Operationalizing Threats to MSR Studies by Simulation-Based Testing

Johannes Härtel and Ralf Lämmel
University of Koblenz, Germany
Software Languages Team

Johannes Härtel and Ralf Lämmel: Operationalizing Threats to MSR Studies by Simulation-Based Testing. MSR 2022

https://conf.researchr.org/program/msr-2022/program-msr-2022/

Technical Papers

Wed 18 May 2022 05:04 - 05:11 at MSR room - Session 3: Introspection, Vision, and Human Aspects Chair(s): Alexander Serebrenik, Sebastian Baltes

☆ Operationalizing Threats to MSR Studies by Simulation-Based Testing

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.

DISTINGUISHED PAPER AWARD

Pre-print

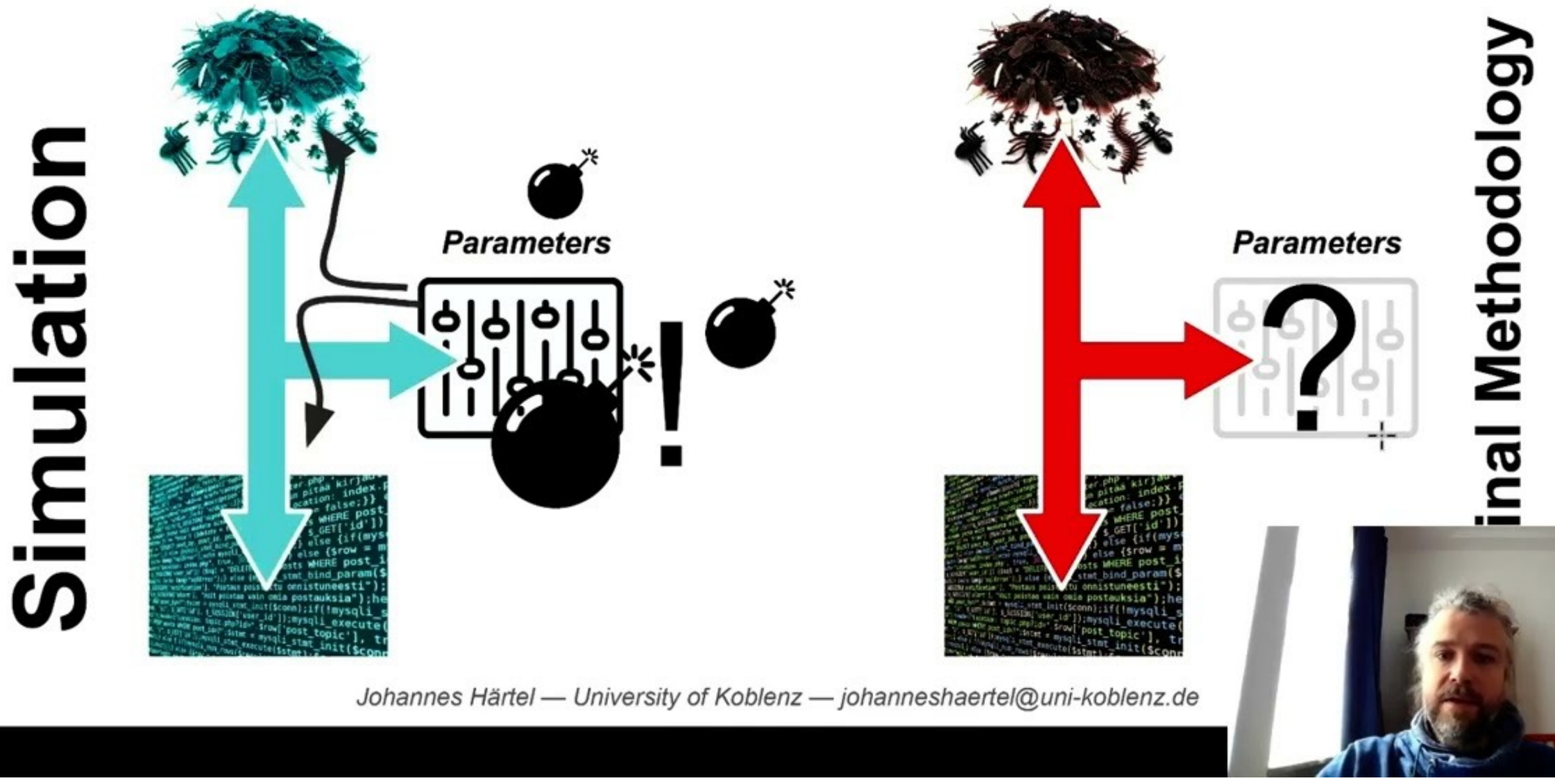
Johannes Härtel
University of Koblenz-Landau, Germany

Ralf Laemmel
Facebook London

Johannes Härtel and Ralf Lämmel: Operationalizing Threats to MSR Studies by Simulation-Based Testing. MSR 2022

Symmetry (3)

We use relationships in revers to produce **artificial defects and code metrics**.



Johannes Härtel and Ralf Lämmel: Operationalizing Threats to MSR Studies by Simulation-Based Testing. MSR 2022

Motivation

- Empirical questions on SE can be answered by analyzing repository data.
- The methodology of such studies is often very complicated.
- ***How do we know that the methodology and answers are “correct”?***

Background

Existing ideas to assure “correct” methodology and answers:

- **Our intuition** as a software developer helps us to judge answers.
- **Meta-analysis** checks the consistency of answers with previous work.
- We can **stick to** the methodology of **prior studies**.
- **Model comparison** with a protection against under- and overfitting helps.

Background

Software developers have some **intuition** on “correct” answers.

Commits Bugginess (F4): Inevitably, during the software development process, developers make changes that introduce bugs [2]. In this context, we analyze if the commits of developers who have previously introduced bugs are more likely to be buggy. To measure this factor, we evaluate the *Percentage of Buggy Commits (PBC)* previously authored by a developer. Our intuition is that *the higher the value of this metric, the more “harmful” developer’s commits are*. The

This sounds reasonable for a software developer.

Source: F. Falcão, C. Barbosa, B. Fonseca, A. Garcia, M. Ribeiro, R. Gheyi: “**On Relating Technical, Social Factors, and the Introduction of Bugs**”, in SANER, 2020.

Background

A **meta-analysis** provides a useful idea of "correct" answers.

Class	Predictor	Effect	Propstns	Reproduced
Org. chng	Size of org.	38%	control	+[17]
	From prior (yrs)	-15%	+1	new
	Until next (yrs)	-4%	+1	new
	Left	26%	+4	new
	Newcomers	N/A	−3,2	new
File	LOC	34%	+5	+various
Change	Logical Deps.	11%	+6	+[6, 5]
	Release Deps.	192%	+10	+[11]
	Change Diffusion	6%	+6	+[16]
Social	Workflow Deps.	35%	+7	+[6, 5, 11]
	Experience (yrs)	18%	+8	−[16]
Geo.	Distributed	15%	+9	−[4],+[12]
	Mentor offshore	69%	+9	new

One may check how results align with previous studies. The +/- signs indicate on successful reproduction.

However, there are still conflicts (-).

Source of excerpt: A. Mockus: "**Organizational volatility and its effects on software defects**", in SIGSOFT FSE, 2010.

Background

Prior studies provide a useful idea of a “correct” methodology.

> *Step 2: Correlation and redundancy analysis.* Highly correlated and redundant features would produce incorrect interpretations for a mixed effect model as noted by prior studies [10, 35]. In this step, we remove the highly correlated and redundant change factors

This is clear advice for a methodology (Caution: Our paper and recent statistic work will give different advice on this).

Source of excerpt: M. Yan, X. Xia, Y. Fan, D. Lo, A. E. Hassan, and X. Zhang, “**Effort-aware just-intime defect identification in practice: a case study at Alibaba**”, in ESEC/SIGSOFT FSE, 2020.

[10]: S. Hassan, C. Tantithamthavorn, C. Bezemer, and A. E. Hassan: “**Studying the dialogue between users and developers of free apps in the google play store**”, in ESE, 2018.

[35]: M. Shepperd, D. Bowes, and T. Hall: “**Researcher bias: The use of machine learning in software defect prediction**”, TSE, 2014.

Background

Model comparison, cross-validation, information criteria and regularization, provides a useful idea of “correct” answers.

Fitting models may still go wrong.

Authors compare models.

Table 2. Multi-level mixed effects logistic model for pull request acceptance

		Model I	Model II	Model III	Model IV
		Pull Request Level		Pull + Submitter Level	Pull + Submitter + Repo Level
Factor	**Variable**	**Odds Ratio**	**Odds Ratio**	**Odds Ratio**	**Odds Ratio**
	(Intercept)	2.934 ***	2.898 ***	2.845 ***	3.925 ***
Technical Contribution Norms (H1)	Test Inclusion	1.059 ***	1.023 *	1.114 ***	1.171 ***
	Commit Size	0.849 ***	0.834 ***	0.736 ***	0.738 ***
	Number of Files Changed	1.165 ***	1.152 ***	0.970 ***	0.927 ***
Social Connection (H2)	Social Distance	1.345 ***	1.461 ***	3.636 ***	2.870 ***
	Prior Interaction	1.423 ***	1.362 ***	1.207 ***	1.356 ***
AIC:		633600	630879	506850	461077

This selection may favor non-causal models.

AIC (an information criterion) is used.

Source of excerpt: J. Tsay, L. Dabbish, and J. Herbsleb: **"Influence of social and technical factors for evaluating contribution in GitHub"** in *ICSE*, 2014.

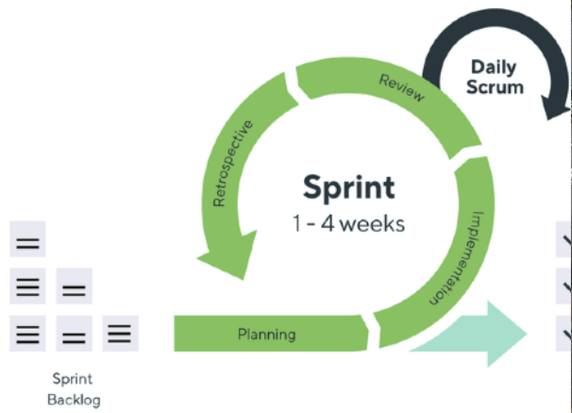
We will present a fresh idea on this topic.

We start with recapitulating typical methodology in MSR/ESE.

Johannes Härtel and Ralf Lämmel: Operationalizing Threats to MSR Studies by Simulation-Based Testing. MSR 2022

The typical methodology

Variables interesting for MSR/ESE



Johannes Härtel and Ralf Lämmel: Operationalizing Threats to MSR Studies by Simulation-Based Testing. MSR 2022

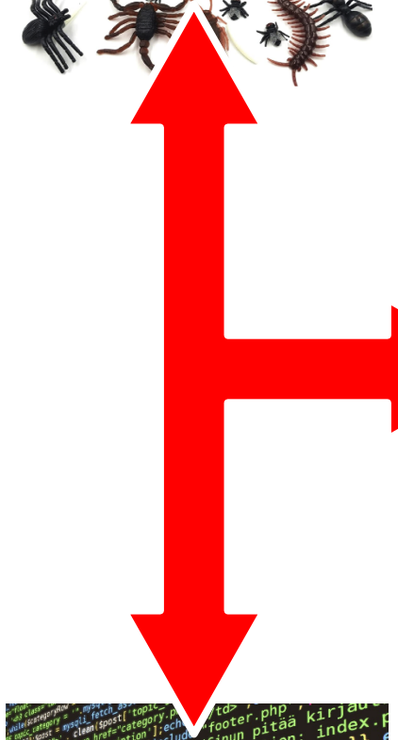
The typical methodology

Relationships between variables interesting for MSR/ESE

Johannes Härtel and Ralf Lämmel: Operationalizing Threats to MSR Studies by Simulation-Based Testing. MSR 2022

The typical methodology

Models interesting for MSR/ESE



Parameters are variables too, e.g.,:

- **effect strength,**
- **variance,**
- **or correlation.**

Parameters

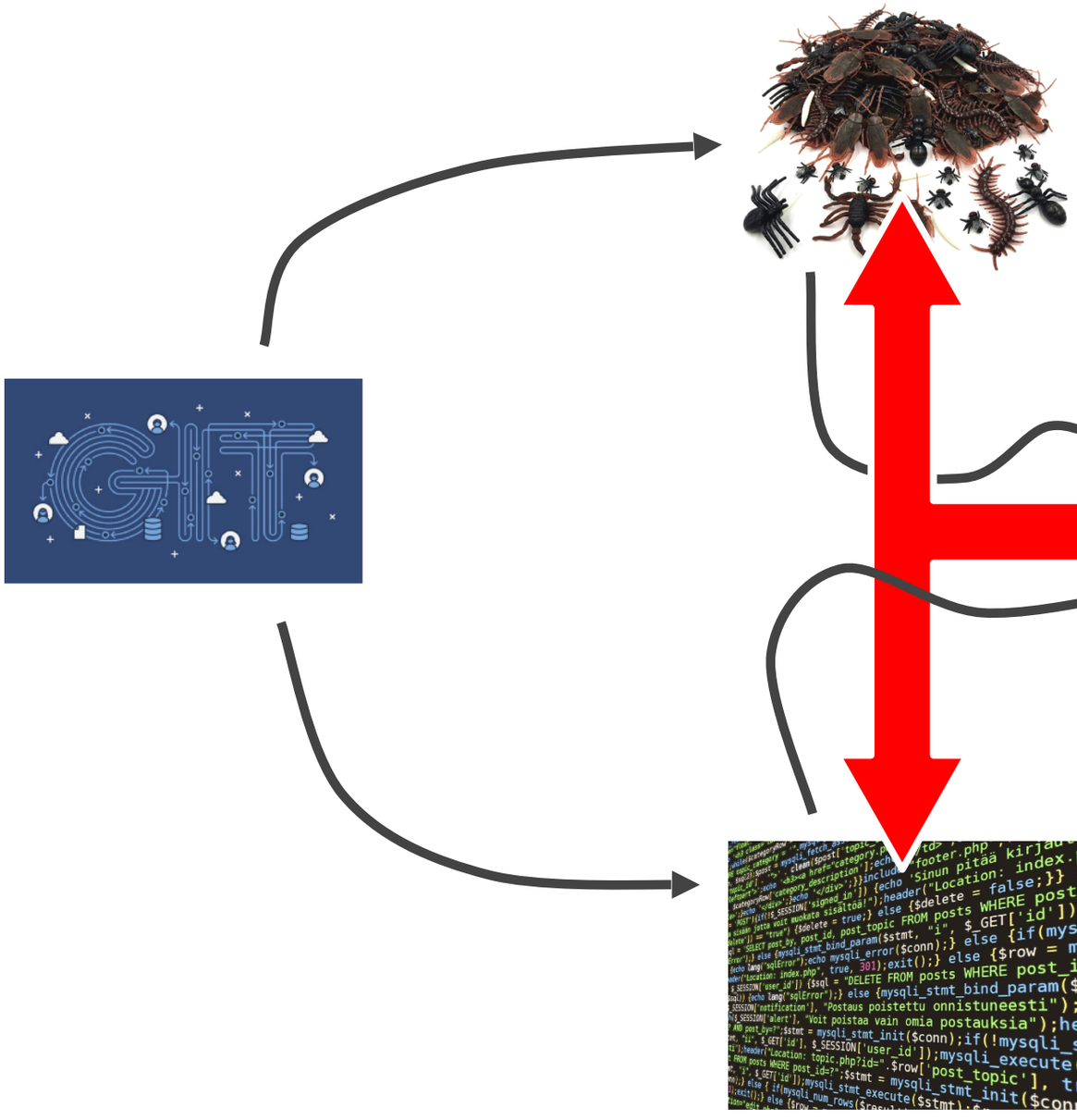
Johannes Härtel and Ralf Lämmel: Operationalizing Threats to MSR Studies by Simulation-Based Testing. MSR 2022

Different perspectives on MSR/ESE models

(Setting the stage for simulation-based testing)

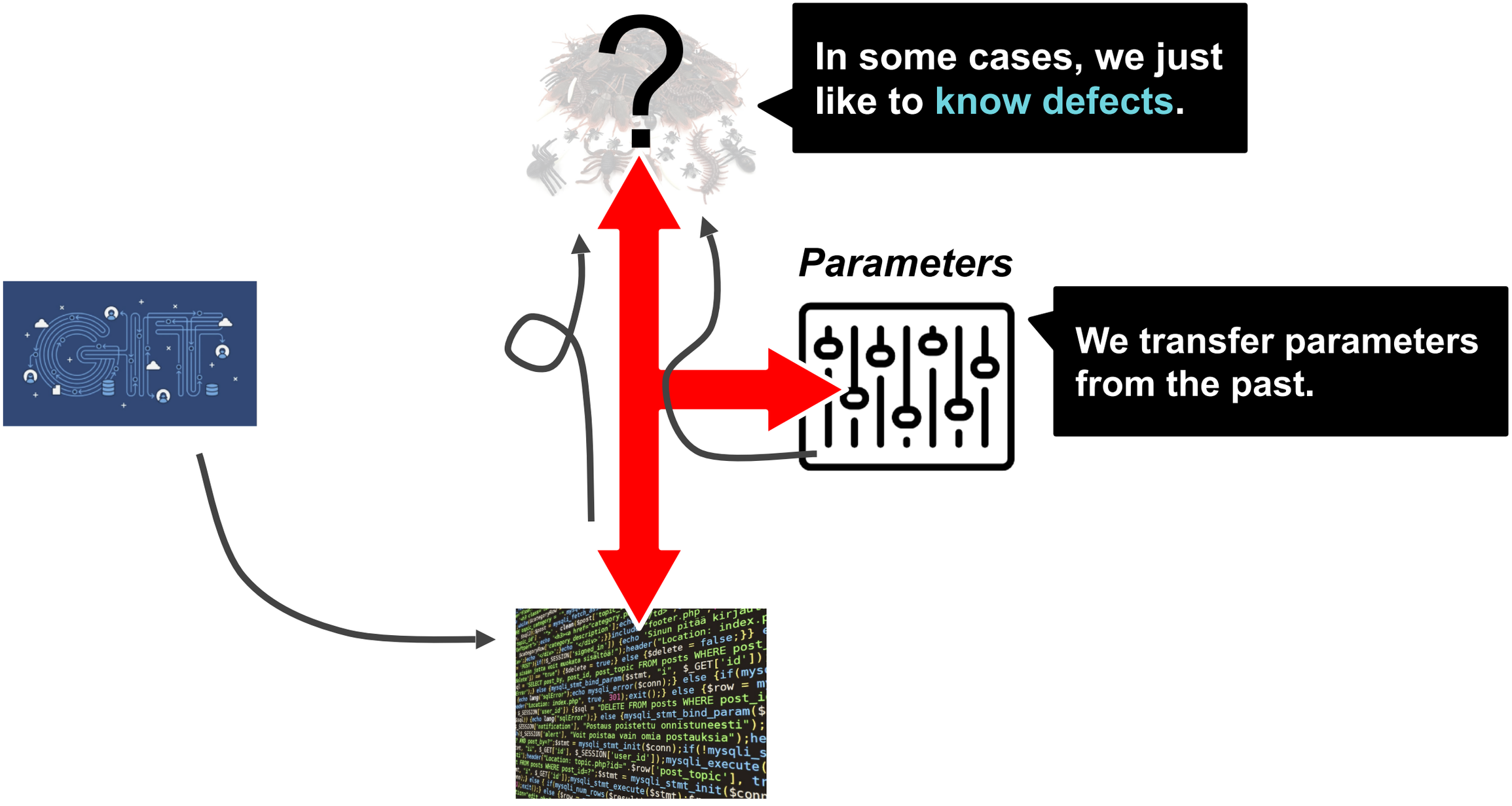
Johannes Härtel and Ralf Lämmel: Operationalizing Threats to MSR Studies by Simulation-Based Testing. MSR 2022

Parameter inference



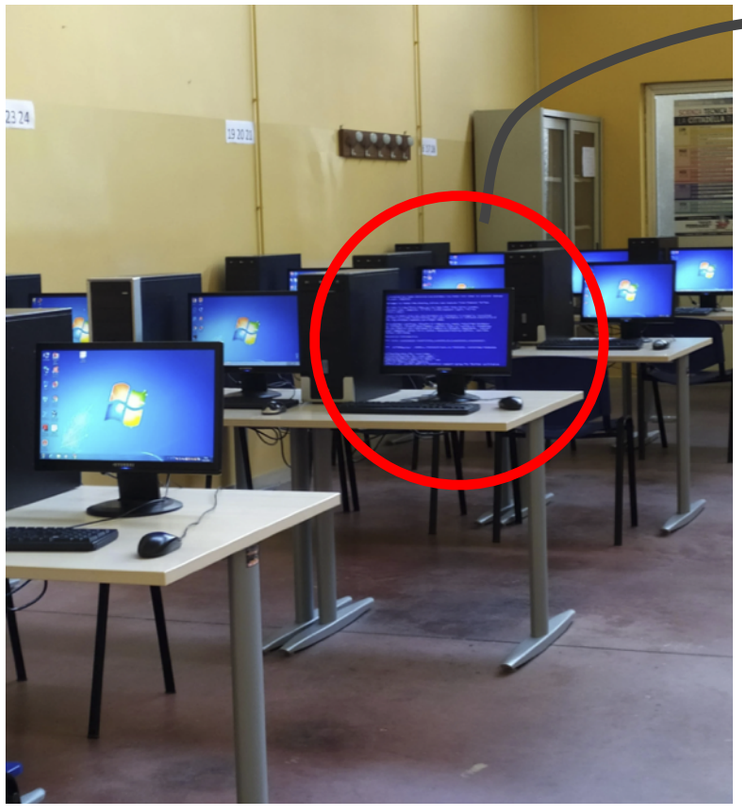
Johannes Härtel and Ralf Lämmel: Operationalizing Threats to MSR Studies by Simulation-Based Testing. MSR 2022

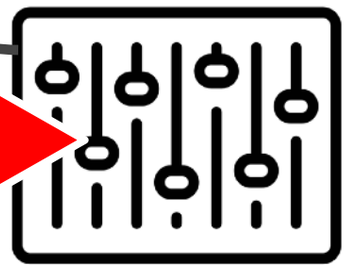
Predict data



Johannes Härtel and Ralf Lämmel: Operationalizing Threats to MSR Studies by Simulation-Based Testing. MSR 2022

Predict (?) data



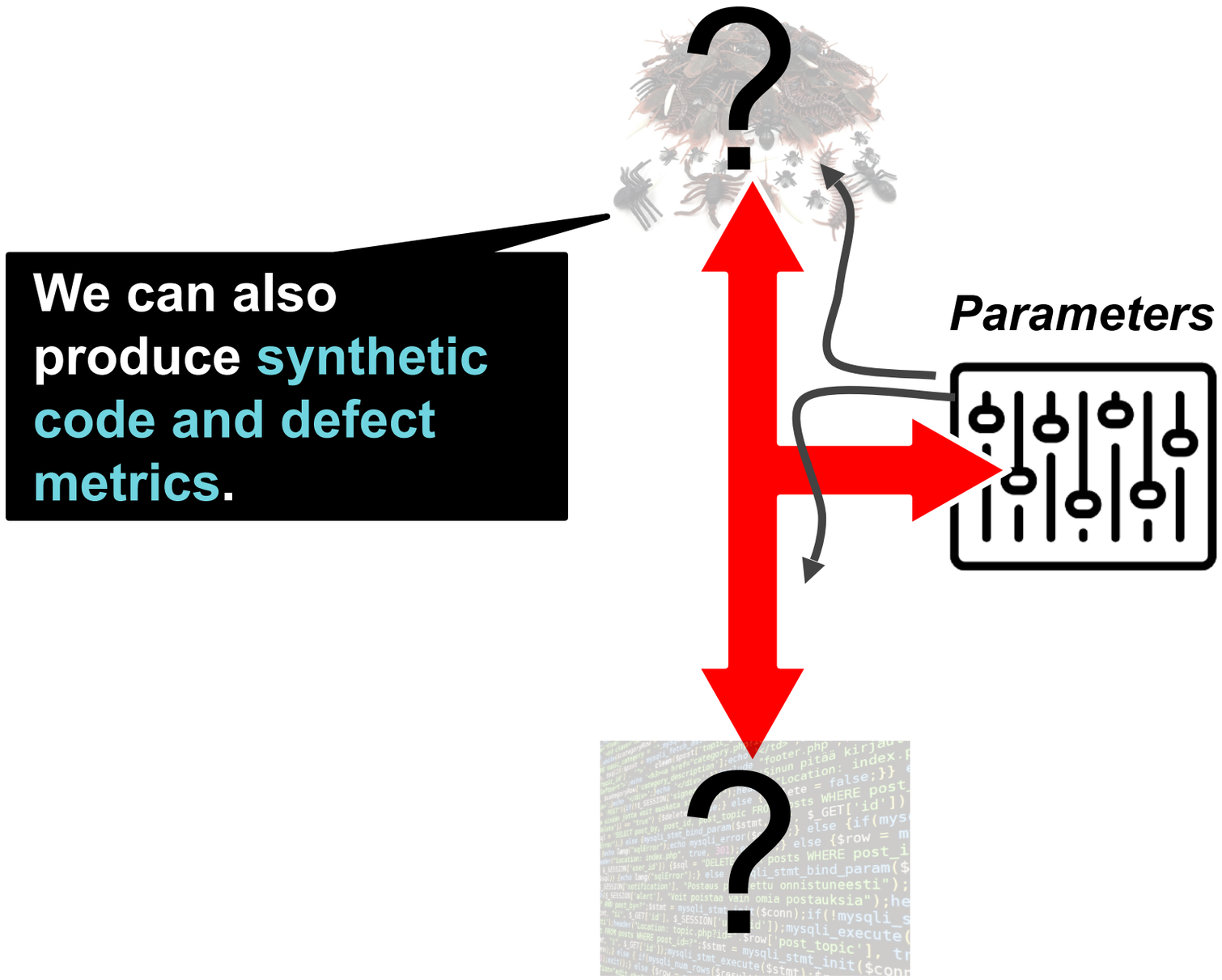


Parameters

While never seen the code, we still have an impression of its complexity because we see defects.

Johannes Härtel and Ralf Lämmel: Operationalizing Threats to MSR Studies by Simulation-Based Testing. MSR 2022

Synthesize data



Johannes Härtel and Ralf Lämmel: Operationalizing Threats to MSR Studies by Simulation-Based Testing. MSR 2022

Simulation-based testing of methodologies

Johannes Härtel and Ralf Lämmel: Operationalizing Threats to MSR Studies by Simulation-Based Testing. MSR 2022

Expected **correspondence** between two workflows

Simulation

This is made up.

Parameters

This is Reality.

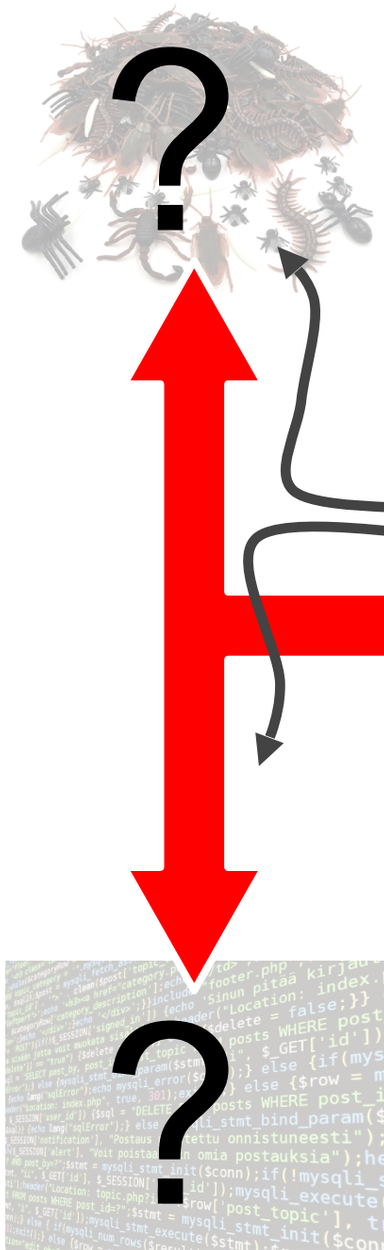
Parameters

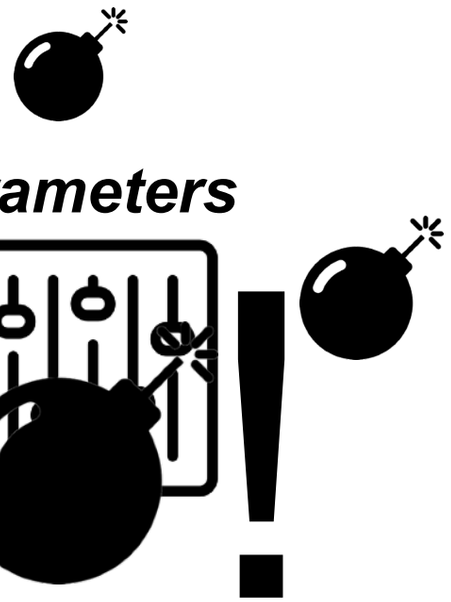
Original Methodology

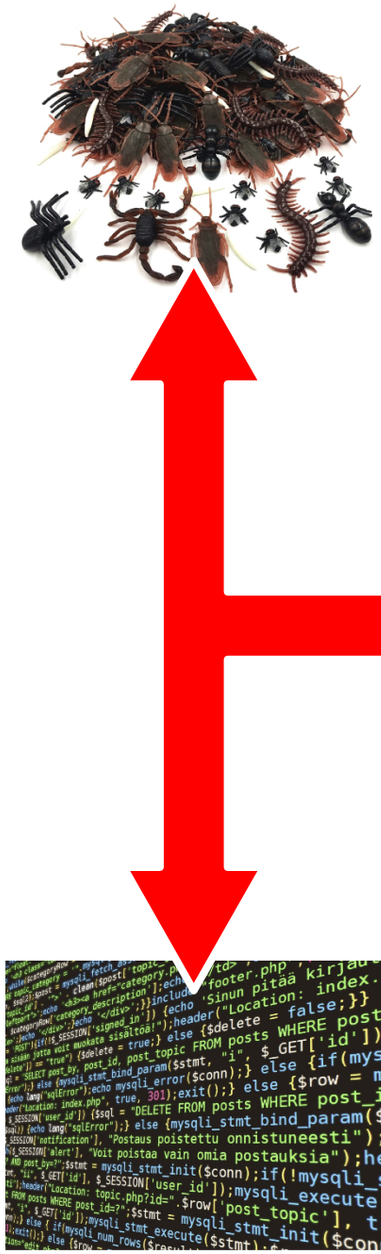
Johannes Härtel and Ralf Lämmel: Operationalizing Threats to MSR Studies by Simulation-Based Testing. MSR 2022

Set plausible but challenging **simulation** parameters

Simulation







Original Methodology

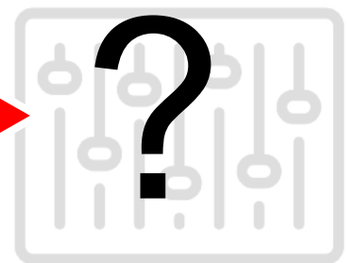
Use relationships in **reverse** to produce **synthetic** data

Simulation

Parameters

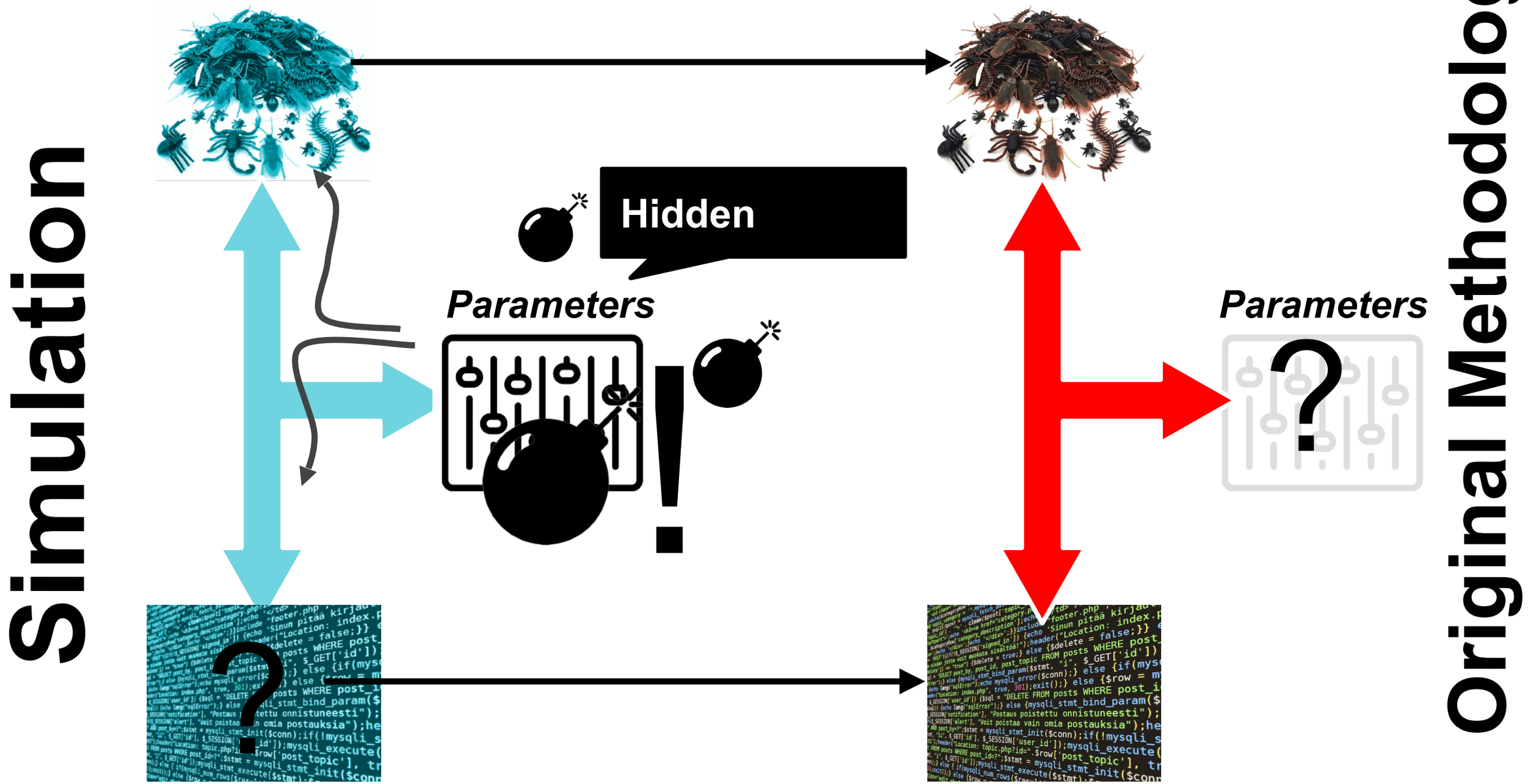
Original Methodology

Parameters



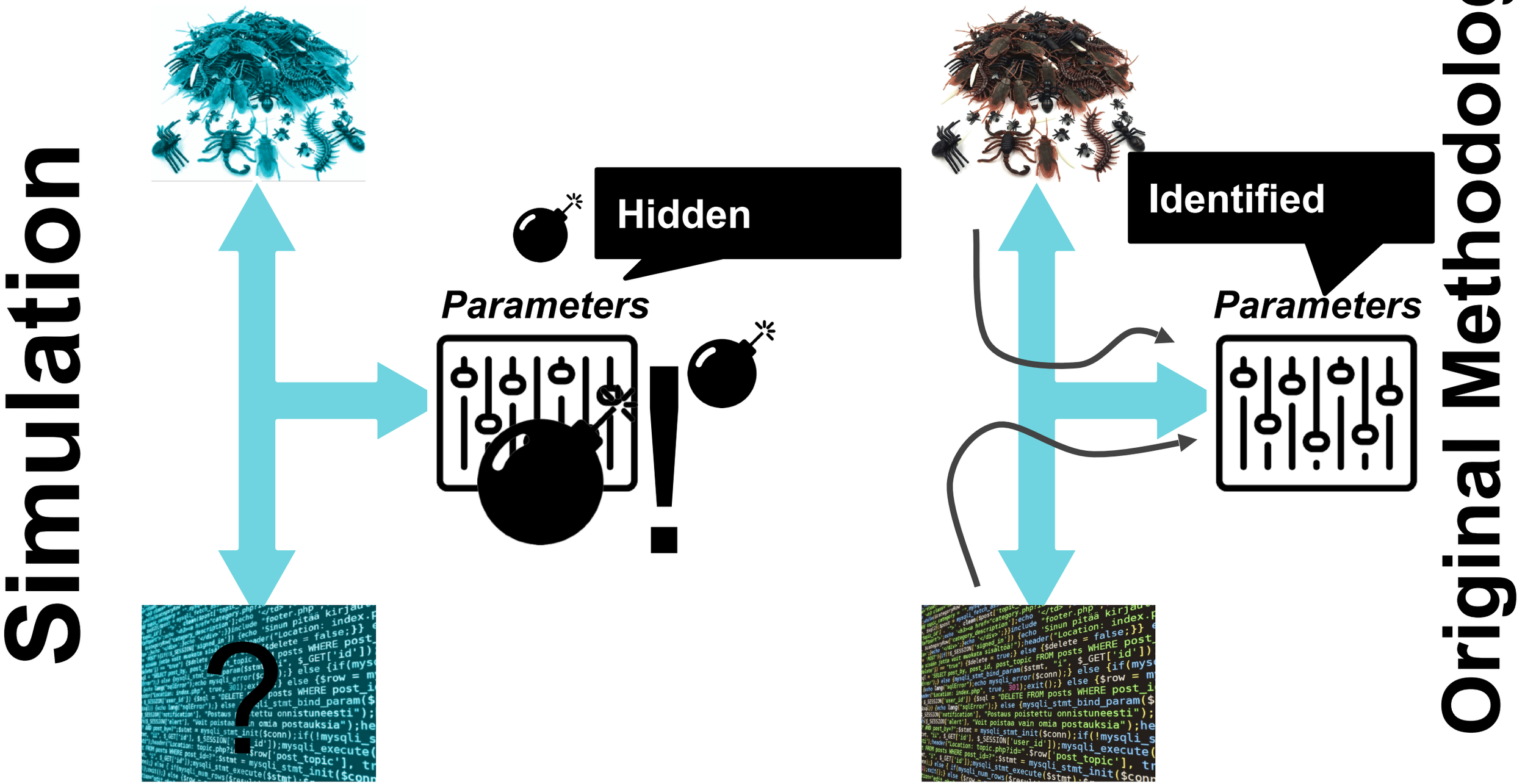
Johannes Härtel and Ralf Lämmel: Operationalizing Threats to MSR Studies by Simulation-Based Testing. MSR 2022

Substitute real variables by synthetic data except parameters



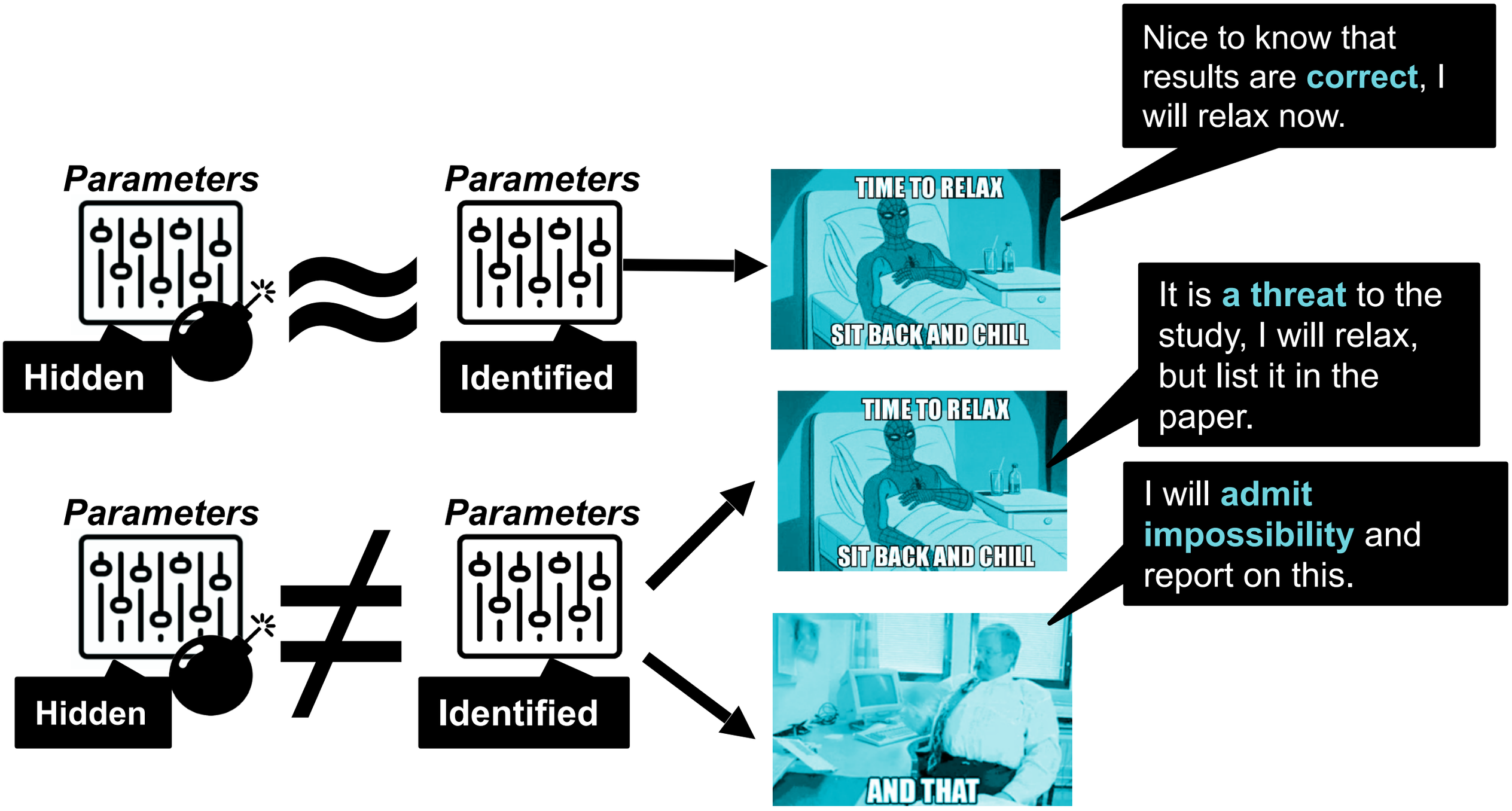
Johannes Härtel and Ralf Lämmel: Operationalizing Threats to MSR Studies by Simulation-Based Testing. MSR 2022

Run original methodology to **identify parameters again**



Johannes Härtel and Ralf Lämmel: Operationalizing Threats to MSR Studies by Simulation-Based Testing. MSR 2022

Check **correspondence**



Johannes Härtel and Ralf Lämmel: Operationalizing Threats to MSR Studies by Simulation-Based Testing. MSR 2022

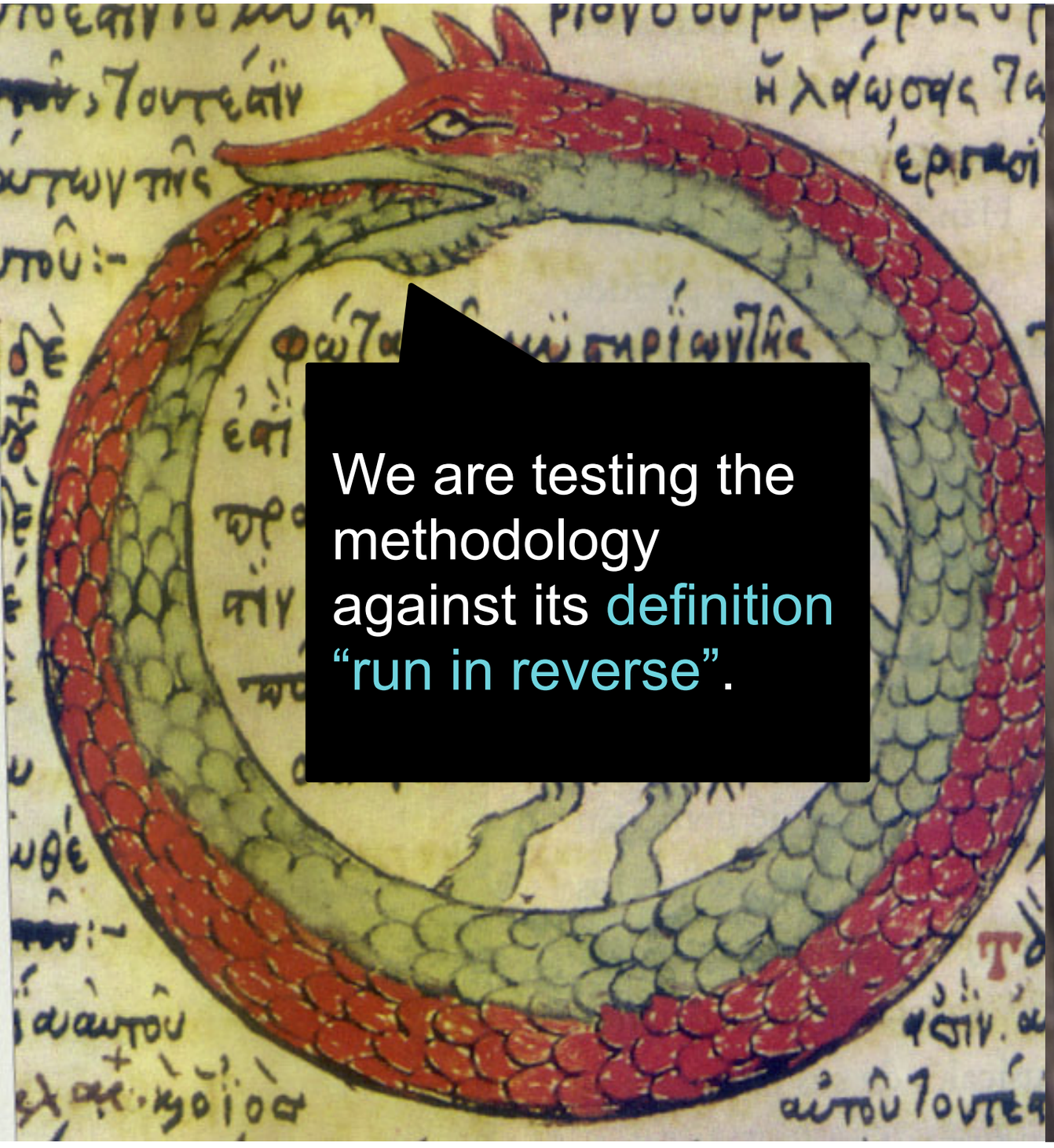
Simulation-based Testing (Summary)

- We **cannot** say that the **original methodology** is **correct/wrong**.
- We **can** say that the **original methodology** is **correct/wrong** under a **given simulation.**
  - **We expect** a methodology to produce **correct results in plausible simulations.**
  - If results **are wrong,**
    - **we expect** a study to **list the simulation** as **threat,**
    - or **admit impossibility**.

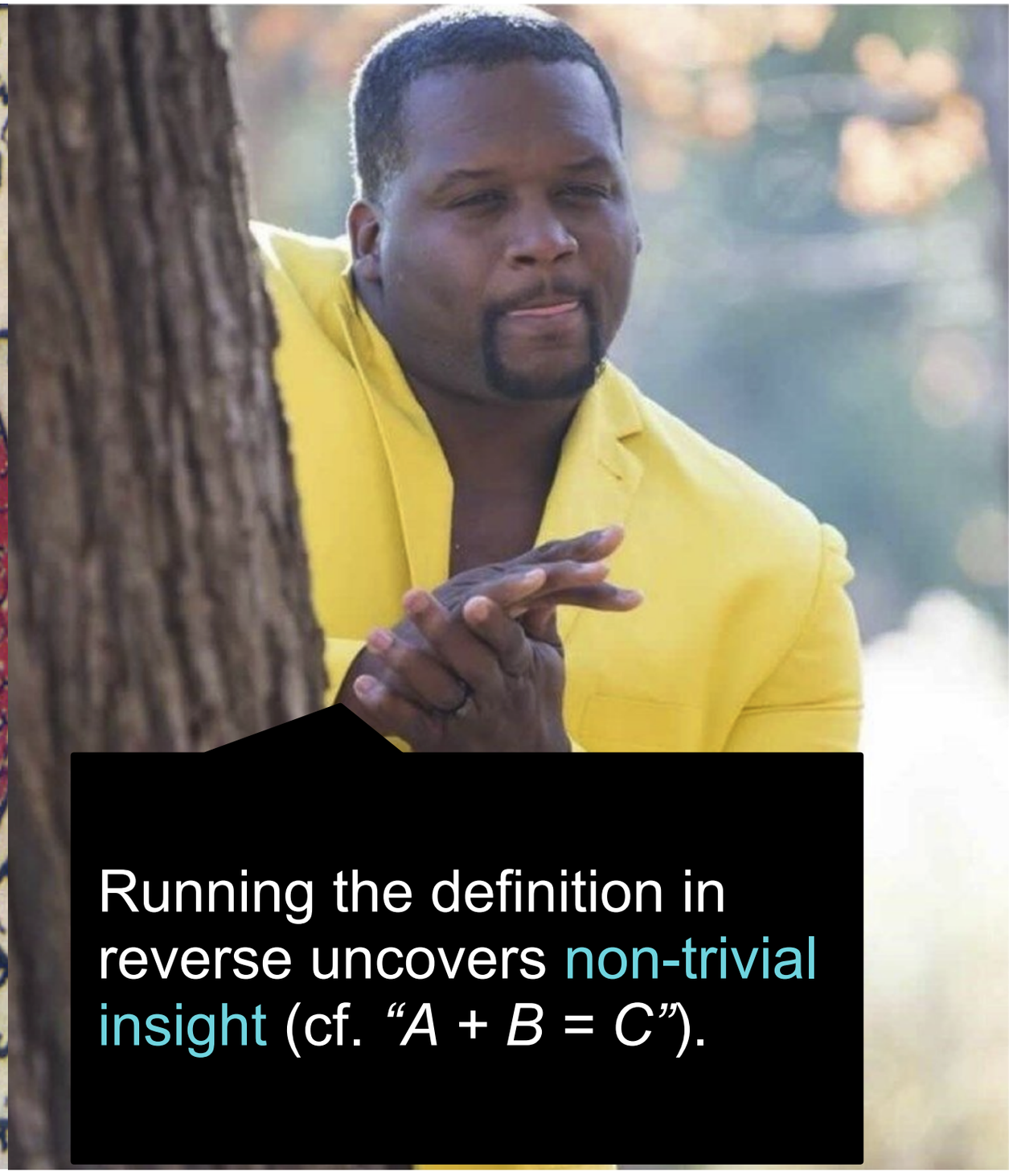
We still need to rate the plausibility of a simulation, which may still be subjective.

Simulations improve the discussion of threats a lot.

Even this should be published.



We are testing the methodology against its definition “run in reverse”.



Running the definition in reverse uncovers non-trivial insight (cf. *“A + B = C”*).

Simulation-based testing in a nutshell: A simple example

Johannes Härtel and Ralf Lämmel: Operationalizing Threats to MSR Studies by Simulation-Based Testing. MSR 2022

Example: Debugging a software defect model

- Observed variables:
  - X — Some software metric (e.g., LOC)
  - Y — Binary defect classification
- Assumptions:
  - Logistic regression model for relationship between variables

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

- Basic methodology:
  - Identify intercept+slope
- Finding:
  - Slope is positive. Thus, commits with more changed lines are more dangerous.
- Debugging:
  - Replace some observed and unobserved variables by synthetic data.

Example: Debugging a software defect model

R code which substitutes variables of the original methodology by synthetic variables

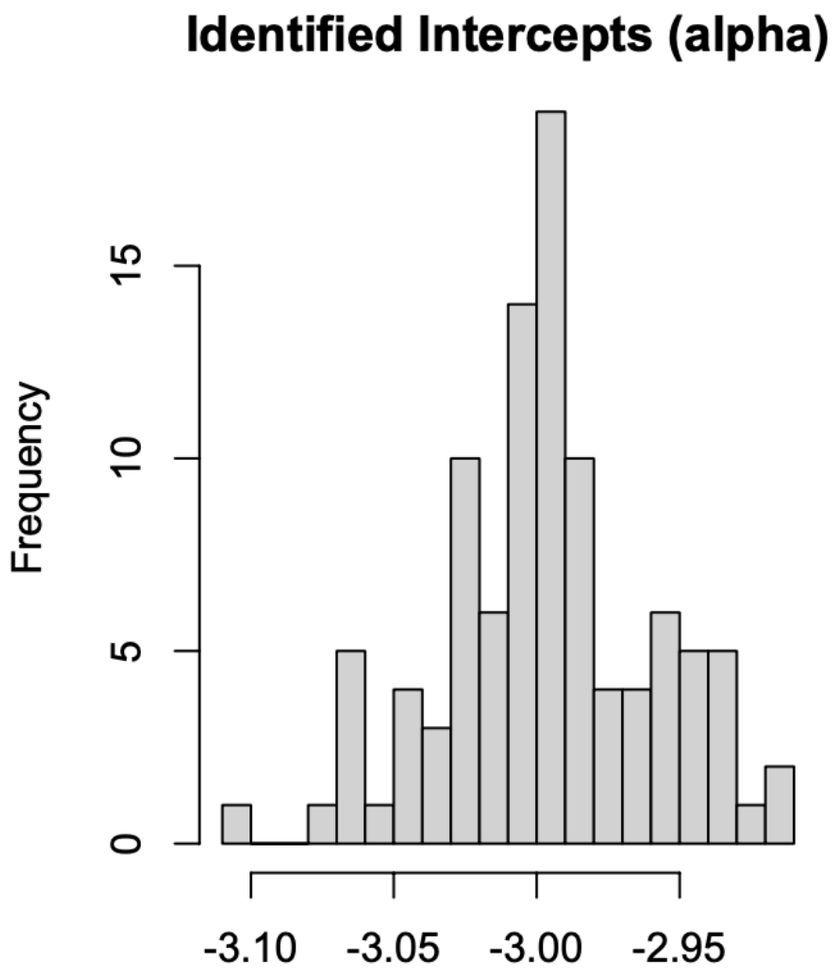
```r
# Kept observed variables.
N ← N # Number of commits.
X ← X # (vector) Keep the original variable X.

# 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.
```

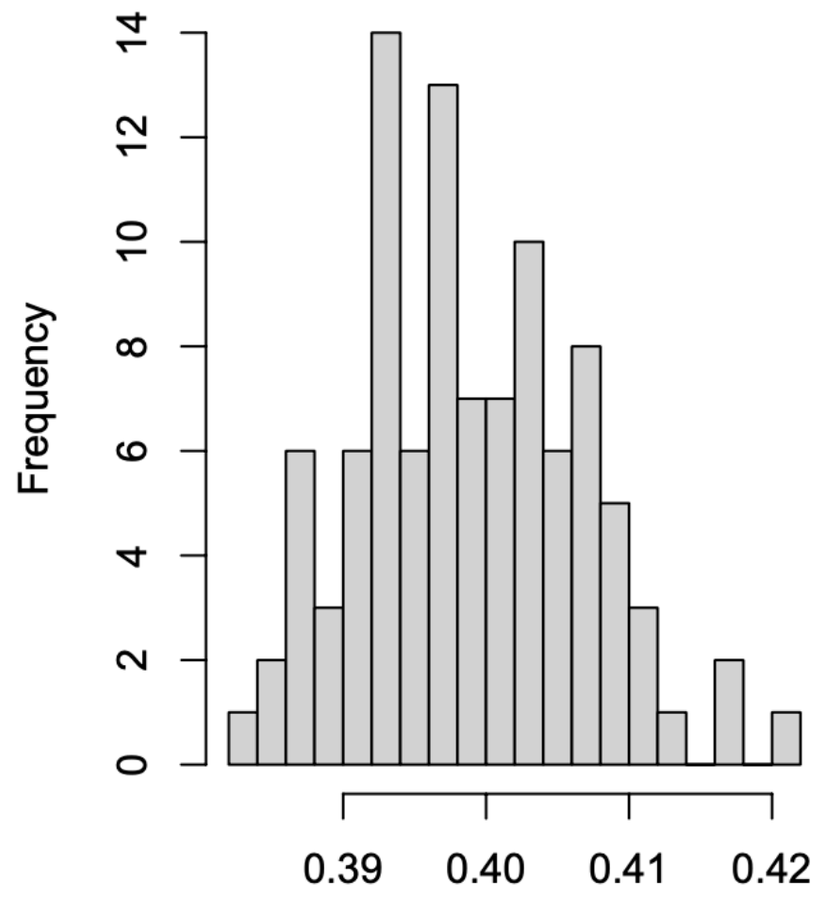
Example: Debugging a software defect model

- Correspondence:
  - alpha = - 2.97 vs. -3.0 and beta = 0.39 vs. 0.4
- Uncertainty:
  - Are we getting the same alpha and beta each time?
  - ***No!***
- Parametrized tests:
  - Does correspondence work for different alpha/beta?
  - ***No!***

Example: Debugging a software defect model

Uncertainty for software defect analysis (100 runs with different seeds)

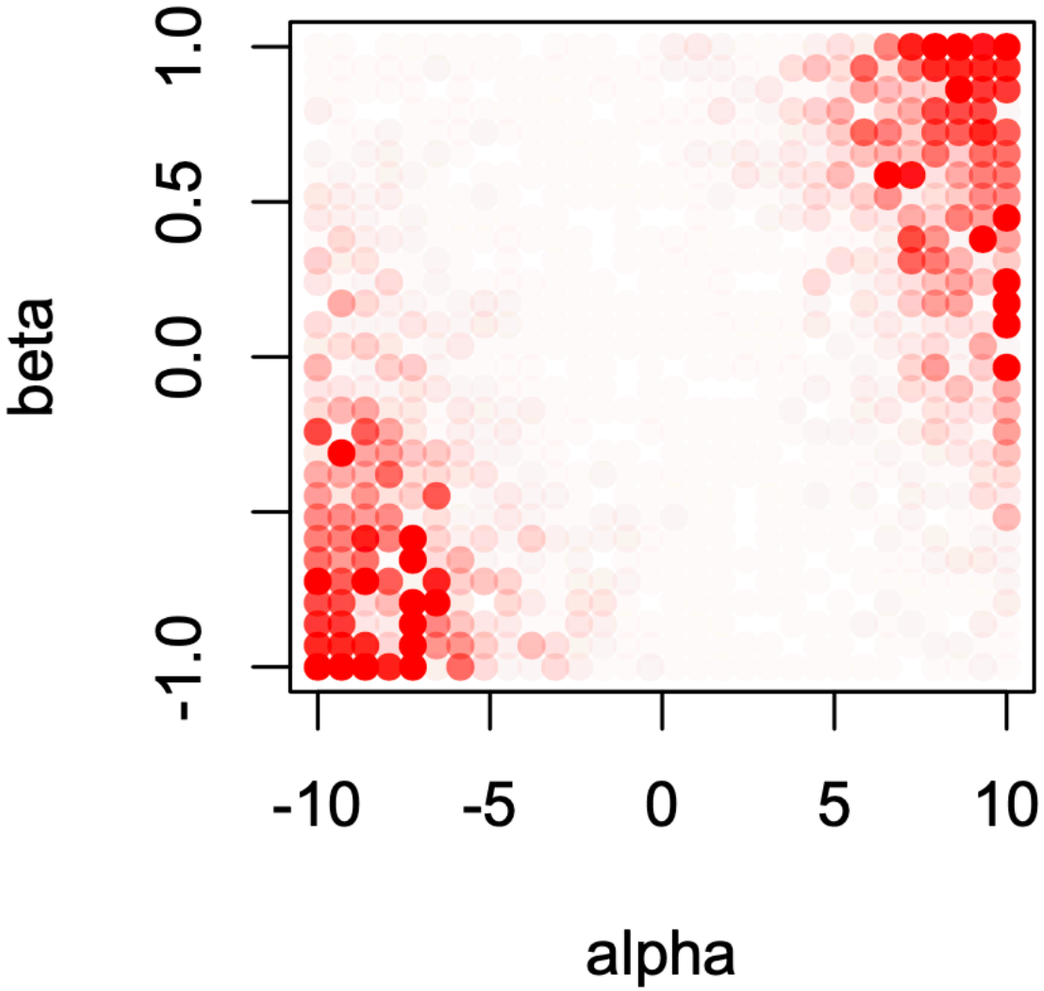




Basic methodology cannot observe *prob!*

Example: Debugging a software defect model

Parametrized tests for software defect analysis (Many runs with alpha/beta on a grid)



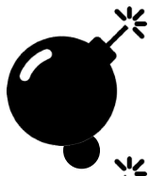
Simulated alpha and beta and the corresponding error in the identification, depicted as red dots (red in- creases with error).

Operationalizing Threats in *Real* Studies

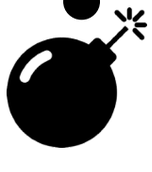
Johannes Härtel and Ralf Lämmel: Operationalizing Threats to MSR Studies by Simulation-Based Testing. MSR 2022

What kind of bombs do we have?

Dependent observations



Causation vs. prediction



Control of variables

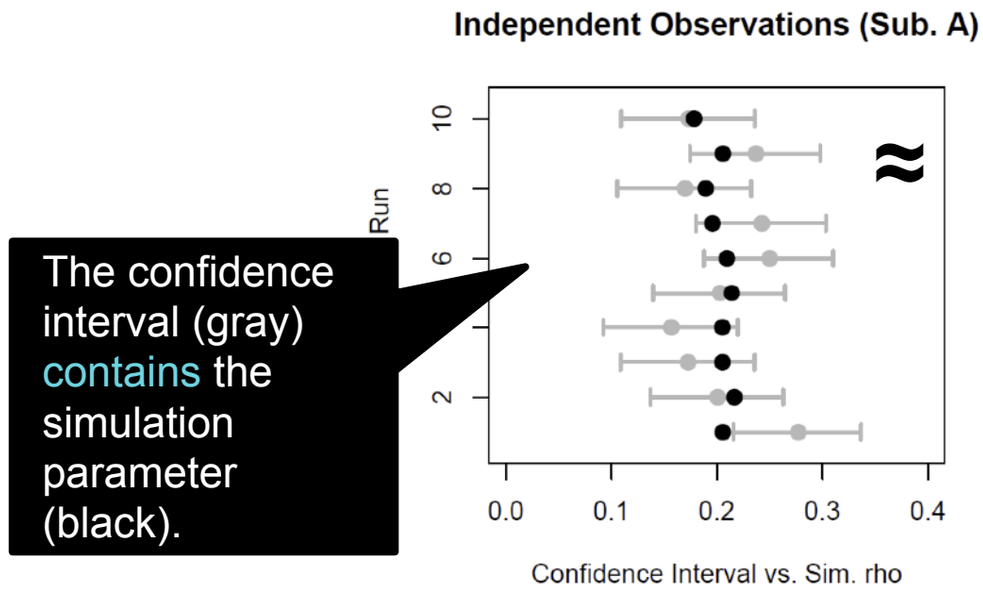
Correlated variables

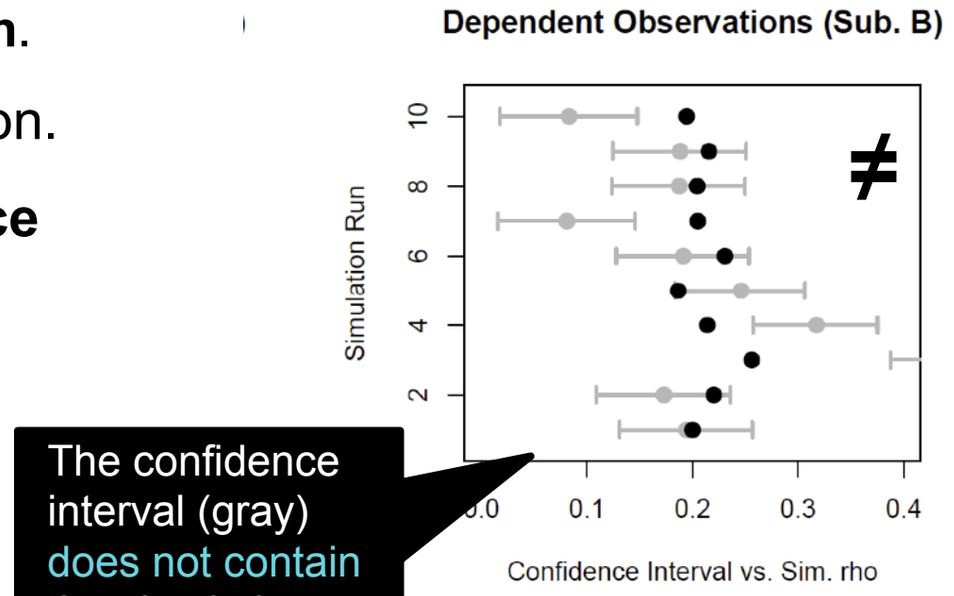
- … future work / community effort needed*!*

We used simulation-based testing to pinpoint threats in real studies (as an evaluation).

Dependent observations (Case 1)

- **Methodology:** [C1] computes **confidence of correlation values** for commit metrics from **9 different repositories**.
- **Simulation:**
  - We synthesize **structure** (i.e., commits in repositories).
  - We synthesize **correlating commit-specific metrics**.
  - We **sample** 9 repositories from 100.
  - We synthesize **repository-specific variation in correlation**.
- **Plausibility:** Many studies prove repository-specific variation.
- **Results:** The simulation shows that the **original confidence computation fails (≠)**.
- **Conclusion:** Use models including the dependency.





[C1]: A. Alali, H. H. Kagdi, and J. I. Maletic: "**What's a Typical Commit? A Characterization of Open Source Software Repositories**", in ICPC, 2008.

Code 1/2 — Dependent observations (Case 1)

```r
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])
}
```

[C1]: A. Alali, H. H. Kagdi, and J. I. Maletic: "**What's a Typical Commit? A Characterization of Open Source Software Repositories**", in ICPC, 2008.

Code 2/2 — Dependent observations (Case 1)

A repository-specific variation

```
for (repo in 1:100) {
  rho ← rnorm(n = 1, mean = 0.2, sd = 0.23) # A
        repository-specific variation in the
        correlation.
  # ...
}
```

[C1]: A. Alali, H. H. Kagdi, and J. I. Maletic: "**What's a Typical Commit? A Characterization of Open Source Software Repositories**", in ICPC, 2008.

Causation vs. prediction (Case 2)

- **<u>Methodology:</u>** [C2] discusses how to answer questions, like *‘whether complex code increases project risk’* [copy, C2] **but does not discuss causation**.
- **<u>Simulation:</u>**
  - We synthesize **X causing Y**.
  - We synthesize **X not depending on other variables.**
- **<u>Plausibility:</u>** Needs discussion, highly subjective.
- **<u>Results:</u>** The simulation shows that we **can claim a casual relationship,** if X does not depend on other variables.
- **<u>Conclusion:</u>** Claim causation, but **discuss the assumptions** of this simulation**.**

Simulations can describe these riddles

[C2]: C. Tantithamthavorn and A. E. Hassan: “**An experience report on defect modelling in practice: pitfalls and challenges**”, in ICSE (SEIP), 2018.

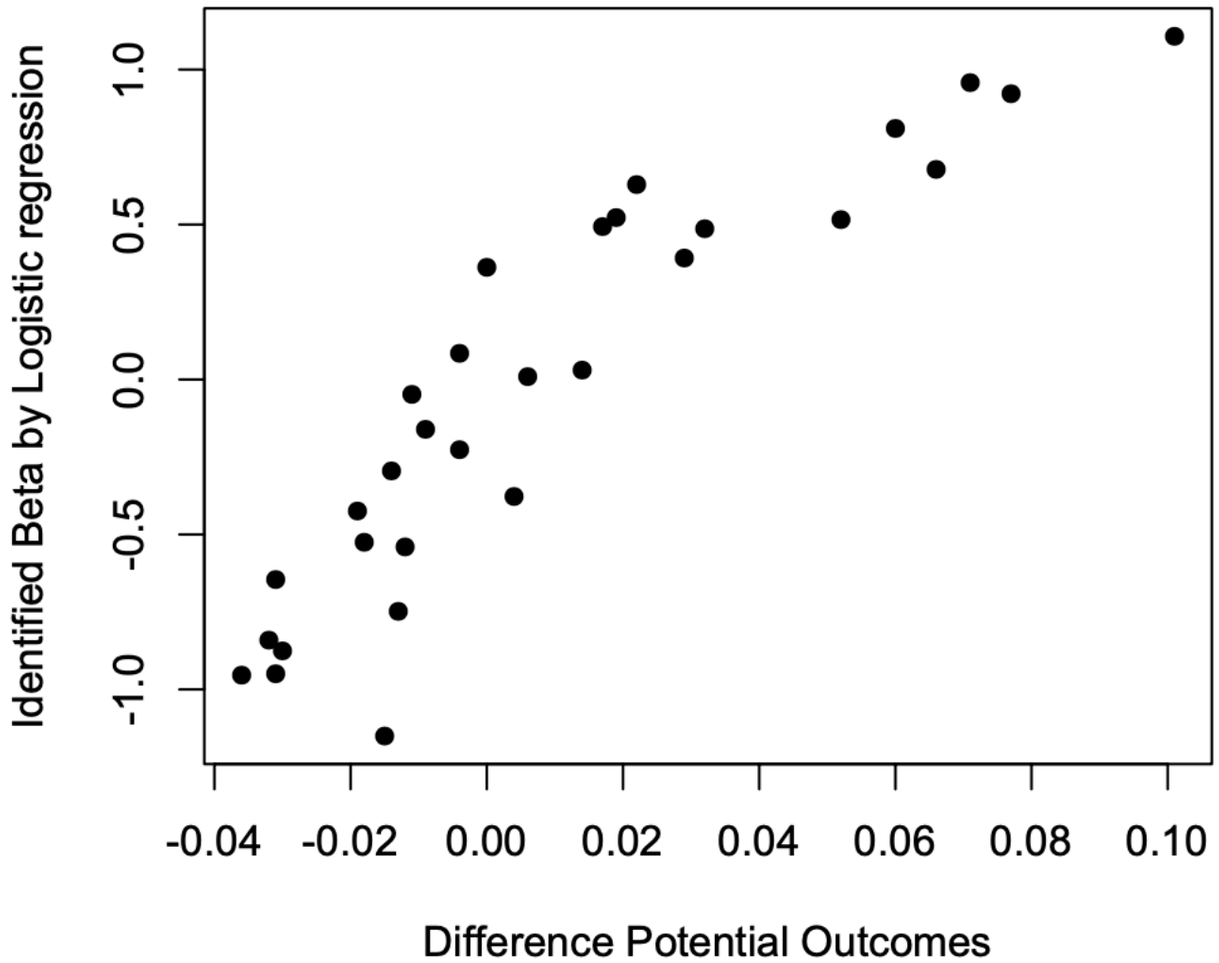
Code — Causation vs. prediction (Case 2)

```r
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)
```

This will get more interesting for control variables, structure, and instrument variables.

[C2]: C. Tantithamthavorn and A. E. Hassan: “**An experience report on defect modelling in practice: pitfalls and challenges**”, in ICSE (SEIP), 2018.

Causation vs. prediction (Case 2)



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. Thus, we claim causation, while assuming that X does not depend on other variables.

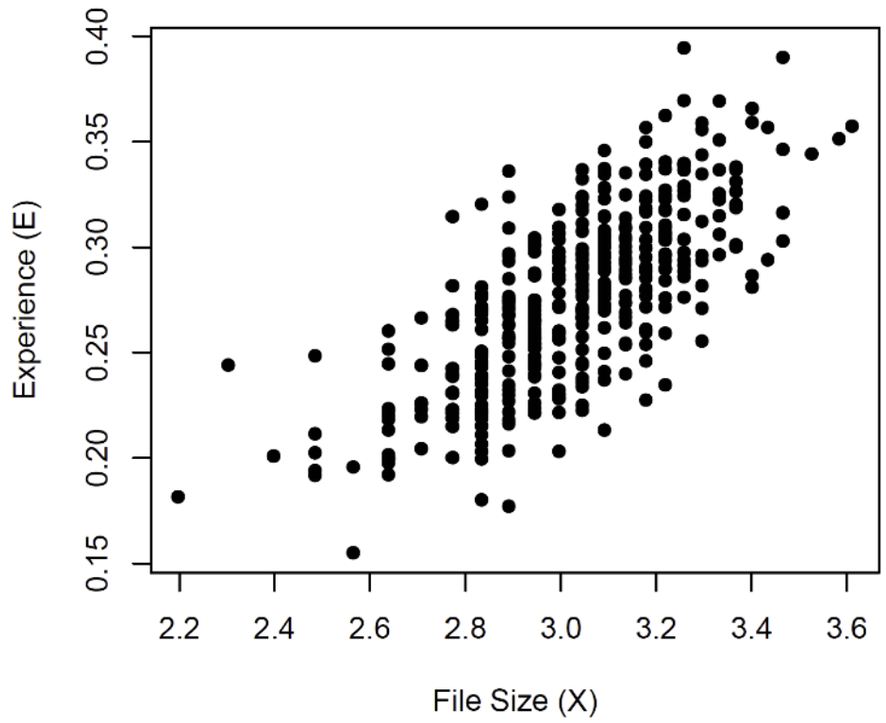
Control of variables (Case 3)

- **<u>Methodology:</u>** [C3] proposes a "**lexical experience metric**" using a **cosine (VSM)** between the commit and a developer's previous commits, showing a relation to defects.
- **<u>Simulation:</u>**
  - We synthesize **artificial vector pairs** for a commit and previous commits.
  - We synthesize an effect of commit's size on defects, but **no effect of the cosine** on defects.
- **<u>Plausibility:</u>** We know that commit's size effects defects. No effect of the new metric is still plausible.
- **<u>Results:</u>** The original methodology, not controlling for commit's size, **accidentally proves an effect of experience (≠)**, caused by a correlation between a commit's size and the cosine.
- **<u>Conclusion:</u>** Control for commit's size is mandatory here.

[C3]: 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., 2017.

Code — Control of variables (Case 3)

```r
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)
```



[C3]: 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., 2017.

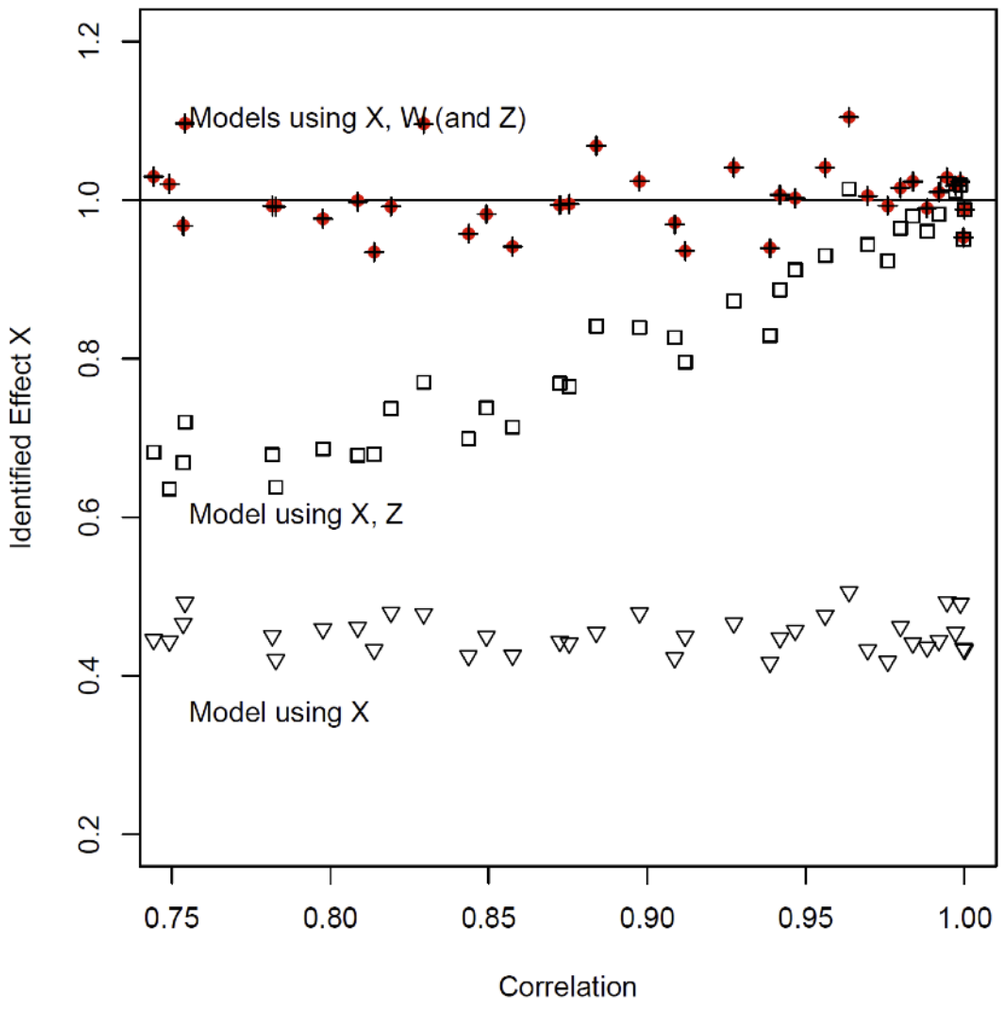
Correlated variables (Case 4)

- **<u>Methodology:</u>** [C4] **recommends** the practice of **removing correlating variables** from defect model to improve interpretation.
- **<u>Simulation:</u>**
  - We synthesize **a basic causal pattern**.
    - We synthesize X, W, Z and defects Y. Only X and W have an effect on Y. W and Z may get strongly correlating. W is a classical confounder.
- **<u>Plausibility:</u>** There is the possibility that this pattern appears.
- **<u>Results:</u>** The original methodology of [C4] (removing W or Z) **does not help / makes identification worse (≠).**
- **<u>Conclusion:</u>** Don't interpret W, Z when correlation is too strong, but **keep both in the model.**

[C4]: J. Jiarpakdee, C. Tantithamthavorn, and A. E. Hassan: "**The Impact of Correlated Metrics on the Interpretation of Defect Models**", IEEE Trans. Software Eng., 2021.

Code — Correlated variables (Case 4)

```r
# 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)
    # ...
```



[C4]: J. Jiarpakdee, C. Tantithamthavorn, and A. E. Hassan: "**The Impact of Correlated Metrics on the Interpretation of Defect Models**", IEEE Trans. Software Eng., 2021.

Conclusion on Simulation-based Testing

- We do **not solve** the problem of **correctness**.
- **Plausibility** of simulations still **need to be rated**.

- **Given that simulations are plausible**, statements on correctness are relevant.
- Simulations operationalize treatment of threats is **more systematic than any informal discussion.**
- It helps to **share and evolve** knowledge on threats.

Ready-to-run simulations available online:

https://github.com/topleet/MSR2022

Johannes Härtel and Ralf Lämmel: Operationalizing Threats to MSR Studies by Simulation-Based Testing. MSR 2022