Exploring the Garden of Forking Paths in Empirical Software Engineering Research: A Multiverse Analysis

NATHAN CASSEE and ROBERT FELDT

In empirical software engineering (SE) research, researchers have considerable freedom to decide how to process data, what operationalizations to use, and which statistical model to fit. Gelman and Loken refer to this freedom as leading to a "garden of forking paths". Although this freedom is often seen as an advantage, it also poses a threat to robustness and replicability: variations in analytical decisions, even when justifiable, can lead to divergent conclusions.

To better understand this risk, we conducted a so-called *multiverse analysis* on a published empirical SE paper. The paper we picked is a Mining Software Repositories study, as MSR studies commonly use non-trivial statistical models to analyze post-hoc, observational data. In the study, we identified nine pivotal analytical decisions–each with at least one equally defensible alternative–and systematically reran all the 3,072 resulting analysis pipelines on the original dataset. Interestingly, only 6 of these universes (<0.2%) reproduced the published results; the overwhelming majority produced qualitatively different, and sometimes even opposite, findings.

This case study of a data analytical method commonly applied to empirical software engineering data reveals how methodological choices can exert a more profound influence on outcomes than is often acknowledged. We therefore advocate that SE researchers complement standard reporting with robustness checks across plausible analysis variants or, at least, explicitly justify each analytical decision. We propose a structured classification model to help classify and improve justification for methodological choices. Secondly, we show how the multiverse analysis is a practical tool in the methodological arsenal of SE researchers, one that can help produce more reliable, reproducible science.

ACM Reference Format:
Nathan Cassee and Robert Feldt. 2026. Exploring the Garden of Forking Paths in Empirical Software Engineering Research: A Multiverse Analysis. 1, 1 (January 2026), 26 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Since the early 2010s, science has faced a crisis of confidence. The so-called replication crisis—sparked by a series of failed replications of prominent psychology findings—has exposed deep flaws in how scientific studies are designed, analyzed, and reported [8]. The problems run deeper than fraudulent practices or overt P-hacking. Even in well-intentioned studies, the sheer number of methodological decisions researchers can make, described by Gelman and Loken as the "garden of forking paths",[1] can silently steer results in different directions [21]. The garden of forking paths represents the idea that scientific findings, like statistical significance, emerge not because of manipulative intent but due to researchers' vast, often unacknowledged flexibility in analyzing their data. A striking illustration of this was provided by Silberzahn et al. [57], who showed that different teams given the same dataset and research question arrived at vastly different conclusions, simply because they made different–but all reasonable–choices in how

[1]Inspired by the short story of Jorge Luis Borges.

Authors' address: Nathan Cassee, nathancassee@uvic.ca; Robert Feldt, robert.feldt@chalmers.se.

Manuscript submitted to ACM

to analyze the data. This highlights how the garden of forking paths might influence study outcomes and why it is vital to study the effect of methodological decisions on outcomes.

One method that can be used to study methodological sensitivity, which is still underutilized in many fields, is the systematic mapping and investigation of how analytical choices influence study outcomes. This form of analysis, sometimes referred to as a *multiverse analysis* after Steegen et al.[59], enables researchers to explore a structured set of alternative analytical paths. In doing so, it exposes how robust or fragile a study's conclusions are to reasonable variations in methodology. While such approaches have begun to gain traction in disciplines such as psychology and epidemiology, to our knowledge, they have not yet been applied in software engineering.

This gap is surprising, especially given that the software engineering (SE) community has actively engaged with other facets of the replication crisis [9, 14, 41, 55]. In particular, the Mining Software Repositories (MSR) community has recognized the methodological challenges inherently present in the data. Although MSR researchers work with rich and powerful data sources [5, 30], uniquely suited to studying software engineering phenomena [17, 36, 46], they must make many decisions to mitigate the known perils associated with such data [5, 30, 45].

Methodological diversity is common in software engineering literature. For instance, a meta-study of MSR studies by Mahadi et al.[38] found considerable variation in how studies addressing the same research question operationalized their analyses. Wyrich et al.[71] demonstrated how even slight differences in how SE researchers define key constructs can lead to results that are difficult to compare and potentially contradictory, and re-analysis of previously reported findings using different analytical methods has already shown how findings can change [18, 20] .

In this work, we want to further understand the impact of these methodological variations, and therefore, we pose:

RQ *How sensitive are the conclusions of Empirical Software Engineering studies to methodological decisions?*

To answer this question, we selected a published MSR study [7] that uses a data-analytical method employed in over ten MSR studies and known for affording researchers a high degree of freedom, making it a fitting choice for our investigation. We systematically explore 3,072 distinct, yet plausible, analytical variants of the original study–each representing different combinations of nine methodological decisions. Our goal is to quantify how often and to what extent the study's conclusions change when alternative, yet defensible, analytical choices are made.

The results are instructive, even in this single case. Among the 3,072 analytic paths we explored, only six (0.2%) reproduced the original study's result. Many paths yielded null or even contradictory outcomes. Each of the nine analytical decisions had the power to flip the result—underscoring the fragility of conclusions drawn. Our findings serve as a cautionary tale, *as we find that if researchers' degrees of freedom increase, confidence in results decreases.* Especially because empirical software engineering, and MSR in particular, rely on data sources that require a high degree of researcher freedom.

More constructively, our study highlights the need for greater transparency in **justifying** methodological decisions in MSR research. We introduce a model that can be used to reason about the different types of justification and discuss practices to strengthen them. Moreover, our findings demonstrate the value of multiverse analyses: by systematically exploring alternative methodological choices–particularly when researcher degrees of freedom are high–one can pinpoint which methodological decisions, if any, most critically affect results.

2 RELATED WORK

The factors contributing to the replication crisis have been studied extensively. Chambers [8] identified seven "sins" that capture problematic research practices. A range of potential solutions has also been proposed [14, 62]. In this

section, we focus on prior work related to methodological freedom, statistical analysis in software engineering, and multiverse analysis.

The validity and reliability of empirical software engineering literature has been studied extensively. Early work by Dybå et al. [13] in 2006 already showed how the sample size in existing software engineering experiments was too low. Similarly, Reyes et al. [47] shows that many experimental software engineering papers make statistical errors seen in other disciplines. Understanding the statistical analysis reported in software engineering is further complicated by inconsistent reporting guidelines. Both de Oliveira Neto et al. [11], Santos et al. [51] describe how heterogeneity and inconsistent reporting guidelines of statistical tests complicate any sort of meta-analysis. To help remediate some of the issues reported previously, guidelines on how to apply statistical methods have been described [2, 33]. However, these guidelines often focus on how to report and visualize results rather than on methodological freedom.

Across several fields, researchers have shown how degrees of methodological freedom can lead to varying outcomes. Silberzahn et al. [57] report that 29 independent analysis teams, tasked with answering the same research question, employed a wide range of analytical methods and reached conflicting conclusions. Similarly, Schweinsberg et al. [54] demonstrates that when given substantial flexibility in data analysis, different teams make divergent choices, producing inconsistent results. Sarstedt et al. [53] further show that even when teams analyze the same model, their different decisions about data processing lead to varying effect sizes.

To address the link between methodological freedom and study outcomes, Steegen et al. [59] introduced the concept of *multiverse analysis*. Giudice and Gangestad [22] describes guidelines for conducting meaningful multiverse analyses, emphasizing the need to explore only reasonable methodological alternatives. Harder [25] discusses how multiverses should be expanded to not just include data analytical decisions, but also decisions related to data collection. While Simonsohn et al. [58] introduces specification curve analysis, a method that uses multiverses to make inferences about the underlying data. To help interpret multiverses, Dragicevic et al. [12] introduces a tool that can interactively explore how methodological choices affect outcomes. Similarly, Liu et al. [37] presents a formalized DSL to systematically explore multiverses and assist in multiverse analysis. Bell et al. [4] apply the multiverse concept to experimental design choices in machine learning benchmarks, proposing a framework to strengthen benchmarking robustness.

To our knowledge, no multiverse analyses have been conducted specifically in software engineering. While sensitivity analyses [50] can also appear similar in nature to multiverse analyses, there are several key differences. Where a sensitivity analysis is often used to show that a specific assumption does not bias the outcome, the point of a multiverse analysis is to show how the freedom of researchers to make analytical decisions influences outcomes.

While no explicit multiverses have been conducted in software engineering, the field has discussed and hinted at the adverse effects of methodological freedom. In a replication study, Mahadi et al. [38] observe that different studies addressing the same research question make numerous, varying methodological choices, complicating direct comparisons. Similarly, Shepperd et al. [56], finds conflicting results across defect prediction studies, and Wessel et al. [68] found that different Regression Discontinuity Design studies report conflicting or incomparable results.

Thus, while prior software engineering research has documented heterogeneity in methodological decisions, it has not systematically examined how sensitive study outcomes are to those choices. This gap is the focus of the present work.

3 STUDY OVERVIEW

To understand how analytical decisions influence the outcome of Mining Software Repositories studies, we conduct a *multiverse* analysis [59]. In this multiverse analysis, we study alternatives (*universes*) to analytical *decisions* made in

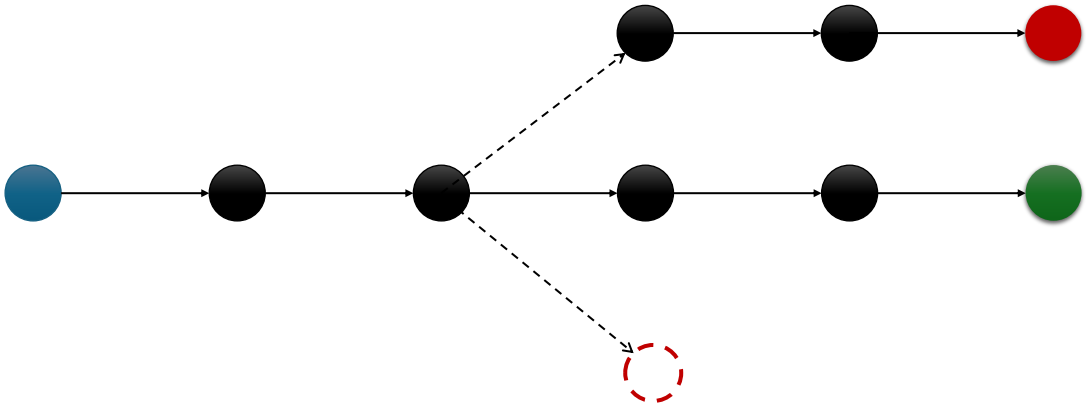

Fig. 1. Visualization of the core idea of a multiverse analysis. A study moves from research question (blue node) to outcome (green node) through a path of methodological decisions (black nodes). In a multiverse analysis, alternative paths (dashed arrows) are explored to study whether these alternatives change outcomes (red node).

an MSR study. By recording the outcomes (significance scores) of the study, we learn whether there is any relation between data analytical decisions and study outcomes

Figure 1 visualizes the core idea of a multiverse analysis. A multiverse analysis explores whether alternative methodological decisions can result in alternative outcomes. **In a multiverse analysis, a single *universe* represents one set of analytical decisions leading from research question to outcome. Meanwhile, the multiverse is a set of universes representing the valid methodological designs to address a research question.** Through a systematic exploration of these universes, created by identifying alternative choices for methodological decisions, we re-examine the original research question. The goal of such an analysis is to identify the sensitivity of the outcome to methodological decisions.

Many studies in empirical software engineering consist of a large number of methodological decisions, with many alternatives to these decisions that are typically considered [48]. Which is why we believe it is important to apply multiverse analyses. However, it's important to note that not every alternative is reasonable [22]. For instance, deciding to use a parametric test on non-parametric data is an example of an alternative we are not interested in exploring – as the alternative (parametric test) does not meet existing assumptions and is known to produce potentially invalid outcomes. Therefore, we carefully construct the universes we explore in this multiverse analysis.

However, before starting the analysis, we first pick and describe a data analytical method commonly applied to MSR data. Then we pick a primary study that applies this method to empirical software engineering data. In the remainder of this section, we first provide background information on the data analytical method Section 3.1; we discuss how this method has been applied to software engineering and how there is quite a lot of heterogeneity in the decisions made when this method is used to study software engineering (Section 3.2). Finally, we pick a single case study that applies this data analytical method, and we describe it (Section 3.3).

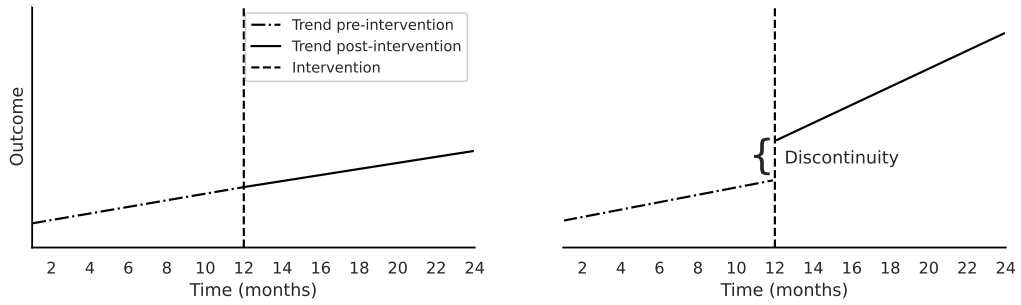

Fig. 2. An example of Regression Discontinuity in Time (RDiT) design. The plot shows two scenarios; in both scenarios, there is an intervention taking place in the 12th month. The left plot shows a scenario where the intervention does not have any effect, with no change in the outcome after the intervention, while the right plot illustrates a scenario where the intervention results in an observable change, with a visible discontinuity both in slope and intercept of the outcome.

3.1 Regression Discontinuity in Time

Studying how an intervention impacts a process is a challenging problem. Regression Discontinuity in Time (RDiT) [26] is one of the statistical methods applied to observational data to study the impact of interventions. RDiT is a quasi-experimental statistical technique applied to observational data in settings where it is impractical or impossible to conduct randomized trials. A commonly used example motivating the use of RDiT is a situation like a power plant installing pollution filters, and scientists wishing to understand whether this reduces pollution in the surrounding plant environment [26]. These are settings where studying the same power plant simultaneously with and without the filter is impossible. Furthermore, as there are usually many sources of pollution, there is insufficient control over the environment to isolate the effect of the power plant on the environment. In those cases, a quasi-experimental technique like RDiT can help quantify the impact of the pollution filters. In the literature, this method is also known as regression discontinuity design (RDD); however, in this manuscript, we will refer to it as RDiT.

In an RDiT design, two separate regressions are fit to a timeline. One regression fits the pre-intervention data points, while the second regression fits the post-intervention data points. The difference between slopes and intercepts of the two regressions is used to approximate the effect of an intervention. By comparing the two regressions, one can understand whether there was an immediate “discontinuity” post-intervention and whether there was a change in the trend. Figure 2 shows a visual example of two RDiT models. In both plots, the x-axis represents time, which is usual in RDiT designs. The timeframes are centered around the intervention point, and in each plot, the two lines show the two fitted regression models. The difference between the two plots is that the right plot suggests a ”discontinuity” immediately after the intervention and a trend change post-intervention. Based on the left plot, one would conclude that the intervention had no effect, whereas, for the right plot, one would conclude that the intervention led to an increase in the dependent variable.

3.2 Regression Discontinuity in Time in Software Engineering

Because of the challenges in studying interventions in software engineering, RDiT has been used to study the effect of interventions on various software engineering activities. In this section, we give an overview of these studies, and we highlight how there is a large variation in the data analytical decisions made in each of these studies. To find RDiT studies in software engineering, we used an informal search process, combining forwards snowballing from two of the

first reported RDiT studies in software engineering ([7, 72]) combined with searches on Google Scholar using keywords like *"Regression Discontinuity Design"* and *"Regression Discontinuity in Time"*.

Table 1: The time modeling choices made in RDiT studies.

Authors	**Topic**	**# Periods**		**Period Length**	**Exclusion**	
Zhao *et al.* [72]	Impact of CI on Commits & PRs	24		30 days	Middle period excluded	✔
Cassee *et al.* [7]	Impact of CI on Code Reviews	24	✔	30 days	Middle period excluded	✔
Guo and Leitner [23]	Impact of CI on merge time	Variable		7 days	No exclusion	
Kavalar *et al.* [31]	Impact of QA tools on issues, churn, PRs & contributors	Variable		30 days	One Month	
Wessel *et al.* [67]	Impact of bots on software engineering	24		30 days	Middle period excluded	
Kinsman *et al.* [32]	Workflows	12		30 days	Middle period excluded	✔
Trockman *et al.* [64]	Impact of badges on dependency age	18		30 days	*No Exclusion*	
Zimmermann *et al.* [73]	Impact of switching on bug trackers	175 and 511 (days)	✔	1 day or 1 week	*No Exclusion*	
Moldon *et al.* [44]	Removal of GH Features	2, 4, 6		weeks	*No Exclusion.*	
Walden *et al.* [66]	Impact of security bugs on development	50	✔	30 days	*No Exclusion*	
Moharil *et al.* [43]	Impact of bot on issues	24		30 days	Middle excluded	
Saraiva *et al.* [52]	Impact of CI on code coverage	24	✔	30 days	No exclusion	✔
Li *et al.* [35]	Impact of issue report templates	24		30 days	*No Exclusion*	
Ayoup *et al.* [3]	Impact of Gamyfing features on DevOps	24		30 days	*No Exclusion*	

Table 1 lists RDiT studies conducted in software engineering and the time modeling choices made in each of these studies. When a paper did not mention details, *emphasis* is used to denote that no mention of any exclusion period was

made. Some papers modeled multiple time series or used different decisions to fit several models. The green checkmarks to the right of a chosen value indicate that the paper **includes** an explicit motivation for choosing that value. In the package of additional materials, we've included an Excel sheet with verbatim copies of the motivations from the listed studies.

To conduct an RDiT study, researchers need to make several data analytical decisions that determine how time is modeled. As seen in Table 1 there is a large variation across these studies in modeling time and excluding time periods to account for instability. Moreover, in each referenced study, justification for the choice of the period studied is often absent or brief. While it is acknowledged that the studied period shouldn't be too long, as mentioned in RDiT literature [26], or that some datapoints should be excluded to account for instability surrounding the intervention, the choice of popular options (excluding one-time period, analyzing 24 periods, and using 30 day long periods) is seldom justified, or only justified by referring to previously published studies.

This heterogeneity is one of the reasons why we focus on RDiT: To apply RDiT, a researcher needs to make many methodological decisions . Table 1 shows researchers actively use freedom to make different analytical decisions across different studies. Moreover, Wessel et al. [68] found that the outcomes of several RDiT studies are incomparable or report conflicting outcomes. This variability in study outcomes further motivates us to use RDiT to conduct a multiverse analysis, as we believe the different data analytical decisions made might explain the incomparable outcomes.

3.3 Primary Study

We pick one study from the recently conducted RDiT studies in empirical software engineering and use it as a case. Because of the variation in analytical decisions across RDiT studies, and the conflicting outcomes in these studies [68], we believe any RDiT study is a good case for a multiverse analysis: It is a recent data analysis method that has been used to study software engineering. Moreover, Table 1 shows how there is a high degree of researcher freedom in the application of RDiT, and Wessel et al. [68] shows RDiT studies have conflicting outcomes.

The study we select for this multiverse analysis is authored by Cassee *et al.* and titled "*The Silent Helper: The Impact of Continuous Integration on Code Reviews*" [7]. Selecting this study has a practical advantage, as one of the authors of this study was also involved in the primary study. This familiarity with the primary study and access to an original data archive allowed us to revisit the analytical decisions in more detail.

For completeness' sake, we briefly summarize the research questions, the data used in the study, the study design, and the study results in the primary study.

Cassee *et al.* studies how the adoption of Continuous Integration (CI) by open-source software projects influences the *Communication* within a code review and the number of *Changes* made during a code review. Using RDiT, they studied the following four hypotheses:

H1 *The adoption of CI influences the number of General Comments.*

H2 *The adoption of CI influences the number of Review Comments.*

H3 *The adoption of CI Influences the number of changes made because of Review Comments.*

H4 *The adoption of CI influences the number of commits made during a Code Review.*

H1 and H2 measured the developer's communication, while H3 and H4 measured the changes made during the code-reviewing process.

To verify these hypotheses, Cassee *et al.* mined code reviews for 685 popular and active GitHub projects that adopted TravisCI, a popular CI service provider [39]. To analyze the data, Cassee *et al.* used an RDiT design. To that end, they created a time series per project centered around the time at which the project started using TravisCI. For each of the four hypotheses, they then fit an RDiT model to understand whether that specific dependent variable was affected by the adoption of TravisCI. The RDiT model includes the three variables used to model time, and a series of independent variables representing the size and the activity of the community.

Cassee *et al.* finds that a year after the adoption of continuous integration, the average code review is conducted with less communication (**H1** & **H2**). Furthermore, no effect of continuous integration on the number of changes made during a code review is found (**H3** & **H4**). Therefore, the authors conclude that continuous integration allows developers to perform the same work in a code review while requiring less communication.

4 METHODOLOGY

In this section, we identify the data analytical decisions of the primary study that we explore in the multiverse analysis (Section 4.1), and in Section 4.2, we describe how we compare the outcomes of the universes. Additional materials, including the scripts used to generate the multiverse and the outcomes from each universe are available as an online repository on Figshare. [2]

4.1 Multiverses

We follow the steps outlined by Steegen et al. [59] to design this multiverse analysis and to analyze the decisions made in the primary study (Cassee et al. [7]). First, we identify a set of *decision points* in the primary study. For each identified decision point, we then justify reasonable alternatives thereby creating the set of alternative *universes* we explore. By re-running the experiments of the primary study in these universes, we obtain a set of outcomes we compare to the outcome of the primary study.

Designing a meaningful multiverse analysis requires carefully balancing the number of decision points with the alternatives per decision points we consider. Giudice and Gangestad [22] warns against exploring too many alternative decisions in one multiverse analysis, as combining a large number of arbitrary alternatives creates an exponential number of universes with a large variety of outcomes, complicating the interpretation and usefulness of the results. Therefore, we only select a limited number of decision points from the primary study to explore.

One set of decision points we focus on include the decisions made to model time. We select these decisions because they are already varied in the existing software engineering RDiT literature (cf. Table 1), showing that in practice, there are many different alternatives being explore across different studies. We combine these decisions with a small number of additional decision points representing the decisions made to aggregate the data and fit the models.

Notably, this excludes other decision points like those related to the processing, filtering, or selection of independent variables. We do not include these decision points for several reasons: some decisions related to data collection cannot be repeated. Re-doing data collection at the time of writing would result in a different dataset, as data on Github will have been altered or removed [30]. Secondly, we exclude decisions like the choice of independent variables, as those are decisions for which different theoretical justifications might exist. Thirdly, to ensure interpretability of the multiverse analysis, we exclude any pre-processing decisions not discussed already, to ensure there's no exponential blow-up of explored universes [22].

[2] https://figshare.com/s/362d209dc52a07a19fe0

4.1.1 Data Processing Decision Points. One group of decision points we explore in this multiverse analysis concerns the modeling of time. We focus on these decisions because they are both crucial to an RDiT study and appear heterogeneous across RDiT studies (See Section 3.2). Additionally, according to Hausman and Rapson [26], these decisions are related to the sensitivity of RDiT studies.

In the primary study, each time series consists of 24 30-day periods centered around the point in time when the project adopted continuous integration. The time series' length (24 time periods) and the resolution (30 days per period) are decisions for which we explore alternative values. To account for a period of instability directly before or after the intervention, the primary study excludes 15 days of data before and after the intervention. This is a common decision, seen in several different RDiT studies [7, 67, 72]. Zhao *et al.* justifies this decision based on a manual inspection of some projects. However, as it is unclear how long this instability lasts, and not all RDiT studies use an exclusion period, we also explore several different lengths for the period of instability.

After determining the length and resolution of the time series and the period of instability, the code-reviewing activity is aggregated within a single time period. In the original study, a log-scaled mean was used; however, other valid aggregation methods include using a different logarithmic base or taking the median. Therefore, the second group of decisions we explore in this multiverse analysis are analytical decisions related to aggregation of data in the time-series.

Table 2: Parameters and parameter values we analyze.

Decision	**Definition**	**Values**
Time series creation	Defined as *#periods* and *periodLength*	#periods $\in \{36, 24, 18, 12\}$ and periodLength $\in \{7, 15, 30, 45\}$.
Period of instability	Defined as (*daysBefore*, *daysAfter*).	daysBefore, daysAfter $\in \{(3.5, 3.5), (15, 15), (0, 7), (0, 15)\}$.
Aggregration	Defined as *scalingMethod* and *averaging*.	scalingMethod $\in \{\text{original}, \ln, \log_{10}\}$ averaging $\in \{\text{mean}, \text{median}\}$.
Rounding	Defined as *digitsPrecision*.	digitsPrecision $\in \{\text{unmodified}, 10, 5\}$

Table 2 lists all of the decision points related to data processing we explore. Notably, this excludes other data processing decisions related to, for instance, operationalizations ("*How is the intervention measured?*"). However, we exclude these decisions to reduce the number of studied universes and to ensure the results of the multiverse analysis are meaningful [22].

4.1.2 Statistical Decision Points. For the decision points related to the statistical models fit for each hypothesis , we only focus on a limited set of decision points. There are two factors we focus on: the exclusion threshold for collinearity, which drops collinear independent variables from the RDiT model , and the fitting algorithm used to fit the model.

Table 3: Parameters and parameter values we analyze in the statistical model.

Decision	Definition	Values
Collinearity	Defined as *vifThreshold*	vifThreshold $\in$ {2.5, 5}
Fitting algorithm	Defined as *REML*	REML $\in$ {true, false}

Table 3 shows these two decisions and the different values we consider.

4.1.3 Instantiating the multiverses. In the primary study, a large dataset of code review data originally mined from GitHub was processed into a set of time series, one per project, resulting in a data frame. To create multiverses specified by the decision points in Tables 2 and 3, we modularized the data processing script of the primary study. Each of the decision points described Section 4.1.1 was encoded as an argument of the data processing script. We then ran the modularized script for each unique combination of decision point values, instantiating a set of dataframes representing the multiverses. The modularized data processing script is available in the repository with additional materials.

After instantiating the data frames belonging to each of the decision points, the next step is determining the outcomes in each universe. To do this, we re-run the analysis notebooks from the replication package of the primary study, fitting four models per universe, one model for each hypothesis. By recording the outcomes for each fitted model, we create a mapping from each unique combination of decision points to its corresponding outcome.

4.2 Data Analysis

To understand the relation between decision points and outcomes ,we record a limited set of outcomes for each universe. In this Section, we argue why these outcomes are meaningful, how we group outcomes, and how we visualize the relation between decision points and outcomes.

Table 4: The outcomes of the primary study that are recorded in the multiverse analysis.

Outcome	Description	Justification
P-values	The three P-values of the variables in the model that are used to model the impact of the intervention.	In the original study, the P-values of these model parameters are used to conclude that the adoption of CI had an effect.
Parameter Sign	Whether the sign of the three model parameters used to model the impact of the intervention are positive or negative	In the original study, the coefficients of these parameters are used to reason whether trends are increasing or decreasing.

For each of those four models, we record the outcomes described in Table 4. These outcomes capture the information commonly used to interpret RDiT models, and we compare the outcomes in that universe to the outcomes of the primary study. There are additional outcomes we could include in the comparison, like the point estimations of the effect sizes. While we do agree that improvements to However, this increases the number of possible ways in which outcomes in a universe can differ from the outcomes reported in the primary study. Therefore, we do not include these additional outcomes, and only focus on the outcomes listed Table 4. To further ease the analysis of the outcomes in

each universe, to the outcomes of the primary study we assign the outcome to a *Bucket* representing the similarity of the outcome to the outcome of the primary study. The four groups we use to bucket the universes are:

- **Full replication**: The significance and sign of the three time-based model parameters are **all** equal to those of the original study.
- **Unconfirmed results**: At least one of the claims made in the primary study can not be confirmed in the universe.
- **Opposite results**: At least one of the time-based variables has significance, while no significance was reported in the primary study. Or the sign of the model parameter is reversed, *i.e.*, if the primary study reported a significant increasing trend, a significant decreasing trend is reported in the universe.
- **Model Fit Failure**: Finally, the decisions made in a universe might result in a dataset, making it impossible to fit a model to the data. While a universe in which models fail to fit usually does not result in a published study, we do want to report these outcomes for the sake of transparency. While usually, universes in which a model fails to fit would be excluded, we do count these, to understand to what extent empirical software engineering is susceptible to the file drawer bias.

First, we assign a universe to an outcome bucket for each of the four dependent variables in the study. Practically, this means that universe #*x* might confirm the results for hypothesis 1, but lead to opposite results for hypothesis 2.

As the primary study combines the outcomes of all four hypotheses to conclude that both the communication and the number of changes made in a code review change , we also assign each universe to a bucket based on the combination of outcomes for each dependent variable. If two or more hypotheses for one universe are in different buckets, we assign the lower bucket of the two. *i.e.*, if for one universe the outcome for *General Comments* confirms the primary study, but for *Commits After Create* the model fails to fit the entire study is placed into the bucket *Model Fit Failure*, as it would not be possible to confirm the findings of the primary study.

We use several complementary visualizations to explore the relation between universes and outcomes. Firstly, we count the number of universes belonging to each of the buckets, both for the entire study and for each of the dependent variables. While this initial counting gives an overview of how many different outcomes there are across the universes, it does not reveal the relation between outcomes and decision points. Therefore, we include the following visualizations:

Specification Curves. are a common tool used to analyze multiverse analysis [24]. These specification curves show the relation between individual decisions and the outcome of the multiverse analysis.

Stability of Individual Decisions. For each of the identified decision points, we look at the variability in outcomes, holding all other decisions constant. For instance, for the decision point *periodLength*, we look at the groups of universes where all decisions, except *periodLength*, remain constant. We then count the unique outcomes in each group and visualize the distribution. For decisions that do not affect outcomes, we expect all groups of universes to fall within the same bucket. Whereas, for decisions that affect the outcome, we expect to see large variation in outcomes across groups.

Timeframe studied. When performing an RDiT analysis, Hausman and Rapson [26] recommends against studying a time frame that is too long. As the studied time period becomes longer, it becomes harder to isolate the effect of the intervention on the time series. Therefore, other time-based effects (confounding factors) might introduce more noise,

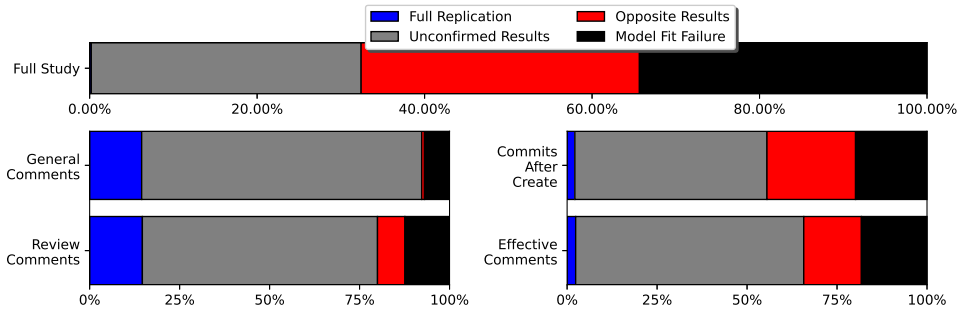

Fig. 3. High-level overview of the outcomes for all of the multiverses, per dependent variable.

thereby making it more difficult to isolate the impact of the intervention. Therefore, we plot the length of the studied time period in days against the number of different outcomes observed for that timeframe.

5 RESULTS

In this multiverse analysis, we explored 3,072 universes, representing the various reasonable alternatives studied. To understand the relation between data analytical decisions and study outcomes, we fitted 12,288 RDiT models. In this Section, we discuss the outcomes in each universe and show how particular choices made in constructing the universes influence the outcomes.

Figure 3 shows a high-level overview of the outcomes. Specifically, it shows how many of the multiverses have an outcome similar to that of the primary study. As discussed in Section 4.2, a universe has been assigned to one of the following outcome classes (*Full Replication*, *Unconfirmed Results*, *Opposite Results*, *Model Fit Failure*).

Most importantly, in only six universes (0.2%), the outcomes for each of the four dependent variables match the outcome of the original study. In $\sim$ 33% of the universes, no significance can be found for at least one of the claims of the original study. More worryingly, in another $\sim$ 33% of the universes, at least one opposite claim can be made (*i.e.*, the primary study reports an increasing trend, whereas, in the universe, a decreasing trend can be reported), and finally, in another $\sim$ 33% of the universes at least one of the four models could not be fit.

Secondly, Figure 3 shows a large difference in the outcomes for each dependent variable. For the dependent variables related to communication (*General Comments*, *Review Comments*), there are considerably more universes in which the outcome matches that of the primary study ($\sim$ 20%). Most importantly, for both General Comments and Review Comments, there is a low number of universes in which opposite results are found. Meanwhile, for the two dependent variables related to the number of changes made during a code review (*Commits After Create*, *Effective Comments*), the number of universes with outcomes similar to the outcome of the primary study is much lower. Even more importantly, for those dependent variables, there are many universes in which claims that are opposite to those of the original study can be made.

Finally, Figure 3 shows how combining outcomes from multiple models into one conclusion only makes it more likely for alternative decisions to result in different outcomes. As an alternative decision might result in an identical outcome for *General Comments*, but result in a different outcome for *Commits After Create*.

Finding

This multiverse analysis shows a small number of the explored universes lead to an outcome similar to the primary study. This shows how a high degree of researcher freedom might in and of itself threaten the validity of empirical software engineering studies .

5.1 Specification curve

To understand the relation between individual decisions and outcomes, Figure 4 shows a specification curve [6]. The plot consists of two parts: The top scatter plot and a row of bar plots. Each point on the top scatterplot represents the number of hypotheses in that universe that match the outcome of the primary study. Each plot below the top plot represents a specific decision, and the presence of a mark on the plot represents the value for the decision used in the universe. The colors represent the outcome categories for the universe, as used in Figure 3.

The distribution of vertical marks for a decision helps understand how a particular choice is related to a specific outcome. From the plot, we can, for instance, infer that the value of *Reml* does not appear to be related to any specific outcomes. Because the distribution of marks over the two values for *Reml* is very similar. However, for *Period Length* we can see that a value of 7 for never results in a universe where more than 6 of the original 12 hypotheses are replicated. At the same time, universes where a *Period Length* of either 30 or 45 is used appear to confirm more of the findings in the original study, and most importantly, the only universes in which all hypotheses are confirmed are the universes in which *Period Length* is set to 45 days.

From Figure 4, we conclude that the study's outcome is strongly influenced by decisions made to model time. When shorter time periods (7 or 15 days) are picked, it becomes impossible to make the same conclusions as the primary study, even when using the same data. Modeling time is known to be a challenging problem [10], and these findings confirm how impactful these decision points can be.

More importantly, Figure 4 allows us to conclude that many other data analytical decisions also affect findings. This includes, for instance, the scaling of variables or the method used to aggregate data . But also the number of datapoints excluded to account for a period of instability. While it is a common practice in RDiT literature to vary these exclusion periods [31, 32, 43, 67, 72], all universes where the exclusion period is shorter than 15 days result in outcomes that are different from those of the primary study. The specification curve shows how a wide variety of reasonable decisions can make it impossible to confirm the findings of the primary study.

Finding

The specification curve shows there is a small number of universes with results similar to the primary study, and how decisions related to the modeling of time in RDiT appear to relate to specific outcomes.

5.2 Change Plots

To complement the specification curve, and to better understand how "*stable*" each decision is, we look at all groups of universes where only a single decisions changed. Studying these groups helps quantify how often each specific decision results in an outcome change. Figure 5 plots a set of three histograms for each decision, each histogram shows the distribution of unique outcomes that can be obtained when changing that decision. For stable decisions, we expect to see that a change results in only one outcome being observed, no matter how many alternative values for that

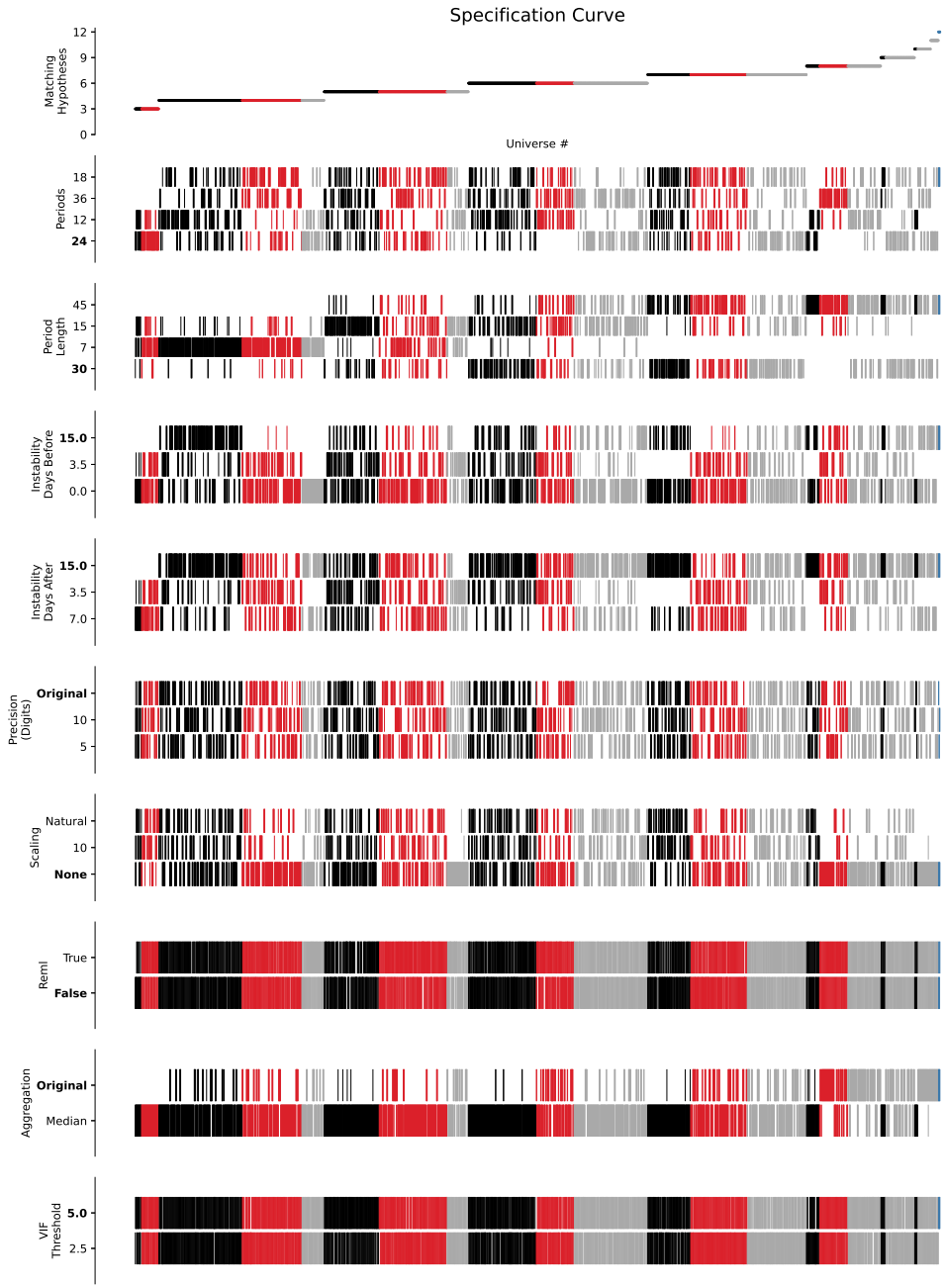

Fig. 4. A specification curve showing the number of hypotheses that can be confirmed in each universe, and the relation between a universe, and the decisions made within in the universe.

decisions are explored. Meanwhile, for unstable decisions the distribution will be skewed towards a higher number of unique outcomes.

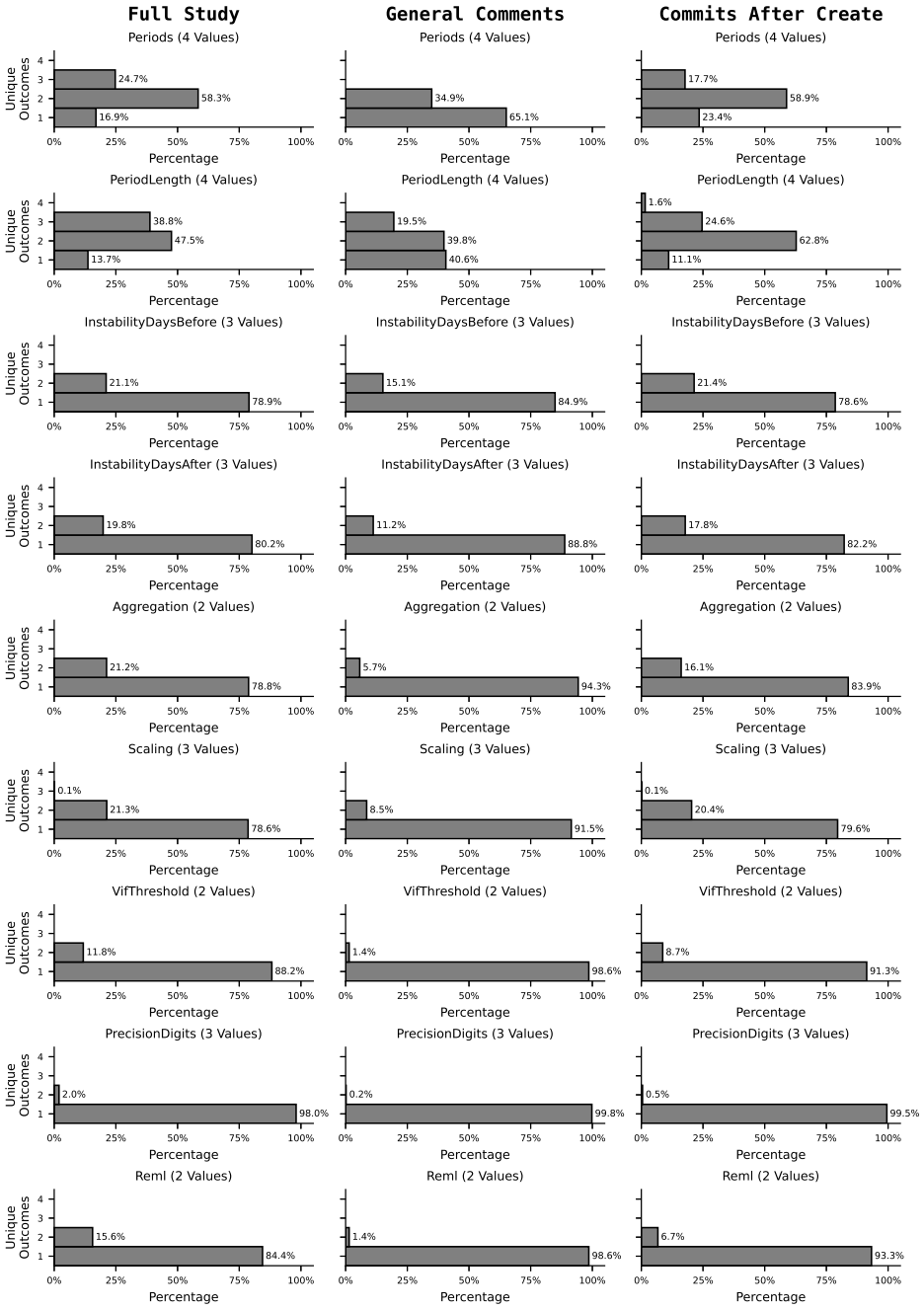

Fig. 5. Distribution plots showing the percentage of universes in which changing one decision leads to an alternative outcome.

Figure 5 confirms that changing time-related decisions, one of the core methodological decisions of RDiT strongly affects outcomes. However, most importantly, it also shows how each decision point can affect the outcome. Even

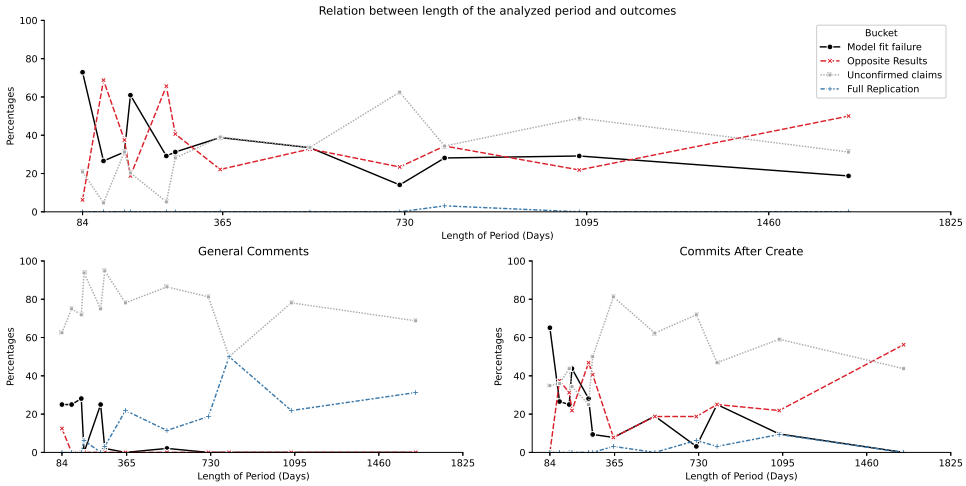

Fig. 6. Relation between the length of the studied time period and the outcomes.

alternatives for the Vif Threshold, or the Reml value, result in a different outcome in ∼ 10% of cases. Finally, most surprisingly, the plot shows that in 2% of universes, even rounding numbers to a different level of precision can change the outcome. As unlikely as that may seem, we believe it is essential to draw attention to this finding: The digits of precision used to instantiate a data frame are an excellent example of a **hidden** decision, a choice made by the default setting of a programming environment, and not something you would expect to be influential. Nevertheless, we find that changing it can lead to a different outcome.

A careful reading of Figure 5 also shows what the most impactful decisions. For the full study, only changing the *Period Length* resulted in a different outcome in 86.3% of universes. When looking at specific hypotheses (*General Comments* and *Commits After Create*), it remains the most impactful decision. The decision points related to the processing of data, like *Aggregation* and *Scaling* are more stable, but can still lead to meaningfully different outcomes in between 20% to 5% of universes.

Finding

Figure 5 shows there is not a single stable methodological decision . In this multiverse analysis Period Length *is the most impactful decision leading to a different outcome in* 86.3% *of universes. Figure 5 also shows how decisions that are easy to make implicitly, like the number of digits of precision to round to, can also change outcomes.*

5.3 Time Curves

To understand whether a multiverse analysis can help inform how time should be modeled, Figure 6 plots the relation between the length of the studied time-period, and the outcomes. The top line plot shows the distribution of outcomes for the full study, while the two bottom plots show outcomes for two dependent variables: *General Comments* and *Commits After Create.*

Figure 6 shows how outcomes can vary greatly when a shorter period of time is studied – even resulting in a high number of opposite outcomes. For periods shorter than a year, the likelihood of obtaining different outcomes is high,

as the frequency of different outcomes varies greatly, and even for minor variations in the studied period result in a model failing to fit, or concluding an opposite result. As the length of the total time period increases, the outcomes tend to vary less. However, there is still a noticeable increase in the number of universes in which opposite results can be reported.

The bottom two plots of Figure 6 show that the pattern differs for different dependent variables. For *General Comments*, there is a much smaller set of universes that would report opposite results. Meanwhile, this percentage of universes is much higher for *Commits After Create*. This difference across dependent variables and between the entire study and individual dependent variables shows how the same decision might affect different dependent variables.

This plot shows the difficulty in picking a stable timeframe to study using RDiT. The literature recommends studying shorter timeframes—or timeframes that are as short as possible. As Hausman and Rapson [26] argues, the risk of unrelated time-based effects (confounding variables) masking the impact of the dependent variable increases as the studied timeframe becomes longer. However, In this case, Figure 6 clearly shows that using a short timeframe introduces a great variation in outcomes. This implies that with the current level of knowledge not enough is known to justify the selection of a single timeframe.

6 DISCUSSION

This study applied a multiverse analysis to a dataset from a published MSR study, mapping nine key methodological decisions and plausible choices for each of them into 3,072 different analysis pipelines. We found that only six of those analytical universes reproduced the findings of the original study. This imbalance reveals a core tension in empirical software engineering. While the richness of data is often celebrated as a promise [5, 30], it also creates analytical fragility by giving researchers a high degree of freedom in decisions. In other words, our findings demonstrate how the greatest promise of empirical software engineering, and MSR in particular , the richness of the data, might also be one of its great perils. Data analysis pipelines in MSR studies typically require many decisions, often made without explicit guidelines, further amplifying this fragility. Gelman and Loken theorized in their work on the "garden of forking paths" [21] how methodological flexibility might affect robustness. In this paper we confirm this idea, and show how *a high degree of researcher freedom can cause even a single software engineering dataset and research question to yield many, sometimes conflicting, outcomes.*

Our work extends previous observations of conflicting empirical software engineering outcomes (e.g., Wessel et al.'s meta-analysis of RDiT studies [68]) provides evidence for the fact that much of that conflict can arise from opaque decision points that are varied across studies. While we find that the most influential decision points are those related to the modeling of time, we find that **all** of the studied decision points affect the outcome of the study. In our opinion, the implicit, or hidden, decision points, that are normally not extensively considered are the most problematic.

Furthermore, we believe that the SE field's traditional tools for addressing threats to validity are insufficient to address this threat. Practices like discussing threats to validity and their mitigation [1, 16, 42, 69] or discussing study trade-offs [48] often amount to afterthoughts or are used simply as checklists [34, 65]. Increasing the complexity or type of data analysis [19], standardizing the peer review process with empirical standards [15] or expanding threat discussions will not solve the underlying problem introduced by methodological freedom. Instead, we believe that empirical software engineering research needs more structure to reason about analytical decisions, and especially, to reason about the justification for specific analytical decisions.

6.1 Justification Ladder of Analytical Choices

We propose the "Justification Ladder of Analytical Choices" (JLAC), shown in Table 5 to support stronger analytical justifications for data analysis choices. This model, developed from our experience with the multiverse analysis presented in this paper, offers empirical software engineering researchers a structured way to classify and, ideally, strengthen justifications for methodological decisions. This justification ladder complements existing initiatives to improve methodological rigor. For instance, where the empirical standard focuses on **what** should be justified [15], the justification ladder helps distinguish specific **types** of justification. Below, we describe the JLAC in more detail and explain how it can help researchers make their analytical decisions more transparent and build greater trust in MSR and, more generally, in empirical software engineering studies.

Table 5: The Justification Ladder of Analytical Choices (JLAC) with stronger and more preferable levels of justification on higher levels (5) of the ladder .

Level	Name	Description
5	Causal	These decisions are made based on causal models or theories of the problem, or domain, under study.
4	Empirical	These decisions are justified based on properties of the data under study, but these decisions are not necessarily linked to a causal model
3	Heuristic	These are decisions made based on general heuristics, often made based on reasoning that it is not directly tied to the domain, or the research question being studied.
2	Conventional	These decisions are made based on existing conventions, where the justification of the decision is made because it is common practice in the field.
1	Implicit	These are decisions that are made implicitly, without the researcher being aware the decisions is being made. These could be defaults of the statistics package being used.

Following the scientific method, researchers begin with a theory to generate hypotheses, then design studies and analyses to test those hypotheses—each step producing falsifiable predictions. When analytic choices ((e.g., variable selection, data transformation, modeling relationships) are derived from a detailed causal theory, two key benefits arise: (a) assumptions are explicitly articulated, tying every transformation or filter to a hypothesized mechanism rather than leaving them to software defaults or chance; and (b) reproducibility is enhanced, as other teams applying the same causal model are more likely to make similar decisions.

In contrast, selecting analyses based on (potentially) empirical quirks of a dataset risks circular reasoning and overfitting. By grounding analytic pipelines in causal models, we transform them into disciplined, theory-driven experiments. This is why detailed scientific theories should be the preferred basis for analytic choices.

Decisions made implicitly or without explicit discussion are more likely to introduce unintended biases. Similarly, choices based on conventions or general heuristics often drift away from the specific domain or context of a study, providing weaker justifications than the data at hand. However, while causal and empirical justifications are preferable, we recognize that theoretical knowledge may not always be available or practical to apply. Thus, our model organizes types of justifications into a hierarchy, from weaker to stronger, reflecting their desirability.

Table 5 lists the types of justifications we distinguish. We argue that empirical software engineering researchers should aim to ground data analytic decisions as high up this justification ladder as possible, ideally using verified causal models to inform analytical choices.

Prior work supports this need to "move up the ladder". Kale *et al.* found that many published studies base decisions on conventions or leave them undiscussed [29]. Conflicting operationalizations [71] and divergent methodological decisions for similar research questions complicate meta-studies, and Wyrich and Apel [70] emphasize the need to make data-driven decisions to quantify threats to validity. This further highlights the need for better tools to help researchers justify their choices more rigorously, thereby enhancing the reliability of individual studies and enabling meta analyses to more effectively compare and synthesize findings.

Furthermore, Gelman and Loken [21] argues that when analytic decisions are shaped by the data itself, findings should be treated as exploratory rather than confirmatory. Our multiverse analysis suggests that this caution also applies to MSR studies that rely on implicit defaults or widely adopted conventions without explicit and strong justification. In such cases, even studies presented as confirmatory may simply reflect one path through a forest of equally plausible analyses. Following Gelman and Loken [21], we contend that these studies might be better characterized as exploratory. We therefore recommend that authors acknowledge these dependencies and, where possible, reinforce their confirmatory claims with sensitivity analyses or clear, theory-driven rationales for their analytic choices.

6.2 Applying JLAC to RDiT Studies

The Justification Ladder of Analytical Choices (JLAC) can be concretely illustrated through the motivations typically reported for the time-series design decisions in Regression Discontinuity in Time (RDiT) studies. In Table 6 we map the three temporal decisions—number of periods, period length, and instability exclusion window—to their corresponding levels on the JLAC.[3] By examining how prior RDiT studies in software engineering justify these choices, we can assess prevailing justification practices and identify opportunities to strengthen them.

Number of periods. Some RDiT studies provide at least a partial rationale for their chosen number of periods, often linked to data availability or pragmatic considerations. For instance, Cassee [7] modeled twelve months before and after the introduction of continuous integration (CI) to ensure that each bucket contained a sufficient number of projects for model estimation (*Level 3, Heuristic*). Zimmermann [73] adopted a conservative specification using a smaller bandwidth to minimize bias but tested an extended window (511 days on each side) as a robustness check (*Level 4, Empirical*). Walden [66] selected 25 months on each side to balance sufficient data points against the risk of confounding factors far from the cutoff (*Level 3, Heuristic*). Saraiva [52] used 24 versions (12 before and 12 after the split event) to maintain symmetry around the intervention point (*Level 3, Heuristic*). Other RDiT papers provide no motivation for this decision.

Period length. None of the reviewed RDiT papers explicitly justify their choice of period length. In all examined cases, the period (e.g., 30 days) seems to be inherited from established practice or software defaults, with no discussion of alternative temporal granularities or their possible impact on the results. According to the JLAC, this lack of explicit reasoning corresponds to Level 1 (Implicit). One might argue that the near-universal use of similar period lengths makes this a de facto community convention and thus closer to Level 2. However, when the choice is neither mentioned nor reflected upon, it is difficult to claim that it was consciously made, so we consider Level 1 the more appropriate classification.

[3]It is important to note that our mapping is based on the interpretations of each of the manuscripts; it might not take into account justifications not explicitly listed in the manuscript.

Instability exclusion window. The choice of how much data to exclude around the intervention event is sometimes justified empirically, but it is mostly based on anecdotal observations, partial observations, or on convention. A prime example of empirical validation is the donut RDDs investigated by Zimmermann and Artis [73]. Meanwhile, Zhao [72] excluded one month of data centered on the CI adoption event, motivated by informal observations of restructuring activity during this period (e.g., changes in build systems or dependencies; Level 3, heuristic). Although one might argue this approaches Level 4, there is no systematic, quantitative analysis to support that classification, so we retain it as Level 3. Cassee [7], Kinsman [32], and Saraiva [52] then followed this convention, excluding 15 days before and after the intervention (Level 2, Conventional). The other studies did not motivate or discuss this choice at all.

Table 6. Application of the Justification Ladder of Analytical Choices (JLAC) to the main time-series design decisions in RDiT studies.

Decision Point	**Example Choices Motivations**	**JLAC Levels**	**Common Limitations**
Number of periods	Ensuring sufficient data or balancing sample size and bias [7]	**4**: [73]; **3**: [7, 52, 66]; **1**: [3, 23, 31, 32, 35, 43, 44, 64, 67, 72]	Often not discussed (i.e. level 1, implicit), driven by heuristic reasoning on "balancing" data or the number of periods (3), or, in one case, based on simulation on data (4).
Period length	Typically 30 days but rarely discussed nor motivated	**1**: [3, 7, 23, 31, 32, 35, 43, 44, 52, 64, 66, 67, 72, 73]	No motivation given, so likely chosen for convenience/ convention, ignoring alternative granularities or behavioral rhythms
Instability exclusion window	Exclusion of 15 days around the intervention to remove transitional noise, based on empirical testing for Zimmermann and Artis [73] , and anectdotal observations in some projects [72], which then became convention [7, 32, 52]	**4**: [73] ; **3**: [72]; **2**: [7, 32, 52] **1**: [3, 23, 31, 35, 43, 44, 52, 64, 66, 67]	Mostly motivated by anecdotal observations rather than quantitative or theoretical grounding, rest motivated by convention or didn't motivate.

Beyond the temporal design choices, five additional analytical decisions were explored in the multiverse analysis: aggregation method, scaling transformation, rounding precision, variance-inflation threshold, model-fitting algorithm (REML or not). While typically not specifically discussed or motivated in the RDiT literature, these decisions largely reflect standard statistical or implementation defaults. For example, the aggregation and scaling choices are often heuristic (Level 3) or by convention (Level 2), while rounding and estimation method defaults remain implicit (Level 1). Together, these patterns reinforce that analytical justification in current software-engineering RDiT studies is concentrated in the lower and middle levels of the JLAC (Levels 1–3), indicating opportunity for methodological strengthening.

Because our JLAC assessment is based solely on what is reported in published manuscripts, it inevitably offers an incomplete picture of how methodological decisions were actually made. Some justifications may have been omitted due to page limits, stylistic choices, or assumptions about what readers consider obvious. For this reason, we do not position the JLAC as a tool for retrospectively judging existing studies; instead, we see it as a forward-looking aid for researchers to structure and scrutinize their own decision points in ongoing and future work. When planning and conducting quantitative analyses, we hope the JLAC will help researchers articulate, compare, and strengthen their methodological justifications in a more systematic way.

6.3 Strengthening Justifications of Analytical Decisions

Our findings and the argument above raise an important question: *What can be done to strengthen the justification of analytical decisions?*

We start by discussing several ways to strengthen methodological decisions in RDiT studies. For example, the *number of periods* could be empirically validated against the minimum data window needed for stable coefficient estimation; *period length* could be aligned with theorized or empirically investigated release or contribution cycles in software development; and the *instability exclusion windows* could be derived from quantitative analyses of recovery or adaptation time after interventions. Grounding such decisions in either observed data patterns or process theory would elevate their justification from Levels 1–3 toward Levels 4 and potentially even 5, enhancing both methodological transparency and the interpretability of RDiT results.

Beyond RDiT, several tools and methods can help move justifications for data-analytical decisions higher up the justification ladder. We envision this as a three-step process. First, identify key decision points. Second, make these decision points and their viable alternatives explicit. Third, evaluate and compare the alternatives to select those most appropriate for the study. Researchers can rely on a range of strategies at this third step, from drawing on established theories to applying quantitative techniques such as simulation-based evaluation.

One way to inform and justify methodological decisions is to draw on existing scientific theory [60]. Well-developed theories provide a strong foundation for choosing models, variables, and analytical strategies. Practical approaches for building and evaluating such theories include causal modeling [19, 20, 40] and structural equation modeling [49, 63], both of which have already been successfully applied in software engineering contexts.

Although justifying every methodological decision with established theory would be ideal, it is rarely feasible in practice. Fortunately, several other approaches can substantially strengthen justifications and guide decisions. One promising option is to use simulated data to test whether a proposed analysis method is appropriate Härtel and Lämmel [27, 28]; by generating data with known effect sizes, researchers can compare alternative choices at key decision points—for example, in an RDiT study, confirming that a specific combination of time-modeling decisions can recover a small effect in noisy, multi-level data. A second approach is pre-registration [14], which reduces the likelihood that study outcomes depend on post hoc analytical choices. A third is mixed-methods research, where MSR studies are complemented with other methods to triangulate and confirm findings [61].

Finally, our study highlights the value of multiverse analyses–systematically exploring alternative analytic decisions–as an effective "smoke test" for assessing the sensitivity of study outcomes. In domains with high methodological freedom, multiverse analyses can map out the range of plausible conclusions that can be drawn from the same data. If a multiverse analysis shows that a single data source supports conflicting conclusions under reasonable alternatives, this should be interpreted as a **red flag**, signalling that domain knowledge is too limited to support the planned analysis with confidence. In such cases, the research community should aim to reduce unnecessary methodological freedom,

constrain analytical choices, and move higher up the ladder of justifications by applying the tools and methods discussed in this section.

7 LIMITATIONS

The multiverse analysis described in this manuscript has its own set of limitations, which stem from our methodological decisions rather than the limitations of the primary study itself.

First, our analysis is restricted by issues of **external validity**. By focusing on a single MSR study, we cannot assume that our findings generalize across the entire MSR literature. We mitigated this risk by selecting a research method, trace-data analysis, that is widely used in software engineering, and by building on a meta study that has already documented conflicting outcomes across RDiT applications [68]. Nonetheless, our conclusions remain specific to this case.

Second, we were unable to revisit all data collection choices made in the original study, particularly the criteria for selecting repositories (e.g., minimum stars, contributor counts). This limitation also touches on **external validity**: although our multiverse explores many decision paths, it does not encompass the full space of repository-selection strategies used across MSR. However, the dependencies we uncover between analytic decisions and outcomes hold for the data set at hand.

Third, a key strength of a multiverse analysis is that it can vary operational definitions, but this very variation introduces a risk to **construct validity**. By systematically changing measures of a specific construct, thresholds for inclusion, and data-cleaning rules, some analytic branches may no longer align well with the original theoretical constructs. Future work could empirically assess which combinations of operational choices best map onto the underlying constructs of interest.

Finally, and clearly, our multiverse analysis does not address the limitations of the primary study's **internal validity** (e.g., causal claims, confounding factors) and therefore does not resolve any threats to that study's design.

Taken together, these limitations underscore that while a multiverse analysis offers better transparency into the effect of methodological choices on results, it also inherits, and in some cases may amplify, the validity concerns of both the primary study and the analytic processes employed.

8 CONCLUSION

The replication crisis, triggered by a series of high-profile failed replications, has become a cornerstone of meta-scientific research on how to improve the reliability of science. Gelman and Loken [21] proposed that one contributor to this crisis is the wide methodological freedom researchers often have when making analytic decisions. Silberzahn et al. [57] empirically confirmed this by demonstrating that different research teams, given the same dataset and research question, frequently reach different conclusions because of variations in their methodological choices.

Empirical software engineering, and Mining Software Repositories (MSR) research in particular, shares this freedom. The richness of data is often presented as a great promises [5, 30], but it also raises concerns about how methodological decisions might shape study outcomes. To investigate this, we selected a published MSR study that employed a data-analytic method allowing considerable researcher freedom. We then conducted a multiverse analysis: systematically exploring reasonable alternatives to the original study's methodological decisions and reporting the outcomes across these alternatives.

Our analysis considered 3,072 alternative "universes," each corresponding to a unique combination of nine methodological decisions. We found that every one of the nine decisions could change the study's results. In fact, only 6 out

of the 3,072 explored universes replicated the original study's findings. This highlights that choices in processing and analyzing post-hoc observational data commonly used in empirical software engineering data may influence outcomes far more than previously assumed.

These results indicate a need for stronger methodological rigor in empirical software engineering . Analytical choices should be grounded in causal or theoretical frameworks when possible. When such grounding proves infeasible, validation via simulation, mixed-methods research with triangulation, or targeted sensitivity checks can help avoid fragile conclusions. Authors might consider conducting and reporting multiverse analyses to reveal how different analytical paths affect their findings. They can also adopt the Justification Ladder of Analytical Choices, which we proposed based on our experiences conducting this multiverse analysis, to document and defend their analytical decisions. Finally, specifically for RDiT studies (and other time-dependent analyses in MSR), defining and justifying domain-specific guidelines for period length and instability exclusion can reduce outcome volatility. Since we have presented only a single case future work should consider multiverse analysis also of other MSR studies and for other types of design.

ACKNOWLEDGMENTS

We acknowledge support from the Science Council (Vetenskapsrådet, contract id 2020-05272) for the project "Automated Boundary Testing for AI/ML models" and from WASP for the project "Bound-Miner". We also want to sincerely thank Julian Frattini, Alexander Serebrenik, and Richard Torkar for their feedback on an earlier version of this paper!

REFERENCES

[1] Apostolos Ampatzoglou, Stamatia Bibi, Paris Avgeriou, Marijn Verbeek, and Alexander Chatzigeorgiou. 2019. Identifying, categorizing and mitigating threats to validity in software engineering secondary studies. *Information and software technology* 106 (2019), 201–230.

[2] Andrea Arcuri and Lionel Briand. 2014. A Hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering. *Software Testing, Verification and Reliability* 24 (5 2014), 219–250. Issue 3. https://doi.org/10.1002/stvr.1486

[3] Patrick Ayoup, Diego Elias Costa, and Emad Shihab. 2022. Achievement unlocked: a case study on gamifying DevOps practices in industry. *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 1343–1354. https://doi.org/10.1145/3540250.3558948

[4] Samuel J. Bell, Onno P. Kampman, Jesse Dodge, and Neil D. Lawrence. 2022. Modeling the Machine Learning Multiverse. arXiv:2206.05985 [cs.LG] https://arxiv.org/abs/2206.05985

[5] Christian Bird, Peter C. Rigby, Earl T. Barr, David J. Hamilton, Daniel M. German, and Prem Devanbu. 2009. The promises and perils of mining git. *2009 6th IEEE International Working Conference on Mining Software Repositories*, 1–10. https://doi.org/10.1109/MSR.2009.5069475

[6] Nate Breznau et al. 2022. Observing many researchers using the same data and hypothesis reveals a hidden universe of uncertainty. *Proceedings of the National Academy of Sciences* 119 (11 2022). Issue 44. https://doi.org/10.1073/pnas.2203150119

[7] Nathan Cassee, Bogdan Vasilescu, and Alexander Serebrenik. 2020. The Silent Helper: The Impact of Continuous Integration on Code Reviews. In *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. 423–434. https://doi.org/10.1109/SANER48275.2020.9054818

[8] Chris Chambers. 2017. *The Seven Deadly Sins of Psychology.* Princeton University Press. https://doi.org/10.2307/j.ctvc779w5

[9] Andy Cockburn, Pierre Dragicevic, Lonni Besançon, and Carl Gutwin. 2020. Threats of a replication crisis in empirical computer science. *Commun. ACM* 63 (7 2020), 70–79. Issue 8. https://doi.org/10.1145/3360311

[10] Jonathan D. Cryer and Kung-Sik Chan. 2008. *Time Series Analysis.* Springer New York. https://doi.org/10.1007/978-0-387-75959-3

[11] Francisco Gomes de Oliveira Neto, Richard Torkar, Robert Feldt, Lucas Gren, Carlo A. Furia, and Ziwei Huang. 2019. Evolution of statistical analysis in empirical software engineering research: Current state and steps forward. *Journal of Systems and Software* 156 (10 2019), 246–267. https://doi.org/10.1016/j.jss.2019.07.002

[12] Pierre Dragicevic, Yvonne Jansen, Abhraneel Sarma, Matthew Kay, and Fanny Chevalier. 2019. Increasing the Transparency of Research Papers with Explorable Multiverse Analyses. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (New York, NY, USA). ACM, 1–15. https://doi.org/10.1145/3290605.3300295

[13] Tore Dybå, Vigdis By Kampenes, and Dag I.K. Sjøberg. 2006. A systematic review of statistical power in software engineering experiments. *Information and Software Technology* 48 (8 2006), 745–755. Issue 8. https://doi.org/10.1016/j.infsof.2005.08.009

[14] Neil A Ernst and Maria Teresa Baldassarre. 2023. Registered reports in software engineering. *Empirical software engineering* 28, 2 (2023), 55.

[15] Paul Ralph et al. 2020. ACM SIGSOFT Empirical Standards. *CoRR* abs/2010.03525 (2020). arXiv:2010.03525 https://arxiv.org/abs/2010.03525

[16] Robert Feldt and Ana Magazinius. 2010. Validity threats in empirical software engineering research-an initial survey.. In *Seke*. 374–379.

[17] Isabella Ferreira, Bram Adams, and Jinghui Cheng. 2022. How heated is it?. In *Proceedings of the 19th International Conference on Mining Software Repositories* (New York, NY, USA). ACM, 309–320. https://doi.org/10.1145/3524842.3527957

[18] Julian Frattini, Davide Fucci, Richard Torkar, and Daniel Mendez. 2024. A Second Look at the Impact of Passive Voice Requirements on Domain Modeling: Bayesian Reanalysis of an Experiment. In *Proceedings of the 1st IEEE/ACM International Workshop on Methodological Issues with Empirical Studies in Software Engineering* (Lisbon, Portugal) *(WSESE '24)*. Association for Computing Machinery, New York, NY, USA, 27–33. https://doi.org/10.1145/3643664.3648211

[19] Carlo Alberto Furia, Robert Feldt, and Richard Torkar. 2019. Bayesian Data Analysis in Empirical Software Engineering Research. *IEEE Transactions on Software Engineering* (2019), 1–1. https://doi.org/10.1109/TSE.2019.2935974

[20] Carlo A. Furia, Richard Torkar, and Robert Feldt. 2023. Towards Causal Analysis of Empirical Software Engineering Data: The Impact of Programming Languages on Coding Competitions. *ACM Transactions on Software Engineering and Methodology* 1 (11 2023). Issue 1. https://doi.org/10.1145/3611667

[21] Andrew Gelman and Eric Loken. 2013. The garden of forking paths: Why multiple comparisons can be a problem, even when there is no "fishing expedition" or "p-hacking" and the research hypothesis was posited ahead of time. *Department of Statistics, Columbia University* 348 (2013), 1–17.

[22] Marco Del Giudice and Steven W. Gangestad. 2021. A Traveler's Guide to the Multiverse: Promises, Pitfalls, and a Framework for the Evaluation of Analytic Decisions. *Advances in Methods and Practices in Psychological Science* 4 (1 2021), 251524592095492. Issue 1. https://doi.org/10.1177/2515245920954925

[23] Yunfang Guo and Philipp Leitner. 2019. Studying the impact of CI on pull request delivery time in open source projects—a conceptual replication. *PeerJ Computer Science* 5 (12 2019), e245. https://doi.org/10.7717/peerj-cs.245

[24] Brian D. Hall, Yang Liu, Yvonne Jansen, Pierre Dragicevic, Fanny Chevalier, and Matthew Kay. 2022. A Survey of Tasks and Visualizations in Multiverse Analysis Reports. *Computer Graphics Forum* 41 (2 2022), 402–426. Issue 1. https://doi.org/10.1111/cgf.14443

[25] Jenna A. Harder. 2020. The Multiverse of Methods: Extending the Multiverse Analysis to Address Data-Collection Decisions. *Perspectives on Psychological Science* 15 (9 2020), 1158–1177. Issue 5. https://doi.org/10.1177/1745691620917678

[26] Catherine Hausman and David S. Rapson. 2018. Regression Discontinuity in Time: Considerations for Empirical Applications. *Annual Review of Resource Economics* 10 (10 2018), 533–552. Issue 1. https://doi.org/10.1146/annurev-resource-121517-033306

[27] Johannes Härtel and Ralf Lämmel. 2022. Operationalizing Threats to MSR Studies by Simulation-Based Testing. In *2022 IEEE/ACM 19th International Conference on Mining Software Repositories (MSR)*. 86–97. https://doi.org/10.1145/3524842.3527960

[28] Johannes Härtel and Ralf Lämmel. 2023. Operationalizing validity of empirical software engineering studies. *Empirical Software Engineering* 28 (11 2023), 153. Issue 6. https://doi.org/10.1007/s10664-023-10370-3

[29] Alex Kale, Matthew Kay, and Jessica Hullman. 2019. Decision-Making Under Uncertainty in Research Synthesis. *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 1–14. https://doi.org/10.1145/3290605.3300432

[30] Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M. German, and Daniela Damian. 2016. An in-depth study of the promises and perils of mining GitHub. *Empirical Software Engineering* 21 (10 2016), 2035–2071. Issue 5. https://doi.org/10.1007/s10664-015-9393-5

[31] David Kavaler, Asher Trockman, Bogdan Vasilescu, and Vladimir Filkov. 2019. Tool Choice Matters: JavaScript Quality Assurance Tools and Usage Outcomes in GitHub Projects. *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, 476–487. https://doi.org/10.1109/ICSE.2019.00060

[32] Timothy Kinsman, Mairieli Wessel, Marco A. Gerosa, and Christoph Treude. 2021. How Do Software Developers Use GitHub Actions to Automate Their Workflows? *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*, 420–431. https://doi.org/10.1109/MSR52588.2021.00054

[33] Barbara Kitchenham, Lech Madeyski, David Budgen, Jacky Keung, Pearl Brereton, Stuart Charters, Shirley Gibbs, and Amnart Pohthong. 2017. Robust Statistical Methods for Empirical Software Engineering. *Empirical Software Engineering* 22 (4 2017), 579–630. Issue 2. https://doi.org/10.1007/s10664-016-9437-5

[34] Patricia Lago, Per Runeson, Qunying Song, and Roberto Verdecchia. 2024. Threats to Validity in Software Engineering–hypocritical paper section or essential analysis?. In *Proceedings of the 18th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. 314–324.

[35] Zhixing Li, Yue Yu, Tao Wang, Yan Lei, Ying Wang, and Huaimin Wang. 2023. To Follow or Not to Follow: Understanding Issue/Pull-Request Templates on GitHub. *IEEE Transactions on Software Engineering* 49 (4 2023), 2530–2544. Issue 4. https://doi.org/10.1109/TSE.2022.3224053

[36] Pei Liu, Mattia Fazzini, John Grundy, and Li Li. 2022. Do customized Android frameworks keep pace with Android?. In *Proceedings of the 19th International Conference on Mining Software Repositories* (New York, NY, USA). ACM, 376–387. https://doi.org/10.1145/3524842.3527963

[37] Yang Liu, Alex Kale, Tim Althoff, and Jeffrey Heer. 2021. Boba: Authoring and Visualizing Multiverse Analyses. *IEEE Transactions on Visualization and Computer Graphics* 27 (2 2021), 1753–1763. Issue 2. https://doi.org/10.1109/TVCG.2020.3028985

[38] Alvi Mahadi, Karan Tongay, and Neil A. Ernst. 2020. Cross-Dataset Design Discussion Mining. In *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 149–160. https://doi.org/10.1109/SANER48275.2020.9054792

[39] Pooya Rostami Mazrae, Tom Mens, Mehdi Golzadeh, and Alexandre Decan. 2023. On the usage, co-usage and migration of CI/CD tools: A qualitative analysis. *Empirical Software Engineering* 28 (3 2023), 52. Issue 2. https://doi.org/10.1007/s10664-022-10285-5

[40] Richard McElreath. 2018. *Statistical Rethinking*. Chapman and Hall/CRC. https://doi.org/10.1201/9781315372495

[41] Daniel Méndez Fernández, Martin Monperrus, Robert Feldt, and Thomas Zimmermann. 2019. The open science initiative of the Empirical Software Engineering journal. *Empirical Software Engineering* 24 (2019), 1057–1060.

[42] Tim Menzies and Martin Shepperd. 2019. "Bad smells" in software analytics papers. *Information and Software Technology* 112 (8 2019), 35–47. https://doi.org/10.1016/j.infsof.2019.04.005

[43] Ambarish Moharil, Dmitrii Orlov, Samar Jameel, Tristan Trouwen, Nathan Cassee, and Alexander Serebrenik. 2022. Between JIRA and GitHub: ASFBot and its influence on human comments in issue trackers. *Proceedings of the 19th International Conference on Mining Software Repositories*, 112–116. https://doi.org/10.1145/3524842.3528528

[44] Lukas Moldon, Markus Strohmaier, and Johannes Wachs. 2021. How Gamification Affects Software Developers: Cautionary Evidence from a Natural Experiment on GitHub. *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, 549–561. https://doi.org/10.1109/ICSE43902.2021.00058

[45] Nuthan Munaiah, Steven Kroh, Craig Cabrey, and Meiyappan Nagappan. 2017. Curating GitHub for engineered software projects. *Empirical Software Engineering* 22 (12 2017), 3219–3253. Issue 6. https://doi.org/10.1007/s10664-017-9512-6

[46] Emerson Murphy-Hill, Ciera Jaspan, Caitlin Sadowski, David Shepherd, Michael Phillips, Collin Winter, Andrea Knight, Edward Smith, and Matthew Jorde. 2021. What Predicts Software Developers' Productivity? *IEEE Transactions on Software Engineering* 47 (3 2021), 582–594. Issue 3. https://doi.org/10.1109/TSE.2019.2900308

[47] Rolando P. Reyes, Oscar Dieste, Efraín R. Fonseca, and Natalia Juristo. 2018. Statistical errors in software engineering experiments. In *Proceedings of the 40th International Conference on Software Engineering* (New York, NY, USA). ACM, 1195–1206. https://doi.org/10.1145/3180155.3180161

[48] Martin P. Robillard, Deeksha M. Arya, Neil A. Ernst, Jin L.C. Guo, Maxime Lamothe, Mathieu Nassif, Nicole Novielli, Alexander Serebrenik, Igor Steinmacher, and Klaas-Jan Stol. 2024. Communicating Study Design Trade-offs in Software Engineering. *ACM Transactions on Software Engineering and Methodology* (3 2024). https://doi.org/10.1145/3649598

[49] Daniel Russo and Klaas-Jan Stol. 2022. PLS-SEM for Software Engineering Research. *Comput. Surveys* 54 (5 2022), 1–38. Issue 4. https://doi.org/10.1145/3447580

[50] Andrea Saltelli, Ksenia Aleksankina, William Becker, Pamela Fennell, Federico Ferretti, Niels Holst, Sushan Li, and Qiongli Wu. 2019. Why so many published sensitivity analyses are false: A systematic review of sensitivity analysis practices. *Environmental Modelling & Software* 114 (4 2019), 29–39. https://doi.org/10.1016/j.envsoft.2019.01.012

[51] Adrian Santos, Sira Vegas, Markku Oivo, and Natalia Juristo. 2021. Comparing the results of replications in software engineering. *Empirical Software Engineering* 26 (3 2021), 13. Issue 2. https://doi.org/10.1007/s10664-020-09907-7

[52] Diego Saraiva, Daniel Alencar Da Costa, Uirá Kulesza, Gustavo Sizílio, José Gameleira Neto, Roberta Coelho, and Meiyappan Nagappan. 2023. Unveiling the Relationship Between Continuous Integration and Code Coverage. *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*, 247–259. https://doi.org/10.1109/MSR59073.2023.00043

[53] Marko Sarstedt, Susanne J. Adler, Christian M. Ringle, Gyeongcheol Cho, Adamantios Diamantopoulos, Heungsun Hwang, and Benjamin D. Liengaard. 2024. Same model, same data, but different outcomes: Evaluating the impact of method choices in structural equation modeling. *Journal of Product Innovation Management* 41 (11 2024), 1100–1117. Issue 6. https://doi.org/10.1111/jpim.12738

[54] Martin Schweinsberg et al. 2021. Same data, different conclusions: Radical dispersion in empirical results when independent analysts operationalize and test the same hypothesis. *Organizational Behavior and Human Decision Processes* 165 (7 2021), 228–249. https://doi.org/10.1016/j.obhdp.2021.02.003

[55] Martin Shepperd, Nemitari Ajienka, and Steve Counsell. 2018. The role and value of replication in empirical software engineering results. *Information and Software Technology* 99 (2018), 120–132.

[56] Martin Shepperd, David Bowes, and Tracy Hall. 2014. Researcher Bias: The Use of Machine Learning in Software Defect Prediction. *IEEE Transactions on Software Engineering* 40 (6 2014), 603–616. Issue 6. https://doi.org/10.1109/TSE.2014.2322358

[57] Raphael Silberzahn et al. 2018. Many Analysts, One Data Set: Making Transparent How Variations in Analytic Choices Affect Results. *Advances in Methods and Practices in Psychological Science* 1 (9 2018), 337–356. Issue 3. https://doi.org/10.1177/2515245917747646

[58] Uri Simonsohn, Joseph P. Simmons, and Leif D. Nelson. 2020. Specification curve analysis. *Nature Human Behaviour* 4 (7 2020), 1208–1214. Issue 11. https://doi.org/10.1038/s41562-020-0912-z

[59] Sara Steegen, Francis Tuerlinckx, Andrew Gelman, and Wolf Vanpaemel. 2016. Increasing Transparency Through a Multiverse Analysis. *Perspectives on Psychological Science* 11 (9 2016), 702–712. Issue 5. https://doi.org/10.1177/1745691616658637

[60] Klaas-Jan Stol and Brian Fitzgerald. 2015. Theory-oriented software engineering. *Science of Computer Programming* 101 (4 2015), 79–98. https://doi.org/10.1016/j.scico.2014.11.010

[61] Margaret-Anne Storey, Rashina Hoda, Alessandra Maciel Paz Milani, and Maria Teresa Baldassarre. 2025. Guiding Principles for Using Mixed Methods Research in Software Engineering. arXiv:2404.06011 [cs.SE] https://arxiv.org/abs/2404.06011

[62] David Trafimow. 2018. An a priori solution to the replication crisis. *Philosophical Psychology* 31 (11 2018), 1188–1214. Issue 8. https://doi.org/10.1080/09515089.2018.1490707

[63] Bianca Trinkenreich, Klaas-Jan Stol, Igor Steinmacher, Marco A. Gerosa, Anita Sarma, Marcelo Lara, Michael Feathers, Nicholas Ross, and Kevin Bishop. 2023. A Model for Understanding and Reducing Developer Burnout. In *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 48–60. https://doi.org/10.1109/ICSE-SEIP58684.2023.00010

[64] Asher Trockman, Shurui Zhou, Christian Kästner, and Bogdan Vasilescu. 2018. Adding Sparkle to Social Coding: An Empirical Study of Repository Badges in the npm Ecosystem. *Proceedings of the 40th International Conference on Software Engineering*, 511–522. https://doi.org/10.1145/3180155.3180209

[65] Roberto Verdecchia, Emelie Engström, Patricia Lago, Per Runeson, and Qunying Song. 2023. Threats to validity in software engineering research: A critical reflection. *Information and Software Technology* 164 (12 2023), 107329. https://doi.org/10.1016/j.infsof.2023.107329

[66] James Walden. 2020. The Impact of a Major Security Event on an Open Source Project. *Proceedings of the 17th International Conference on Mining Software Repositories*, 409–419. https://doi.org/10.1145/3379597.3387465

[67] Mairieli Wessel, Alexander Serebrenik, Igor Wiese, Igor Steinmacher, and Marco A. Gerosa. 2020. Effects of Adopting Code Review Bots on Pull Requests to OSS Projects. *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 1–11. https://doi.org/10.1109/ICSME46990.2020.00011

[68] Mairieli Santos Wessel, Alexander Serebrenik, Igor Wiese, Igor Steinmacher, and Marco Aurélio Gerosa. 2022. Quality gatekeepers: investigating the effects of code review bots on pull request activities. *Empir. Softw. Eng.* 27, 5 (2022), 108. https://doi.org/10.1007/S10664-022-10130-9

[69] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Planning*. Springer Berlin Heidelberg, 89–116. https://doi.org/10.1007/978-3-642-29044-2_8

[70] Marvin Wyrich and Sven Apel. 2024. Evidence Tetris in the Pixelated World of Validity Threats *(WSESE '24)*. Association for Computing Machinery, New York, NY, USA, 13–16. https://doi.org/10.1145/3643664.3648203

[71] Marvin Wyrich, Marvin Muñoz Barón, and Justus Bogner. 2024. Apples, Oranges, and Software Engineering: Study Selection Challenges for Secondary Research on Latent Variables. In *Proceedings - 2024 IEEE/ACM International Workshop on Methodological Issues with Empirical Studies in Software Engineering, WSESE 2024*. Association for Computing Machinery, Inc, 42–47. https://doi.org/10.1145/3643664.3648213

[72] Yangyang Zhao, Alexander Serebrenik, Yuming Zhou, Vladimir Filkov, and Bogdan Vasilescu. 2017. The {Impact} of {Continuous} {Integration} on {Other} {Software} {Development} {Practices}: {A} {Large}-scale {Empirical} {Study}. *Proceedings of the 32Nd {IEEE}/{ACM} {International} {Conference} on {Automated} {Software} {Engineering}*, 60–71. http://dl.acm.org/citation.cfm?id=3155562.3155575

[73] Theo Zimmermann and Annali Casanueva Artis. 2019. Impact of Switching Bug Trackers: A Case Study on a Medium-Sized Open Source Project. *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 13–23. https://doi.org/10.1109/ICSME.2019.00011