Information and Software Technology 99 (2018) 148–163

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Reproducibility and replicability of software defect prediction studies

Zaheed Mahmood*,[a], David Bowes[a], Tracy Hall[b], Peter C.R. Lane[a], Jean Petrić[a]

[a] University of Hertfordshire, UK
[b] Brunel University London, UK

ARTICLE INFO

Keywords:
Replication
Reproducibility
Software defect prediction

ABSTRACT

Context: Replications are an important part of scientific disciplines. Replications test the credibility of original studies and can separate true results from those that are unreliable.

Objective: In this paper we investigate the replication of defect prediction studies and identify the characteristics of replicated studies. We further assess how defect prediction replications are performed and the consistency of replication findings.

Method: Our analysis is based on tracking the replication of 208 defect prediction studies identified by a highly cited Systematic Literature Review (SLR) [1]. We identify how often each of these 208 studies has been replicated and determine the type of replication carried out. We identify quality, citation counts, publication venue, impact factor, and data availability from all 208 SLR defect prediction papers to see if any of these factors are associated with the frequency with which they are replicated.

Results: Only 13 (6%) of the 208 studies are replicated. Replication seems related to original papers appearing in the Transactions of Software Engineering (TSE) journal. The number of citations an original paper had was also an indicator of replications. In addition, studies conducted using closed source data seems to have more replications than those based on open source data. Where a paper has been replicated, 11 (38%) out of 29 studies revealed different results to the original study.

Conclusion: Very few defect prediction studies are replicated. The lack of replication means that it remains unclear how reliable defect prediction is. We provide practical steps for improving the state of replication.

1. Introduction

Defect prediction is a very active area of research in software engineering. However the quality of defect prediction modelling is regularly criticised [2,3]. Replications are an important way in which to identify the quality of original studies and to increase the confidence that we can have in results [4,5]. Replications also test the claim that "*most research findings are false*" [6] and that a "*little replication goes a long way*" to separate true research findings from false positives [7]. The more replication studies are performed, the more opportunities there are for defect prediction studies to be improved and the state-of-the-art to mature.

This paper aims to quantify the subsequent replications of 208 defect prediction studies identified by Hall et al. [1]. We use Wohlin's [8] forward snowballing approach to identify papers that cite these original 208 studies. Within these citing papers, we identify replications of the original 208 defect prediction studies. We compare the prediction performance of an original study with its accompanying replication study. We measure performance agreement between studies. Agreements or disagreements show replication success or failure, and also indicate the replicability of studies. We extract the characteristics of original studies which have been replicated. Knowing the characteristics of replicated original studies should help authors of primary studies produce studies more accessible to replication. We also present a landscape of how replications are done in defect prediction. We aim to answer the following four research questions:

RQ1: Are defect prediction studies replicated?
RQ2: How are replications performed in defect prediction?
RQ3: What features of a defect prediction study make it likely to be replicated?
RQ4: Do original and replication studies in defect prediction agree?

We make the following contributions. First, we present a methodology for analysing replications that is based on using an existing SLR. Second, we provide a small baseline set of 39 defect prediction studies

* Corresponding author.
E-mail addresses: z.mahmood4@herts.ac.uk (Z. Mahmood), d.h.bowes@herts.ac.uk (D. Bowes), tracy.hall@brunel.ac.uk (T. Hall), peter.lane@bcs.org.uk (P.C.R. Lane), j.petric@herts.ac.uk (J. Petrić).

https://doi.org/10.1016/j.infsof.2018.02.003
Received 29 May 2017; Received in revised form 6 February 2018; Accepted 7 February 2018
Available online 10 February 2018
0950-5849/

Table 1
The identified replications in this study that are mapped & tagged to the considered categories of the Gómez et al. [5] replication taxonomy.

Replication type	Protocol	Operationalisation	Populations	Experimenters	Replication name & changed (Δ) components
Literal	=	=	=	=	Repetition
Operational	=	=	=	≠	Δ-experimenter (A)
	=	=	≠	=	Δ-populations
	=	=	≠	≠	Δ-populations/-experimenter (B)
	=	≠	=	=	Δ-operationalisation
	=	≠	=	≠	Δ-operationalisation/-experimenters (C)
	=	≠	≠	=	Δ-operationalisation/-populations
	=	≠	≠	≠	Δ-operationalisation/-populations/-experimenters (D)
	≠	=	=	=	Δ-protocol
	≠	=	=	≠	Δ-protocol/-experimenters (E)
	≠	=	≠	=	Δ-protocol/-populations
	≠	=	≠	≠	Δ-protocol/-populations/-experimenters (F)
	≠	≠	=	=	Δ-protocol/-operationalisation
	≠	≠	=	≠	Δ-protocol/-operationalisation/-experimenters (G)
	≠	≠	≠	=	Δ-protocol/-operationalisation/-populations
	≠	≠	≠	≠	Δ-protocol/-operationalisation/-populations/-experimenters (H)
Conceptual	Unknown	Unknown	Unknown	Unknown	(only hypotheses are retained)

(originals with their corresponding studies) for researchers to use in future studies. Third, we identify a set of characteristics of original studies for researchers to incorporate into their work to encourage subsequent replication. Finally, we provide practical recommendations which could increase the number of replications performed.

The paper is structured as follows: Section 2 gives background about replication and related work. Section 3 details the methodology while Section 4 provides results. Threats to validity are given in Section 5 and the implications and recommendations for replication are discussed in Section 6. Finally, Section 7 concludes the study.

2. Background and related work

Defect prediction has *"many researchers continuously proposing novel approaches to predict defects in software systems"* [9]. Ioannidis [6] reports that there is a high risk of false results in rapidly growing fields with many research groups (defect prediction can be described this way). Moonesinghe et al. [7] shows that the probability of a research claim being true is increased by replications. Quantifying replications in defect prediction is therefore important.

The number of replications in software engineering has previously been investigated by da Silva et al. [4] who found 96 software engineering papers replicating 72 original software quality and testing studies between 1994 and 2010. A total of 70% of the replications were conducted after 2004, and 70% of those were self replications. Even though replication growth is evident, it does not keep pace with the growth of empirical primary studies; therefore, more external replications are needed [4]. We set out to quantify external replications in one strand of software engineering, i.e. defect prediction. We based our analysis on tracking the replications of a representative sample of defect prediction studies from Hall et al. [1] because the study is one of the *"very prominent 'gold sets' of published SLRs"* and the authors *"define their work in enough detail for us to construct data sets for simulations"* [10].

The terms replication and reproducibility are often interchanged, but they carry different meaning. Replication means to repeat an experiment by independent researchers within a different environment, with changes to the original study aimed at getting consistent results. Reproduction is to recompile the same artefacts used for a study, including data, analysis and procedures for validation [11,12] to get the same results.

Bias may be a major threat to repeatability. Shepperd et al. [13] found that bias introduced by researchers accounts for most of the variance in defect prediction-model performance. So,"*it matters more who does the work than what was done*" and "*Clearly Research Group is a basket for a number of concepts including prior knowledge, statistical and data processing skills, interests, opportunities, relationships with practitioners and so forth*" [13]. These bias factors suggest that a study done by a research group may not be repeatable by others. Previous work has looked at the reproducibility of data mining studies [14,15]. Defect prediction studies invariably are based on data mining. González-Barahona and Robles [14] propose a process model to gauge the reproducibility of data mining studies by identifying key elements of the research including: data source, retrieval methodology, raw dataset, extraction methodology, study parameters, processed dataset, analysis methodology, and results dataset.

Goodman et al. [16] suggest that in any scientific field the kind of replication must be clearly specified. We adopt part of the Gómez et al. [5] replication taxonomy that tracks changes made to components of an original study, and identifies the different types of replications that can be performed. The taxonomy was originally defined for software engineering human-centric experiments, but we adapt it to defect prediction experiments (Section 3 presents our adaption).

According to Gómez et al.'s [5] taxonomy, replication in software engineering can be categorised into three broad types (see Table 1). *Literal* is a type of replication done by authors of the original study. In effect, this type of replication is often named Repetition because no component of the original study is changed; the same experiment is run by the same authors using the exact tools on the same data to avoid bias in the results. Modifying any component of the original study changes the type of replication to *Operational*. For example, if different authors replicate an original study while data and tools remain the same, it is the *Operational* replication type with Changed-experimenter (in effect the same as reproduction). Under the Operational replication, 15 changes can be made to the original study, and each change is given the appropriate name to reflect the change (Table 1 identifies these changes) for example the populations being studied may change. The third replication type is called *Conceptual* because every aspect of the original study is changed except the hypotheses. Applying this taxonomy to new and existing replications is crucial in aggregating replication types and results, to consolidate and synthesise new knowledge.

We aim to identify the number of replicated original defect prediction studies, and identify characteristics of these studies likely to relate to a paper being replicated. The characteristics of the original study we focus on are: study quality, publication venue, citation count and dataset. We focus on quality because Aksnes [17] deems quality as the core knowledge that leads to further developments by other researchers, with lasting significance. We focus on publication venue and study influence as Garousi and Fernandes [18] report that highly cited papers make studies influential. Aksnes [17] also reports that such influential papers tend to be published in journals. We focus on dataset as the availability and usability of data is likely to influence replication

Table 2
Summarised $quality_{4p}$ criteria defined by Hall and Bowes [2][a,b] from [1].

$Quality_{4p}$ assessment phases	Details of phases
Phase 1: Establishing that the study is a prediction study.	-Is a prediction model reported? -Is the prediction model tested on unseen data?
Phase 2: Ensuring sufficient contextual information is reported.	-Is the source of data reported? -Is the maturity of data reported? -Is the size of data reported? -Is the application domain of data reported? -Is the programming language of the data reported?
Phase 3: Establishing that sufficient model building information is reported	-Are the independent and dependent variables clearly reported? -Is the granularity of the dependent variables reported? -Are the modelling techniques used reported?
Phase 4: Checking the model building data	-Is the fault data acquisition process described? -Is the independent variables data acquisition process described? -Is the faulty and non-faulty balance of data reported?

[a] Phase 1 assesses defect prediction methodological approaches.
[b] Phases 2, 3 and 4 assess reporting of prediction studies.

potential. Only the quality characteristics are not directly measurable. We use the $quality_{4p}$ assessment process to characterise the quality of original studies as used by Hall et al. [1]. The $quality_{4p}$ process assesses defect prediction studies in terms of whether they employ a reliable *methodological* approach to building prediction models and whether studies *report* sufficient information to comprehend a study [1] (Table 2 summarises the quality criteria). A more detailed description of $quality_{4p}$ is outlined in Hall and Bowes [2].

3. Methodology

Our methodology has six stages with each stage further broken down.

3.1. Stage 1: Identification of replication papers

We use as our base set of studies the 208 original studies published in the 2012 SLR in defect prediction [1]. We used forward snowballing [8] to identify papers that subsequently cite and replicate the 208 original studies between 2000–2017 (15th April). This means that we sift through papers that cite original studies, identifying all possible papers that may replicate an original SLR study.

We used Google Scholar to identify citing papers for each of the 208 original studies. On the *'cited by #papers'* page of each paper we used the ~Replicate OR ~Replication OR ~Replicated string and selected the *'search within citing articles'* feature. In effect, only papers that used these terms or their synonyms (denoted by tilde (~)) were returned. Applying this technique reduces the number of papers to be assessed as replications and reduces false positives. We then read from the returned results page, the paper title and its summarised phrases to identify if the paper was a replication of an original study in the 208. If not sufficient, we accessed the whole document to find the context in which the term was used, as suggested by Wohlin [8]. For this search the in-built search feature of the web browser used or document reader was used to find where the term replication is used. If the replication term is not in the document we read the paper in full to establish if it was a replication. Using this approach we identified a set of papers that replicated a subset of the 208 original studies.

3.2. Stage 2: Inclusion criteria

Our focus is to find external replications (i.e., replications not by the original authors, as these are considered true replications [4,19]) of the original studies. We exclude a paper if the replication is by any original author(s), or was extended work by any of the original author(s). If the author(s) have extended an original piece of work, we considered this work to be one paper, and any replication of either of these two is a replication of an original study. We consider any author (whether a lead author or not) to be an author of the paper. Consequently we track all replications by all authors of original studies.[1] We found 13 original studies that have been replicated by 26 replication papers. These 39 papers are our *final-set*.

3.3. Stage 3: Data extraction process

3.3.1. Tool for extraction

For reproducibility (i.e. the ability of our research to be compiled and produce the same result), we used our SLuRp tool.[2] SLuRp is a web-based tool developed to make Systematic Literature Reviews (SLR) reproducible and also provides effective information storage and retrieval. SLuRp was assessed as the best of the SLR tools by Marshall et al. [21]. We did not use all of SLuRp's functionality, many more useful SLR management features are described in Bowes et al. [20]. We provide the following steps as a summary of SLuRp together with how we used it for data extraction.

1. Import BIBTeX files and store references to all original and replicated studies.
2. Assign two researchers (authors of this paper) to independently store extracted information from each paper.
3. Allow researchers to modify and approve extracted information.
4. Disagreements between researchers are flagged by SLuRp.
5. Create forms based on contextual and methodological information that must be extracted from each paper.
6. Store extracted information in the SLuRp database.
7. Retrieve stored information using SQL queries and organise into result tables.
8. Export tables as LaTeX tables. Graphs and box plots are available.
9. Edit entire paper with SLuRp LaTeX editor, including results, tables and compiled to produce the final paper.

3.3.2. Extraction of selected data from final set

Three sets of data were extracted that allowed us to answer RQ2, RQ3 and RQ4. The first set of extracted data (for RQ2) characterises how defect prediction studies are performed. This dataset is based on the defect prediction characteristics presented in Hall et al. [1] and Hall and Bowes [2]. These characteristics include:

[1] To clarify: If paper P is authored by Anne, Ben and Ceri. And paper Q is authored by any of Anne, Ben or Ceri, paper Q is NOT in the set of replicated papers. If paper R is an extension of paper P and has none of the original authors, R is included.
[2] Bowes et al. [20] available at https://bugcatcher.stca.herts.ac.uk/bugcatchers/faces/slurp/SLuRp.xhtml.

1. dependent variables
2. independent variables
3. algorithms
4. dataset
5. tuning
6. cross validation
7. statistical analysis

This set of defect prediction characteristics data allows us to gain insights on replication practice and to categorise replications based on changes replications make to the original studies in terms of these characteristics. The information we collect allows us to categorise replications into their respective categories (as defined in Table 1). Of the 39 *final-set*of studies, 5 papers were read independently between two authors (by way of a validation check on the data extraction process) and their data extracted, while agreements were reached on this data extraction using SLuRp to minimise threats to validity. Information on the remaining papers was then extracted by one of the authors. The second set of data extracted (for RQ3) allows us to determine which features of defect prediction studies make it likely that an original study will be replicated. This set of data is: study quality, publication venue, citation count and dataset (as presented in Section 2). The extraction of this data is described in Section 3.5. The third set of data extracted (for RQ4) allows us to establish whether the results of a replication study are comparable to the original. Section 3.6 describes the process by which we establish study outcome agreement.

3.4. Stage 4: Categorisation of replications into types

The Gómez et al. [5] replication taxonomy (Table 1) requires understanding of a study and its individual components before being applied to categorise replications into types. We breakdown defect prediction study components for replication classification by adapting the general component structure proposed by Gómez et al. [5] (see Appendix A). These study components represent changeable aspects of an original study during its replication (as described above). Each component changed may assist in the discovery of unknown factors that affect replication results.

The component data extracted from the *final-set*of papers are organised into tables (see Appendix B). We mapped each replication study to its type in Gómez et al. [5] taxonomy based on changes researchers made to the original study components during replication. We detail the four components of a study as follows;

Protocol is the overall study design. In defect prediction the framework that pulls together different sub-components to build a prediction system is the overall study design (protocol). Table B.16 shows the protocol sub-components we have used are:

1. Cross validation scheme used
2. Whether parameter tuning was performed
3. Which statistics were used to compare performance results
4. Whether data cleaning was used

These factors are motivated by Hall et al. [1] and Hall and Bowes [2] as outlined previously in Section 3.3. The protocol is the design before it is implemented (i.e. operationalised).

Operationalisation has two aspects, cause and effect. The cause is the process of implementing the protocol and considers the implementation environment (as shown in Table B.17) we consider the following implementation factors (again motivated by Hall et al. [1] and Hall and Bowes [2]):

1. Tools used
2. Algorithms used
3. Independent variables used

It should be noted that algorithms have been embedded into data mining tools like Weka [22], in effect the tools carry out the treatments required to implement a prediction framework. Therefore such tools and their versions must be considered because they may cause differences in replication results. Effect is the process of determining and defining the aspects of a model to be measured and selecting the appropriate measure. Since measures already exist (e.g. recall; measures the proportion of actual defects a model correctly predicted), it is a question of which appropriately measures the effect of the treatments in the model's prediction outcome. Consequently Table B.17 shows that the final operationalisation factor we collect is the dependent variable.

Population is based on the systems analysed in studies. These systems are then mined from source code repositories (open or closed sources). Changing a repository to mine data also changes the population. Table B.18 shows that the population factors that are considered are:

1. Data source
2. Domain
3. Language
4. Granularity of defect data

The granularity, i.e. method or class level, where the defective or non-defective data are gathered is also part of this. The programming languages used, size of project (KLOC), maturity (years of use and development), etc. Changing any of these sub-components affects the population and likely the replication results.

Experimenters are the researchers that conducted the study.

3.5. Stage 5: Identification of factors associated with replication

For all 208 papers, as discussed previously, we extracted 6 factors to find out if any of the factors have a relationship with the number of subsequent replications:

- $quality_{4p}$
- Number of citations of a paper
- Publication venue
- Publication venue's impact factor
- Data sharing/availability

We extracted $quality_{4p}$ assessment outcomes using Hall et al.'s quality check for defect prediction studies [1] for the 208 original studies that have been replicated (see Table 2 for a summary of $quality_{4p}$).

$Quality_{4p}$ overlaps extensively with González-Barahona and Robles' [14] reproducibility criteria which includes checking the: data source, retrieval method, raw data, extraction method, study parameters, analysis method, results method, identification and description. Two elements of Barahona and Robles' [14] reproducibility criteria are missing in $quality_{4p}$ and these are data availability and data flexibility. We additionally collect availability data (i.e. an element's tendency to exist in the future). We explicitly checked all the links of each study to confirm if data are accessible (in September 2017). We additionally collect Barahona and Robles' [14] flexibility criteria, i.e. adaptability to different environments by extracting the formats of shared data in terms of e.g. csv, arff etc. For open or closed source code repositories, metrics (e.g. object oriented metrics calculated on defective/non-defective code) can be collected to form defect data used as input for building prediction models [Org1, 2, 3]. For example the NASA MDP program provided defect datasets calculated from the raw source code of critical systems (e.g. Flight and Satellite systems). The raw source code, being proprietary, were not available. However, it is possible to reproduce a study based on the defect data which was shared even though if it was generated from a closed source.

We extracted impact factor values for their publication venue from

Table 3
ERA ranking categories.

Rankings	Description
A*	Flagship conference, a leading venue in a discipline area
A	Excellent conference, and highly respected in a discipline area
B	Good conference, and well regarded in a discipline area
C	Other ranked conference venues that meet minimum standards
Unranked	A conference for which no ranking decision has been made

http://www.core.edu.au/conference-portal.

journalmetrics (details are in Table 9). We used the Source Normalised Impact Average (SNIPA) [23] values which are based on the average citation per paper of a journal in that subject area. In addition, we extracted the ratings of journal/conference venues from Excellence in Research for Australia (ERA). In 2009, the Australian Research Council consulted the public, expert reviewers and academic bodies to rank journals and conferences, and produced the ERA rankings. We used the ERA 2010 rankings since other ranking bodies only provide journal impact factors and omit any ranking of conferences. ERA has 5 ranks according to research quality, see Table 3.

3.6. Stage 6: Assessing agreements between studies

We checked whether the performance reported in the original studies matched those reported in the replications. If replications agree then original studies are replicable. We also assessed reproducibility (getting the same results) since replications tend to vary because of contextual differences. By comparing predictive performance measures for both original and replication studies in the same context (i.e. same data, classifiers, metrics etc.). If the performance is different by $< 1\%$ we assess this as having being *reproduced*. If the change is $< 5\%$, it is *similar* and if it is $> 5\%$, we classify this as *different*. We chose these values based on the intervals used in statistical testing, i.e. 1% probability and 5% probability using standard statistical tests.

4. Results

4.1. RQ1: Are defect prediction studies replicated?

> Only 6% of 208 original studies were replicated, suggesting replication and reproducibility are largely neglected in defect prediction studies.

Only 13 out of the set of 208 original studies were replicated by different researchers reported in 26 papers (Table 4): 6% of the original studies, a significantly lower rate than the 94% non-replicated original studies. Which means that the lack of replication is substantial, consequently, there is a significant number of studies that have not been confirmed to report valid results via replication.

4.2. RQ2: How are replications performed in defect prediction?

> Replication studies make many changes to original studies.

Overall Table 4 shows that all replication studies made changes to the original study. Typically replications made three sets of changes to components of original studies.

Two replication studies (Hamill and Goseva-Popstojanova (Rep[1,2]), Hongyu Zhang (Rep[3,4])) replicated more than one original study, these two papers appear twice making the number of replication studies 26; these papers then appear twice in the 'Replication studies' column of Table 4 as Hamill and Goseva-Popstojanova (Rep[1,2]), and Hongyu Zhang (Rep[3,4]).

Three original studies (Andersson and Runeson (Org[4]) (Rep[5]), Lessmann et al. (Org[5]) (Rep[6]), Ostrand et al. (Org[6]) (Rep[7])) also conducted replications of other original studies and within them, certain aspects of their study have also been replicated. For example Lessmann et al. (Rep[6]) replicated Menzies et al. (Org[7]) and built a new classifier benchmarking framework. Ghotra et al. (Rep[8]) subsequently replicated the new framework. Therefore Lessmann et al.would appear twice in the first two columns of Table 4 with (Org[5]) (Rep[6]).

Table 4 shows that many changes are made to studies:

Table 4
13 replicated original studies out of the 208 with their replication studies, data sets, replication types and agreements between studies.

Replicated original studies	Replication studies	Agreements	Operation-al rep.0[a]
D'Ambros et al. (Org[1])	Mende (Rep[9])	Yes, Yes	(A), (G)
Andersson and Runeson (Org[4])	Hamill and Goseva-Popstojanova (Rep[1])	Yes	(H)
	Zhang (Rep[3])	No	(H)
Lessmann et al. (Org[5])	Ghotra et al. (Rep[8])	Yes, No	(A), (H)
Ostrand et al. (Org[6])	Marek (Rep[10])	Partial	(H)
	Mende and Koschke (Rep[11])	Unknown	(G)
Fenton and Ohlsson (Org[8])	Andersson and Runeson (Rep[5])	Partial	(H)
	Grbac et al. (Rep[12])	Partial	(H)
	Ostrand et al. (Rep[7])	Yes	(H)
	Zhang (Rep[4])	No	(H)
	Devine et al. (Rep[13])	Yes	(H)
	Hamill and Goseva-Popstojanova (Rep[2])	No	(H)
Menzies et al. (Org[7])	Turhan and Bener (Rep[14])	Yes	(G)
	Zhang et al. (Rep[15])	Yes	(G)
	Lessmann et al. (Rep[6])	Yes	(G)
	Song et al. Rep[16]	Yes, No	(A),(H)
	Singh and Verma (Rep[17])	Yes	(H)
Moser et al. (Org[9])	Krishnan et al. (Rep[18])	Yes	(H)
Kim et al. (Org[10])	Rahman et al. (Rep[19])	Yes	(H)
Zimmermann and Nagappan (Org[2])	Tosun et al. (Rep[20])	Yes	(H)
	Nguyen et al. (Rep[21])	Yes	(H)
	Premraj and Herzig (Rep[22])	Yes	(H)
Amasaki et al. (Org[11])	Okutan and Yıldız (Rep[23])	Unknown	(H)
Schröter et al. (Org[12])	Duala-Ekoko and Robillard (Rep[24])	Yes	(H)
Zimmermann et al. (Org[3])	Kpodjedo et al. (Rep[25])	Unknown	(H)
Khoshgoftaar and Seliya (Org[13])	Li et al. (Rep[26])	Yes	(H)

[a] Replication name tags: (A) changed-experimenters, (G) changed-protocol/-operationalisation/-experimenters, (H) changed-protocol/-operationalisation/-populations/-experimenters.

Table 5
Study-components of original studies that were changed during replication.

Protocol	Stats	CrossVal	DataClean	Parameter tuning
	19	8	5	2
Operationalisation	IndepVar	DepVar	Algorithm	Tools
	15	4	12	14
Populations	Granularity	Domain	SourceCode	ProgLang
	14	12	11	7

Full field names: statistical analysis, cross validation, data cleaning, optimising parameters, independent and dependent variables, programming language.

1. Changed-experimenters (tag A, 3 papers)
2. Changed-protocol/-operationalisation/-experimenter (tag G, 5 papers)
3. Changed-protocol/-operationalisation/-populations/-experimenters (tag H, 21 papers)

Replications in which most components are changed together dominates. Table 4 shows that Mende, Song et al., Ghotra et al.replicate with sets of two study-component changes (A,G and A,H). With changed-experimenter (A) as the first change and (H) as the last, changes (B,..., F) have been omitted for all replications indicating gaps in steps that need to be taken during replications.

The data we synthesised from all studies (in Tables 11–13 for original studies, and in the appendix Tables B.16–B.18 for replication studies) depicts a landscape of some of the tools, algorithms, and statistical analyses used in defect prediction. Table 5 shows that the statistical test component has the most changes compared to parameter tuning with the least changes. Replications tend to focus more on finding the most suitable statistical methods to describe data (e.g. Zhang (Rep[3]) suggests distribution of software faults are better described as a Weibull distribution, not in terms of the Pareto principle as originally proposed by Fenton and Ohlsson (Org[8])). While tuning the parameters of the prediction models to improve performance is considered the least.

There are 3 replication studies (Rep[16]), (Rep[8]), (Rep[9]) that did multiple runs of a single original study. The first run reproduced the original study as it is, and the second run either modified the protocol (Rep[9]), (Org[1]) protocol by adding a cross validation step, Song et al. Rep[16] used feature selection that ensured the test instances are not seen by the prediction model), or dataset (Ghotra et al. (Rep[8]) used less noisy data and a new dataset, Song et al. Rep[16] also added more datasets). These multiple runs have implications for agreements between studies and the types of replications performed, though such multiple runs are generally good practice.

4.3. RQ3: What features of a defect prediction study make it likely to be replicated?

> Our results suggest that studies based on industry closed source data published in the Transactions on Software Engineering journal (highest impact in software engineering (during the time period covered by Hall et al. [1])) leads to a paper being replicated.

We analysed the factors we extracted from each paper statistically.[3] We use a χ^2 test to establish the relationship between each binary factor and replications and Kendall's Tau rank correlation to test the relationship between citations and replications (as citations is continuous data) (see Table 6).

Table 7 shows the data format of the papers with datasets. Table 7 shows that there are few papers using formats other than arff. The small numbers do not allow a sound statistical analysis to be carried out for the affect of flexibility on the ability to be replicated.

Table 6
The 208 papers categorised as having $quality_{4p}$, shared data, appeared in TSE w.r.t being replicated*.

	$Quality_{4p}$		Shared data		InTSE*	
Replicated	Yes	No	Yes	No	Yes	No
Yes	3	10	5	8	5	8
No	33	162	70	125	10	185

* chosen as TSE dominates in Table 8.

Table 7
The formats of the data and the number of papers which use the format and the availability of the data.

Data format (flexibility)	Not replicated	Replicated
arff	60	2
csv	0	1
csv, arff	2	1
csv, xml	1	0
excel	1	0
xml	3	1

Table 8
Statistical tests for assessing $quality_{4p}$, shared data, TSE and citations, individually against replications.

Chi Square test	χ^2	*p*-value		
$Quality_{4p}$* replication	0.322	0.570		
Shared data * replication	0.035	0.852		
InTSE * replication	20.237	< **0.0001**		
Kendall		*z*	τ	*p*-value
Citations * replication		4.7614	0.269	< **0.0001**

There were 85 venues in which the 208 papers appeared (Online-Appendix [4]). Only 6 venues published papers that were subsequently replicated: PROMISE, MSR, ESEM, ISSRE, ICSE and TSE. TSE has the highest number of papers published with subsequent replications (Table 9). Table 8 shows that papers published in TSE are more likely to be replicated. We do not consider the impact factor of venues directly since, for non-replicated studies, impact factors are not available for many (63) publication venues.

Table 8 shows that a paper's influence (citations) has an impact on replication. However the quality of original papers or shared data use is not associated with subsequent replication.

Table 9 shows 10 of the 13 replicated studies have not passed the $quality_{4p}$ assessments. A replication not based on $quality_{4p}$ has ramifications on the validity of findings. For instance, data cleaning of the $quality_{4p}$ may have been overlooked or not reported, an indication that some findings may be erroneous. It is particularly true for the noisy NASA datasets used by 59 original studies (Table OA.1 in Online-Appendix).

Table 10 shows that 21 of 26 replication studies replicated original studies which were based on closed source industrial data (these will have needed to be replicated with different datasets). This suggests that studies based on closed source industrial data may be more attractive for replication.

[3] Using `R 3.3.1` open source statistical software.

[4] https://bugcatcher.herts.ac.uk/replication/Online-Appendix.html.

Table 9
The replicated original studies and whether they extracted contextual factors.

Citations	Original studies	Journal	ERA[a]	Impact[b]	$Quality_{4p}$ assessment[c] failed at phase	#Reps
128	Andersson and Runeson (Org[4])	TSE	A*	4.423	Phase1 No prediction done	2
577	Lessmann et al. (Org[5])	TSE	A*	4.423	Phase2 NASA data used	1
535	Ostrand et al. (Org[6])	TSE	A*	4.423	Phase4 Model building	2
684	Fenton and Ohlsson (Org[8])	TSE	A*	4.423	Phase1 No prediction done	6
816	Menzies et al. (Org[7])	TSE	A*	4.423	Phase2 NASA data used	5
361	Moser et al. (Org[9])	ICSE	A	2.988	Pass all	1
330	Kim et al. (Org[10])	ICSE	A	2.988	Phase4 model building	1
393	Zimmermann and Nagappan (Org[2])	ICSE	A	2.988	Phase4 model building	3
240	D'Ambros et al. (Org[1])	MSR	C	1.876	Pass all	1
43	Amasaki et al. (Org[11])	ISSRE	A	1.383	Phase2 contextual information	1
159	Schröter et al. (Org[12])	ESEM[d]	A	0.992	Pass all	1
126	Khoshgoftaar and Seliya (Org[13])	ESEM[e]	A	0.992	Phase2 contextual information	1
508	Zimmermann et al. (Org[3])	PROMISE	U	0.001	Phase2 contextual information	1

[a] CORE contributed to ERA rankings. Both rankings agree except on ICSE; A by ERA, A* by CORE (TSE not ranked).
[b] Source is journalmetrics, 2015 source normalised impact (SNIPA); takes average citation per paper of a journal in subject area.
[c] QA details summarised in Table 2.
[d] ISESE,
[e] METRICS are now part of ESEM http://www.esem-conferences.org/history.php .

Table 10
Descriptions of replicated original studies based on the type of data source and defect data sharing.

Source code	Shared data	No. of papers	No. of reps
Closed	No	6	15
	Yes	2	6
Open	No	2	2
	Yes	3	3

originals (i.e. where some of the replicated results were the same as the originals but not all). Additionally 3 studies did not report the level of agreement with the original study.

Our results show a variety of disagreements between the original and replicated results. There are a range of reasons for these disagreements that we will now discuss. Song et al. Rep[16] did 2 replication runs of Menzies et al. (Org[7]). In the first run the replication agrees with Menzies et al. (Org[7]). In the second run, Song et al. Rep [16] disagreed and report a flaw in (Org[7])'s attribute selection approach which meant that the test data included seen information and

Table 11
Protocol: Original studies.

Org. Studies	Cross Val.	Parameter tuning	Statistics	Data Cleaning
Andersson and Runeson (Org [4]), (Rep[5])	No	No	Pearson product-moment correlation	Yes: Duplicate failures
Lessmann et al. (Org[5]), (Rep [6])	Hold-out set	Yes	Friedmans test (rank classifiers), Nemenyi post hoc (statistical significance test on classifiers)	No
Ostrand et al. (Org[6]), (Rep[7])	No	No	*t*-test	No
Fenton and Ohlsson (Org[8])	No	No	Alberg diagrams	No
Menzies et al. (Org[7]	10 by 10 randomised	No	Quartile chart	No
Moser et al. (Org[9])	10 by 10 randomised	No	Kruskal–Wallis test	No
Kim et al. (Org[10])	No	No	No	No
Zimmermann and Nagappan (Org[2])	split-sample	No	Spearman correlation, Pearson, Nagelkerke (predictive power of logistic regression models), F-tests	No
Amasaki et al. (Org[11])	No	No	Error rate, Fishers exact test (correlation between 2 variables)	No
Schröter et al. (Org[12])	random splits	No	Two *t*-test, Spearman rank correlation	No
D'Ambros et al. (Org[1])	50 by 10fold randomised	No	*F*-test (explanative significance), Spearman correlation (evaluating predictive power of models) with Spearman coefficient (skewed data)	No
Zimmermann et al. (Org[3])	No	No	Spearman correlation, Pearson correlation	No
Khoshgoftaar and Seliya (Org [13])	10 fold cross validation	No	*Z*-test	No

4.4. RQ4: Do original and replication studies in defect prediction agree?

> It is difficult to confirm agreements in published results as there is inconsistent reporting of the performance measures.

Overall our analysis shows that the performance of 18 replicated experiments.[5] agreed with original performance values. This suggests that 62% of the replicated experiments were successful. The performance of 5 replicated experiments (17%) did not agree with originals and 3 replicated experiments (10%) resulted in partial agreement with therefore inflated performance of the defect prediction models.

Ghotra et al. (Rep[8]) did 2 replication runs of Lessmann et al. (Org [5]). The first run was based on uncleaned NASA data (including duplicate and inconsistent instances, see [24]) to confirm if no single classifier is best as in the original (Org[5]). The Friedman test used in Lessmann et al. (Org[5]) showed the ranking of model performances are not random; subsequently Nemenyi post hoc test was applied to detect which of the classifiers differed significantly. Ghotra et al. (Rep [8]) agree with Lessmann et al. (Org[5]) in the first run with the same data and different statistics, but disagree in the second run with a cleaned dataset curated by Shepperd et al. [25] and different statistics. In the second run, Ghotra et al. (Rep[8]) reported;

[5] Some papers conduct more than one experiment, there are 26 papers running 29 experiments.

Table 12
Operationalisation: Original studies.

Org. Studies	Tools	Algorithms	Independent Var	Dependent Var
Andersson and Runeson (Org[4]), (Rep[5])	No	No	Size (LOC)	Number of faults, fault density pre-release and post-release
Lessmann et al. (Org[5]), (Rep[6])	YALE machine learning	Statistical(7), Nearest Neighbours(2), Neural Networks(3), SVMs(5), Tree-based (3), Ensembles (2)	static code metrics	Defective or Not-defective
Ostrand et al. (Org[6]), (Rep[7])	VCS	Negative binomial regression	code age, programming language, logKloc,file status, release	Number of faults
Fenton and Ohlsson (Org [8])	ERIMET (metrics),FCTOOL (formal description language)	No	Complexity (McCabe cyclomatic complexity), size (LOC), communication (SigFF;new and modified signals count, inter and intra modules)	Number of faults, fault density prerelease and post-release
Menzies et al. (Org[7])	WEKA	OneR, J48, and Naive Bayes	Static code metrics	Defective or Not defective
Moser et al. (Org[9])	WEKA	Logistic regression, Naive Bayes, J48 (version 8)	Process, change, static code	Defective or Not defective
Kim et al. (Org[10])	Kenyon Infrastructure, APFEL (metrics), SVN, CVS	Least recently used (LRU)	spatial locality, temporal locality,Changed-entity and new-entity locality (churn)	Number of faults
Zimmermann and Nagappan (Org[2])	MaX (dependency information tracker), Ucinet 6 (network metrics)	Linear and logistic regression	Network measures on Dependency graphs, OO metrics, static code metrics	Number of defects, Defective or Not defective
Amasaki et al. (Org[11])	Netica (bayesian belief network software)	Bayesian Belief Network	design (product size), effort (person-day), detected faults, test items	Number of residual faults at acceptance test
Schröter et al. (Org[12])	R, BUGZILLA, CVS, SZZ	Linear and Ridge regression, Regression trees, SVM	import relationships (e.g. org.eclipse.ui) packages and imported classes	Number of defects, Defective or Not defective (post-release)
D'Ambros et al. (Org[1])	CVS, SVN, Bugzilla, Jira, Famix-Compliant OO model (scm metrics), Infusion (source code converter into FAMIX model), Moose (scm calculator), Churasco (history model, bug data extractor, classes linker, system files and bugs versioner)	linearReg	process, change, entropy of change, entropy, churn of source code, source code metrics, CK, OO	Number of defects (post-release)
Zimmermann et al. (Org [3])	Java parser (complexity metrics)	Linear regression (ranking), logistic regression (classification)	static code metrics (complexity), structure of abstract syntax tree (no of nodes etc.)	Number of defects, Defective or Not defective (prerelease, post-release)
Khoshgoftaar and Seliya (Org[13])	S-Plus (advanced data analysis), EMERALD - Datrix(fault data collection)	Least squares tree, s-plus, least absolute deviation	Design	Number of faults

Table 13
Populations: Original studies.

Org. Studies	Prog. Lang.	Domain	Granularity	Source	Availability
Andersson and Runeson (Org[4]), (Rep[5])	C, Java	Telecom	Module	Commercial	Not shared
Lessmann et al. (Org[5]), (Rep[6])	C, Java	Satellite, Flight, Storage	Module (predictions), code (metrics)	NASA, PROMISE	Shared
Ostrand et al. (Org[6]), (Rep[7])	Java, C, Makefiles, sql, shell, html, other	Inventory System, Provisioning System	File (predictions)	Industrial	Not Shared
Fenton and Ohlsson (Org[8])	No	Telecom	Module	Ericsson Telecom AB	Not shared
Menzies et al. (Org[7])	C, Java	Satellite, Flight, Storage	Method	NASA, PROMISE	Shared
Moser et al. (Org[9])	Java	IDE	File (predictions)	Eclipse 2.0, 2.1, 3.0	Not shared
Kim et al. (Org[10])	C, C++, Java	Web server, Browser, Text editor, Version control, Database, IDE, Email client	File, method (predictions)	Apache 1.3, JEdit, Subversion, PostgreSQL, Columba, Eclipse, and Mozilla	Not shared
Zimmermann and Nagappan (Org[2])	C++	Operating System	Binaries (predictions), Binaries (metrics)	Windows Server 2003	Not shared
Amasaki et al. (Org[11])	No	Embedded software	development process (metrics), directed graphs	Industrial	Not shared
Schröter et al. (Org[12])	Java	IDE	File, package (predictions), import packages and classes (metrics)	ECLIPSE plug-ins 52nos	Shared
D'Ambros et al. (Org[1])	Java	IDE	Class (predictions, metrics)	Eclipse (Mylyn, Equinox, PDE, Lucene, Score)	Shared
Zimmermann et al. (Org[3])	Java	IDE	Files, packages (predictions, metrics)	Eclipse 2.0, 2.1, 3.0	Shared
Khoshgoftaar and Seliya (Org[13])	Protel	Telecom	Modules (predictions), design documents (metrics)	Industrial	Not shared

Agree: replications that confirm original results, Disagree: replications that do not confirm original results, Partial: replications that confirm part of the original results, Unknown: replication that does not report agreement or disagreement.

> "*We used the Scott–Knott test to overcome the confounding issue of overlapping groups that are produced by several other post hoc tests, such as Nemenyis test [13], which was used by the original study. Nemenyis test produces overlapping groups of classification techniques, implying that there exists no statistically significant difference among the defect prediction models trained using many different classification techniques.*"

The curated data by Shepperd et al. [25] has been cleaned further by Petrić et al. [24]. The data errors found during this further cleaning may have also affected previous models. Overall, these findings suggest that replication leads to the discovery of mistakes and provides the opportunity to remedy those shortcomings.

Overall our results suggest that replications in defect prediction are possible with or without $quality_{4p}$ in the original study. Of the 29 studies, partial agreements (3) and disagreements (5) make up to 28% of the results, indicating that replication is able to detect errors and limitations of studies. Unreported (3) replications results (10%) are relatively high. We suggest that all replications need to state agreements and disagreements.

For a more detailed assessment of agreements, we extracted the performance values of replications with only 'changed - experimenter', as this type of replication is useful for assessing the reproducibility of research. Reproducibility aims to get the same result as the original study [11,12]. We categorise a paper as reproducible if the difference in the performance between an original and its replication does not go beyond 5%. We identified 5 replications of Menzies et al. ((Org[7]) shown in Table 14). Table 14 shows that Turhan and Bener (Rep[14]), and Zhang et al. (Rep[15]) report $>5\%$ different recall performance (64% and 85%), which means that neither study has succeeded in reproducing Menzies et al. (Org[7]) (71% recall). Table 14 also shows that Lessmann et al.'s replication (Rep[6]) used a different measure (auc) to the original measure reported making it difficult to assess reproducibility; similarly Song et al. Rep[16] reported only one performance measure (balance).[6] These results show that reporting inconsistencies between replications and original studies make it difficult to confirm agreements.

We investigate reproducibility further by ourselves reproducing (Org[5]), (Org[7]). Table 14 shows that our results are mixed despite matching closely all study components. In reproducing these original studies a number of anomalies with the original studies arose which may explain the differences in our performance values compared to the original studies. These anomalies include that the datasets we downloaded varied from the original in terms of number of defective units, number of instances etc. and also that our feature selection outcomes were not the same as the originals. Our Online-Appendix provides full details of these anomalies.

5. Threats to validity

The main threat to validity is that replication is currently performed so seldom that it is difficult to draw conclusions from the population of replications that we have. Many more replications need to be performed before it is possible to draw highly reliable conclusions about replication.

Another important threat is the identification of papers that replicate the 208 original studies and the tool used for the search, that is Google Scholar. The main search ended in 2016 and since then we have automatically monitored replicated papers with triggered mail alerts of new citing papers. The search string is saved and is run automatically by Google Scholar with every new citation of the replicated study. Each paper is checked to confirm if it was a replication or not; no new

[6] Mende (Rep[9]) reproduced D'ambros et al. (Org[1]), the results are mostly the same about ($<1\%$), but with a few differences that could not be explained. We do not include Mende (Rep[9]) due to lack of space, but all the results can be found in our replication package https://bugcatcher.herts.ac.uk/replication.

Table 14
Model performance measure from Menzies et al. set of replications and Lessmann et al.

Data	Naive Bayes							SVM			VP		
	Recall					auc	balance	auc					
	(Org[7])	Us	(Rep[14])	(Rep[15])	(Rep[17])	(Rep[6])	(Rep[16])	(Org[5])	Us^{log}	Us	(Org[5])	Us^{log}	Us
pc4	98	87	–	–	72.6	85	82.6	92	95	50	83	87	52
pc3	80	79	–	–	80.6	81	71.4	77	94	53	74	74	47
kc4	79	80	–	–	–	68	71.9	77	85	60	73	79	75
kc3	69	78	–	–	99	83	74.1	86	85	50	74	83	62
pc1	48	73	–	–	66.2	79	64.6	80	94	56	75	79	53
cm1	71	77	–	–	81.5	72	72.7	70	96	51	72	79	54
pc2	72	86	–	–	83.3	85	81.8	85	71	50	50	50	50
mw1	52	78	–	–	100	80	70.5	65	83	50	73	77	52
avg	71	79.7	64	85	83	79	74	79	89	53	72	76	56

NB: Us^{log} denotes our results with a log transformation.

replication has been identified and we believe this threat has been mitigated.

There are different search engines (Scopus, ISI Web of Science etc.) and we chose Google Scholar because it has been effective as demonstrated by Wohlin [8] for this type of search. In addition between 2011 and 2012 Google Scholar has *"very significantly expanded its coverage... at a stable rate"* Harzing [26]. Primarily, we are concerned about getting a reliable number of citations for our analysis and not usability. Although we found it useful to reduce the number of papers to read manually due to the 'search within citing articles' feature. We are confident that Google Scholar is sufficient for our work.

Threats also exists in assessing and extracting information. We mitigated these threats; two authors in this study read and extracted information from 5 of the *final-set* of 39 papers and for the six factors extracted from all the 208 papers. Using the SLuRp tool Bowes et al. [20] any disagreements were identified and then resolved and the data updated.

The features of the data collected introduces another threat. In particular the analysis of citation count and the number of replications involves data with many ties. We therefore used Kendal's Tau correlation, rather than Spearman's correlation because it is known to deal with ties better.

We show that most threats have been minimised and believe to the best of our ability our findings are sound. We hope researchers replicate our study and our replication package is available (Online-Appendix). Under such conditions, significance tests of the Kendell correlation coefficient may be unreliable.

6. Discussion

Overall we have shown that defect prediction suffers from a lack of external replications with only 6% of 208 studies replicated. Silva et al. [4] identified 96 articles, reporting 133 replications performed between 1994 and 2010 in software engineering, indicating that replication in software engineering is carried out more frequently than in defect prediction. We also show that the few replications performed are not consistently systematic and of the 29 replications we analysed, only 18 (62%) results agreed with the original paper.

The characteristics of replicated original studies include those studies being published in the TSE journal and being based on closed source industrial data. Most of the replicated original studies do not satisfy a quality assessment ($quality_{4p}$) this despite being largely published in high impact venues. Such a potential lack of quality in original studies is surprising and suggests unreliable findings may be being propagated.

Reporting inconsistencies are also problematic for interpreting the outcomes of replications. For example, agreements are not always reported clearly, performance values of original studies are not always reported in the replication.

Our findings suggest that defect prediction replication can offer valuable lessons that can be built upon by others. The original studies that have multiple replications have demonstrated opportunities to improve defect prediction and develop more stable conclusions (e.g. (Org[5])).

Conversely, the lack of replication studies could be an indication of the need to define new research goals in defect prediction; our results may simply demonstrate decreasing interest in defect prediction. Recent criticisms of the area focus on the lack of impact that defect prediction research has in industry. For example, Lanza et al. [9] reported that the problem with defect prediction lies in how the approaches are evaluated and benchmarked and further suggested that *"researchers should seriously consider putting their predictors out into the real world and having them used by developers who work on a live code"*. Shepperd [27] mentioned that the evaluation of prediction models is problematic and *"that the concerns of researchers need to be better aligned with the likely end-users"*. Kitchenham [28] highlights the importance of these issues in relation to replication when she talked in-depth about the 4Rs (Rigour, Reproducibility, Replication and Relevance) and how they are linked; with good Rigour, there is value in Reproducing the work and also useful Replicating reproducible work to check stability across multiple organisations provided they are relevant to what the practitioners need. Kitchenham claims that *"very few papers consider practical issues"* [28] and suggests the need for obtaining more realistic datasets and collaborations with industry partners. It is criticisms related to these industry issues that are currently affecting the area of defect prediction.

Our results suggest that far more replications are needed. Furthermore that replications need to be done much more systematically. We show that important replication steps have been missed out based on the taxonomy that we applied. Incremental changes should be made to original studies in replications while analysing the effect of changes on model performance. Being systematic may be easier when artefacts are open source (for reuse and reduced variability) so that a researcher can break down a study into separate components. The typical components of a defect prediction study include, tools, statistics, cross validation, feature selection, parameter optimising, etc. (e.g. Table 5). Replicated experiments should be run with the same components as the original (reproduced), with intentional variation of changeable components implemented systematically, i.e. change one after another while recording their effect on model performance. This

systematic approach has the potential to discover those factors affecting results. A good example of systematic replication is Song et al. (Rep [16]); the study first reproduced the original Menzies et al. (Org[7]) to confirm it, then performed several combinations of the components while recording the effect on model performance.

6.1. Practical recommendations for the replication of defect prediction studies

We make the following recommendations for replication in defect prediction studies. These suggestions are not a hard and fast set of rules and as such should not be used as a mechanism to exclude papers from being replicated.

[Recommendation 1] Highly cited papers should be replicated as such papers tend to influence future defect prediction practice. Other papers should also be replicated.

[Recommendation 2] Use a replication infrastructure (e.g. OpenML [http://www.openml.org/] [29], or Zenodo [https://zenodo.org/]). Such infrastructures typically include an application programming interface API (Weka, R, REST, Java, .Net, Python, mlR, Moa) based repository designed to allow experiments to be configured on it and run on a user's machine. This keeps one version of datasets, the results, the protocol for easy sharing, and has persistence; most likely going to have the availability attribute [14] for researchers to use in future.

[Recommendation 3] Better use of existing reporting guidelines should be made. This requires the development of comprehensive software engineering reporting guidelines. These should be based on existing guidelines, including Runeson and Höst's [30] on case study design, Kitchenham et al.'s [31] on empirical software engineering, Carver's [32] on reporting replications, da Silva et al.'s [4] on designing and reporting replication studies and Mende's (Rep[9]) on replication remedies, pitfalls and challenges. Crucially, these guidelines must be collected and structured as a repository similar to the repositories that already exist in the Medical field (e.g. Munafò et al. [33, http://www.equator-network.org]).

[Recommendation 4] Replication Impact Factors should be put into practice. As Schimmack says: *"Demonstrating replicability should become an important criterion of research excellence that can be used by funding agencies and other stakeholders to allocate resources to research that advances science"* [34]. The following are possible ways in which replication can be implemented in impact factors:

- Use number of replications per study as additional impact factor metric $R-index$ [34].
- Use number of reproductions per study as additional impact factor metric $Repro_{index}$.
- Use number of replications and reproductions as the most significant impact factor metric RR_{index}.

[Recommendation 5] Quality assessments (e.g. $quality_{4p}$) should be applied to original studies. Researchers should consider quality in two parts; the quality of the methodology and quality of the reporting. These quality checks should be made on original studies before replication to minimise the spread of potentially erroneous results.

[Recommendation 6] The replication of important studies needs to be incentivised. Currently there is little reason for a researcher to replicate a study, as original studies are more likely to be cited than a replication. Highly rated publication venues should specifically encourage replications.

[Recommendation 7] Reproduction should be carried out before replication. This will demonstrate how close the replicating authors can get to the original study. There is little point attempting to replicate results if reproduction is not possible because, e.g. the raw defect data is not both accessible and held in a secure source.

These suggestions are not exhaustive. We hope that future researchers will evaluate, refine and extend these recommendations.

7. Conclusion

Replication is reported to be very important [6], yet not often enough performed in software engineering [4]. In this paper we particularly investigated replication in defect prediction - a very active area of research in software engineering. In this study we investigated the replication of 208 original defect prediction studies identified by a highly cited SLR [1].

Our findings suggest low replication in defect prediction and potential low quality in defect prediction studies. Only 13 of the 208 original studies have been replicated by researchers that are independent to those of the original studies. Only 3 of the 13 original replicated studies are assessed as quality studies with regards to research methodology and reporting. We have also shown some of the difficulty in comparing original results with replicated results, as replications can report their results using measures not used by original studies. This reporting inconsistency makes comparing results difficult.

We have given some practical suggestions to incentivise and standardise aspects of replication suggesting, for example the calculation of a new Replication and Reproduction impact factor, data sharing, and guidelines of reporting.

Our results show that studies published in a high impact journal (in particular TSE) tend to attract replications. This means that there is an opportunity that these publication venues could come up with ways to encourage more replications, for example a best replication paper award could be created. Industrial based original studies also seem to have more replications.

We hope our study drives discussions along the line of our suggestions and we hope researchers replicate and extend this study to get more insight into replication across Software Engineering.

Appendix A. Components that make up a defect prediction study

Table A.15
Changeable components of defect prediction studies adapted from [5].

Protocol	Operationalisation	Populations	Experimenter
Definition: The configuration of subcomponents to observe an outcome **Changes:** -Experimental design of how treatments are allocated, e.g. model building framework configures data preprocessing, parameter optimisation, cross validation, prediction, data collection framework configures defect linking, extracting and labelling -Statistical analyses	**Definition:** Mode of applying treatments (techniques) e.g. training a model on train and test set gives unrealistic results than on train set only **Changes (cause construct: cause of differences in results):** -Literature sources, training, instructions for applying a procedure during experiments -Tools used for running experiments e.g. IDE -Algorithms for, building prediction models, dealing with imbalance, linking defects **Changes (effect construct: effect on results):** -Defining dependent/ independent variables, e.g. number of defects post release/code complexity -Process of calculating the variables, e.g. number of defects fixed after released to customers and linked to the point the defect was introduced -Measuring model performance with different measures	**Definition:** The subject and objects properties used in a controlled experiment **Changes:** -Source code of project (open (Eclipse) or closed (NASA) source) -Design documents, programming language, size, complexity, maturity (years used and growth), domain etc. -Granularity of independent variables (metrics) such as class or method level, granularity of dependent variables such as defective or not and number of defects	**Definition:** The designer, trainer, monitor, measurer and analyst involved in the experiments (authors). **Changes:** -Different authors may or may not vary the parameters of an experiment on the same dataset

Appendix B. Changed components data extracted from replication studies

Table B.16
Protocol: Replication studies.

Rep. studies	Cross val.	Parameter Ttuning	Statistics	Data cleaning
Hongyu Zhang (Rep[3]), (Rep[4])	No	No	Computed the coefficient of determination and the Standard Error of Estimate	No
Ghotra et al. (Rep[8])	Yes	Yes	Scott–Knott statistical test	Mixed: Yes on NASA, No on Apache family
Leszak Marek (Rep[10])	No	No	No	No
Mende and Koschke (Rep[11])	10 by 10 cross validation	No	Friedman test, Nemenyi post hoc test	No
Galinac Grbac et al. (Rep[12])	NA	NA	Pearson correlation coefficient, nonparametric Spearman correlation, vote counting	Yes: removed duplicates, outliers
Devine et al. (Rep[13])	No	No	Spearman correlation	No
Hamill and Goseva-Popstojanova (Rep[2])				
Turhan and Bener (Rep[14])	10 by 10 randomised	No	*t*-test	No
Zhang et al. (Rep[15])	10 by 10 randomised	No	No	Removed duplicates, missing values
Song et al. Rep[16]	10 by 10 randomised	Yes	% difference, Wilcoxon signed 1-tailed	Removed outliers, missing values
Singh and Verma (Rep[17])	10 by 10 stratified randomised	No	No	No
Krishnan et al. (Rep[17])	10 by 1000 randomised	No	Figner–Killeen, Kruskal–Wallis, one-way ANOVA, *t*-test with Bonferroni correction of *p*-value	No
Rahman et al. (Rep[19])	No	No	Wilcoxon one sided paired	No
Tosun et al. (Rep[20])	split-sample 10 by 5	No	Spearman, Pearson	No
Nguyen et al. (Rep[21])	split-sample by 50	No	Spearman rank correlation, Wilcoxon rank test, ANOVA	No
Premraj and Herzig (Rep[22])	stratified hold-out	No	Kruskal–Wallis two pairs test, ANOVA	Yes
Okutan and Yıldız (Rep[23])	10 by 20 randomised stratified	No	*t*-test	Yes
Duala-Ekoko and Robillard (Rep[24])	No	No	Chi-square	No
Mende (Rep[9])	50 by 10fold and 10fold Cross-Val			
Kpodjedo et al. (Rep[25])	No	No	Wilcoxon signed rank test, Cohen-d statistics	No
Li et al. (Rep[26])	No	No	Weibull, Power, Gamma, Exponential, Theil	No

Table B.17
Operationalisation: Replication studies.

Rep. studies	Tools	Algorithms	Independent var	Dependent var
Hongyu Zhang (Rep[3]), (Rep[4])	SPSS	Non linearReg	Static code (complexity), structure of abstract syntax tree (no of nodes etc.)	Number of defects (prerelease, post-release).
Ghotra et al. (Rep[8])	Weka	Statistical, Clustering, Rule-based, Nearest Neighbours, NeuralNet, SVMs, Tree-based, Ensembles	Static code, CK, QMOOD, Martin's	Defective or Not defective
Leszak Marek (Rep[10])	ClearDDTSTM (from IBM-Rational), ClearCaseTM (from IBM-Rational)	Correlation analysis	Process, Complexity of changes, source file size, file age, defect density (file, release)	Number of defects (prerelease, post-release)
Mende and Koschke (Rep[11])	R	RandomForest	Static code metrics	Defective or Not defective
Galinac Grbac et al. (Rep[12])	No	No	Size (LOC)	Number of faults (pre-release, analyse post-release)
Devine et al. (Rep[13])	SourceMonitor (metrics), StatSVN (analyse SVN logs)	Stepwise regression	Source code, change, fault metrics	Number of defects, defect density
Hamill and Goseva-Popstojanova (Rep[1]) (Rep[2])	No	No	Fault types, detection activities, severity	Number of faults (prerelease, post-release)
Turhan and Bener (Rep[14])	Matlab: no version	Naive Bayes, Linear Discriminant, Quadratic Discriminant	Static code	Defective or Not defective
Zhang et al. (Rep[15])	Function to Component level data aggregator, Weka: No version.	BayesNet, Bagging, k-NN, RandFor, NeuralN, LogisticR, RBFNet, SVM, NaiveBayes, C4.5(J48), K-Star, AdaBoostM1	LOC, CyclComplex and HalsteadVol	Defective or Not defective
Song et al. Rep[16]	No	Naive Bayes, J48, OneR	Static code	Defective or Not defective
Singh and Verma (Rep[17])	No	K-means	Static code	Defective or Not defective
Krishnan et al. (Rep[18])	CVSPS(capture commit transactions), Weka, R	J48	Change	Defective, Not-defective
Rahman et al. (Rep[19])	Git	Least recently used (LRU)	Churn, temporal locality, spatial locality	defect density and the cost effectiveness of inspection
Tosun et al. (Rep[20])	Ucinet 6 Network Analysis tool (network metrics), open-source metrics extraction tool, Prest (dependencies)	NaiveBayes, LogisticReg, LinearReg	Complexity, network	Defective or Not defective
Nguyen et al. (Rep[21])	UCINET (network metrics), Structure101 (extract dependencies), Understand (complexity metrics)	linearReg, logisticReg	Import packages, Network metrics, Complexity, OO, Function	Number of faults(post-release), Defective or Not defective
Premraj and Herzig (Rep[22])	Understand V2.0 Build 505 (metrics for Java, C++), JDT frame (map class back to file), UCINET (network metrics), R	KNN, LogisticReg, NaiveBayes, Rpart, SVM, Tree-Bagging	Code, socio-technical (network metrics), combined	Number of defects, Defective or Not defective
Okutan and Yıldız (Rep[23])	PMD source code analyser plugin in Netbeans (for LOCQ), Weka (no version)	Bayesian Networks	OO, static code	Defective or Not defective, Number of defects
Duala-Ekoko and Robillard (Rep [24])	SemDiff and Mylyn (mapping bug fixes to bug reports)	No	Import packages, classes	Number of faults
Mende (Rep[9])	R	logisticReg, NaiveBayes	Process, change, entropy of change, entropy, churn of source code, source code metrics, CK, OO	Number of defects
Kpodjedo et al. (Rep[25])	PADL (extract CK metrics)	logistic regression	Design metrics (class diagrams)	Defective or Not defective
Li et al. (Rep[26])	R	PCA, linearReg, k-means, non-linearReg, CART	Change, development metrics (no of in-dev process defects)	Number of faults

Table B.18
Populations: Replication studies.

Rep. Studies	Prog. Lang.	Domain	Granularity	Source
Hongyu Zhang (Rep[3]), (Rep[4])	Java	IDE	Package	Eclipse Versions 2.0, 2.1, 3.0
Ghotra et al. (Rep[8])	C, Java	(Satellite, Flight, Storage), web	Module (predictions), code (metrics)	NASA, PROMISE, Apache, GNU
Leszak Marek (Rep[10])	C++, C, shell script, tcl/tk, perl	Telecom	File (metrics)	Lucent
Mende and Koschke (Rep[11])	R		Static code metrics	Defective or Not defective
Galinac Grbac et al. (Rep[12])	PLEX	Telecom	Module, file	Industrial 5 releases
Devine et al. (Rep[13])	Java	Software testing tools	Component	PolyFlow Software Product Line: 4 projects
Hamill and Goseva-Popstojanova (Rep[1]) (Rep[2])	No	Flight software safety-critical	Component	NASA (not public)
Turhan and Bener (Rep[14])	C, Java	Satellite, Flight, Storage	Method	NASA
Zhang (Rep[15])	C, Java	Satellite, Flight, Storage	Component	NASA
Song et al. Rep[16]	C, Java	Satellite, Flight, Storage	Method	NASA, PROMISE
Singh and Verma (Rep[17])	C, Java	Satellite, Flight, Storage	Method	NASA, PROMISE
Krishnan et al. (Rep[18])	Java, C, C++	IDE	File (predictions)	Eclipse 2.0, 2.1, 3.0, 3.3, 3.4, 3.5, 3.6
Rahman et al. (Rep[19])	C, Java	Web server, Image manipulator, File manager, Email client, Text search engine	File (predictions)	Apache Httpd, Gimp, Nautilus, Evolution, Lucene
Tosun et al. (Rep[20])	C, Java	IDE, Embedded systems	Function, file (predictions), file (metrics)	Eclipse, Embedded systems
Nguyen et al. (Rep[21])	Java	IDE	Class, package (predictions) Classes, Packages, Function (metrics)	Eclipse
Premraj and Herzig (Rep[22])	Java	IDE	Source files (predictions), class and method (metrics)	JRuby, ArgoUML, Eclipse
Okutan and Yıldız (Rep[23])	Java	Web Server	Class (predictions)	Apache Family
Duala-Ekoko and Robillard (Rep[24])	Java	IDE	File, Method (predictions)	Eclipse version 2,3
Mende (Rep[9])	Java	IDE	Class (predictions, metrics)	Eclipse (Mylyn, Equinox, PDE, Lucene, Score)
Kpodjedo et al. (Rep[25])	Java	IDE, UML, Javascript Interpreter	Class (predictions), Class diagram (metrics)	ArgoUML, Rhino
Li et al. (Rep[26])	No	Operating System	Release (predictions)	IBM

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.infsof.2018.02.003.

References

[1] T. Hall, S. Beecham, D. Bowes, D. Gray, S. Counsell, A systematic literature review on fault prediction performance in software engineering, Software Eng. IEEE Trans. 38 (6) (2012) 1276–1304, http://dx.doi.org/10.1109/TSE.2011.103.

[2] T. Hall, D. Bowes, The state of machine learning methodology in software fault prediction, Machine Learning and Applications (ICMLA), 2012 11th International Conference on, 2 (2012), pp. 308–313, http://dx.doi.org/10.1109/ICMLA.2012.226.

[3] D. Bowes, T. Hall, D. Gray, Dconfusion: a technique to allow cross study performance evaluation of fault prediction studies, Autom. Software Eng. 21 (2) (2014) 287–313, http://dx.doi.org/10.1007/s10515-013-0129-8.

[4] F.Q. Silva, M. Suassuna, A.C. França, A.M. Grubb, T.B. Gouveia, C.V. Monteiro, I.E. Santos, Replication of empirical studies in software engineering research: a systematic mapping study, Empir. Software Engg. 19 (3) (2014) 501–557, http://dx.doi.org/10.1007/s10664-012-9227-7.

[5] O.S. Gómez, N. Juristo, S. Vegas, Understanding replication of experiments in software engineering: a classification, Inf. Software Technol. 56 (8) (2014) 1033–1048.

[6] J.P. Ioannidis, Why most published research findings are false, PLoS Med. 2 (8) (2005) e124.

[7] R. Moonesinghe, M.J. Khoury, C.J. Janssens, Most published research findings are false–but a little replication goes a long way, PLoS Med. 4 (2) (2007).

[8] C. Wohlin, Guidelines for snowballing in systematic literature studies and a replication in software engineering, Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, EASE '14, ACM, New York, NY, USA, 2014, pp. 38:1–38:10, http://dx.doi.org/10.1145/2601248.2601268.

[9] M. Lanza, A. Mocci, L. Ponzanelli, The tragedy of defect prediction, prince of empirical software engineering research, IEEE Software 33 (6) (2016) 102–105, http://dx.doi.org/10.1109/MS.2016.156.

[10] Z. Yu, N.A. Kraft, T. Menzies, How to read less: better machine assisted reading methods for systematic literature reviews, CoRR (2016). abs/1612.03224

[11] J.T. Leek, R.D. Peng, Opinion: reproducible research can still be wrong: adopting a prevention approach, Proc. Natl. Acad. Sci. 112 (6) (2015) 1645–1646, http://dx.doi.org/10.1073/pnas.1421412111.

[12] L. Madeyski, B. Kitchenham, Would wider adoption of reproducible research be beneficial for empirical software engineering research? J. Intell. Fuzzy Syst. 32 (2) (2017) 1509–1521, http://dx.doi.org/10.3233/JIFS-169146.

[13] M.J. Shepperd, D. Bowes, T. Hall, Researcher bias: the use of machine learning in software defect prediction, IEEE Trans. Software Eng. 40 (6) (2014) 603–616, http://dx.doi.org/10.1109/TSE.2014.2322358.

[14] J.M. González-Barahona, G. Robles, On the reproducibility of empirical software engineering studies based on data retrieved from development repositories, Empir. Software Eng. 17 (1) (2012) 75–89, http://dx.doi.org/10.1007/s10664-011-9181-9.

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

[16] S.N. Goodman, D. Fanelli, J.P.A. Ioannidis, What does research reproducibility mean? Sci. Transl. Med. 8 (341) (2016), http://dx.doi.org/10.1126/scitranslmed.aaf5027. 341ps12–341ps12

[17] D.W. Aksnes, Characteristics of highly cited papers, Res. Eval. 12 (3) (2003) 159–170, http://dx.doi.org/10.3152/147154403781776645.

[18] V. Garousi, J.M. Fernandes, Highly-cited papers in software engineering: the top-100, Inf. Software Technol. 71 (2016) 108–128, http://dx.doi.org/10.1016/j.infsof.2015.11.003.

[19] D. Fucci, G. Scanniello, S. Romano, M. Shepperd, B. Sigweni, F. Uyaguari, B. Turhan, N. Juristo, M. Oivo, An external replication on the effects of test-driven development using a multi-site blind analysis approach, Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM '16, ACM, New York, NY, USA, 2016, pp. 3:1–3:10, http://dx.doi.org/10.1145/2961111.2962592.

[20] D. Bowes, T. Hall, S. Beecham, Slurp: A tool to help large complex systematic literature reviews deliver valid and rigorous results, Proceedings of the 2nd International Workshop on Evidential Assessment of Software Technologies, EAST '12, ACM, New York, NY, USA, 2012, pp. 33–36, http://dx.doi.org/10.1145/2372233.2372243.

[21] C. Marshall, P. Brereton, B. Kitchenham, Tools to support systematic reviews in software engineering: A feature analysis, Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, EASE '14, ACM, New York, NY, USA, 2014, pp. 13:1–13:10, http://dx.doi.org/10.1145/2601248.2601270.

[22] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I.H. Witten, The weka data mining software: an update, SIGKDD Explor. Newslett. 11 (1) (2009) 10–18, http://dx.doi.org/10.1145/1656274.1656278.

[23] H.F. Moed, Measuring contextual citation impact of scientific journals, J. Informetr. 4 (3) (2010) 265–277, http://dx.doi.org/10.1016/j.joi.2010.01.002.

[24] J. Petrić, D. Bowes, T. Hall, B. Christianson, N. Baddoo, The jinx on the nasa software defect data sets, Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering, EASE '16, ACM, New York, NY, USA, 2016, pp. 13:1–13:5, http://dx.doi.org/10.1145/2915970.2916007.

[25] M. Shepperd, Q. Song, Z. Sun, C. Mair, Data quality: some comments on the nasa software defect datasets, Software Eng. IEEE Trans. 39 (9) (2013) 1208–1215, http://dx.doi.org/10.1109/TSE.2013.11.

[26] A.-W. Harzing, A longitudinal study of google scholar coverage between 2012 and 2013, Scientometrics 98 (1) (2014) 565–575.

[27] M. Shepperd, Machine learning may be the answer but is it trustworthy?.

[28] B. Kitchenam, Back to basics- the 4r's of software estimation.

[29] J. Vanschoren, J.N. van Rijn, B. Bischl, L. Torgo, OpenML: networked science in machine learning, SIGKDD Explor. 15 (2) (2013) 49–60, http://dx.doi.org/10.1145/2641190.2641198.

[30] P. Runeson, M. Höst, Guidelines for conducting and reporting case study research in software engineering, Empir. Software Eng. 14 (2) (2009) 131–164, http://dx.doi.org/10.1007/s10664-008-9102-8.

[31] B. Kitchenham, H. Al-Khilidar, M.A. Babar, M. Berry, K. Cox, J. Keung, F. Kurniawati, M. Staples, H. Zhang, L. Zhu, Evaluating guidelines for reporting empirical software engineering studies, Empir. Software Eng. 13 (1) (2008) 97–121.

[32] J.C. Carver, Towards reporting guidelines for experimental replications: a proposal, Citeseer, 2010.

[33] M.R. Munafò, B.A. Nosek, D.V.M. Bishop, K.S. Button, C.D. Chambers, N. Percie du Sert, U. Simonsohn, E.-J. Wagenmakers, J.J. Ware, J.P.A. Ioannidis, A manifesto for reproducible science, Nat. Hum. Behav. 1 (2017). 0021 EP – http://dx.doi.org/10.1038/s41562-016-0021

[34] U. Schimmack, Quantifying statistical research integrity: the replicability index, (2014), https://wordpress.com/post/replication-index.wordpress.com/920.

Original Studies

[Org1] M. D'Ambros, M. Lanza, R. Robbes, An extensive comparison of bug prediction approaches, Mining Software Repositories (MSR), 2010 7th IEEE Working Conference on, (2010), pp. 31–41, http://dx.doi.org/10.1109/MSR.2010.5463279.

[Org2] T. Zimmermann, N. Nagappan, Predicting defects using network analysis on dependency graphs, Proceedings of the 30th International Conference on Software Engineering, ICSE '08, ACM, New York, NY, USA, 2008, pp. 531–540, http://dx.doi.org/10.1145/1368088.1368161.

[Org3] T. Zimmermann, R. Premraj, A. Zeller, Predicting defects for eclipse, Predictor Models in Software Engineering, 2007. PROMISE'07: ICSE Workshops 2007. International Workshop on, IEEE, 2007, p. 9.

[Org4] C. Andersson, P. Runeson, A replicated quantitative analysis of fault distributions in complex software systems, software engineering, IEEE Trans. 33 (5) (2007) 273–286, http://dx.doi.org/10.1109/TSE.2007.1005.

[Org5] S. Lessmann, B. Baesens, C. Mues, S. Pietsch, Benchmarking classification models for software defect prediction: a proposed framework and novel findings, software engineering, IEEE Trans. 34 (4) (2008) 485–496, http://dx.doi.org/10.1109/TSE.2008.35.

[Org6] T. Ostrand, E. Weyuker, R. Bell, Predicting the location and number of faults in large software systems, software engineering, IEEE Trans. 31 (4) (2005) 340–355, http://dx.doi.org/10.1109/TSE.2005.49.

[Org7] T. Menzies, J. Greenwald, A. Frank, Data mining static code attributes to learn defect predictors, Software Eng. IEEE Trans. 33 (1) (2007) 2–13, http://dx.doi.org/10.1109/TSE.2007.256941.

[Org8] N. Fenton, N. Ohlsson, Quantitative analysis of faults and failures in a complex software system, Software Eng. IEEE Trans. 26 (8) (2000) 797–814, http://dx.doi.org/10.1109/32.879815.

[Org9] R. Moser, W. Pedrycz, G. Succi, A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction, Software Engineering, 2008. ICSE '08. ACM/IEEE 30th International Conference on, (2008), pp. 181–190, http://dx.doi.org/10.1145/1368088.1368114.

[Org10] S. Kim, T. Zimmermann, E.J. Whitehead Jr., A. Zeller, Predicting faults from cached history, Proceedings of the 29th International Conference on Software Engineering, ICSE '07, IEEE Computer Society, Washington, DC, USA, (2007), pp. 489–498, http://dx.doi.org/10.1109/ICSE.2007.66.

[Org11] S. Amasaki, Y. Takagi, O. Mizuno, T. Kikuno, A bayesian belief network for assessing the likelihood of fault content, Software Reliability Engineering, 2003. ISSRE 2003. 14th International Symposium on, (2003), pp. 215–226, http://dx.doi.org/10.1109/ISSRE.2003.1251044.

[Org12] A. Schröter, T. Zimmermann, A. Zeller, Predicting component failures at design time, Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering, ACM, 2006, pp. 18–27.

[Org13] T. Khoshgoftaar, N. Seliya, Tree-based software quality estimation models for fault prediction, Software Metrics, 2002. Proceedings. Eighth IEEE Symposium on, (2002), pp. 203–214, http://dx.doi.org/10.1109/METRIC.2002.1011339.

Replicated Studies

[Rep1] M. Hamill, K. Goseva-Popstojanova, Exploring fault types, detection activities, and failure severity in an evolving safety-critical software system, Software Qual. J. 23 (2) (2015) 229–265, http://dx.doi.org/10.1007/s11219-014-9235-5.

[Rep2] M. Hamill, K. Goseva-Popstojanova, Exploring fault types, detection activities, and failure severity in an evolving safety-critical software system, Software Qual. J. 23 (2) (2015) 229–265, http://dx.doi.org/10.1007/s11219-014-9235-5.

[Rep3] H. Zhang, On the distribution of software faults, Software Eng. IEEE Trans. 34 (2) (2008) 301–302, http://dx.doi.org/10.1109/TSE.2007.70771.

[Rep4] H. Zhang, On the distribution of software faults, Software Eng. IEEE Trans. 34 (2) (2008) 301–302, http://dx.doi.org/10.1109/TSE.2007.70771.

[Rep5] C. Andersson, P. Runeson, A replicated quantitative analysis of fault distributions in complex software systems, Software Eng. IEEE Trans. 33 (5) (2007) 273–286, http://dx.doi.org/10.1109/TSE.2007.1005.

[Rep6] S. Lessmann, B. Baesens, C. Mues, S. Pietsch, Benchmarking classification models for software defect prediction: a proposed framework and novel findings, Software Eng. IEEE Trans. 34 (4) (2008) 485–496, http://dx.doi.org/10.1109/TSE.2008.35.

[Rep7] T. Ostrand, E. Weyuker, R. Bell, Predicting the location and number of faults in large software systems, Software Eng. IEEE Trans. 31 (4) (2005) 340–355, http://dx.doi.org/10.1109/TSE.2005.49.

[Rep8] B. Ghotra, S. McIntosh, A.E. Hassan, Revisiting the impact of classification techniques on the performance of defect prediction models, Proc. of the 37th Int'l Conf. on Software Engineering (ICSE), (2015).

[Rep9] T. Mende, Replication of defect prediction studies: Problems, pitfalls and recommendations, Proceedings of the 6th International Conference on Predictive Models in Software Engineering, PROMISE '10, ACM, New York, NY, USA, 2010, pp. 5:1–5:10, http://dx.doi.org/10.1145/1868328.1868336.

[Rep10] M. Leszak, Software defect analysis of a multi-release telecommunications system, Product Focused Software Process Improvement, Springer, 2005, pp. 98–114.

[Rep11] T. Mende, R. Koschke, Effort-aware defect prediction models, Software Maintenance and Reengineering (CSMR), 2010 14th European Conference on, (2010), pp. 107–116, http://dx.doi.org/10.1109/CSMR.2010.18.

[Rep12] T.G. Grbac, P. Runeson, D. Huljenić, A second replicated quantitative analysis of fault distributions in complex software systems, Software Eng. IEEE Trans. 39 (4) (2013) 462–476, http://dx.doi.org/10.1109/TSE.2012.46.

[Rep13] T.R. Devine, K. Goseva-Popstojanova, S. Krishnan, R.R. Lutz, J.J. Li, An empirical study of pre-release software faults in an industrial product line, 2012 IEEE Fifth International Conference on Software Testing, Verification and Validation, (2012), pp. 181–190, http://dx.doi.org/10.1109/ICST.2012.98.

[Rep14] B. Turhan, A. Bener, A multivariate analysis of static code attributes for defect prediction, Quality Software, 2007. QSIC '07. Seventh International Conference on, (2007), pp. 231–237, http://dx.doi.org/10.1109/QSIC.2007.4385500.

[Rep15] H. Zhang, X. Zhang, M. Gu, Predicting defective software components from code complexity measures, Dependable Computing, 2007. PRDC 2007. 13th Pacific Rim International Symposium on, (2007), pp. 93–96, http://dx.doi.org/10.1109/PRDC.2007.28.

[Rep16] Q. Song, Z. Jia, M. Shepperd, S. Ying, J. Liu, A general software defect-proneness prediction framework, Software Eng. IEEE Trans. 37 (3) (2011) 356–370, http://dx.doi.org/10.1109/TSE.2010.90.

[Rep17] P. Singh, S. Verma, An efficient software fault prediction model using cluster based classification, Int. J. Appl. Inf. Syst. (IJAIS) 7 (3) (2014) 35–41.

[Rep18] S. Krishnan, C. Strasburg, R.R. Lutz, K. Goseva-Popstojanova, K.S. Dorman, Predicting failure-proneness in an evolving software product line, Inf. Software Technol. 55 (8) (2013) 1479–1495, http://dx.doi.org/10.1016/j.infsof.2012.11.008. http://www.sciencedirect.com/science/article/pii/S0950584912002340 .

[Rep19] F. Rahman, D. Posnett, A. Hindle, E. Barr, P. Devanbu, Bugcache for inspections: hit or miss? Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering, ESEC/FSE '11, ACM, New York, NY, USA, 2011, pp. 322–331, http://dx.doi.org/10.1145/2025113.2025157.

[Rep20] A. Tosun, B. Turhan, A. Bener, Validation of network measures as indicators of defective modules in software systems, Proceedings of the 5th International Conference on Predictor Models in Software Engineering, PROMISE '09, ACM, New York, NY, USA, 2009, pp. 5:1–5:9, http://dx.doi.org/10.1145/1540438.1540446.

[Rep21] T. Nguyen, B. Adams, A. Hassan, Studying the impact of dependency network measures on software quality, Software Maintenance (ICSM), 2010 IEEE International Conference on, (2010), pp. 1–10, http://dx.doi.org/10.1109/ICSM.2010.5609560.

[Rep22] R. Premraj, K. Herzig, Network versus code metrics to predict defects: areplication study, Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium on, (2011), pp. 215–224, http://dx.doi.org/10.1109/ESEM.2011.30.

[Rep23] A. Okutan, O.T. Yıldız, Software defect prediction using bayesian networks, Empir. Software Eng. 19 (1) (2014) 154–181.

[Rep24] E. Duala-Ekoko, M.P. Robillard, A detailed examination of the correlation between imports and failure-proneness of software components, Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement, ESEM '09, IEEE Computer Society, Washington, DC, USA, (2009), pp. 34–43, http://dx.doi.org/10.1109/ESEM.2009.5316047.

[Rep25] S. Kpodjedo, F. Ricca, P. Galinier, Y.G. Guéhéneuc, G. Antoniol, Design evolution metrics for defect prediction in object oriented systems, Empir. Software Eng. 16 (1) (2011) 141–175, http://dx.doi.org/10.1007/s10664-010-9151-7.

[Rep26] P.L. Li, M. Shaw, J. Herbsleb, P. Santhanam, B. Ray, An Empirical Comparison of Field Defect Modeling Methods, (2005).