Open-Source Software Projects Curating Model for Empirical Software Engineering Studies

Juan Andrés Carruthers

Departament of Informatics – National University of the Northeast (UNNE)
Corrientes – Argentina

jacarruthers@exa.unne.edu.ar

Abstract. *Software projects are common inputs in Empirical Software Engineering (ESE), and they are often selected without following a specific strategy, leading to biased samples. To avoid this problem, researchers choose to use publicly available datasets instead of picking the projects themselves. However, some datasets are not maintained, containing old versions of projects, or even deprecated ones. This may raise some representativeness issues due to major changes in development practices and technologies over time. The main goal of this research is to develop a procedures model to construct and maintain a software project dataset with their product quality metrics, to support the development of ESE studies.*

1. Introduction

Software Engineering works mainly with the "construction of multi-version software applications" [Parnas 2001]. Thus, many of the activities associated with software application require revisions to improve the functionality or correct errors, especially in agile methodologies [Irrazábal et al. 2001]. Quality in software development can be studied from two angles: the software development process and the source code [Lehman 1996]. In the latter case, empirical methods are necessary to demonstrate the quality of the software [Kitchenham and Pfleeger 1996], using evidence related to the software product in form of metrics and indicators directly linked to quality [Garvin 1984].

The massive use of open source repositories (e.g. Github, SourceForge and Maven) made publicly available vast amounts of software projects data for researchers and software engineers to conduct Empirical Software Engineering (ESE) studies [Vidal et al. 2016], [Vázquez et al. 2019]. However, ESE studies performed with arbitrarily picked samples could lead to unrealistic and potentially inaccurate conclusions due to the proportion of noise in these repositories [Munaiah et al. 2017], contrasting with the ESE notions of replicable and generalizable results. In particular, replicability is essential not only for publications in prestigious journals or conferences of the discipline; but also for software development companies that often want to analyze the evolution of their own projects or use them as a benchmark in software audit reviews.

A common practice to demonstrate the effectiveness of metrics as predictors of software quality characteristics is the construction of datasets [Tempero et al. 2010]. Datasets of software projects work as inputs for researchers and as comparison mechanisms for different types of ESE studies. Examples of popular datasets are:

Qualitas corpus [Tempero et al. 2010], NASA dataset [Shepperd et al. 2013] or Jureczko and Madeyski dataset [Jureczko and Madeyski 2010], which their latest available version has more than seven years. Thus, it is necessary to review the versions of each project and recalculate their metrics.

With that being said, our hypothesis is "have a procedure model to construct, maintain and curate a dataset of software projects with their product quality metrics can provide a sound strategy to conduct ESE studies; improving replicability, experimental consistency, and flexibility for a controlled evolution of a software project dataset over time".

2. Research Goals

The main goal of our research is "develop a procedures model to construct, maintain and curate a dataset of software projects with their product quality metrics, to support the development of ESE studies". This objective is divided into two parts:

- Methodological support. Analyze and design rules to construct and curate a representative sample of open source software projects, and the metadata necessary to motivate ESE studies.
- Technological support. Develop tools to allow automated generation of metadata, to control rules for the curation of software projects, and to help researchers and engineers manage their use for the design and execution of experiments.

3. Related works

In ESE studies the results obtained must be, at some degree, replicable and generalizable. Replicability is important to validate the results of a scientific study and can be tackled by providing the datasets and scripts of the conducted research [Baltes and Ralph 2020].

There were attempts to provide a software project dataset for ESE studies over the years. These datasets include source code repositories [Tempero et al. 2010], information in issue trackers [Rahman and Roy 2018], product quality metrics [Jureczko and Madeyski 2010], [Shepperd et al. 2013], and even evolutionary data such as source code change logs [Palomba et al. 2018]. However, they are often unsuitable for the task due to methodological issues when sampling the software projects, reducing the generalization of their conclusions. The lack of representativeness in a sample can bias the generalization of the findings.

For example, Mockus et al. (2002) stated that free/open-source software contained fewer defects than proprietary systems, basing their conclusions on two open-source projects (Apache and Mozilla). Samples with two subjects cannot provide enough evidence to generalize to the entire population of open-source software.

Jureczko and Madeyski (2010) collected object-oriented product quality metrics and defect information from 48 releases of 15 open source projects and 27 releases of 6 proprietary projects. Shepperd et al. (2013) gathered size and complexity metrics from C and C++ Nasa software. In both cases, the authors provide access to the datasets and they are frequently used in current ESE studies. But, neither of them reported the sampling strategy.

Another popular dataset Qualitas Corpus [Tempero et al. 2010], contains the source code of 100 open-source Java projects and includes the guidelines followed to select the projects, keeping even deprecated ones. Using deprecated projects might guarantee the replicability of experiments but also raises issues about the applicability of the results obtained in current projects.

Few studies take into consideration building samples using a probability sampling technique, and they recur to convenience or purposive sampling techniques. The ideal case would be to randomly select a statistically significant sample of software projects, or maybe apply approaches such as Nagappan et al. (2013), to construct diverse and representative samples.

Finally, we identified scarce efforts to provide a framework or procedure model to maintain the temporal validity of the projects in the datasets. Since constant changes in development practices and technologies over time can produce different outcomes in ESE studies, a dataset of software projects constructed several years ago can rarely be representative of recently developed projects. Lewowski and Madeyski (2020) partially tackled this with their tool to create evolving datasets, but did not precise how to deal with deprecated projects.

4. Methodology

We organized the research strategy following the Design Science Research framework (Johannesson and Perjons, 2014), an approach to create and validate novel artifacts, providing solutions in the form of models, methods, and systems that support people in developing, using, and maintaining IT solutions. Also, it aims at producing and communicating new knowledge relevant for a global practice, thus the results produced are twofold, the artifact created, and the knowledge generated.

The purpose of creating new and generalisable knowledge requires that design science projects make use of rigorous research strategies and methods. Such methods are essential for creating results that can be critically discussed, evaluated, and validated. Finally, the new knowledge should be communicated to both practitioners and researchers.

With that being said, to achieve the main goal we propose the following tasks framed by the mentioned framework:

4.1 Explicate the problem

The goal of the first activity is to formulate the initial problem precisely, justify its importance, and investigate its underlying causes.

A1. Define the knowledge base and establish the state of the art on the datasets of software projects used in ESE studies.

The research methods used will be Systematic Literature Review [Kitchenham 2004] and Systematic Mapping Study [Petersen et al. 2008]. These methods identify, evaluate, and interpret all the information related to the particular research topic, in a systematic and replicable way. The application of systematic reviews in the field of Software Engineering allows giving a scientific value to the review of the literature that is done, defining a search strategy for the articles to be evaluated and finally obtaining a hypothesis for or against the revised literature.

4.2 Define Requirements

This activity goal is to identify and outline the artefact that can address the explicated problem, and elicit requirements on that artefact.

A2. Carry out a literature reviews on the datasets of software projects developed to date, in order to compare them and evaluate the necessary characteristics towards ESE studies.

A3. Perform a survey among research and innovation groups to identify the construction, curation and communication necessities for a dataset of software projects.

Two research methods will be used. First, systematic literature reviews [Kitchenham 2004] and Systematic Mapping Study [Petersen et al. 2008]. Second, surveys [Kitchenham and Pfleeger 2002] will be used to collect information from research groups and innovation teams.

4.3 Design and Develop Artefact

The third activity creates an artefact fulfilling the requirements from the previous activity, including the design of the functionality and structure of the artefact.

A4. Build the methodological support based on the analysis and design of the rules for the construction and curation of a representative sample of open source software projects, as well as the necessary metadata oriented to the ESE studies.

A5. Build the technological support with a source code analysis ecosystem and an extraction tool to obtain reports of software quality metrics from the dataset.

To receive feedback about the developed artifact and refine it we will conduct a focus group. This can be seen as an interview in which a group of respondents participate and discuss a specific topic. The aim is to enable the participants to be more creative and pursue the topic addressed in greater depth than in one-to-one interviews.

4.4 Demonstrate Artefact

The fourth activity of the method framework is to demonstrate the use of the artefact in one case, thereby proving its feasibility.

A6. Create datasets with the technological support and conduct controlled experiments with them in order to evaluate sample representativeness.

We will use Wohlin et al. (2012) guidelines to perform experiments. An experiment is a formal, rigorous and controlled investigation, that its key factors are identified and manipulated, while other factors in the context are kept unchanged.

4.5 Evaluate Artefact

Finally, the fifth activity determines how well the artefact is able to solve the explicated problem and to what extent it fulfils the requirements.

A7. Validate the methodological and technological support through the replication of case studies and action-research experiments.

For resolution and validation, the Research-Action method will be used. Originally, this method [Lewin 1947] sought to link the experimental approach of the social sciences with programs of social action that responded to certain main social problems. This method (with a necessary adaptation) has been widely accepted by the Information Systems community [Avison et al. 1999], [Seaman 1999].

5. Results

As part of this research, we carried out a systematic mapping study following the guidelines proposed by Petersen et al. (2008) to obtain an overview of the current technical development or level of practice of a research area. This study aimed to show research groups followed practices for experimentation with software projects, exposing the problems found that compromise sample representativeness and experiment replication.

For this study, we carried out a manual search in the journal Empirical Software Engineering; and the conferences Empirical Software Engineering and Measurement, and International Conference on Evaluation and Assessment in Software Engineering from 2013 to 2020. The systematic search resulted in 122 empirical studies with software projects. The study was presented at The Journal "Computación y Sistemas".

The results showed that, researchers most common practices are: making their own guidelines to select projects and using existing software project datasets. We did not evidence a unified or automated framework to select software projects, due to the large diversity of aspects considered by the researchers. Also, the main programming language of selected software projects was Java.

On the other hand, we are performing another systematic mapping study, in which we aim to determine and assess the general state of the software projects datasets in ESE studies in terms of their validity, purpose, sampling strategy, information extracted from the projects and the tools employed to extract that information. To that end, we formulated the following research questions:

RQ1: What are the characteristics of the software projects in the dataset for ESE studies?

RQ1.1: What criteria was considered to select the software projects in the dataset for ESE studies?

RQ1.2: What were the main programming languages of the projects selected for the dataset for ESE studies?

RQ2: What data was extracted from the software projects that compose the dataset?

RQ3: What tools were used to obtain the data extracted from the software projects that compose the dataset?

RQ4: Why was the dataset created?

RQ5: Is the dataset updated?

6. Expected Contributions

The expected contribution of this research is two fold. First, provide a procedures model to construct and maintain a dataset of software projects, allowing better replicability and representativeness in ESE studies. The key aspect is that the model can be fed with evolutionary information from projects, and at the same time, is extensible to add quality metrics and other useful artifacts. The model will be implemented as a software solution in order to validate and automatize our approach.

Second, share an open source software project dataset with their quality metrics. The dataset will be created and maintained with the implementation of the aforementioned model.

7. Future Steps

Once the second systematic mapping study is finished, we have two planned studies ahead. In first place, we will perform a replication study of a quasi-experiment originally made with the 20130901 release of the Qualitas Corpus; but instead of that version, we will use the 2021 version of the Qualitas Corpus, updating every non-deprecated project inside the 20130901 release. Our goal is to evaluate the degree of replicability between versions and review if temporal validity plays a role in the representativeness of the sample.

Secondly, we will conduct an academic questionnaire to Software Quality research groups in order to assess the results obtained in the studies mentioned in Section 4.2.

Acknowledgments

This research is supported by CONICET through the Internal Doctoral Grant, with a duration of 60 months.

References

Avison, D., Lau, F., Myers, M. and Nielsen, P. A. (1999). Action Research. Communications of the ACM, 42(1), 94–97.

Baltes, S. and Ralph, P. (2020). Sampling in Software Engineering Research: A Critical Review and Guidelines. https://arxiv.org/abs/2002.07764v5

Garvin, D. A. (1984). What Does "Product Quality" Really Mean? MIT Sloan Management Review, 25–43. https://sloanreview.mit.edu/article/what-does-product-quality-really-mean/

Irrazábal, E., Vásquez, F., Díaz, R. and Garzás, J. (2011). Applying ISO/IEC 12207:2008 with SCRUM and Agile Methods. Communications in Computer and Information Science, 155 CCIS, 169–180. https://doi.org/10.1007/978-3-642-21233-8_15.

Johannesson, P. and Perjons, E. (2014). An introduction to design science (Vol. 10, pp. 978-3). Cham: Springer.

Jureczko, M. and Madeyski, L. (2010). Towards identifying software project clusters with regard to defect prediction. ACM International Conference Proceeding Series, 1. https://doi.org/10.1145/1868328.1868342

Kitchenham, B. (2004). Procedures for Performing Systematic Reviews. Keele University, 33, 1–26. https://www.researchgate.net/publication/228756057

Kitchenham, B. and Pfleeger, S. L. (1996). Software quality: the elusive target. IEEE Software, 13(1), 12–21. https://doi.org/10.1109/52.476281

Lehman, M. M. (1996). Laws of software evolution revisited. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 1149, 108–124. https://doi.org/10.1007/BFb0017737

Lewin, K. (1947). Frontiers in Group Dynamics: Concept, Method and Reality in Social Science; Social Equilibria and Social Change. Human Relations, 1(1), 5–41. https://doi.org/10.1177/001872674700100103

Lewowski, T. and Madeyski, L. (2020). Creating Evolving Project Data Sets in Software Engineering. Studies in Computational Intelligence, 851, 1–14. https://doi.org/10.1007/978-3-030-26574-8_1

Mockus, A., Fielding, R. T. and Herbsleb, J. D. (2002). Two case studies of open source software development. ACM Transactions on Software Engineering and Methodology (TOSEM), 11(3), 309–346. https://doi.org/10.1145/567793.567795

Munaiah, N., Kroh, S., Cabrey, C. and Nagappan, M. (2017). Curating GitHub for engineered software projects. Empirical Software Engineering, 22(6), 3219–3253. https://doi.org/10.1007/s10664-017-9512-6

Nagappan, M., Zimmermann, T. and Bird, C. (2013). Diversity in Software Engineering Research. Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering - ESEC/FSE 2013. https://doi.org/10.1145/2491411

Palomba, F., Bavota, G., Penta, M. Di, Fasano, F., Oliveto, R., & Lucia, A. De. (2018). On the diffuseness and the impact on maintainability of code smells: a large scale empirical investigation. *Empirical Software Engineering*, *23*(3), 1188–1221. https://doi.org/10.1007/S10664-017-9535-Z/TABLES/11

Parnas, D. L. (2001). Some software engineering principles. In Software fundamentals: collected papers by David L. Parnas (pp. 257–266). https://dl.acm.org/doi/10.5555/376584.376632

Petersen, K., Feldt, R., Mujtaba, S. and Mattsson, M. (2008). Systematic Mapping Studies in Software Engineering. Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering, 68–77. www.splc.net

Rahman, M. M. and Roy, C. K. (2018). Improving IR-based bug localization with context-aware query reformulation. *ESEC/FSE 2018 - Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 621–632. https://doi.org/10.1145/3236024.3236065.

Seaman, C. B. (1999). Qualitative methods in empirical studies of software engineering. IEEE Transactions on Software Engineering, 25(4), 557–572. https://doi.org/10.1109/32.799955

Shepperd, M., Song, Q., Sun, Z. and Mair, C. (2013). Data quality: Some comments on the NASA software defect datasets. IEEE Transactions on Software Engineering, 39(9), 1208–1215. https://doi.org/10.1109/TSE.2013.11

Tempero, E., Anslow, C., Dietrich, J., Han, T., Li, J., Lumpe, M., Melton, H. and Noble, J. (2010). The Qualitas Corpus: A curated collection of Java code for empirical studies. Proceedings - Asia-Pacific Software Engineering Conference, APSEC, 336–345. https://doi.org/10.1109/APSEC.2010.46

Vázquez, H. C., Bergel, A., Vidal, S., Díaz Pace, J. A. and Marcos, C. (2019). Slimming javascript applications: An approach for removing unused functions from javascript libraries. Information and Software Technology, 107, 18–29. https://doi.org/10.1016/J.INFSOF.2018.10.009

Vidal, S. A., Bergel, A., Marcos, C. and Díaz-Pace, J. A. (2016). Understanding and addressing exhibitionism in Java empirical research about method accessibility. Empirical Software Engineering, 21(2), 483–516. https://doi.org/10.1007/s10664-015-9365-9

Vidal, S., Bergel, A., Díaz-Pace, J. A. and Marcos, C. (2016). Over-exposed classes in Java: An empirical study. Computer Languages, Systems & Structures, 46, 1–19. https://doi.org/10.1016/J.CL.2016.04.001

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2012). Experimentation in software engineering. Springer Science & Business Media.