Reproducibility and Credibility in Empirical Software Engineering: A Case Study based on a Systematic Literature Review of the use of the SZZ algorithm

Gema Rodríguez-Pérez[1], Gregorio Robles[1], Jesús M. González-Barahona[1]

Abstract

Context: Reproducibility of Empirical Software Engineering (ESE) studies is an essential part for improving their credibility, as it offers the opportunity to the research community to verify, evaluate and improve their research outcomes. **Objective:** We aim to study reproducibility and credibility in ESE with a case study, by investigating how they have been addressed in studies where SZZ, a widely-used algorithm by Śliwerski, Zimmermann and Zeller to detect the origin of a bug, has been applied. **Methodology:** We have performed a systematic literature review to evaluate publications that use SZZ. In total, 187 papers have been analyzed for reproducibility, reporting of limitations and use of improved versions of the algorithm. **Results:** We have found a situation with a lot of room for improvement in ESE as reproducibility is not commonly found; factors that undermine the credibility of results are common. We offer some lessons learned and guidelines for researchers and reviewers to address this problem. **Conclusion:** Reproducibility and other related aspects that ensure a high quality scientific process should be taken more into consideration by the ESE community in order to increase the credibility of the research results.

Keywords: Credibility, reproducibility, SZZ algorithm, systematic literature review.

Email addresses: gema.rodriguez@urjc.es (Gema Rodríguez-Pérez), grex@gsyc.urjc.es (Gregorio Robles), jgb@gsyc.es (Jesús M. González-Barahona)
[1]Universidad Rey Juan Carlos, Madrid

1. Introduction

The reproducibility of a study is one of the essential characteristics of the scientific method [1]. In particular, without reproducibility in Empirical Software Engineering (ESE), academics and practitioners do not have the opportunity to verify, evaluate or improve research outcomes of fellow researchers. This may raise concerns about the reliability of the results. Fortunately, this issue has received increasing attention during the last years. For instance, there are specific tracks in international conferences, such as the REproducibility Studies and NEgative Results *RENE*[2] at the International Conference on Software Analysis, Evolution, and Reengineering (*SANER*)[3]. Many researchers have become aware of the importance and the problem of reproducibility in ESE. So, Juristo and Vegas state that reproducibility "is important to increase and consolidate the body of empirical knowledge" [2], and Robles shows that reproducibility may be hindered by many factors [3]. In this paper, we adopt the definition of reproducibility by Madeyski and Kitchenham [4] which says that "reproducible research is the extent to which the report of a specific scientific study can be reproduced (in effect, compiled) from the reported text, data and analysis procedures, and thus validated by other researchers." Despite the fact that reproducibility and replication are different concepts, we assume that when a research work incorporates reproducibility, it is more likely to be replicated.

The credibility of a study is related to the level of confidence practitioners and researchers can have in its conclusions and results. Common wisdom, assumptions, intuitions and speculations are not reliable sources of credible knowledge, making ESE research necessary [5]. Researchers should find ways to help others to replicate their results. Empirical studies gain credibility by reproducing their results, because these new results can be compared with the original outcomes, supporting them or not. However, reproducibility (and by extension credibility, since multiple replications of an experiment increase it [2])

[2] http://saner.unimol.it/negativerestrack

[3] http://saner.unimol.it/index

may be a complex task, as access to data sources, use of specific tools, availability of detailed documentation has to be handled. Detecting factors that hinder reproducibility should help strengthening the credibility of the empirical studies [6].

This paper addresses how the scientific practice of the ESE research community affects the reproducibility and credibility of the results. In particular, we want to address studies that use techniques or algorithms based on assumptions and heuristics. This is the case of the SZZ algorithm, published in 2005 in "When do changes induce fixes?" by Śliwerski, Zimmermann and Zeller [7] at the MSR workshop[4]. SZZ has been largely used in academia, counting, as of February 2018, with more than 590 citations in Google Scholar[5].

In this paper, we present a Systematic Literature Review (SLR) on the use and reproducibility of the SZZ algorithm in 187 academic publications. Although this paper offers a case study of a widely used research *technique*, it goes beyond credibility and reproducibility to offer a wider picture of lack of high quality scientific practices. The credibility of results presented in ESE research is negatively affected by this situation.

We make five significant contributions by presenting:

1. An overview on the impact that the SZZ has had so far in ESE.
2. A measure of reproducibility of 187 studies that use the SZZ algorithm.
3. An overview of how studies that use the SZZ algorithm address the reproducibility in their research work.
4. An analysis of how these studies manage the limitations of SZZ.
5. Lessons learned for researchers wanting others to reproduce their studies, and for the software engineering research community aiming at promoting reproducible studies.

[4]MSR is today a working conference, but at that time it was a co-located workshop with ICSE in its second edition.

[5]https://scholar.google.es/scholar?cites=3875838236578562833

The remainder of this paper is structured as follows. First of all, we present a detailed description of the SZZ algorithm in Section 2. Next, we describe the method used in the SLR, with the research questions and inclusion/exclusion criteria in Section 3. Section 4 describes how to extract the data, followed by the results from systematically analyzing 187 publications in Section 5. Related work is then presented in Section 6. We then discuss the implications and offer lessons learned in Section 7. The threats to the validity of our study can be found in Section 8. Finally, we draw conclusions and point out future research in Section 9.

2. The SZZ Algorithm

In software engineering research, SZZ is a popular algorithm for identifying bug-introducing changes [SLR[8]][6]. SZZ relies on historical data from versioning and issue tracking systems to identify change sets in the source code that induce bug fixes. The algorithm can be divided in two parts.

In the first part, the algorithm identifies bug-fixing commits by matching commits with bug reports labeled as *fixed* in the issue tracking system. The matching is based on regular expressions to identify bug numbers and keywords in the commit messages that are likely to point out a real bug fixing change.

The second part of the algorithm is concerned with the identification of the bug introducing commit(s). The algorithm employs the *diff* functionality implemented in the control version systems to determine the lines that have been changed (to fix the bug) between the fix commit version and its previous version. Then, using *annotate/blame* functionality, SZZ locates the commit that modified or deleted those lines for the last time in previous change(s), and, if they were committed before the bug was reported, those change(s) are flagged as suspicious of being the bug introducing change(s).

[6] Publications included the Systematic Literature Review (SLR) are cited in this paper using following format: [SLR#ref]

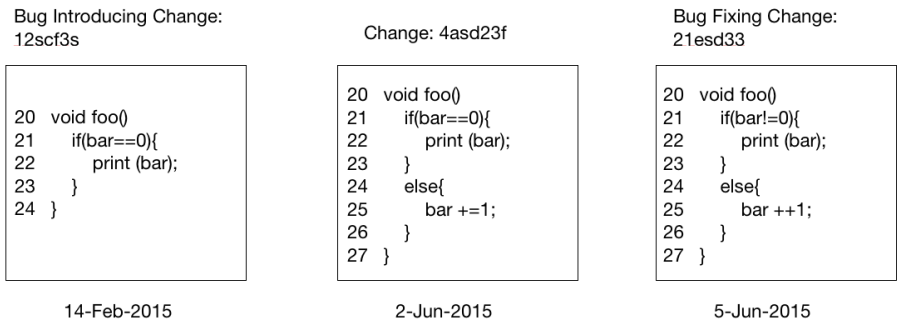

Figure 1: Example of changes committed in a file, the first change is the bug introducing change and the third change is the bug fixing change.

As a representative example, Figure 1 represents three different snapshots of the code. The first change, *12cf3s*, is the bug introducing change; the bug is introduced in line 21 where the condition for the *if* statement is used incorrectly. The second change, *4asd23f*, adds code to the *foo()* function in lines 24, 25 and 26 after the bug was reported. The third change, *21esd33*, fixes the bug by modifying two lines: line 21 (the buggy one) and line 25.

Figure 2 shows how the algorithm works. First, after the introduction of bug notification *#159* in the issue tracking system, the algorithm identifies the bug fixing commit *21esd33* by looking in the log for a commit with the commit message containing bug number *#159*. After that, the second part of the algorithm searches for the bug introducing commit by using *diff* and *annotate/blame*. In this example, lines 21 and 25 have been changed in the bug fixing commit in order to fix the bug, so both lines are suspicious of being the ones where the bug was introduced. These lines have been inserted in two different commits. However, as line 25 was introduced in a commit after the bug was reported, the bug introducing commit can be identified as the one that changed line 21. To sum up, SZZ points out to *12cf3s* as being the bug introducing change.

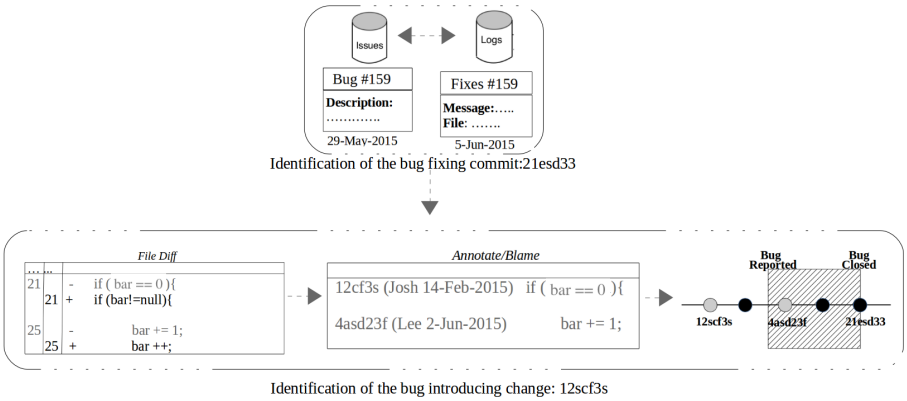

Figure 2: First and Second part of the SZZ algorithm

2.1. Limitations of the SZZ Algorithm

Despite SZZ being largely used in ESE to locate bugs, it suffers from multiple limitations which make it error prone. Table 1 offers a detailed overview of the limitations in SZZ as reported in the literature.

In the first part of the algorithm the limitation lies in how bug reports are linked to commits. If the fixing commit (in the versioning system) does not contain a reference to the bug (usually the reference is the unique id assigned by the bug tracking system, but it could be certain keywords as well), it is very difficult to link both data sources. Sometimes this linking is incorrect due to the bug fixing commit not corresponding to the bug report. If the fixing commit is not identified, the bug introducing change cannot be determined and this causes a *false negative*. Studies have demonstrated that 33.8% [SLR[9]] to 40% [SLR[10]] of the bugs in the issue tracker are misclassified, i.e., issues categorized as bugs are actually functionality requests or refactoring suggestions. We have a *false positive* when a bug report does not describe a *real* bug, but a fixing commit is linked to it. Herzig *et al.* pointed out that 39% of files marked as defective have never had a bug [SLR[9]].

In the second part of the algorithm, lines might be incorrectly identified by SZZ as the ones where the bug was introduced, causing a *false positive*. It

Table 1: Limitations when using SZZ.

Part	Type	Description
First part	Incomplete mapping [SLR[14]	The fixing commit cannot be linked to the bug
	Inaccurate mapping [SLR[15]]	The fixing commit has been linked to a wrong bug report, they don't correspond to each other
	Systematic bias [SLR[14]	Linking fixing commit with no *real* bug report
Second part	Cosmetic changes, comments, blank lines [SLR[16]]	Variable renaming, indentation, split lines, etc.
	Added lines in fixing commits [SLR[8]]	The new lines can not be tracked back
	Long fixing commits [SLR[8]]	The larger the fix, the more false positives
	Semantic level is weak [SLR[11]]	Changes with the same behavior are being blamed
	Correct changes at the time of being committed [SLR[8]]	Changes in other parts of the source code base trigger a bug issue in another part
	Commit Squashing [17]	Might hide the bug introducing commit, loosing authorship information

may also be that the buggy line was not analyzed by SZZ, producing a *false negative*. In some cases, the bug had been introduced before the last change to the line; then, the history of the line has to be traced back until the *true* source of the bug is found [SLR[11]]. An example of this can be found when SZZ flags changes to *style* (i.e., non-semantic/syntactic changes such as changes to white spaces, indentation, comments, and some changes that split or merge lines of code) as bug introducing changes [SLR[8]], or when a project allows *commit squashing*, since this option removes authorship information resulting in more *false positives*. It may also happen that the bug may have been caused by a change in another part of the system [12]. A final possibility is that the bug fix modified the surrounding context rather than the problematic lines, misleading the algorithm [SLR[13]].

2.2. Enhancements proposed to SZZ

After enumerating the limitations of the SZZ found in the literature, we focus on how researchers have addressed them over time, and in how far the enhancements mitigate them.

Da Costa *et al.* have created a framework to eliminate unlikely bug introducing changes from the outcome of SZZ. Their framework is based on a set of requirements that consider the dates of the suspicious commit and of the bug report [SLR[8]]. By removing those commits that do not fulfill these requirements, the number of false positives provided by SZZ is lowered significantly.

The misclassification problem has been further investigated by researchers, aiming at mitigating the limitations found in the first part of SZZ [SLR[9]],[SLR[18]],[SLR[10]]. Tools and algorithms have been created based on the information from version control systems and issue tracking systems to map bug reports to fixing commits [SLR[19]],[SLR[20]],[21], [22]. For instance, GitHub encourages linkage by automatically closing issues if the commit message contains *#number*. Several authors have suggested semantic heuristics [SLR[23]],[24],[SLR[25]]; others have proposed solutions that rely on feature extraction from bugs and issue tracking system metadata [SLR[19]],[SLR[20]],[SLR[21]]. In addition, many Free/Open Source Software projects have adopted as a good practice to use the "# fix-bug —" keywords in their commit comments when a bug is fixed, as it has been reported for the Apache HTTP web server[7] in [SLR[26]], and for VTK[8], and ITK[9] in [27].

As a result of these efforts, the first part of SZZ has seen how its accuracy has significantly increased. For example, *FRlink*, an existing state-of-the-art bug linking approach, has improved the performance of missing link recovery compared to existing approaches, and it outperforms the previous one by 40.75% (in F-Measure) when achieving the highest recall [22].

Related to the second part of SZZ, two main improvements have been proposed in the literature. We call them SZZ-1 and SZZ-2 in this paper:

- *SZZ-1*) Kim *et al.* suggest an SZZ implementation that excludes cosmetic changes, and propose the use of an *annotation graph* instead of using

[7] https://httpd.apache.org/
[8] http://www.vtk.org/
[9] https://itk.org/

annotate[10] [SLR[16]].

- *SZZ-2*) Williams and Spacco propose to use a mapping algorithm instead of annotation graphs; this approach uses weights to map the evolution of a source code line and ignores comments and formatting changes in the source code with the help of `DiffJ`, a Java-specific tool [SLR[11]].

The second part of the algorithm, however, has still room for further improvements. Addressing its limitations often requires a manual, tedious validation process.

3. Research Method

The purpose of a SLR is to identify, evaluate and interpret all available studies relevant to a particular topic, research question, or effect of interest [28]. A SLR provides major information about the effects of a particular topic across a wide range of previous studies and empirical methods. As a result, a SLR should offer evidence with consistent results and suggest areas for further investigation. We follow the SLR approach proposed by Kitchenham and Charters [28].

3.1. Research Questions

The aim of this SLR is to study and analyze the credibility and reproducibility of the SZZ algorithm. Therefore, we address following research questions (RQs):

RQ1: What is the impact of the SZZ algorithm in academia?

We want to investigate the impact of SZZ in academia. We have divided this RQ in several subquestions:

RQ1.1: How many publications use the complete SZZ algorithm?

[10] Notice that annotate is used in SVN and blame is used in git.

Motivation. The SZZ algorithm has been shown to be a key factor to locate when a change induced fixing commits. Many papers, however, use only the first part of the algorithm to link bug fix reports to commits. As the second part of SZZ is the one that shows to have significant threats, we identify those publications that use both parts, or at least the second part, of the SZZ algorithm. In addition to this, we offer other metrics on the publications, such as the number of authors and the geographic diversity of the institutions they work for, in order to provide insight of how widespread the use of SZZ is.

RQ1.2: How has the impact of SZZ changed over time?

Motivation. Our goal is to visualize the impact of SZZ over time, to see whether SZZ has been only adopted in the first years after its publication or if it is still widely in use nowadays.

RQ1.3: What are the most common venues with publications using the SZZ algorithm?

Motivation. Our goal is to addresses the maturity and diversity of the publications where SZZ has been used in order to understand its audience. We address the *maturity* of a publication by analyzing if it has been accepted in a workshop, a conference, a journal, or a *top* journal. Diversity is given by the number of distinct venues where publications using SZZ can be found.

RQ2: Are studies that use SZZ reproducible?

Motivation. Reproducibility is a crucial aspect of a credible study in ESE [1]. Piwowar *et al.* state that reproducibility improves the impact of research [29]. In addition, when a research work incorporates reproducibility, it is more likely to be replicated. However, there is evidence in the ESE literature that replicable studies are not common [3]. By providing a replication package (or a detailed description of the analysis and the environment and data used), the authors facilitate others to replicate or to reproduce their experiment, which increases the credibility of their results [2]. In addition, replication packages help in the training of novice researchers [4].

RQ3: Do the publications mention the limitations of SZZ?

Motivation. We have already shown that limitations of SZZ are well-known in the research literature. With this question, we would like to find out how many papers report any of these. Therefore, we study whether authors mention the limitations of SZZ that may affect their findings, be it in the *description of the method*, in the *threats to validity* or in the *discussion*.

RQ4: Are the improvements to SZZ (SZZ-1 and SZZ-2) used?

Motivation. The improved versions of the original SZZ algorithm address some of its limitations. We analyze whether any of the improvements to the SZZ algorithm can be found in the primary studies included in the SLR. Thus, we search for any mention of their use, be it in the *description of the method* or in the *threats to validity*. Answering this research question allows to further understand how authors who use SZZ behave given the limitations of SZZ.

RQ5: Does the reproducibility of the studies improve when authors (1) report limitations or (2) use improved versions of SZZ?

Motivation. We investigate whether the reporting of limitations (from *RQ3*) or the use of improvements to SZZ (from *RQ4*) are related to the reproducibility of a paper (*RQ2*).

We hypothesize that authors who *report limitations* or *use improved versions of SZZ* should be more sensitive to *fostering the reproducibility* of their research, as they know that SZZ has limitations. Thus, they offer the research community a way to mitigate this problem.

However, at the same time, we understand that *reporting of limitations* and *use of improved versions of SZZ* offer short-term gains for the authors by increasing their chances of publishing their research, while reproducibility is a long-term benefit whose main beneficiaries are, in general, other researchers. In such a scenario, reproducibility may be neglected.

3.2. Inclusion Criteria

After enumerating the research questions, we present the inclusion and exclusion criteria for the SLR. In addition, we describe the search strategy used for primary studies, the search sources and the reasons for removing papers from the list.

The inclusion criteria address all published studies written in English that cite either:

1. the publication where SZZ was originally described,"When do changes induce fixes?" [7], or

2. (at least) one of the two publications with improved versions of the algorithm, "Automatic Identification of Bug-Introducing Changes" [SLR[16]] and "SZZ Revisited: Verifying When Changes Induce Fixes" [SLR[11]].

There was no need to further investigate the references to the resulting set of publications (a process known as *snowballing*): if one of these papers contained as well a reference to the papers that fit the inclusion criteria, we suppose it is already in our sample.

Before accepting a paper into the SLR, we excluded publications that are duplicates, i.e., a *matured* version (usually a journal publication) of a *less matured* version (conference, workshop, PhD thesis...). In those cases, we only considered the *matured* version. When we found a short and a long version of the same publication, we have chosen the longer version. However, in those cases where the publication is a PhD thesis and a related (peer-reviewed) publication exists in a workshop, conference or journal, we have discarded the thesis in favor of the latter, because conference and journal publications are peer-reviewed and PhD theses not. Documents that are a *false alarm* (i.e., not a *real*, scientific publication) have also been excluded.

3.3. Search Strategy used for Primary Studies

The studies were identified using Google Scholar and Semantic Scholar as of November 8th 2016. We have looked exclusively in Google Scholar and Semantic

Table 2: Number of citations of the SZZ, SZZ-1 and SZZ-2 publications by research databases.

	Google Scholar	Semantic Scholar	ACM Digital Library	CiteSeerX
# SZZ	493	295	166	26
# SZZ-1	141	100	60	18
# SZZ-2	26	15	8	0

Scholar because of i) their high accuracy in locating citations, providing more results than other databases (from Table 2 it can be seen that they contain three times more citations than other platforms, such as the ACM Digital Library), and ii) because we have seen that they offer a superset of the other databases, i.e., we have checked that no publication in the other sources is missing from the list provided by Google Scholar and Semantic Scholar. However, Google Scholar gives many *false alarms*, in the sense that they are not publications but slide sets, notes, etc. Examples of those *false alarms* are "Strathprints Institutional Repository"[11] or "Home Research"[12]), which we removed manually from our set. Some academic databases which are commonly used for SLRs, such as Scopus, could not be employed to gather citations, because SZZ was published at a time when MSR was a workshop, and thus the original publication [7] is not included in those databases.

3.3.1. Study Selection Criteria and Procedures for Including and Excluding Primary Studies

Table 3 shows that our searches elicited 1,070 citation entries. After applying the inclusion criteria described above, we obtained a list of 458 papers. This process was performed by the first author. The process is objective, as it involves discarding false alarms, duplicates, and papers not written in English.

Then, the first author analyzed the remaining 458 papers looking for the use of SZZ, SZZ-1 and SZZ-2 in the studies. This resulted in 193 papers being removed because of three main reasons: i) they only cited the algorithm as part

[11] `https://core.ac.uk/download/pdf/9032200.pdf`

[12] `http://ieeexplore.ieee.org/document/1382266/#full-text-section`

Table 3: Number of papers that have cited the SZZ, SZZ-1 and SZZ-2 publications by joining the research databases Google Scholar and Semantic Scholar during each stage of the selection process.

Selection Process	#SZZ	#SZZ-1	#SZZ-2
Papers extracted from the databases	788	241	41
Sift based on false alarms	29 removed	10 removed	2 removed
Sift based on not available/English writing	40 removed	4 removed	0 removed
Sift based on duplicates	308 removed	187 removed	32 removed
Full papers considered for review	411	40	7
Removed after reading	149 removed	32 removed	4 removed
Papers accepted to the review	262	8	3

of the introduction or related work but they never used it, ii) they only cited the algorithm to support a claim during their results or the discussion, and iii) the papers were essays, systematic literature reviews or surveys. This process was discussed in advance by all the authors. The second author partially validated the process by analyzing a random subset comprising 10% of the papers. The agreement between both authors was measured using Cohen's Kappa coefficient, which resulted in a value of 1 (perfect agreement). These papers were removed on the basis that they do not answer our research questions. After this process 273 papers were included in this SLR.

4. Study Quality Assessment

This section explains how we obtained the data to show an overall picture of credibility and reproducibility of the SZZ algorithm in ESE studies.

4.1. Quality Assessment Criteria

Our approach to study the quality assessment is based on Kitchenham and Charter's [28] concept of quality. Thus, our assessment is focused on identifying only those papers that report factors related with the credibility and reproducibility of the studies using SZZ. The specific criteria are described in the next phase.

4.1.1. Phase 1: Establishing that the study uses the complete SZZ algorithm

In this SLR we only consider studies that use the complete algorithm, or at least its second part. Even though limitations have been reported in both parts of the SZZ algorithm (see Section 2), most of the limitations present in the first part have been successfully addressed in the last years.

To analyze the ease of reproducibility of each study, based on our experience in conducting similar research [3, 1], we looked for (1) a replication package provided by the authors or (2) a detailed description. A detailed description must have: (a) precise dates when the data were retrieved from the projects under analysis, (b) the versions of the software and systems used in the analysis, (c) a detailed description of the methods used in each phase of the study, and (d) enumerate the research tools used. It should be noted that we did not inspect whether the replication package is still available, or whether elements in the package make the study reproducible. We assume that the authors and the reviewers checked the availability of the replication package at the time of submission. We do not claim the availability of these packages in the long term. For instance the reproduction package from the original SZZ paper [7] is no longer available.

Applying our criteria to the set of 273 papers, we obtain 187 papers that fulfill this criterion.

4.2. Extracting Data from Papers

We have read and analyzed the 187 papers, and extracted the following data information in a first reviewing phase to answer the RQ1:

1. Title,
2. Authors,
3. Countries of the authors' institutions,
4. Purpose of the study,
5. Outcome of the study, and

6. Venue and class of publication (journal, conference, workshop or university thesis).

Then, in a second phase, we have carefully analyzed each publication looking:

1. For a replication package (as in [3]), to answer RQ2.
2. For a detailed description of the methods and data used (as in [3]), to answer RQ2.
3. Whether limitations are mentioned, to answer RQ3.
4. Whether a manual inspection to verify the results has been done, to answer RQ3.
5. Whether authors use an improved version of SZZ (differentiating between a version found in the research literature and *ad-hoc* improvements implemented by the authors), to answer RQ4.

The first author of this paper extracted data in the two phases. Then, the second author of this paper randomly validated 10% of these results to ensure that the articles included in the SLR were suitable. The agreement using Cohen's Kappa coefficient was 0.73 (good agreement). This coefficient was computed to see whether a paper is using the SZZ algorithm or not, and to identify which part is being used[13].

4.3. Overview across Studies

Cruzes and Dybå reported that synthesizing findings across studies is specially difficult, and that some SLRs in software engineering do not offer this synthesis [30]. For this SLR we have extracted and analyzed both quantitative and qualitative data from the studies, but we have not synthesized the studies,

[13]The discordance was primarily because in some papers the description of the methodology is spread over the paper, and some parts may have been omitted by one of us. This occurs more often in those papers that do not explicitly mention the algorithm's acronym (SZZ) or support the description with a citation.

as they are too diverse. Doing a meta-analysis would offer limited and unstructured insight [31] and results would suffer from some of the limitations in SLRs published in other disciplines [32]. Thus, we combined both our quantitative and qualitative data to generate an overview of how authors have addressed the reproducibility and credibility of the studies. The results are presented in Section 5.

In addition, we have constructed a quality measure[14] that assesses the ease of reproducibility of a study. This measure is based on the score of five characteristics of the papers that we have looked for in the second reviewing phase. If questions were answered positively, the paper was marked with a positive score, otherwise with 0:

1. Does the study report limitations of using SZZ? (score = 1 point)
2. Do the authors carry out a manual inspection of their results? (score = 1 point)
3. Does the study point to a reproducibility package? (score = 2 point)
4. Does the study provide detailed description of the methods and data used? (score = 1 point)
5. Does the study use an improved version of SZZ? (score = 2 point)

According to the authors' opinion, those elements with a higher impact on ease reproducibility of the the studies are scored with 2 points. Partial scores are summed up to obtain an overall score. Table 4 offers a mapping of this overall measure with the ease of reproducibility of a study.

5. Results

This section presents the results of our SLR. All the details, at the publication level and for each of the RQs, can be found in the on-line replication

[14]The main goal of this quality measure is to determine the reproducibility and credibility of the studies in the moment in which the study was submitted.

Table 4: Mapping of overall score and the quality measure on ease of reproducibility of a study.

Score	Quality Measure
0 – 1	*Poor* to be reproducible and to have credible results
2 – 4	*Fair* to be reproducible and to have credible results
5 – 6	*Good* to be reproducible and to have credible results
7	*Excellent* to be reproducible and to have credible results

Table 5: Quantitative results from the 187 studies.

Purpose:	Outcome:	Common Metrics:	Versioning:
Bug Prediction (BP) (32%) Bug Proneness (BProne) (27%) Bug Detection (BD) (22%) Bug Location(BL) (19%)	New Method/Approach (NM) (41%) Empirical Study (ES) (37%) New Tool (NT) (8%) Human Factors (HF) (8%) Replication (R) (3%) Create a Dataset (D) (2%)	Commits (26%) Bug Reports (16%) LOC (15%) Changes (12%) Files (8%) Faults (8%) Revisions (6%) Modules (4%)	CVS (50%) Git (28%) SVN (25%) Mercurial (6%)

package (see Section 10).

5.1. Data analysis

Here, we present our quantitative and qualitative data analysis extracted from the 187 studies analyzed during the SLR. Table 5 summarizes the frequency of the different outcomes, purposes, versioning systems and metrics in the studies. The most common purpose is *bug prediction* followed by *bug proneness*. The most common outcomes are the *development of a new method/approach* and the *evidence of empirical results*.

The qualitative data, summarized in Table 6, consists in the number of papers that belong to each quality measure levels. We can see that 67% of the papers present a fair reproducibility measure, and only 2% of them provide excellent means to be reproducible.

The combination of quantitative and qualitative data is addressed in Table 7, where the purpose and outcome of each individual study is grouped according to our quality measure. It can be observed that the distribution by purpose or outcome does not differ much from the global distribution, although studies

Table 6: Results of the calculating the quality measure of reproducibility and credibility.

Quality Measure	#papers
Poor to be reproducible and to have credible results	34 (18%)
Fair to be reproducible and to have credible results	126 (67%)
Good to be reproducible and to have credible results	24 (13%)
Excellent to be reproducible and to have credible results	3 (2%)

offering new tools (NT) and replications (R) offer slightly better results than the rest.

Table 7: Distribution of the ease of reproducibility quality measure of studies depending on purpose and outcome. Acronyms are defined in Table 5.

Quality	**Purpose**				**Outcome**					
	BP	**BProne**	**BD**	**BL**	**ES**	**HF**	**NM**	**NT**	**R**	**D**
Poor	15 (25%)	10 (19%)	6 (15%)	3 (9%)	14 (20%)	2 (13%)	17 (22%)	1 (7%)	0 (0%)	0 (0%)
Fair	38 (64%)	32 (62%)	29 (72%)	27 (77%)	49 (70%)	13 (81%)	50 (65%)	8 (53%)	3 (60%)	3 (75%)
Good	6 (10%)	9 (17%)	5 (13%)	4 (11%)	5 (7%)	1 (10%)	10 (13%)	5 (33%)	2 (40%)	1 (25%)
Excellent	1 (1%)	1 (2%)	0 (0%)	1 (3%)	2 (3%)	0 (0%)	0 (0%)	1 (7%)	0 (0%)	0 (0%)

The type of paper (journal, conference, workshop and university thesis) as well as the size (short, medium, long) of the paper might be a restriction to provide means of reproducibility. We have assumed size to be less than 8 pages for short, from 9 to 50[15] pages for medium, and more than 50 pages for long publications. Table 8 reports the type and the size of each individual study grouped according to our quality measure of reproducibility and credibility. Again, the results do not differ much from the global distribution. However, it can be seen that (a) workshop papers perform worst than the rest, and (b) reproducibility increases slightly with the size of the publication.

[15] We argue that master and PhD theses should be categorized as long publications. We have chosen 50 as limit between medium and long papers because in our data we have observed that all master theses have more than 50 pages whereas none of the journals articles have more than 50 pages.

Table 8: Results of the measure the ease of reproducibility and credibility of the studies depending on the type of paper and their size.

Quality	Venue				Size		
	Journal	**Conference**	**Workshop**	**University**	**Short**	**Medium**	**Long**
Poor	5 (12%)	20 (20%)	5 (38%)	4 (13%)	10 (26%)	21 (17%)	3 (13%)
Fair	32 (76%)	65 (63%)	7 (54%)	22 (74%)	25 (64%)	85 (68%)	16 (70%)
Good	4 (10%)	15 (15%)	1 (8%)	4 (13%)	4 (10%)	16 (13%)	4 (17%)
Excellent	1 (2%)	2 (2%)	0 (0%)	0 (0%)	0 (0%)	3 (2%)	0 (0%)

5.1.1. RQ1.1: How many publications use the complete SZZ algorithm?

Our final sample included 187 papers. These publications have been authored by more than 370 different authors from institutions located in 24 different countries, offering evidence that SZZ is a widespread and well-known algorithm.

5.1.2. RQ1.2: How has the impact of SZZ changed over time?

Figure 3 shows the evolution of the number of publications that have cited and used SZZ, SZZ-1 or SZZ-2 up to November 2016. The SZZ algorithm was published in 2005 and afterwards 178 studies have cited it. SZZ-1 was published in 2006 and its number of citations is 53. Finally, SZZ-2 was published in 2008 and counts with 16 publications[16].

The number of studies per year peaked in 2013, with 30 papers using a SZZ version. In general, since 2012 the number of studies using this algorithm seems to have stabilized with over 15 citations/year for the use of the complete algorithm.

5.1.3. RQ1.3: What are the most common venues with publications using the SZZ algorithm?

Table 9 shows the different types of venues with publications where SZZ has been used. We have classified the venues in four different categories: university theses, workshop papers, conference and symposium publications, and journal

[16]Note that a paper can cite more than one version of SZZ.

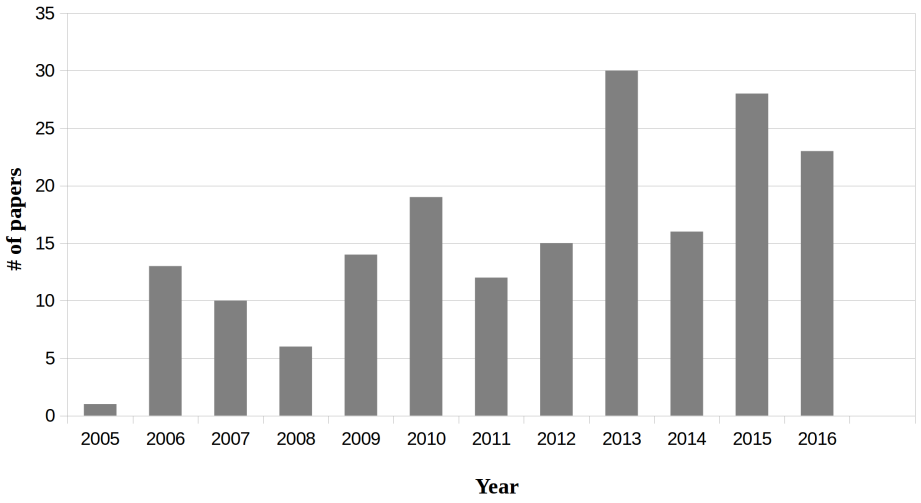

Figure 3: (RQ1.2) Sum of the number of publications using the (complete) SZZ, SZZ-1 or SZZ-2 by year of publication (N=187).

articles. Master theses, student research competitions and technical reports have been grouped under *university theses*. Diversity and maturity can be found in the sample, as it can be seen from the number of different venues (second column in Table 9) and the considerable number of journal publications (third column in Table 9).

Table 10 offers further insight into those venues that have published more studies that use SZZ. The most frequent one is the conference where SZZ itself was presented, the Working Conference on Mining Software Repositories

Table 9: (RQ1.3) Most frequent types of publications using (the complete) SZZ (N=187). *# different* counts the different venues, *# publications* counts the total number of publications in that type of venues.

Type	# different	# publications
Journals	21	42
Conferences & Symposiums	40	102
Workshops	13	13
University theses	20	30

Table 10: (RQ1.3) Most popular media with publications using SZZ, SZZ-1 and SZZ-2 (N=187). "J" stands for journal and "C" for conference/symposium.

Type	Name	Rating	# papers)
C	Conf Mining Softw Repositories (MSR)	Class 2 - CORE A	15 (8%)
C	Intl Conf Software Eng (ICSE)	Class 1 - CORE A*	12 (6%)
C	Intl Conf Soft Maintenance (ICSME)	Class 2 - CORE A	10 (5%)
J	Empirical Software Eng (EmSE)	JCR Q1	9 (5%)
J	Transactions on Software Eng (TSE)	JCR Q1	9 (5%)
C	Intl Symp Emp Soft Eng & Measurement (ESEM)	Class 2 - CORE A	8 (4%)
C	Intl Conf Automated Softw Eng (ASE)	Class 2 - CORE A	7 (4%)
C	Symp Foundations of Software Eng (FSE)	Class 1 - CORE A*	6 (3%)

(MSR). Two top conferences, such as the International Conference on Software Maintenance and Evolution (ICSME) and the International Conference of Software Engineering (ICSE), are second and third. SZZ can also been frequently found in high quality journals, such as Empirical Software Engineering (EmSE) and Transactions on Software Engineering (TSE). The quality rating of conferences given in Table 10 has been obtained from the GII-GRIN-SCIE (GGS) Conference Rating[17]; Class 1 (CORE A*) conferences are considered *excellent, top notch events* (top 2% of all events), while Class 2 (CORE A) are *very good events* (given by the next top 5%). For journals, we offer the quartile as given by the well-known Journal Citation Reports (JCR) by Clarivate Analytics (previously Thomson Reuters).

RQ1: The impact of the SZZ algorithm is significant: 458 publications cite SZZ, SZZ-1 or SZZ-2; 187 of these use the complete algorithm. The popularity and use of SZZ has risen quickly from its publication in 2005 and it can be found in all types of venues (high *diversity*), ranging from top journals to workshops and PhD theses; SZZ related publications have often been published in high quality conferences and top journals (high *maturity*).

[17] http://gii-grin-scie-rating.scie.es/

Table 11: (RQ2) Publications by their reproducibility: Rows: *Yes* means the number of papers that fulfill each column, whereas the complement is *No*. Columns: *Package* is when they offer a replication package, *Environment* when they provide a detailed methodology and dataset. Note that *Both* is the intersection of *Package* and *Environment*. (N=187)

	Package Only	Environment Only	Both	None
Yes	19	72	24	72
No	168	96	163	115

5.2. RQ2: Are studies that use SZZ reproducible?

To provide trustworthy results in ESE research, authors should offer a replication package and/or a detailed description of the research steps, and the environment and data used. This would allow others to reproduce or replicate their studies [1].

Table 11 shows the number of analyzed studies that a) offer a replication package or b) have carefully detailed the methodology and the data used to allow the reproducibility of their studies. We have classified the publications in four groups: i) publications that offer a replication package (*Package*), ii) publications that detail the methodology and data used (*Environment*), iii) publications that have both (*Both*), and iv) none (*None*).

From the 187 analyzed publications, 43 offer a replication package, and 96 carefully detail the steps followed and the data used. Furthermore, only 24 provide both, the replication package and the detailed methodology and data. 72 of the papers do not offer a replication package or a detailed description of the methodology and data.

RQ2: Only 13% of the publications using any of the variants of SZZ provide a replication package and carefully describe each step to make reproduction feasible. 39% of the papers do not provide replication package or a detailed description of each step, making their reproduction very unlikely.

Table 12: (RQ3) Number of publications that mention limitations of SZZ in their Threats To Validity (TTV). Mentions can be to the first (TTV-1st), second (TTV-2nd) or both parts (Complete-TTV). The absence of mentions is classified as No-TTV. Note that *Complete-TTV* is the intersection of *TTV-1* and *TTV-2*.

	No-TTV	TTV-1st only	TTV-2nd only	Complete-TTV
Yes	94	44	10	39
No	93	143	177	148

5.3. RQ3: Do the publications mention the limitation of the SZZ?

We have classified publications in four groups, depending on how they address limitations in SZZ as a threat to validity (TTV). Thus, we have publications that i) mention limitations of the complete algorithm ($Complete\text{-}TTV$), ii) mention only limitations in the first part ($TTV\text{-}1^{st}$), ii) mention only limitations in the second part ($TTV\text{-}2^{nd}$), and iv) do not mention limitations at all ($No\text{-}TTV$).

Table 12 offers the results of our analysis. From the 187 publications, only 39 mention limitations of the complete SZZ as a threat to validity, whereas 83 refer to limitations in the first part, and 49 do it for the second part. The rest, 94 studies, do not mention any limitation.

In a more profound review, we found 82 publications where a manual inspection had been done to assess these limitations: 33 of them referred to issues related to the first part of the SZZ algorithm, while 30 analyzed aspects from the second part (i.e., the bug introducing changes). In the remaining 19 papers, the manual validation of results did not focus on outputs of the SZZ algorithm.

RQ3 (a): Almost half (49.7%) of the analyzed publications mention limitations in the first or second part of SZZ as a threat to validity. Limitations to the first part are reported more often than to the second part.

5.4. RQ4: Are the improvements to SZZ (SZZ-1 and SZZ-2) used?

As discussed in Section 2, the most used improvements to the SZZ algorithm are those proposed by Kim *et al.* [SLR[16]] and Williams and Spacco [SLR[11]].

Our analysis tries to find out how often they have been used.

It is difficult to determine which improvement has been used when the authors do not mention it in the publication. Thus, if the authors do not explicitly specify having used an improvement, we assume that they use the *original* version of SZZ. We have classified the publications into one of the next groups, depending of the kind of improvement they use, as follows:

- *original SZZ*: Those only citing the original version and not mentioning improvements.
- *SZZ-1*: Those citing the improved version of Kim *et al.* [SLR[16]].
- *SZZ-2*: Those citing the improved version of Williams and Spacco [SLR[11]].
- *SZZ-mod*: Those citing the original SZZ with some (own) modification (by the authors). Publications in this group contain statements like "we adapt SZZ", "the approach is similar to SZZ' or "the approach is based on SZZ", but do not refer explicitly to SZZ-1 or SZZ-2.

Table 13 shows how many publications have used improvements to SZZ to mitigate the limitations of the original SZZ. The largest groups correspond to publications where authors use their own enhancements/adaptations (40%) and the original SZZ algorithm (38%). This suggests that researchers prefer to address the limitations of SZZ themselves instead of using enhancements proposed by others. Notice that the "Mixed" column in Table 13 refers to those papers that have used either the original version, the improved versions or some adaptations of SZZ in the same study (e.g., to compare their performance in the same case study).

RQ4: 38% of the publications use the *original* SZZ, whereas 40% use a version *modified* ad-hoc by the authors. Only 14% of the publications used one of the two (improved) versions of SZZ. Even if SZZ-1 or SZZ-2 are used, some publications still refer to SZZ, making it difficult to follow (and reproduce the study).

Table 13: (RQ4) Number of papers that have used the original SZZ, the improved versions of SZZ or some adaptations to mitigate the threat.

	Original SZZ only	SZZ-improved only	SZZ-mod only	Mixed
# publications	71 (38%)	26 (14%) [a]	75 (40%)	15 (8%)

[a] 22 (12%) of the papers use SZZ-1 and only 4 (2%) of the papers use SZZ-2.

5.5. RQ5: Does the reproducibility of the studies improve when authors (1) report limitations or (2) use improved versions of SZZ?

Finally, we want to see if we can find a relationship between some of the characteristics studied so far. In particular, between i) reproducibility and reporting of limitations, ii) reproducibility and the use of an improved version of SZZ, and iii) the SZZ version and reporting of limitations of SZZ. It should be noted that we measure the association between these variables, i.e., causation cannot be inferred from our results.

Hypothesis formulation

- *Hypothesis*$_0^1$*:* There is no association between the *reproducibility* of the papers and *reporting limitations* of SZZ.
- *Hypothesis*$_0^2$*:* There is no association between the *reproducibility* of the papers and the *SZZ version used*.
- *Hypothesis*$_0^3$*:* There is no association between the *SZZ version used* and *reporting of limitations* of SZZ.

For all the above hypotheses, the alternative hypothesis is their negation.

Variables

Reproducibility is a categorical variable that can take three values:

1. *NoReproduction*, for publications that do not provide means to reproduce the paper.
2. *PartialReproduction*, for publications that provide only a detailed methodology and the presence of data or only a replication package.

3. *FullReproduction*, for publications that provide both (replication package and detailed methodology and data).

Version of SZZ used in the studies is a categorical variable that can take three different values:

1. *SZZ-Original*, for publications that use the original version of SZZ.
2. *SZZ-Mod*, for publications that use the original SZZ with some own modification(s)
3. *SZZ-1 & SZZ-2*, for publications that use improved versions.

Reporting limitations of SZZ is a categorical variable that can take two values:

1. *R+* is given to publications that report limitations in either part of SZZ (in following sections: *description of the method*, *threats to validity* or *discussion*).
2. *R-* is given to those that do not report limitations.

Design

The data we have collected can be aggregated and jointly displayed in a tabular form to find out associations and interactions between variables. Tables 14, 15 and 16 show the cross tabulation between variables.

To test hypotheses H_0^1, H_0^2, and H_0^3, we use a Chi-Square test which can be used when there are two categorical variables, each with two or more possible values. If we cannot reject the null hypothesis, we conclude that there is no association between variables. When we can reject the null hypothesis, we can conclude that there is an association between variables. As is customary, the tests will be performed at the 5% significance level. Furthermore, we should consider the multiple testing problem which implies that when multiple hypothesis are tested, the chance of a rare event increases and, as a consequence, the probability of incorrectly rejecting a null hypothesis increases as well [33]. Thus,

Table 14: Cross Tabulation between the categorical variables *Reproducibility* and *Reporting* for hypothesis H_0^1

	Full Repro	Partial Repro	No Repro	Total
R+	10	18	11	39
R-	9	45	40	94
Total	19	63	51	133

Table 15: Cross Tabulation between the categorical variables *Reproducibility* and *version of SZZ* for hypothesis H_0^2

	Full Reprodu	Partial Repro	No Repro	Total
SZZ Original	13	39	19	71
SZZ-Improved	2	10	13	25
SZZ-Mod	7	38	30	75
Total	22	87	62	171

Table 16: Cross Tabulation between the categorical variables *Reporting* and *version of SZZ* for hypothesis H_0^3

	SZZ original	SZZ-Improved	SZZ-Mod	Total
R+	16	10	9	35
R-	32	9	47	88
Total	48	19	56	123

Table 17: (RQ5) P-values for each of the hypothesis. Note that after the Bonferroni correction, the significance level is 0.017 (0.05/3).

	Hypothesis_0^1	Hypothesis_0^2	Hypothesis_0^3
p-value	0.0905	0.1162	**0.0059**

in order to address this problem, we have performed the *Bonferroni correction* for multiple tests. This means that we do not declare any results statistically significant unless their p-values are smaller than the desired significance level divided by the number of tests performed. Therefore, to reject a null hypothesis, the p-value should be smaller than 0.017 (0.05/3 = 0.016666).

Table 17 shows the p-values for each of our hypotheses, indicating that we cannot reject H_0^1 and H_0^2 as their p-values are above the significance level of

0.017. In other words: we have not found association between reproducibility and reporting limitations (H_0^1) and reproducibility and the version of SZZ used (H_0^2). So, contrary to our expectations, papers who follow a *better* approach in one aspect do not necessarily have to follow a *better* approach in the other aspect.

On the other hand, we can reject the null hypothesis H_0^3. We expected to find association between version of SZZ used and reporting limitations. However, Table 16 shows a trend in the opposite direction than the one we assumed, as in particular proportionally few papers that use *ad-hoc* modifications on top of SZZ (*SZZ-mod*) report limitations. This finding can be seen paradoxical at first, since to enhance an algorithm you first have to be aware of its problems; but what we have found is that those who create their own enhanced versions are less prone to report limitations, i.e., they may overestimate the reach of their solution (and, thus, do not report its limitations).

When rejecting the null hypothesis, we also need to understand how these variables are related. This is done by observing the standardized Pearson residuals, which measure the difference between the observed and expected frequencies. When a standardized residual for a category is greater than 2 (in absolute value), we can conclude that it is a major contributor to the significant Chi-square distribution. None of the 21 residual values meets that condition[18].

RQ5: Reporting limitations of SZZ and the use of improved versions of SZZ are not associated with a higher reproducibility of the papers. The use of an improved version of SZZ is associated with reporting limitations, although not in the direction we initially expected; especially authors who perform *ad-hoc* improvements report limitations less frequently.

[18]The tables with all residuals can be found in the replication package.

6. Related Work

In this section, we compare our work with other, related research.

To the best of our knowledge, this work is the first one to make a SLR on the credibility and reproduction of SZZ, by reviewing 187 papers that have cited SZZ or its two major improvements. Some previous literature exists that analyzes the SZZ algorithm and its shortcomings: Davies *et al.* explain some of the limitations that the text-based approach has in order to find the origin of a bug and present enhancements to this approach to solve some of these limitations. Furthermore, they compare the text-based approach with the dependence-based in order to understand which performs better [SLR[13]]. Da Costa *et al.* have carried out a survey of the implementations, usage and evaluation of the SZZ algorithm [SLR[8]]. They briefly detail previous works using SZZ to study how bugs are introduced or to understand just-in-time quality assurance. Therefore, they survey the use of SZZ and the improvements done in a non-systematic way (the number of papers reviewed is not even mentioned), and offer a framework that provides a systematic mean for evaluating the results generated by a given SZZ implementation. In comparison to this work, our work is systematic, extensively larger and more profound; our work is focused on the reproducibility and credibility of the SZZ algorithm, instead of in the evaluation of several approaches of SZZ.

Other systematic literature reviews, surveys or mapping studies published in the field of ESE exist related to SZZ, but their focus differs from the one presented in this paper, and relate to the applicability and use of SZZ for different purposes, such as:

Mining Software Repositories (MSR). Kagdi *et al.* perform a survey on different approaches for mining software repositories in the context of software evolution. They present a taxonomy in four dimensions, understanding the type of software repositories mined (what), the purpose (why), the adopted/invented methodology used (how), and the evaluation method (quality) [SLR[34]]. Woosiung *et al.* carried out a survey on MSR as well, where they describe the data sources,

data types and techniques used, in addition to an evaluation of approaches, opportunities and challenges [SLR[35]]. Amann *et al.* review studies published in the top venues for Software Engineering research (such as ICSE, ESEC/FSE and MSR) with the purpose of describing the current state of the art, mined artifacts, pursued goals and the reproducibility of the studies. Their findings show, in line with then ones presented in this paper, that only 40% of the studies are potentially replicable [SLR[36]].

Bug Reports. Zhang *et al.* review the work on bug report analysis and present an exhaustive survey. They give some guidance to cope with possible problems and point out the necessity to work on bug report analysis because none of the existing automatic approaches has achieved a satisfactory accuracy [SLR[37]]. Other authors, such as Bachmann and Bernstein, have performed a survey on five open source software projects and one closed source software project in order to understand the quality and characteristics of the data gathered from issue tracking databases. This study shows that some projects present *bad* bug report links to commits, and that the process of bug fixing could be designed more efficiently [SLR[38]].

Defect Prediction. Jureczko and Madeyski carried out a survey on process metrics in defect prediction [SLR[39]]. The authors discuss some of those metrics such as number of revisions, number of distinct committers, number of modified lines, among others, and also present a taxonomy of the analyzed process metrics, showing that process metrics can be an effective addition to software defect prediction models. Nam has performed a survey on software defect prediction, but he focuses on the discussion of various approaches, applications and other emerging topics, concluding with the identification of some challenging issues in defect prediction [SLR[40]]. Hall *et al.* performed a SLR to understand how some variables such as the context of models, the independent variables used and the modeling techniques applied, influence the fault prediction models. Their results indicate that models based on simple modeling techniques such as Naïve Bayes or Logistic Regression perform well [SLR[41]].

7. Discussion

In this paper we have studied the use of SZZ, a widely used algorithm in ESE research. We have shown that SZZ is certainly relevant, not limited to a niche audience, and can be found in publications in top journals and prominent conferences. In this regard, we can see its study as a case study of how a software engineering practice spreads across academia.

We have observed that limitations to SZZ are well known and documented. Improvements have been proposed, with unequal success up to the moment. While the limitations to its first part –related to linking fix commits and bug tracking issues– has improved significantly, the enhancements for the second part –that has to do with finding the bug introducing change– are still limited and accuracy has room for improvement.

Even if limitations have been widely documented, from our study we can see that this has not made ESE practices stronger. From the detailed study of the threats to validity of publications using SZZ, SZZ-1 and SZZ-2, we have seen that most publications are not reporting the limitations, and interestingly enough, limitations to the first part –which have shown to be less relevant– are discussed more often than those to the second part. The fact that 38% of the publications use the *original* SZZ is indicative in this regard.

We have found that reproducibility of the publications is limited, and replication packages are seldom offered. The results presented in our research are in line with previous research [SLR[36]], although not as *bad* as the ones found for MSR in 2010 [3]. In any case, we think they are not satisfactory enough, and should raise some reflections about the scientific methods used in our discipline.

Even if using one of the improved versions of the algorithm helps in the accuracy of the SZZ approach as pointed out in [SLR[42]]: "Accuracy in identifying bug introducing changes may be increased by using advanced algorithms (Kim et al. 2006, 2008)", they are seldom used – only 14% of the publications use one of the two (improved) revisions of SZZ. It seems that researchers prefer to *reinvent the wheel*, 40% use an ad-hoc *modified* version of SZZ in their

publications, than to use others' improvements. One possible reason for this is that papers that describe the SZZ algorithm or any of its improvements do not provide a software implementation. Thus, researchers have to implement it from scratch for their investigation. Another possible reason can be because it exits a lack of awareness about SZZ-1 and SZZ-2. Our results show that in such a situation what researchers do is to take the base SZZ algorithm and then add some modifications, resulting in an *ad-hoc* solution. For all *ad-hoc* solutions identified, we have not found a rationale of why other enhancements to SZZ have not been implemented. Another major problem when using improvements to SZZ is that they have not been given a version/label. Even if a revision of SZZ is used, publications often refer to it as SZZ, making it difficult to follow, to reproduce, to replicate and to raise awareness on this issue.

We have observed that ease of reproducibility are rarely found in the studied publications; we could classify only 15% of the papers as being of *good or excellent quality with respect to reproducibility*. The research community should direct more attention to these aspects; we believe that too much attention is put on the final result(s) (the *product* of the research: new knowledge) in comparison to the research process. As researchers in the field of software engineering, we know that both –a high quality product and a high quality process- are essential for a successful advancement in the long term [43].

All these factors undermine the credibility of ESE research, and require a profound consideration by the research community.

7.1. Lesson learned

We consider it important to highlight some of the lessons learned after performing this SLR. It turns out that, when empirical studies are based on heuristics or assumptions, authors must be conscious that to produce reproducible studies the best method is to include a replication package that can be publicly available together with its publication (ideally, for ever). Researchers have however to be aware that some issues, such as the software environments, might change, causing that both programs and data become obsolete, so a detailed

description of the elements, methods and software used during the study is also valuable.

On the other hand, to provide more trustable results, we recommend that researchers specify (and argue) the use of those methods/algorithms that mitigate the limitations of their studies, be aware of the risk of every assumptions that is being used and, if needed, provide a manual analysis of the results. For those studies where the study size is large, researchers can select a random sample to validate it manually.

As it is the case in software projects with release numbering [44], it would be desirable to have a similar mechanism for software implementations used in such type of research, although this may not always be possible given the decentralized nature of research. We, therefore, recommend researchers who develop modifications to the SZZ algorithm to publish the software implementation in development sites such as `GitHub`, so other researchers can *fork* the project. These *forks* could be easily traced, and the authors could ask for a specific citation to their solution if other researchers make use of it.

Finally, we offer a simple way to measure the ease of reproducibility and credibility of research papers. Even if this measure has been conceived with those studies that make use of SZZ in mind, we think it can be adapted to other ESE research easily. Thus, authors can easily assess if their paper offers a reproducible and trustable work (i.e., with scores above or equal to 5).

We should not forget the responsibility of reviewers in the scientific process. We have seen that authors may often be short-term focused when presenting their research. We have found that reproducibility is not associated with reporting limitations or the use of improved versions of SZZ. Reviewers should have the required vision to evaluate the studies having a long-term perspective that is beneficial for the scientific community, helping authors to raise the level of their research. Thus, we recommend reviewers to ask themselves the questions proposed in Section 4.3, adapted to the context of the research, when reviewing publications that are based on heuristics and assumptions.

8. Threats to validity

Wohlin *et al.* discuss four main types of validity threats in ESE research: conclusion, internal, construct and external [45].

Conclusion validity, being related to how sure we can be that the treatment we used in an investigation is related to the actual outcome we observed, does not affect our approach.

Internal validity is the extent to which a causal conclusion based on a study is warranted, which is determined by the degree to which a study minimizes systematic errors. We have attempted to minimize this threat by following the procedures for performing SLRs described in [46], and offer a replication package so that third parties can inspect our *sources*. However, there might be a selection bias due to having chosen Google Scholar and Semantic Scholar as the source of all publications on SZZ; other publications may exist that are not indexed by Google Scholar or Semantic Scholar. In addition, other publications could make use of SZZ and not cite the original publication or its improvements. Another factor affecting internal validity may be the maturity of the field, in the sense that our study may have been done in a too early a stage to draw conclusions. We think, however, that more than 10 years is enough time to allow to extract valid lessons from its study.

Construct validity is the degree to which an investigation measures what it claims to be measuring. In this paper, we measure *impact* by number of publications and types and diversity of venues, *reproducibility* by the availability of a replication package or of a detailed description and data set, and have manually analyzed hundreds of papers for specific, sometimes very detailed aspects, so human errors may have occurred. We think that the effect on *impact* is small, given the large number of publications and venues that we have found. The effect on *reproducibility* is to be considered, as we have not reproduced or replicated the studies by ourselves; in this regard, we offer an upper limit of publications that may be reproducible. Our replication package does not remove the human errors that we may have incurred, but it offers the possibility

to others to check and improve our work.

External validity is the degree to which results can be generalized to other contexts. In this paper, we have selected a single case study, with its particularities and peculiarities. We cannot claim that our results can be generalized to ESE research. However, the value of case studies should be not undermined; Flyvbjerg provides several examples of individual cases that contributed to discoveries in physics, economics, and social sciences [47], while Beveridge observed for social sciences: "More discoveries have arisen from intense observation than from statistics applied to large groups" (as quoted in [48], page 95).

9. Conclusions and further research

We have performed a case study of ESE practice by means of conducting a SLR on the use of the well-known SZZ algorithm. Our sample of publications has consisted of 187 publications that make use of the complete SZZ algorithm (first and second part), out of 458 publications that cite the seminal paper where SZZ was presented or one of the two papers that propose an enhanced version of SZZ.

Our study sheds some light on some of the problems that ESE research faces when assumptions, as it is the case in SZZ, are made in research methods: publications are primarily non-reproducible, limitations are just occasionally reported, improved versions are seldom used and difficult to identify, researchers prefer to use their own enhancements, and awareness and use of improved versions do not suppose an enhancement in reproducibility. All in all, we have observed that the result is a situation with a lot of room for improvement as excellence in reproducibility is seldom found, and where factors that undermine the credibility of the results are common. We finish by offering some advice in how this situation could be improved in the future, sharing our reflections and stressing the importance of reviewers.

Future research should address the limitations of this study and extend it to other domains. There is a need for boosting reproducibility and to investigate

how it should be addressed by the ESE community. Additional case studies should be carried out on other empirical techniques to ascertain if our findings hold in other scenarios and contexts, for instance, when confidential data are involved.

10. Replication package

The replication package can be found in https://gemarodri.github.io/replication/

11. Acknowledgments

We want to express our gratitude to the Spanish Government, as the authors are funded in part by it, through project TIN2014-59400-R. We would also thank Alexander Serebrenik and Marcos Román-González for their feedback and help.

12. References

Following the guidelines described in [SLR[41]], we have cited those publications using the [SLR nn] format if they were one of the (458) full papers considered for review in the SLR; otherwise they have been cited using the normal format [#ref]. The complete list of publications of the SLR can be found in the replication package. There, each reference is followed by a code indicating the status of the paper, whether it passed (P) or failed (F) our criteria to be included in the SLR. In case the paper failed our assessment, we also indicate in which phase (1, 2).

[1] J. M. González-Barahona, G. Robles, On the reproducibility of empirical software engineering studies based on data retrieved from development repositories, Empirical Software Engineering 17 (1-2) (2012) 75–89.

[2] N. Juristo, S. Vegas, Using differences among replications of software engineering experiments to gain knowledge, in: Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement, IEEE Computer Society, 2009, pp. 356–366.

[3] G. Robles, Replicating MSR: A study of the potential replicability of papers published in the Mining Software Repositories proceedings, in: Mining Software Repositories (MSR), 2010 7th IEEE Working Conference on, IEEE, 2010, pp. 171–180.

[4] L. Madeyski, B. Kitchenham, Would wider adoption of reproducible research be beneficial for empirical software engineering research?, Journal of Intelligent & Fuzzy Systems 32 (2) (2017) 1509–1521.

[5] V. R. Basili, F. Shull, F. Lanubile, Building knowledge through families of experiments, IEEE Transactions on Software Engineering 25 (4) (1999) 456–473.

[6] D. E. Perry, A. A. Porter, L. G. Votta, Empirical studies of software engineering: a roadmap, in: Proceedings of the conference on The future of Software engineering, ACM, 2000, pp. 345–355, (F,1).

[7] J. Śliwerski, T. Zimmermann, A. Zeller, When do changes induce fixes?, in: ACM sigsoft software engineering notes, Vol. 30, ACM, 2005, pp. 1–5.

[8] D. A. da Costa, S. McIntosh, W. Shang, U. Kulesza, R. Coelho, A. Hassan, A framework for evaluating the results of the szz approach for identifying bug-introducing changes, IEEE Transactions on Software Engineering(P).

[9] K. Herzig, S. Just, A. Zeller, It's not a bug, it's a feature: how misclassification impacts bug prediction, in: Proceedings of the 2013 International Conference on Software Engineering, IEEE Press, 2013, pp. 392–401, (P).

[10] G. Rodríguez-Pérez, J. M. Gonzalez-Barahona, G. Robles, D. Dalipaj, N. Sekitoleko, Bugtracking: A tool to assist in the identification of bug reports, in: IFIP International Conference on Open Source Systems, Springer, 2016, pp. 192–198, (F,1).

[11] C. Williams, J. Spacco, Szz revisited: verifying when changes induce fixes, in: Proceedings of the 2008 workshop on Defects in large software systems, ACM, 2008, pp. 32–36, (P).

[12] D. M. German, A. E. Hassan, G. Robles, Change impact graphs: Determining the impact of prior codechanges, Information and Software Technology 51 (10) (2009) 1394–1408.

[13] S. Davies, M. Roper, M. Wood, Comparing text-based and dependence-based approaches for determining the origins of bugs, Journal of Software: Evolution and Process 26 (1) (2014) 107–139, (P).

[14] C. Bird, A. Bachmann, E. Aune, J. Duffy, A. Bernstein, V. Filkov, P. Devanbu, Fair and balanced?: bias in bug-fix datasets, in: Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering, ACM, 2009, pp. 121–130, (F,1).

[15] T. F. Bissyande, F. Thung, S. Wang, D. Lo, L. Jiang, L. Reveillere, Empirical evaluation of bug linking, in: Software Maintenance and Reengineering (CSMR), 2013 17th European Conference on, IEEE, 2013, pp. 89–98, (F,1).

[16] S. Kim, T. Zimmermann, K. Pan, E. James Jr, et al., Automatic identification of bug-introducing changes, in: Automated Software Engineering, 2006. ASE'06. 21st IEEE/ACM International Conference on, IEEE, 2006, pp. 81–90, (P).

[17] G. Gousios, The ghtorent dataset and tool suite, in: Proceedings of the 10th Working Conference on Mining Software Repositories, IEEE Press, 2013, pp. 233–236.

[18] M. Tan, L. Tan, S. Dara, C. Mayeux, Online defect prediction for imbalanced data, in: Proceedings of the 37th International Conference on Software Engineering-Volume 2, IEEE Press, 2015, pp. 99–108, (P).

[19] R. Wu, H. Zhang, S. Kim, S.-C. Cheung, Relink: recovering links between bugs and changes, in: Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering, ACM, 2011, pp. 15–25, (F,1).

[20] A. T. Nguyen, T. T. Nguyen, H. A. Nguyen, T. N. Nguyen, Multi-layered approach for recovering links between bug reports and fixes, in: Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering, ACM, 2012, p. 63, (F,1).

[21] T.-D. B. Le, M. Linares-Vásquez, D. Lo, D. Poshyvanyk, Rclinker: Automated linking of issue reports and commits leveraging rich contextual information, in: Program Comprehension (ICPC), 2015 IEEE 23rd International Conference on, IEEE, 2015, pp. 36–47.

[22] Y. Sun, Q. Wang, Y. Yang, Frlink: Improving the recovery of missing issue-commit links by revisiting file relevance, Information and Software Technology 84 (2017) 33–47.

[23] A. Schröter, T. Zimmermann, R. Premraj, A. Zeller, If your bug database could talk, in: Proceedings of the 5th international symposium on empirical software engineering, Vol. 2, Citeseer, 2006, pp. 18–20, (P).

[24] D. Čubranić, G. C. Murphy, Hipikat: Recommending pertinent software development artifacts, in: Proceedings of the 25th international Conference on Software Engineering, IEEE Computer Society, 2003, pp. 408–418.

[25] T. Zimmermann, R. Premraj, A. Zeller, Predicting defects for eclipse, in: Proceedings of the third international workshop on predictor models in software engineering, IEEE Computer Society, 2007, p. 9, (P).

[26] A. Bachmann, C. Bird, F. Rahman, P. Devanbu, A. Bernstein, The missing links: bugs and bug-fix commits, in: Proceedings of the eighteenth ACM SIGSOFT international symposium on Foundations of software engineering, ACM, 2010, pp. 97–106, (F,1).

[27] S. McIntosh, Y. Kamei, B. Adams, A. E. Hassan, An empirical study of the impact of modern code review practices on software quality, Empirical Software Engineering 21 (5) (2016) 2146–2189.

[28] B. Kitchenham, S. Charters, Guidelines for performing systematic literature reviews in software engineering (2007).

[29] H. A. Piwowar, R. S. Day, D. B. Fridsma, Sharing detailed research data is associated with increased citation rate, PloS one 2 (3) (2007) e308.

[30] D. S. Cruzes, T. Dybå, Research synthesis in software engineering: A tertiary study, Information and Software Technology 53 (5) (2011) 440–455.

[31] M. Clarke, A. Oxman, Cochrane reviewers' handbook, Update Software, 2000.

[32] R. Rosenthal, M. R. DiMatteo, Meta-analysis: Recent developments in quantitative methods for literature reviews, Annual review of psychology 52 (1) (2001) 59–82.

[33] R. C. Mittelhammer, G. G. Judge, D. J. Miller, Econometric Foundations Pack with CD-ROM, Vol. 1, Cambridge University Press, 2000.

[34] H. Kagdi, M. L. Collard, J. I. Maletic, A survey and taxonomy of approaches for mining software repositories in the context of software evolution, Journal of Software: Evolution and Process 19 (2) (2007) 77–131, (F,1).

[35] W. Jung, E. Lee, C. Wu, A survey on mining software repositories, IEICE TRANSACTIONS on Information and Systems 95 (5) (2012) 1384–1406, (F,1).

[36] S. Amann, S. Beyer, K. Kevic, H. Gall, Software mining studies: Goals, approaches, artifacts, and replicability, in: Software Engineering, Springer, 2015, pp. 121–158, (F,1).

[37] J. Zhang, X. Wang, D. Hao, B. Xie, L. Zhang, H. Mei, A survey on bug-report analysis, Science China Information Sciences 58 (2) (2015) 1–24, (F,1).

[38] A. Bachmann, A. Bernstein, Software process data quality and characteristics: a historical view on open and closed source projects, in: Proceedings of the joint international and annual ERCIM workshops on Principles of software evolution (IWPSE) and software evolution (Evol) workshops, ACM, 2009, pp. 119–128, (F,1).

[39] M. Jureczko, L. Madeyski, A review of process metrics in defect prediction studies, Metody Informatyki Stosowanej 5 (2011) 133–145, (F,1).

[40] J. Nam, Survey on software defect prediction, Department of Compter Science and Engineerning, The Hong Kong University of Science and Technology, Tech. Rep(F,1).

[41] T. Hall, S. Beecham, D. Bowes, D. Gray, S. Counsell, A systematic literature review on fault prediction performance in software engineering, IEEE Transactions on Software Engineering 38 (6) (2012) 1276–1304, (F,1).

[42] F. Rahman, C. Bird, P. Devanbu, Clones: What is that smell?, Empirical Software Engineering 17 (4-5) (2012) 503–530, (P).

[43] S. H. Kan, Metrics and models in software quality engineering, Addison-Wesley Longman Publishing Co., Inc., 2002.

[44] A. Israeli, D. G. Feitelson, The linux kernel as a case study in software evolution, Journal of Systems and Software 83 (3) (2010) 485–501.

[45] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, A. Wesslén, Experimentation in software engineering, Springer Science & Business Media, 2012.

[46] B. Kitchenham, Procedures for performing systematic reviews, Keele, UK, Keele University 33 (2004) (2004) 1–26.

[47] B. Flyvbjerg, Five misunderstandings about case-study research, Qualitative inquiry 12 (2) (2006) 219–245.

[48] A. Kuper, The social science encyclopedia, Routledge, 2013.