

An initiative to improve reproducibility and empirical evaluation of software testing techniques

Francisco G. de Oliveira Neto
Software Practices Laboratory
Federal University of Campina Grande
Campina Grande, Brazil
francisco.neto@computacao.ufcg.edu.br

Richard Torkar
Dept. of Computer Science and Engineering
Chalmers and the University of Gothenburg
Gothenburg, Sweden
richard.torkar@cse.gu.se

Patrícia D. L. Machado
Software Practices Laboratory
Federal University of Campina Grande
Campina Grande, Brazil
patricia@computacao.ufcg.edu.br

Abstract—The current concern regarding quality of evaluation performed in existing studies reveals the need for methods and tools to assist in the definition and execution of empirical studies and experiments. However, when trying to apply general methods from empirical software engineering in specific fields, such as evaluation of software testing techniques, new obstacles and threats to validity appears, hindering researchers’ use of empirical methods. This paper discusses those issues specific for evaluation of software testing techniques and proposes an initiative for a collaborative effort to encourage reproducibility of experiments evaluating software testing techniques (STT). We also propose the development of a tool that enables automatic execution and analysis of experiments producing a reproducible research compendia as output that is, in turn, shared among researchers. There are many expected benefits from this endeavour, such as providing a foundation for evaluation of existing and upcoming STT, and allowing researchers to devise and publish better experiments.

I. INTRODUCTION

Much effort has been focused in improving the approaches and techniques to empirically evaluate software engineering research. The empirical software engineering community has gathered effort to change researcher’s minds and start disseminating the thought of evaluation methodologies as also an end instead of just a mean. Fortunately, several researchers have become aware of the importance of empirical evaluation. Performing an experiment with solid statistical analysis and validation is acknowledged to be a good practice among researchers and are often a requirement to achieve publications in several journals and conferences with high impact factors.

In fact, we have lately seen a trend that researchers are advised to provide a Reproducible Research Compendium (RRC) when publishing studies. The RRC is a container for all components necessary to reproduce the research, such as papers describing the technique and the computational environment where the study was executed. However, how many researchers are really able or prepared to perform experiments? A solution to disseminate proper usage of experiments is to encourage reproduction, replication and re-analysis of experiments.

Through replication and extension of experiments, researchers can investigate and extend the results to better understand experiments, discover improvements or even new applications for the investigated techniques [1]. However,

migrating those approaches and techniques to specific software engineering areas, such as *software testing*, can lead to many validity threats that inexperienced experimenters tend to overlook, or even disregard.

This paper focuses on issues and solutions regarding evaluation of software testing techniques (STT). Those techniques rely on several concepts related to computer science and other disciplines. However, studies indicate a general lack of empirical evaluation of the techniques proposed [2], [3], which leads to a set of techniques devoid of any formal foundation that are risky to transfer to industry. The reason is that, most of the times, the techniques require executable artefacts such as source code or test cases, which are often unavailable [4]. Furthermore, empirical studies usually lack details in their description, raising concerns about the reliability of the results obtained, or at the least, introduce difficulties in replicating and extending experiments.

Researchers have been discussing the issues and needs for evaluation of software testing techniques in general [2]–[6]. However, no methodology was yet proposed to perform studies with software testing techniques focused on reproducibility. Besides hindering reproducibility, this problem also affects definition and execution of empirical evaluations of STT. In fact, studies in literature state that empirical methods in software testing require specific guidelines; to improve the research and reporting processes, or to establish common standards used to conduct and report experiments [4], [6].

This paper proposes an initiative, by exposing the issues and existing contributions towards the general methods to experiment with software testing techniques and then proposing a collaborative effort of reproducing and sharing experiments with STT. The goal is to build a tool and a repository that allows experimenters to reproduce their fellow researchers’ experimental findings. The solution is to produce reproducible research compendiums containing the specific elements of software testing research. We will refer to these specific compendia as *Reproducible Software Testing Research Compendia* or simply RSTRC.

As a consequence, we encourage standardization and feedback regarding evaluation of proposed STT. We guide our discussion according to the following questions: RQ1: *Which are the current problems with evaluation of STT?* RQ2:

Which specific reproducible attributes can be found in studies¹ concerning software testing? Section II discusses RQ1, while discussion regarding RQ2 is presented in Section III. Finally, Section IV presents concluding remarks and the next steps towards our proposed initiative.

II. WHAT ARE THE ISSUES?

The empirical software engineering community has been dedicating efforts in breaking down the fundamental elements of an experiment in software engineering. Despite the efforts and discussions regarding appropriate methods to conduct experiments in software engineering, evaluating STT is not easy since many information and artefacts are required in order to thoroughly evaluate the capabilities of a testing technique. First of all, software testing itself is a very multidisciplinary area, that is able to comprise distinguished and combined types of software such as stand alone, web services, real time, critical systems, embedded systems, etc. Therefore, drawing a precise line to establish the scope affecting the test is very challenging.

At the same time, trying to draw that line is one of the most important steps to define a good experiment, since it allows researchers to identify: context, subjects, limitations, and more importantly, which variables can/must be controlled. Already in 2004, Juristo et al. stated that our testing technique knowledge is very limited, given that most of it is based on impressions and perceptions [5]. Engström et al. and Dias Neto et al. reached similar conclusion years after that statement, still pointing to a general lack of empirical evaluation of specific types of testing techniques (respectively, regression and model-based testing) [2], [3]. Why are we not seeing, over the years, a significant increase in the number of empirical findings regarding STT?

There is an array of classifications [1], [5], configurations [7] and guidelines [8], [9] that help researchers to define, conduct and report their experiments. But manifesting those methods for constructs specific to STT evaluation is very challenging. For example, statistics often require large sets of significant data (usually hard to obtain or unavailable) to achieve conclusive results. In addition, the need for human involvement in testing makes the generation of a large number of test sets infeasible. Similarly, obtaining defect data for larger test sets is nearly impossible since detection of defects will always be, to a certain degree, a stochastic process [4].

Availability and access to information is still limited when evaluating STT. For instance, the System Under Test (SUT) may not be ready for testing, test cases may present limited coverage, defect data is unknown, among others. Even when available, many researchers disregard the representativeness of their artefacts (e.g. choice of inappropriate programs or unreal defects) [5] or even the minimum sample size required to claim statistical significance of results. That creates gaps and validity threats that seriously compromise credibility of results. There are existing techniques for stochastic generation of data for

testing [10], [11]. More specifically, they rely on models to generate well-formed data. Those types of strategies enable control over automatic generation of large sets of artefacts, hence assisting experimenters who struggle with availability and accessibility of objects in an experiment. Thus, existing approaches can already be applied to overcome those issues.

Besides getting data, organization and analysis of all elements in an experiment intimidates many researchers. Devising an appropriate experimental design to comply with one's hypotheses, objectives and variables is overwhelming and a poor design lead to confusing reports. Similarly, lack of experience and/or knowledge in statistics hinder researchers to harness full potential of their data causing them to either enhance or destroy credibility of conclusions. For example, many researchers recklessly use parametric tests (such as ANOVA or *t*-test) as a rule of thumb, without proper investigation of its assumptions, sometimes leading to erroneous conclusion [12].

In its early start, the empirical software engineering community turned to other disciplines that have been dealing with empirical evaluation over decades (e.g. biology and social sciences) in order to overcome general difficulties in managing data, subjects, statistical analysis, etc. The software testing community seems to be performing more experiments. However, they neglect validation of existing experiments and proposal of innovative strategies to help their fellow researchers in overcoming the specific challenges of evaluating STT. That being said, validation of experiments needs to be performed through reproduction, replication or re-analysis of existing experiments with STT.

III. WHAT CAN WE DO?

The initiative begins with a collaborative effort within the testing community to share and standardize methods and artefacts used to evaluate STT. Therefore, the research comprises the state of art of empirical studies in software engineering, software testing, statistics, experimental designs, among others. The outcome is the development of techniques and tools to provide support and execution of experiments. Therefore, researchers will extend the body of knowledge by *proposing new methodologies* to perform reproducible studies with STT. Even though replication and re-analysis are just as important as reproducibility, we believe that focusing first in reproducibility will allow researchers to quickly overcome the general lack of availability and accessibility of data required to replicate/re-analyse the existing experiments. Thus, as the initiative strengthens, we enable more replication because the community will rely on a larger repository of experiments.

In summary, we intend to encourage practices similar to the initiative of reproducible software engineering but adapting them to the software testing community. We begin by defining input and output of our process. The input are the study parameters, such as the objects of study, a SUT, sets of test cases, number of subjects, a general hypothesis and goals. The output, in turn, is a compendium (i.e. a package), named *Reproducible Software Testing Research Compendium*

¹The studies may comprise research focused on software testing, other fields of computer science, as well as other disciplines from both industry and academia.

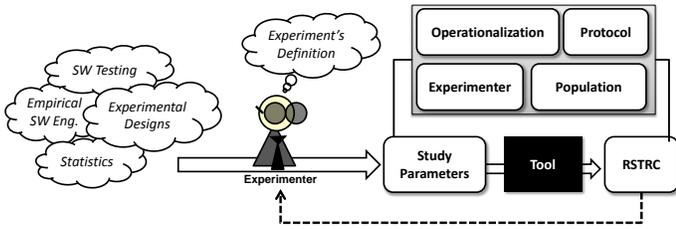


Fig. 1. Proposed reproducible research initiative for evaluation of STT.

(RSTRC), containing all elements able to describe and re-execute the experiment. A summary is presented in Fig. 1.

Each study parameter will be divided into the following categories: operationalization, protocol, population and experimenter. Those four elements cover the general configuration of an experiment, as discussed by Gómez et al. [7]. The *operationalization* include elements describing the act of translating a construct into its manifestation; the *protocol* is the set of materials, apparatus, forms and procedures used in the experiment, whereas objects and subjects belong to the *population* category and, finally, the *experimenter* includes the people involved in the experiment.

That allows classification of elements within RSTRC as well, enabling search engines to determine which are the more/less frequently used constructs, tools, response variables, forms, among others. By analysing the elements in those categories, we can determine which are the reproducible elements for STT research and ultimately the guidelines required to apply these elements. Given that there are many known (and perhaps some unknown) study parameters, we intend to first determine the elements of an RSTRC. They must be based on strategies to improve the description of studies and methods to facilitate access to artefacts. The resulting guidelines will then, allow researchers to converge their *modus operandi*, thus leading to standardization among reported results within the software testing community.

That convergence allows identification of the most significant attribute(s) to reproduce STT research². Although that may sound greedy, we believe that the targeted information is already dispersed within existing surveys, systematic reviews and publications that discuss fundamental metrics and artefacts required to evaluate STT. Thus, raising awareness and encouraging reproducibility can lead to a massive search and, eventually, those fundamental elements, significant for evaluating STT, will become recurrent in most empirical evaluation.

On the other hand, there are complementary attributes found only in applications of STT in industry, since, when applied to production, other elements of an STT appear, such as training of staff, testers' expertise, etc. However, two main aspects of reproducibility may hinder collaboration with practitioners: sharing and availability of information. That has already been an issue with existing evaluation in industry due to non-

²That can happen initially at short term, but then refined at long term.

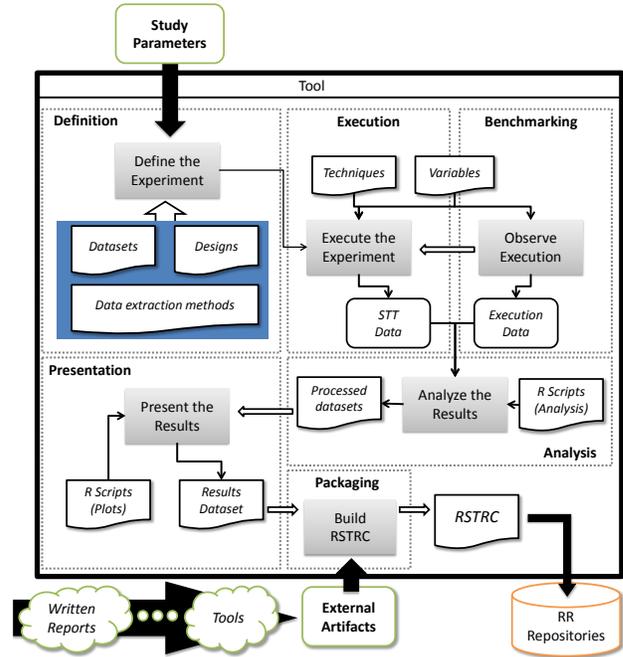


Fig. 2. Overview of a tool that executes experiments with STT.

disclosure agreements. Although some industrial data can be shared sometimes, we cannot assume that full disclosure is allowed. Consequently, we suggest two different categories of RSTRC: *industrial* and *academic*. Researchers sharing and using the first category should expect limited availability and accessibility due to involvement with practitioners, whereas those using the second category can assume full disclosure and access regarding methods and artefacts used. Note that, unlike replications, reproduction of experiments allows for different artefacts and even contexts. Therefore, even if full disclosure is unavailable, some aspects of the study can still be reproduced.

As study parameters are formalized and the protocol for different experiments become available through shared RSTRC, we can start to build a tool to allow automatic execution of experiments. The idea is that researchers can implement their STT, execute experiments and share their RSTRC by using interfaces and libraries. In turn, other researchers can reproduce studies or execute different techniques with common interfaces to compare results. Fig. 2 shows an overview of the tool and its interacting modules. The activities executed automatically are marked as shaded rectangles, while artefacts and libraries of the tool are white documents.

Each module covers stages of an experiment, and the idea is that these artefacts and libraries are reused during reproduction of experiments. For example, specific experimental designs can be stored to be reused in other studies; different methods can provide default measures for widely used dependent variables in testing, such as Average Percentage of Fault Detection (APFD), or recall. Also, different studies may use the same variables, techniques or data extraction methodology. By sharing the tool and the RSTRC that is produced, we

encourage reproducibility practices within the community.

Furthermore, the tool will also provide benchmarking of experiments being executed, for example, time to execute the entire experiment (with or without replications), number of data points collected, metrics of the artefacts used (size in LOC, number of test cases, etc). In the end, researchers will be able to compare performance of execution of different studies, leading, for example, to the creation or improvement of experimental designs. Ultimately, we would also be able to perform meta-analysis, focusing on, for instance effect size [12] to discern the total effect a number of test techniques have on a particular domain, e.g. model-based testing.

Besides providing execution of empirical studies, R³ scripts will be implemented and included in the tool to provide automatic statistical analysis of data collected during the study. Moreover, experienced experimenters can also write scripts to simplify analysis of complex experimental designs (e.g. factorial designs). For instance, he or she can write scripts to compare samples, obtain *p*-values, coefficients, linear regression models, etc. However, there will be a level of dependence between the analysis and the response variables investigated, or subjects used. Therefore, the scripts need to consider the study parameters provided as input.

Regarding evaluation of the tool, we target two aspects: (i) The increase in reproducibility. (ii) Its adequacy to evaluate STT. For that, we will initially, reproduce studies described in research literature. Researchers can then assess the reproducibility of the ‘original’ study using existing scoring methods [13]. Then, they reproduce the experiment using the tool, resulting in a new compendia (in this case, an RSTRC). The goal is to observe an increase in the reproducibility scores when comparing the RSTRC obtained using the tool and the compendia of the original study.

IV. CONCLUDING REMARKS

This paper discussed issues and goals regarding the need for more reproducible research with software testing techniques. Based on existing achievements and contributions from empirical software engineering, we proposed introduction of guidelines, artefacts and methods to encourage researchers to produce compendia targeting evaluation of STT (RSTRC). The objective is to define, execute and deploy experiments with improved description, accessibility and availability of required artefacts to reproduce, replicate and re-analyse the experimental findings. Through a collaborative effort among researchers, sharing and reproduction allows the community to identify which experimental findings are consequences of an actual cause effect relationship, rather than an artefactual result. That leads to a more reliable body of knowledge, increasing confidence in existing research and setting high standards for upcoming testing techniques.

On the other hand, that initiative requires a lot of effort from the community, by committing to set and disseminate standards. As of April 2014, we acquired funds to conduct

this research project as a collaboration between the Federal University of Campina Grande (Brazil) and Chalmers and the University of Gothenburg (Sweden) under the *Science Without Borders* (SWB)⁴ scholarship. We are currently working on the definition of the RSTRC and designing our tool, hence our next steps is gathering collaborations and feedback from both industry and academia in order to start building our shared repository of reproducible research. Ultimately, the main benefit with this project is to encourage researchers and provide them an early familiarization with empirical methods. Proper evaluation of results based on solid evidence obtained through consistent statistical analysis significantly benefits the state of art, researchers and the community itself. Consequently, researchers become more critical towards the quality of results achieved, and more cautious and demanding regarding planning, development and presentation of their work.

REFERENCES

- [1] O. Gómez, N. Juristo, and S. Vegas, “Replication, Reproduction and Re-analysis: Three ways for verifying experimental findings,” in *1st International Workshop on Replication in Empirical Software Engineering Research, RESER 2010*, vol. 35, 2010.
- [2] E. Engström, M. Skoglund, and P. Runeson, “Empirical evaluations of regression test selection techniques: a systematic review,” in *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*, ser. ESEM '08. New York, NY, USA: ACM, 2008, pp. 22–31.
- [3] A. C. Dias Neto, R. Subramanyan, M. Vieira, and G. H. Travassos, “A survey on model-based testing approaches: a systematic review,” in *Proceedings of the 1st ACM international workshop on Empirical assessment of software engineering languages and technologies*. ACM, 2007, pp. 31–36.
- [4] L. Briand and Y. Labiche, “Empirical studies of software testing techniques: Challenges, practical strategies, and future research,” *SIGSOFT Software Engineering Notes*, vol. 29, no. 5, pp. 1–3, 2004.
- [5] N. Juristo, A. Moreno, and S. Vegas, “Reviewing 25 years of testing technique experiments,” *Empirical Software Engineering*, vol. 9, no. 1–2, pp. 7–44, 2004.
- [6] S. U. Farooq and S. M. K. Quadri, “Empirical evaluation of software testing techniques – need, issues and mitigation,” *Software engineering: an international Journal*, vol. 3, no. 3, pp. 41–51, 2013.
- [7] O. S. Gómez, N. Juristo, and S. Vegas, “Understanding replication of experiments in software engineering: A classification,” *Information and Software Technology*, vol. 56, no. 8, pp. 1033–1048, 2014.
- [8] P. Runeson and M. Höst, “Guidelines for conducting and reporting case study research in software engineering,” *Empirical Software Engineering*, vol. 14, no. 2, pp. 131–164, Apr. 2009.
- [9] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering*. Springer, 2012.
- [10] F. G. de Oliveira Neto, R. Feldt, R. Torkar, and P. D. L. Machado, “Searching for models to evaluate software technology,” in *Proceedings of the 1st International Workshop on Combining Modelling and Search-Based Software Engineering*, May 2013, pp. 12–15.
- [11] R. Feldt and S. Poulding, “Finding test data with specific properties via metaheuristic search,” in *24th International Symposium on Software Reliability Engineering (ISSRE)*, 2013, Nov 2013, pp. 350–359.
- [12] A. Arcuri and L. Briand, “A hitchhiker’s guide to statistical tests for assessing randomized algorithms in software engineering,” *Software Testing, Verification and Reliability*, vol. 24, no. 3, pp. 219–250, 2014.
- [13] J. M. González-Barahona and G. Robles, “On the reproducibility of empirical software engineering studies based on data retrieved from development repositories,” *Empirical Software Engineering*, vol. 17, no. 1-2, pp. 75–89, 2012. [Online]. Available: <http://dx.doi.org/10.1007/s10664-011-9181-9>

³<http://www.r-project.org/>

⁴Grant n. 88881.030428/2013-01, project n. 152146. More info about SWB here: <http://www.cienciasemfronteiras.gov.br/web/csf-eng/>