# STRESS: A Semi-Automated, Fully Replicable Approach for Project Selection

Davide Falessi
California Polytechnic State University
USA
Email: dfalessi@calpoly.edu

Wyatt Smith
California Polytechnic State University
USA
Email: wsmith@calpoly.edu

Alexander Serebrenik
Eindhoven University of Technology
The Netherlands
Email: a.serebrenik@tue.nl

***Abstract*—Background: The mining of software repositories has provided significant advances in a multitude of software engineering fields, including defect prediction. Several studies show that the performance of a software engineering technology (e.g., prediction model) differs across different project repositories. Thus, it is important that the project selection is replicable. Aims: The aim of this paper is twofold: 1) We investigate the possibility to replicate the project selection in previous studies, 2) We provide a semi-automated and fully replicable solution called STRESS. STRESS is a tool that allows researchers to select projects by configuring the desired level of diversity, fit, and quality. STRESS records the rationale behind the researcher decisions and allows different users to re-run or modify such decisions. STRESS is open-source and it can be used locally or even online (www.falessi.com/STRESS/). Method: We perform a systematic mapping study that considers studies that analyzed projects managed with JIRA and Git to asses the project selection replicability of past studies. We validate the feasible application of STRESS in realistic research scenarios by applying STRESS to select projects among the 211 Apache Software Foundation projects. Results: Our systematic mapping study results show that none of the 68 analyzed studies is completely replicable. Regarding STRESS, it successfully supported the project selection among all 211 ASF projects. It also supported the measurement of 100 projects characteristics, including the 32 criteria of the studies analyzed in our mapping study. Conclusions: The mapping study and STRESS are, to our best knowledge, the first attempt to investigate and support the replicability of project selection. We plan to extend them to other technologies such as GitHub.**

***Index Terms*—mining software repositories, replication**

## I. Introduction

### A. Context

Mining software repositories (MSR) provided significant and valuable advances in a multitude of software engineering fields such as defect prediction [1], traceability recovery [2], technical debt management [3], technology adoption [4] and socio-technical analysis [5], [6].

In order to avoid misunderstandings, we now define the terms project and repository in the context of this paper. A *project*, such as Accumulo[1], is a set of software engineering activities that produce different artifacts such as code, requirements, bugs, tasks, and their traces. Some of these artifacts are managed and linked with issue control systems, such as JIRA or BugZilla, and with version control systems, such as Git or Subverison. Systems that store project data, e.g., issue control systems and version control systems, are known as software *repositories*. In the context of this paper, a project typically has two repositories, an issue tracking system (e.g., JIRA) and a version control system (e.g., Git). In other studies, such as the ones focusing on GitHub, a project may have only one repository, since GitHub can act as both the version control and the issue tracking system.

A crucial step for mining software repositories is choosing the projects to mine. In the linguistic context, Hunston reported the most important characteristics for a corpus are size, contents, representativeness and permanence [7]. Tempero et al. have argued that the same concerns pertain to software corpora [8]. Regarding the number of projects Nagappan et at. [9] state that "more is not necessarily better."

### B. Problem

The mining results are frequently related to the projects where they have been observed [1], [10]. For instance, Hall et al. [1] suggest that some application domains (e.g., embedded systems) may be more difficult than others to build reliable prediction models for. Similarly, within-company effort predictions are usually more accurate than cross-company ones [11]. Thus, project selection affects both external and internal validity [12], [13].

Regarding external validity, it is important that the researchers provide a good description of the used projects so that the reader can reason on how much the results are generalizable on other contexts. Regarding internal validity, in most of the cases the technology validation group has relation with the technology development group and hence the validation group might have interest in the output of the validation, a.k.a. conflict of interests. Researchers may consciously, or even unconsciously, prefer projects showing that their new technology is reliable [2] or it even outperforms past ones. As reviewers, editors, and readers, we have no way to replicate the project selection, and, hence, cannot know whether it could have been biased. As authors, in case of a conscious project selection focused on not biasing the results, we are in trouble when solid justifications for the chosen projects should be provided.

Past studies provided tools [14] and algorithms [9] to support researchers in project selection. However, our perception is that the project selection, in past mining papers, is hard to

[1]http://accumulo.apache.org/

replicate. Our mapping study will confirm such a perception. Indeed, replication of previous studies is essential in empirical software engineering research as a means to gain confidence and to understand the limitations of research results [15].

### C. Aim

The **aim** is twofold: 1) We investigate the possibility to replicate the project selection in previous studies, 2) We provide a semi-automated and fully replicable solution called STRESS. STRESS is a tool that allows researchers to select projects by configuring the desired level of diversity, fit, and quality. A systematic mapping study of previous MSR studies analyzing projects managed with JIRA and Git allowed us 1) to investigate the possibility to replicate the project selection in previous studies and 2) to decide upon the project selection criteria to be supported in STRESS we perform We validate feasibility of application of STRESS in realistic research scenarios by 1) applying STRESS to select projects among the 211 Apache Software Foundation (ASF) projects[2] and 2) checking whether STRESS supports the project selection criteria adopted in the past.

### D. Structure

After discussing the related work in Section II, in Section III we report the mapping study design and results. Section IV presents STRESS. Section V reports the threats to validity and Section VI concludes the paper.

## II. Related work

### A. Replicability of MSR studies

Amann et al. [16] revisited a decade of software mining studies and highlighted mining goals, study replicability, and trends in mined artifacts. They reported that only 40% of the studies provided their datasets for reuse and only 20% of the studies made the tools actually available. Ghezzi and Gall [17] provided a distributed and collaborative software evolution analysis platform for supporting Software Analysis as a Service (SOFAS). They concluded that the replication of MSR studies is still at a rather early stage despite the fact that the replication is just as fundamental as the studies themselves [17]. Robles [18] analyzed the replicability of 171 MSR papers in terms of project data, processed data, and tool availability. He found that data and tool are frequently unavailable, even in cases where studies reported otherwise.

We believe hence that no previous study has investigated the replicability of, or the criteria used for, project selection.

### B. Tool support for project selection

Nagappan et al. [9] provided an algorithm to choose the projects by maximizing the sample coverage, i.e., the percentage of projects in a population that are similar to a given sample. We reused and extended their approach. Specifically, we improved the usability by encapsulating their algorithm in a website with a GUI and with a database of ASF projects that can be analyzed, and then selected for a follow-up study. STRESS aims at improving the usability of the work of Nagappan et al. [9] without decreasing the scientific rigor.

Rozenberg et al. [14] recently presented a tool called RepoGrams to support researchers in qualitatively comparing and contrasting projects over time using a set of software metrics. We share with them the vision to support researchers in selecting projects to make more reasoned choices. STRESS differs from RepoGrams in 1) supporting a quantitative, rather than qualitative, approach and 2) in supporting replication.

Munaiah et al. [19] proposed a tool called Reaper, to enable researchers to select GitHub repositories that contain evidence of an engineered software project. Reaper is similar to STRESS in the sense that it aims at supporting researchers in selecting projects. However, STRESS focuses and allows replicability. The advantage of Reaper over STRESS is that it allows the user to chose among GitHub projects which are several order of magnitude more numerous than ASF projects. The disadvantage of Reaper is that it may lead to chose projects that are not engineered.

Hence, while there are studies supporting project selection, no study investigated the replication of project selection.

## III. Systematic mapping study

The mapping study addresses two research questions:

- RQ1: Is project selection of past studies replicable?
- RQ2: Which project selection criteria have been used?

### A. Design

#### 1) Research Questions:

*RQ1: Is project selection of past studies replicable?:* The aim of STRESS is to support a fully replicable project selection. It is therefore important to understand if and why past studies do not allow the replication of their project selection. In order to be replicable, a project selection strategy must be documented, based on measurable project characteristics, and complete, i.e. consider all projects satisfying the selection criteria. Moreover, the time when the projects characteristics, related to the selection criteria, are measured should be reported. We structure the following sub-research questions to investigate each of the above replicability conditions.

*RQ1.1: Are criteria reported?:* Obviously the researchers must report the criteria to allow other researchers to replicate their application in project selection.

*RQ1.2: Are criteria measurable?:* Some criteria can be more vague and subjective than others. For instance, the domain of a project can be subject to interpretation. However, the same domain criterion can be objective and formally measured if it is defined as a set of measurable characteristics such as LOC or specific technology in use.

*RQ 1.3: Is the search complete?:* We differentiate between studies making sure that the selected projects have specific characteristics representing some part of the corpus having those characteristics, as opposed to studies that analyzed an entire population and selected all the projects having those characteristics. Only in the latter case can the researchers

[2]https://projects.apache.org/

claim the absence of bias in project selection; i.e., they used a quantifiable and replicable project selection strategy.

*RQ1.4: Is the time of the project characteristics measurment reported?:* Because projects evolve over time, their characteristics, such as size (number of commits) or technology in use (e.g., programming language), or even their availability, can change over time. It is therefore important to report the specific time when the projects characteristics, related to the decision criteria, have been measured.

*RQ2: Which project selection criteria have been used?:* This question investigates the specific criteria used. This investigation is important because in STRESS we want to support the criteria used in previous studies.

*2) Research Method:* The systematic mapping study described in this section has been carried out in the period June 2016–March 2017. We adhere to the guidelines for conducting systematic literature reviews as formulated by Kitchenham and Charters [20]. The purpose of our study is to characterize the criteria used by MSR studies.

In particular, we focus on scientific studies of software projects using JIRA and Git, and on project selection criteria used in these studies. The main reason is that Jira and Git are commonly considered by the bug prediction community; it has been acknowledged that the bug reports in Jira are better linked to the version control systems than in Bugzilla [21], [22], [23]. Finally, we decided to focus on Jira and Git since our industry collaborators and colleagues, such as Keymind [24], tend to use this combination to manage their software projects.

To identify the previous studies, we perform a search on Google Scholar using "JIRA Git" as the search string. We opt for Google Scholar since compared to traditional collections of scientific papers such as Web of Science, IEEEXplore, ACM DL or Springer Link, it provides the most complete coverage of scientific literature available and does not suffer from the idiosyncrasies reported earlier for traditional collections [25]. Finally, the work on JIRA started in 2002, i.e., scientific publications preceding this date do not refer to the issue control system but, e.g., have been authored by a researcher with a surname Jira. Hence, we exclude papers predating 2001.

Next, we exclude some of the scientific studies retrieved using Google Scholar. We start by applying the practical screen [26]: the content covered should be related to software (as opposed to, e.g., JIRA being a surname of one of the authors) and the language of publication should be English. Next, we apply the methodological quality screen [26]. We require that the publication should be peer-reviewed. We assume workshop/conference papers, journal and magazine articles to be peer-reviewed. Theses, technical reports, manuals, books, and book chapters *might* be peer-reviewed but are not *necessarily* peer-reviewed. Therefore, we exclude theses, technical reports, manuals, books, and book chapters from consideration. We also exclude papers that do not report empirical studies of JIRA repositories as opposed to, e.g., surveys with participants reporting that they use JIRA. We do not, however, restrict the purpose of the study: while our work has been motivated by the empirical research on defect prediction, we do bot exclude empirical studies of other subjects. Finally, we also exclude the MSR studies that used only one project as those studies are typically case studies and are not intended to be generalizable.

Application of the inclusion and exclusion criteria has been performed by the third author. Since application of the criteria by one researcher only might threaten the reliability of the study [27] the final set of primary studies has been reviewed and analyzed by the first author.

### B. Data Extraction and Synthesis

In order to answer the research questions we read the papers and look for an indication whether selection of specific projects has been explained (*RQ1.1*), whether the study reports how the selection criteria have been measured (*RQ1.2*), and when have the selection criteria have been applied (*RQ1.4*).

As opposed to those research questions, *RQ1.3: Is the search complete?* and *RQ2: Which criteria have been used?* are more "open" in nature. Hence, when studying those questions we tagged the information derived from the papers.

In order to answer *RQ1.3*, we searched for explicit indication whether the authors included all the projects satisfying the selection criteria or merely some such projects. We classify each study according to the completeness of their search using one of the following tags: "complete", "uncertain", "incomplete" and "not applicable". In the "complete" cases the authors clearly included all projects that met some criteria. In the "uncertain" cases the authors did not clearly report whether the projects included are all projects in the population satisfying the selection criteria or only a casual sample of those. For example, a sentence like "We chose these projects because they satisfy the criterion" is ambiguous and, hence, not replicable when compared to a sentence like "We chose all projects that satisfy the criterion" which is unambiguous and replicable. In the "incomplete" cases where the authors clearly included projects that casually met some criteria. An example of the incomplete tag is a sentence like "We chose these projects among many others that satisfy the criterion". Finally, the tag "not applicable" relates to studies not reporting any criteria for choosing projects or datasets.

In order to answer *RQ2*, we have checked the criteria used by the primary studies and applied the open coding process [28]. Then we have looked for similarities and differences.

### C. Results

Searching for "JIRA Git" on Google Scholar returned 840 hits corresponding to studies published from 2001 onwards. Excluding studies not related to software engineering resulted in 774 hits, which were further reduced to 582 after exclusion of texts published in languages other than English. Since multiple hits might be related to the same primary study we have further excluded 9 duplicates. After applying the practical screen, we keep 573 primary studies.

Our decision to focus on peer-reviewed work eliminates 324 studies, leaving 249 for further consideration. Indeed, 144 excluded studies are bachelor, master or PhD theses; 98 are

books, chapters, and manuals. A large amount of non-peer reviewed work is inherent for Google Scholar. Among 249 primary studies, 75 cover empirical studies of projects that make use of JIRA. 7 out of the 75 primary studies report case studies involving one project only. We exclude these papers as they do not aim at generalization. Hence, 68 studies are kept for further investigation.

*RQ1.1: Are criteria reported?:* Our results show that **35% (24 out of 68) of the studies reported the criteria adopted for choosing the projects**.

*RQ1.2: Are criteria measurable?:* The 68 studies use a total of 32 different criteria, and some studies use more than one criterion. We classify the following 22 criteria (69%) as being measurable: artifact (requirements, defects, traces), age, organization, programming language, technology, active, quality (% Linkage), replica, size (revisions, columns, contributors, LOC, tables, years), API, hybrid repos, JAVA, JIRA, process (agile), and SQL. We classify the following 10 criteria (31%) as not being measurable: domain, functionality, maturity, randomness, stability, successfulness, being top-ranked (without specifying the metric), level of usage in software engineering studies, being well commented or well known.

Unfortunately the domain criterion is not replicable because the affinity of a project to a domain is subjective. For instance, the project Accumulo can be considered belonging to multiple domains such as API, Storage, and Back-end. Regarding randomness, unfortunately all three studies using random project selection fail to report the random number generation algorithm or the seed used. Finally, we note that no studies use the sampling strategy proposed by Nagappan et al. [9]. This could be due to the fact that this strategy has the Ohloh.net corpus as the native project target whereas our studies focus on projects managed with JIRA and Git. Among the 68 studies, **only 11 (16%) use criteria that are all measurable**, 13 studies (19%) use one or more non-measurable criterion, and 44 (65%) studies report no criteria.

*RQ1.3: Is the search complete?:* **Only six studies (9%) explicitly analyze all the projects before making the final selection**. In eight studies (12%) it is unclear if the search is complete, six studies have an incomplete search, and the search completeness condition does not apply because they do not report any rationale for choosing the projects or datasets. We conjecture that the low number of studies featuring a complete search can be due to reasons such as 1) lack of access to the complete projects corpus, 2) size of the projects corpus or 3) the need to develop special measurement tools merely to decide whether a project should be selected. Development of such tools, and application to projects that might not after all be selected, requires significant effort. For instance, a researcher cannot identify the top 10 projects ASF or GitHub without developing the code to do so and analyze all the projects in the corpus. Therefore, in most of the cases, the researcher reasonably stops as soon as a desired number of projects satisfy the defined criteria. This problem is solved by using STRESS; it provides the mechanisms to measure the needed metrics and the measurement resulted to require reasonable resources (time and hardware) for measuring all 211 ASF projects that are managed by JIRA and Git.

We note that the description of 12% of the studies is unclear on the completeness of the search. We believe STRESS would make many of these studies replicable by providing researchers with the possibility to export, and publish, the project selection rationale documentation.

*RQ1.4: Is the time of the search reported?:* **None of the analyzed papers report the time when the characteristics of the projects have been measured**.

*RQ2: Which criteria have been used? :* We have identified the following exclusive macro criteria categories:

- Artifact: It regards the type of artifacts developed and managed in the project such as bugs and their traces to requirements.
- Diversity: It regards the type of diversity of the sampled projects. For instance, researchers might apply a full random selection or might select projects to maximize the number of different programming languages used.
- Replica: It regards choices of projects due to the need to confirm or reject hypotheses on the exact same projects or the selection of projects already selected so that the related repositories are already computed and available.
- Size: It regards the dimension of the projects such as number of commits, LOC and number of tickets.
- Technology: It regards the technology used for developing the project such as the programming language.
- None: When no criterion is reported.
- Miscellaneous: It regards criteria different from the ones mentioned above.

Regarding the macro criteria used for project selection, **the most used criterion is Diversity (43%) followed by Miscellaneous (16%), Size (16%), and Technology (15%)**. Note that each criterion is counted every time it has been used by a study, even if a study used multiple criteria.

According to the specific criteria used in the studies, we decompose each of these macro-categories in several micro-categories. Regarding the macro criterion Diversity (16 studies total), the following micro criteria are used in one study (6%): Functionality, Maturity, Organization, Size (LOC), and Technology. Programming Languages is used in two studies (12%), Random in three studies (19%) and domain in five studies (31%). Regarding the macro criterion Miscellaneous (11 studies total), the only micro criterion used in two studies is Successfulness (18%).The following micro criteria are used in only one study (1%): Active, Maturity, Quality (Linkage), Stable, Topranked, Usage in SE, Well commented, and Well known. Regarding the macro criterion Size (11 studies total), the following micro criteria is used in one study: Columns, Revisions, and Tables. LOC is used in two studies (18%) and Contributors and Years in three studies (27%). We note that all criteria belonging to Size are measurable. Regarding the macro criterion Technology (11 studies total), JIRA is used in five studies (45%), Java in two studies (18%) and the following micro criteria in only one study (1%) API, process (agile), hybrid repos, SQL. We note that all studies use JIRA, but only

14% reported JIRA as an explicit criteria for project selection. Regarding the macro criterion Artifact (4 studies total), the related micro criteria are Requirements (50%), Defects (25%) and Traces (25%).

## IV. STRESS: A PROTOTYPE TOOL SUPPORT

Given the identified difficulties in replicating project selection (see Section III) we developed an open-source tool called STRESS (Semi-auTomated REplicable projectS Selection)[3], a semi-automated and fully replicable approach that allows researchers to select projects. Specifically, projects can be selected by configuring the desired level of diversity, fit, and quality. It records the rationale behind the researcher decisions and allows different users to re-run or modify such decisions. STRESS can be used locally or on-line[4].

Interaction with STRESS starts when a user can chose to load a previous project selection or to create a new one. If the user has chosen to create a new project selection, STRESS presents the list of projects characteristics it supports. STRESS currently supports 100 project characteristics including the 32 criteria of the studies analyzed in our mapping study. For example, Pat is a researcher and would like to choose all ASF projects satisfying the following criteria: 1) linkage higher than 80%, 2) number of bug tickets higher than 100, and 3) number of months tracked higher than 12. Thus, in the STRESS interface Pat selects linkage, number of bug tickets, and number of months tracked (see Fig. 1).

Next STRESS allows the user to set constraints on the characteristics of interest. Continuing with the previous example, Pat defines the following criteria: 1) linkage higher than 0.8, 2) number of bug tickets higher than 100, and 3) number of months tracked higher than 12. Fig. 2 provides a partial screen-shot of the criteria definition, as defined by Pat.

Afterward, the user is presented with the list of projects (rows) satisfying all the provided constraints and their characteristics (columns). Here the user can sort the projects according to a specific characteristic, by clicking on a column, in descending or ascending order. The user can now select the projects 1) manually, 2) fully randomly, or 3) by maximizing the sample representativeness. In case 1, the user can click on the checkbox next to the project name. In case 2, the user is asked to provide a seed (used by STRESS for random selection among the projects satisfying the criteria) and the number of desired projects. In case 3, the user is asked to provide the number of desired projects and STRESS samples them according to algorithm proposed by Nagappan et al. [9] among the ones satisfying the defined constraints. Thus, considering the previous example, Pat checks the three boxes and clicks on "Use Selection". Fig. 3 provides a partial screen-shot of the projects selection, as selected by Pat.

Finally STRESS shows the list of selected projects and allows the user to store the selection criteria as an external document, which can then be used to re-run or modify the selection choices by the current or subsequent researchers. Among 211 ASF projects, three satisfy the selection criteria used by Pat: Ambari, Syncope and UMIA.

[3]https://github.com/wyattjsmith1/STRESS

[4]http://www.falessi.com/STRESS


Fig. 1. STRESS: selection of projects characteristics of interest.

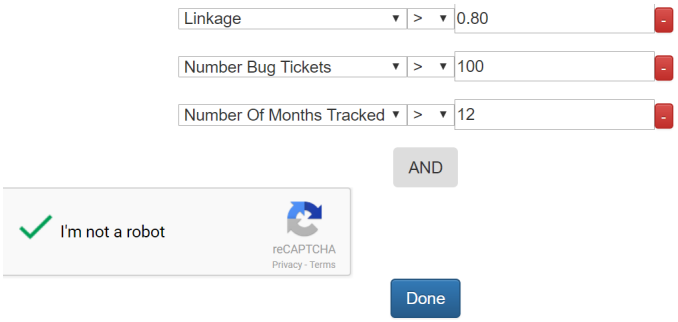


Fig. 2. STRESS: criteria definition.

## V. THREATS TO VALIDITY

In this section, we report the threats to validity related to our study. The threats are organized by type (i.e., Conclusion, Internal, Construct, and External).

We do not see any major conclusion validity threat related to the systematic mapping study.

The main internal threat to validity is the possible incorrectness of our replicability conditions (see Section III). To mitigate this threat we have validated these conditions by trying to replicate the project selection in the most replicable studies identified in the systematic mapping study.

Construct validity threats are more probable than the others. The first threat to construct validity concerns the tagging of the systematic mapping study and, more specifically, the possibility that we misinterpreted the natural language of the authors explaining the rationale for their project selection. In order to mitigate this threat, we analyze the rationale description of each of the studies at least three times. Moreover, we created a tag unclear when the natural language of the authors explaining the rationale for their project selection was prone to misunderstandings (see Section III-C). A similar threat is the possibility that we missed information provided in the papers. Again, in order to mitigate this threat, we analyze the rationale description of each of the studies at least three times. However, we cannot be sure that no information was been overlooked.

One of the major external validity threats is that we analyzed in our systematic mapping study only papers using the terms


Fig. 3. STRESS: projects selection.

JIRA and Git. We believe our systematic mapping study results cannot be generalized to studies using technologies different from JIRA and Git. Thus, we plan to expand this studies to include papers and projects managed with more technologies.

## VI. Conclusions

Replication is essential in any scientific field because it allows for an increase in confidence and adds to the understanding of study limitations. In this work, we focused on the replication of the project selection in studies mining software repositories. We performed a systematic mapping study and we considered studies analyzing projects that are managed with JIRA and Git. The study identified a lack of project selection replication; hence, we propose a solution called STRESS, a semi-automated and fully replicable approach that allows researchers to select projects in a replicable way. Specifically, STRESS records the rationale behind the researcher decisions and allows different users to re-run or modify such decisions. STRESS supports the measurement of 100 projects characteristics including the 32 criteria of the studies analyzed in our mapping study. STRESS can support the project selection among all 211 ASF projects.

The main limitation of this study is to have focused on specific technologies in both STRESS development and systematic mapping study. However STRESS and the mapping study are, to our best knowledge, the first attempt to support and investigate the replicability of project selection. We plan to extend the mapping study and STRESS to other technologies and repositories such as GitHub.

## VII. Acknowledgement

We thank Stacy Neely for proof reading the manuscript.

## References

[1] T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell, "A systematic literature review on fault prediction performance in software engineering," *IEEE Trans. Softw. Eng.*, vol. 38, no. 6, pp. 1276–1304, Nov 2012.

[2] D. Falessi, M. Di Penta, G. Canfora, and G. Cantone, "Estimating the number of remaining links in traceability recovery," *Empirical Softw. Engg.*, pp. 1–32, 2016.

[3] E. Maldonado, R. Abdalkareem, E. Shihab, and A. Serebrenik, "An empirical study on the removal of self-admitted technical debt," in *ICSME*. IEEE, 2017.

[4] Y. Zhao, A. Serebrenik, Y. Zhou, V. Filkov, and B. Vasilescu, "How do software engineering practices change following adoption of continuous integration?" in *ASE*. IEEE, 2017.

[5] D. Gachechiladze, F. Lanubile, N. Novielli, and A. Serebrenik, "Anger and its direction in collaborative software development," in *ICSE NIER*. IEEE, 2017, pp. 11–14.

[6] B. Lin, G. Robles, and A. Serebrenik, "Developer turnover in global, industrial open source projects: Insights from applying survival analysis," in *ICGSE*. IEEE, 2017, pp. 66–75.

[7] S. Hunston, *Corpora in Applied Linguistics*, ser. Cambridge Applied Linguistics. Cambridge University Press, 2002.

[8] E. Tempero, C. Anslow, J. Dietrich, T. Han, J. Li, M. Lumpe, H. Melton, and J. Noble, "The qualitas corpus: A curated collection of java code for empirical studies," in *APSEC*, Nov 2010, pp. 336–345.

[9] M. Nagappan, T. Zimmermann, and C. Bird, "Diversity in software engineering research," in *FSE*. New York, NY, USA: ACM, 2013, pp. 466–476.

[10] J. Keung, E. Kocaguneli, and T. Menzies, "Finding conclusion stability for selecting the best effort predictor in software effort estimation," *Automated Software Engineering*, vol. 20, no. 4, pp. 543–567, 2013.

[11] B. A. Kitchenham, E. Mendes, and G. H. Travassos, "Cross versus within-company cost estimation studies: A systematic review," *IEEE Trans. Softw. Eng.*, vol. 33, no. 5, pp. 316–329, May 2007.

[12] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering: An Introduction*. Kluwer Academic Publishers, 2000.

[13] J. Siegmund, N. Siegmund, and S. Apel, "Views on internal and external validity in empirical software engineering," in *ICSE*. IEEE, 2015, pp. 9–19.

[14] D. Rozenberg, I. Beschastnikh, F. Kosmale, V. Poser, H. Becker, M. Palyart, and G. C. Murphy, "Comparing repositories visually with repograms," in *MSR*. ACM, 2016, pp. 109–120.

[15] F. J. Shull, J. C. Carver, S. Vegas, and N. Juristo, "The role of replications in empirical softw. engg." *Empirical Softw. Engg.*, vol. 13, no. 2, pp. 211–218, Apr. 2008.

[16] S. Amann, S. Beyer, K. Kevic, and H. Gall, "Software mining studies: Goals, approaches, artifacts, and replicability," in *Software Engineering: International Summer Schools*, B. Meyer and M. Nordio, Eds. Springer International Publishing, 2015, pp. 121–158.

[17] G. Ghezzi and H. C. Gall, "A framework for semi-automated software evolution analysis composition," *Automated Software Engineering*, vol. 20, no. 3, pp. 463–496, 2013.

[18] G. Robles, "Replicating msr: A study of the potential replicability of papers published in the mining software repositories proceedings," in *MSR*, May 2010, pp. 171–180.

[19] N. Munaiah, S. Kroh, C. Cabrey, and M. Nagappan, "Curating github for engineered software projects," *PeerJ PrePrints*, vol. 4, p. e2617, 2016.

[20] B. Kitchenham and S. Charters, "Guidelines for performing Systematic Literature Reviews in Software Engineering," Keele University and Durham University Joint Report, Tech. Rep. EBSE 2007-001, 2007.

[21] A. Bachmann, C. Bird, F. Rahman, P. Devanbu, and A. Bernstein, "The missing links: Bugs and bug-fix commits," in *FSE*. ACM, 2010, pp. 97–106.

[22] C. Bird, A. Bachmann, E. Aune, J. Duffy, A. Bernstein, V. Filkov, and P. Devanbu, "Fair and balanced?: Bias in bug-fix datasets," in *ESEC/FSE*. ACM, 2009, pp. 121–130.

[23] R. Wu, H. Zhang, S. Kim, and S.-C. Cheung, "Relink: Recovering links between bugs and changes," in *ESEC/FSE*. ACM, 2011, pp. 15–25.

[24] D. Falessi, M. Shaw, and K. Mullen, "Achieving and maintaining cmmi maturity level 5 in a small organization," *IEEE Software*, vol. 31, no. 5, pp. 80–86, Sept 2014.

[25] D. Landman, A. Serebrenik, and J. J. Vinju, "Challenges for static analysis of java reflection literature review and empirical study," in *ICSE*. IEEE, 2017, pp. 507–518.

[26] A. Fink, *Conducting Research Literature Reviews: From the Internet to Paper*. SAGE Publications, 2010.

[27] K. Petersen, S. Vakkalanka, and L. Kuzniarz, "Guidelines for conducting systematic mapping studies in software engineering: An update," *Information and Software Technology*, vol. 64, pp. 1–18, 2015.

[28] M. Di Penta, "Combining quantitative and qualitative methods (when mining software data)," in *Perspectives on Data Science for Software Engineering*, T. Menzies, L. Williams, and T. Zimmermann, Eds. Boston: Morgan Kaufmann, 2016, pp. 205–211.