Operationalizing Threats to MSR Studies by Simulation-Based Testing

Johannes Härtel
University of Koblenz
Germany
johanneshaertel@uni-koblenz.de

Ralf Lämmel
University of Koblenz
Germany
laemmel@uni-koblenz.de

ABSTRACT

Quantitative studies on the border between Mining Software Repository (MSR) and Empirical Software Engineering (ESE) apply data analysis methods, like regression modeling, statistic tests or correlation analysis, to commits or pulls to better understand the software development process. Such studies assure the validity of the reported results by following a sound methodology. However, with increasing complexity, parts of the methodology can still go wrong. This may result in MSR/ESE studies with undetected threats to validity. In this paper, we propose to systematically protect against threats by operationalizing their treatment using simulations. A simulation substitutes observed and unobserved data, related to an MSR/ESE scenario, with synthetic data, carefully defined according to plausible assumptions on the scenario. Within a simulation, unobserved data becomes transparent, which is the key difference to a real study, necessary to detect threats to an analysis methodology. Running an analysis methodology on synthetic data may detect basic technical bugs and misinterpretations, but it also improves the trust in the methodology. The contribution of a simulation is to operationalize testing the impact of important assumptions. Assumptions still need to be rated for plausibility. We evaluate simulation-based testing by operationalizing undetected threats in the context of four published MSR/ESE studies. We recommend that future research uses such more systematic treatment of threats, as a contribution against the reproducibility crisis.

CCS CONCEPTS

• **General and reference → Empirical studies**; • **Software and its engineering** → *Software defect analysis*; *Empirical software validation*.

KEYWORDS

Simulation, empirical studies, threats, synthetic data, testing.

ACM Reference Format:
Johannes Härtel and Ralf Lämmel. 2022. Operationalizing Threats to MSR Studies by Simulation-Based Testing. In *19th International Conference on Mining Software Repositories (MSR '22), May 23–24, 2022, Pittsburgh, PA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3524842.3527960

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
MSR '22, May 23–24, 2022, Pittsburgh, PA, USA
© 2022 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-9303-4/22/05…$15.00
https://doi.org/10.1145/3524842.3527960

1 INTRODUCTION

Quantitative studies on the border between Mining Software Repositories (MSR) and Empirical Software Engineering (ESE) apply data analysis methods, like regression modeling, statistic tests or correlation analysis, with the aim to understand, and finally improve the software engineering process [1, 2]. Recent research examines, e.g., defects [3], pull accepts [4] or reviewing activity [5]. It uses the data mined from repositories to make statements or tools that ease software development in general (see [6–8]).

1.1 Empirical Challenges on Repositories

Opposed to producing empirical insights by well-understood randomized experiments, producing empirical insights using observational repository data is highly challenging.

The complexity reaches its peak in examining *causation*, crucial for every scientific domain (see general work of leading authors examining causation [9]). Other challenges that arise when examining software repositories are the sampling process, control variables or correlated variables.

Such challenges complicate the methodology of MSR/ESE studies in that the validity cannot be judged easily. Informal practice is to judge the study results according to previous knowledge and expectations, given by anecdotes or references (e.g., as done in [10]). However, validity often remains vague in such informal discussion.

1.2 Simulation-Based Testing

In this paper, we propose to systematically protect against threats by operationalizing their treatment using simulations.

We map the core challenges of empirical MSR research to a missing data problem. Study results are unobserved variables (such as *parameters in a regression model*, *correlation*, or *confidence intervals*) inferred from observed variables that are mined from repositories (such as *defects* or *changed lines-of-code*). We argue that the crucial point, complicating the treatment of threats, is that we do not know the unobserved variables definitely – otherwise we would not need to do empirical research.

We cannot contribute any improvements to the conceptual problem of unobserved variables in real studies; however, we can operationalize tests for a study's methodology working with them.

We propose to use simulations that substitute observed and unobserved variables, related to an MSR/ESE scenario, with synthetic variables, carefully defined according to plausible assumptions on the scenario. Within a simulation, unobserved variables become transparent, which is the key difference to a real study, necessary to detect threats to an analysis methodology.

Running an analysis methodology on synthetic variables may detect basic technical bugs and misinterpretations, but it also improves the trust in the methodology.

1.3 Plausibility of Simulations

The contribution of a simulation is to operationalize testing the impact of important assumptions on the results of a methodology. Assumptions are given in terms of substitutions of variables. Whether threats detected by such simulations can be transferred back to real studies, requires rating the plausibility of underlying assumptions in the concrete case.

1.4 Evaluation

We prove that simulation-based testing contributes to the treatment of threats to validity in four published MSR/ESE studies, using very basic substitutions that, we believe, are plausible in MSR/ESE. **The full simulation-based tests are provided online**[1].

1.5 Contributions

Our contributions are: i) the presentation of simulation-based testing for MSR/ESE studies to operationalize threats to validity; ii) an evaluation showing the relevance of simulation-based testing by applying it to published studies in MSR/ESE; and iii) an initial catalog of relevant threats to MSR/ESE studies.

1.6 Road-Map

Sec. 2 introduces simulation-based testing; Sec. 3 introduces a basic example; Sec. 4 is an acknowledgement to the author of the original work that we examine in the following sections; Sec. 5-8 apply simulation-base testing to existing studies in MSR/ESE; Sec. 9 lists related work; Sec. 10 concludes.

2 SIMULATION-BASED TESTING

This section introduces simulation-based testing.

2.1 Models

We start with a definition of a model, and the corresponding terminology, as we use it in this paper. A model consists of two parts:

- **Variables**: Variables label relevant data for the research. Typical variables in MSR/ESE are the occurrence of defects, lines-of-code, or effect strengths. 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 [3]), other variables are *unobserved* and need to be inferred as the goal of an empirical study (e.g., effect strength of lines-of-code influencing defects).
- **Relationships between variables**: Relationships describe how variables relate to each other. Relationships may be functional or stochastic, the latter invokes a notion of uncertainty. Relationships can never be observed directly, so they are always assumptions, part of our model.

A model can use relationships to *infer*, *predict* or *simulate* variables using other variables.

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

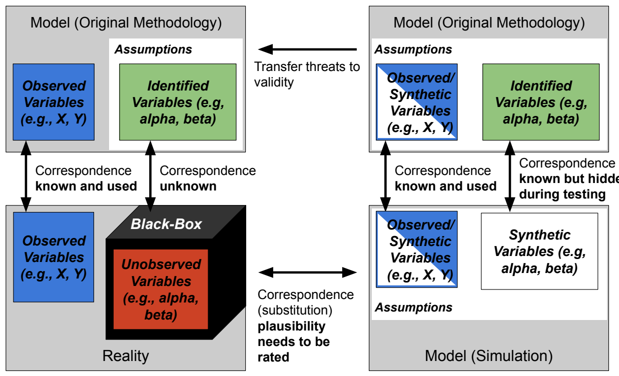
Model (Original Methodology)
Assumptions
Observed Variables (e.g., X, Y)
Identified Variables (e.g, alpha, beta)
Transfer threats to validity
Model (Original Methodology)
Assumptions
Observed/ Synthetic Variables (e.g., X, Y)
Identified Variables (e.g, alpha, beta)
Correspondence known and used
Correspondence unknown
Correspondence known and used
Correspondence known but hidden during testing
Observed Variables (e.g., X, Y)
Black-Box
Unobserved Variables (e.g., alpha, beta)
Reality
Correspondence (substitution) plausibility needs to be rated
Observed/ Synthetic Variables (e.g., X, Y)
Synthetic Variables (e.g, alpha, beta)
Assumptions
Model (Simulation)

Figure 1: Replacing the reality by a simulation: The substitution of observed (blue) and unobserved (red) variables with synthetic variables (white), allows checking the correspondence between identified variables (green) and the synthetic counterparts (white). Detected threats to validity can possibly be transferred back, if the substitution is plausible.

Depending on which variables are observed or not, how unobserved variables are treated, and which relationships are used, authors may call this procedure differently: The model may *predict* unobserved defects; it may *infer* parameters relevant to a relationship; or it may *simulate* complete data sets following plausible assumptions on unobserved variables (like on the relationship between defects and lines-of-code). In the following, we will use the term *identify* as a summary of this procedure.

2.2 Methodology of an Empirical Study

A standard empirical study takes the following steps: The study converts its hypothesis on reality, like the software development in a repository, into a single or many such models; the data collection mines a sample of repositories to replace some unobserved with observed variables; the model attempts to identify the remaining unobserved variables which are relevant to research. Finally, the results of the different models are compared and conclusions on the reality are drawn.

2.3 Reality as a Black-Box

The crucial point why the treatment of threats is difficult in such methodology is because unobserved variables and relationships are not known in general. They are hidden in some sort of *black-box* dictated by reality (see the illustration in Figure 1 which presents the conceptual problem with a focus on variables). This makes it impossible to test the analysis methodology in a classical sense, i.e., comparing the identified variables with the correct counterparts from reality.

2.4 Simulation as a Substitution for Reality

A simulation-based test creates a second model which substitutes observed and unobserved variables, related to an MSR/ESE scenario, with synthetic variables, carefully defined according to plausible assumptions on the scenario. Such simulation sets unobserved variables, defines them according to assumptions on distributions, uses assumed relationships (in reverse), and replaces other related variables accordingly.

Technically, a simulation implements stochastic or functional relationships between variables, i.e., drawing variables from random number generators (stochastic) or producing variables according to basic mathematical functions (functional). Such simulations boil down to very basic code, not needing specific libraries or extensive statistic background knowledge.

A study's original methodology then processes the synthetic variables as if they were the real data. The variables which are unobserved in the original study are still kept hidden in the simulated study run. The original identification mechanism is executed. Finally, the correspondence between the identified variables and the synthetic counterparts can be checked objectively because everything in the simulation is transparent.

This objective answer on identifiability, given by checking the correspondence, can be used to decide if the methodology is prone to a threat or not. The simulation operationalizes this threat.

2.5 Plausibility of Simulations

The contribution of a simulation is to operationalize testing the impact of important assumptions on the results of a methodology. Assumptions are given in terms of substitutions of variables. Whether threats detected by such simulations can be transferred back to real studies, requires rating the plausibility of underlying assumptions in the concrete case.

There are formal ways to check assumptions, but often there are subjective aspects, too. Hence, simulation-based testing still requires a reader or reviewer to rate the plausibility of assumptions – but compared to an informal discussion – simulation-based testing captures the assumptions and corresponding threats in a much more systematic way.

When designing, reviewing or revising a methodology, the aim is to stick to the most basic methodology to identify unobserved variables, 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 case. However, in some cases, simulations may also detect unidentifiable variables given the assumptions. In this case, the simulation remains as a threat to validity or renders the research aim impossible.

2.6 Generality, Complexity, and Automation

This section provides a preliminary answer on the generality, complexity, and the potential for automation in simulation-based testing. However, follow-up work is needed by us/the community.

We believe that there are three study types using simulation-based testing, increasing in generality, complexity and manual effort, allowing some sharing of work on the treatment of threats:

- A 'passive study' relies on an existing catalog of prototypical MSR/ESE simulations (such as the cases we will provide in the remainder of this paper) and transfers insights (directly read from the simulation) to the given empirical scenario. The focus on such a catalog limits us, but it also implies an ease of application with the potential for automation.
- An 'active study' goes one step further and applies minor modifications to existing simulations, tailoring them towards a new empirical scenario. Understanding of assumptions, methodology, and threats will be sharpened by putting them into the context. Tailoring is more flexible, more complicated, and less automated.
- An 'original study' contributes simulation to capturing novel understanding of threats in MSR/ESE studies, thereby extending the catalog we may have started with. Bias on bug-fix and historical data definitely calls for extensions to our paper's catalog. Contributing simulations is original work, it is fully flexible, but cannot be automated at all.

3 SIMULATION-BASED TESTING IN A NUTSHELL

We start with a trivial example on how a simulation may operationalize threats to a study's methodology. We will check the application of a basic logistic regression model, comparable to the methodology in many past and recent studies on software defects (e.g., in [11–19]).

3.1 Original Methodology

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. Understanding the relationship is a central aim of such research, e.g., in [11–19].

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 on the *elasticsearch* project, published in the context of examining defects in [12]. We use a binary defect classification for our observed variable Y (computed according to SZZ [3]). 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 model in R is given in Listing 1:

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

Listing 1: The original methodology applies a logistic regression to model the relationship between X and Y.

The model reports that the unobserved intercept is −3.28 and the slope is 0.45. From an empirical perspective, the interesting aspect is the strength and the direction 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.

3.2 Reality as a Black-Box

In such trivial case, the methodology 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 sound – although we have never seen 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[2].

3.3 Simulation as a Substitution for Reality

We will now substitute some observed and unobserved variables with synthetic variables, carefully simulated according to very basic assumptions on this scenario. We provide this simulation code in Listing 2 and as an online resource. All simulation code in the paper is written in standard R, not using libraries[3].

```
 1  # Kept observed variables.
 2  N ← N # Number of commits.
 3  X ← X # (vector) Keep the original variable X.
 4
 5  # Substituted unobserved variables.
 6  alpha ← -3.0
 7  beta ← 0.4
 8  prob ← 1 / (1 + exp(-(alpha + beta * X))) # (vector)
        Assumption of the logistic regression model on
        the relation between X and Y.
 9
10  # Substituted observed variable Y.
11  Y ← rbinom(N, size = 1, prob = prob) # (vector)
        Assumption on the output distribution.
```

Listing 2: The R code substitutes variables of the original methodology by synthetic variables.

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.

Next, the code assigns the unobserved intercept and slope variable in terms of *alpha* and *beta*. This is the crucial part of the simulation code, where the unobserved variables get replaced by synthetic variables. We *invent* both values. We use slightly different values, compared to those produced in the original study run (-3.0 for the intercept and 0.4 for the slope). We will test other options in the next section.

Nothing of the remaining code (line 8 and 11) is arbitrary. It follows the basic assumptions of a logistic regression model on the relationship between X and Y (just run in reverse). MSR/ESE studies (e.g. [11–19]) make this assumption implicitly when using a logistic regression models; 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 unknown variable that can never be observed. In reality, we can just observe the final defect classification Y. It is defined 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[4].

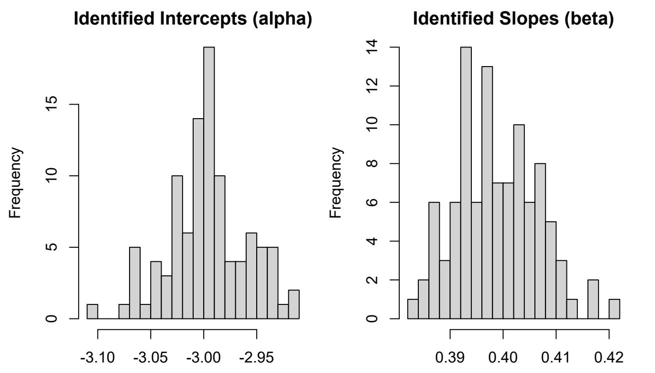

Figure 2: Identified unobserved variables intercept (alpha) and slope (beta) in 100 repeated simulation runs.

3.4 Correspondence, Uncertainty, and Parameterized Tests

We can now process the synthetic data as if it was the real data, using the original methodology from Listing 1 with one important difference: *We know alpha, beta, and prob.*

3.4.1 Correspondence. The original methodology identifies the unobserved variables in the simulated run to be $alpha \approx -2.97$ and $beta \approx 0.39$. We can objectively deny a threat to the methodology under the substitution because *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.

3.4.2 Uncertainty. First, it is unclear why the methodology does not exactly meet the synthetic *alpha* and *beta*. The answer is explicit in the simulation. It is because of the stochastic relationship used between *prob* and Y (Listing 2, line 11). The methodology only observes Y, but the corresponding *prob* is not known for sure; hence, also the identified values for *alpha* and *beta* are uncertain. This is one reason why studies often report on confidence estimates.

If repeating the simulation, using different initial seeds, the identified *alpha* and *beta* distribute as depicted in Figure 2. This shows that on average, the identified alpha and beta variables are excellent, but not totally exact.

3.4.3 Parametrized Tests. Second, it is not clear what happens if we use different synthetic values for *alpha* and *beta*. The simulation can examine this by going through multiple combinations of both, checking the implications on the identification by the methodology.

The results for a grid of 30×30 combinations is shown in Figure 3 (left). The plot illustrates the correspondence between the identified and synthetic beta (the error under the substitution). It shows that if synthetic alpha and beta are both high or low, identification struggles, producing a high error.

The phenomenon is well-known, but often not calibrated to the particular MSR/ESE scenario. If counting the number of defects in

[2]In causal inference, meeting collider variables, cross-validation of prediction performance may even be misleading (see general work [20], page 192).

[3]We remind the reader of vectorized operations in R; an introduction is beyond the scope of this paper.

[4]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*).

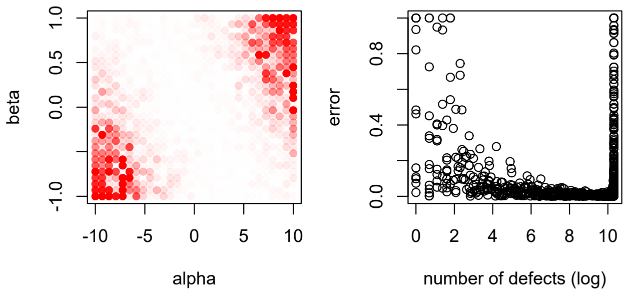

Figure 3: Left: Simulated alpha and beta and the corresponding error in the identification, depicted as red dots (red increases with error). Right: The total sum of defects in the synthetic variable Y (log scale) and the relation to the error.

the synthetic Y variable and relating it to this error, we see that the identification only fails on extremely high or low defect counts (see right of Figure 3).

We know this under the name *class-imbalance* [21] or in terms of instructions on events-per-variable (EPV) [22]. Our simulation does detailed suggestions, i.e., that if we have more than approximately 400 defects in the data set, the methodology should be safe (from too low defect counts). In our real data set, we have 5771 defects, so we can deny the plausibility of this threat.

4 ACKNOWLEDGEMENT

What we will do in the remainder of this paper is difficult, as we will criticize published methodology that appears in MSR/ESE.

We first 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 clarity and originality. However, we believe that evaluating the importance of simulation-based testing for handling threats to validity, is not credible on unpublished examples. We need to continue on real studies.

The threats operationalized in the following examples are also relevant to other published studies, and just an excerpt of what can be done by simulation-based testing. We believe that reoccurring threats are an inevitable aspect of a scientific process, rather than a fault of the people who run into the threats.

Making such threats more explicit is a step towards improving the scientific process, to assure that future studies improve, and a contribution against the reproducibility crisis. The following sections may be considered as an initial catalog providing simulations of important threats to MSR/ESE studies.

5 DEPENDENT OBSERVATION (CASE 1)

After laying the foundations, we start examining a first publication using simulation-based testing. The study is by Alali et al. [23]. The work describes the 'typical commit'.

We believe that the major part of this study can be read without additional limitations. However, this study runs into a specific methodological threat, that we believe, is characteristic for the analysis of repository data. The authors make statements on uncertainty, ignoring the structure of the analyzed sample of commits.

5.1 Original Methodology

The original 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 statements done by the study are not necessary 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 [1].

However, in this case, the methodology requires statements on uncertainty because variables identified using the nine repositories might not exactly correspond to values in other unobserved repositories. A methodology can give such statements in terms of confidence estimates.

Alali et al. report on confidence, i.e., p-values for the correlation between the variables. In the remainder, we will adjust this and report on confidence intervals, since p-values are less intuitive for most readers. Threats for p-values are the same and can be shown the same way.

We assume that the original study computes p-values according to standard practice because the paper does not report on counteractions against dependent observations. In our reproduction of the original methodology, we also stick to standard practice[5].

5.2 Substitution A

The data of [23] is not available, so we go for a completely synthetic version of this study. We simplify the original research and examine the correlation between just two variables $X1$ and $X2$.

Listing 3 shows substitution 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*[6] with $rho = 0.2$.

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

[6]The details on how to simulate correlation are not relevant for understanding this case, but we want to allow copy-and-past, so we show it.

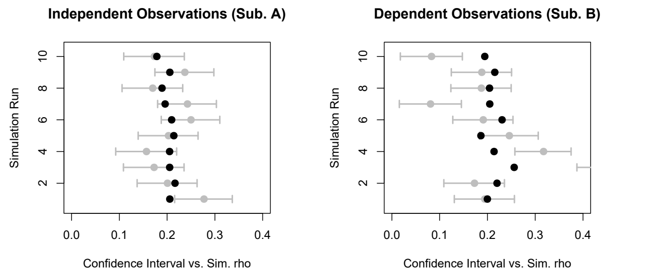

Figure 4: Ten simulation runs showing confidence intervals for rho, identified based on 9 repositories (gray line), and rho computed on all 100 repositories (black dot). We distinguish between independent (left) and dependent observations (right) produced according to substitutions A and B.

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.

5.3 Substitution B

The threat of dependent observations is added by a small modification to substitution A.

```
for (repo in 1:100) {
  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 substitution 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. 3.4.3. The rest of the substitution is the same as in the previous case.

5.4 Correspondence

The original methodology used by Alali et al. [23] to report on the uncertainty of correlation works under substitution A, but not under substitution B. 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 methodology 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 Figure 4, we see that the number of failing confidence intervals is different in both substitutions:

- Under *substitution A* (left), almost all confidence intervals include the correct correlation.
- Under *substitution B* (right), we see that in 4 out of 10 simulation runs, the methodology 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.

5.5 Plausibility and Conclusion

We have shown that dependency in observations is an evident threat for statements on uncertainty. Indicators for the plausibility of substitution B, compared to substitution 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 [11, 12]). Even if there is no dependency of correlation in real repositories, we can at least expect the methodology to be prepared for this threat.

We again want to emphasize that we only criticize statements on uncertainty, which are overall rare in [23]. However, we started with this case because we believe that the general problem of dependent observations and the impact on reported uncertainty is still underdeveloped in MSR/ESE research. Related threats might appear in several places when examining highly structured repository data. This may lead to study reproductions that are guaranteed to fail.

Generic advices on resolving this issue are difficult to give, but can be guided by evolving the presented simulation. In general, a resolution informs a model of the structured sampling process (e.g., see work on hierarchical/multilevel/mixed-effect models [24]).

6 PREDICTION OR CAUSATION (CASE 2)

The next case that we examine is an experience report on analytical defect modelling, done by Tantithamthavorn et al. in [25]. The report discusses challenges and actionable guidelines. We expect this report to have a big impact on our community.

We selected this work because of its progressive understanding of defect modeling to be more than just defect prediction. However, this work fears to name the actual challenge, i.e., *examining causation.*

We will accomplish this view by operationalizing threats of claiming causation, showing how to systematically extend the guidelines.

6.1 Original Methodology

The original work provides guidelines for other studies and thereby lists methodological steps exactly. 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 [25]). 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 [9]). Even more dramatic, 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.

We will show how claims on causation are possible within additional assumptions, operationalized by simulations. Often they are plausible in that we can claim causation within manageable threats.

6.2 Substitution C

Following the definition of Imbens et al. [9] (page 6), a claim for 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 substitution, 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. 3. 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.

6.3 Correspondence

According to our previous strategy of hiding synthetic but unobserved variables from the original methodology, we run the original methodology just using one of the potential outcomes, keeping the other hidden. However, we can still identify the causal relationship by the logistic regression under this substitution.

Multiple runs of the simulation, with different synthetic values for *beta*, shows that there is a clear correspondence between the identified *beta* on the observed variables, and the difference between both potential outcomes (see Figure 5).

6.4 Plausibility and Conclusion

This very basic simulation shows that under clear assumptions, claims on causation are possible. We are not the first who noticed this (for a summary of past work, see [9], page 23).

However, the assumptions of substitution C are tough: The effect of X (*beta*) is stable across all our observations. We assume that there is no dependency between observations. We assume that X is just a random variable, not influenced by anything else.

The critical assumption, that the variable X is not influenced by anything else, for instance, can be subject to further extensions to

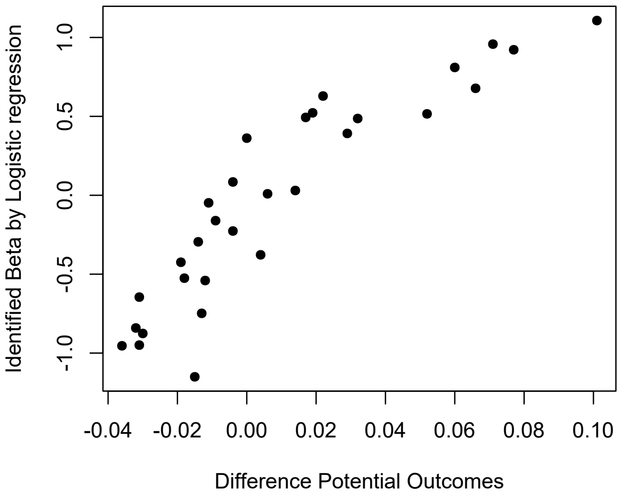

Figure 5: Identifying causation under strong assumptions (substitution C).

the simulation. A methodology may react by the control of variables. We do not show this, but evolving the simulation is straightforward. Hence, we can still formulate plausible claims on causation, even if we expect an influence.

Our simulation shows that a clean notion of assumptions, operationalized by simulations, can be a great help when talking about causation and corresponding threats. It can be used to extend the guide by Tantithamthavorn et al., being more precise on what the *interpretation of defect models* is.

7 CONTROL OF VARIABLES (CASE 3)

In [26], the authors propose a new metric intended to reflect a developers' ability to correctly understand the code. A basic statistic test shows a relation between the new metric and defects.

The study suggests the usage of such finding in prediction. However, a big part of the conclusions drawn in the study are of subliminal causal nature and prediction is not evaluated. We will focus on causation, but also conclude on prediction.

7.1 Original Methodology

The original study follows the standards in defect modeling as described in Sec. 3. It relates a novel experience metrics E to defects Y. The methodology is slightly different because the authors show the relation using a basic statistic test, rather than a logistic regression.

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*). This 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 cosine.

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

This is a serious threat as we know from works in MSR/ESE, e.g., [10–12], and from Sec. 3, 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.

7.2 Substitution D

This means that the original methodology needs to protect against such false conclusions. We can operationalize a test for a protection by a simulation. This substitution 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 synthetic file and background pairs using vectors synthesized by 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 substitution D.

Finally, the code synthesizes the probability *prob* and the defects Y, as we have done in the previous sections. In this substitution, we assume that there is no effect of E. If the methodology manages to detect no effect of E, we are safe, at least under substitution D.

7.3 Correspondence

However, the original methodology of [26] does not manage to handle substitution D correctly. The Mann-Whitney test used in the original study rejects the null-hypothesis in approximately 45 out of 100 simulation runs at a confidence level of 95%. This means that the result is almost comparable to flipping a coin. The reason gets clear when having a look at Figure 6, showing the strong positive relation between X and E. As expected, it is an artifact of the technical computation of the cosine. The very simplistic Mann-Whitney test accidentally attributed the effect of X to E.

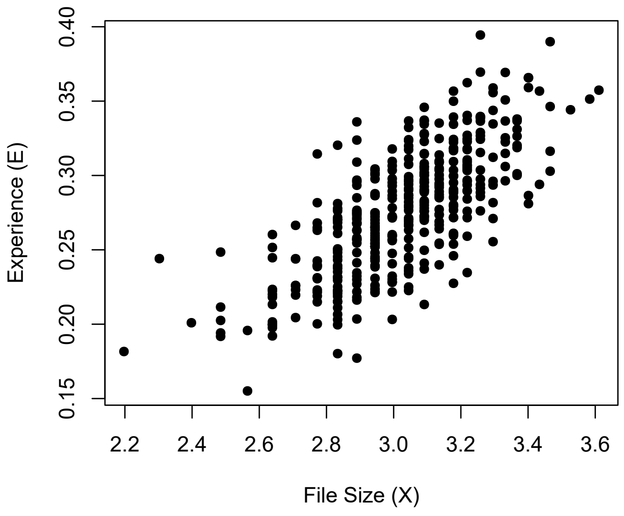

Figure 6: Relation between file size X and experience E computed by the cosine in a single simulation run.

7.4 Plausibility and Conclusion

The previous substitution shows how a methodology is technically not in position to correctly produce negative results under given assumptions. Substitution D is plausible enough in that we can demand a methodology to resolve it. Claiming that a developer-specific factor relates to defects, while it is just the file size, can hardly be justified as a novel insight. This holds for examining causation as well as for prediction.

We selected this work because it is an instance of applying an over-simplistic test. When working with observational data, one needs to be prepared for relationships to other variables, like a file's size. Basic statistic tests (whether parametrized or not) seldom suffice to answer research questions correctly on observational MSR/ESE data.

7.5 Revision

Indeed, there are ways to improve the original methodology. We want to 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 methodological threat. 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 synthetic data produced by substitution D. See the online resources extending the simulation for this insight.

We now apply the methodology (with and without control) to the real data and revise the original study. The relevant information on the logistic regression models can be found in Table 1. For further details, we refer to the reproduction code.

Table 1: A model with and without control, 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

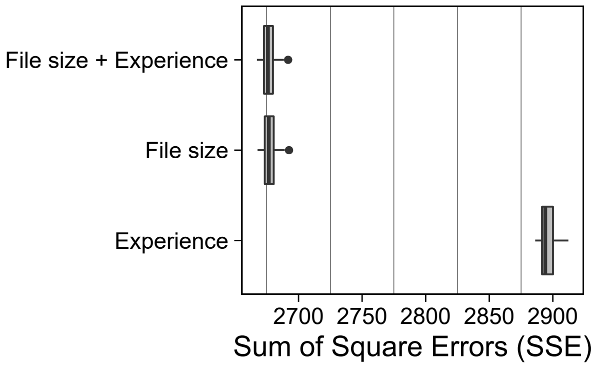

Figure 7: The performance impact of the new experience metric evaluated in cross-validation on the original data.

- 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 Figure 7.

A simulation would have discovered this threat with the computation of the cosine and the insufficient methodology early.

8 CORRELATED VARIABLES (CASE 4)

The last study that we examine is by Jiarpakdee et al. [15]. The study aims at methodological improvements when interpreting defect models with correlated variables. The authors motivate the practice of removing variables based on correlation or VIF thresholds (VIF [27]).

8.1 Original Methodology

In a nutshell, the original study compares model interpretation on a data set with and without correlated variables. The authors make statements on the practice of removing correlated variables, comparing properties of the identified variables between:

- different interpretation techniques and
- data sets with and without correlating variables.

We assume that it is sufficient to stay on this abstract level and refer to the original study for details.

The important pitfall of the original work is that it does not ask whether the identification of the unobserved variables is correct. In a simulation, it is easy to show that the practice of automated removal of correlated metrics does not improve the identification. It may even worsen it.

8.2 Substitution E

In the following, we focus on a fully synthetic 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.

8.3 Correspondence

We want to cover two scenarios: 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 Figure 8).

In this case, the original methodology of Jiarpakdee et al. should kick in. 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 scenario, 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 Figure 9. The model not including W and Z fails as it runs into confounding. The model, including all variables, succeeds like the model deciding for 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.

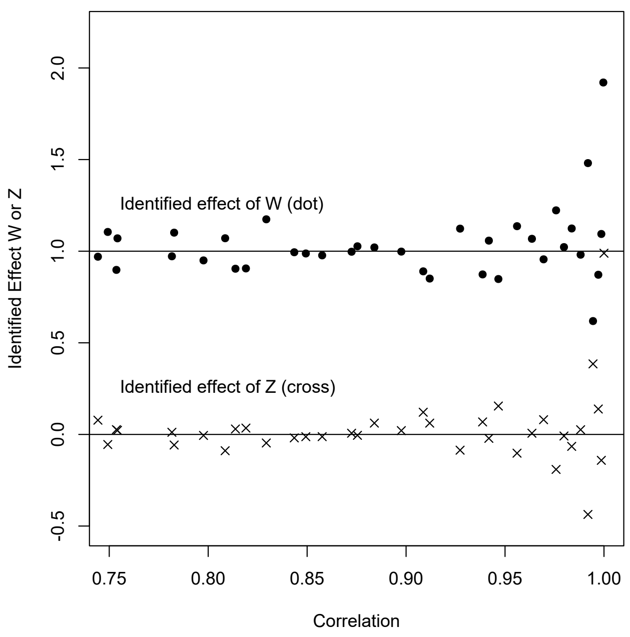

Figure 8: The identified effect of W and Z (which should be 1.0 and 0.0 respectively) under different correlation.

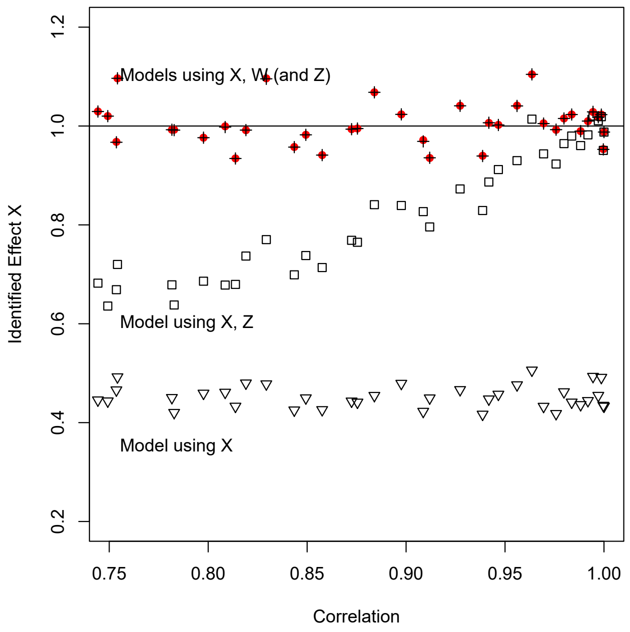

Figure 9: Models in simulation runs and the identified effect of X (which should be 1.0) under different correlation and different control variables.

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.

8.4 Plausibility and Conclusion

The substitution that we show is a very basic causal pattern. We can expect a study that recommends dropping correlated variables, with the aim to improve interpretation of defect models, to exactly resolve such issues. We do not expect the identification of variables to get worse.

We are not aware of any substitution where the recommended practice brings real benefits for an analysis. Even for prediction, dropping correlated variables just decreases performance (corresponding simulations are straightforward to implement).

We selected this work because we want to warn of the practice of automated dropping of variables, which should not become standard practice in MSR/ESE, especially when interpreting defect models. Which variables should be included, needs to be a conscious decision of the researcher. This conforms to recent statistic guidelines (e.g., see [20], page 169).

9 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, the use of simulation in other domains and test case generation in software engineering.

9.1 Empirical Studies

We are not aware of empirical studies in MSR/ESE that test a methodology by simulation and report on this. In the following, we discuss some other studies that may potentially benefit from simulation-based testing.

In [28], reasons for long duration builds in continuous integration pipelines are examined using multilevel models. Boh et al. [29] shows 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 methodology shown in our first case (Sec. 5). However, multilevel models are complicated. Our experience is that simulations can be a great help in testing and understanding how multilevel models react to the threat of a structured sampling process in MSR/ESE.

In [14], a multilevel (mixed-effect) defect model is fit, which is over-parameterized. One can see this with some experience, looking at the provided results in the paper. However, that the reported amount of data does not suffice to identify the variables can also be shown in a simulation.

Several works in MSR/ESE uses a methodology that assumes completely independent observations and thereby invokes threats (e.g., [26, 30, 31]). Such work may benefit from simulating the structured sampling process, and other reoccurring structural entities, like artifacts and developers, for detecting the potential dangers.

There is work discussing aggregation or disaggregation strategies on software engineering data [31, 32]. In simulations, it is easy to show that aggregation artificially increases correlation. Simulation-based tests may guide novel ideas on how to resolve issues with correlated variables (Sec. 8), potentially by disaggregated analysis of repository data.

Further, we assume that a series of work, relying on the well known SZZ algorithm [3], 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 can easily 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 effect. It may be resolved in parts by the control of variables (Sec. 7)

Bird at al. [33] examine the empirical challenges of incorrectly labeled bugs in historical defect data, which is an important threat to following up methodology. 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 assumption 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 (black-box of reality).

Authors of [34] report on the occurrence of well-known threats in existing literature. Opposed to a plain literature survey, simulation-based testing is a method to operationalize the relation between threats, assumptions, and methodology that are not yet well-known.

9.2 Simulation

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

In [35–37], 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. In such case, simulations are used to extrapolate, which is reasonable if configured with the right prior knowledge on unobserved variables. In [38], agent-based simulations for OS development are created using prior literature to set the relevant unobserved variables. In[39], multi-agent simulations predict next moves of agents. In [40], social coding dynamics are simulated based on historic data to forecast information spread.

In contrast to such work, our simulations operationalize threats. A simulation should be used to test empirical practice in MSR/ESE research, to spot cases where a methodology start to fail. Often, it is not clear how a methodology reacts to assumptions before seeing the consequences in a simulation.

9.3 Simulation in Other Domains

Statistic work evaluates cross-validation using a simulation study in [41]. In [42], cross-validation is evaluated on simulated structural data in the field of ecology. In [43], the impact of random effect structures is examined by simulation. The introduction to Bayesian statistics in [20] or regression modelling in [44] contain simulations as devices for illustration. In [45], 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 [46]). The authors of [47] discuss simulation studies in medicine. The evaluation of statistic methods by simulation is discussed by [48] (also in medicine). In [49, 50], authors discuss the role of simulation in learning statistics. We discuss the role of simulations in operationalizing threats specific to MSR/ESE studies.

9.4 Test Case Generation

Another related area is software testing, in particular, the generation of test cases. For instance, grammars have been used to enumerate test cases for compilers and language infrastructure [51–53]. Generating instances of metamodels to test corresponding methods can be found in [54].

Simulation-based testing and test generation have in common that the test space is highly parametric. Both try to systematically spot the weak points in such space where methods potentially fail.

10 CONCLUSION

In this paper, we introduce simulation-based testing to operationalize the treatment of threats to validity, targeting the methodology of quantitative studies in MSR/ESE. A simulation substitutes observed and unobserved data, related to an MSR/ESE scenario, with synthetic data, carefully defined according to plausible assumptions on the scenario. Within a simulation, unobserved data becomes transparent, which is the key difference to real studies, necessary to detect threats to an analysis methodology.

We illustrate that simulation-based testing contributes to handling threats to validity in four published MSR/ESE studies, using very basic substitutions that, we believe, are plausible in MSR/ESE. The simulations show that the original studies need to be improved, assuming that our substitutions are plausible. In the end, a simulation-based test is only as strong as its plausibility. This needs to be rated by the reader or reviewer and is not the central objective of this work.

We believe that simulation-based testing can assist future research in MSR/ESE to resolve very persistent threats to validity, that appear in several studies, by capturing them in an operationalized and systematic manner. Our future work will continue being verified by simulation-based tests. We suggest equipping more studies with simulations as an additional quality check. Making such threats more explicit is a step towards improving the scientific process, to assure that future studies improve, and a contribution against the reproducibility crisis.

REFERENCES

[1] V. Cosentino, J. L. C. Izquierdo, and J. Cabot, "Findings from GitHub: methods, datasets and limitations," in *MSR.* ACM, 2016, pp. 137–141.

[2] A. E. Hassan, "Mining Software Repositories to Assist Developers and Support Managers," in *ICSM.* IEEE Computer Society, 2006, pp. 339–342.

[3] J. Sliwerski, T. Zimmermann, and A. Zeller, "When do changes induce fixes?" in *MSR.* ACM, 2005.

[4] R. N. Iyer, S. A. Yun, M. Nagappan, and J. Hoey, "Effects of Personality Traits on Pull Request Acceptance," *IEEE Transactions on Software Engineering*, 2019.

[5] M. M. Rahman, C. K. Roy, and J. A. Collins, "CoRReCT: code reviewer recommendation in GitHub based on cross-project and technology experience," in *ICSE (Companion Volume).* ACM, 2016, pp. 222–231.

[6] A. Jbara and D. G. Feitelson, "On the effect of code regularity on comprehension," in *ICPC.* ACM, 2014, pp. 189–200.

[7] J. Härtel, H. Aksu, and R. Lämmel, "Classification of APIs by hierarchical clustering," in *ICPC.* ACM, 2018, pp. 233–243.

[8] J. Härtel, M. Heinz, and R. Lämmel, "EMF Patterns of Usage on GitHub," in *ECMFA*, ser. Lecture Notes in Computer Science, vol. 10890. Springer, 2018, pp. 216–234.

[9] G. W. Imbens and D. B. Rubin, *Causal inference in statistics, social, and biomedical sciences.* Cambridge University Press, 2015.

[10] A. Mockus, "Organizational volatility and its effects on software defects," in *SIGSOFT FSE.* ACM, 2010, pp. 117–126.

[11] Y. Kamei, E. Shihab, B. Adams, A. E. Hassan, A. Mockus, A. Sinha, and N. Ubayashi, "A Large-Scale Empirical Study of Just-in-Time Quality Assurance," *IEEE Trans. Software Eng.*, vol. 39, no. 6, pp. 757–773, 2013.

[12] F. Falcão, C. Barbosa, B. Fonseca, A. Garcia, M. Ribeiro, and R. Gheyi, "On Relating Technical, Social Factors, and the Introduction of Bugs," in *SANER.* IEEE, 2020, pp. 378–388.

[13] A. Mockus and D. M. Weiss, "Predicting risk of software changes," *Bell Labs Technical Journal*, vol. 5, no. 2, pp. 169–180, 2000.

[14] M. Yan, X. Xia, Y. Fan, D. Lo, A. E. Hassan, and X. Zhang, "Effort-aware just-in-time defect identification in practice: a case study at Alibaba," in *ESEC/SIGSOFT FSE.* ACM, 2020, pp. 1308–1319.

[15] J. Jiarpakdee, C. Tantithamthavorn, and A. E. Hassan, "The Impact of Correlated Metrics on the Interpretation of Defect Models," *IEEE Trans. Software Eng.*, vol. 47, no. 2, pp. 320–331, 2021.

[16] N. Nagappan, A. Zeller, T. Zimmermann, K. Herzig, and B. Murphy, "Change Bursts as Defect Predictors," in *ISSRE.* IEEE Computer Society, 2010, pp. 309–318.

[17] T. Zimmermann, R. Premraj, and A. Zeller, "Predicting defects for eclipse," in *PROMISE 2007.* IEEE, 2007, p. 76.

[18] T. Zimmermann and N. Nagappan, "Predicting defects using network analysis on dependency graphs," in *ICSE.* ACM, 2008, pp. 531–540.

[19] P. Thongtanunam, S. McIntosh, A. E. Hassan, and H. Iida, "Revisiting code ownership and its relationship with software quality in the scope of modern code review," in *ICSE.* ACM, 2016, pp. 1039–1050.

[20] R. McElreath, *Statistical rethinking: A Bayesian course with examples in R and Stan.* CRC press, 2020.

[21] M. Tan, L. Tan, S. Dara, and C. Mayeux, "Online Defect Prediction for Imbalanced Data," in *ICSE (2).* IEEE Computer Society, 2015, pp. 99–108.

[22] P. Peduzzi, J. Concato, E. Kemper, T. R. Holford, and A. R. Feinstein, "A simulation study of the number of events per variable in logistic regression analysis," *Journal of clinical epidemiology*, vol. 49, no. 12, pp. 1373–1379, 1996.

[23] A. Alali, H. H. Kagdi, and J. I. Maletic, "What's a Typical Commit? A Characterization of Open Source Software Repositories," in *ICPC.* IEEE Computer Society, 2008, pp. 182–191.

[24] A. Gelman and J. Hill, *Data analysis using regression and multilevel/hierarchical models.* Cambridge university press, 2006.

[25] C. Tantithamthavorn and A. E. Hassan, "An experience report on defect modelling in practice: pitfalls and challenges," in *ICSE (SEIP).* ACM, 2018, pp. 286–295.

[26] M. Tufano, G. Bavota, D. Poshyvanyk, M. D. Penta, R. Oliveto, and A. D. Lucia, "An empirical study on developer-related factors characterizing fix-inducing commits," *J. Softw. Evol. Process.*, vol. 29, no. 1, 2017.

[27] J. Cohen, P. Cohen, S. G. West, and L. S. Aiken, *Applied multiple regression/correlation analysis for the behavioral sciences.* Routledge, 2013.

[28] T. A. Ghaleb, D. A. da Costa, and Y. Zou, "An empirical study of the long duration of continuous integration builds," *Empirical Software Engineering*, vol. 24, no. 4, pp. 2102–2139, 2019.

[29] W. F. Boh, S. Slaughter, and J. A. Espinosa, "Learning from experience in software development: A multilevel analysis," *Manag. Sci.*, vol. 53, no. 8, pp. 1315–1331, 2007.

[30] F. Rahman and P. T. Devanbu, "Ownership, experience and defects: a fine-grained study of authorship," in *ICSE.* ACM, 2011, pp. 491–500.

[31] F. Zhang, A. E. Hassan, S. McIntosh, and Y. Zou, "The use of summation to aggregate software metrics hinders the performance of defect prediction models," *IEEE Trans. Software Eng.*, vol. 43, no. 5, pp. 476–491, 2017.

[32] K. Herzig and A. Zeller, "The impact of tangled code changes," in *MSR.* IEEE Computer Society, 2013, pp. 121–130.

[33] C. Bird, A. Bachmann, E. Aune, J. Duffy, A. Bernstein, V. Filkov, and P. T. Devanbu, "Fair and balanced?: bias in bug-fix datasets," in *ESEC/SIGSOFT FSE.* ACM, 2009, pp. 121–130.

[34] R. P. Reyes, O. Dieste, E. R. F. C., and N. Juristo, "Statistical errors in software engineering experiments: a preliminary literature review," in *ICSE.* ACM, 2018, pp. 1195–1206.

[35] V. Honsel, D. Honsel, and J. Grabowski, "Software Process Simulation Based on Mining Software Repositories," in *ICDM Workshops.* IEEE Computer Society, 2014, pp. 828–831.

[36] V. Honsel, D. Honsel, S. Herbold, J. Grabowski, and S. Waack, "Mining Software Dependency Networks for Agent-Based Simulation of Software Evolution," in *ASE Workshops.* IEEE Computer Society, 2015, pp. 102–108.

[37] V. Honsel, "Statistical Learning and Software Mining for Agent Based Simulation of Software Evolution," in *ICSE (2).* IEEE Computer Society, 2015, pp. 863–866.

[38] T. Seo and H. Lee, "Agent-based Simulation Model for the Evolution Process of Open Source Software," in *SEKE.* Knowledge Systems Institute Graduate School, 2009, pp. 170–177.

[39] J. Blythe, J. Bollenbacher, D. Huang, P. Hui, R. Krohn, D. Pacheco, G. Muric, A. Sapienza, A. Tregubov, Y. Ahn, A. Flammini, K. Lerman, F. Menczer, T. Weninger, and E. Ferrara, "Massive Multi-agent Data-Driven Simulations of the GitHub Ecosystem," in *PAAMS*, ser. Lecture Notes in Computer Science, vol. 11523. Springer, 2019, pp. 3–15.

[40] N. H. Bidoki, M. Schiappa, G. Sukthankar, and I. Garibay, "Modeling social coding dynamics with sampled historical data," *Online Soc. Networks Media*, vol. 16, p. 100070, 2020.

[41] J. Shao, "Linear model selection by cross-validation," *Journal of the American statistical Association*, vol. 88, no. 422, pp. 486–494, 1993.

[42] D. R. Roberts, V. Bahn, S. Ciuti, M. S. Boyce, J. Elith, G. Guillera-Arroita, S. Hauenstein, J. J. Lahoz-Monfort, B. Schröder, W. Thuiller, D. I. Warton, B. A. Wintle, F. Hartig, and C. F. Dormann, "Cross-validation strategies for data with temporal, spatial, hierarchical, or phylogenetic structure," *Ecography*, vol. 40, no. 8, pp. 913–929, 2017.

[43] D. J. Barr, R. Levy, C. Scheepers, and H. J. Tily, "Random effects structure for confirmatory hypothesis testing: Keep it maximal," *Journal of Memory and Language*, vol. 368, no. 3, pp. 255–278, 2013.

[44] F. E. Harrell, *Regression modeling strategies: with applications to linear models, logistic and ordinal regression, and survival analysis.* Springer, 2015, vol. 2.

[45] A. Gasparini, K. R. Abrams, J. K. Barrett, R. W. Major, M. J. Sweeting, N. J. Brunskill, and M. J. Crowther, "Mixed-effects models for health care longitudinal data with an informative visiting process: A Monte Carlo simulation study," *Statistica Neerlandica*, vol. 74, no. 1, pp. 5–23, 2020.

[46] J. Härtel and R. Lämmel, "Incremental Map-Reduce on Repository History," in *SANER.* IEEE, 2020, pp. 320–331.

[47] A. Burton, D. G. Altman, P. Royston, and R. L. Holder, "The design of simulation studies in medical statistics," *Statistics in Medicine*, vol. 25, no. 24, pp. 4279–4292, 2006.

[48] T. P. Morris, I. R. White, and M. J. Crowther, "Using simulation studies to evaluate statistical methods," *Statistics in Medicine*, vol. 38, no. 11, pp. 2074–2102, 2019.

[49] D. M. Jamie, "Using computer simulation methods to teach statistics: A review of the literature," *Journal of Statistics Education*, vol. 10, no. 1, 2002.

[50] M. Wood, "The role of simulation approaches in statistics," *Journal of Statistics Education*, vol. 13, no. 3, 2005.

[51] C. J. Burgess, "The Automated Generation of Test Cases for Compilers," *Softw. Test. Verification Reliab.*, vol. 4, no. 2, pp. 81–99, 1994.

[52] A. S. Kossatchev and M. Posypkin, "Survey of compiler testing methods," *Program. Comput. Softw.*, vol. 31, no. 1, pp. 10–19, 2005.

[53] J. Härtel, L. Härtel, and R. Lämmel, "Test-Data Generation for Xtext," in *SLE*, ser. Lecture Notes in Computer Science, vol. 8706. Springer, 2014, pp. 342–351.

[54] E. Brottier, F. Fleurey, J. Steel, B. Baudry, and Y. L. Traon, "Metamodel-based Test Generation for Model Transformations: an Algorithm and a Tool," in *ISSRE.* IEEE Computer Society, 2006, pp. 85–94.