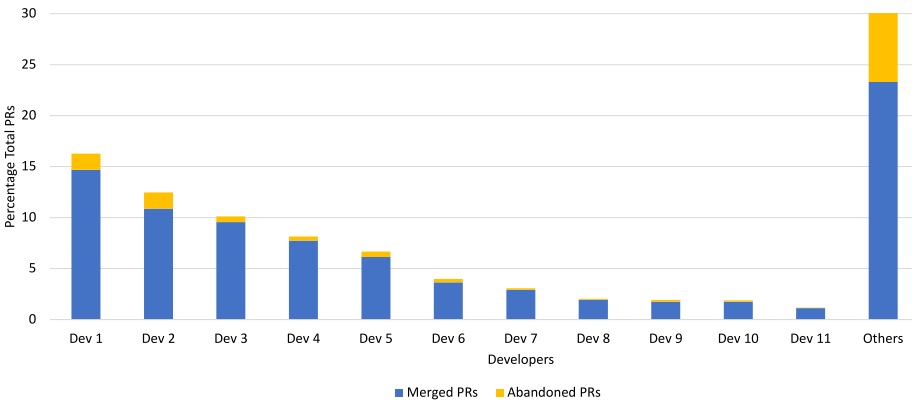

Fig. 4 Distribution of the number of PRs by the developers who contributed 1% or more of the total PRs (Dev 1–Dev 11) and all others (Other)

4.2 Themes identified using topic models

In the following subsections, we first present the themes labeled by the domain expert in each of the two iterations. We provide examples of themes from CRCs for both abandoned and merged code changes.

4.2.1 First Iteration (with GSDMM) themes

We generated five topics each for CRCs from abandoned (T_a1–T_a5) and merged (T_m1–T_m5) changes in the first iteration. The domain expert gave the theme names for these topics. Table 3 summarizes the theme names, from the first iteration, assigned to the generated topics and the corresponding topic share (Barua et al., 2014) for each theme. The topic share is the ratio of the number of CRCs for a topic (or theme) compared to the total number of CRCs and indicates the relative size of a topic. As can be seen from Table 3, the topic shares for

Table 3 Manually assigned themes (first iteration) for the CRCs from abandoned (T_a1–T_a5) and merged (T_m1–T_m5) PRs together with their topic share

Topic	Manually assigned theme	Topic share
T_a1	Java code quality	0.38
T_a2	Preferences localization	0.15
T_a3	External resources & path handling	0.24
T_a4	Code architecture	0.10
T_a5	Code formatting	0.14
T_m1	Shorter code	0.21
T_m2	Modern test style	0.31
T_m3	Simplify control flow	0.14
T_m4	JavaFX architecture	0.18
T_m5	UX	0.17

topics from abandoned PRs are varied, thus suggesting that certain topics are more frequently discussed by the JabRef reviewers.

T_a1: *Java code quality*

With a topic share of 0.38, *Java code quality* is the most prevalent theme for CRCs of abandoned PRs. The CRCs in this theme are related to conforming with common principles and patterns to foster maintainability. An example of a CRC illustrating this theme is shown below (pull request 3418).

> *PR 3418:* "I would propose to make this class non-static / not a singleton. You then have a (public) constructor that accepts the current version string and a default constructor that uses *'java_version'*. In this way, you can also easily write a test to verify the methods in this class." (Italics added for readability)

T_a2: *Preferences localization*

The CRCs in theme *Preferences localization* address issues related to JabRef's customization code, which deals with preferences related to the user interface, language, and localizations. Such preferences and localizations also need to be reflected in the coding style. An example of a CRC illustrating this theme is shown below.

> *PR 7181:* "I wouldn't set a maximum width for year and type columns. If users prefer to give more space to these columns, why not? Simply setting *'prefWidth'* should be fine." (Italics added for readability)

T_a3: *External resources & path handling*

With a topic share of 0.24, *External resources & path handling* is the second largest theme for CRCs of abandoned PRs. The CRCs in this theme deal with the adequate handling of external resources such as file paths, resources identified by URLs, user and global directories, and exception handling. An example of a CRC illustrating this theme is shown below.

> *PR 7728:* "I don't think this comment should be here. If you give this method a relative path it is implementation-dependent I don't think it is always going to be JabRef's home directory. I guess that in practice we should avoid giving it an absolute path."

T_a4: *Code architecture*

Code architecture is the smallest theme for CRCs of abandoned PRs (topic share=0.1). Its CRCs deal with issues related to placing functionality in appropriate classes and methods. An example of a CRC illustrating this theme is shown below.

> *PR 557:* "When a utility method is only used once, it should not be in a utility class. In that case, it would only be used for this call. What is more, I do not like utility classes for domain-specific things like external file types. It is OK for IO methods like reading from a file, interfacing with the file system or for type conversion such as String to int."

T_a5: Code formatting

Code formatting is another relatively small theme (topic share=0.14). The CRCs in this theme are related to code styling, check-style issues, and setting up the workspace. An example of a CRC illustrating this theme is shown below.

> *PR 7172:* "Please set up checkstyle configuration according to our workspace set up guide. Wildcard imports are not allowed."

T_m1: Shorter code

The CRCs in theme *Shorter code* are related to recommendations for using built-in methods. The primary motivation for such recommendations is that shorter code is perceived to be more maintainable. An example of CRC illustrating this theme is shown below.

> *PR 6434:* "One of the *Date.parse* method overloads accepts already *Optional* as parameter so you can make your code a bit easier and remove the *ifPressent* checks for Year and month..." (Italics added for readability)

T_m2: Modern test style

With a topic share of 0.31, *Modern test style* is the most prevalent theme for CRCs of merged PRs. The CRCs in this theme are related to recommendations for using JUnit's parametrized test functionality instead of duplicating test code. The size of the theme indicates that contributing developers often have to be asked to reuse test code. The domain expert observed that groups of contributing developers focused on improving the project's test code, which may explain the increase in testing-related discussions in CRCs. Similarly, when PRs contribute to specific modules related to the core logic and user interface, reviewers often ask them to provide test cases to evaluate the contribution. An example of a CRC illustrating this theme is shown below.

> *PR 6479:* "Thanks for adding so many tests. This is really good. I would propose to split them a bit into two categories: tests for parsing and test for representation. The former should take a string and test against a *'AuthorList'*. The latter should take a *'AuthorList'* and test against a string..." (Italics added for readability)

T_m3: Simplify control flow

The CRCs in this theme focus on suggestions to simplify the code flow, e.g., by reducing the number of code branches and lines of code in a code branch to improve maintainability. An example of a CRC illustrating this theme is shown below.

> *PR 379:* "I think it would be better to find out the exact Exception (should be easy with the test) and then write a multi-catch block..."

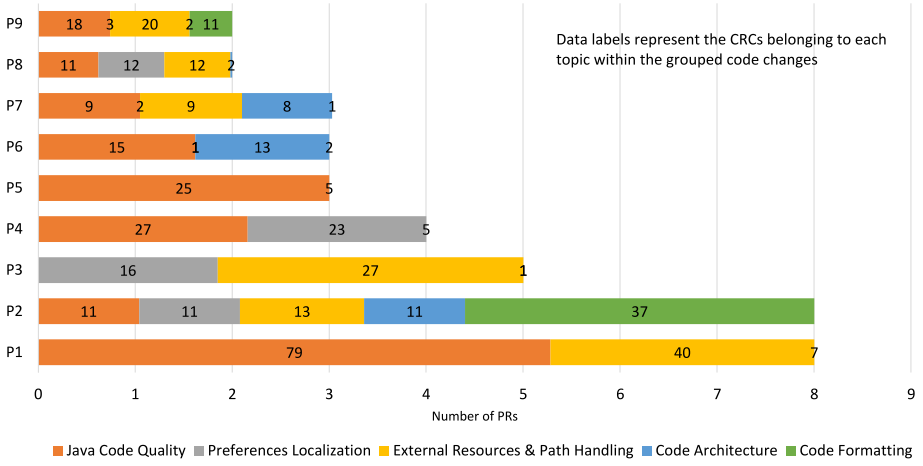

Fig. 5 Profiles for abandoned changes from the first iteration

T_m4: *JavaFX architecture*

The CRCs in this theme discuss the JavaFX[10] architecture that is used in JabRef for user interface design. An example of a CRC illustrating this theme is shown below.

> *PR 4227:* "This works because the dialog is very simple but goes against the usual strategy of *JavaFX / MVVM*. You should add properties in the *ViewModel* class... Please have a look at the other *JavaFX* dialogs to see how this is done..." (Italics added for readability)

T_m5: *UX*

The CRCs in this theme deal with issues related to meeting the needs of the user experience expected from intermediate users of JabRef and the behavior of JabRef's user interface on different operating systems. An example of a CRC illustrating this theme is shown below.

> *PR 1390:* "I find it counterintuitive that the same button sometimes resets only a few bindings and sometimes all. Proposal: add a third column to the table which contains a small reset button (only icon light gray by default dark gray on hovering the row)..."

Understanding abandoned and merged changes

The results for the first iteration are shown in Fig. 5 for abandoned changes and Fig. 6 for merged changes. We observed nine profiles (P1 – P9) for the abandoned code changes shown in Fig. 5. The most frequently discussed combination of themes, *Java code quality* and *External resources & path handling*, is considered in five out of nine profiles containing 60% (23 out of 38) code changes. While we noted in Table 3, 38% CRCs relate to *Java code quality*, interestingly, it is discussed in eight of nine profiles, suggesting that the theme is frequently highlighted across distinct code changes. *Preference locatlization* with only one percent more CRCs compared to *Code formatting* is discussed in twice as many profiles as the latter, thus indicating that the topic impact of a theme and the number of profiles

[10] https://openjfx.io/

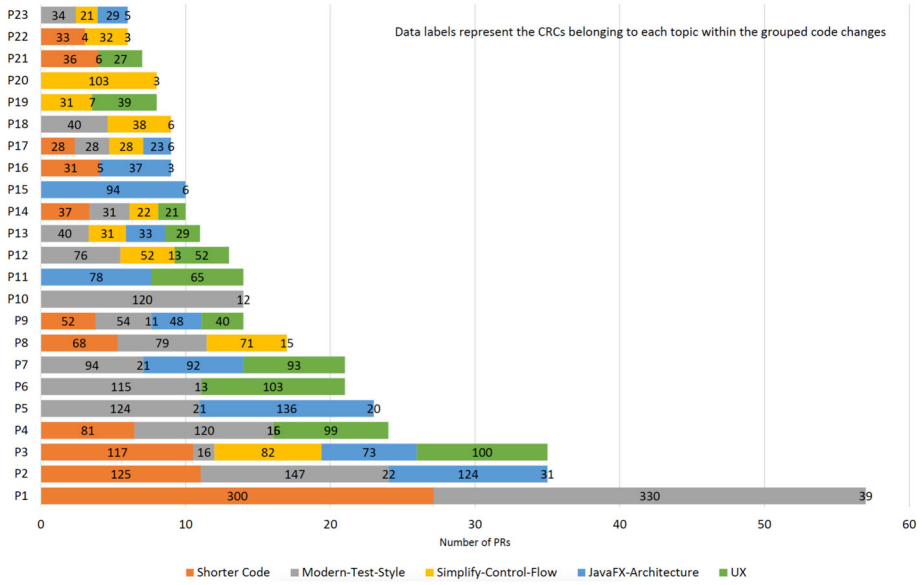

Fig. 6 Profiles (23 out of 32) with more than five merged changes from the first iteration

discussing the theme is non-linear and may differ for themes. In Fig. 5, we also report the CRCs belonging to each theme within the profile to depict the relationship between profiles and CRCs.

In Fig. 6, we report 23 profiles (P1 – P23) with five or more merged changes. The frequently discussed pair of themes, *Shorter code* with *Modern test style* and *Modern test style* with *Simplify control flow*, is considered in eight out of 23 profiles. While we noted in Table 3, 31% CRCs relate to *Modern test style*, it is highlighted in 70% (16 out of 23) profiles, suggesting that the theme is significantly emphasized across distinct code changes. Furthermore, Profile P1, with the highest number of code changes, focuses on only two themes, *Shorter code* and *Modern test style*, with 15% (57 out of 388) of the code changes belonging to the profile. *Modern test style* theme is most often discussed in combination with other themes and is discussed across 16 profiles. Incidentally, the other four themes are highlighted across 11 out of 23 profiles each, despite varied topic share. Figure 6, we further report the CRCs for each theme within the profile to highlight the relationship between profiles and CRCs.

4.2.2 Second iteration (with BERTopic) themes

Using the steps described in Section 3.4, we generated four themes for the CRCs from abandoned changes ($T_{aB}1$–$T_{aB}4$) and 19 themes for CRCs from merged changes ($T_{mB}1$–$T_{mB}19$), respectively.

Table 4 summarizes the theme names assigned to the generated topics from the second iteration and the corresponding topic share (Barua et al., 2014) for each theme. As can be seen from Table 4, $T_{aB}1$ is the most prominent theme within the abandoned PRs, while there

Table 4 Manually assigned themes (second iteration) for the CRCs from abandoned ($T_{aB}1$–$T_{aB}4$) and merged ($T_{mB}1$–$T_{mB}19$) PRs together with their topic share

Topic	Manually assigned theme	Topic share
$T_{aB}1$	Basic JabRef style	0.87
$T_{aB}2$	Code format	0.04
$T_{aB}3$	Column handling	0.03
$T_{aB}4$	Code indent	0.02
$T_{mB}1$	String methods	0.10
$T_{mB}2$	Agreement	0.09
$T_{mB}3$	Datatype bibentry	0.08
$T_{mB}4$	File handling in java & JabRef	0.07
$T_{mB}5$	Proper testing	0.07
$T_{mB}6$	Dialogs	0.05
$T_{mB}7$	*Unlabeled topic*	0.04
$T_{mB}8$	BibTeX fetchers	0.03
$T_{mB}9$	Comment	0.03
$T_{mB}10$	Localization	0.03
$T_{mB}11$	Scoping and naming	0.03
$T_{mB}12$	Logging	0.02
$T_{mB}13$	Git branch handling	0.02
$T_{mB}14$	Empty lines	0.02
$T_{mB}15$	Null handling	0.02
$T_{mB}16$	Removal	0.02
$T_{mB}17$	Comment javadoc	0.02
$T_{mB}18$	Handling storage	0.01
$T_{mB}19$	Concrete code suggestions	0.01

Note: Among the CRCs from abandoned changes, four percent of the CRCs were considered outliers. Twenty-four percent of CRCs were considered outliers in the CRCs from merged changes

is a similar variation in the topic share for the themes from merged PRs, with $T_{mB}1$ having the highest topic share.

Second iteration themes for CRCs from abandoned changes

In this subsection, we describe two of the four most prominent (based on the highest topic share) themes from abandoned changes.

$T_{aB}1$: *Basic JabRef style*

The CRCs in the theme *Basic JabRef style* relate to basic knowledge of JabRef's coding style The code reviewers often have to provide feedback on coding style and design-related issues, making this theme the most prominent theme in the abandoned PRs. The domain expert considered the theme to be generic and uninformative to improve the practices at JabRef. The theme is similar to T_a5, which also focuses on code formatting-related issues while also identifying design-related issues. An example of a CRC that illustrates this theme is shown below.

> *PR 8762:* "Please avoid code-describing comments instead try to write self-explaining code and put your ratilonale into a comment."

$T_{aB}3$: *Column handling*

The CRCs in this theme deal with issues related to handling table columns in the JabRef GUI. The domain expert considered the theme challenging to label yet meaningful for JabRef GUI development since GUI is one of the largest packages of JabRef. With a topic share of only 0.03, it is one of the smallest themes in the abandoned PRs. While the theme is similar to T_m5, it focuses more on specific issues related to column handling from a user's perspective. An example of a CRC that illustrates this theme is shown below.

> *PR 7181:* "I think this might lead to unused space on the right when one of the columns has a higher preferred width than its maximum (because you don't redistribute the difference among the other columns)."

Second iteration themes for merged changes

Among the themes from merged PRs, the domain expert considered several topics to be "basic" or common knowledge, thus being less informative, and observed that "anyone who has spent 10 hours with the code base" can learn to apply them during software development. An example of basic themes include *Agreement* ($T_{mB}2$) described by the "agreeing to code changes." An example of CRC belonging to this theme is provided below.

> *PR 1381:* "Ok then we leave it like that for the moment and maybe fix it later."

Other examples for themes from merged changes include *Empty Lines* ($T_{mB}14$) described as addressing code styling w.r.t empty lines, *Null handling* ($T_{mB}15$) focusing on suggestions on properly dealing with null values, and *Comment* ($T_{mB}9$) including suggestions to write proper code comments. Furthermore, *Removal* ($T_{mB}16$) and *Comment javadoc* ($T_{mB}17$) were also cited as coherent yet non-informative themes by the domain expert. A few examples of the informative themes from merged PRs are given below.

$T_{mB}3$: *Datatype bibentry*

With a topic share of 0.08, Datatype bibentry is the third largest theme for CRCs of merged PRs. The CRCs in this theme focus on modifying and using the built-in datatype that models a Bib(La)TeX entry in JabRef. The domain expert considered the theme easy to interpret and meaningful since newcomers often need help understanding the built-in datatypes within JabRef. An example of a CRC illustrating this theme is shown below.

> *PR 1596:* "Better create a *'Map<BibEntry List<File» expected'* and compare that it equals *'results'*." (Italics added for readability)

$T_{mB}5$: *Proper testing*

The CRCs in this theme focus on suggestions for properly testing the submitted code for edge cases and coverage, thus reducing the chances of bugs and defects in the main repository. With a topic share of 0.07, it is the fourth largest theme in the merged PRs and is similar to the T_m2 from the first iteration. The domain expert mentioned that reviewers often request

additional tests to ensure high-quality source code is submitted for review. An example of a CRC illustrating this theme is shown below.

> *PR 8531:* "Could you add a new test case please? One with *'day'* and one without *'day'*? - Both case(s) might appear in the wild and should be tested." (Italics added for readability)

$T_{mB}6$: *Dialogs*

With a topic share of 0.05, Dialogs is the fifth most prominent theme for CRCs of merged PRs. The CRCs in this theme deal with the proper use of JabRef's prepared dialog classes to display messages to the users. The theme shares similarities with the T_m5 since both themes deal with the user interface while being different in their focus on specific dialog classes. An example of a CRC illustrating this theme is shown below.

> *PR 4983:* "Can you please try to pass the dialog service as a constructor argument (reason: dependencies should be specified in the constructor)..."

Understanding abandoned and merged changes

We report six profiles (P1 – P6) for the abandoned changes in Fig. 7. In the case of the merged changes, we report 11 profiles in Fig. 8, with at least five or more PRs belonging to the profile.

Profile P1 is the most frequently discussed profile, with 60% (23 out of 38) code changes focusing on the *Basic JabRef style*. We note in Table 4, 86% CRCs belong to the *Basic JabRef style*, and expectedly, it is discussed across five out of six profiles. The other three themes with a collective 7% CRCs belonging to them are reflected in two profiles each.

We report 11 profiles in the merged changes, with five or more PRs belonging to each profile, depicted in Fig. 8. These 11 profiles only discuss six of the 19 themes from the merged changes. Themes with relatively high topic share, such as *Agreement* ($T_{mB}2$), *Datatype bibentry* ($T_{mB}3$), and *Dialogs* ($T_{mB}6$) are discussed in PRs with less than five occurrences in the merged changes. Compared to the first iteration, a smaller number of profiles contained five or more PRs for merged changes. Profiles P1 – P4, with more than ten PRs each, discuss only a single theme, with a relatively smaller number of CRCs belonging to each theme (see

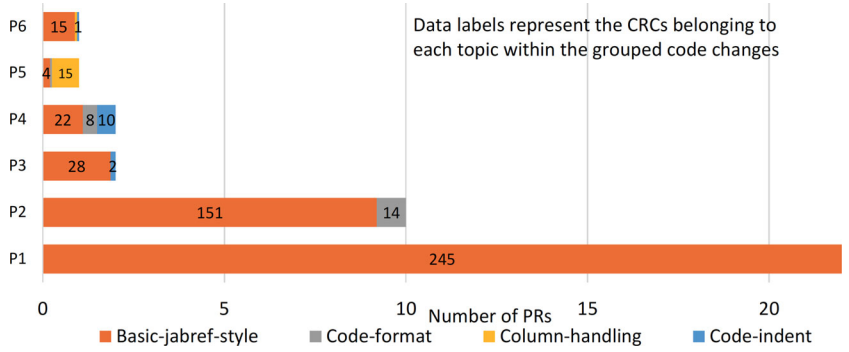

Fig. 7 Profiles for abandoned changes from the second iteration

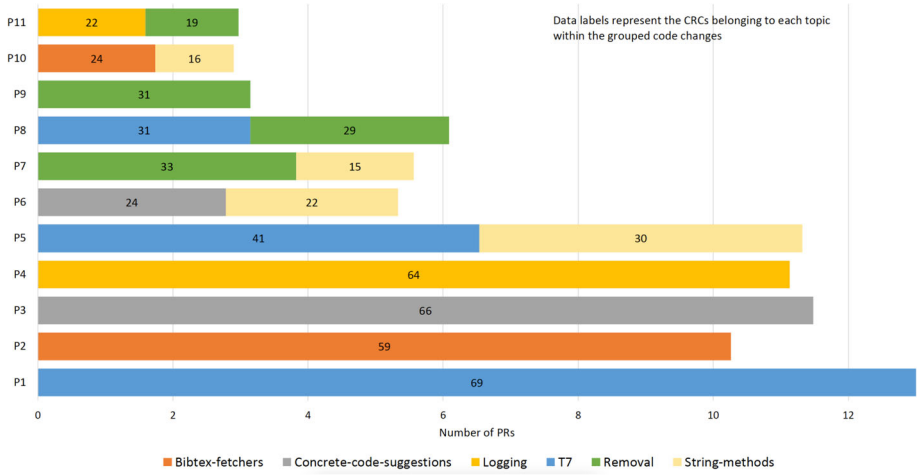

Fig. 8 Profiles (11 out of 23) with more than five merged changes from the second iteration

Table 4). Interestingly, *String methods*, with 10% CRCs belonging to it, and *Removal*, with only 2% CRCs belonging to it, are the most frequently discussed themes across different profiles. The domain expert considered Profile P1, with the highest number of PRs, to be too diverse to be labeled.

4.3 RQ1: Interpretability and meaningfulness of the identified themes

The domain expert considered the themes generated from the first iteration to be relatively easier to interpret compared to the themes generated from the second iteration. The objective coherence, measured using C_v, for the topics abandoned changes was *0.35* while the objective coherence of topics from merged changes was marginally higher at *0.53*. The domain expert then named these topics as described in Section 3.7.

Regarding the generated themes from the first iteration, the domain expert considered three-fourths of the themes to be easy to provide a name for. In contrast, one-fourth of the identified themes needed to be more coherent. However, after reading CRCs from other themes, it was also possible to name such themes.

According to the domain expert, the generated topics were informative to improve the practices for JabRef. Themes such as T_a2, T_a4, T_m1 till T_m5 were highlighted as informative to the practice at JabRef. However, the domain expert considered T_a1 as a generic theme.

The domain expert considered topics from the second iteration comparatively challenging to interpret. Compared to the first iteration, the objective coherence, C_v, for the topics from abandoned changes is significantly higher at *0.68*. The objective coherence of topics from merged changes was also higher at *0.65* compared to the first iteration. The domain expert used the same process as the first iteration to name these topics described in Section 3.7.

Regarding the degree of interpretability of the generated topics from the second iteration, the domain expert considered half of the topics easy to provide a name. In contrast, the domain expert considered the other half of the topics challenging to interpret. One topic, $T_{mB}7$, could not be interpreted as it was considered to be too diverse by the domain expert. Our second iteration shows that higher objective coherence in terms of C_v did not correlate with the interpretability of the topics based on the domain expert's opinion.

The domain expert perceived the themes from the first iteration as relatively easier to interpret than the themes from the second iteration. While the second iteration yielded topics with higher objective coherence scores, the domain expert considered them more challenging to interpret. This suggests that objective coherence (measure C_v) and perceived interpretability by a domain expert may not be correlated.

4.4 RQ2: themes labeled using large-language models

Assigning theme labels can be resource-intensive (Silva et al., 2024) as it requires the involvement of the domain expert and, thus, also limits the number of approaches that can be evaluated by the domain expert. As mentioned in Section 3.7, we triangulated the manually assigned themes by the domain expert using ChatGPT. The results from ChatGPT and the corresponding theme name from manual labeling for the two iterations are provided in Tables 5 and 6, showing that, in our case, automatically labeled themes by LLM closely resemble the manually assigned themes in several instances. Some examples of closely resembling themes in the first iteration include *Java code quality* and *Code architecture*, which the LLM identified as *Refinement & documentation for code quality* and *Refinement & architecture enhancement*, respectively. In contrast, themes with low resemblance include *External resources & path handling*, which the LLM identified as *Refinement & precision in code development*.

Similarly, in the second iteration, examples of closely resembling themes include *Column handling* and *File handling in java and JabRef*, which LLM labeled as *Table column width management* and *Improving file and path handling in Java code*, respectively. In contrast, themes with low resemblance include *Basic JabRef style* and *String methods*, which the LLM identified as *Code quality and best practices* and *Code simplification and optimization*.

For one of the unlabeled topics ($T_{mB}7$), we presented the domain expert with the theme label generated by the LLM. However, the domain expert observed the theme label was

Table 5 Comparison of manually and automatically assigned themes for the CRCs from abandoned (top) and merged (bottom) changes using the first iteration

Manual	Automatic
Java Code Quality	Refinement & Documentation for Code Quality
Preferences localization	User Interface Enhancement & Preference Handling
External resources & path handling	Refinement & Precision in Code Development
Code architecture	Refinement & Architecture Enhancement
Code formatting	Code Standards Adherence and Refactoring
Shorter Code	Code Optimization & Best Practices
Modern test style	Test Refinement & Maintenance
Simplify control flow	Refinement for Enhanced Code Quality & Readability
JavaFX architecture	Improving GUI Architecture & Responsiveness
UX	User Experience Enhancement & Functionality Optimization

generic and uninformative. For one more topic ($T_{mB}18$), the domain expert used the LLM-generated theme label to derive a suitable theme label during the interview, thus highlighting the usefulness of LLM in supporting topic labeling tasks.

Our results open up further opportunities for automation in our approach to identifying prevalent quality issues.

> Manually assigning theme labels is resource-intensive. The study compared expert-assigned themes with LLM-generated labels, showing a resemblance in several cases, while other themes showed lower similarity. Our results suggest a potential for further automation in identifying prevalent quality issues.

5 Discussion

We used two iterations, using the approach by Iftikhar et al. (2023) and the BERTopic-based topic modeling method, to identify recurring quality issues in code changes by analyzing the CRCs. In this section, we discuss the results of using the two topic modeling methods and the implications for software development practice.

5.1 Performance of topic models

According to the domain expert, the STTM method in the first iteration leads to comparatively more interpretable topics. Silva et al. (2024) compared four STTM models and reported that GSDMM led to more comprehensible (measured using the average number of names per topic by human participants and the number of topics named) topics in their dataset. However, their study did not evaluate any implementation of BERTopic; thus, a direct comparison with their results cannot be made. Similar to the findings of the comparative study by Udupa et al. (2022), BERTopic produced more coherent topics based on objective measures compared to GSDMM. However, Udupa et al. (2022) did not consider human judgment.

One possible explanation for the difficulty in interpreting topics from the second iteration may be the large number of topics generated. The domain expert indicated that the cognitive load of naming four times as many topics for the merged case may be a factor that impacts the interpretability of the topics in the second iteration. The number of topics generated was one of the factors when selecting the embedding models used during the second iteration. The higher objective coherence observed in the second iteration and the number of topics resulted in "focused" topics with a relatively small individual topic share. This may explain the domain expert's reflections regarding the less interesting topics for the state-of-the-practice at JabRef. We achieved higher average objective coherence, i.e., average objective coherence of topics from CRCs on abandoned and merged changes, with two different embedding models. "CodeT5"[11] provided an average objective coherence of 0.685, and "CodeBertA"[12] produced an average objective coherence of 0.68, compared to the average objective coherence of 0.66 using the embedding model described in Section 3.4. The quantitative results of the parameters explored for both topic models, GSDMM and BERTopic, along with practical considerations, are reported in (Iftikhar, 2024). We opted not to use both embedded models

[11] https://huggingface.co/Salesforce/codet5-small

[12] https://huggingface.co/huggingface/CodeBERTa-small-v1

due to the relatively higher number of topics generated (the "Code T5" embedding model produced 36 topics, and "Code BertA" generated 40 topics).

Based on the recommendations of Martin Borčin (2024), we considered using 20% as a suitable percentage of outliers when selecting *min cluster size* and *min sample size*. However, Martin Borčin (2024) achieved their results on generic datasets from newsgroup posts and BBC articles and may not be a suitable choice for the analysis of CRCs.

Furthermore, we also evaluated an alternate clustering approach, KMeans, which does not assume any outliers. However, KMeans produced an average objective coherence of 0.63 compared to the average objective coherence of 0.665 from using HDBScan. Thus, we adopted an HDBScan-based approach.

5.2 Objective coherence & human assessment

The objective coherence measure used as a basis to select the topics presented to the domain expert is another aspect that impacts the quality of themes generated and their interpretability. While Röder et al. (2015) recommend using objective coherence, C_v, based on its correlation with human rating, Silva et al. (2024) observed that none of the objective coherence measures used in their study analyzing developer communication bear similarity to human comprehension of topics. Intuitively, comprehension is required for the interpretability of a topic. The same result was observed in our results, where a higher theoretical objective coherence did not translate to improved human interpretability of generated topics as assessed by the domain expert. Since objective coherence measures are based on the statistical distribution of topic terms and human assessment involves understanding the meaning of topic terms, Silva et al. (2024) recommend using objective coherence measures *in addition* to human assessment as both these measures complement each other.

Based on the results, we recommend using a lower number of topics and using LLMs to determine the interpretability of generated topics before evaluation by a domain expert. Future researchers can evaluate the interpretability of the topics when using KMeans-based methods that do not create outliers.

5.3 Contextualization of code changes

Beyond the various choices in topic models and objective coherence measures, the analysis of CRCs may be further improved by involving the context of the code changes. The CRCs in a submitted code change under review are linked as they discuss the same snippet of code changes. Tang et al. (2013) have suggested contextualization may improve classification tasks. Aggregating the CRCs belonging to a code change into a single document may improve the contextualization and the quality of the themes. However, contextualization impacts the topic modeling methods available as aggregating CRCs in a single document may no longer be short-text data. Due to the resource-intensive nature of involving the domain expert in evaluating such an approach, we intend to assess such an approach as part of our future work.

5.4 Themes related to code quality

The common themes indicate a focus on long-term readability (such as T_m1, T_a5, $T_{aB}4$, $T_{mB}14$), testability (for example, T_m2, $T_{mB}5$,), and code structure (see T_m3, T_m4, $T_{aB}2$). The common themes in abandoned and merged changes can be broadly categorized as related to

maintainability, e.g., by focusing on code structure and utilizing well-tested built-in functions, which is aligned with existing taxonomies of issues found in CRCs (Mäntylä and Lassenius, 2009; Beller et al., 2014). Other themes identified relate to alternate code suggestions (such as $T_{mB}1$, $T_{mB}19$) and code review process (examples include $T_{mB}13$, $T_{mB}2$) which has been identified in the taxonomy proposed by Ochodek et al. (2022). The identified themes in abandoned changes from the first iteration highlight issues related to the code architecture, formatting of code, and code quality, while the themes from the second iteration focus on code formatting and JabRef-specific styling. In contrast, the themes in merged changes from the first iteration emphasize alternate suggestions that help to improve the issues related to implementation choices related to JavaFX architecture, testing options, and built-in functions. Meanwhile, the themes in the merged changes from the second iteration focus on testing, alternate implementation suggestions, file handling, naming of variables, and process-related issues.

5.5 Theme labels using LLMs

The identified themes need to be assigned an appropriate name, which is currently a manual process requiring practitioners' input, which limits the approach. An automated method that aids in naming the identified themes, e.g., by using Large Language Models (OpenAI, 2023), may be used as an initial step to shortlist results from an approach before involving the domain expert, thus improving the efficiency of the topic naming steps.

5.6 Implications for state-of-practice

The domain expert observed two potential areas where the proposed analysis could improve the state of the practice for JabRef. The identified themes can provide ideas to initiate focused discussions on specific aspects of the JabRef discussion forums. By creating Q&As forms for each identified theme, the discussion forums can be used to acquire important feedback on identified themes, thus improving the interactivity of the discussion forums. Additionally, JabRef provides guidelines to help newcomers.[13] The identified themes can give concrete ideas on which content to update in the guidelines to improve their effectiveness and address some of the challenges faced by newcomers (Steinmacher et al., 2015).

5.7 Theme evolution

To analyze the evolution of common themes over time, we depicted scatter plots to show the topic impact (Barua et al., 2014) over the months considered used by existing studies (Wen et al., 2022; Iftikhar et al., 2023). While the evolution graphs may be helpful to visualize the overall trend for a theme, the domain expert found it challenging to explain specific trends in the topic evolution graphs, Fig. 9 depicts the evolution of theme T_m3 *Simplify control flow* as an example. For a meaningful analysis of the topic evolution graphs, the domain expert observed that more data is needed, e.g., which pull requests are active in those months and who are the top contributors for the given month. The main reason is that different contributors with different preferences were active in different periods of the project. Due to the relatively small dataset, we chose not to empirically evaluate the themes version-wise as

[13] https://devdocs.jabref.org/getting-into-the-code/guidelines-for-setting-up-a-local-workspace/

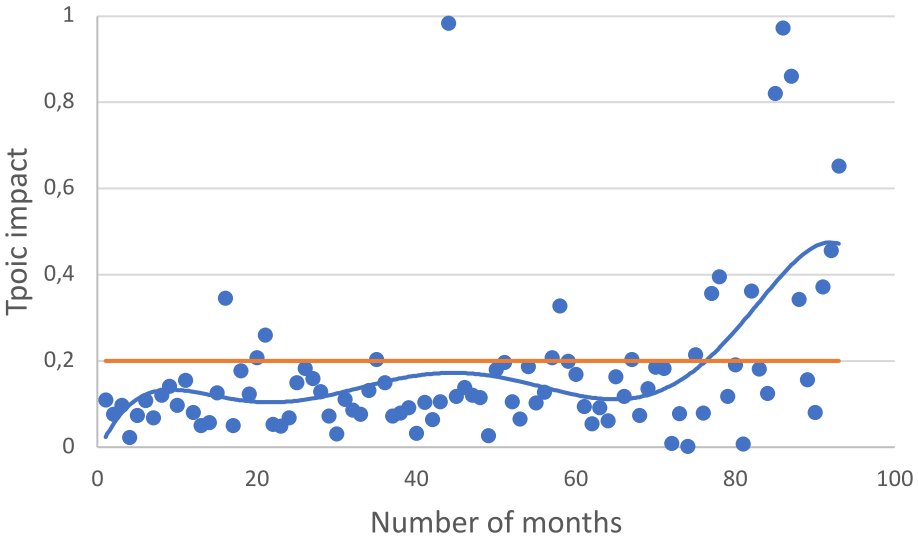

Fig. 9 Topic evolution for theme from CRCs on merged changes $T_m 3$ *Simplify control flow* in JabRef

suggested previously (Iftikhar et al., 2023), as version-wise analysis for larger datasets may lead to interesting results.

The data characteristics discussed in Section 4.1 indicate that there is a potential link between the number of PRs submitted by developers and the outcome of the code review, thus indicating that there may be other non-technical factors that impact the outcome of the code review process, which is aligned with previous results (Thongtanunam et al., 2015; Fan et al., 2018).

5.8 Potential applications for presented approach

The potential application of the presented approach includes analyzing the code review comments in various scenarios, e.g., studying prevailing code quality issues, trend analysis of code quality issues, and the impact of an intervention on code quality. Similar to the current study's setup, we can use the presented approach to compare the quality issues in two groups of code changes based on the code review comments, e.g., studying quality issues from code changes that take longer to merge compared to code changes that are merged quickly, or studying differences in feedback given in projects following a monolithic design approach and micro-services approach. For practitioners and researchers, we briefly summarize the approach we followed in this study and discuss further applications.

1. **Step 1:** Collect CRCs; if one is interested in comparing groups of CRCs, split the CRCs into separate subsets. In this study, we compared prevalent themes in merged and abandoned code changes. Therefore, we split the CRCs into comments made by reviewers on merged code changes and comments on abandoned code changes.
2. **Step 2:** Using the scripts available in the replication package, pre-process CRCs and generate topics for each dataset.
3. **Step 3:** A domain expert reads each topic's top CRCs and top terms to assign a suitable theme (see Section 3.7).
4. **Step 4:** Using the named themes, create profiles of prevalent quality issues for each dataset.

6 Threats to validity

We use the classification suggested by Runeson and Höst (2009) to discuss the validity threats.

6.1 Reliability

We used automated tools and scripts to reduce the possibility of human error during data curation to a minimum. To ensure that we only used CRCs as input from the extracted data, we removed discussion replies from developers by removing the discussion comments where the comment's author was the same as the change author. We incorporated an embedding model (Efstathiou et al., 2018) derived from posts in StackOverflow,[14] a platform to discuss software code issues, to improve the quality of the generated topics further. Our selected word embedding model is based on software development terminologies, which we believe is suitable.

Although studies have suggested removing highly frequent words to help create distinct topics (Buntine and Mishra, 2014), this can remove important words (Xu et al., 2017). We chose not to remove highly frequent words and short CRCs during preprocessing, as we consider frequent words by reviewers and short CRCs relevant to the analysis performed in the study. To support the repeatability of the study, we have provided a replication package containing the extracted datasets and Python scripts used for the data extraction, preprocessing, and topic modeling.

6.2 Internal validity

The CRC data extracted from the GitHub platform may only partially capture the recurring quality issues. Some quality issues may be discussed using other modes of communication, e.g., the discussion forums, which are not reflected using the approach used in the study. However, since open-source communities extensively use GitHub for collaboration among contributors, we sufficiently capture the recurring quality issues in JabRef. The structured questionnaire shared with the domain expert was curated by the first author and reviewed by the second author for content validity and clarity. However, the order of the data shared with the domain expert for topic naming, e.g., the order of 20 CRCs and the order of the topics presented, may introduce a response bias. Due to practical considerations, we could only involve a single code developer from JabRef. Engaging additional practitioners from JabRef could have further mitigated potential bias in the naming of topics and the subjective assessment regarding their interpretability and meaningfulness. Our dataset comprised CRCs from 45 reviewers. We consider the number of reviewers involved to sufficiently ensure diversity of review feedback which does not impact the code review comments provided.

6.3 Construct validity

While we have selected only topic stability (Agrawal et al., 2018) to select the appropriate number of topics in the first iteration, other fitness measures, such as silhouette coefficient (Panichella et al., 2013), may lead to different topics.

[14] https://stackoverflow.com/

6.4 External validity

Since we utilized only JabRef, other systems with different developers, reviewers, and review practices may lead to distinct results. A large number of contributors to JabRef are engineering students. The language used in the CRCs may vary for other open-source and industrial systems, limiting our results' generalizability. Intuitively, the granularity of reviewer feedback varies between reviewers; thus, the number of reviewers involved in the review also impacts the identified themes in CRCs. Further studies are needed using varied datasets and accommodating differences in feedback due to individual reviewers.

7 Conclusions

We conducted a participatory case study using two design iterations to support practitioners through automation in identifying and profiling prevalent quality issues, using common themes discussed in CRCs from abandoned and merged changes. We followed the approach of an existing study (Iftikhar et al., 2023) in the first iteration and a BERTopic-based approach in the second iteration.

The common themes named by the domain expert demonstrate that the approach can help identify recurring code quality issues discussed in CRCs. We identified different themes from CRCs in abandoned and merged changes in both iterations. The prevalent code quality issues broadly aim to address the maintainability-focused issues in JabRef. Furthermore, we observed unique profiles for code quality issues discussed in pull requests from abandoned and merged changes. In addition, we outline the steps required to apply a similar approach to other potential applications.

The results derived from the analysis of CRCs can help in improving the guidelines for new developers. They can assist in directing focused discussions in the developer forums, thus potentially enhancing the current practices for JabRef. Although many identified themes were easy to assign a name to by the domain expert in the first iteration, we observed a reduction in the interpretability of the topics based on the domain expert's opinion when using BERTopic, even though it produced more coherent themes based on the objective measures of coherence. More research is required to explore how and if objective measures that are more reflective of the subjective assessment of coherence can be developed.

In future studies, we plan to explore variants of BERT (Liu et al., 2019), evaluating how the percentage of outliers impacts the interpretability of the topics and exploring whether contextualization of CRCs using code change improves the interpretability by human practitioners. Furthermore, we plan to investigate whether Large Language Models (OpenAI, 2023) can support assessing the interpretability of the generated topics. Using our approach, we plan to use industrial datasets to improve development guidelines and data-driven discussions to improve development practices.

Appendix

LLM prompt used

"You are an experienced code reviewer who has extensive knowledge of programming and reading source code. Here are 10 code review comments that are separated by #####. You

will analyze each one of these code review comments and give them each a suitable title that relates to the main theme each code review comment is discussing. Ensure that you give the same number of titles as the code review comments. The 10 code review comments are: *Here we provide ten CRCs from each topic*."

Table 6 Comparison of manually and automatically assigned themes for the CRCs from abandoned (top) and merged (bottom) changes using the second iteration

Manual	Automatic
Basic JabRef style	Code Quality and Best Practices
Code format	Code Cleanup & Refactoring
Column handling	Table Column Width Management
code-indent	Consistency in Code Indentation
String methods	Code Simplification and Optimization
Agreement	General Feedback and Minor Adjustments
Datatype bibentry	Working with BibEntry Objects in BibTeX/BibLaTeX Processing
File handling in Java and JabRef	Improving File and Path Handling in Java Code
Proper testing	Improving Test Cases and Mocking in Java Code
Dialogs	Improving Dialog Management and Refactoring in JavaFX Applications
T7 (Unlabelled Topic)	Enhancing JabRef Codebase through Improved Configuration Management, Testing, Code Quality, Error Handling, and General Code Maintenance
BibTeX fetchers	Enhancing Exception Handling and Fetcher Implementation in Code
Comment	Improving Code Clarity and Documentation through Effective Commenting
Localization	Improving Localization and Translation Consistency in Code
Scoping & naming	Best Practices for Method and Class Design in Object-Oriented Programming
Logging	Best Practices for Logging & Error Handling in Software Development
Git branch handling	Managing Merge Processes and Resolving Conflicts in Version Control
Empty-lines	Code Cleanup & Simplification
Null-handling	Effective Null Handling in Java
Removal	Code Cleanup and Removal
Comment Javadoc	Documentation and Code Quality
Handling storage	Enhancing Database Handling in the Codebase
Concrete code suggestions	Documentation & Code Annotations

Acknowledgements We would like to thank Dr. Mirosław Ochodek from Poznan University of Technology for their valuable consultation and insightful discussions that greatly assisted the research.

Author Contributions Umar Iftikhar, Jürgen Börstler, and Nauman Bin Ali contributed to the study's conception, methodology, and visualization. Umar Iftikhar carried out data curation and software development, and the other authors discussed and reviewed all the steps. Umar Iftikhar wrote the first draft of the manuscript, and all authors commented on intermediate versions. Oliver Kopp contributed to the validation of the results.

Funding Open access funding provided by Blekinge Institute of Technology. This work has been supported by ELLIIT, a Strategic Research Area within IT and Mobile Communications, funded by the Swedish Government. The work has also been supported by a research grant for the GIST project (reference number 20220235) from the Knowledge Foundation in Sweden.

Data Availability The Python scripts and datasets are available online https://doi.org/10.5281/zenodo.13453045.

Declarations

Competing Interests The authors declare no competing interests.

References

Agrawal, A., Fu, W., & Menzies, T. (2018). What is wrong with topic modeling? And how to fix it using search-based software engineering. *Information and Software Technology, 98*, 74–88.

Ahasanuzzaman, M., Asaduzzaman, M., Roy, C. K., & Schneider, K. A. (2020). Caps: a supervised technique for classifying stack overflow posts concerning api issues. *Empirical Software Engineering, 25*, 1493–1532.

Aletras, N., Stevenson, M. (2013). Evaluating topic coherence using distributional semantics. In: In proceedings of the 10th international conference on computational semantics, pp. 13–22

Arafat, Y., Shamma, S.S.H. (2020). Categorizing review comments by mining software repositories. *International Conference on Advances in Computing and Data Sciences*, *12*

Baca, D., & Petersen, K. (2013). Countermeasure graphs for software security risk assessment: An action research. *Journal of Systems and Software, 86*(9), 2411–2428.

Bacchelli, A., Bird, C. (2013). Expectations, outcomes, and challenges of modern code review. In: Proceedings of the 35th international conference on software engineering, pp. 712–721

Barua, A., Thomas, S. W., & Hassan, A. E. (2014). What are developers talking about? An analysis of topics and trends in stack overflow. *Empirical Software Engineering, 19*, 619–654.

Bavota, G., Russo, B. (2015). Four eyes are better than two: On the impact of code reviews on software quality. In: Proceedings of the 31st ieee international conference on software maintenance and evolution, pp. 81–90

Beller, M., Bacchelli, A., Zaidman, A., Juergens, E. (2014). Modern code reviews in open-source projects: which problems do they fix? In: Proceedings of the 11th working conference on mining software repositories, pp. 202–211

Biggers, L. R., Bocovich, C., Capshaw, R., Eddy, B. P., Etzkorn, L. H., & Kraft, N. A. (2014). Configuring latent dirichlet allocation based feature location. *Empirical Software Engineering, 19*, 465–500.

Bosu, A., Carver, J. C., Bird, C., Orbeck, J., & Chockley, C. (2017). Process aspects and social dynamics of contemporary code review: Insights from open source development and industrial practice at microsoft. *Proceedings of the IEEE Transactions on Software Engineering, 43*(1), 56–75.

Bouma, G. (2009) Normalized (pointwise) mutual information in collocation extraction. In: In Proceedings of german society of computational linguistics & language technology, *30*, pp. 31–40

Buntine, W.L., Mishra, S. (2014). Experiments with non-parametric topic models. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining, pp. 881–890

Devlin, J., Chang, M.-W., Lee, K., Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv:1810.04805

Efstathiou, V., Chatzilenas, C., Spinellis, D. (2018). Word embeddings for the software engineering domain. In: Proceeding of the 15th international conference on mining software repositories, pp. 38–41

Fan, Y., Xia, X., Lo, D., & Li, S. (2018). Early prediction of merged code changes to prioritize reviewing tasks. *Empirical Software Engineering, 23*(6), 3346–3393.

Fregnan, E., Petrulio, F., Di Geronimo, L., & Bacchelli, A. (2022). What happens in my code reviews? An investigation on automatically classifying review changes. *Empirical Software Engineering, 27*, 89.

Gottigundala, T., Sereesathien, S., Da Silva, B. (2021). Qualitatively Analyzing PR Rejection Reasons from Conversations in Open-Source Projects. In: Proceedings of the 13th IEEE/ACM international workshop on cooperative and human aspects of software engineering, pp. 109–112

Grootendorst, M. (2020). BERTopic: Neural topic modeling with a class-based TF-IDF procedure. arXiv:2203.05794

Gunawardena, S., Tempero, E., & Blincoe, K. (2023). Concerns identified in code review: A fine-grained, faceted classification. *Information and Software Technology,153*, 107054

Han, J., Shihab, E., Wan, Z., Deng, S., & Xia, X. (2020). What do programmers discuss about deep learning frameworks. *Empirical Software Engineering, 25*, 2694–2747.

Haque, M.U., Iwaya, L.H., Babar, M.A. (2020) Challenges in docker development: A large-scale study using stack overflow. In: Proceedings of the 14th international symposium on empirical software engineering and measurement, pp. 1–11

Henß, S., Monperrus, M., Mezini, M. (2012). Semi-automatically extracting faqs to improve accessibility of software development knowledge. In: 2012 34th International conference on software engineering (ICSE), IEEE, pp. 793–803

Iftikhar, U. (2024). Practical considerations and solutions in nlp-based analysis of code review comments-an experience report. In: International conference on product-focused software process improvement, Springer, pp. 342–351.

Iftikhar, U., Börstler, J., Ali, N.B. (2023). On potential improvements in the analysis of the evolution of themes in code review comments. In: 49th Euromicro conference on software engineering and advanced applications (SEAA), pp. 340–347

Johnson, B., Song, Y., Murphy-Hill, E., Bowdidge, R. (2013). Why don't software developers use static analysis tools to find bugs? In: Proceedings of the 35th international conference on software engineering, pp. 672–681

Kindon, S., Pain, R., Kesby, M. (2007). Participatory action research approaches and methods. Connecting people, participation and place. Abingdon: Routledge *260*

Kononenko, O., Rose, T., Baysal, O., Godfrey, M., Theisen, D., De Water, B. (2018). Studying pull request merges: a case study of shopify's active merchant. In: Proceedings of the 40th international conference on software engineering: software engineering in practice, pp. 124–133

Kopp, O., Snethlage, C. C., & Schwentker, C. (2023). JabRef: BibTeX-based literature management software. *TUGboat, 44*(3), 441–447.

Li, Z., Yu, Y., Yin, G., Wang, T., Fan, Q., Wang, H. (2017). Automatic classification of review comments in pull-based development model. In: Proceedings of the 29th international conference on software engineering and knowledge engineering, pp. 572–577

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. arXiv:1907.11692

Mäntylä, M. V., & Lassenius, C. (2009). What types of defects are really discovered in code reviews? *IEEE Transactions on Software Engineering, 35*(3), 430–448.

Martin Borčin, J.M.J. (2024). Optimizing BERTopic: Analysis and reproducibility study of parameter influences on topic modeling. In: Proceedings of the 46th european conference on information retrieval, *14611*. Springer, ???

McConnell, S. (2004). *Code Complete*. Boston: Pearson Education.

McInnes, L., Healy, J., Saul, N., & Großberger, L. (2018). Umap: Uniform manifold approximation and projection. *Journal of Open Source Software, 3*(29), 861.

McIntosh, S., Kamei, Y., Adams, B., Hassan, A.E. (2014). The impact of code review coverage and code review participation on software quality: A case study of the qt, vtk, and itk projects. In: Proceedings of the 11th working conference on mining software repositories, pp. 192–201

Nuñez-Varela, A. S., Pérez-Gonzalez, H. G., Martínez-Perez, F. E., & Soubervielle-Montalvo, C. (2017). Source code metrics: A systematic mapping study. *Journal of Systems and Software, 128*, 164–197.

Ochodek, M., Staron, M., Meding, W., Söder, O. (2022). Automated code review comment classification to improve modern code reviews. In: Proceedings of the 14th international conference on software quality, pp. 23–40

Olsson, T., Ericsson, M., Wingkvist, A. (2017). The relationship of code churn and architectural violations in the open source software jabref. In: Proceedings of the 11th european conference on software architecture: companion proceedings, pp. 152–158

OpenAI. (2023). GPT-4 Technical Report. arXiv:2303.08774 [cs]

Paixao, M., Krinke, J., Han, D., Ragkhitwetsagul, C., & Harman, M. (2019). The impact of code review on architectural changes. *IEEE Transactions on Software Engineering, 47*(5), 1041–1059.

Panichella, A. (2021). A Systematic Comparison of search-Based approaches for LDA hyperparameter tuning. *Information and Software Technology,130*, 106411

Panichella, A., Dit, B., Oliveto, R., Di Penta, M., Poshynanyk, D., De Lucia, A. (2013) How to effectively use topic models for software engineering tasks? an approach based on genetic algorithms. In: Proceedings of the 35th international conference on software engineering, pp. 522–531

Papadakis, N., Patel, A., Gottigundala, T., Garro, A., Graham, X., Da Silva, B. (2020). Why Did your PR Get Rejected?: Defining Guidelines for Avoiding PR Rejection in Open Source Projects. In: Proceedings of the IEEE/ACM 42nd international conference on software engineering workshops, pp. 165–168

Qiang, J., Qian, Z., Li, Y., Yuan, Y., & Wu, X. (2020). Short text topic modeling techniques, applications, and performance: a survey. *IEEE Transactions on Knowledge and Data Engineering, 34*(3), 1427–1445.

Röder, M., Both, A., Hinneburg, A. (2015). Exploring the space of topic coherence measures. In: In Proceedings of the eighth ACM international conference on web search and data mining, pp. 399–408

Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering, 14*, 131–164.

Schneider, N., Shouei, S., Ghantous, S., Feldman, E. (2023). Hate Speech Targets Detection in Parler using BERT. In: In proceedings of the 6th workshop on online abuse and harms

Silva, C.C., Galster, M., Gilson, F. (2024). Applying short text topic models to instant messaging communication of software developers. *Journal of Systems and Software*, 112111

Silva, C. C., Galster, M., & Gilson, F. (2021). Topic modeling in software engineering research. *Empirical Software Engineering, 26*(6), 1–62.

Souza, L. B., Campos, E. C., Madeiral, F., Paixão, K., Rocha, A. M., & Almeida Maia, M. (2019). Bootstrapping cookbooks for apis from crowd knowledge on stack overflow. *Information and Software Technology, 111*, 37–49.

Steinmacher, I., Silva, M. A. G., Gerosa, M. A., & Redmiles, D. F. (2015). A systematic literature review on the barriers faced by newcomers to open source software projects. *Information and Software Technology, 59*, 67–85.

Sun, X., Li, B., Leung, H., Li, B., & Li, Y. (2015). Msr4sm: Using topic models to effectively mining software repositories for software maintenance tasks. *Information and Software Technology, 66*, 1–12.

Tang, J., Zhang, M., Mei, Q. (2013). One theme in all views: modeling consensus topics in multiple contexts. In: In Proceedings of the 19th ACM SIGKDD international conference on knowledge discovery and data mining, pp. 5–13

Thongtanunam, P., Tantithamthavorn, C., Kula, R.G., Yoshida, N., Iida, H., Matsumoto, K.-i. (2015). Who should review my code? a file location-based code-reviewer recommendation approach for modern code review. In: Proceedings of the IEEE 22nd international conference on software analysis, evolution, and reengineering (SANER), pp. 141–150

Turzo, A.K., Faysal, F., Poddar, O., Sarker, J., Iqbal, A., Bosu, A. (2023). Towards Automated Classification of Code Review Feedback to Support Analytics. In: Proceedings of the 17th ACM/IEEE international symposium on empirical software engineering and measurement (2023)

Udupa, A., Adarsh, K.N., Aravinda, A., Godihal, N.H., Kayarvizhy, N. (2022). An Exploratory Analysis of GSDMM and BERTopic on Short Text Topic Modelling. In: Fourth international conference on cognitive computing and information processing, pp. 1–9

Unterkalmsteiner, M., Badampudi, D., Britto, R., Ali, N.B. (2024). Help me to understand this commit! - A vision for contextualized code reviews. CoRR arXiv:2402.09528

Vassallo, C., Panichella, S., Palomba, F., Proksch, S., Gall, H. C., & Zaidman, A. (2020). How developers engage with static analysis tools in different contexts. *Empirical Software Engineering, 25*, 1419–1457.

Wang, Q., Xia, X., Lo, D., & Li, S. (2019). Why is my code change abandoned? *Information and Software Technology, 110*, 108–120.
Wen, R., Lamothe, M., McIntosh, S. (2022). How does code reviewing feedback evolve?: A longitudinal study at Dell EMC. In: Proceedings of the 44th international conference on software engineering: software engineering in practice, pp. 151–160
Xu, Y., Yin, Y., & Yin, J. (2017). Tackling topic general words in topic modeling. *Engineering Applications of Artificial Intelligence, 62*, 124–133.
Yin, J., Wang, J. (2014). A dirichlet multinomial mixture model-based approach for short text clustering. In: Proceedings of the 20th international conference on knowledge discovery and data mining, pp. 233–242

Umar Iftikhar is a Ph.D. candidate at the Blekinge Institute of Technology (BTH), Sweden. His research interests include source code quality and the analysis of code review artifacts to support practitioners in improving software quality and software processes.

Jürgen Börstler is a professor of software engineering at Blekinge Institute of Technology (BTH), Sweden. He received a Ph.D. in computer science from Aachen University of Technology (RWTH), Germany. Prof. Börstler is a member of SERL-Sweden, the Software Engineering Research and Education Lab at BTH and a senior member of the Swedish Requirements Engineering Network. Furthermore, he is a founding member of the Scandinavian Pedagogy of Programming Network. His research interests are in behavioral software engineering, research methods, software process improvement, software quality, program comprehension, and computer science education.

Nauman Bin Ali is a senior lecturer at Blekinge Institute of Technology. He is involved in empirical research in the field of software engineering. His research interests include using AI and ML for software engineering, lean software development, software testing, software process simulation, software metrics, and evidence-based software engineering. He received his Ph.D. (2015) in software engineering from Blekinge Institute of Technology, Sweden.

Dr. Oliver Kopp researches the intersections of software engineering, business process management (BPM), cloud computing, and IoT, contributing to both academia and industry. Currently, he researches in the field of software-defined vehicles. In the open-source community, he actively maintains and contributes to several projects, including JabRef and the Markdown Architectural Decision Records.