Tracking the decisions to select repositories for Mining Software Repositories experiments

Hiero Henrique Barcelos Costa[1], **Guilherme Marques de Oliveira**[1], **Victor Souza Salles**[1], **Gleiph Ghiotto Lima Menezes**[1]

[1]Departamento de Ciência da Computação – Universidade Federal de Juiz de Fora (UFJF) – Juiz de Fora – MG – Brazil

hiero.costa@ice.ufjf.br, guilhermemarques1@ice.ufjf.br, victor.salles@estudante.ufjf.br, gleiph@ice.ufjf.br

Abstract. *Mining Software Repositories analyzes and cross-links the data available in software repositories. This enables MRS to recognize patterns in software repositories. For example, to study how developers resolve conflicting merges. However, two main problems exist in the selection process of repositories: the limitations presented in traditional approaches used when selecting repositories and the lack of a systematic process for choosing repositories, turning off the experiments' reproducibility. This approach is proposed to resolve identified limitations and assist users in software repositories' selection. Initial results show that this approach returns at least 1.8 times more repositories, overcoming, for instance, the main language restriction in searches.*

1. Introduction

The Mining Software Repositories (MSR) field analyzes and cross-links the rich data available in software repositories to uncover interesting and actionable information about software systems and projects [4]. For example, previous work on MSR collected information on how developers resolve merge conflicts [2] and the Java language usability[6].

Selecting the repositories is a challenging task due to the characteristics desired for each experiment. For example, one researcher would filter the repositories considering the programming language and the number of commits, while others would filter the repositories considering the number of stars and the database type.

According to the literature, tools like *ModelMine* [8], *GHTorrent* [3], and *GitHub Advanced search*[1] are proposed to support repositories selection. However, they have limitations like the limit of the 1000 repositories for each search, the lack of some meaningful metadata (*e.g.,* the number of *commits* and secondary languages), and the existence of a database that is not able to keep its data up to date, making researchers use outdated information for their studies[10].

Another limitation is the deficiency of information about the rationale used to select the repositories, which would be helpful in reproducing the experiments. According to Vidoni [9], 83% of the papers on MRS do not have a systematic process for choosing repositories, whereas 37% of the total do not even show what was done in the selection of

[1]https://github.com/search/advanced

the repositories. This stunts the experiments' reproducibility, since the criteria to include or exclude the repositories from the experiments' corpus are not stored during the process of selecting repositories, which complicates other researchers to understand the decisions previously made.

This paper proposes an approach that overcomes previous limitations through the fragmentation of the searches, enriching them with other metadata and creating criteria that can be associated with each repository to store the selection rationale. Thus, this approach can support the selection of software repositories for sociotechnical studies to help the construction of knowledge in all fields of knowledge, and, in particular, in the field of Information Systems [1].

This paper is composed of five sections, including the Introduction. Section 2 exposes the overview of the state of the art in Mining Software Repositories (MSR) and the tools found in the literature for selecting software repositories. Section 3 introduces the proposed approach with the features used to overcome the highlighted limitations of previous approaches. Section 4 shows the current state of the approach's implementation and feasibility analysis. Finally, Section 5 presents the main contributions and future work.

2. Background

The selection of repositories for MSR studies is supported by tools such as *ModelMine*, *GHTorrent*, and *GitHub Advanced Search*. *ModelMine* [8] is designed to extract code files and *commits* details from *Git* repositories stored in *GitHub* and *GitLab* platforms. In addition, several filter techniques are integrated into it, making it possible to filter repositories by programming language and/or other metadata like the repository size, popularity (*i.e.*, stars' number), creation date, and the date of the last time it was changed. *GHTorrent* [3] is a scalable, queryable, and offline database that tries to mirror the GitHub's data. However, *GHTorrent* has a delay in the data collection, resulting in out-of-date information. Finally, *GitHub Advanced Search* expands the information that can be used to query it (*i.e.*, owners, date of creation, and number of comments in issues). This expansion permits a more detailed search for *Git* repositories and other information in *GitHub*.

The tools present in the literature have limitations in the amount of returned repositories, the lack of metadata, and out-of-date information. The limitation to 1000 repositories returned per search restricts a search for the repositories that have a number of stars greater or equal to 10,000, which returns approximately 3,000 repositories on March 9, 2024. The lack of some important metadata (*e.g.*, secondary languages and the number of *commits*) limit the search to repositories considering only the main language and excluding repositories that have a significant amount of code in a secondary language. Finally, the out-of-date information in the database can lead to answers that do not reflect the current state of the repository.

According to Vidoni [9], 83% of the papers on MRS do not have a systematic process for choosing repositories, whereas 37% of the total do not even show what was done to select repositories for the studies. The author highlights that it can be a result of the lack of frameworks or methodologies to guide the process of collecting information from repositories.

To improve the reliability of MSR studies, Vidoni proposed seven guidelines to

be followed that are divided into three groups: planning, execution, and results. The planning is divided into three guidelines: G1 which defines the research scope through the objectives and the context in which MSR is applied; G2 which describes the sources' details, which encompasses all the steps to select the sources where searches for software repositories will be executed; and G3 that describes the process and the criteria for repositories selection and their evaluation. The execution comprises two guidelines: G4 comprises the registration of the process to select repositories, reporting the evaluation of the returned repositories; and G5 describes the process of mining data from the repositories. Finally, the results contain two guidelines: G6 which consists of the extraction of relevant information from the already processed and organized data for the MSR study, and G7 which guides the presentation of the results that were obtained.

To the best of our knowledge, there is no approach that supports Vidoni's approach integrally or partially. Thus, the proposition of approaches that would help researchers follow the proposed Guidelines would support them during the MSR process, especially in the selection of repositories, which can have many criteria and results to be reported.

3. Approach's Proposition

This paper proposes an approach to resolve the limitations found in the literature: the limit of 1000 results per search, the restriction of metadata in the selection process, the out-of-date date, and the lack of traceability of the criteria used to select the repositories of the corpus' experiment. To do so, the approach fragments the queries that return more than 1,000 repositories and stores the decisions made by the user during the selection of the resulting repositories.

The approach enables researchers to define the research objectives and the platform in which the searches will be performed. During the scope definition, the researcher is able to register the research objectives, which can be from a single to multiple objectives. The researcher is able to choose the platform in which the search will be performed.

After establishing the definitions and criteria for repository selection, the approach starts to search for the repositories that match the restriction. Figure 1 shows an overview of the process that comprises four phases: (1) the definition of criteria to select repositories, (2) the fragmentation of searches, (3) the selection of repositories, and (4) the enrichment of the repositories' metadata with researches data.

The definition of criteria to select repositories starts with the definition of the metadata that will be used during the selection of repositories. For instance, one researcher can define that she is interested in repositories that have more than 10,000 stars and that the Java language is present. Thus, the approach will treat the search string to overcome the platform limitations. For instance, using the *GitHub* platform as an example, the search will restrict the repositories considering only stars and then will extract the languages present in the repositories to select the ones that have Java language.

The fragmentation of searches is applied when the amount of returned repositories is greater than the limit of results returned by the platform's API. The process of fragmentation uses the metadata of the platforms to limit the number of repositories returned to the limit of the platform's API. Thus, the approach performs multiple searches on the platform and stores the results of each search in a local database. Considering *GitHub*

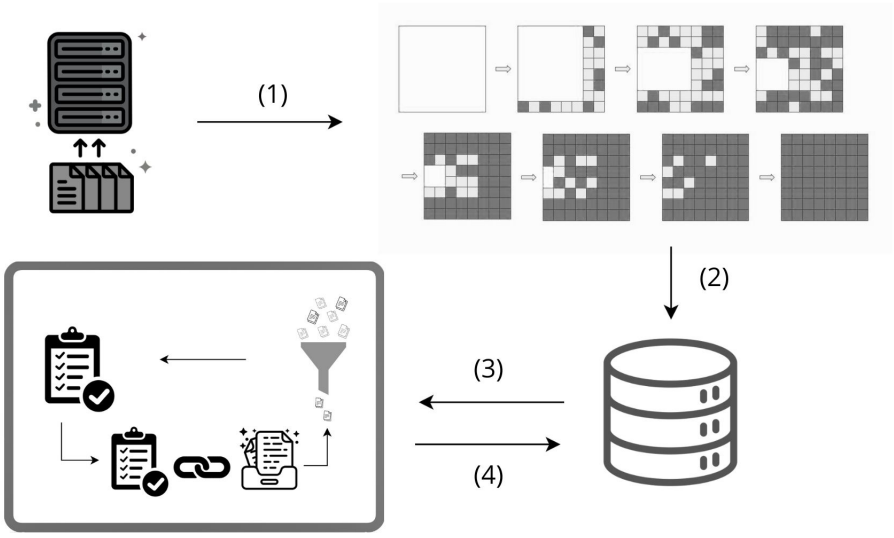

Figure 1. Approach operating phases

platform, the fragmentation is performed until the fragments result in groups of less than 1,000 repositories. The metadata used for this process of fragmentation comprises the number of stars, the date on which the repository was created, and the size spent in the disk of the repository.

The selection of repositories is performed by the researchers after the fragmentation starts to save the results in the database. Once the repositories are saved in the database, the researcher is able to see the repositories' metadata as the number of stars, the number of commits, the number of watchers, and so on. Thus, the researcher can start to filter the resulting repositories, considering any of the metadata stored locally.

The enrichment of the repositories' metadata with researchers' data is the phase in which other information can be stored in the database, which can be available for future searches and ongoing analysis. While the researcher is selecting the repositories based on the data stored locally, she is able to access other data on the platform's website, extract information that is not present in the platform database, and store the information locally. For instance, the researchers interested in projects that use a specific API will be able to verify this and record this in the repository information from the database's approach. One example is that the researcher can add a piece of information that the repository uses *Spring Boot*[2] and this can be shared with other researchers who will be able to use that information.

Since the researcher finishes the four phases, she is able to select the repositories that match her goals. For instance, considering *GitHub*'s metadata and the use of *Spring Boot*, the researcher can define the criteria that include and exclude the repository from the final set of repositories. A possible case would be the repositories that have more than 10,000 stars and more than 10,000 commits (inclusive criteria) and do not use *Spring Boot* (exclusive criteria). Thus, all the steps used to select the repositories would be tracked, enabling experiments' reproducibility.

[2]https://spring.io/projects/spring-boot

4. Actual state

To enable an initial evaluation of the proposed approach, we implemented a proof-of-concept capable of capturing repositories from the *GitHub* platform. *GitHub* is the most famous platform of open source code and, since 2016, it has already over 10 million *Git* repositories which should be enough for the range of the search [7, 5]. Moreover, *GitHub* has the *GitHub API* V4 with *GraphQL*, which is used to collect the repositories from *GitHub* the database.

A feasibility experiment was performed on the current implementation, considering three scenarios. All the scenarios consider repositories that have the Java programming language with the number of stars being greater than 5,000, 10,000, and 50,000 for each scenario. Since the time can change for each execution, due to non-deterministic factors, each scenario was executed three times.

Table 1 shows the result extracted for the three scenarios comparing the results obtained by this approach to the results of the *GitHub Advanced search*. The column *Scenario* identifies each experiment by the number of stars since all the scenarios use Java language. The other columns show the average **Time** in minutes spent on each scenario, the amount of *Retrieved repositories* from ***GitHub*** considering only the number of stars, the number of repositories returned by our **Approach**, and the number of repositories returned by the ***GitHub*** *Advanced search*.

Scenario	Time(min)	Retrieved repositories	Approach	GitHub
>5,000 stars	29:34	8,343	993	549
>10,000 stars	8:48	3,436	401	217
>50,000 stars	2:35	224	27	12

Table 1. Results of the searches conducted on repositories that contain Java language and different numbers of stars on March 9, 2024

The results show that the number of repositories returned by this approach is higher than the number of repositories returned by the ***GitHub*** *Advanced search* in 1.8 times. Considering the scenario with more than 5,000 stars, the average time spent was 29:34 resulting in 8,343 repositories returned from *GitHub* considering only the stars criteria. It takes place because the *GitHub API* considers only the main language to select repositories. Thus, other repositories that have Java code in the second language would be excluded from the analysis. The 8,343 repositories are analyzed considering the Java language as a filter, which returns 993 repositories. Comparing the result to the *GitHub*, our approach returns 1.8 times repositories, this ratio increases for the other scenarios.

5. Conclusion

This paper presents an approach to support MSR studies in the step of selecting repositories by tracking the steps performed by researchers. It supports the definition of the research goals, the collection of huge amounts of data from platforms that store repositories, and the collection of the rationale to reach a set of repositories. The rationale is one of the main points considering reproducibility, which is neglected in 83% of the papers on MRS [9].

This approach supports the set of Guidelines proposed by Vidoni [9]. Considering the planning group, the approach is able to help in the Guidelines G1, G2 and G3. Regarding the execution, the approach supports the registration of the process to select the repositories (G4). Initial results show it is able to overcome limitations like the searches in which the researchers are not interested only in the projects' main language.

Some perspectives for future work are the insertion of the tool in a server and its introduction on a Web platform on the Internet that allows public access; the creation of a kind of social network, enabling users their research and their data; and the study to compare the efficiency and the performance while performing searches.

Acknowledgements

The authors thank the support provided by CAPES and CNPq.

References

[1] Isabel Cafezeiro, José Viterbo, Leonardo Cruz da Costa, Luciana Salgado, Marcelo da Costa Rocha, and Rodrigo Salvador Monteiro. Strengthening of the sociotechnical approach in information systems research. *Sociedade Brasileira de Computação*, 2017.

[2] Gleiph Ghiotto, Leonardo Murta, Márcio Barros, and André van der Hoek. On the nature of merge conflicts: A study of 2,731 open source java projects hosted by github. *IEEE Transactions on Software Engineering*, 46(8):892–915, 2020.

[3] Georgios Gousios. The ghtorent dataset and tool suite. In *2013 10th Working Conference on Mining Software Repositories (MSR)*, pages 233–236, 2013.

[4] Ahmed E. Hassan. The road ahead for mining software repositories. In *2008 Frontiers of Software Maintenance*, pages 48–57, 2008.

[5] Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M German, and Daniela Damian. An in-depth study of the promises and perils of mining github. *Empirical Software Engineering*, 21:2035–2071, 2016.

[6] Mark J Lemay. Understanding java usability by mining github repositories. In *9th Workshop on Evaluation and Usability of Programming Languages and Tools (PLATEAU 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[7] Victor A. Luzgin and Ivan I. Kholod. Overview of mining software repositories. In *2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, pages 400–404, 2020.

[8] Sayed Mohsin Reza, Omar Badreddin, and Khandoker Rahad. Modelmine: A tool to facilitate mining models from open source repositories. In *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, MODELS '20, New York, NY, USA, 2020. Association for Computing Machinery.

[9] M. Vidoni. A systematic process for mining software repositories: Results from a systematic literature review. *Information and Software Technology*, 144:106791, 2022.

[10] Abdulkadir Şeker, Banu Diri, Halil Arslan, and Mehmet Amasyali. A systematic mapping of software engineering challenges: Ghtorrent case, 03 2020.