Towards Improving Experimentation in Software Engineering

Edson OliveiraJr, Viviane Furtado, Henrique Vignando, Carlos Luz, André Cordeiro
State University of Maringá (UEM)
Maringá, PR, Brazil
edson@din.uem.br
{viviane.rfurtado,rickuev}@gmail.com
carlos.danilo.luz@gmail.com
cordeiroandrefelipe@gmail.com

Igor Steinmacher
Federal University of Technology - Paraná (UTFPR)
Campo Mourão, PR, Brazil
igorfs@utfpr.edu.br

Avelino Zorzo
Pontifical Catholic University of Rio Grande do Sul (PUCRS)
Porto Alegre, RS, Brazil
avelino.zorzo@pucrs.br

ABSTRACT

[Background:] Experimentation in Software Engineering plays a central role on sharing and verifying scientific findings. As experiments have increased significantly in Software Engineering area, we observe that most of them fail to provide a way to be repeated, replicated or reproduced, thus jeopardizing or delaying the evolution of the Software Engineering area. **[Aims:]** In this vision paper, we present and discuss techniques and infrastructure to continuously improve experiments towards repeatability, replicability, and reproducibility. **[Method:]** We define these techniques and infrastructure based on experiences of our research groups and existing literature. Furthermore, we follow Open Science principles. **[Results:]** We provide incipient results and foresee a central infrastructure composed of two repositories and two recommendation systems to support techniques for: reporting experiments; developing ontologies for experiments and open educational resources; mining and recommending experiments; specifying data management plans, identifying families of experiments; and teaching and learning experimentation. **[Conclusions:]** Our techniques and infrastructure will prospectively motivate and benefit Software Engineering evolution by improving the conduction and further reproducibility of experiments.

CCS CONCEPTS

• **Software and its engineering → Empirical software validation**;

KEYWORDS

Experimentation Improvement; Ontology, Repositories; Recommender Systems; Data Management Plans; Teaching/Learning; Continuous Experimentation

ACM Reference Format:
Edson OliveiraJr, Viviane Furtado, Henrique Vignando, Carlos Luz, André Cordeiro, Igor Steinmacher, and Avelino Zorzo. 2021. Towards Improving Experimentation in Software Engineering. In *Brazilian Symposium on Software Engineering (SBES '21), September 27-October 1, 2021, Joinville, Brazil.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3474624.3477073

1 INTRODUCTION

Software Engineering (SE) experimentation research and practice have significantly grown in the last years. This is due to important movements from academia and industry toward well-reported experiments[1], practical evaluation of theories and technologies, and building reliable body of knowledge, which might lead to technology transfer [14, 15]. For instance, continuous experimentation has been adopted by a prominent portion of the software industry to improve quality and aggregate value to their products [15, 19].

Several initiatives for promoting SE experiments are reported in the literature [2, 4, 14]. However, none of them provides techniques and infrastructure toward Repeat/Replicate/Reproduce (Rep*) activities on experiments, as we envision in this paper.

Our concept of improving SE experiments encompasses: guidelines for reporting experiments appropriately; development of ontologies to support teaching and learning of SE experiments and Open Educational Resources (OER) for experimentation in SE; mining of SE experiments; recommendation of SE experiments; data management plans; continuous experimentation; and repositories for SE experiments. Therefore, in this vision paper, we present a roadmap for improving experimentation in SE (ESE), which includes techniques and an infrastructure to support prospective research and practice in this area.

Novelty in this paper relies on architecting different well-known and successful techniques to work in the benefit of ESE improvement in the perspective of Dissemination, Education, and Practice. Although the techniques used in our proposal already exist in the literature, they are straightforward used for other research topics, but are not in for ESE.

2 EXPERIMENTAL-BASED SOFTWARE ENGINEERING EVOLUTION

Publishing research findings is essential for disseminating knowledge and to provide opportunity of evolution to research communities. However, it is fundamental that these findings are validated by researchers. Therefore, published findings should provide means to be evaluated. One practical way is to conduct formal experiments based on research questions and assumptions, stating hypotheses

[1]In this paper we are focused on controlled experiments and quasi-experiments.

and variables, and verifying the established cause-effect of a certain phenomenon considering well-reported results [18].

Unfortunately, this is not enough to evolve Science. It is necessary to verify the results from the original experiments. To do so, one might repeat, replicate, or reproduce the original experiment. This is key to enable new ideas, assumptions, and hypotheses, as well as to perform meta-analyses and confirm results [3, 4, 16]. In Figure 1, we foresee how research, especially in SE, evolves based on experiments.

In SE, research usually starts by defining a `research question` and the `methods` to answer it. `Results` of the research are **discovered**, then should be empirically **evaluated**, in this case, based on one or more experiments. Each experiment has a set of `hypotheses` and `variables` that are stated based on `ideas` and/or `assumptions` previously gathered. `Findings` of an experiment are **verified** based on `Repeat/Replicate/Reproduce (Rep*)` activities. In such activities, `new ideas`, `assumptions`, or even `hypothesis` might arise. `Meta-analysis` also aids to **verify** the findings of a set of experiments, thus **improving experiments**.

Rep* activities aim to discuss the outcomes of an original experiment and whether it is valid for target populations. Feitelson [3] discusses on the differences among Rep* activities as summarized in Table 1.

Table 1: Repeat, Replicate and Reproduce an Original Experiment (based on [3])

Original Experiment's	Repeat	Replicate	Reproduce
Main idea	original	original	original
Artifacts	original	recreated	recreated
Procedures	original	original	similar to original
Results	original	close	same as original result*
What is tested?	design issues	protocol	scoping and limitations

**result meaning the outcome, not the scientific fact.*

The main principle for supporting Rep* is to verify whether a scientific fact is valid, it does not matter who performs an experiment and how. Generally, some authors use replicability for situations in which a past experiment is rerun in the exactly same way, whereas reproducibility is used focusing on the result being verified [3].

We have observed that the majority of analyzed Software Engineering experiments provide no Rep* capability, mainly because of missing experimental elements, which are not properly reported (see Section 3.1.1). Thus, this jeopardizes continuous improvement and scientific evolution based on experiments. As stated by Peng [12], we evolved in computational science, but we do not properly evaluate published findings as they are in general for publication purpose only.

In this paper, we follow Feitelson's [3] repeat/replicate/reproduce terminologies. Therefore, we understand that performing specific activities to improve Software Engineering experiments, such as, properly sharing data, recommending experiments, and teaching ESE, might provide a means to promote verification of results, as well as searching for problems, auditability/accountability and, consequently, Software Engineering evolution.

3 CONTINUOUSLY IMPROVEMENT OF EXPERIMENTS

We advocate that prospective techniques and infrastructure might promote improvement, reproduce/replicate/repeat activities, and evolution of Software Engineering experiments. These techniques and infrastructure are based on the main principles for Open Science, such as: open access, open methodology, open data, open source, and open educational resources [11]. As open (meta)data, we follow the four FAIR[2] principles: Findable, Accessible, Interoperable, and Reusable. To do so, we strongly deal with open data as much as possible.

Open access is guaranteed to techniques, infrastructure, and produced data by adopting licenses like Creative Commons Licenses CC-BY,[3], as well as using/creating open source tools and providing open experimental produced educational resources.

We organize our vision on improving SE experiments based on three main pillars: **Practice**, **Education**, and **Dissemination**. Figure 2 depicts such pillars.

The foundation of these three pillars is the well-know Software Engineering **Experimentation Process**, which is composed of elements (*e.g.*, hypotheses and variables) and findings, as in the Wohlin *et al.*'s process [18], for example. Note, in Figure 2, that all pillars are interrelated (big arrows), however they do not represent a workflow with ordered activities. Actors from one pillar might act in another pillar as, for instance, SE Researchers and Students in Education might perform Rep* activities in Practice. Therefore, in Figure 2, we represent the prominent actors of each pillar.

A central infrastructure supports the pillars. This infrastructure is composed of two repositories: **Experiments Repository** with data and metadata for **Practice** and **Dissemination** of SE experiments; and the **Experiments OER Repository** with data and metadata of Open Educational Resources (OER) for **Teaching/Learning Experimentation**. These repositories might be built, for instance, using a NoSQL database, and synchronized to a data-driven platform (*e.g.* GitHub) or a content-based environment (*e.g.* Moodle). We are currently working on how to mine and exchange different experimental elements and data from different sources of SE experiments by an API.

We foresee an architecture based on microservices or even one based on distributed components with a middleware broker. Therefore, our technologies and approaches will not depend on a specific technology.

3.1 Dissemination

One of our *foci* is to investigate how to formalize the reported concepts and their relationships involved in the Experimentation Process. By doing this, we provide means to make them persistent, public, citable, and accessible. To do so, we detail how techniques and the infrastructure might assist this pillar.

3.1.1 ***Reporting Guidelines.*** Reporting Software Engineering experiments plays a straightforward role, for example, in: reporting material and methods and decision making during scoping, planning, conducting, and analyzing results; facilitating Rep* activities

[2]https://www.go-fair.org/fair-principles
[3]https://creativecommons.org/licenses/by/2.0

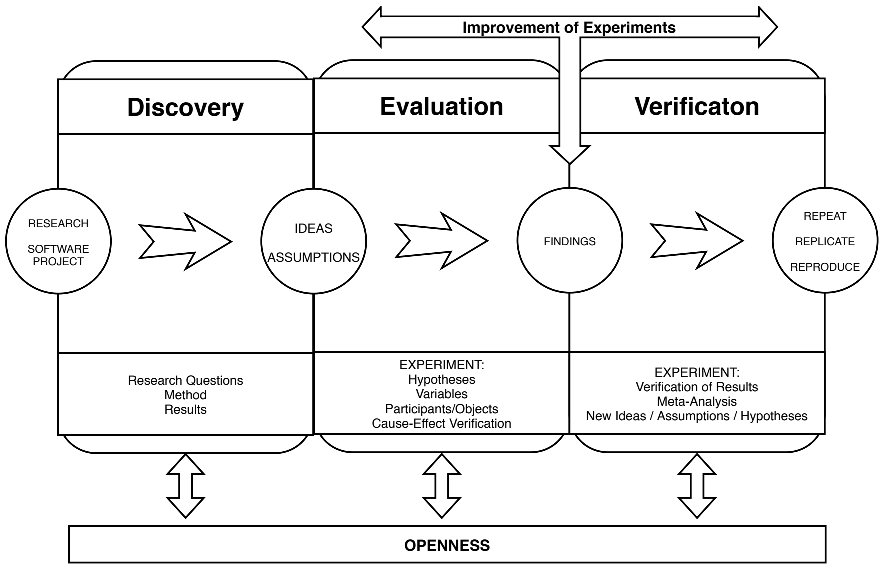

Figure 1: Science evolution based on experimental evaluation of findings

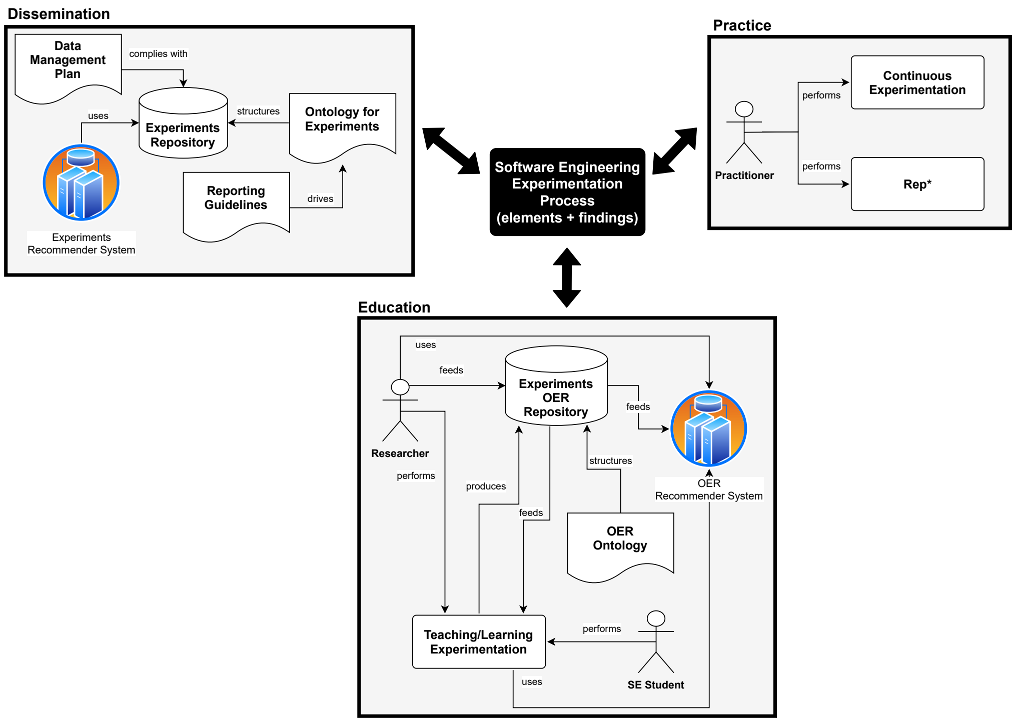

Figure 2: Techniques and infrastructure for improving SE experiments

guaranteeing consistent results; reducing general threats to validity; and, increasing reliability and audition on laboratory packages.

Specifically in Software Engineering, experiments usually take several pages to be reported. Therefore, an objective and self-content experiment report is expected.

In the literature, several authors propose guidelines for documenting Software Engineering experiments. Although each of these proposals have their merits, none of them have been accepted as a *de facto* standard. The closest to tool support are the Empirical Standards of SIGSOFT[4].

[4]https://web.cs.dal.ca/SIGSOFT-Empirical-Standards

In a Systematic Mapping Study on Software Product Line experiments [omitted] we recently concluded that the authors of 211 analyzed primary studies generally present their tools, approaches, techniques, and algorithms, but only a brief section is dedicated for the experiments, *i.e.*, few studies barely report their experiments. Only 23% (48 out of 211) of the experiments followed some reporting template. Besides, none of the 211 experiments evaluated their quality, despite the existence of several experiment quality evaluation approaches [1, 6, 8, 9].

Based on this mapping study, we defined a preliminary a set of guidelines [5] and a conceptual model [6] to objectively report Software Engineering experiments.

The main difference of our guidelines from existing ones is that ours are supported by an ontology (Section 3.1.3) with the formalization of experiment concepts and their relationships. These guidelines are implemented in an experiment repository (Section 3.1.2) with metadata from the ontology. Data Management Plans (DMP) (Section 3.1.4) might support data provenance, curation, and preservation. Also, recommender systems (Section 3.1.5) can act in the structured repository to recommend experiments based on different parameters once the repository is structured according to an ontology and a DMP.

3.1.2 ***Repositories.***

With the rising of digital preservation and curation, and data science, repositories of Software Engineering experiments became essential infrastructures to enable the definition and application of our techniques. In fact, we consider Trusted Digital Repositories (TDR), which are reliable, long-term access to managed digital resources to its designated community, now and into the future.

We have also investigated how to provide TDR characteristics following guidelines from the Digital Preservation Capability Maturity Model[7] and the Open Archival Information System (OAIS) Reference Model (ISO 14721)[8] to our repositories or even extend an existing one, such as 3TU. Datacentrum, CSIRO Data Access Portal, Dryad, figshare, Dataverse, and Zenodo.

3.1.3 ***Ontologies.***

Software Engineering experiments include several formal concepts and properties, such as: independent and dependent variables, hypothesis, instrumentation, data, hypothesis tests, pilot projects, and training. Thus, they can be highly formalized as an ontology, which is an explicit specification of a conceptualization. An ontology provides a formal definition of objects, concepts, properties, and their interrelationships. It also supports logical operations to infer relationships between concepts [5]. In the case of SE experiments, we understand all experimental concepts and their relations must be explicitly defined.

Ontologies for Software Engineering experiments might contribute to improve experiments for Rep* activities. Our guidelines for documenting of Software Engineering experiments and our conceptual model, are important sources of information for an ontology.

We understand that ontologies have a central role in formalizing Software Engineering experiments. Therefore, they might facilitate the data organization in the repository of Figure 2, as well as provide means to new tools to support automation of ESE. We understand that all experimental elements might not be as important for industry as for academia, however, formalization of repositories and tools are relevant for both environments.

3.1.4 ***Data Management Plans.***

DMPs are formal documents in which data are described both during and after a research project has finished. It usually includes several elements, such as: metadata, storage and backup, responsibility, access and sharing procedures, and licensing [11].

DMPs are becoming mandatory by several research project funding agencies, thus its importance has increased over the last years. Guidance for DMP content varies for different specific research areas, such as, those for Computers & Information Sciences & Engineering (https://www.nsf.gov). DMPTool and DMPOnline, for instance, provide several DMP models for hundreds of different research areas.

We understand that DMPs are essential for improving Software Engineering experiments and for enabling Rep* activities by making data available, organized and ready for curation and preservation. Therefore, we have investigated a DMP model for Software Engineering experiments by revisiting existing literature on how data is acquired and shared in experiments of several topics in Software Engineering. We understand that it must support: data types and sources, content and format, sharing and preservation procedures, protection and licensing, period of data retention, data dissemination and policies, and rationale. Therefore, our reporting guidelines and ontology are of great source of information for such DMP.

3.1.5 ***Recommending Experiments.***

Another technique that contributes to improve Software Engineering experiments and Rep* activities is the recommendation of experiments. A recommendation system [13] for Software Engineering experiments is useful for recommending methods, processes, guidelines, among others, to perform an experiment, thus facilitating its development, and encouraging the culture of continuous experimentation in academia and industry. In our perspective, recommendation of experiments with defined parameters or filters, for both experts and newcomers, might be performed based on how data is organized in the experiment repository.

Conducting a quality Software Engineering experiment requires statistics foundations and considerable experience in experimental design and Software Engineering. Thus, the learning curve becomes long to appropriately plan, conduct, and analyze the quality of the results of this kind of experiment. Besides, the industry has increasingly adopted Software Engineering solutions, thus demanding an experimental body of knowledge in the area. Therefore, we believe it is important to provide a content-based filtering recommendation system of Software Engineering experiments based on ontologies containing hundreds of experiments instances and the support of an appropriate infrastructure, for example, as the ones presented in this paper. Therefore, recommender systems might be used to enable the dissemination of good practices on planning, conducting, and analyzing experiments, thus providing a basis for Rep* activities and SE evolution.

[5]https://doi.org/10.5281/zenodo.3957649
[6]https://doi.org/10.5281/zenodo.3948161
[7]http://www.securelyrooted.com/dpcmm
[8]http://www.oais.info

3.2 Education

Education is the pillar responsible for understanding real-world problems, thus investigating and producing scientific knowledge to mitigate such problems before transferring solutions to the industry. Therefore, in our vision, **Researchers** and **SE Students** play a central role in contributing to the improvement of the **Experimentation Process** and in widely providing an experimentation culture to both academic and industry.

Researchers have one of the most challenging missions in the world: identify research gaps, investigate them, then provide science evolution throughout the scientific process, in which the Experimentation Process is crucial.

To support researchers to play their roles, we envision that Open Educational Resources (OER) can assist them to teach ESE. OER is a well-known and accepted resource type as they can be disseminated in standard formats and open licenses, such as the Creative Commons.

In a recent survey[9] with ESE lecturers we identified that: (i) 72% of experiment concepts are taught in a separate (specific ESE) course; (ii) 64% of produced materials for teaching ESE neither follow any type of open license nor are publicly available; and (iii) the most used method to verify learning outcomes is practical (actual) projects and seminars on the findings of the experiments. These findings suggest a structured manner of developing and sharing OER for SE experiments in a way researchers and students might contribute to improve and evolve such materials. Therefore, an **Experiments OER Repository** for ESE educational materials is more than welcome. This repository might also be used by **Practitioners**.

To structure such repository, an **OER Ontology** is fundamental, because of the diversity of OER types and their elements, such as, storage and visualization formats, type of content, relation to other OER, links to external materials, rationale, course level, and audience. We are currently investigating an ontology as an extension of the existing ONTOER+, in an incipient stage of design based on the conducted survey with researchers who teach ESE.

Researchers and SE students will take advantage of such an ontology as they will be aware of how to use, modify, evolve, and distribute such materials for future use.

With respect to recommending OERs for ESE, we envision providing assistance to instructors, students, and practitioners. For instructors, the idea is to recommend high-quality experiments or available OER such as, Massive Open Online Courses (MOOC), slides, videos so they can use or produce OER for ESE using such well-reported and organized experiments as examples for ESE classes and activities. For students and practitioners, we expect them to use the system as a way to search for experiments or OER, thus they can learn ESE with different domains and alternative OERs.

We understand that teaching and learning experimentation in SE is crucial for evolving SE based on experimentation, thus contributing to establish continuous experimentation in the software industry.

[9]Preparing to submit to a journal.

3.3 Practice

In this pillar the main focus is how to assist and motivate practitioners, researchers, and students to perform Rep* activities to evolve SE. Particular attention is given to **Practitioners** conducting continuous experimentation in industry as a way to improve the software process, mitigate common threats, create datasets and baselines, statically guide new developments, and aggregate value to their products.

Continuous Experimentation or Systematic Experimentation in Software Engineering follows the principle that product or service ideas can be developed by constantly conducting systematic experiments and collecting user feedback [19]. Continuous experimentation has been largely and successfully applied in the industry for software development [2, 15, 19].

Continuous experimentation models and implementation techniques have been proposed in the literature and used by the industry, as reported by Schermann *et al.* [15]. Regression-driven, used to identify whether changes have a negative impact on system properties, and business-driven experiments, which are requirement-engineering techniques and are used to validate business hypotheses, have been largely conducted in industry [15].

An example of implementation technique of experiments is traffic routing [15], in which multiple versions of an application or service run in parallel, thus technical debt is avoided due to considered experimentation logic and source code . Another experimental setup technique is Feature Toggles [15], which has conditional statements deciding about which code block to execute next.

Conducting systematic experiments benefits the promotion of improvements on SE experiments and provides a way to facilitate Rep* activities, mainly based on the experiences and domains from the industry stakeholders. Therefore, we believe that, by providing a way to continuous experimentation of SE artifacts and procedures based on pilot projects, we might successfully guide ESE in industry. A systematic approach for experimentation in Software Engineering contributes directly to retrieve and to identify families of experiments and/or artifacts that might be reused/shared among different experiments, such as, experimental knowledge and elements, such as, datasets, instruments, and data analysis techniques.

Furthermore, having continuous experimentation procedures incorporated in software projects is a great opportunity of iterative/incrementally evolve software processes and product artifacts. Still, it may increase reliability of a company from the perspective of customers.

3.4 General Techniques

In addition to our vision of the three pillars, we understand a couple of other techniques might also support them.

3.4.1 ***Mining Experiments.*** Once Software Engineering experiment concepts are well organized and formalized, we might use open repositories mostly based on conceptual modeling and ontologies. Such repositories can be mined, thus providing prospective experiment solutions.

Several mining software repositories techniques are available in the literature [10, 17, 20]. We have investigated such techniques to provide a way of proper experiments data analysis and exchanging. For instance, EvoOnt [7] might be adopted as a data format language, based on Web Ontology Language (OWL). Therefore, users might analyze experiments data for any reasons, including: obtaining a list of evidence by extracting experiments data; coupled change analysis of original and replicated experiments; perform meta-analysis of related experiments; evolution of an experiment by repository commits; and, to apply experimental-driven metrics to available data.

González-Barahona and Robles [4] provide mining repository solutions for promoting reproducibility of data-based on ESE studies. Such solutions might, for example, indicate the level of reproducibility a certain dataset has. The authors also provide a technique for assessing the reproducibility of studies supported by a tool.

We believe that Software Engineering experiment mining techniques are important to provide capability of identifying related experiments, their data provenance and curation, and the variant aspects of such experiments. These techniques might reveal families of experiments [14]. Furthermore, selecting unsuitable techniques to propose experiment families may undermine their potential to provide in-depth insights from the results and threaten Rep*. Thus, mining experiment techniques seems appropriate to mitigate such issue and to improve SE experiments.

3.4.2 ***Families of Experiments.*** Experiments with similar goals and results that might be combined to evolve certain SE research topic are defined as a "family of experiments." Such family basically provides capabilities to increase the reliability of findings and statistical power [14].

We focus our techniques and infrastructure on providing means for identifying and establishing families of experiments from the repositories. As we understand, aggregation techniques (*e.g.*, Individual Participant Data - IPD, Aggregated Data meta-analysis - AD, and Aggregation of p-values) [14] might be used to aid gathering up such families from our repositories, thus users benefit from first-hand knowledge of all the experiments settings and access to raw data.

As an exemplary first case, we are currently specifying a family of experiments for the evaluation of variability modeling approaches for UML-based software product lines from more than 15 experiments of our research group.

4 OUR ROADMAP

We established the following steps for our roadmap to improve SE experimentation based on the pillars we described:

(1) revisit existing and specify new guidelines for reporting experiments;
(2) specify an ontology for experiments;
(3) structure a TDR repository for experiments driven by the ontology;
(4) specify a DMP complying with the experiment repository;
(5) design and implement an experiment recommender system;
(6) specify an ontology for OER;
(7) structure a TDR repository driven by the OER ontology with integration to other repositories;
(8) design and implement an OER recommender system.

5 FINAL REMARKS

Although Software Engineering experiments are still not mature, the community has significantly advanced the discussion about this topic in the last years.

We strongly believe openness and reproducibility are keys to achieve considerable improvements on ESE. Therefore, we understand our vision will aid to improve SE experiments by: (i) motivating wide sharing of experimental elements and data (except those that require privacy); (ii) providing a common, open science-based infrastructure to experiment conduction; (iii) motivating teaching and learning of SE experimentation, increasing knowledge and promoting a culture of experimentation in academia and industry; (iv) streamlining the experimentation process in industry; (v) providing formal guidelines about how to disseminate work; and, (vi) facilitating sharing and retrieving well-conducted experiments.

REFERENCES

[1] O. Dieste, A. Grim, N. Juristo, and H. Saxena. 2011. Quantitative determination of the relationship between internal validity and bias in software engineering experiments: Consequences for systematic literature reviews. In *ESEM*. 285–294.
[2] A. Fabijan, P. Dmitriev, H. H. Olsson, and J. Bosch. 2017. The Evolution of Continuous Experimentation in Software Product Development: From Data to a Data-Driven Organization at Scale. In *ICSE*. 770–780.
[3] Dror G. Feitelson. 2015. From Repeatability to Reproducibility and Corroboration. *SIGOPS Oper. Syst. Rev.* 49, 1 (2015), 3–11.
[4] Jesús M. González-Barahona and Gregorio Robles. 2012. On the reproducibility of empirical software engineering studies based on data retrieved from development repositories. *Empirical Software Engineering* 17, 1 (2012), 75–89.
[5] J. Guo. 2016. Ontology Learning and Its Application in Software-Intensive Projects. In *ICSE*. 843–846.
[6] V. Kampenes. 2007. *Quality of design, analysis and reporting of software engineering experiments: A systematic review.* Ph.D. Dissertation. Faculty of Mathematics and Natural Sciences - University of Oslo, Oslo, Norway.
[7] C. Kiefer, A. Bernstein, and J. Tappolet. 2007. Mining Software Repositories with iSPAROL and a Software Evolution Ontology. In *MSR*. 10–10.
[8] B. Kitchenham, D.I.K. Sjøberg, O.P. Brereton, D. Budgen, T. Dybå, M. Höst, D. Pfahl, and P. Runeson. 2010. Can We Evaluate the Quality of Software Engineering Experiments?. In *ESEM*. 1–8.
[9] B. A. Kitchenham, D. I. K. Sjøberg, T. Dyba, D. Pfahl, P. Brereton, D. Budgen, M. Høst, and P. Runeson. 2012. Three empirical studies on the agreement of reviewers about the quality of software engineering experiments. *Information and Software Technology* 54, 8 (2012), 804–819.
[10] Matias Martinez and Martin Monperrus. 2019. Coming: A tool for mining change pattern instances from git commits. In *ICSE*. IEEE, 79–82.
[11] National Academies of Sciences. 2018. *Open Science by Design: Realizing a Vision for 21st Century Research.* The National Academies Press, Washington, DC.
[12] Roger D. Peng. 2011. Reproducible Research in Computational Science. *Science* 334, 6060 (2011), 1226–1227.
[13] Martin Robillard, Robert Walker, and Thomas Zimmermann. 2010. Recommendation Systems for Software Engineering. *IEEE Softw.* 27, 4 (2010), 80–86.
[14] A. Santos, O. Gómez, and N. Juristo. 2020. Analyzing Families of Experiments in SE: A Systematic Mapping Study. *IEEE Transactions on Software Engineering* 46, 5 (2020), 566–583.
[15] G. Schermann, J. Cito, and P. Leitner. 2018. Continuous Experimentation: Challenges, Implementation Techniques, and Current Research. *IEEE Software* 35, 02 (2018), 26–31.
[16] Forrest J. Shull, Jeffrey C. Carver, Sira Vegas, and Natalia Juristo. 2008. The role of replications in Empirical Software Engineering. *Empirical Software Engineering* 13, 2 (2008), 211–218.
[17] Davide Spadini, Maurício Aniche, and Alberto Bacchelli. 2018. Pydriller: Python framework for mining software repositories. In *ESEC/FSE*. 908–911.
[18] C. Wohlin, P. Runeson, M. Höst, C. Ohlsson, B. Regnell, and A. Wesslén. 2012. *Experimentation in software engineering* (2nd. ed.). Springer Publishing Company, New York, NY.
[19] Sezin Gizem Yaman, Myriam Munezero, Jürgen Münch, Fabian Fagerholm, Ossi Syd, Mika Aaltola, Christina Palmu, and Tomi Männistö. 2017. Introducing continuous experimentation in large software-intensive product and service organisations. *J. Syst. Soft.* 133 (2017), 195–211.
[20] Andy Zaidman, Bart Van Rompaey, Serge Demeyer, and Arie Van Deursen. 2008. Mining software repositories to study co-evolution of production & test code. In *ICST*. IEEE, 220–229.