Different Debt: An Addition to the Technical Debt Dataset and a Demonstration Using Developer Personality

Lorenz Graf-Vlachy
lorenz.graf-vlachy@tu-dortmund.de
TU Dortmund University
Dortmund, Germany
University of Stuttgart, Institute of Software Engineering
Stuttgart, Germany

Stefan Wagner
stefan.wagner@iste.uni-stuttgart.de
University of Stuttgart, Institute of Software Engineering
Stuttgart, Germany

ABSTRACT

Background: The "Technical Debt Dataset" (TDD) is a comprehensive dataset on technical debt (TD) in the main branches of more than 30 Java projects. However, some TD items produced by SonarQube are not included for many commits, for instance because the commits failed to compile. This has limited previous studies using the dataset. *Aims and Method:* In this paper, we provide an addition to the dataset that includes an analysis of 278,320 commits of all branches in a superset of 37 projects using Teamscale. We then demonstrate the utility of the dataset by exploring the relationship between developer personality by replicating a prior study. *Results:* The new dataset allows us to use a larger sample than prior work could, and we analyze the personality of 111 developers and 5,497 of their commits. The relationships we find between developer personality and the introduction and removal of TD differ from those found in prior work. *Conclusions:* We offer a dataset that may enable future studies into the topic of TD and we provide additional insights on how developer personality relates to TD.

CCS CONCEPTS

• **Software and its engineering** → **Maintaining software**; **Risk management**; • **Human-centered computing** → **Open source software**; • **Social and professional topics** → **Systems development**; **Software maintenance**; **User characteristics**; • **General and reference** → **Empirical studies**;

KEYWORDS

technical debt, developer personality, Teamscale

1 INTRODUCTION

1.1 Technical Debt

The metaphorical notion of technical debt (TD) was introduced by Ward Cunningham more than 30 years ago [9]. Despite slight variations in the exact definition of the term, there is a reasonable consensus on TD being a "collection of design or implementation constructs that are expedient in the short term, but set up a technical context that can make future changes more costly or impossible" [4, p. 112]. Over time, despite its inherent limitations [27], the TD metaphor has gained acceptance and found widespread use in both academic and practitioner circles [25].

Researchers have identified different types of TD and made efforts to categorize them. Tom et al., for example, identify eight different dimensions of TD, comprising code, design and architecture, operational processes, among others [29]. Relatedly, Alves et al. and Rios et al. each distinguish between 15 different types of TD, including, for example, design debt, code debt, test debt, and documentation debt [2, 26].

By definition, TD comes with advantages and disadvantages. Reported advantages lie particularly in increased short-term developer velocity [24]. Reported disadvantages [6] are decreased long-term velocity [24], reduced developer morale and motivation [5, 10, 25], as well as lower code quality and increased uncertainty and risk [29].

Beyond conceptual considerations [27], qualitative studies [10], and surveys [6], researchers have increasingly become interested in performing large-scale quantitative studies on TD using data mined from software repositories [8, 18–20].

1.2 The Technical Debt Dataset and its Limitations

To enable such studies of TD, Lenarduzzi et al. developed the "Technical Debt Dataset" (TDD) [21]. It is a dataset of TD (primarily code debt) in various Apache Software Foundation (ASF) projects written in Java. It has been used in a variety of studies on TD [8, 18–20]. In its most recent version 2.0, it contains a comprehensive analysis of the main branches of 31 projects. Aside from data obtained from PyDriller, Refactoring Miner, Jira, and Ptidej, the dataset most notably includes TD information generated using SonarQube.

Importantly, SonarQube requires the build of a commit to complete before it can provide TD information. For various reasons (e.g., deficient code or missing dependencies), however, builds may fail. Consequently, the TDD does not contain complete information for such commits. According to our own analyses, the SonarQube

analyses were incomplete for more than 60% of commits in the covered projects.

Recent research has found that this is particularly problematic in cases where differences in TD between commits are of interest, because then missing information in either the focal commit or its parent commit leads to missing data in the ultimate analysis. This problem has been reported to reduce the size of samples dramatically, potentially impacting the validity of analyses performed on them. For instance, in the recent study on developer personality and TD by Graf-Vlachy and Wagner, the authors collected personality data for 121 developers who made commits that are within the scope of the TDD, but they could only use data from 19 developers due to missing TD information in the TDD [13].

Further, it is well-known that TD detection tools may come to different assessments of TD [17]. As the TDD only contains TD items from SonarQube, it is thus naturally limited in this way, too.

2 AN ADDITION TO THE TECHNICAL DEBT DATASET

To address these limitations, we develop an addition to the TDD that includes information on TD for essentially all commits. The following describes the used tools, the process of constructing the dataset, and the resulting dataset itself.

2.1 Teamscale

We develop our addition to the TDD using Teamscale in version 9.1.2. Teamscale is a tool for analyzing code quality and tests [14, 16]. For the Java language, such analyses can be performed directly on the source code without the need for compiled bytecode. Teamscale can be run locally and allows the user to access its analyses through a web interface or a REST API. It has been previously used in research on empirical software engineering [23]. Teamscale is commercial in nature but CQSE, the company developing it, offers free licenses for open-source projects and academic users.

Notably, the philosophy of the company behind Teamscale discourages the use of single-indicator metrics to assess the maintainability of, and thus the TD in, software projects [22]. Consequently, Teamscale does not provide a singular metric of TD comparable to SonarQube's "technical debt" metric (variable sqale_index) that has been used in prior work [13]. Instead, Teamscale provides various detailed measures related to TD. These include, for instance, excessive nesting depth, cyclomatic complexity, malformed comments, name shadowing, hard-coded credentials, or unused code.

2.2 Construction of the Dataset

We constructed the dataset in the following way. First, we identified the projects included in the TDD. Although version 2 of the TDD only includes 31 projects, we opted to additionally include all projects listed in the original TDD paper (*Accumulo, Ambari, Atlas, Aurora, Beam, MINA SSHD*) [21]. Similarly, we decided not to restrict our analyses to the projects' main branches as TDD version 2 did but to analyze all branches. We then implemented a Python tool that performs several steps. First, it clones the repositories locally. It then imports these local copies into Teamscale, which is also running locally. Once a project is successfully imported,

Teamscale begins to perform an analysis of all commits in the background. To ensure complete data availability, our tool waits until data processing within Teamscale is completed. The tool then uses Teamscale's REST API to request all available relevant datapoints for each commit in each branch of each project. Finally, the tool writes these datapoints out to the local disk.

We only analyzed Java code. We used Teamscale's default settings except for two cases. First, we ensured that Teamscale would not only analyze the main branch but all branches by enabling "Branch support". Second, we switched on the "Preserve empty commits" commits option to ensure that Teamscale would retain all commits.

Data analysis took multiple weeks on a dedicated Windows virtual server with eight cores and 192 GB RAM. The analysis script, the Teamscale configuration file, and the resulting dataset are available at https://doi.org/10.6084/m9.figshare.24550840.

The dataset and all code are licensed under Apache License 2.0.

2.3 Description of the Dataset

There are two key elements which constitute our new dataset. For one, there is a folder for each project with JSON files for each commit in the project that includes all information Teamscale has about the respective commit. This is provided only for advanced use cases. The filenames include each commit's hash for easy identification. In the further analyses of this paper, these files will not be used.

For another, there is a set of CSV files that comprise selected TD information on each commit in the projects. Specifically, there are three types of CSV files. First, there is a "report" file. This file contains various aggregated pieces of information that Teamscale provides for each commit. This includes, for instance, the number of parent commits, the number of files in the commit, the lines of code in the commit, the number of findings added and removed in the focal commit, and the number of findings above a certain severity level so Teamscale flags them as "yellow" or "red", respectively. Several of these data points are provided as an absolute value for the focal commit and as a difference to the parent commit. (Note that Teamscale provides difference data even when the parent is on a different branch. Although the behavior in the case of merge commits is not specifically documented, our investigations lead us to believe that Teamscale compares a focal commit's data to its oldest parent commit.) Table 1 describes the "report" file further.

Second, there is a "findings" file. It contains the non-aggregated information on all Teamscale findings per commit (as identified by commit hash). This includes 57 different types of findings, categorized into architecture, comprehensibility, correctness, documentation, efficiency, error handling, redundancy, security, structure, testing, and others. The data is separated out by whether the finding was added or removed in the focal commit, or found in changed code, as well as by finding severity (either "yellow" or "red").

Third, there is a file on "findings_messages", which provides the detailed Teamscale messages for all findings per commit (as identified by commit hash).

All types of files include the project name, the branch name, and the commit hash as identifiers that allow the data to be linked to each other as well as to the TDD. (Note that some projects—e.g., *Accumulo* and *Batik*—have renamed their main branches from "master" to "main" between the release of the TDD and our analyses.)

The "report" output also includes the first commit of the repository. For these commits, Teamscale lists the Author Name as "Teamscale import" but reports no further data through the API although the web interface shows analysis reports. However, due to the particular characteristics of these initial orphan commits, it is likely best to discard these commits in analyses anyway. The "findings" output does not include any information on these commits.

Each CSV file exists once for each specific project and once in a combined form that covers all projects. Further information and statistics on the dataset are available in a separate document in the data package at https://doi.org/10.6084/m9.figshare.24550840.

Note that our dataset is more extensive than the TDD in at least three dimensions regarding TD. It covers more projects, more branches, and it spans a timeframe until the end of October 2023.

Insofar as the two datasets overlap, they can be readily linked using the commit hashes.

Table 1: Contents of "report" CSV file

Category	Variable	Notes
Identifiers	Project Branch Commit Hash	Project name Branch name Commit hash
Commit data	Num Parent Commits Timestamp Author Name Author Email	Number of parent commits (0 for orphans, 1 for regular commits, >1 for merge commits) Teamscale timestamp of commit Name of author Email address of author
Analysis results	Files_<val> Lines of Code_<val> Source Lines of Code_<val> Longest Method Length_<val> Maximum Nesting Depth_<val> Change cnt_<val> Close Coverage_<val> Line Coverage_<val> Num Findings Red_<val> Num Findings Yellow_<val> Maximum Cyclomatic Complexity_<val> File Size assm_<color>_<val> Method Length assm_<color>_<val> Nesting Depth assm_<color>_<val> Comment Completeness assm_<color>_<val> Cyclomatic Complexity assm_<color>_<val> Added Findings cnt Removed Findings cnt Findings in Changed Code cnt	• <val> can be "abs" or "diff" • <color> can be "g" (green), "y" (yellow), or "r" (red) • "cnt" = "count" • "assm" = "assessment"

3 DEVELOPER PERSONALITY AND TECHNICAL DEBT REDUX

We demonstrate the utility of our dataset in an exploration of the relationship between developer personality and TD. To do so, we replicate an analysis of developer personality and TD that was hampered by the limitations of the TDD [13].

3.1 Description of Original Study

In their recent study, Graf-Vlachy and Wagner used the TDD to explore developer personality [13] in the context of TD. Specifically, they studied the relationship between three broad personality constructs and the introduction and removal of TD. The three personality constructs are the five traits of the Five Factor Model (extraversion, agreeableness, conscientiousness, emotional stability, and openness to experience), the personality characteristic of regulatory focus (comprising promotion focus and prevention focus), and narcissism. They propose that incurring TD is a form of

risk-taking (also see [12]), and they argue that different personality characteristics relate, through their relationship with risk-taking, to TD. They find that conscientiousness, emotional stability, openness to experience, and prevention focus are negatively linked to TD. They find no significant results for extraversion, agreeableness, promotion focus, or narcissism.

To gather developer personality data, Graf-Vlachy and Wagner surveyed all 1,555 developers having made any commits that are part of the TDD version 2. Importantly, they measured all variables using validated scales [30]. The five-factor model personality traits were captured using the Ten-Item Personality Measure (TIPI) [11]. Regulatory focus was measured using six items (three for promotion focus and three for prevention focus) from the Regulatory Focus Composite Scale (RF-COMP) [15]. Narcissism was captured using the short version of the Narcissistic Personality Inventory (NPI-16) [3]. Reliability metrics like Cronbach's α were sufficiently high.

Graf-Vlachy and Wagner also identified developers' age at the time of each commit by capturing developers' age in years and then subtracting the difference between 2022 and the year in which the focal commit was made from the provided age.

After accounting for missing data and implausible values, they obtained complete data on the characteristics of 121 developers.

3.2 Demonstration Using Our Dataset

In the following, we describe our analysis using our new dataset. Importantly, we do not theorize ex ante about any individual relationships between personality and TD. Instead, we simply explore the data to see if we find patterns similar to the ones reported by Graf-Vlachy and Wagner [13].

Note that, in contrast to their analysis (which only used the net amount of TD created or removed by a commit), we study the number of TD items (Teamscale "findings") that were added in a commit and those that were removed in a commit separately. For comparability, we additionally use the difference between the two values (i.e., the net change) as a third dependent variable.

3.2.1 Sample. Our sample is the result of a merge between the TDD and our dataset by commit hash. It is thus restricted to commits made to the main branches of the projects, which also alleviates concerns over the potentially experimental nature of non-main branches. We only consider normal commits and drop merge and orphan commits [1] based on information from the TDD. Merge commits do not allow a sensible calculation of changes in TD (due to multiple parent commits) and orphan commits likely have particular characteristics that may distort the analyses. We further obtained the developer personality data collected by Graf-Vlachy and Wagner and linked it to our newly developed dataset. Overall, our sample comprises 5,497 commits from 111 developers. This is substantially larger than the sample of Graf-Vlachy and Wagner, who analyzed 2,145 commits from only 19 developers [13]. Notably, we still cannot analyze all commits because even Teamscale does not provide data for all. This is the case, for instance, for cross-repository commits.

3.2.2 Analysis Strategy. We follow the method used by Graf-Vlachy and Wagner [13]. This means that we used panel regressions because each developer is observed repeatedly, once for each commit they made. We clustered standard errors at the developer to account

for the fact that such multiple commits from the same developer are not statistically independent. In our model, we controlled for developer age at time of commit (from [13]) and lines of code (LOC) added and LOC removed (from the TDD as Teamscale does not provide these metrics). To account for unobserved time-invariant aspects of each project (for instance, specific coding conventions), we included dummy variables (fixed effects) for each project.

Notably, for the analyses of the number of added and removed findings, a Poisson estimator would be econometrically appropriate because these dependent variables are counts [31]. However, because this estimator did not converge when analyzing our dataset, we report the results of a random effects panel model instead. Such a model is the appropriate choice for our third dependent variable, the net change in findings. We will focus our interpretation of the results on this dependent variable, also because it allows for a direct comparison with the original study [13].

All analyses were performed in Stata 17.0. All analysis scripts are available at https://doi.org/10.6084/m9.figshare.24550840.

3.2.3 Findings. As is evident in Table 2, we find that LOC added and LOC removed are related to the number of added and removed findings in the way one would expect. We further find a positive effect of extraversion on added findings and net change, negative effects of promotion focus and narcissism on removed findings, and a negative effect of age at commit on net change. Surprisingly, only the finding on age at commit is in line with the prior research from Graf-Vlachy and Wagner [13]. All findings regarding personality differ. Specifically, we do not reproduce any of Graf-Vlachy and Wagner's significant findings, and all our significant findings were not present in their work [13].

4 DISCUSSION

4.1 Threats to validity

4.1.1 Construct validity. Our measures of TD relies on automated analyses that may not produce perfectly accurate results. Teamscale can be configured extensively, but we use the default settings since we do not have grounds to make a different choice. In particular, to remain consistent across projects, we do not make use of Teamscale's feature to allow for manually identified "tolerated" or "false positive" findings. Different configurations might lead to different results. We also use a simple count of findings as our dependent variables, implicitly assuming that every individual finding represents the same amount of TD. Future research might wish to weigh different types of findings differently. Further, Teamscale largely captures only code debt, but not other types of TD [2, 26, 29].

The used personality data may not be perfectly reliable since it is based on self-reports using short scales [28]. Finally, developers' personality data was collected after they made the analyzed commits. This time gap might potentially affect the accuracy of the personality data in case personality would change over time [7].

4.1.2 Internal validity. Despite following prior work in our selection of control variables, our regressions might suffer from omitted confounding variables, thus limiting the internal validity of our study. Since we use control variables from the TDD, we can also only analyze commits that are from the main branches of the projects. Developers' characteristics may also be related to whether their

Table 2: Results of panel regression analyses

	Added findings	Removed findings	Net change
Extraversion	5.85*	2.97	2.88*
	(2.27)	(1.85)	(1.42)
Agreeableness	-4.23	-2.37	-1.86
	(3.34)	(2.12)	(3.62)
Conscientiousness	-1.82	-0.24	-1.58
	(3.43)	(3.40)	(2.54)
Emotional stability	-0.62	-2.48	1.87
	(2.59)	(1.97)	(2.51)
Openness to experience	-3.32	-2.02	-1.30
	(3.30)	(3.14)	(2.48)
Promotion focus	0.63	-1.91*	2.54
	(1.56)	(0.87)	(1.73)
Prevention focus	-0.46	-0.43	-0.03
	(0.67)	(0.56)	(0.56)
Narcissism	-2.12	-2.28*	0.15
	(1.32)	(0.94)	(1.42)
Age at commit	-0.27	0.63	-0.89*
	(0.50)	(0.34)	(0.41)
LOC added	0.05**	-0.02***	0.07***
	(0.02)	(0.00)	(0.02)
LOC removed	-0.02*	0.05**	-0.07***
	(0.01)	(0.02)	(0.02)
Constant	48.50	47.65	0.85
	(52.25)	(46.49)	(33.72)
Project fixed effects	Yes	Yes	Yes
Observations	5,497	5,497	5,497
Clusters	111	111	111

Dependent variable indicated in top row.
Table reports coefficients, clustered standard errors in parentheses.
* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

code is incorporated into the main branch in the first place, which might affect our results.

4.1.3 External validity. As a matter of course, our study is restricted to developers of large ASF projects. This limits the generalizability of our results to other contexts, such as smaller or closed-source projects. Further, although our analyzed sample is much larger than that of prior work [13], the overall response rate of developers in the survey capturing personality information is still low, potentially creating sample selection issues.

4.1.4 Reliability. Reliability is likely of limited concern. All used personality scales are well-established in psychology. We provide the script to re-run the Teamscale analyses as well as the dataset. Unfortunately, we cannot share the dataset that includes personality data for obvious privacy reasons.

4.2 Implications and Conclusion

First and foremost, our research provides a fine-grained dataset for future studies of TD. Since we also provide the scripts to generate the dataset, future researchers can recreate it with other Teamscale settings however they see fit. In particular, as our dataset fully integrates with the TDD (by linking via commit hash), we enable extensions of prior studies conducted with it.

In terms of practical implications, the findings from our demonstration using the dataset caution practitioners to not overweight results from any single study, such as the original study using the TDD. In fact, we show how an enlarged sample and different measures of TD may yield very different results. In sum, we hope that our empirical findings and dataset spur further research into the link between developer characteristics and TD.

ACKNOWLEDGMENTS

REFERENCES

[1] Reem Alfayez, Pooyan Behnamghader, Kamonphop Srisopha, and Barry Boehm. 2018. An exploratory study on the influence of developers in technical debt. In *Proceedings of the 2018 International Conference on Technical Debt (ACM Conferences)*, Robert L. Nord (Ed.). ACM, New York, NY, 1–10. https://doi.org/10.1145/3194164.3194165

[2] Nicolli S.R. Alves, Thiago S. Mendes, Manoel G. de Mendonça, Rodrigo O. Spínola, Forrest Shull, and Carolyn Seaman. 2016. Identification and management of technical debt: A systematic mapping study. *Information and Software Technology* 70 (2016), 100–121. https://doi.org/10.1016/j.infsof.2015.10.008

[3] Daniel R. Ames, Paul Rose, and Cameron P. Anderson. 2006. The NPI-16 as a short measure of narcissism. *Journal of Research in Personality* 40, 4 (2006), 440–450. https://doi.org/10.1016/j.jrp.2005.03.002

[4] Paris Avgeriou, Philippe Kruchten, Ipek Ozkaya, and Carolyn Seaman. 2016. Managing Technical Debt in Software Engineering (Dagstuhl Seminar 16162). *Dagstuhl Reports* 6, 4 (2016), 110–138.

[5] Terese Besker, Hadi Ghanbari, Antonio Martini, and Jan Bosch. 2020. The influence of Technical Debt on software developer morale. *Journal of Systems and Software* 167 (2020), 110586. https://doi.org/10.1016/j.jss.2020.110586

[6] Terese Besker, Antonio Martini, and Jan Bosch. 2017. The Pricey Bill of Technical Debt: When and by Whom will it be Paid?. In *ICSME 2017*, IEEE International Conference on Software Maintenance and Evolution (Ed.). IEEE, Piscataway, NJ, 13–23. https://doi.org/10.1109/ICSME.2017.42

[7] Fabio Calefato, Filippo Lanubile, and Bogdan Vasilescu. 2019. A large-scale, in-depth analysis of developers' personalities in the Apache ecosystem. *Information and Software Technology* 114 (2019), 1–20. https://doi.org/10.1016/j.infsof.2019.05.012

[8] Zadia Codabux and Christopher Dutchyn. 2020. Profiling Developers Through the Lens of Technical Debt. In *Proceedings of the 14th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. ACM, New York, NY, USA, 1–6. https://doi.org/10.1145/3382494.3422172

[9] Ward Cunningham. 1992. The WyCash Portfolio Mangement System. In *Addendum to the Proceedings of OOPSLA 1992*. 29–30.

[10] Hadi Ghanbari, Terese Besker, Antonio Martini, and Jan Bosch. 2017. Looking for Peace of Mind? Manage Your (Technical) Debt: An Exploratory Field Study. In *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE, 384–393. https://doi.org/10.1109/ESEM.2017.53

[11] Samuel D. Gosling, Peter J. Rentfrow, and William B. Swann. 2003. A very brief measure of the Big-Five personality domains. *Journal of Research in Personality* 37, 6 (2003), 504–528. https://doi.org/10.1016/S0092-6566(03)00046-1

[12] Lorenz Graf-Vlachy. 2023. The Risk-Taking Software Engineer: A Framed Portrait. In *2023 IEEE/ACM 45th International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*. IEEE, 25–30. https://doi.org/10.1109/ICSE-NIER58687.2023.00011

[13] Lorenz Graf-Vlachy and Stefan Wagner. 2023. The Type to Take Out a Loan? A Study of Developer Personality and Technical Debt. In *2023 ACM/IEEE International Conference on Technical Debt (TechDebt)*. IEEE, 27–36. https://doi.org/10.1109/TechDebt59074.2023.00010

[14] Roman Haas, Rainer Niedermayr, and Elmar Juergens. 2019. Teamscale: Tackle Technical Debt and Control the Quality of Your Software. In *2019 IEEE/ACM International Conference on Technical Debt (TechDebt)*. 55–56. https://doi.org/10.1109/TechDebt.2019.00016

[15] Kelly L. Haws, Utpal M. Dholakia, and William O. Bearden. 2010. An Assessment of Chronic Regulatory Focus Measures. *Journal of Marketing Research* 47, 5 (2010), 967–982. https://doi.org/10.1509/jmkr.47.5.967

[16] Lars Heinemann, Benjamin Hummel, and Daniela Steidl. 2014. Teamscale: software quality control in real-time. In *Companion Proceedings of the 36th International Conference on Software Engineering*, Pankaj Jalote, Lionel Briand, and André van der Hoek (Eds.). ACM, New York, NY, USA, 592–595. https://doi.org/10.1145/2591062.2591068

[17] Jason Lefever, Yuanfang Cai, Humberto Cervantes, Rick Kazman, and Hongzhou Fang. 2021. On the Lack of Consensus Among Technical Debt Detection Tools. In *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 121–130. https://doi.org/10.1109/ICSE-SEIP52600.2021.00021

[18] Valentina Lenarduzzi, Francesco Lomio, Heikki Huttunen, and Davide Taibi. 2020. Are SonarQube Rules Inducing Bugs?. In *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 501–511. https://doi.org/10.1109/SANER48275.2020.9054821

[19] Valentina Lenarduzzi, Antonio Martini, Davide Taibi, and Damian Andrew Tamburri. 2019. Towards surgically-precise technical debt estimation: early results and research roadmap. In *Proceedings of the 3rd ACM SIGSOFT International Workshop on Machine Learning Techniques for Software Quality Evaluation - MaLTeSQuE 2019*, Francesca Arcelli Fontana, Bartosz Walter, Apostolos Ampatzoglou, Fabio Palomba, Gilles Perrouin, Mathieu Acher, Maxime Cordy, and Xavier Devroey (Eds.). ACM Press, New York, New York, USA, 37–42. https://doi.org/10.1145/3340482.3342747

[20] Valentina Lenarduzzi, Nyyti Saarimaki, and Davide Taibi. 2019. On the Diffuseness of Code Technical Debt in Java Projects of the Apache Ecosystem. In *2019 IEEE/ACM International Conference on Technical Debt (TechDebt)*. IEEE, 98–107. https://doi.org/10.1109/TechDebt.2019.00028

[21] Valentina Lenarduzzi, Nyyti Saarimäki, and Davide Taibi. 2019. The Technical Debt Dataset. In *Proceedings of the Fifteenth International Conference on Predictive Models and Data Analytics in Software Engineering*, Leandro Minku, Foutse Khomh, and Jean Petrić (Eds.). ACM, New York, NY, USA, 2–11. https://doi.org/10.1145/3345629.3345630

[22] Rainer Niedermayr. 2016. Why we don't use the Software Maintainability Index. https://www.cqse.eu/en/news/blog/maintainability-index/

[23] Rainer Niedermayr, Tobias Röhm, and Stefan Wagner. 2019. Too trivial to test? An inverse view on defect prediction to identify methods with low fault risk. *PeerJ. Computer science* 5 (2019), e187. https://doi.org/10.7717/peerj-cs.187

[24] Ken Power. 2013. Understanding the impact of technical debt on the capacity and velocity of teams and organizations: Viewing team and organization capacity as a portfolio of real options. In *2013 4th International Workshop on Managing Technical Debt (MTD 2013)*, Philippe Kruchten (Ed.). IEEE, Piscataway, NJ, 28–31. https://doi.org/10.1109/MTD.2013.6608675

[25] Robert Ramač, Vladimir Mandić, Nebojša Taušan, Nicolli Rios, Sávio Freire, Boris Pérez, Camilo Castellanos, Darío Correal, Alexia Pacheco, Gustavo Lopez, Clemente Izurieta, Carolyn Seaman, and Rodrigo Spinola. 2022. Prevalence, common causes and effects of technical debt: Results from a family of surveys with the IT industry. *Journal of Systems and Software* 184 (2022), 111114. https://doi.org/10.1016/j.jss.2021.111114

[26] Nicolli Rios, Manoel Gomes de Mendonça Neto, and Rodrigo Oliveira Spínola. 2018. A tertiary study on technical debt: Types, management strategies, research trends, and base information for practitioners. *Information and Software Technology* 102 (2018), 117–145. https://doi.org/10.1016/j.infsof.2018.05.010

[27] Klaus Schmid. 2013. On the limits of the technical debt metaphor some guidance on going beyond. In *2013 4th International Workshop on Managing Technical Debt (MTD 2013)*, Philippe Kruchten (Ed.). IEEE, Piscataway, NJ, 63–66. https://doi.org/10.1109/MTD.2013.6608681

[28] Frank L. Schmidt and John E. Hunter. 1996. Measurement error in psychological research: Lessons from 26 research scenarios. *Psychological Methods* 1, 2 (1996), 199–223. https://doi.org/10.1037/1082-989X.1.2.199

[29] Edith Tom, Aybüke Aurum, and Richard Vidgen. 2013. An exploration of technical debt. *Journal of Systems and Software* 86, 6 (2013), 1498–1516. https://doi.org/10.1016/j.jss.2012.12.052

[30] Stefan Wagner, Daniel Mendez, Michael Felderer, Daniel Graziotin, and Marcos Kalinowski. 2020. Challenges in Survey Research. In *Contemporary Empirical Methods in Software Engineering*, Michael Felderer and Guilherme Horta Travassos (Eds.). Springer International Publishing, Cham, 93–125. https://doi.org/10.1007/978-3-030-32489-6{_}4

[31] Jeffrey M. Wooldridge. 2010. *Econometric analysis of cross section and panel data* (2nd ed. ed.). MIT Press, Cambridge.