Dealing with Time Dependence in Empirical Software Engineering

Mikel Robredo
University of Oulu
Oulu, Finland
mikel.robredomanero@oulu.fi

ACM Reference Format:
Mikel Robredo. 2025. Dealing with Time Dependence in Empirical Software Engineering. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In Empirical Software Engineering (ESE) researchers frequently explore relationships between various variables derived from the historical data of software projects. For instance, scholars have dedicated their efforts to examining correlations among variables such as code smells, design smells, and architectural smells [14, 17], scrutinizing their temporal evolution [12], and assessing the influence of different software attributes [10, 13].

Nevertheless, there is still a tendency in the research community to omit the evidence of temporal interdependence among variables and, therefore, obtain potential vulnerabilities in the results due to the application of statistical techniques which cannot thoroughly analyze temporally dependent data. For instance, introducing a code smell in a software commit heavily relies on the state of the codebase *preceding* said commit. Recent research has already proposed potential solutions to address this issue [1, 8]. Yet, this message remains far from integrated into the research community, primarily due to the absence of well-defined statistical methodologies for handling such temporal dependencies.

2 DEALING WITH TIME DEPENDENCE

From a theoretical point of view, and adapted to the SE field, the concept of time dependence between variables refers to the existing dynamic temporal relationships happening among diverse software metrics or parameters over subsequent time intervals [3]. This specific type of correlation demonstrates that the observed values of these metrics show patterns that evolve during the time through different stages as the object under observation progresses [4]. Therefore, the analysis of time as a crucial factor is fundamental for gaining knowledge about the evolution of time-dependent relationships, identifying new patterns and trends, detecting anomalies, and performing accurate predictions about future development cycles [6]. Time dependence in SE, like in many other fields, can be observed in diverse ways, identifying patterns like release cycles, code commit frequencies, fault induction, or stochastic variations evolved from newly added factors. Current research in potential analytical techniques for time-dependent analysis could discern trends, predict potential issues, and draw solutions to emerging challenges for software engineers [16].

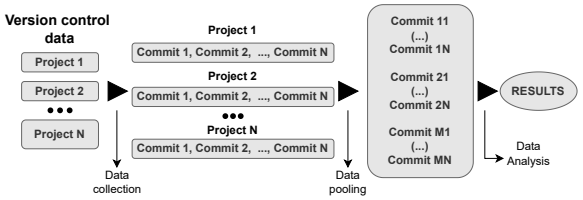

Figure 1: Current data collection and analysis approach in MSR studies (N: N° of projects, M: N° of commits)

However, just as in any other data analysis methodology, time dependence can be analyzed from different perspectives and focused on diverse goals. The choice of methods, models, and data analysis methodology makes a crucial impact in the analysis stage of a research process. Therefore, it is vital that researchers are provided with robust guidelines on how to perform data analysis when dealing with time-dependent data.

The Research questions that we pose at the start of the work are:

- **RQ$_1$.** What are the current data analysis issues in ESE studies?
- **RQ$_2$.** Which data analysis techniques should be applied in the different Software Engineering studies?
- **RQ$_3$.** Which methodology should be followed to identify which analysis technique needs to be used in each study scenario?

3 CURRENT ISSUES

The possibility of obtaining data through the mining of software projects' version control systems has opened a wide source of data for current research. In particular, MSR studies utilize features such as commits, refactorings, and releases located in the source code of projects to study their quality [2, 7, 10, 13]. Frequently, in these and other studies of a similar nature, researchers tend to collect the version control data of the selected projects during the data collection process and subsequently pool the considered metrics all together into the same dataset. Then, the considered data analysis is performed on the dataset to obtain a global result. Figure 1 presents graphically the described procedure.

As briefly mentioned in the introduction, there are three main concerns about the current research approach in MSR:

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

- **Discarding the temporal nature of commits:** Commits are temporally ordered within the version control of projects. Each commit depends on what existed before, and therefore there exists a time dependency between commits. For instance, an existing bug in the current commit may be the result of mistakes in the code developed in previous commits, or colloquially speaking, "A mistake in the code today may lead to a bug tomorrow." This potential relationship vanishes when all commits from different projects are pooled together.
- **Assumption of independent data:** Most of the mentioned studies in the current literature perform the commonly used statistical methods such as Mann-Whitney, Wilcoxon, or ANOVA in their data analyses [22]. Among the multiple assumptions these methods require is the independence of observations inside groups. While pooling the projects' commits together, it is most likely that time-dependent commits from the same projects may end up in the same group. Therefore, as the assumption of independence among groups is not fulfilled, the results from the analysis cannot be entirely correct.
- **Dimensional difference among projects:** To ensure the generalizability of the results, multiple studies consider including projects of different dimensions. However, often all projects are pooled together regardless of the number of commits the different projects have. Hence, when the data pooling is performed, the weight of bigger projects exceeds that of smaller projects, removing representativity in the study from the second group.

These reported issues, and therefore ignoring the existence of time dependence between the studied variables can lead to negative consequences and errors in the subsequent results. Hence, the researcher must carefully choose which statistical methods to use in the data analysis based on the assumptions fulfilled by the distribution of the data. If the researchers fail at this stage, some of the potential negative effects are the following:

- **Considering wrong assumptions:** Most statistical methods currently in use rely on different sets of assumptions that the data points must respect. Suppose time dependence is not considered among the assumptions of the chosen statistical method to analyze time-dependent data. In that case, the obtained results can lead to misleading interpretations and poor model performance.
- **The threat of losing temporal relationships:** Time-dependent data often exhibit patterns between consecutive data points. Losing the temporal relationship among observations could yield wrong results that do not identify valuable insights or anomalies, e.g., concluding that mistakes in the code and bugs are not related.
- **Misleading results:** In an era where good results can mask possible misinterpretations of the data, the mistake of not considering time-dependent methods for time-dependent data can cause the resulting analysis to hide the true nature of the data. That is, results relying on wrong methods can yield misleading interpretations, e.g. concluding that more mistakes in the code can lead to fewer bugs.

In essence, the consequences of bad code administration are not obvious at the initial stage after the mistake has been made. Time-dependent methods are a powerful approach to analyze the spread of mistakes created over time. Similarly, assessing bad practices in the code based on the prevention of future bugs may not have instant improvements, especially for big projects where multiple professionals are involved. Still, this practice will make a positive impact in the long run. Therefore, addressing time dependence becomes a healthy consideration that, if not taken, can lead the research to face the effects described above.

4 EXAMPLES IN SOFTWARE ENGINEERING

We now present two examples that depict the described current issues in the SE research field. We first introduce a theoretical example with the aim of better conceptualizing the possible consequences of ignoring time dependence when analyzing time-dependent data. In the second example, we describe a real application of the current data analysis approach in MSR studies by observing the work [10].

4.1 Theoretical example

Let us consider a theoretical example showing the importance of considering the assumption of time dependence when analyzing time-dependent data. In this example, there is a *dependent* variable (red) which at time $t + 1$ is highly correlated to the *independent* variable (blue) at time t. Figure 2 depicts the first 50 observed points of the temporal relationship within the data.

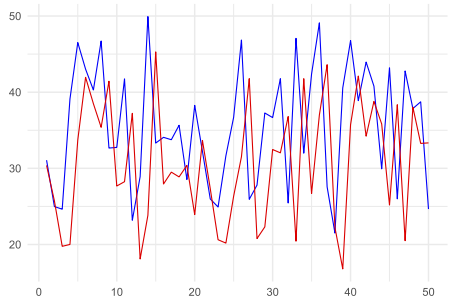

Figure 2: Temporal evolution of the example variables.

If we were not conscious of the temporal relationship of the data, the omission of the time dependence between the described variables could lead us to consider erroneous assumptions. An exploratory correlation plot of such a scenario could be depicted in Figure 3. At first glance, this perspective of the example variables could lead the researcher to discard the existing time dependence. If the assumption of time dependence is removed, we need to prioritize this stage in the analysis of the two variables. An erroneous conclusion about the correlation between the variables would lead to a misinterpretation, for instance, a nonexistent correlation between the variables.

If the researcher considers the assumption of an existing time dependence among the dependent variable and the independent variable, the exploratory analysis of the variables could explain that the dependent variable is the same as the independent variable in the previous point, given a sort of small variables. Once this assumption is considered, the correlation plot of the variables displayed in Figure 2 would depict an existing temporal relationship similar to the one displayed in Figure 4.

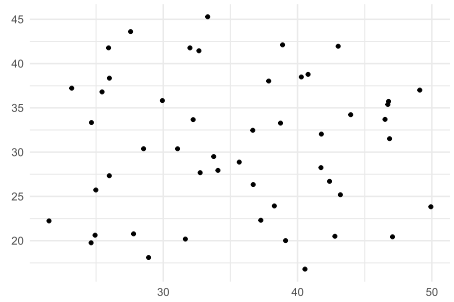

Figure 3: Correlation plot of the example variables considering the same time point.

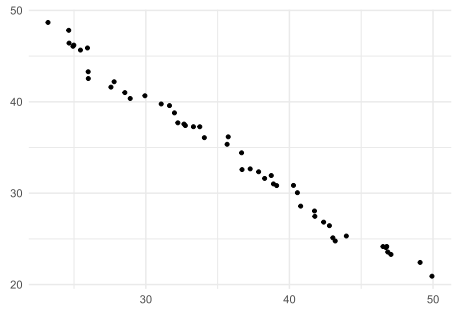

Figure 4: Correlation plot of the example variables considering a lag of one-time point between them.

4.2 Real example

Let us now consider a real case application of the current data collection and data analysis approach in MSR studies. The chosen example is the study performed by [10], which investigates the diffuseness of SonarQube issues on 33 Java projects from the Apache Software Foundation, and assesses their impact on code changes and fault-proneness, considering also their different types and severities. To accomplish that, they investigated the differences between classes that are not affected by Sonar issues (**clean**) and classes affected by at least one Sonar issue (**dirty**). Thus, their study compares the change-proneness and fault-proneness of the classes within these two groups.

The study collected commits from each of the 33 mentioned projects once every 180 days (the average time between releases in the analyzed projects). Subsequently, they calculated the change-proneness and fault-proneness for each of the existing classes from the current commit to the incoming commit within all projects. To compare the distribution of the two groups of classes, they used two widely used non-parametric statistical tests. First, they used the *Mann-Whitney* test to compare the statistical significance of each of the group distributions. This statistic is used to test whether the distributions of two groups are statistically different, i.e. whether the distributions of observations for both groups of variable of interest have the same shape or a different shape. Furthermore, the method chooses random samples from both groups to perform the comparison, given the assumption of independence among

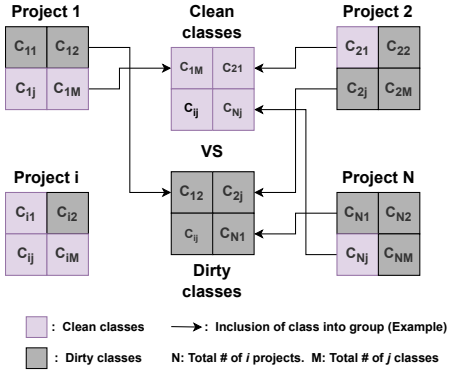

Figure 5: Graphical representation of the class groups built by pooling all together.

observations in each group. Secondly, they used the *Cliff's Delta* effect size test to investigate the magnitude of the size difference between the two groups. In either case, these tests were used with the preliminary assumption of independence among the group observations. The study described concluded that classes affected by Sonar issues may be more change—and fault-prone than non-affected classes. However, the differences between the clean and affected classes proved to be small.

Taking a closer look into the described data analysis, the *Mann-Whitney* statistic assumes independence among the observations in each group, which in this case are classes in the dirty and clean groups. For instance, assume that a given project has multiple clean and dirty Java classes and consider the scenario in which classes are connected within the same project so that changes in one of the classes may have an impact on the other classes in the short or long run. If two or more connected classes from the same project were located inside the same group, the assumption of independence between observations of the same group would be erroneous, therefore the results from the performed analysis would lead to misleading interpretations.

Moreover, within the 33 Java projects considered, large projects overwhelmed small ones, which was reflected in the number of commits collected for each of the projects. This fact can create a dimensional gap in the impact of the considered projects in the study and further overlap the temporal patterns existing between the defined 180-day period among projects. This evidence depicts an informational gap that might be missing in the reported results and that it could develop into a different outcome if a different approach had been considered.

5 THE PLAN

This project will be implemented in multiple stages:

Data analysis issues identification. We will conduct a literature review (e.g. Systematic Literature Review [9] or Systematic Mapping Study [15]) of the existing data analysis approaches and methods adopted in Software Engineering to analyze the collected data. This step allows us to have a complete and exhaustive knowledge of the current situation to identify some biases and issues. Then, we will investigate what type of data has been used in the

performed analyses, which techniques have been implemented, and whether the required assumptions have been fulfilled or ignored. Additionally, we aim to identify existing studies that have tried to respect the required assumptions of the adopted techniques to answer the same research questions. We hypothesize that, despite studies using additional unconventional techniques already existing in the literature, a large portion of the ESE relies on data analysis techniques that do not fulfil all the required statistical assumptions.

Conceptualizing potential approaches. As we have seen, limiting to static analysis of variables neglecting the influence of time disregards some important intra- and inter-variable dependencies, which only become obvious over time. Simply put, today's bad coding practices translate into tomorrow's bugs. We thus believe that *time*—or at least synchronization points—should be considered for the analysis of dependent variables. Thus, we propose a set of existing methods potentially eligible for analyzing the evolution of different numbers of variables over time:

- *Time Series Analysis:* These models can exclusively identify the temporal dependencies and trends that precede a bug and consequently forecast its introduction in the code as well as provide forecasting intervals considering the existing observations and their mentioned temporal dependencies [4].
- *Bayesian statistics:* These approaches are based on Bayes' theorem, where available knowledge about parameters in a statistical model is updated with the information in observed data to determine the posterior distribution [20].
- *Transformers:* These models have shown great modeling ability for long-range dependencies and interactions in sequential data and time series modeling [21, 23]. Many variants of Transformer have been proposed to address special challenges in time series modelling and have been successfully applied to various time series tasks, such as forecasting [23] and anomaly detection [19].
- *Neurosymbolic:* These models refer to AI systems that seek to integrate neural network-based methods with symbolic knowledge-based approaches [18].
- *Temporal Logic:* These models differ by the ontological assumptions made about the nature of time in the associated models, by the logical languages involving various operators for composing temporalized expressions, and by the formal logical semantics adopted for capturing the precise intended meaning of these temporal operators [5, 11].

Approach comparison and Internal Validation. We will emphasize the needed format the data will need to have to perform the selected potential data analysis technique accordingly, as well as the respective interpretation of the obtained results. The goal is to compare the defined data analysis approaches and the currently used approach to specify the differences between them. Then, we aim to evaluate the proposed methodology by selecting a set of published ESE studies. We will replicate these studies by applying our methodology and comparing the results. Moreover, the outcome of the analysis will be discussed with experts in ESE to evaluate its validity.

Guidelines definition. Based on the results obtained in the previous stages, we will define a set of user guidelines for identifying the optimal prelude scenario for each of the provided data analysis techniques and applying the latter one accordingly. These guidelines will aim to help the research community implement robust data analysis techniques while yielding the existing statistical assumptions for each technique. The guidelines will be based on existing data analysis used in external research fields as well as those already adopted in the SE field.

REFERENCES

[1] Saïd Assar, Markus Borg, and Dietmar Pfahl. 2016. Using text clustering to predict defect resolution time: a conceptual replication and an evaluation of prediction accuracy. *Empirical Software Engineering* 21 (2016), 1437–1475.

[2] Maria Teresa Baldassarre, Valentina Lenarduzzi, Simone Romano, and Nyyti Saarimäki. 2020. On the diffuseness of technical debt items and accuracy of remediation time when using SonarQube. *Information and Software Technology* 128 (2020), 106377.

[3] Peter J Brockwell and Richard A Davis. 1991. *Time series: theory and methods*. Springer science & business media.

[4] Peter J Brockwell and Richard A Davis. 2002. *Introduction to time series and forecasting*. Springer.

[5] Valentin Goranko. 2023. *Temporal Logics*. Cambridge University Press.

[6] Rob J Hyndman and George Athanasopoulos. 2018. *Forecasting: principles and practice*. OTexts.

[7] Foutse Khomh, Massimiliano Di Penta, and Yann-Gael Gueheneuc. 2009. An exploratory study of the impact of code smells on software change-proneness. In *2009 16th Working Conference on Reverse Engineering*. IEEE, 75–84.

[8] Riivo Kikas, Marlon Dumas, and Dietmar Pfahl. 2016. Using dynamic and contextual features to predict issue lifetime in github projects. In *Proceedings of the 13th International Conference on Mining Software Repositories*. 291–302.

[9] Barbara Kitchenham, O. Pearl Brereton, David Budgen, Mark Turner, John Bailey, and Stephen Linkman. 2009. Systematic literature reviews in software engineering – A systematic literature review. *Information and Software Technology* 51, 1 (2009), 7–15.

[10] Valentina Lenarduzzi, Nyyti Saarimäki, and Davide Taibi. 2020. Some sonarqube issues have a significant but small effect on faults and changes. a large-scale empirical study. *Journal of Systems and Software* (2020), 110750.

[11] Fabrizio Maria Maggi, Marco Montali, and Rafael Peñaloza. 2020. Temporal Logics Over Finite Traces with Uncertainty. In *Conference on Artificial Intelligence, AAAI 2020*. 10218–10225.

[12] S. Olbrich, D. S. Cruzes, V. Basili, and N. Zazworka. 2009. The evolution and impact of code smells: A case study of two open source systems. In *International Symposium on Empirical Software Engineering and Measurement*. 390–400.

[13] Fabio Palomba, Gabriele Bavota, Massimiliano Di Penta, Fausto Fasano, Rocco Oliveto, and Andrea De Lucia. 2018. On the diffuseness and the impact on maintainability of code smells: a large scale empirical investigation. *Empirical Software Engineering* 23, 3 (01 Jun 2018), 1188–1221.

[14] Fabio Palomba, Damian Andrew Tamburri, Francesca Arcelli Fontana, Rocco Oliveto, Andy Zaidman, and Alexander Serebrenik. 2018. Beyond Technical Aspects: How Do Community Smells Influence the Intensity of Code Smells? *IEEE Transactions on Software Engineering* (2018), 1–1.

[15] Kai Petersen, Robert Feldt, Shahid Mujtaba, and Michael Mattsson. 2008. Systematic Mapping Studies in Software Engineering. In *International Conference on Evaluation and Assessment in Software Engineering*. 68–77.

[16] Nyyti Saarimäki, Sergio Moreschini, Francesco Lomio, Rafael Penaloza, and Valentina Lenarduzzi. 2022. Towards a Robust Approach to Analyze Time-Dependent Data in Software Engineering. In *International Conference on Software Analysis, Evolution and Reengineering (SANER)*. 36–40.

[17] Tushar Sharma, Paramvir Singh, and Diomidis Spinellis. 2020. An Empirical Investigation on the Relationship between Design and Architecture Smells. *Empirical Software Engineering* 25 (2020).

[18] Amit Sheth, Kaushik Roy, and Manas Gaur. 2023. Neurosymbolic ai-why, what, and how. *arXiv preprint arXiv:2305.00813* (2023).

[19] Shreshth Tuli, Giuliano Casale, and Nicholas R. Jennings. 2022. TranAD: deep transformer networks for anomaly detection in multivariate time series data. *Proc. VLDB Endow.* 15, 6 (feb 2022), 1201–1214.

[20] Rens van de Schoot, Sarah Depaoli, Andrew Gelman, Ruth King, Bianca Kramer, Kaspar Märtens, Mahlet G. Tadesse, Marina Vannucci, Joukje Willemsen, and Christopher Yau. 2021. Bayesian statistics and modelling. *Nature Reviews Methods Primers* 1 (14 Jan. 2021). https://doi.org/10.1038/s43586-020-00003-0

[21] Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, and Liang Sun. 2023. Transformers in Time Series: A Survey. arXiv:cs.LG/2202.07125

[22] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Bjorg Regnell, and Anders Wesslén. 2000. *Experimentation in Software Engineering: An Introduction*. Springer Publishing Company, Incorporated.

[23] Tian Zhou and et al. 2022. FEDformer: Frequency Enhanced Decomposed Transformer for Long-term Series Forecasting. In *International Conference on Machine Learning (Machine Learning Research)*, Vol. 162. 27268–27286.