Noname manuscript No.
(will be inserted by the editor)

Operationalizing Validity of Empirical Software Engineering Studies

Simulation-Based Testing

Johannes Härtel · Ralf Lämmel

Received: date / Accepted: date

Abstract Empirical Software Engineering studies apply methods, like linear regression, statistic tests, or correlation analysis, to better understand software engineering scenarios. Assuring the validity of such methods and corresponding results is challenging but critical. This is also reflected by quality criteria on the validity that are part of the reviewing process for the corresponding research results. However, such criteria are often hard to define operationally and thus hard to judge by the reviewers.

In this paper, we describe a new strategy to define and communicate the validity of methods and results. We conceptually decompose a study into an empirical scenario, a used method, and the produced results. Validity can only be described as the relationship between the three parts. To make the empirical scenario fully operational, we convert informal assumptions on it into executable simulation code that leverages artificial data to replace (or complement) our real data. We can then run the method on the artificial data and examine the impact of our assumptions on the quality of results. This may operationally i) support the validity of a method for a valid result, ii) threaten the validity of a method for an invalid result if assumptions are controversial, or iii) invalidate a method for an invalid result if assumptions are plausible.

We encourage researchers to submit simulations as additional artifacts to the reviewing process to make such statements explicit. Rating if a simulated scenario is plausible or controversial is subjective and may benefit from involving a reviewer. We show that existing empirical software engineering studies can benefit from such additional validation artifacts.

F. Author
first address
Tel.: +123-45-678910
Fax: +123-45-678910
E-mail: fauthor@example.com

S. Author
second address

Keywords Simulations · Empirical Methods · Validity · Threats · Testing

1 Introduction

The reviewing process of empirical work is challenging because quality criteria on the validity of methods and results are hard to define and communicate. For example, the *International Conference on Software Maintenance and Evolution* (ICSME) includes a dedicated category for empirical work. Calls-for-papers in the years 2021, 2022, and 2023 include this statement to describe the reviewing of such a category:

An empirical work is a *'[...] paper in which the main contribution is the empirical study of a software evolution technology or phenomenon. [...]* ***The authors should provide convincing arguments [...] why certain methods or models are needed. Such a contribution will be judged on its study design, the appropriateness and correctness of its analysis, and its discussion of threats to validity.*** *Replications are welcome.'* (copy from the call-for-papers of ICSME 2021, 2022, and 2023)

Similar statements on the validity (or correctness) of work can be found in other empirical fields. While reproducibility and replicability are somewhat understood [20, 74], standardized and operational ways to define and communicate the validity of methods and results, and the threats to it, are less understood. An example of recent work that points out challenges of defining and communicating threats, in the context of program comprehension experiments, is [8]. Our paper focuses on this latter aspect.

1.1 Meta Research Questions

In this paper, we try to better understand the general problems related to the validity of methods and results in empirical research with a focus on scenarios coming from the field of Mining Software Repositories (MSR) and Empirical Software Engineering (ESE). We hope that this effort also contributes to an improved understanding in other empirical fields.

We derive the following meta research questions, that we later instantiate for concrete empirical studies:

- **RQ 1:** What assumptions of ESE and MSR studies can be operationalized?
- **RQ 2:** What is the impact of such assumptions on the study results?

The first question asks for a more operational way of expressing assumptions about an empirical scenario. Such operational form can be helpful in the communication and the reviewing process. This is the foundation to discuss validity. The second question asks about the impact of assumptions on the results of a study. This is relative to the method used to produce results and alternatives to it.

1.2 Relevance

Our research questions matter from an author and a reviewer perspective: Authors of empirical research try to assess and explain the validity of their empirical research. Reviewers need guidelines on what validation they should search for in submissions.

1.3 A New Validation Strategy: Simulation-based Testing

This paper presents a new strategy that operationalizes statements about the validity of empirical studies.

We operationalize assumptions on an empirical research scenario, typically informal in a paper, by simulations that produce artificial data and results. The impact of a used method on its results can be checked in such a transparent setting.

We encourage researchers to **submit simulation code as validation artifacts in the reviewing process of empirical work** to define and communicate properties related to the validity of their methods.

1.4 Meta-Validation

What we propose is a general strategy (or method) to validate methods and results for a concrete empirical scenario. To provide a validation on our part, we applied our general method to 6 real scenarios examined in published studies in MSR and ESE. In each case, we instantiate our meta research questions and show how we can answer them.

We prove that benefit can be expected by such additional validation artifacts. We show that we can either: i) support validity ii), threaten validity, or iii) prove invalidity of the used methods and results by simulated scenarios.

Data Availability Statement

All artifacts and data sets are provided online on GitHub[1].

Contributions and Delta to the Conference Version

The following points describe the original contributions of the conference version of this paper [33] and the delta of this journal version.

[1] `https://github.com/topleet/MSR2022`

- **Conference 1**: We contribute **simulation-based testing** as a general strategy to operationalize statements about the validity of empirical research.
- **Conference 2**: We contribute a first attempt at **meta-validation** showing the relevance of simulation-based testing by applying it to published research in MSR/ESE. We show what benefit can be expected for the validation of concrete studies.
- **Conference 3**: We contribute an initial **catalog of artifacts** operationalizing statements about the validity of MSR/ESE research. The artifacts are available on GitHub.
- **Journal 1**: The journal version adds **two empirical scenarios** on top of the four of the conference version. These additional scenarios add new dimensions to the discussion. The first scenario shows how parts of the results, created using an inappropriate method, can still be proven valid. The second simulation shows empirical research conducting experiments, thereby switching from observational to experimental research methods.
- **Journal 2**: We provide a **stronger background** on the motivation for this work. This includes a positioning in the context of the reviewing process of empirical work. The validation of methods and results is an essential part of it. In particular, we introduce meta research questions, relevant for reviewing, and instantiate them for concrete empirical studies to operationalize validation in a more uniform manner.
- **Journal 3**: We improved the **introduction, structure, terminology and description**, based on the comments of the journal reviewers, and misunderstandings that we spotted in discussions that took place, after the publication of the conference version.

Roadmap

Sec. 2 starts with **existing strategies** for the validation of empirical work. **Sec. 3** provides an **overview** of our general validation strategy. **Sec. 4** exercises the strategy for a simple introductory example and **shows how to write simulation artifacts**. **Sec. 5** describes the common structure for applying simulation-based testing to MSR/ESE studies, as exercised in subsequent sections, and summarizes the main insights. **Sec. 6-11** provide the **meta-validation** of the new strategy instantiating the meta research questions in a number of MSR/ESE studies with complementary aspects of validity and simulation. **Sec. 12** presents a discussion of related work. **Sec. 13** concludes the paper.

2 Existing Validation Strategies for Empirical Research

We start with the discussion of existing strategies for the validation of empirical work that come close to our general strategy. We add some known examples of empirical work where validation has gone wrong. This motivates the complexity of validation.

The section can be skipped if a positioning in the context of reviewing and validating empirical work is not of immediate interest upon first reading.

2.1 What is a Valid Method?

Reasoning about the validity of a method used in empirical research is hard:

– There might be trivial problems. An example of a study, where columns in the data are accidentally flipped, is described in [51]. The authors needed to withdraw five publications because of this mistake in their method. Such trivial bugs might be a detail that nobody notices for years, but a detail that changes the results dramatically. Nothing prevents researchers from running into such problems.
– There might be more subtle problems. An instance is a study about cultures with moralizing gods in the field of anthropology, criticized in [10]. Here, a small difference in the method, in particular, on how to handle missing values in historical records, leads to dramatic changes in the results. There is no real solution to the problem, as discussed in [50] (page 512). Such reasoning relates very plausible assumptions on the empirical scenarios, in particular, about the origin of missing data, to the method and its incapability of producing valid results.

What can be noticed in such discussions is that it is hard to judge a method by its results. A comparison of methods might indicate differences in the results, but differences alone cannot always tell something about validity. An understanding of the empirical scenario is necessary for claims on validity.

2.2 Typical Strategies

Typical strategies that may help with the validity of empirical work will be listed next. The use of simulations is not established. The following list summarizes our experience with validation strategies we spotted in past publications of MSR/ESE.

– **Intuition**: Research in MSR/ESE is typically conducted, reviewed and read by software engineers. This implies that all results and empirical scenarios can, to some extent, be judged by our intuition. We often see sections in publications that trigger this, giving small anecdotes, and explaining why particular results are intuitive or not. Some examples can be found here [26,30,83,59,67,42,16]. The most characteristic text passage we may find in papers is '*. . . results confirms our intuition that . . .* '. However, such judgment of results might be dangerous.

- **Authority**: In some cases, authors may establish authority, which creates an impression of validity. This may be done by an extensive discussion of related work or referring to previous efforts done by the authors.
- **Cookbook Methods**: In several cases, claims on the validity of methods (and parts of it) are outsourced by using previously established methods. Papers list references to previous work, using the method too, to prove validity. Established methods that have been used for a while, are for instance, the events-per-variable (EPV), introduced in [58] and used as part of MSR/ESE methods in [30,43,76,77,60], or AIC, introduced in [1], and used in MSR/ESE method like [80,60,62,19,40]. However, if a methods works out-of-the-box in a new empirical scenario is not always clear.
- **Comparison Results**: Examining the consistency of results with previous work is also typical. This can be done by replication of previous studies, or by a meta-analysis. Typically, studies are applied in a closely similar scenario and on fresh data. An example is presented in [52], where a detailed table makes explicit to which previous studies the new results conform or not. Consistency is assumed to be a good sign in favor of a study. Less formal replication, comparison, and confirmation of previous results, located in the text, rather than in a table, can be found in [19,47,81]. However, whether consistency alone may be taken as a sign for a valid study can be doubted. It may also be caused by the repeated application of an invalid method.
- **Comparison Methods**: The comparison of methods may indicate a difference in the results. In specific cases, as in the case of a model comparison (see the next item), the difference is meaningful. However, in general, such a difference may not necessarily indicate which method is valid. Without a clear understanding of the relation between the empirical scenario, method and results, statements about validity are limited.
- **Model Comparison Method**: We consider the comparison of models that are proxies for different hypotheses as one particular method. It is maybe the most established and useful in empirical research. Here, a comparison selects between different models by preferring those that fit the collected data best. Relevant part of such method is the protection against over and under-fitting by cross-validation (used in [7,18,23,34,56]), information criteria (used in [80,60,62,19,40]) or regularization (used in [57]). Model comparison is well understood, but still limited. Model comparisons can, for instance, be misleading when talking about causality. Non-causal models are often preferred because they improve the fit. In general, associating models and hypotheses is not trivial and might go wrong.
- **Simulations**: The use of simulation is a recent trend that is currently starting to spread in teaching statistics [29,50]. According to our knowledge, it is not a formal requirement in the reviewing process.

In this paper, we will go this new way of simulating empirical scenarios and results to support or threaten the validity of methods used in MSR and ESE.

3 A New Validation Strategy: Simulation-Based Testing

This section introduces simulation-based testing to operationalize statements about the validity of empirical research in MSR and ESE.

We call our strategy 'simulation-based testing' because of the analogy between writing simulation artifacts and writing test cases.

3.1 A Simplified Empirical Study

For a structured discussion, we decompose empirical research studies into i) an empirical scenario, ii) a method, and iii) the results. This is a strong simplification.

– The *empirical scenario* is the domain-specific and more 'informal' part of a study that is the subject to research.
– The *method* consists of the steps to produce results.
– The *results* are non-trivial statements about the empirical scenario.

The following logic statement (that should not be read with formal ambitions) illustrates the argumentation on validity in many studies: *'Valid assumptions about the empirical scenario and a valid method imply valid results.'*

$$\text{empirical scenario} \land \text{method} \rightarrow \text{results}$$

Judging the validity of the results is often complicated and influenced by expectations. Instead, researchers judge the assumptions about the empirical scenario and the validity of the used method. Correct results are implied ($\rightarrow$). However, such argumentation requires a clear understanding of the relationship between the three parts: How do the assumptions on the empirical scenario and the method influence the results?

In this paper, we will show how to operationalize statements about the validity in terms of the relationship between the assumption about the empirical scenario, the method, and the results.

3.2 Variables and Relationships

In one form or another, an empirical study suggests a description of a data generation process:

– **Variables** label relevant data for the research scenario. Typical variables in MSR/ESE are the occurrence of defects, lines-of-code, or effect strengths. The measurement of such variables matters: Some variables can be *observed* (e.g., lines-of-code); some variables can only be observed with uncertainty

(e.g., defects due to inaccuracy of SZZ [73]); other variables are conceptual and thereby in principle *unobserved.* They can be inferred as the result of an empirical study (e.g., effect strength of lines-of-code influencing defects).
- **Relationships between variables** describe how variables relate to each other. Relationships may be functional or stochastic. The latter invokes a notion of uncertainty. Uncertainty is handy, since we do not find exact relationships between variables. The direction of such relationships is relevant for claims on causation and to describe a process. Temporal precedence of variables sometimes limits the plausible directions in which a relationship may operate. Relationships can never be measured directly.

Different algorithms can now use relationships to *infer*, *predict*, *identify*, *learn* or *simulate* variables using other variables.

Depending on which variables are observed or not, how unobserved variables are treated, and which relationships are used, the names for the procedure and the involved algorithms may differ:

- Algorithms may *predict* unobserved defects.
- Algorithms may *infer*, *learn* or *identify* unobserved parameters relevant to a relationship. We stick to the term *identify.*
- Algorithms may *simulate* complete data sets following plausible assumptions on unobserved variables, like on the relationship between defects and lines-of-code.

3.3 The Baseline Empirical Method

We now describe a common underlying method used in many empirical studies. It is not part of our new strategy, but relevant to it. We describe it here to make this discussion self-contained.

Studies often execute a variation of the following steps to arrive at the results:

- A study's method uses the toolbox of variables and relationships to decode hypotheses about the empirical scenario as 'models'. There are often stereotypical ways of connecting variables by relationships, with well-known names and algorithmic support. For instance, there are linear (regression) models, logistic (regression) models, mixed-effect models, autoregressive models, or generalized additive models. These names just refer to blueprints that still need to be customized.
- The data collection mines a sample, for instance repositories, to replace some unobserved with observed variables.
- Algorithms attempt to identify the remaining unobserved variables in the models.
- The identified unobserved variables, often called the parameters, and the model that fits the data best is considered as the result of a study. Hypothesis are discussed respectively.

We can find comparable practice in the following papers [46,24,82,15,25,68,53,85,43,55,63,88,87,78,80,76].

3.4 Simulation-Based Testing in a Nutshell

Judging the validity of such a method and variations of it is complicated because it is connected to assumptions on the empirical scenario. **To make the assumptions transparent, we operationalize them in terms of a simulation.**

A simulation-based test replaces the real empirical scenario, in essence the collected data, by a ‘simulation’ that reflects our assumptions about the empirical scenario. The simulation produces artificial data.

The simulation-based test often works in an ‘opposite way’ when compared to the original method. Instead of using the observed data to produce results, it reverts the algorithms, and produces artificial data for given results. The results can be purely fictional (or counterfactual) and are typically invented for the purpose of simulation.

An empirical study’s original method, or any other method that potentially applies, can be run on the simulated data.

The simulated empirical scenario, the method, and the results are now transparent. The relationship can operationally be judged. The impact of our assumption on the results can operationally be judged.

Possible Statements on Validity: Simulated scenarios operationalize statements about validity by describing the relationship between (plausible or controversial) simulated empirical scenarios, (alternative) methods, and the impact on the results. We have different options to operationalize statements about validity:

– **Supporting Validity**: We want to show that for all plausible simulations (i.e., those likely to correspond to the real scenarios) our method produces valid results.
– **Threatening Validity**: We accept that for some controversial simulations (i.e., those unlikely to correspond to the real scenarios) our method produces invalid results.
– **Invalidity**: We do not accept that for a plausible simulation (i.e., one likely to correspond to the real scenarios) our method produces invalid results.

Our paper encourages researchers to capture those kinds of statements in simulations operationally. This may happen upfront to research or during revisions.

3.5 Explicit, Operational Assumptions

Running the method of the empirical study in reverse, capturing all its assumptions in simulation code, is not as trivial as it seems. It leads to non-trivial insights, as we will show in the meta-validation section.

The simulation can 'hard-code' assumptions that are mostly implicit in the original method of a study. The simulation can be explicit on the process, given by the direction of relationships, on how the data is sampled from a bigger population, on mechanisms that lead to missing data, on unobserved data like the results, or on the way how measurement works.

This creates an operational and potentially parametrized simulation of our assumed reality. For instance, a simulation may first simulate artificial data, and then simulate different mechanisms on how parts of the data get lost. This simulates problems with measurement. How a methods reacts to such data can then be examined operationally by running the simulation and afterwards the method on the produced artificial data.

3.6 Plausible and Controversial Simulated Scenarios

A simulated scenario can be considered as a complex but operational form of an assumption.

We iterate plausible or controversial simulations to strengthen the method by examining the impact on the results. Comparable to a regular assumption, everything derived in that way is conditional on the plausibility. In this case, it is the plausibility of the simulation that we deliver as an operational artifact. Rating if simulations are plausible or controversial is often subjective and can benefit from the involvement of reviewers. However, we now have an artifact that supports such review and discussion.

When designing, reviewing, or revising a method, the aim should be to stick to the most basic method, which is most resistant against the important threats operationalized by simulations. A catalog covering simulations of typical threats in MSR/ESE may help in such a case.

4 A Simple Example: Logistic Regression For Defects

We start with a simple example of how a simulation may operationalize statements on the validity of a method. We will check the application of a basic logistic regression model, comparable to the method in many past and recent studies on software defects (e.g., in [46,24,53,85,43,55,88,87,78]).

Technically, a simulation implements stochastic and functional relationships between variables. It draws variables from random number generators (stochastic) or produces variables according to basic mathematical functions (functional). Such simulations boil down to very basic code.

Understanding the following section does not involve specific libraries or extensive statistic background knowledge.

4.1 Original Method

A study that uses logistic regression to examine defects formulates a model that describes the relationship between some observed variable X, typically a software metric, and the observed defect Y. Both variables can be mined from repositories. The results of such method improve the understanding of the relationship, for instance, discussed in [46,24,53,85,43,55,88,87,78].

The most basic form of a logistic regression characterizes the relationship in terms of two unobserved variables, identified using the observed X and Y variable:

- **Intercept (alpha)**: The intercept is an unobserved variable reflecting the average probability of defects when $X = 0$.
- **Slope (beta)**: The slope is an unobserved variable reflecting the change in the probability of defects when X increases by one unit.

To exemplify this analysis, we borrow data from the *elasticsearch* project, published in the context of examining defects in [24]. We use a binary defect classification for our observed variable Y (computed according to SZZ [73]). The variable X is the stated-log transformed lines-of-code changed by a commit. For further details, we refer to the original study.

The code we use to invoke a logistic regression, in essence, an algorithm that identifies alpha and beta using X and Y, is given in Listing 1:

```
model ← glm(Y ∼ X, family = binomial())
```

Listing 1 The original method applies a logistic regression to model the relationship between X and Y (R code).

When running this code, it reports that the unobserved intercept is −3.28 and the slope is 0.45. From the perspective of the results, the interesting aspect is the existence, the strength, the direction, and a potential causal nature of the relationship between changed lines-of-code and defects. Because of the positive slope (0.45), we may now conclude that commits with more changed lines are more dangerous, as this increases the probability of defects.

In such a trivial case, the method is often not further questioned. We trust in the fact that we apply the logistic regression correctly, that the software works, that we have enough data, and that our interpretation is valid, although we have never seen the real relationship, and the real values of the unobserved variables and compared them with our identified values −3.28 and 0.45.

We may implement additional checks by cross-validation. However, cross-validation is an important remedy against overfitting, it does not resolve the conceptual issue of not knowing the real intercept and slope.

4.2 Replacing the Empirical Scenario by a Simulation

We will now replace the real scenario with a simulation. In particular, this means that we replace some (or all) observed and unobserved variables with artificial counterparts, carefully simulated according to assumptions about the real empirical scenario. We will capture this in an operational simulation artifact.

We provide the simulation code in Listing 2 and as an online resource. Comments help to distinguish between scalars, vectors, and matrices. All simulation code in the paper is written in standard R, not using advanced libraries.

```r
# Kept observed variables.
N ← N # Number of commits.
X ← X # (vector) Keep the original variable X.

# Substituted unobserved variables.
alpha ← −3.0
beta ← 0.4
prob ← 1 / (1 + exp(−(alpha + beta ∗ X))) # (vector) Assumption
     of the logistic regression model on the relation between X
    and Y.

# Substituted observed variable Y.
Y ← rbinom(N, size = 1, prob = prob) # (vector) Assumption on
    the output distribution.
```

Listing 2 The simulation artifact: R code that replaces data used by the original method (X and Y) with artificial data.

The first three lines of Listing 2 denote that we keep the original variable X and the number of observations N to stick close to the original data. However, this is not necessary. We could also rely on fully artificial data.

Next, the code assigns the unobserved intercept and slope variable in terms of *alpha* and *beta*. This is a crucial part of the simulation code, where the unobserved variables that correspond to our results, get replaced by artificial variables. We *invent* both values. We use slightly different values, compared to those that are the result of the original study run, to point out the fictional character of this replacement. We now use −3.0 for the intercept and 0.4 for the slope. We will test alternatives in the next section more systematically.

All the remaining code (line 8 and 11) directly follows the basic assumptions of a logistic regression on the relationship between X and Y (just run in reverse). MSR/ESE studies (e.g. [46,24,53,85,43,55,88,87,78]) make this assumption implicitly when using a logistic regression model; authors are aware of this definition.

The code specifies the exact probability *prob* of facing a defect as a (logistic) function of *alpha*, *beta* and X. This vector of exact probabilities is another intermediate unknown that can never be observed. In reality, we can just observe the final defect classification Y. We simulate defects Y by a stochastic function producing uncertainty, a binomial distribution (*rbinom*) with one trial and the probability set to the vector *prob*. It is a simple random number generator[2].

4.3 Running the Original Method on the Simulated Data

We can now process the artificial data (X and Y) as if it were the real data, using the original method from Listing 1 with one important difference: *We know alpha, beta, and prob.*

4.3.1 Correspondence of the Results

The original method identifies the unobserved variables (the results we like to interpret) in the simulated run to be *alpha* ≈ -2.97 and *beta* ≈ 0.39. We can objectively support the validity of the method under the simulation because the results *alpha* and *beta* are very close to those values set in the simulation (-3.0 and 0.4). However, there are two remaining questions on such correspondence.

4.3.2 Uncertainty of the Results

First, it is unclear why the method does not exactly meet the artificial *alpha* and *beta*. It is because of the stochastic relationship used between *prob* and Y (Listing 2, line 11). The method only observes Y, but the corresponding *prob* is not known for sure; hence, also the identified values for *alpha* and *beta* are uncertain.

If repeating the simulation, using different initial seeds for the used random number generators, the identified *alpha* and *beta* distribute as depicted in Fig. 1. This shows that on average, the identified alpha and beta variables are excellent, but not totally exact. This is a reason many studies report on confidence intervals and p-values.

[2] In R, random number generators are vertorized and start with a letter *r* followed by an abbreviation for the distribution family (we will see *rbinom*, *rnorm* and *rpoisson*).

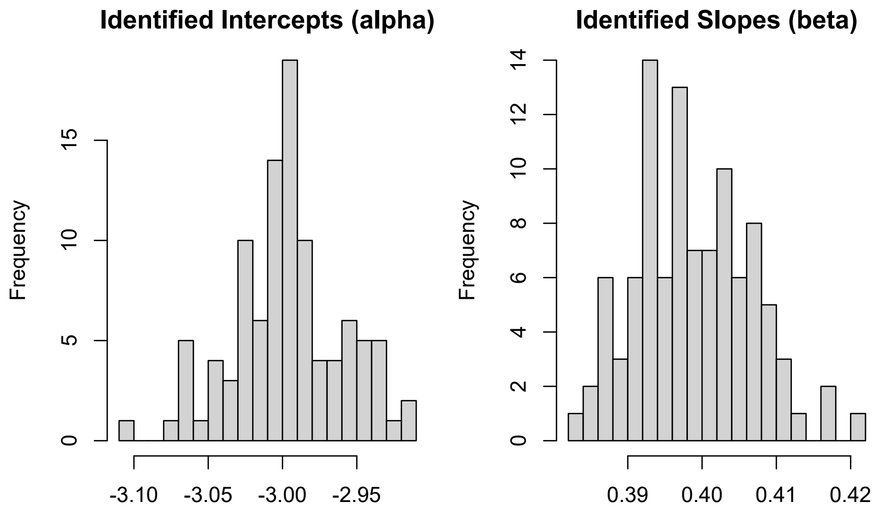

Fig. 1 Identified unobserved variables intercept (alpha) and slope (beta) by the method in 100 repeated simulation runs.

4.3.3 Parametrized Tests

Second, it is not clear what happens if we use different artificial values for *alpha* and *beta*. The simulation can examine this by going through multiple combinations of both, checking the impact of the method on its results.

The results for a grid of 30×30 combinations is shown in Fig. 2 (left). The plot illustrates the impact of a particular combination of our assumptions, operationally given by the artificial alpha and beta, on the difference between the identified and the artificial beta. Such a difference can be interpreted as an error in the identification done by the method. Fig. 2 (left) shows this as a scatter plot. The assumptions in terms of artificial alpha and beta can be read from the two axis, and the error is depicted by the red color of a dot. When alpha and beta are both high or low, identification struggles, it produces a high error (red dots). On the diagonal, e.g., when alpha is high and beta is low, errors are low (white dots). We do not show the error in the identification of alpha, but a plot can be derived in analogy.

The phenomenon is well-known. If counting the number of defects in the artificial Y variable and relating it to this error, we see that the identification only struggles on extremely high or low defect counts (see right of Fig. 2).

We know this under the name *class-imbalance* [75] or in terms of instructions on events-per-variable (EPV) [58]. Our simulation does detailed suggestions, i.e., that if we have more than approximately 400 defects in the data set, the method should be safe from the threat of too low defect counts. In our real data set, we have 5771 defects, so we can deny the plausibility of this simulated scenario, and thereby also the impact on the error in results.

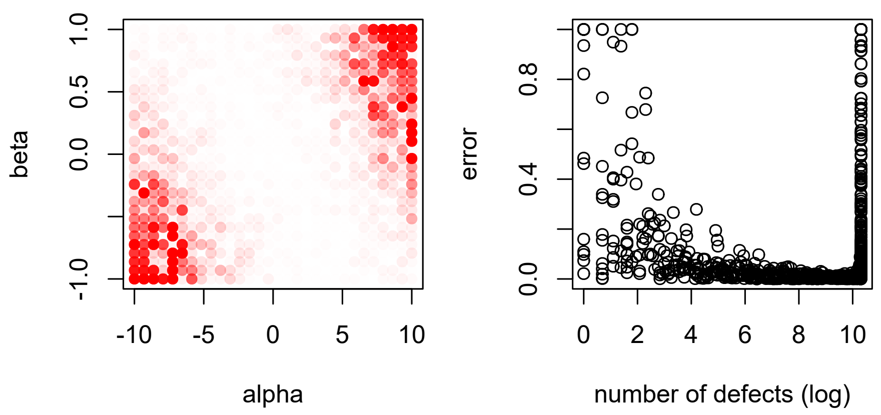

Fig. 2 Dots are repeated simulation runs. **Left:** Artificial alpha and beta and the error in the identification, depicted as red dots (red increases with error). **Right:** The total sum of defects in the artificial variable Y (log scale) and the relation to the error in the identification.

5 Meta-Validation

To provide a validation on our part, we apply simulation-based testing to six real scenarios that are studied in MSR and ESE. We show what benefit can be expected by simulations as additional validation artifacts. We show that we can either i) support validity, ii) threaten validity, or iii) show the invalidity of the used methods and results by simulated scenarios.

5.1 Section Structure

The following six sections will provide this meta-validation of simulation-based testing (Sec. 6-11). These sections follow the structure:

– **Research Question**: We instantiate our meta research questions for a particular empirical study. We discuss how we will answer the research questions in the concrete case by a simulation artifact.
– **Original Method**: We describe the original method of a study (or of a type of study) that may or may not have problems with validity.
– **Simulated Scenarios**: We describe how we replace the assumptions about an empirical scenario, subject to the study, by one or more simulations of the scenario. The simulation artifacts are provided online.
– **Rating Results**: We discuss the results provided by the original method when applied to the simulated scenarios.
– **Conclusions on Validity**: We conclude on the validity of the original method conditional on the plausibility of the simulations.
– **Revision (Optional)** We show how to improve the original method so that it produces correct results for the simulated scenarios.

5.2 Summary

We give a short sketch of the **main findings and argumentation patterns** for the cases examined in this meta-validation.

– **Dependent Observation (Sec. 6, Invalidity)**: We show that a used method computes results with incorrect confidence intervals in a simulated scenario where observations are dependent. Especially in MSR, observations are often sampled from the same repositories which may introduce dependency. This is a plausible scenario, and thereby may invalidate parts of the original study.
– **Prediction or Causation (Sec. 7, Supports Validity)**: We show that when using a logistic regression method, the results can also be interpreted causally if the underlying simulated scenario is sufficiently basic. The simulation supports the validity of the original study. We hint at limits and potential improvements of the plausibility of the proposed simulation. We explain how extending simulations may also lead to simulations that threaten claims on causation.
– **Control of Variables (Sec. 8, Invalidity)**: We show that under a plausible simulation, a method to prove the relevance of a novel metric, produces trivial and misleading results. This renders the method as invalid. We show how to revise the method.
– **Correlated Variables (Sec. 9, Invalidity)**: We show that under a series of plausible simulations, a new method always fails to produce correct results. No simulated scenario shows an improvement of the results. In some scenarios, the quality of results even decreases. This renders the proposed method as invalid.
– **Distribution Types (Sec. 10, Supports Validity)**: We show that a certain type of mismatch between method and simulated scenario does not influence a certain interpretation of the results. The interpretation of the original study is still valid. We support the validity.
– **Experimental Research (Sec. 11, Supports Validity)**: We simulate a random experiment and show the impact of some alternative methods on the interpretation of results. We support the validity of such studies.

5.3 Acknowledgement

We want to acknowledge the original work of the authors in the studies, subject to the following illustrations. All studies have been selected because of their originality. However, we believe that this meta-validation of simulation-based testing is not credible on unpublished examples. We need to continue on real studies.

6 Dependent Observation (Case 1)

After laying the foundations, we start with the first case for our meta-validation. We examine a study by Alali et al. [2]. The work examines the 'typical commit'.

6.1 Research Question

The original paper examines a research question, simplified as follows:

– **RQ***: What is the typical commit in a software repository?

In this paper, we are interested in operationalized statements about the validity of the way the study attempts to answer this research question. We instantiate our meta research questions for the study accordingly:

– **RQ*****1**: What assumptions of this study on repositories, commits, and properties of the typical commit can be operationalized?
– **RQ*****2**: What is the impact of such assumptions on the result of the study regarding the properties of the typical commit?

We believe that most results of Alali et al. are valid. However, the study runs into a specific problem, that we believe, is characteristic for the analysis of repository data. The results include statements on uncertainty, while the method ignores a potential structure in the analyzed sample of commits.

We will provide simulation artifacts that operationalize these assumptions, and then show that they invalidate confidence intervals for the results.

6.2 Original Method

The original method of the study mines the history of nine open-source software systems. It reports on the number of lines, files, and hunks changed by commits. Further, the correlation between the variables is described.

Examining the 'typical commit' indicates that the resulting statements are not necessarily specific to the nine observed repositories. Statements may also hold for other repositories, those that the authors had in mind, but did not examine due to computational overhead. Such sampling is standard practice in MSR [22].

A method needs to produce results with a notion of uncertainty because variables identified using the nine repositories might not exactly correspond to values computed including the unobserved repositories. A method can give such statements in terms of confidence estimates.

The method of Alali et al. report on confidence, i.e., p-values for the correlation between the variables. In the remainder, we will leverage instead the notion of confidence intervals, since p-values are less intuitive for most readers. Problems with p-values are the same and can be shown the same way.

We assume that the original method computes p-values according to standard practice because the paper does not report on counteractions against dependent observations. In our reproduction of the original method, we also stick to standard practice[3].

6.3 Simulated Scenario A

The data of [2] is not available, so we simulate all the data of the empirical scenario. We simplify the original research and examine the correlation between just two variables $X1$ and $X2$.

Listing 3 shows simulated scenario A, which simulates two correlating variables in a structured sampling process.

```
X1all ← NULL # X1 collected over repositories.
X2all ← NULL # X2 collected over repositories.
N ← 100 # Number of repositories.

for (repo in 1:N) {
    rho ← 0.2 # Rho is the same in each repository.
    M ← 100 # Number of commits in each repository.
    # Simulating X1 and X2 for a repository.
    X1 ← rnorm(M, mean = 0, sd = 1)
    X2 ← rnorm(M, mean = 0, sd = 1)
    # Producing the correlation (rho).
    sigma ← matrix(c(1, rho, rho, 1), 2, 2)
    X ← cbind(X1, X2) %*% chol(sigma)
    # Collecting X1 and X2.
    X1all ← c(X1all, X[, 1])
    X2all ← c(X2all, X[, 2])
}
```

Listing 3 Simulating two correlating variables X1 and X2 for 100 repositories, each with 100 commits.

The code produces data for N=100 repositories, each with M=100 commits. Within a repository (inside the loop), we simulated two variables $X1$ and $X2$ for M commits following a normal distribution (*rnorm*). A correlation between $X1$ and $X2$ is simulated using the *Cholesky decomposition* with $rho = 0.2$.

Finally, we simulate the random sampling step (see the online resources for the complete code). We randomly decide on 91 repositories where we consider $X1$ and $X2$ as unobserved variables, and 9 repositories where we consider $X1$ and $X2$ as observed variables.

6.4 Simulated Scenario B

The assumption of dependent observations is added by a small modification to simulated scenario A.

```
for (repo in 1:100) {
```

[3] All our reproductions of other papers are fully available online to guarantee the reproduction of this paper.

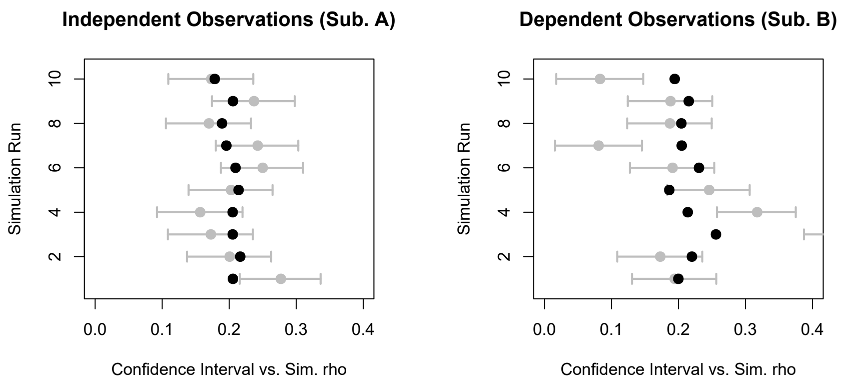

Fig. 3 Ten simulation runs showing confidence intervals for rho, identified based on 9 repositories (gray line), and rho computed on all 100 repositories (black dot). We distinguish between independent (left) and dependent observations (right) produced according to simulated scenarios A and B.

```
  rho ← rnorm(n = 1, mean = 0.2, sd = 0.23) # A repository-
      specific variation in the correlation.
  # ...
}
```

Listing 4 Simulating a repository-specific variation in the correlation rho.

In simulated scenario B, the correlation *rho* is sightly different for each repository, but on average 0.2, since we draw it from a normal distribution with mean 0.2. The standard deviation (*sd*) decides on the severity of the threat, we set it to 0.23, but other configurations can be explored in the same fashion as done in Sec. 4.3.3. Some values that cannot serve as a correlation *rho* need to be filtered out. The rest of the simulated scenario is the same as in the previous simulated scenario A.

6.5 Rating Results

The original method of Alali et al. [2], used to report on the uncertainty of the results on the typical correlation, works under simulated scenario A, but not under simulated scenario B. Repeating ten simulation runs suffice to show this.

We compare the correlation (rho) computed on all 100 repositories, including the unobserved, with the confidence interval identified according to the original method on the 9 observed repositories. Since we report on confidence intervals of 95%, 0.5 out of 10 simulation runs are allowed to fail. When having a look at Fig. 3, we see that the number of failing confidence intervals is different in both simulated scenarios:

– Under *simulated scenario A* (left), almost all confidence intervals include the correct correlation.

– Under *simulated scenario B* (right), we see that in 4 out of 10 simulation runs, the method fails to include the correct correlation in the confidence interval. The small variation regarding repositories has a big impact for statements on uncertainty. The chance that our confidence interval includes the correct correlation, is almost comparable to flipping a coin.

6.6 Conclusions on Validity

We have operationally defined the assumption of dependency in observations, and then we have shown that it may invalidate the results produced by the original method regarding uncertainty.

Indicators for the plausibility of simulated scenario B, compared to simulated scenario A, can be found in every analysis that proves that repositories are different (e.g., see the highly different slopes in regression models computed on different repositories in [24,46]). We may even check such assumption on dependency on the data (which we don't have in this case).

The impact of our assumptions about the number of repositories N, and the standard deviation sd can be explored by a parameterization of the simulation. This may uncover that more repositories and lower standard deviations decrease the error in computed confidence intervals.

6.7 Revision

General advice on improving the method can be guided by informing the model of the structured sampling process. In the simulation above, we simplified. We recommend casting the problem as a linear model, where slopes correspond to the correlation we like to examine. The linear model can be evolved to a *hierarchical linear model*, where the repository-specific impact on the correlation can be identified as a random effect. The random effect may follow a normal distribution and the standard deviation is thereby identified, too.

We recommend testing such models in simulations. We recommend work on hierarchical/multilevel/mixed-effect models for guidance [28].

7 Prediction or Causation (Case 2)

The next case in our meta-validation examines an experience report by Tantithamthavorn et al. in [76]. The report discusses challenges and actionable guidelines when using methods for defect modelling.

We selected this work because of its progressive understanding of defect modeling to be more than just defect prediction. We expect this report to have a big impact on our community. However, this work fears to name one actual challenge, which is *examining causation*. We will show how to sharpen this understanding by simulations.

7.1 Research Question

The original paper examines a research question, simplified as follows:

– **RQ***: What are the challenges and actionable guidelines when using methods from defect modeling?

We are interested in operationalized statements that complement such guidelines. We instantiate our meta research questions for the study accordingly:

– **RQ*****1**: What assumptions of this study on causes for defects in commits can be operationalized?
– **RQ*****2**: What is the impact of such assumptions on the result of the study when talking about causation?

We will accomplish the original study by simulation artifacts that support the validity of claims on causation. This sharpens the guidelines of Tantithamthavorn et al.

7.2 Original Method

Practitioners may use an analytical defect model not just to predict defects, but also to answer questions, like *'whether complex code increases project risk'* (copy from [76]). Project risk refers to defects.

Within this setting, the term *increases* can be understood in different ways, and the original study is unclear on this:

– Do we wish to compare or relate complex code with project risk? If so, we can use this insight to (mentally) predict one variable using the other.
– Do we wish to know if modifying the complexity of code causes the project risk to change? This insight is efficient to guide our decisions. It corresponds to what most people have in mind when hearing the statement above.

Tantithamthavorn et al. and many other researches in MSR/ESE fear to claim causation (second statement). This is not surprising. Causation is difficult to examine, definite claims are impossible, even in randomized experiments.

On the other hand, most of the operational decisions should be driven by causal relationships. Their examination is the crucial point of modern empirical research (see the preface of [39]). Even more dramatically, statements on the relation between *complex code* and *project risk* are intrinsically hard to understand if not aiming at causation. Understanding may get harder if a defect model grows, without having causation in mind, including more variables.

Simulated scenarios are a useful extension to the original paper, since they can make clear when claims on causation are valid.

7.3 Simulated Scenario C

Following the definition of Imbens et al. [39] (page 6), a claim on causation can unambiguously be given by comparing the potential outcomes of different treatments in exactly the same situation. Reality does not allow observing more than one potential outcome in the same situation, but a simulation-based test allows synthesizing both.

```
X ← rnorm(N) # Synthetic variable X.
# Producing two potential  probabilities.
prob_pot1 ← 1 / (1 + exp(−(alpha + beta * X)))
prob_pot2 ← 1 / (1 + exp(−(alpha + beta * (X + 1))))
# Corresponding potential defects.
Y_pot1 ← rbinom(N, size = 1, prob = prob_pot1)
Y_pot2 ← rbinom(N, size = 1, prob = prob_pot2)
```

Listing 5 Simulating causation.

Listing 5 implements such simulated scenario, using 'treatment' X, but also continuing with the modified $X + 1$, in exactly the same situation. The relationships are the same as in the basics on logistic regression provided in Sec. 4. In the simulation, we get two potential outcomes. According to the definition, the difference between Y_{pot1} and Y_{pot2} reflects the causal relationship between X and Y.

7.4 Rating Results

According to our previous strategy of hiding artificial but unobserved variables from the original method, we run the original method just using one of the potential outcomes, keeping the other hidden. However, the simulated scenario shows that we can identify the causal relationship by the logistic regression method.

Multiple runs of the simulation, with different artificial values for *beta*, show that there is a clear correspondence between the identified *beta* using only the observed variables, and the difference between both potential outcomes also including unobserved variables (see Fig. 4).

7.5 Conclusions on Validity

We have provided operational assumptions about a scenario on causes for defects, and did show the impact of such a scenario for claims on causation in the context of the method examined in the original study.

Our simulation artifacts provide a clean notion of assumptions and the implications on the validity of claims on causation. Our simulation artifacts can be used to sharpen the guide by Tantithamthavorn et al., being more precise on what the *interpretation of defect models* is.

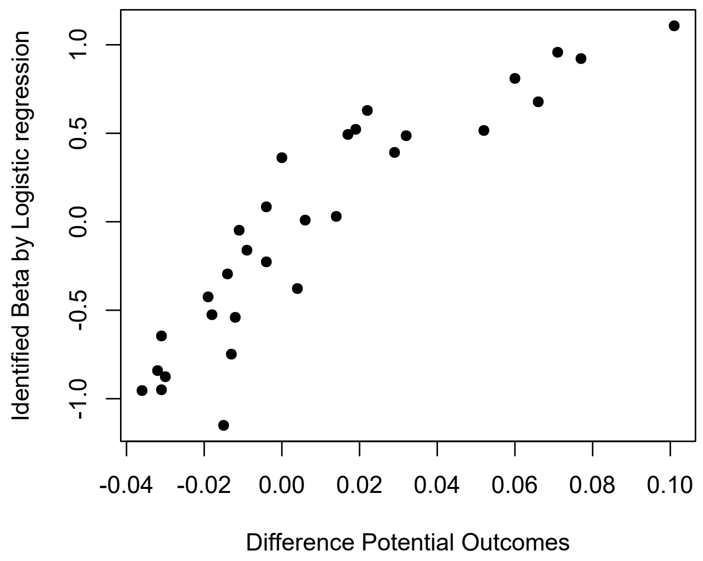

Fig. 4 Identifying causation under strong assumptions (simulated scenario C).

However, simulated scenario C may not be considered as the most plausible scenario. In fact, it is very limited. It assumes the effect of X (*beta*) to be stable across all our observations; no dependency between observations; X is just a random variable, not influenced by anything else. The simulation may be evolved based on these insights, and the results of methods can be checked if they are valid claims on causation. This may render some threats to claims on causation that we prefer to make explicit.

8 Control of Variables (Case 3)

In [81], the authors propose a new metric to reflect a developer's ability to correctly understand the code.

8.1 Research Question

The original paper examines a research question, simplified as follows:

– **RQ***: What is a metric that reflects a developer's ability to correctly understand code?

We are interested in operationalized statements about the validity of the method used to prove the relevance of the new metric. We instantiate our meta research questions for the study accordingly:

– **RQ*****1**: What assumptions of this study on a new metric regarding the developer's ability to correctly understand code can be operationalized?

– **RQ*2**: What is the impact of such assumptions on the result of the study, which is aimed at correctly proving relevance?

We show simulations for which the used method always proves relevance. The simulation uncovers that the used method is incapable of producing negative results. This renders the method as invalid.

8.2 Original Method

The original method follows the standard in defect modeling as described in Sec. 4. It relates the novel experience metric E to defects Y. Defects are used as a proxy for a developer's ability to correctly understand the code. The method is slightly adapted because the authors show the relation using a basic statistic test, rather than a logistic regression. The proof of the existence or non-existence of such a relationship can be considered as the result of the method.

The novel experience metric is defined using the cosine similarity between *files* touched by a commit, and the lexical background of the contributing developer (*back*). Defects may increase, as similarity decreases. This lexical background is composed out of all modifications done by the developer in the past. We refer to the original work for details on how to compute the exact metric.

Our first intuition reading the original paper was that the technical computation using the cosine may accidentally influence the results of the method. We expected a potential correlation between the cosine and the file size.

This is a serious threat if we aim to interpret the fact that it is experience, and not the file size, that influences defects. We know from works in MSR/ESE, e.g., [52,24,46], and from Sec. 4, that the file size (or lines-of-code) strongly relates to defects. Such *confounding effect* of the new experience metric over variable *files size* would not be surprising and not much of interest. The new metric would just be an overcomplicated proxy for file size.

8.3 Simulated Scenario D

This means that the method used in the study needs to protect against such false claims on the existence of a relationship. We can write a simulation-based test to check such a protection. This scenario simulates a part of the metric computation to produce variable X and E using the cosine.

```
N ← 8000
X ← NULL
E ← NULL
for(n in 1:N){
  nTerms ← 200
  # Generate two random term vectors.
  back ← rpois(n = nTerms, lambda = 5.0)
  file ← rpois(n = nTerms, lambda = 0.1)
  # Compute the similarity defining experience.
  E ← c(E, cosine(back, file))
  # Size of the file.
  X ← c(X, log(sum(file) + 1))
}

alpha ← −3.0
beta ← 0.4
prob ← 1 / (1 + exp(−(alpha + beta * X)))

# Substituted observed variable Y.
Y ← rbinom(N, size = 1, prob = prob)
```

Listing 6 Simulating the computation of experience.

Listing 6 produces N artificial file and background pairs using vectors sampled from a Poisson distribution (stochastic function). The number of terms ($nTerms$) in the VSM is set to 200. The Poisson distribution works well for simulating term vectors because it only produces discrete positive vector entries. The average term frequency is set by the *lambda* parameters. The code collects the new experience metric E defined by the cosine, but also the corresponding file size X as the stated-log transformed sum of its terms (as often assumed in defect modeling). Alternatives can easily be explored using the online material. According to our knowledge, they do not influence our conclusions under simulated scenario D.

Finally, the code simulates the probability *prob* and the defects Y, as we have done in the previous sections. In this simulated scenario, we assume that there is no effect of E. This is clear from the code because E is not input to the function producing *prob*.

8.4 Rating Results

The original method does not manage to handle the simulated scenario D correctly. The Mann-Whitney test used in the original study rejects the null-hypothesis (which it should not do according to our assumptions of the simulation) in approximately 45 out of 100 simulation runs at a confidence level of 95%. That means that in 45 runs, it incorrectly detects the existence of a relationship. Doing it in 5 runs would correspond to the confidence level. Hence, the result is invalid.

The reason gets clear when looking at the raw data in Fig. 5, showing a strong positive correlation between X (file size) and E (novel experience metric). The very simplistic Mann-Whitney test accidentally attributes the effect of X (file size) to E (novel experience metric). As expected, it is caused by the technical computation of the cosine.

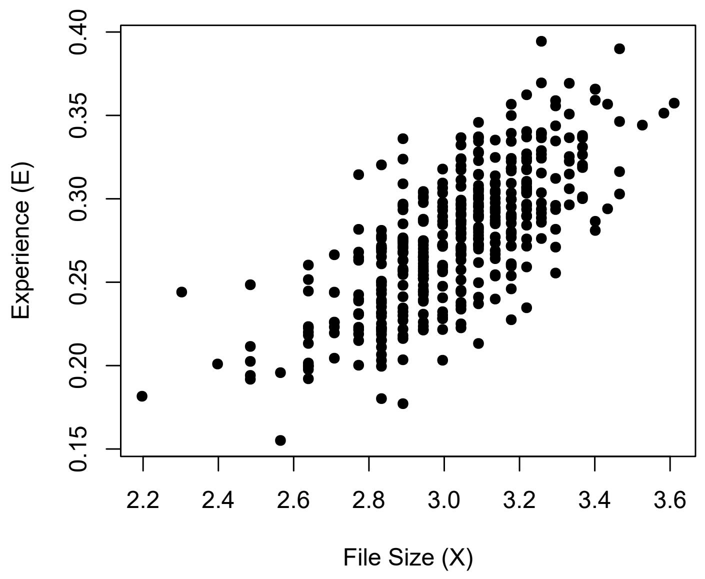

Fig. 5 Relation between file size X and experience E computed by the cosine in a single simulation run.

8.5 Conclusions on Validity

The previous simulated scenario operationally shows how a method is technically not suited to produce valid results. We show this in an important case for the researchers, proving their new contribution to be wrong.

Simulated Scenario D is plausible enough in that we can demand a method to resolve it. Claiming that a developer-specific factor relates to defects, while it is essentially just the file size, does not provide a novel insight. This holds for examining causation as well as for prediction.

8.6 Revision

Indeed, there are ways to improve the original method. We appreciate that the original data is provided by the authors so that we can rearrange the statistical checks.

We convert the original test (Mann-Whitney) into a logistic regression model, which allows the *control of variables.* The control for the variable *file size* is the mandatory step that resolved the threat to a method. It blocks the *confounding effect* of the new experience metric over file size, in that we get the *direct effect* of experience that we are interested in. Such a model indeed succeeds in *not detecting* an effect on the artificial data produced by simulated scenario D. See the online resources extending the simulation for this insight.

We now apply the method (with and without control) to the real data and revise the original study. The relevant information on the logistic regression

Table 1 A model with and without control applied to the real data, showing the effect strength, the usual significance encoding, and AIC.

Variables	Original	Control
Experience	1.42***	0.12
File Size		0.01***
AIC	19968	18282

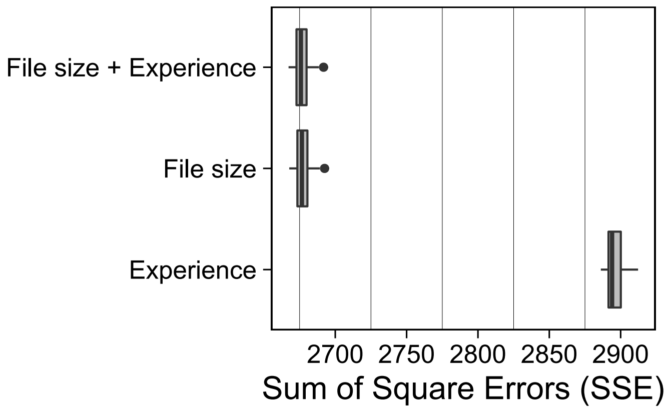

Fig. 6 The performance impact of the new experience metric evaluated in cross-validation on the original data.

models can be found in Table 1. For further details, we refer to the reproduction code.

- We start with a reformulation of the original statistic test, describing the effect of *experience* on defects as a basic logistic regression. This model is called **Original** and reports that, comparable to the original work, the effect of the novel experience on defects is positive and highly significant.
- The second model, called **Control**, adds the file size as a control metric. The effect of the experience drops by a factor of ten (from 1.42 to 0.12). The effect stops being significant. This conforms to our initial intuition that the study just proves a *confounding effect* over file size and not the importance of a new metric.

Even when evaluating the new metric in prediction, adding it next to file size, as a new predictor, does not lead to an improvement. Cross-validation results for prediction on the original data can be found in Fig. 6.

A simulation would have discovered this problem with the computation of the cosine and the invalid method early.

9 Correlated Variables (Case 4)

The next study that we examine is by Jiarpakdee et al. [43]. The study aims at showing a general improvement to methods interpreting defect models with correlated variables. The authors motivate the practice of removing variables based on correlation or VIF thresholds (VIF [21]).

9.1 Research Question

The original paper examines a research question, simplified as follows:

– **RQ***: Can we improve the results of a method for defect modeling by a threshold-based removal of correlated variables?

We are interested in operationalized statements about the validity of this improvement of the method. We instantiate our meta research questions for the study accordingly:

– **RQ*****1**: What assumptions of this study on the typical interpretations of results and strong correlation between variables can be operationalized?
– **RQ*****2**: What is the impact of such assumptions on the result of the study, which applies the new proposed variation of the method?

We will show that the proposed improvement to the method does not improve the results in a series of plausible simulated scenarios. In some scenarios, it even causes additional problems. We are not aware of a simulation that supports the validity in terms of showing that the improvement to the method provides better results. The new method can be considered as invalid.

9.2 Original Method

In a nutshell, the original study claims that model interpretation methods, that are run on data sets where correlated variables have been removed by thresholds, provide better results. Strongly simplified, Jiarpakdee et al. support this claim by showing that both methods provide different results. We refer to the original paper for the details on how the authors assume this argumentation to work.

9.3 Simulated Scenario E

We will simulate a series of scenarios with strong correlation, and check if a method that removes correlated variables leads to better results.

In the following, we focus on a fully artificial data set with the variables X, Z, W and the resulting defects Y. We will simulate a causal pattern where W is a confounding variable for the relation between X and Y, while Z and W may get strongly correlated depending on a simulation parameter.

```
# Alternative standard deviation of Z produces different
    correlation strength between Z and W.
for (sdZ in seq(0, 1, length.out = 40)) {
```

```
# Stochastic relationships between W, X and Z.
W ← rnorm(N)
X ← rnorm(N, mean = -W, sd = 1)
Z ← rnorm(N, mean = W, sd = sdZ)

prob ← 1 / (1 + exp(-(W + X)))
Y ← rbinom(N, size = 1, prob = prob)
# ...
```

Listing 7 Simulating relationships, producing correlated variables and defects.

In this simulation, no variable influences W. Variable W influences X and Z. Both variables X and Z are given as stochastic functions following a normal distribution, with the mean set to be W or $-W$. Further, the stochastic function simulating Z is configured using different values for the standard deviation. This means that with decreasing standard deviation sdZ, the variable Z becomes a perfect copy of W.

The final defects Y are produced as a stochastic function of X and W. The variable Z is not related to defects.

9.4 Rating Results

We want to cover two 'interpretations': First, we are interested in the effect of Z and W. Both effects are unobserved variables that we set in the simulation. When running the logistic regression including all variables, the identified effect of Z and W is correct until the correlation reaches a threshold of about 0.9 (see Fig. 7).

The improved method of Jiarpakdee et al. should resolve this. We expect it to drop Z, since the variable Z is not related to defects according to the simulation. However, this insight cannot be made by using the correlation or VIF values, since both are symmetric. A selection would be comparable to flipping a coin.

In a second 'interpretation', we are interested in the effect of X. We show the identified effect of X in models, including variables Z, W, none and both in Fig. 8. The model not including W and Z fails as it runs into the problem with confounding. The model, including all variables, succeeds like the model including W. The model using Z fails up to the point that the correlation gets so high in that Z can be used as a replacement for W.

We see that neither dropping W nor Z makes sense, as the model including both does a perfect identification of the effect of X. If we decide between W or Z, we risk to drop the wrong variable.

9.5 Conclusions on Validity

The previous simulated scenario operationally shows how a proposed improvement to a method does not improve results.

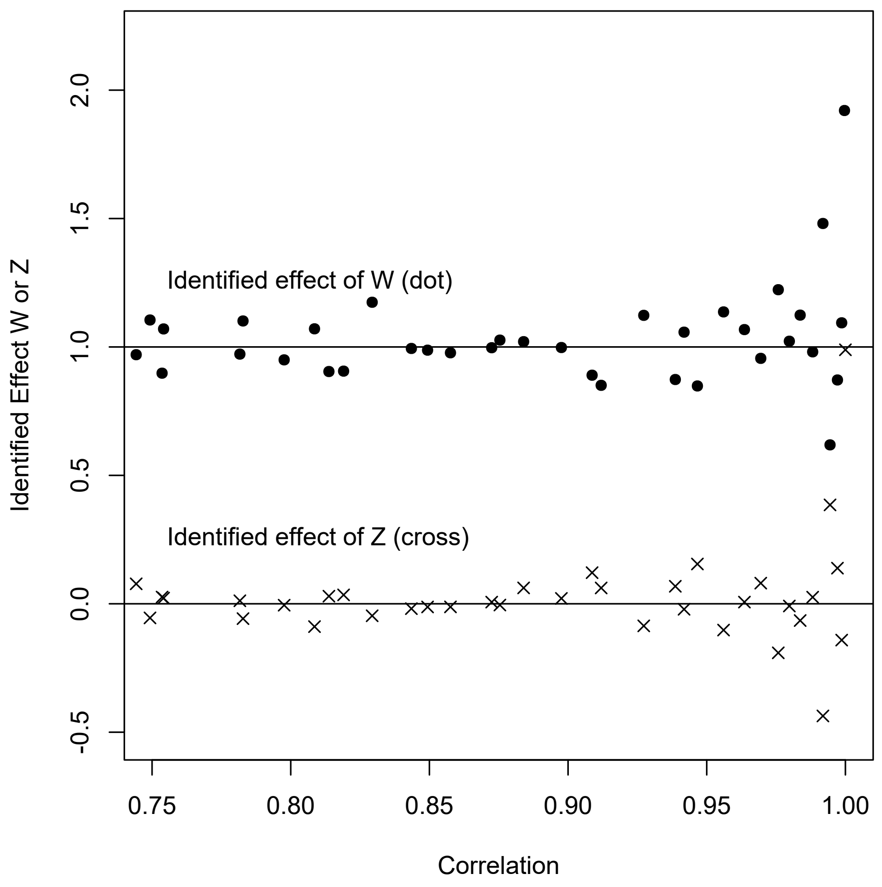

Fig. 7 The identified effect of W and Z (which should be 1.0 and 0.0 respectively) under different correlation.

The simulated scenarios that we show include a very plausible causal pattern. We can expect a study that recommends dropping correlated variables, with the aim of improving the interpretation of defect models, to exactly resolve such issues. We do not expect the results of a method to get worse.

We are not aware of any simulated scenario where the recommended practice brings real benefits for results. Even for prediction, dropping correlated variables just decreases performance (corresponding simulations are straightforward to implement). This conforms to recent statistic guidelines (e.g., see [50], page 169).

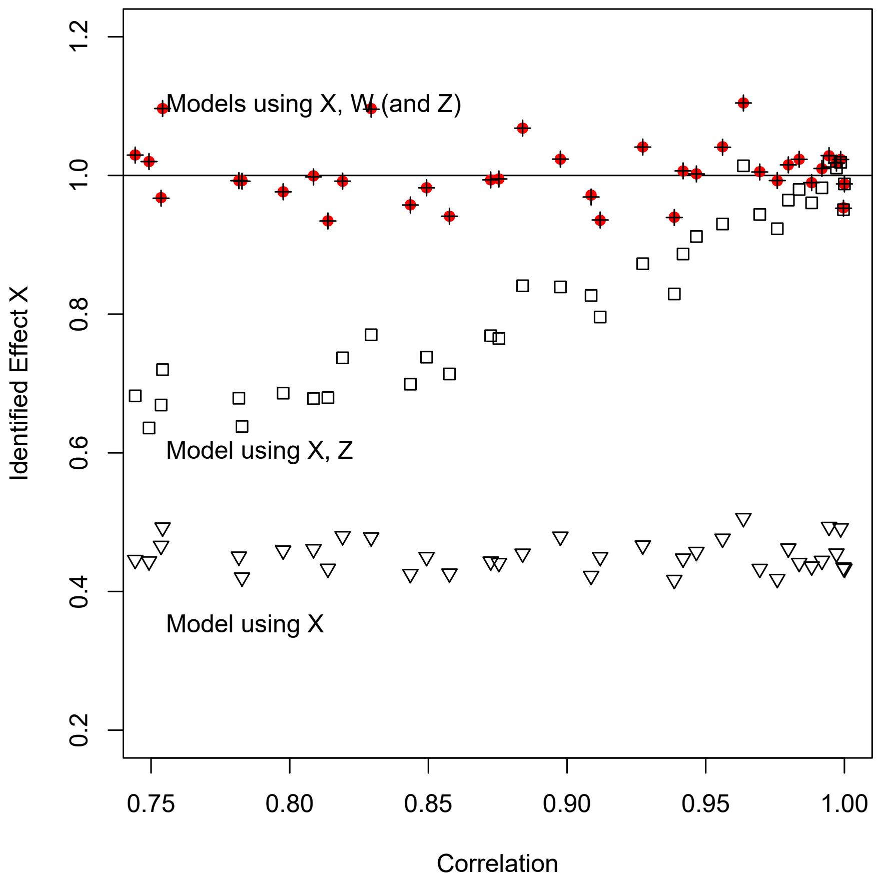

Fig. 8 Models in simulation runs and the identified effect of X (which should be 1.0) under different correlation and different control variables.

10 Distribution Types (Case 5)

The next example will focus on one of our previous works that is presented in [68]. We will illustrate the implications of choosing the wrong output distribution for a regression model. It is a mistake that we did in [68].

10.1 Research Question

The original paper examines a research question, simplified as follows:

- **RQ***: What are the characteristic differences of repositories using SPARQL and Cypher queries?

We are interested in operationalized statements about the validity of resulting statements. We instantiate our meta research questions for the study accordingly:

– **RQ*1**: What assumptions of this study on the differences between repositories using SPARQL and Cypher can be operationalized?
– **RQ*2**: What is the impact of such assumptions on the result of the study, regarding statements about the difference?

In this case, we show how parts of the results of an obviously wrong method can still be valid under plausible assumptions.

10.2 Original Method

We will only focus on the parts of the method that run a regression model in this section. Other elements of the method should also be reviewed within the framework of the new strategy.

Originally, we used the regression model to better understand the decision of a software project between two alternative graph query languages, SPARQL, and Cypher. The model tries to associate the decision between the two languages with different properties of a project, i.e., with its age (*created_ days_ ago*), the popularity (*stargazer_ count*), the number of active developers working on graph queries (*active_ developers*), and the active files that include graph queries (*active_files*). In essence, the study results suggest that SPARQL is preferred by projects that are older, more popular, more active, and that have more files including graph queries.

In this section, we particularly focus on the output distribution of such a model, which needs to reflect the decision between SPARQL and Cypher. It thereby conforms to the binomial distribution with a single trial, where one language (e.g., SPARQL) is represented by 1 and the other by 0. The model thereby aligns with our previous discussion of defects.

Table 2 Parameters identified by different models that describe the decision for SPARQL: The table shows the difference in identified parameters when using a linear or a logistic regression model (normal vs. binomial output distribution).

Output Distribution Type	Normal	Binomial	Normal	Binomial
Variables	Parameters		Scaled Parameters	
created_days_ago	0.040	0.187	0.935	0.860
stargazers_count	0.017	0.086	0.392	0.393
active_developers	0.117	0.498	2.731	2.287
active_files	0.060	0.515	1.405	2.365

Due to a lack of experience with such modeling practice at the time, we decided to go for what, we believed, was the more established method. We used the wrong output distribution. We used a linear regression and thereby a normally distributed output. This is clearly a mistake. We provide the results of our original method on the original data, and a method using the improved binomial output distribution type, in Table 2.

The middle part of the table illustrates our mistake. We can see that the identified parameters differ for a model with normal and binomial output. For instance, the variable *created_ days_ ago* has an effect of 0.040 for a normally distributed output, while for a binomial output, we have an effect of 0.187.

However, our interpretation of the parameters, i.e., if a variable has a positive or negative effect on the decision for SPARQL, does not change. For instance, older projects with a higher variable *created_ days_ ago* still seem to prefer SPARQL. This consistent interpretation gets more obvious by scaling the identified parameters of both models, dividing them by their standard deviation. On the right side of Table 2, showing the scaled parameters, we notice that the results of the different models are very close to each other. Hence, if ignoring the scale, our interpretation of them still appears to be almost correct.

10.3 Simulated Scenario F

We will examine this in detail, to show that it is the regular behavior, if making this mistake, and not a fortunate coincidence. We start with producing a fully artificial version of the problem.

We use this showcase as an opportunity to generalize the simulation for a logistic regression, that we have developed so far, to a version with a flexible number of variables M. This simulation applies to defect modeling the same way.

```
N ← 120 # Number of repositories.
M ← 4 # Number of variables to examine.

Xs ← matrix(rnorm(N * M), nrow = N, ncol = M) # Producing a N *
    M matrix of random normally distributed values.
```

Listing 8 Simulating M variables for N repositories.

The previous code produces a matrix of artificial normally distributed variables. It contains N rows for the repositories to examine, and M columns that store M variables for each repository. By adjusting N and M, we can change the characteristics of this simulation.

Next, we need to produce the binary output that reflects the decision for one of the two query languages, based on the variables in matrix *Xs*. In essence, this is the same as previously presented for defects in Listing 2. However, we need to change this code to operate on a matrix with a flexible number of variables.

```
# Producing a vector of M + 1 random betas, including one
    variable for a random intercept.
betas ← rnorm(M + 1) # Random betas.

# Adding a column of ‘ones’ at the left of the matrix, later
    multiplied with the first beta and serves as intercept.
```

```
5  Xs ← cbind(1, Xs)
6
7  # Matrix multiplication ('%*%') of Xs and betas, then applying
       the logistic function.
8  prob ← 1 / (1 + exp(−(Xs %*% betas))) # The probability
       deciding for one of the two languages.
9
10 # Producing a decision (same as in the previous simulations).
11 Y ← rbinom(N, size = 1, prob = prob)
```

Listing 9 Simulating the decision for M variables and N repositories.

Instead of setting *alpha* and *beta* as individual variables in the code (as done in Listing 2), we generalize and use a vector called *betas* (line 2). This vector includes the intercept *alpha* as its first entry. The vector is chosen randomly, since we do not care about the particular effects of variables in the simulation.

As a convenient way to cope with the intercept *alpha*, which is somehow special in not being multiplied with a corresponding variable, the matrix *Xs* is extended by a column of 'ones' on the left (line 5). The matrix multiplication (written $\% * \%$, line 8) then multiplies each variable (including the ones) with the corresponding betas (including the intercept), forms the sum, and thereby produces what is finally feed into the logistic function.

10.4 Rating Results

We can now examine the correspondence of betas identified by models using a binomial and the wrong normal output distribution. Furthermore, we can do this on simulated problems with a flexible number of variables and repositories. We will limit us to M and N as set in the previous code. The parameters can be explored systematically using the online resources.

In Fig. 9, we show six simulation runs that apply the wrong and the correct model. In each gray block, we show a single simulation run, depicting the simulated betas excluding the intercept ($\times$), and the counterparts identified by the logistic ($\bullet$) and linear regression ($\circ$). Simulated and identified variables are scaled to have a standard deviation of 1.0 to make them comparable.

We see that simulated betas ($\times$) are typically different from the identified betas ($\bullet$ and $\circ$). This is not a surprise and reflects the discussion of uncertainty (see Sec. 4.3.2). However, we also notice that the identified betas by both models ($\bullet$ and $\circ$) are close to each other in the six runs, although one of the two models uses a wrong output distribution that does not correspond to the simulation.

10.5 Conclusions on Validity

In essence, this shows that we can trust the identified betas, and interpret them, if ignoring their scale. The distribution type of the output is not relevant for this particular statement that we did in our original work. This is an

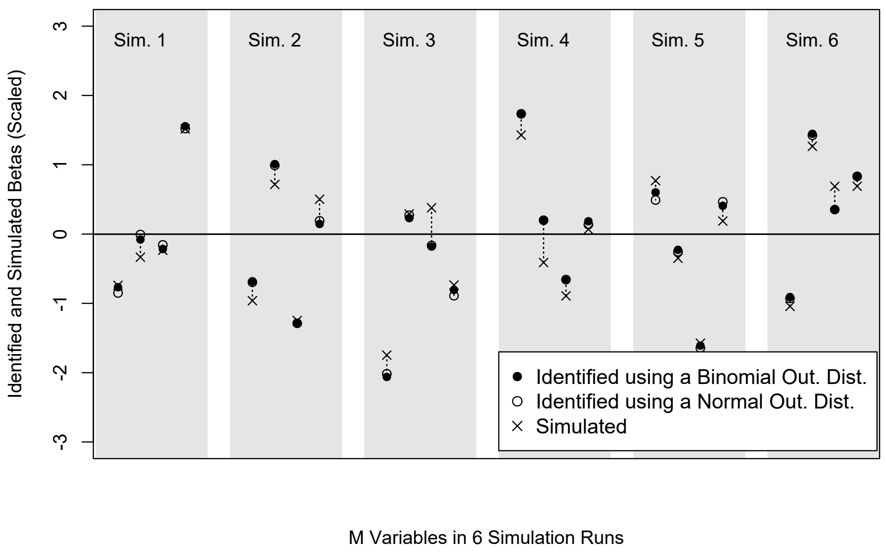

Fig. 9 Running simulations to explore the difference between logistic and linear regression for the identification of betas.

interesting insight as is shows how methods behave if not exactly mirroring the assumptions of the simulation.

However, we want to emphasize that other properties of the wrong model will indeed be wrong. For instance, estimates of uncertainty for the betas will be wrong. We can extend the simulation to show this.

The insights that we presented here conform to general statistic literature [21] (page 483). Extracting this cookbook instruction from literature and transferring it to our MSE/ESE cases, when needed, is not easy. We produced the same insights by a simulation-based test here.

11 Experimental Research (Case 6)

For the last example, we will switch to an experimental method, as an alternative to the analysis of existing data observed from repositories. Experimenting provides major benefits, in particular, in the examination of causal relationships [70]. This is because some assumptions on our scenario can be fixed by the experiment's design. Works on software engineering and ESE research relies on this too [49,44,79,72,3–5,48,45,66].

11.1 Research Question

We ask a simplified research question that may be part of many experimental studies in software engineering. This case does not correspond to a specific study but unifies aspects of the studies listed above. We ask a research question, simplified as follows:

– **RQ***: Does our tool improve software development?

We are interested in operationalized statements about the validity of the experimental results. We instantiate our meta research question for the study accordingly:

– **RQ*****1**: What assumptions of the study on the design of an experiment for testing a tool can be operationalized?
– **RQ*****2**: What is the impact of such assumptions on the result of the study, regarding the improvement of software development by the tool?

In this section, will show two apects on validity:

– We emphasize the strength of experimental research in ESE by operationalized assumptions on the study design. In experimental research, some assumptions are plausible by design, which is an interesting difference to observational studies.
– We show the importance of methods delivering confidence estimates for the interpretation of results, and basic insights of a power analysis.

We believe that even such basic discussion is important because experiments are widespread (see instances like [3–5, 48, 45, 66]). However, they are also run by researchers that encounter this kind of practice for the first time. The simulation of an experiment can help to raise a researcher's confidence in a correctly applied method and corresponding statistic devices, upfront to the actual execution of the experiment.

11.2 Original Method

We stick to a basic experimental design to prove the benefit of a new tool. The method that we follow can be considered as a simplified version of instances found in related work, e.g., examining the effect of artifact formats [3], the role of use cases [4], unit testing techniques [5], performance evaluation of software architectures [48], textual vs. graphical software design descriptions [45], or run-time configuration frameworks [66].

We are doing a randomized experiment with 20 subjects. We start with the selection of the 20 subjects. Ideally, the subjects conducting our experiment are representative of a population. What this means is hard to formalize, and best described by the process on how the subjects are selected. We will ignore aspects of sampling.

The experimenters then randomly assign each subject to the *treatment* or *control* group. A subject may use our new tool as treatment, or an established (or no) tool as control. We measure the outcome of the experiment in terms of the time each subject needs. Depending on the group, this may be with or without the new tool. Finally, we compute the difference in time, needed by subjects in the treatment and control group.

11.3 Simulated Scenario H

In the following simulation, we simulate the experiment producing an artificial version, where our new tool improves how a subject handles a task. The time needed is reduced by four minutes.

We will hard-code relevant parameters, like the number of subjects (20), the effect of the tool (- 4 minutes), or the severity of difference in the subject's preconditions (given by a standard deviation of 5 minutes). We recommend changing these parameters and exploring the implications based on the online material to get a better understanding of the impact on results.

```
 1 N ← 20 # Number of subjects.
 2 S ← rnorm(N, mean = 60, sd = 5) # Unobserved preconditions of
      subjects.
 3
 4 # Randomly assigning treatment and control.
 5 G ← sample(c("treatment", "control"), N, replace = T)
 6
 7 # The effect of our treatment.
 8 X ← ifelse(G == "treatment", −4, 0)
 9
10 # Composing the time that a subject actually needs.
11 Y ← S + X
```

Listing 10 Simulating a randomized experiment.

In a first step, the code decides on the number of subjects N (line 1). We simulate the preconditions of subjects as the unobserved variable *S*, i.e., the time a subject would need for the task ignoring tool support. We use a stochastic function following a normal distribution with a mean of 60 minutes and a standard deviation of 5 minutes (line 2). We thereby have artificial subjects as unobserved preconditions *S*.

Following the standard method of randomized experiments, we now randomly assign our subjects to treatment and control group. We use the stochastic function *sample* to do such random assignment to *treatment* or *control* (line 5). We store it as observed variable *G*. We then simulate the effect of our new tool X for the assignments, as a function of G, using a basic *ifelse* (line 8). We make our treatment (new tool) decrease the time needed by exactly four minutes.

Here we face an important difference to our previous definitions of X, e.g., in Sec. 7, since we can assure by the design of the experiment, that G, and thereby also X, is independent of anything else.

Finally, we simulate running the experiment, producing the time Y as the sum of X and S (line 11). The time is decreased by our treatment but still influenced by each subject's unobserved precondition.

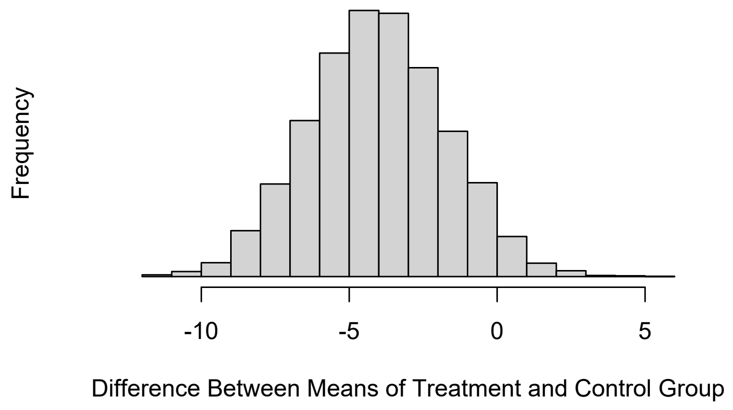

Fig. 10 Repeating the simulated experiment and interpreting the plain difference between treatment and control group.

11.4 Rating Results

Obviously, we should recognize that the experiment is a success by just looking at the difference between treatment and control group's average time needed for the task. Our simulation clearly defines that the treatment (the tool) decreases the time by four minutes.

However, the simulated random preconditions of subjects will complicate showing the success of our tool. We can illustrate the invalidity of an oversimplistic method that only interprets the difference between treatment and control group, by repeating the simulated experiment many times. We record plain difference between treatment and control (see online resources) and report on it in the histogram in Fig. 10.

Fortunately, most of the simulated experiments suggest that our new tool is indeed a success. Often, we almost exactly meet the -4 minutes. However, there are also cases where we run into simulated experiments that suggest that the new tool slows down the task. We know that this is not the case, since the simulation is fully transparent.

The reason behind these cases are unlucky assignments of subjects to treatment and control, where the majority of the subjects with good preconditions concentrate on the control group. This is something that happens. There is no way to resolve this problem, without a better observation of the preconditions (which is often impossible).

11.5 Revision

We will now show the conceptual remedy, that resolves this problem by reasoning about the uncertainty associated with the preconditions of the subjects. It is the typical device of a t-test or confidence interval, that almost all studies

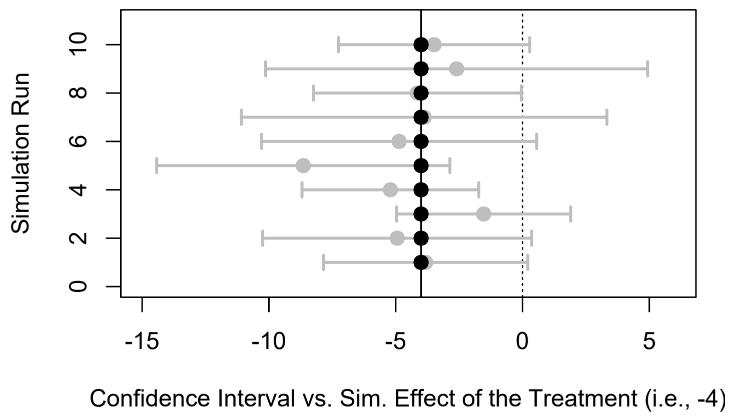

Fig. 11 Showing 10 simulated experiments and the corresponding uncertainty for the difference between treatment and control. The black dot reflects the simulated difference, the gray dot the difference that we find, and the gray bar the 95% confidence interval around the difference.

doing experiments are aware of. We again focus on confidence intervals, which we believe are more intuitive for most readers.

Instead of interpreting the plain difference that we find between treatment and control group, we now prefer to interpret a confidence interval around this difference. We illustrate this in Fig. 11 in the same way as we have done it previously in Sec. 6. The confidence intervals for 10 simulation runs uncover two important insights:

- First, we resolve the threat of accidentally claiming a wrong effect of our tool. In all simulation runs, we see that the confidence interval includes the simulated effect of our tool -4. When running the actual experiment, we can thereby be sure that that our statement, which is now less accurate by including a notion of uncertainty, is valid in claiming the success of our tool under the simulated assumptions.
- However, we spotted a new problem. Most of our confidence intervals (8 out of 10) suggest that we cannot be sure that our tool has an influence at all because the 0 is included in the intervals. Some people say that we missed rejecting the null hypothesis. This operationalizes invalidity of statements about the '*absence of an improvement by our new tool*'. The simulation states that the effect is there. If we claim absence because of a confidence interval that included the 0, we are wrong. This may happen for about 80% of the runs, but we can decrease the chance. For instance, by increasing the number of subjects, the confidence intervals will get more narrow and the 80% will decrease. We do this until we think the chance is small enough to be accepted. If we now still face a confidence interval including the 0, the effect of our tool may be truly negligible. We face an instance of a power analysis based on a simulation.

11.6 Conclusions on Validity

The important property of such experiment is that the *independence* of the assignment to treatment and control can be assured by the design of the experiment. We operationalize this assumption in our simulation.

As the consequence, this assures the *independence* of the effect of the tool used in the experiment (the effect that we are actually interested in). We can continue to reason about causation, as sketched in Sec. 7, while having the most relevant assumption in the simulation plausibly being assured by our design. There is no need to control for variables, as required in Sec. 8.

Furthermore, this example again revisits aspects of uncertainty, first presented in Sec. 4.3.2, and refined in the context of dependent observations in Sec. 6. However, having dependent observations in experimenting is not completely implausible. Subjects might influence each other when running the experiment in the same room, or we might design the experiment so that we measure the same subject executing multiple tasks. Such dependent observations require advanced methods, e.g., subject-item designs, examined in simulations in [6].

We also distinguished between claiming the existence and non-existence of effects. We show a basic version of a power analysis by a simulation. We recommend exploring the online material, adjusting N, the effect of the tool, or the severity of difference in the subject's preconditions given by the standard deviation for simulating S, to examine the impact for resulting claims on existence and non-existence.

12 Related Work

The related work section covers other studies in MSR/ESE that can benefit from simulation-based testing, work that already relates to simulation and the use of simulation in other domains.

12.1 Empirical Studies

We are not aware of MSR/ESE research that tests a method by simulation and reports on this. In the following, we discuss some other studies that may potentially benefit from simulation-based testing.

In [30], reasons for long duration builds in continuous integration pipelines are examined using multilevel models. Boh et al. [14] show an effect of experience on productivity using multilevel models. The authors of the previous papers are aware of the issues of dependent observations using advanced solutions, not comparable to the method shown in our first case (Sec. 6). However, multilevel models are complicated. Our experience is that simulations can be of great help in testing and understanding how multilevel models react to the threat of a structured sampling process in MSR/ESE.

Several works in MSR/ESE use a method that assumes completely independent observations and thereby invokes threats (e.g., [61,81,86]). Such work may benefit from simulating the structured sampling process, and other reoccurring structural entities, like artifacts and developers, for detecting the potential dangers.

In the conference version of this paper, we have raised the concern that the model used by Yan et al. in [85] may be over-parametrized and therefore does not fit properly. Meanwhile, initial simulations of us did not manage to show this. The model works in a simplified version of the presented scenario. We do not want to make any further claims on the validity of this study without an in-depth study by simulations. However, this mistake by us again shows the relevance of our more operational ways of examining validity by simulations.

There is work discussing aggregation or disaggregation strategies on software engineering data [86,35]. In simulations, we have managed to show that aggregation artificially increases correlation. Simulation-based tests may guide novel ideas on how to resolve issues with correlated variables (Sec. 9), potentially by disaggregated analysis of repository data. This is a promising direction for future work.

Further, we assume that a series of work, relying on the well known SZZ algorithm [73], may benefit from simulation-based testing. Defect classification produced by SZZ is critically influenced by the sampling process, and the temporal evolution of commits in a repository. Simulations of commit and fix behavior of developers may uncover that SZZ classifications share a natural correlation with time because for later commits, opportunities being fixed are just getting rare. This can be considered as a systematic measurement error. Hence, the effect of every metric correlating with time, e.g., experience measures, may be confused with such an effect. It may be resolved in parts by the control of variables (Sec. 8)

Bird at al. [12] examine the empirical challenges of incorrectly labeled bugs in historical defect data, which is an important threat to following up steps of a method. Transferring this reference to our terminology, a 'fix' is an observed variable, but the actual 'bug' is unobserved. We may simulate both to examine the impact of different assumptions on this relation. Bird at al. does an initial step in the examination, but does not use synthetic fix-bug-pairs. This makes forming a precise picture complicated.

Authors of [64] report on the occurrence of well-known threats in existing literature. Opposed to a plain literature survey, simulation-based testing is a strategy to operationalize statements about the relationship between empirical scenario, method, and results.

12.2 Simulation

The distinction between simulation and prediction is often vague. In the following, we list work in MSR/ESE that refers to their own approach as simulation.

In [37,38,36], simulations of the software development process are introduced to help project managers to extrapolate future scenarios. Data mined from repositories is used to construct the simulations. The authors use agent-based systems. Simulations are used to extrapolate, which is reasonable if configured with the right prior knowledge on unobserved variables. In [69], agent-based simulations for OS development are created using prior literature to set the relevant unobserved variables. In [13], multi-agent simulations predict the next moves of agents. In [11], social coding dynamics are simulated based on historic data to forecast information spread.

In contrast to such work, our simulations operationalize statements on the validity of a study. A simulation should be used to test empirical practice in MSR/ESE research, to spot cases where a method fails or is threatened. Often, it is not clear how a method reacts to assumptions before seeing the consequences in a simulation.

12.3 Simulation in Other Domains

An exhaustive discussion of the use of simulations in other domains is beyond the scope of this paper. However, we list some selected and influential work here.

Statistic work evaluates cross-validation using a simulation study in [71]. In [65], cross-validation is evaluated on simulated structural data in the field of ecology. In [9], the impact of random effect structures is examined by simulation. Subject-item designs are examined in simulations in [6].

The introductions to statistics in [50,29,31] contain simulations as central devices for illustration. Most notably, such a trend is reflected in the book by Gelman et al. from 2020, where the preface states: *'Existing textbooks on regression typically have some mix of cookbook instruction and mathematical derivation. We wrote this book because we saw a new way forward, focusing on understanding regression models, applying them to real problems, and using simulations with fake data to understand how the models are fit.'* (direct citation of [29])

In [27], the authors simulate what happens if something informative is ignored, which is part of longitudinal health data. We assume that temporal structure is also critical for MSR and deserves more attention (for efforts in the longitudinal MSR data collection, see [32]). The authors of [17] discuss simulation studies in medicine. The evaluation of statistic methods by simulation is discussed by [54] – also in medicine. In [41,84], authors discuss the role of simulation in learning statistics.

13 Conclusion

This paper describes and validates a new strategy to validate methods and results of empirical research studies. While reproducibility and replicability are somewhat understood, standardized and operational ways to define and communicate such validity of empirical research studies are less understood.

Our strategy operationalizes important assumptions, that are typically informal in papers, by simulations. We use the simulations to show how the assumptions impact the results of a study. We call the strategy *simulation-based testing* because of the analogy between writing simulation artifacts and test cases.

In a (meta) validation, we show how simulation-based testing instantiates research questions on the validity of studies in six real scenarios. We show that we can either support validity, threaten validity, or invalidate studies.

We encourage researchers to accompany submissions of research works with simulation artifacts, thereby proving the status of an operationalized validation; thereby, helping reviewers in assessing validity. In this way, the reviewing process of future empirical work would be improved.

Conflict of Interest

The authors have no conflict of interest.

References

1. Akaike, H.: Information Theory and an Extension of the Maximum Likelihood Principle. In: E. Parzen, K. Tanabe, G. Kitagawa (eds.) Selected Papers of Hirotugu Akaike, pp. 199–213. Springer (1998)
2. Alali, A., Kagdi, H.H., Maletic, J.I.: What's a Typical Commit? A Characterization of Open Source Software Repositories. In: ICPC, pp. 182–191. IEEE Computer Society (2008)
3. Albayrak, Ö., Carver, J.C.: Investigation of individual factors impacting the effectiveness of requirements inspections: a replicated experiment. Empir. Softw. Eng. **19**(1), 241–266 (2014)
4. Anda, B., Sjøberg, D.I.K.: Investigating the Role of Use Cases in the Construction of Class Diagrams. Empir. Softw. Eng. **10**(3), 285–309 (2005)
5. Apa, C., Dieste, O., G., E.G.E., C., E.R.F.: Effectiveness for detecting faults within and outside the scope of testing techniques: an independent replication. Empir. Softw. Eng. **19**(2), 378–417 (2014)
6. Baayen, R.H., Davidson, D.J., Bates, D.M.: Mixed-effects modeling with crossed random effects for subjects and items. Journal of Memory and Language **59**(4), 390–412 (2008)
7. Bangash, A.A., Sahar, H., Hindle, A., Ali, K.: On the time-based conclusion stability of cross-project defect prediction models. Empirical Software Engineering pp. 1–38 (2020)
8. Barón, M.M., Wyrich, M., Graziotin, D., Wagner, S.: Evidence profiles for validity threats in program comprehension experiments. In: ICSE, pp. 1907–1919. IEEE (2023)
9. Barr, D.J., Levy, R., Scheepers, C., Tily, H.J.: Random effects structure for confirmatory hypothesis testing: Keep it maximal. Journal of Memory and Language **368**(3), 255–278 (2013)
10. Beheim, B., Atkinson, Q.D., Bulbulia, J., Gervais, W., Gray, R.D., Henrich, J., Lang, M., Monroe, M.W., Muthukrishna, M., Norenzayan, A., Purzycki, B.G., Shariff, A., Slingerland, E., Spicer, R., Willard, A.K.: Treatment of missing data determined conclusions regarding moralizing gods. Nature **595**(7866), 1476–4687 (2021)
11. Bidoki, N.H., Schiappa, M., Sukthankar, G., Garibay, I.: Modeling social coding dynamics with sampled historical data. Online Soc. Networks Media **16**, 100070 (2020)

12. Bird, C., Bachmann, A., Aune, E., Duffy, J., Bernstein, A., Filkov, V., Devanbu, P.T.: Fair and balanced?: bias in bug-fix datasets. In: ESEC/SIGSOFT FSE, pp. 121–130. ACM (2009)
13. Blythe, J., Bollenbacher, J., Huang, D., Hui, P., Krohn, R., Pacheco, D., Muric, G., Sapienza, A., Tregubov, A., Ahn, Y., Flammini, A., Lerman, K., Menczer, F., Weninger, T., Ferrara, E.: Massive Multi-agent Data-Driven Simulations of the GitHub Ecosystem. In: PAAMS, *Lecture Notes in Computer Science*, vol. 11523, pp. 3–15. Springer (2019)
14. Boh, W.F., Slaughter, S., Espinosa, J.A.: Learning from Experience in Software Development: A Multilevel Analysis. Manag. Sci. **53**(8), 1315–1331 (2007)
15. Borges, H., Hora, A.C., Valente, M.T.: Predicting the Popularity of GitHub Repositories. In: PROMISE, pp. 9:1–9:10. ACM (2016)
16. Borle, N.C., Feghhi, M., Stroulia, E., Greiner, R., Hindle, A.: Analyzing the effects of test driven development in GitHub. Empirical Software Engineering **23**(4), 1931–1958 (2018)
17. Burton, A., Altman, D.G., Royston, P., Holder, R.L.: The design of simulation studies in medical statistics. Statistics in Medicine **25**(24), 4279–4292 (2006)
18. Canfora, G., Lucia, A.D., Penta, M.D., Oliveto, R., Panichella, A., Panichella, S.: Defect prediction as a multiobjective optimization problem. Softw. Test. Verification Reliab. **25**(4), 426–459 (2015)
19. Casalnuovo, C., Devanbu, P.T., Oliveira, A., Filkov, V., Ray, B.: Assert Use in GitHub Projects. In: ICSE (1), pp. 755–766. IEEE Computer Society (2015)
20. Clyburne-Sherin, A., Fei, X., Green, S.A.: Computational reproducibility via containers in psychology. Meta-psychology **3** (2019)
21. Cohen, J., Cohen, P., West, S.G., Aiken, L.S.: Applied multiple regression/correlation analysis for the behavioral sciences. Routledge (2013)
22. Cosentino, V., Izquierdo, J.L.C., Cabot, J.: Findings from GitHub: methods, datasets and limitations. In: Proc. MSR, pp. 137–141 (2016)
23. Dias, M., Bacchelli, A., Gousios, G., Cassou, D., Ducasse, S.: Untangling fine-grained code changes. In: SANER, pp. 341–350. IEEE Computer Society (2015)
24. Falcão, F., Barbosa, C., Fonseca, B., Garcia, A., Ribeiro, M., Gheyi, R.: On Relating Technical, Social Factors, and the Introduction of Bugs. In: SANER, pp. 378–388. IEEE (2020)
25. Fang, H., Lamba, H., Herbsleb, J.D., Vasilescu, B.: "this is damn slick!" estimating the impact of tweets on open source project popularity and new contributors. In: ICSE, pp. 2116–2129. ACM (2022)
26. Gabel, M., Su, Z.: A study of the uniqueness of source code. In: SIGSOFT FSE, pp. 147–156. ACM (2010)
27. Gasparini, A., Abrams, K.R., Barrett, J.K., Major, R.W., Sweeting, M.J., Brunskill, N.J., Crowther, M.J.: Mixed-effects models for health care longitudinal data with an informative visiting process: A Monte Carlo simulation study. Statistica Neerlandica **74**(1), 5–23 (2020)
28. Gelman, A., Hill, J.: Data analysis using regression and multilevel/hierarchical models. Cambridge university press (2006)
29. Gelman, A., Hill, J., Vehtari, A.: Regression and other stories. Cambridge University Press (2020)
30. Ghaleb, T.A., da Costa, D.A., Zou, Y.: An empirical study of the long duration of continuous integration builds. Empirical Software Engineering **24**(4), 2102–2139 (2019)
31. Harrell, F.E.: Regression modeling strategies: with applications to linear models, logistic and ordinal regression, and survival analysis, vol. 2. Springer (2015)
32. Härtel, J., Lämmel, R.: Incremental Map-Reduce on Repository History. In: SANER, pp. 320–331. IEEE (2020)
33. Härtel, J., Lämmel, R.: Operationalizing threats to MSR studies by simulation-based testing. In: MSR, pp. 86–97. IEEE (2022)
34. He, Z., Peters, F., Menzies, T., Yang, Y.: Learning from Open-Source Projects: An Empirical Study on Defect Prediction. In: ESEM, pp. 45–54. IEEE Computer Society (2013)
35. Herzig, K., Zeller, A.: The impact of tangled code changes. In: MSR, pp. 121–130. IEEE Computer Society (2013)

36. Honsel, V.: Statistical Learning and Software Mining for Agent Based Simulation of Software Evolution. In: ICSE (2), pp. 863–866. IEEE Computer Society (2015)
37. Honsel, V., Honsel, D., Grabowski, J.: Software Process Simulation Based on Mining Software Repositories. In: ICDM Workshops, pp. 828–831. IEEE Computer Society (2014)
38. Honsel, V., Honsel, D., Herbold, S., Grabowski, J., Waack, S.: Mining Software Dependency Networks for Agent-Based Simulation of Software Evolution. In: ASE Workshops, pp. 102–108. IEEE Computer Society (2015)
39. Imbens, G.W., Rubin, D.B.: Causal inference in statistics, social, and biomedical sciences. Cambridge University Press (2015)
40. Iyer, R.N., Yun, S.A., Nagappan, M., Hoey, J.: Effects of Personality Traits on Pull Request Acceptance. IEEE Transactions on Software Engineering (2019)
41. Jamie, D.M.: Using computer simulation methods to teach statistics: A review of the literature. Journal of Statistics Education **10**(1) (2002)
42. Jbara, A., Matan, A., Feitelson, D.G.: High-MCC Functions in the Linux Kernel. Empir. Softw. Eng. **19**(5), 1261–1298 (2014)
43. Jiarpakdee, J., Tantithamthavorn, C., Hassan, A.E.: The Impact of Correlated Metrics on the Interpretation of Defect Models. IEEE Trans. Software Eng. **47**(2), 320–331 (2021)
44. Johnson, J., Lubo, S., Yedla, N., Aponte, J., Sharif, B.: An Empirical Study Assessing Source Code Readability in Comprehension. In: ICSME, pp. 513–523. IEEE (2019)
45. Jolak, R., Savary-Leblanc, M., Dalibor, M., Wortmann, A., Hebig, R., Vincur, J., Polásek, I., Pallec, X.L., Gérard, S., Chaudron, M.R.V.: Software engineering whispers: The effect of textual vs. graphical software design descriptions on software design communication. Empir. Softw. Eng. **25**(6), 4427–4471 (2020)
46. Kamei, Y., Shihab, E., Adams, B., Hassan, A.E., Mockus, A., Sinha, A., Ubayashi, N.: A Large-Scale Empirical Study of Just-in-Time Quality Assurance. IEEE Trans. Software Eng. **39**(6), 757–773 (2013)
47. Kochhar, P.S., Lo, D.: Revisiting Assert Use in GitHub Projects. In: EASE, pp. 298–307. ACM (2017)
48. Martens, A., Koziolek, H., Prechelt, L., Reussner, R.H.: From monolithic to component-based performance evaluation of software architectures - A series of experiments analysing accuracy and effort. Empir. Softw. Eng. **16**(5), 587–622 (2011)
49. McChesney, I.R., Bond, R.R.: Observations on the Linear Order of Program Code Reading Patterns in Programmers with Dyslexia. In: EASE, pp. 81–89. ACM (2020)
50. McElreath, R.: Statistical rethinking: A Bayesian course with examples in R and Stan. CRC press (2020)
51. Miller, G.: A Scientist's Nightmare: Software Problem Leads to Five Retractions. Science **314**(5807), 1856–1857 (2006)
52. Mockus, A.: Organizational volatility and its effects on software defects. In: SIGSOFT FSE, pp. 117–126. ACM (2010)
53. Mockus, A., Weiss, D.M.: Predicting risk of software changes. Bell Labs Technical Journal **5**(2), 169–180 (2000)
54. Morris, T.P., White, I.R., Crowther, M.J.: Using simulation studies to evaluate statistical methods. Statistics in Medicine **38**(11), 2074–2102 (2019)
55. Nagappan, N., Zeller, A., Zimmermann, T., Herzig, K., Murphy, B.: Change Bursts as Defect Predictors. In: ISSRE, pp. 309–318. IEEE Computer Society (2010)
56. Nam, J., Fu, W., Kim, S., Menzies, T., Tan, L.: Heterogeneous Defect Prediction. IEEE Trans. Software Eng. **44**(9), 874–896 (2018)
57. Nam, J., Pan, S.J., Kim, S.: Transfer defect learning. In: ICSE, pp. 382–391. IEEE Computer Society (2013)
58. Peduzzi, P., Concato, J., Kemper, E., Holford, T.R., Feinstein, A.R.: A simulation study of the number of events per variable in logistic regression analysis. Journal of clinical epidemiology **49**(12), 1373–1379 (1996)
59. Penta, M.D., Cerulo, L., Guéhéneuc, Y., Antoniol, G.: An empirical study of the relationships between design pattern roles and class change proneness. In: ICSM, pp. 217–226. IEEE Computer Society (2008)
60. Posnett, D., Filkov, V., Devanbu, P.T.: Ecological inference in empirical software engineering. In: ASE, pp. 362–371. IEEE Computer Society (2011)

61. Rahman, F., Devanbu, P.T.: Ownership, experience and defects: a fine-grained study of authorship. In: ICSE, pp. 491–500. ACM (2011)
62. Rahman, F., Posnett, D., Devanbu, P.T.: Recalling the "imprecision" of cross-project defect prediction. In: SIGSOFT FSE, p. 61. ACM (2012)
63. Rahman, M.M., Roy, C.K., Collins, J.A.: CoRReCT: code reviewer recommendation in GitHub based on cross-project and technology experience. In: ICSE (Companion Volume), pp. 222–231. ACM (2016)
64. Reyes, R.P., Dieste, O., C., E.R.F., Juristo, N.: Statistical errors in software engineering experiments: a preliminary literature review. In: ICSE, pp. 1195–1206. ACM (2018)
65. Roberts, D.R., Bahn, V., Ciuti, S., Boyce, M.S., Elith, J., Guillera-Arroita, G., Hauenstein, S., Lahoz-Monfort, J.J., Schröder, B., Thuiller, W., Warton, D.I., Wintle, B.A., Hartig, F., Dormann, C.F.: Cross-validation strategies for data with temporal, spatial, hierarchical, or phylogenetic structure. Ecography **40**(8), 913–929 (2017)
66. Sayagh, M., Kerzazi, N., Petrillo, F., Bennani, K., Adams, B.: What should your runtime configuration framework do to help developers? Empir. Softw. Eng. **25**(2), 1259–1293 (2020)
67. Scholtes, I., Mavrodiev, P., Schweitzer, F.: From Aristotle to Ringelmann: a large-scale analysis of team productivity and coordination in Open Source Software projects. Empir. Softw. Eng. **21**(2), 642–683 (2016)
68. Seifer, P., Härtel, J., Leinberger, M., Lämmel, R., Staab, S.: Empirical study on the usage of graph query languages in open source Java projects. In: SLE, pp. 152–166. ACM (2019)
69. Seo, T., Lee, H.: Agent-based Simulation Model for the Evolution Process of Open Source Software. In: SEKE, pp. 170–177. Knowledge Systems Institute Graduate School (2009)
70. Shadish, W.R., Cook, T.D., Campbell, D.T.: Experimental and quasi-experimental designs for generalized causal inference. Houghton Mifflin Company (2002)
71. Shao, J.: Linear model selection by cross-validation. Journal of the American statistical Association **88**(422), 486–494 (1993)
72. Sjøberg, D.I.K., Hannay, J.E., Hansen, O., Kampenes, V.B., Karahasanovic, A., Liborg, N., Rekdal, A.C.: A Survey of Controlled Experiments in Software Engineering. IEEE Trans. Software Eng. **31**(9), 733–753 (2005)
73. Sliwerski, J., Zimmermann, T., Zeller, A.: When do changes induce fixes? In: MSR. ACM (2005)
74. Stodden, V., Seiler, J., Ma, Z.: An empirical analysis of journal policy effectiveness for computational reproducibility. Proc. Natl. Acad. Sci. USA **115**(11), 2584–2589 (2018)
75. Tan, M., Tan, L., Dara, S., Mayeux, C.: Online Defect Prediction for Imbalanced Data. In: ICSE (2), pp. 99–108. IEEE Computer Society (2015)
76. Tantithamthavorn, C., Hassan, A.E.: An experience report on defect modelling in practice: pitfalls and challenges. In: ICSE (SEIP), pp. 286–295. ACM (2018)
77. Tantithamthavorn, C., McIntosh, S., Hassan, A.E., Matsumoto, K.: An Empirical Comparison of Model Validation Techniques for Defect Prediction Models. IEEE Trans. Software Eng. **43**(1), 1–18 (2017)
78. Thongtanunam, P., McIntosh, S., Hassan, A.E., Iida, H.: Revisiting code ownership and its relationship with software quality in the scope of modern code review. In: ICSE, pp. 1039–1050. ACM (2016)
79. Tichy, W.F., Lukowicz, P., Prechelt, L., Heinz, E.A.: Experimental evaluation in computer science: A quantitative study. J. Syst. Softw. **28**(1), 9–18 (1995)
80. Tsay, J., Dabbish, L., Herbsleb, J.D.: Influence of social and technical factors for evaluating contribution in GitHub. In: ICSE, pp. 356–366. ACM (2014)
81. Tufano, M., Bavota, G., Poshyvanyk, D., Penta, M.D., Oliveto, R., Lucia, A.D.: An empirical study on developer-related factors characterizing fix-inducing commits. J. Softw. Evol. Process. **29**(1) (2017)
82. Vasilescu, B., Posnett, D., Ray, B., van den Brand, M.G.J., Serebrenik, A., Devanbu, P.T., Filkov, V.: Gender and Tenure Diversity in GitHub Teams. In: CHI, pp. 3789–3798. ACM (2015)
83. Vokác, M.: Defect Frequency and Design Patterns: An Empirical Study of Industrial Code. IEEE Trans. Software Eng. **30**(12), 904–917 (2004)

84. Wood, M.: The role of simulation approaches in statistics. Journal of Statistics Education **13**(3) (2005)
85. Yan, M., Xia, X., Fan, Y., Lo, D., Hassan, A.E., Zhang, X.: Effort-aware just-in-time defect identification in practice: a case study at Alibaba. In: ESEC/SIGSOFT FSE, pp. 1308–1319. ACM (2020)
86. Zhang, F., Hassan, A.E., McIntosh, S., Zou, Y.: The Use of Summation to Aggregate Software Metrics Hinders the Performance of Defect Prediction Models. IEEE Trans. Software Eng. **43**(5), 476–491 (2017)
87. Zimmermann, T., Nagappan, N.: Predicting defects using network analysis on dependency graphs. In: ICSE, pp. 531–540. ACM (2008)
88. Zimmermann, T., Premraj, R., Zeller, A.: Predicting defects for eclipse. In: PROMISE 2007, p. 76. IEEE (2007)