 Acta Univ. Sapientiae Informatica **13,** 2 (2021) 335–360

DOI: 10.2478/ausi-2021-0016

Reproducibility in the technical debt domain

Kristóf SZABADOS
Eötvös Loránd University,
Budapest, Hungary
email: SzabadosKristf@gmail.com

Izabella Ingrid FARKAS
Eötvös Loránd University,
Budapest, Hungary
email: Ingrid.Farkas@inf.elte.hu

Attila KOVÁCS
Eötvös Loránd University,
Budapest, Hungary
email: Attila.Kovacs@inf.elte.hu

Abstract.

Context: It is crucial to understand how reproducible the measurement results in the scientific publications are, as reproducibility is one of the cornerstones of engineering.

Objective: The goal of this study is to investigate the scientific publications presented at the premier technical debt conferences by understanding how reproducible the reported findings are.

Method: We conducted a systematic literature review of 135 unique papers published at the "International Workshop on Managing Technical Debt" and the "International Conference on Managing Technical Debt", the premier scientific conference series on technical debt.

Results: Only 44 of the investigated 135 papers presented numerical evidence and only 5 papers listed the tools, the availability of the tools, and the version of the tools used. For the rest of the papers additional information would have been needed for the potential reproducibility. One of the published papers even referred to a pornographic site as a source of a toolset for empirical research.

Computing Classification System 1998: D.2.3, D.2.4, D.2.5, D.2.7, D.2.8, D.2.9,
Mathematics Subject Classification 2010: 68N30
Key words and phrases: technical debt, reproducibility, literature review, software quality, research reporting quality

Conclusions: The field of technical debt research might have a reproducibility crisis as only approx. 32% of the papers published at the most prestigious workshop/conference series of the field present measurement results, of which only approx. 11% (4% of all papers investigated) identify the tools used, to make reproduction possible. Our findings might even point out to a bias in the field: researchers of this domain pursue Technical Debts in order to improve the quality of software products, assuming that the software products used for measurement are flawless and no longer possible to improve upon, which is a contradiction in and of itself.

1 Introduction

Reproducibility is one of the cornerstones of engineering science. Results, that can not be consistently reproduced (within the same boundary conditions and field-specific accepted divergence) by independent or groups of researchers, using different measurement tools, can not be reliably built upon to extend scientific knowledge. Such scientific publications can even erode the general public's trust in scientific claims.

Reproducing results with different measurement instruments was already shown to be hard in the technical debt domain research. Results of different tools claiming to measure technical debt might not be statistically correlated ([48]), might even use different terms and metrics ([10]). Even when using the same tool, the exact version used for measurements might matter.

On one hand, researchers and tool vendors are actively working on identifying new ways and methods for software product quality improvement, for example by identifying constructs that might lead to faults ([58]). Detecting such new constructs may increase the technical debt related measures reported by tools in the future.

On the other hand, the reported data are frequently overestimated ([52]). In case of junior developers it could even be up to 2 - 20 times ([40]). The fault proneness of the found issues (how likely they would be to lead to bugs) might also be unclear ([39], [44]). Improving the theory and precision of the detection algorithms might lead to tools reporting different (maybe drastically smaller) technical debt related measures in the future.

Also, these tools sometimes evolve even with backwards incompatible changes, which can make community extensions obsolete or unable working together with newer versions ([11]). Research including such extensions might produce misleading output.

In this paper we overview the articles published at the domain's premier workshop series "Workshop on Managing Technical Debt" ([14], [63], [64], [65], [66], [67], [68]) and conference series "International Conference on Technical Debt" ([69], [70], [71]), [72]) in order to see how many of them include the measured values:

- The name of all of the tools used for the measurements.
- The public availability of those tools.
- Exact versions of the applied tools.

These information would be indispensable for independent researchers to be able to reproduce the presented results. Please note that we did not require the input data to be publicly available.

We have found that of the 135 articles published in the proceedings of these conferences, only 44 presented any numerical measurement results. Of these 44 articles, only 5 contained all of the information about the used tools that might be needed to reproduce the results. We also found an article, published in 2018 ([8]), referencing a website as a source of "a toolset developed to support empirical software engineering research", which was by 2021 showing pornographic content.

The paper is organised as follows. In Section 2 we present some earlier work related to the subject. Section 3 presents our research methodology. Section 4 presents our findings. Section 5 deals with the validity of our results. Finally, Section 6 summarises our findings and Section 7 offers ideas for further research.

2 Related work

The term *debt*, in a technical context, was first used in 1992 by Cunningham [20] in his experience report to illustrate the difference between a theoretical waterfall development[51] and incremental growth (enabled by object oriented programming at the time). "The traditional waterfall development cycle has endeavored to avoid programming catastrophe by working out a program in detail before programming begins", "we recognize this amounts to preserving the concept of payment up-front and in-full". In incremental growth development can start before all of the requirements are collected, all architecture is completely designed and all details are fixed, but comes with a risk: "Shipping first time code is like going into debt. A little debt speeds development so long as it is paid back promptly with a rewrite. Objects make the cost

of this transaction tolerable". Stating that incremental growth "leads to the most appropriate product in the shortest possible time" and concluding that "The modularity offered by objects and the practice of consolidation make the alternative, incremental growth, both feasible and desirable in the competitive financial software market".

Since 1992 the technical debt research domain has greatly expanded. The first scientific workshop titled "Workshop on Managing Technical Debt" ([14]) was organized already in 2010. Khomyakov et al. [36] found 603 papers published between 2011 and 2017 related to the automated measurement of technical debt. According to Lenarduzzi et al. [38] in 2018 alone (between March 2018 to December 2018) 384 unique papers were published related to technical debt research. By the end of 2021 already 7 workshops have been organized in the series "Workshop on Managing Technical Debt" ([14], [63], [64], [65], [66], [67], [68]) and 4 conferences in the conference series "International Conference on Technical Debt" ([69], [70], [71], [72]).

While the domain expanded with papers reporting valuable results, some of the papers pointed out to some challenges. Barta et al. ([11]) presented how backward incompatible changes were introduced during the evolution of SonarQube's API. They investigated 66 plug-ins listed as community plug-ins on github[1]. They found that 22 were by the time of their investigation obsolete, 5 of the 23 most recently updated plug-ins (at that time) were already incompatible to the point of not working with the newest SonarQube version. Parodi et al. [48] showed that even different static analysis tools (Sonar, FindBugs) claim to measure technical debt, their outputs are not statistically correlated. They conclude that "Static code analysis tools must be thoroughly studied in order to evaluate if they represent meaningful proxies for Technical Debt" Paris et al. [10] studied numerous tools (both commercial and research prototypes) that claimed to measure technical debt. They found among others that "different tools adopt different terms, metrics, and ways to identify and measure technical debt". They also observed fragmentation of communities on discussion forums on the internet around tools: different tools being popular on different web forums. Saarimki et al. [52] investigated the accuracy of the fixing time estimations for technical debt items by SonarQube on 15 projects with 65 student developers. They found that the estimations were only correct for 2 projects, for 9 projects they were overestimated. Lenarduzzi et al. [40] investigated (among others) how long it takes for junior developers to refactor technical debt items detected by SonarQube. They found that the time

[1] https://github.com/SonarQubeCommunity (last accessed 2021.10.05)

estimated by SonarQube was always at least 2 times higher than the actual fixing time, in some cases 20 times higher. Marcilio et al. [44] studied the realistic use of SonarQube on 421 976 issues from 246 projects. They found that only approx. 13% of the issues reported by SonarQube were resolved. They conjectured that "just a subset of the checkers reveal real design and coding flaws, and this might artificially increase the technical debt of the systems". Lenarduzzi et al. [39] conducted an empirical study on 21 open-source projects to understand how fault-prone SonarQube's violations are. They found that of the 202 violations defined for Java only 26 have a low fault-proneness, "violations classified as "bug" does not seem to be the root cause of faults", warning that the current way of calculating technical debt is incorrect as several non-fault prone rules are counted as fault-prone, while some other rules should be considered fault-prone. Concluding that "companies should carefully consider which rules they really need to apply".

Even if only looking at the release notes of SonarQube (the tool used most often in the reviewed articles), the tool obviously goes through changes that might impact the measures it reports for a given source code:

- On one hand, the number of rules supported for Java increased from 400+ in version 7.3[2] (released on August 13, 2018) to 550+ by version 8.0[3] (released on October 16th, 2019). More rules, finding more instances of quality issues, potentially lead to higher technical debt values reported for the same source codes.

- On the other hand, version 8.2[4] describes its Java related changes as "Ground-up rewrite brings more accurate analysis, with fewer false positives". More precise detection reporting fewer false positives, potentially leads to lower technical debt values reported for the same source codes.

Other software products used for checking code quality are also not immune to software quality issues, that might affect their measured and reported values. We have already reported some issues to the producers of such products. For example:

- The functionality of PMD, checking if a public function is commented or not, might not correctly detect in some cases that the function has a comment, falsely reporting that the function is undocumented [5].

[2] https://www.sonarqube.org/sonarqube-7-3/ (last accessed 2021.10.05)
[3] https://www.sonarqube.org/sonarqube-8-0/ (last accessed 2021.10.05)
[4] https://www.sonarqube.org/sonarqube-8-2/ (last accessed 2021.10.05)
[5] https://github.com/pmd/pmd-eclipse-plugin/issues/138 (last accessed 2021.10.05)

- We have found a situation, where the correctness of the code was not evaluated in enough depth in a sub-tool, that lead to false-positive finding reported in the main tool and complexities in the issue reporting procedure. In a particular case[6] it turned out, that:

  1. The root of the problem is not in the given product, but in a different product they use to offer their functionalities.
  2. The developers/maintainers don't have the resources needed to help in handling the issue, or to help report the issue (with additional information) to the correct source.
  3. We, as users, are not well equipped to identify which sub-product embedded in/used by this main-product is the root of the problem. Also, not well equipped to provide the developers of that sub-product with additional information on how their product is called/integrated.

In general, at the time of writing this article there are:

- 729 open bugs for the "Static Analyzer" component of Clang[7]
- 8 066 for Clang itself[8].
- At least 10 000 for GCC[9]
- 519 open issues for PMD[10]
- 386 open issues for SpotBugs[11]
- SonarSource also has an active community[12] with many topic related to shortcomings of some products (like SonarQube itself).

[6] https://github.com/Ericsson/codechecker/issues/3098 (last accessed 2021.10.05)

[7] https://bugs.llvm.org/buglist.cgi?bug_status=__open__&component=Static%20Analyzer&limit=0&order=priority%2Cbug_severity&product=clang&query_format=advanced (last accessed 2021.10.05, needs registered user)

[8] https://bugs.llvm.org/buglist.cgi?bug_status=__open__&limit=0&no_redirect=1&order=priority%2Cbug_severity&product=clang&query_format=specific (last accessed 2021.10.05, needs registered user)

[9] https://gcc.gnu.org/bugzilla/buglist.cgi?bug_status=__open__&limit=0&no_redirect=1&order=priority%2Cbug_severity&product=gcc&query_format=specific (last accessed 2021.10.05, needs registered user)

[10] https://github.com/pmd/pmd/issues (last accessed 2021.10.05)

[11] https://github.com/spotbugs/spotbugs/issues (last accessed 2021.10.05)

[12] https://community.sonarsource.com/tags (last accessed 2021.10.05)

3 Methodology

In order to understand how reproducible the results published in the technical debt domain research are, we conducted a systematic literature review. In this section, we describe the goal of our research, report our search strategy and paper evaluation method.

3.1 Goal of our research

The goal of the research presented in this paper was to investigate the existing body of knowledge in software engineering to understand how reproducible the academic papers published in the technical debt domain research are.

3.2 Search strategy

The search strategy involves the bibliographic sources, the definition of the inclusion and exclusion criteria.

`Bibliographic sources:` To get a good view of the field of technical debt research, we decided to use the domain's premier workshop series "Workshop on Managing Technical Debt" ([14], [63], [64], [65], [66], [67], [68]) and conference series "International Conference on Technical Debt" ([69], [70], [71]), [72]) as sources. We used the ACM Digital Library[13] and IEEEXplore Digital Library[14] to reach the articles that were published as part of the proceedings of these conferences.

`Inclusion criteria:`

- Papers presenting measured values in relation to technical debt (numerical values that might be reproduced and compared to in a replication study).

`Exclusion criteria:`

- Papers not presenting a measurement result (example: [3], [13]).
- Papers using surveys as input for measurements (example: [12]).

Please note, that we did include papers that performed their measurements on projects that were anonymized in the paper. We believe that the field needs to have information from the industry, which might come with such limitations.

[13] `https://www.acm.org/` (last accessed 2021.10.05)
[14] `https://ieeexplore.ieee.org` (last accessed 2021.10.05)

`Search process:` The search was conducted in March 2021.

Based on the described process, we retrieved a total of 122 papers for the review.

3.3 Paper evaluation

To decide if a result presented in a paper might be reproducible, we tried to identify what tools were described as being used to create the result, and checked how well they were identified:

- Is the name of the tool represented?
- Is there a direct link provided, from where the tool could be accessed (either for free or commercially)?
- Is the version of the tool identified?

During the evaluation of the papers, we only considered those tools, for each paper, that were presented as being used for creating the published result. Tools mentioned in relation to other works or in comparison with the used tools, were not counted as used in the paper.

Please note, that this method comes with a limitation: as we did not actually try to reproduce the published result, we could not detect if the researchers had to use some undisclosed tools to get to the published result.

4 Results

In this section we present our findings. In a number of investigated papers it is not clear what tools the researchers were using. Even if the tool's name resembles a well known tool we could not precisely cite the tool, as it is not clear if they meant the same tool and if so which version of it. For this reason in our short assessments of all investigated papers we list the tool names found in a paper, in apostrophes first. This is done to provide a common presentation format and to indicate that the name might not refer to the tool readers could automatically associate it with.

4.1 Details for 2010 ([14])

According to the Software Engineering Institute of Carnegie Mellon University [17] the results of the first workshop organized for the topic of managing technical debt was published as [14]. This was a single paper, that only introduced the idea and importance of holding international workshops on managing technical debt, but in itself did not disclose measurements.

Table 1: The general result of our Systematic Literature Review

Venue	Nr. of articles	Presents measured results	Fully identifies used tools
MTD 2010 ([14])	1	0	0
MTD 2011 ([63])	9	4	0
MTD 2012 ([64])	11	3	0
MTD 2013a ([65])	11	2	0
MTD 2013b ([27])	1	0	0
MTD 2014 ([66])	9	4	1
MTD 2015 ([67])	12	7	0
MTD 2016 ([68])	7	3	1
TechDebt 2018 ([69])	22	7	0
TechDebt 2019 ([70])	24	3	0
TechDebt 2020 ([71])	15	4	1
TechDebt 2021 ([72])	13	7	2

4.2 Details for: "MTD 2011: Proceedings of the 2nd Workshop on Managing Technical Debt" ([63])

Out of the 9 papers published as part of the 2nd workshop on Managing Technical Debt we have found 4 that disclosed some measurement results. None of which might be reproducible.

The papers that might have reproducibility problems:

- "An Empirical Model of Technical Debt and Interest" ([47]) mentions that they use the "Software Analysis Toolkit" of the "Software Improvement Group", but there is no reference to the tool or a version number disclosed.
- "From Assessment to Reduction: How Cutter Consortium Helps Rein in Millions of Dollars in Technical Debt" ([33]) mentions that "Cutter's technical debt assessment is an automated analysis of code deficits", but the availability or version of the tool is not disclosed. The only referenced document does not seem to be available any longer.
- "An Extraction Method to Collect Data on Defects and Effort Evolution in a Constantly Modified System" ([34]) mentions querying the "Mantis Tool" and contains a reference to it, but does not disclose the version used.

- "Prioritizing Design Debt Investment Opportunities" ([62]) discloses the metrics they measured and the methods they used, but not the tool.

4.3 Details for: "MTD 2012: Proceedings of the Third International Workshop on Managing Technical Debt" ([64])

Out of the 11 papers published as part of the third International Workshop on Managing Technical Debt we have found 3 that disclosed some measurement results, none of which might be reproducible.

The papers that might have reproducibility problems:

- "Estimating the Size, Cost, and Types of Technical Debt" ([21]) uses the "Application Intelligence Platform" of "CAST", without disclosing the version used.
- "Investigating the Impact of Code Smells Debt on Quality Code Evaluation" ([30]) uses several tools:
  - "iPlasma": the referenced web page is not available.
  - "Eclipse Metrics": version used is not disclosed, although according to the referenced page the tool was not updated since 2013.
  - "Google CodePro Analytix": the referenced webpage is not available.
- "What Is the Value of Your Software?" ([24]) discloses taking the data from the "software analysis warehouse" of the "Software Improvement Group", without disclosing a reference to the data, when it was accessed, with which tool version the data was measured.

4.4 Details for: "MTD 2013: Proceedings of the 4th International Workshop on Managing Technical Debt" ([65])

Out of the 11 papers published as part of the 4th International Workshop on Managing Technical Debt we have found 2 that disclosed some measurement results, none of which might be reproducible.

The papers that might have reproducibility problems:

- "Exploring Software Supply Chains From a Technical Debt Perspective" ([45]) uses the tools "Sonar", "Understand" and "Cytoscape". Although their web pages are referenced and their last access time is presented, it is not clear which version of these tools was used for the measurement

(that might be different from the version available on the web page at the last time it was accessed).

- "Generating Precise Dependencies for Large Software" ([61]) uses a tool written by the authors for measurements. There is no link to the tool, not even it's name is disclosed.

4.5 Details for: "MTD 2013: Proceedings of the fifth International Workshop on Managing Technical Debt" ([27])

According to the Software Engineering Institute of Carnegie Mellon University ([17]) the summary of the fifth workshop organized for the topic of managing technical debt was published as [27].

The workshop does not seem to have had a proceedings published. Although the Software Engineering Institute of Carnegie Mellon University ([17]) makes the slides of some of the presentations of the workshops available, we excluded them as not being published articles.

This was a single paper, that summarized the discussions of the workshop, but in itself did not disclose measurements.

4.6 Details for: "MTD 2014: Proceedings of the 2014 Sixth International Workshop on Managing Technical Debt" ([66])

Out of the 9 papers published as part of the sixth International Workshop on Managing Technical Debt we have found 4 that disclosed some measurement results, only 1 also disclosing the version of the tool used.

The papers that might have reproducibility problems:

- "A Framework for Estimating Interest on Technical Debt by Monitoring Developer Activity Related to Code Comprehension" ([55]) uses the tools "Blaze" and "Understand". The version of neither of them is disclosed, also the reference of Blaze points to a paper, not directly to the tool.
- "Are the Methods in Your Data Access Objects (DAOs) in the Right Place?A Preliminary Study" ([9]) uses the tool "calculadora-de-daos". The link of the tool, is either corrupted or points to a source code repository no longer available.
- "The Correspondence between Software Quality Models and Technical Debt Estimation Approaches" ([35]) uses the tools "PMD", "findbugs", "SonarQube", "Understand", "inFusion" all of which identified by their websites only, without disclosing their versions used. In the case of inFusion the website does not seem to be available.

The paper fully identifying the tool used:

- "Explicating, Understanding and Managing Technical Debt from Self-Driving Miniature Car Projects" ([43]) uses the tools "SonarQube", "Sonar-Runner" and "SonarWay" disclosing the version of these tools used for their measurements.

4.7 Details for: "2015 IEEE 7th International Workshop on Managing Technical Debt (MTD 2015)" ([67])

Out of the 12 papers published as part of the 7th International Workshop on Managing Technical Debt we have found 7 that disclosed some measurement results, none of which might be reproducible.

The papers that might have reproducibility problems:

- "A Contextualized Vocabulary Model for Identifying Technical Debt on Code Comments" ([23]) uses the tool "eXcomment" developed by the authors. The tool is only identified by a web link, that no longer works.
- "Detecting and Quantifying Different Types of Self-Admitted Technical Debt" ([42]) uses "jDeodorant" for information extraction and an unnamed tool supporting manual classification. The version of jDeodorant is not disclosed and the reference points to an academic paper, not directly a tool.
- "Estimating the Breaking Point for Technical Debt" ([16]) uses the tools "JCaliper" and "SEAgle". Both of them developed by the authors, neither of them has their version disclosed. SEAgle is identified by a reference to an academic paper, while jCaliper is not identified at all.
- "Technical Debt of Standardized Test Software" ([57]) uses "Titan" for measurement, only identified by a reference to an academic paper.
- "Towards a Prioritization of Code Debt: A CodeSmell Intensity Index" ([31]) uses the tool "JCodeOdor" developed by the authors. The tool is only identified by a reference to an academic article, without a direct reference to the tool, or its version used.
- "Towards an Open-Source Tool for Measuring and Visualizing the Interest of Technical Debt" ([28]) uses the tool "MIND (ManagIng techNical Debt)" built as a plug-in of "SonarQube". Both tools are identified by a link to their web pages, neither of them has their version disclosed.

- "Validating and Prioritizing Quality Rules for Managing Technical Debt: An Industrial Case Study" ([29]) uses the tool "Technical Debt Analyzer" built by the authors as a plug-in for "SonarQube". SonarQube is only identified by a link to its web page, Technical Debt Analyzer is not identified in any way.

4.8 Details for: "2016 IEEE 8th International Workshop on Managing Technical Debt (MTD 2016)" ([68])

Out of the 7 papers published as part of the 8th International Workshop on Managing Technical Debt we have found 3 that disclosed some measurement results, only 1 also disclosing the version of the tool used.

The papers that might have reproducibility problems:

- "Practical Technical Debt Discovery by Matching Patterns in Assessment Graph" ([54]) does not identify the tools used for measurements.
- "The Perception of Technical Debt in the Embedded Systems Domain: An Industrial Case Study" ([7]) does not identify the tool used for measuring maintainability (used as a proxy for technical debt).

The paper fully identifying the tool used:

- "Technical Debt Indexes provided by tools:a preliminary discussion" ([32]) uses the tools "CAST", "inFusion", "Sonargraph", "SonarQube" and "Structure101". Each identified by their respective web pages and the exact version of the tools used.

4.9 Details for: "TechDebt '18: Proceedings of the 2018 International Conference on Technical Debt" ([69])

Out of the 22 papers published as part of the 2018 International Conference on Technical Debt we have found 7 that disclosed some measurement results, none of which might be reproducible.

The papers that might have reproducibility problems:

- "A Framework for Managing Interest in Technical Debt: An Industrial Validation" ([8]) uses the tools "SonarQube", "Percerons Client" and "Breaking Point Calculator".
  - For SonarQube only the documentation is referenced, with a link, not the used version of the tool or the tool itself.

  - Breaking Point Calculator was developed by the authors, is not referenced and does not seem to be available for the community.
  - "Percerons Client" is identified by a link to, what seems to be a website for pornographic movies[15] (see figure 1).

- "An exploratory study on the influence of developers in technical debt" ([4]) uses "SonarQube" and the author's own scripts. SonarQube is only identified by a reference to an academic article, without a direct reference to the tool, or its version used. The author's own scripts are not available.
- "Design Debt Prioritization: A Design Best Practice-Based Approach" ([50]) uses "MUSE" a tool developed by the authors, identified by a reference to an academic article, without a direct reference to the tool, or its version used.
- "Evaluating Domain-Specific Metric Thresholds: An Empirical Study" ([46]) uses the tools "CK Tool" and "TDTool". "CK Tool" is identified with a link to its source code repository, without revealing the version used. TDTool is identified by a reference to an academic article, without a direct reference to the tool, or its version used.
- "From Lasagna to Spaghetti, a Decision Model to Manage Defect Debt" ([2]) uses a tool developed by the authors for measurements, but does not identify the tool or its availability.
- "Prioritize Technical Debt in Large-Scale Systems Using CodeScene" ([60]) uses "CodeScene". The tool itself is only indirectly referenced in a reference of a data set, that is stored on a web page similar to the tool's name. The version used is not identified.
- "The developer's dilemma: factors affecting the decision to repay code debt" ([6]) uses "SonarQube", which is only identified by a reference to the tool's website, without disclosing the version used for measurements.

4.10 Details for: "TechDebt '19: Proceedings of the Second International Conference on Technical Debt" ([70])

Out of the 24 papers published as part of the 2019 International Conference on Technical Debt we have found 3 that disclosed some measurement results, none of which might be reproducible.

[15] Already on 2021.05.18, last checked on 2021.10.05

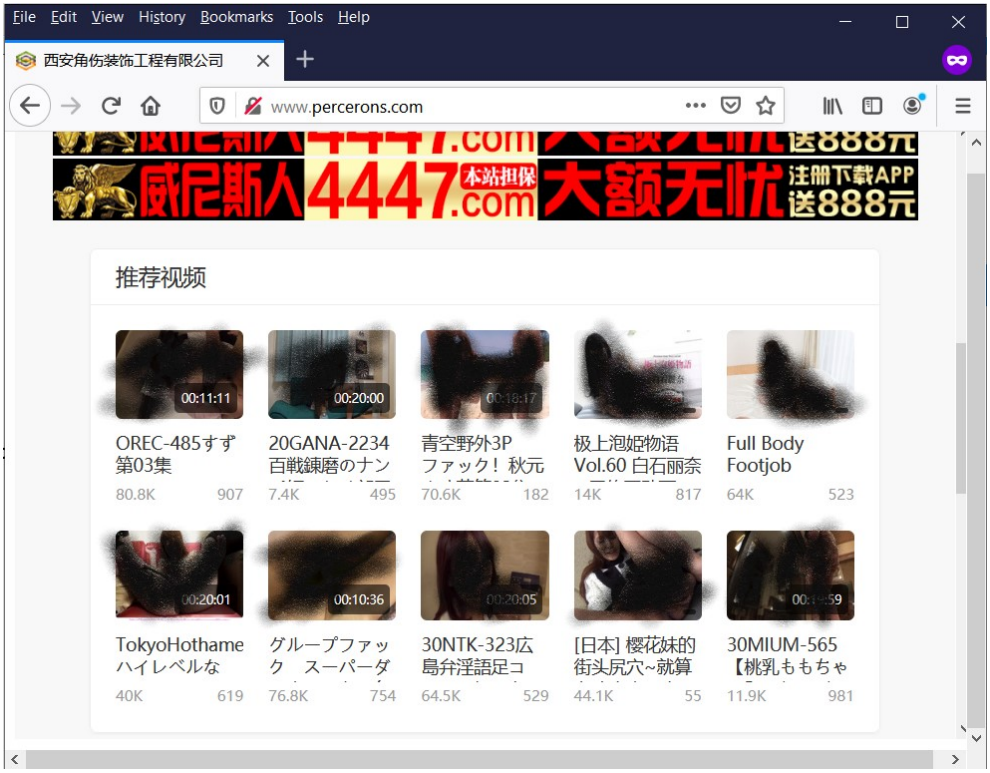

Figure 1: The pornographic site referenced by Ampatzoglou et al. ([8]) as the source of "a toolset developed to support empirical software engineering research". (last accessed 2021.10.05)

The papers that might have reproducibility problems:

- "Investigating on the Impact of Software Clones on Technical Debt" ([41]) uses "NiCad Clone Detector" and "SonarQube", neither of them being identified in the article.
- "On the Diffuseness of Code Technical Debt in Java Projects of the Apache Ecosystem" ([53]) uses "SonarQube", identified only by a reference to the tool's website, without disclosing the version used for measurements.
- "The Delta Maintainability Model: Measuring Maintainability of Fine-Grained Code Changes" ([25]) proposes and uses a new maintainability model "Delta Maintainability Model" (DMM) for fine-grained measure-

ments of code changes, without making the tool implemented and used for measurements available (or at least it is not referenced in the article).

4.11 Details for: "TechDebt '20: Proceedings of the 3rd International Conference on Technical Debt" ([71])

Out of the 15 papers published as part of the 2020 International Conference on Technical Debt we have found 4 that disclosed some measurement results, only 1 also disclosing the version of the tool used.

The papers that might have reproducibility problems:

- "Detecting Bad Smells with Machine Learning Algorithms: An Empirical Study" ([19]) uses the tools "JDeodorant", "JSpirit", "PMD", "DECOR", "Organic". While PMD is identified with a link to it's open source repository, the other tools are identified by references to academic papers. None of the tools used has it's version disclosed.

- "Towards Microservice Smells Detection" ([49]) uses "Arcan", a tool developed by the authors, identified by a reference to 2 academic articles, without a direct reference to the tool, or its version used.

- "The Hidden Cost of Backward Compatibility: When Deprecation Turns into Technical Debt - an Experience Report" ([56]) uses a unidentified tool developed by the authors of the article.

The paper fully identifying the tool used:

- "An Empirical Study on Self-Fixed Technical Debt" ([59]) uses the tool "SonarQube", disclosing also the version used for their measurements.

4.12 Details for: "TechDebt '21: Proceedings of the 4th International Conference on Technical Debt" ([72])

Out of the 13 papers published as part of the 2021 International Conference on Technical Debt we have found 7 that disclosed some measurement results, only 2 also disclosing the version of all of the tools used.

The papers that might have reproducibility problems:

- "Assessing Smart Contracts Security Technical Debts" ([1]) uses several tools:

  - "Slither", "SmartCheck", "Securify" and "Mythril" are identified by reference to academic articles, without direct references to the tools, or their versions disclosed.
  - "Manticore", "Solhint" and "Ethlint" are identified by links to their GitHub repositories, without disclosing the used version.
  - "Sfuzz" is referenced by an academic article, without disclosing the used version. It is also disclosed, that a web interface was used, not a local installation.

- "Business-Driven Technical Debt Prioritization: An Industrial Case Study" ([22]) uses an undisclosed tool.

- "Predicting Relative Thresholds for Object Oriented Metrics" ([5]) uses a tool developed by the authors for measurements, but does not identify the tool or its availability.

- "Worst Smells and Their Worst Reasons" ([26]) uses "SonarCloud", identified by a reference to the tools website, without disclosing the version used for measurements.

- "Impact of Opportunistic Reuse Practices to Technical Debt" ([15]) uses "SonarQube" and "Arcan". SonarQube is identified by a link to its website and the version used. Arcan is identified by a reference to an academic article, without a direct reference to the tool or its version disclosed.

The paper fully identifying the tool used:

- "Carrot and Stick approaches revisited when managing Technical Debt in an educational context" ([18]) uses the tool "SonarQube", disclosing also the version used for their measurements. (They also use "PostgreSQL" and "MySQL server" without any identification, "Tablon" identified by a link to a web page without dislosing the version used. These tools are not used to measure technical debt.)

- "Experiences on Managing Technical Debt with Code Smells and AntiPatterns" ([37]) uses the tools "CodeMR" and "IntelliJ IDEA code inspection tool". CodeMR is dientified with a link to its website and the version used is disclosed. The "IntelliJ IDEA code inspection tool" also has the used version disclosed.

5 Threats to validity

While we believe that our results are valid in their described context, we have to point out that this is a limited context.

We did not try to reproduce the results published in the papers, and we did not include in our reproducibility criteria the public availability of the projects used as input to the tools. This way, it was not possible to detect potential data manipulation or required tools not being mentioned.

We only reviewed the papers published in the workshop and conference series we believed to be the most prestigious in the field. It might very well be the case that reviewing all publications in the field would find a different number and ratio of reproducible papers.

6 Summary

In this paper we presented our work and findings on the reproducibility of research results in the technical debt research domain, published at the most prestigious workshop and conference series.

We have found that of the 135 papers published in the proceedings of these conferences, only 44 presented any numerical measurements. Of these 44 papers, only 5 contained the information about the used tools that might be needed to reproduce the result.

This might indicate a bias in the field. Researchers working to improve the quality of software products might not be aware that the software products they use for measurements could also need quality improvements.

One of the papers we investigated ([8]) referenced a website showing pornographic content. This situation is problematic for both the publisher of the proceedings (that now indirectly advertises pornographic content) and the field in general (in 2 years after publication, entire toolset can move or disappear).

We also found articles where the measurement tool was not available (either was never made public, or their repository already was deleted).

Our final observation is that the field should use more stringent practices when reviewing and handling research publications to ensure that the published results can be reproduced/replicated (and so safely built upon) by others.

We believe this paper is a step forward to the technical debt researching community. Improving the reproducibility of papers published in this field by

raising the scientific standard could enable more scrutiny towards the published results, making them safer to build on.

We would recommend authors to follow (and reviewers to check) the following guidelines to make their papers more reproducible:

- Disclose in the paper the tools used:
  - Hardware used.
  - Software stack used (operating system etc..)
  - The names, configuration, precise version identification for all of the tools used and a link to a web page from where the tool can be obtained (even if commercially).
  - Governance completeness. When using cloud based tools or ones shared with other groups, if the team does not have complete and exclusive governance, the tool might be updated and setting changed even during measurements, without the team being notified of such actions[16].
  - Any other special circumstances that could affect the numbers reported. For example:
    - According to the download site[17] of SonarQube the detection of injection flows for Java is only available in the commercial editions of the software.
    - Before version 6.6 of SonarQube[18] the built-in profiles could be changed. Potentially leading to researchers miss-reporting what setting were used.
- Make a direct statement that all of the tools used are listed. No other scripts, methods, transformations were needed to produce the reported results.
- If the input was open-source data, identify it. If the input data is the property of a company, include information that might reasonably be needed for other researchers to contact that company and request access to the data.

[16] Please note, that determining this property might require care. For example CAST promotes its solution as being SaaS subscription, but actually the measurements are done locally, only the results are uploaded into the cloud. https://www.castsoftware.com/products/highlight/pricing (last accessed 2021.10.05)

[17] https://www.sonarqube.org/downloads/ (last accessed 2021.10.05)

[18] https://blog.sonarsource.com/sonarqube-6.5-in-screenshots (last accessed 2021.10.05)

7 Further work

There are several ways to extend the research presented in this paper.

We have only reviewed articles published at the most prestigious workshop and conference series. The whole field of technical debt research is bigger, containing papers published at other venues and journals, that should be reviewed.

We have limited resources, so we did not try to reproduce the results in the papers we have found to be potentially reproducible. It would be possible to get a deeper understanding of the situation by doing the actual reproducibility check.

It would also be interesting to investigate how cloud-based technical debt tools impact research and work efforts. On the one side, we can expect cloud-based solutions to spread in both work and research environments, as they free people from the mundane task of allocating and managing hardware resources, installing and managing software tools, making sure these tools are up to date and configured for security, etc. On the other side, the automated updating of the platform and software resources can cause problems for research reproducibility as these changes might happen independently from the researchers at any point in time (even during experiments).

Acknowledgment

The third author was supported by the Project no. TKP2020-NKA-06 (Application domain specific highly reliable IT solutions) with the support from the National Research, Development and Innovation Fund of Hungary, financed under the Thematic Excellence Program funding scheme.

References

[1] S. Ahmadjee, C. Mera-Gómez, R. Bahsoon, Assessing Smart Contracts Security Technical Debts, *2021 IEEE/ACM International Conference on Technical Debt (TechDebt)*, 2021, pp. 6-15. ⇒350

[2] A. Aldaeej, C. Seaman, From Lasagna to Spaghetti: A Decision Model to Manage Defect Debt, *2018 IEEE/ACM International Conference on Technical Debt (TechDebt)*, 2018, pp. 67-71. ⇒348

[3] R. Alfayez, W. Alwehaibi, R. Winn, E. Venson, B. Boehm, A systematic literature review of technical debt prioritization, *In Proceedings of the 3rd International Conference on Technical Debt (TechDebt '20)*, 2020, pp. 1–10. ⇒341

[4] R. R. Alfayez, P. Behnamghader, K. Srisopha, B. Boehm, An Exploratory Study on the Influence of Developers in Technical Debt, *2018 IEEE/ACM International Conference on Technical Debt (TechDebt)*, 2018, pp. 1-10. ⇒348

[5] S. Alhusain, Predicting Relative Thresholds for Object Oriented Metrics, *2021 IEEE/ACM International Conference on Technical Debt (TechDebt)*, 2021, pp. 55-63. ⇒351

[6] T. Amanatidis, N. Mittas, A. Chatzigeorgiou, A. Ampatzoglou, L. Angelis, The Developer's Dilemma: Factors Affecting the Decision to Repay Code Debt, *2018 IEEE/ACM International Conference on Technical Debt (TechDebt)*, 2018, pp. 62-66. ⇒348

[7] A. Ampatzoglou, A. Ampatzoglou, A. Chatzigeorgiou, P. Avgeriou, P. Abrahamsson, A. Martini, U. Zdun, K. Systa, The Perception of Technical Debt in the Embedded Systems Domain: An Industrial Case Study, *2016 IEEE 8th International Workshop on Managing Technical Debt (MTD)*, 2016, pp. 9-16. ⇒347

[8] A. Ampatzoglou, A. Michailidis, C. Sarikyriakidis, A. Ampatzoglou, A. Chatzigeorgiou, P. Avgeriou, A Framework for Managing Interest in Technical Debt: An Industrial Validation, *2018 IEEE/ACM International Conference on Technical Debt (TechDebt)*, 2018, pp. 115-124. ⇒337, 347, 349, 352

[9] M. F. Aniche, G. A. Oliva, M. A. Gerosa, Are the Methods in Your Data Access Objects (DAOs) in the Right Place? A Preliminary Study, *Sixth International Workshop on Managing Technical Debt (MTD '14)*, 2014, pp. 47-50. ⇒345

[10] P. C. Avgeriou, D. Taibi, A. Ampatzoglou, A. F. Fontana, T. Besker, A. Chatzigeorgiou, V. Lenarduzzi, A. Martini, A. Moschou, I. Pigazzini, N. Saarimäki, D. Sas, S. Toledo, A. Tsintzira. An Overview and Comparison of Technical Debt Measurement Tools, in *IEEE Software*, vol. 38, no. 3, pp. 61-71, May-June 2021. ⇒336, 338

[11] B. Barta, G. Manz, I. Siket, R. Ferenc, Challenges of SonarQube Plug-In Maintenance, *IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 2019, pp. 574-578. ⇒336, 338

[12] T. Besker, A. Martini, J. Bosch, Technical debt cripples software developer productivity: a longitudinal study on developers' daily software development work, *In Proceedings of the 2018 IEEE/ACM International Conference on Technical Debt (TechDebt '18)*, 2018, pp. 105–114. ⇒341

[13] J. Bohnet, J. Döllner, Monitoring code quality and development activity by software maps, *In Proceedings of the 2nd Workshop on Managing Technical Debt (MTD '11)*, ACM, 2011, New York, NY, USA, 9–16. ⇒341

[14] N. Brown, Y.g Cai, Y. Guo, R. Kazman, M. Kim, P. Kruchten, E. Lim, A. MacCormack, R. L. Nord, I. Ozkaya, R. Sangwan, C. Seaman, K. Sullivan, N. Zazworka, Managing technical debt in software-reliant systems, *In Proceedings of the FSE/SDP workshop on Future of software engineering research (FoSER '10)*, 2010, pp. 47–52. ⇒337, 338, 341, 342, 343

[15] R. Capilla, T. Mikkonen, C. Carrillo, F. A. Fontana, I. Pigazzini, V. Lenarduzzi, Impact of Opportunistic Reuse Practices to Technical Debt, *2021 IEEE/ACM International Conference on Technical Debt (TechDebt)*, 2021, pp. 16-25. ⇒351

[16] A. Chatzigeorgiou, A. Ampatzoglou, A. Ampatzoglou, T. Amanatidis, Estimating the breaking point for technical debt, *2015 IEEE 7th International Workshop on Managing Technical Debt (MTD)*, 2015, pp. 53-56. ⇒346

[17] Carnegie Mellon University, Software Engineering Institute, Managing Technical Debt Workshops, previous editions, 2010, https://resources.sei.cmu.edu/news-events/events/techdebt/past.cfm, last visited: 2021.10.05. ⇒342, 345

[18] Y. Crespo, A. Gonzalez-Escribano, M. Piattini, Carrot and Stick approaches revisited when managing Technical Debt in an educational context, *2021 IEEE/ACM International Conference on Technical Debt (TechDebt)*, 2021, pp. 99-108. ⇒351

[19] D. Cruz, A. Santana, E. Figueiredo, Detecting bad smells with machine learning algorithms: an empirical study, *In Proceedings of the 3rd International Conference on Technical Debt (TechDebt '20)*, 2020, ACM, New York, NY, USA, 31–40. ⇒350

[20] W. Cunningham, The WyCash Portfolio Management System, *In Addendum to the proceedings on Object-oriented programming systems, languages, and applications (Addendum) (OOPSLA '92)*. ACM, NY, USA, 1992, 29–30. ⇒337

[21] B. Curtis, J. Sappidi, A. Szynkarski, Estimating the size, cost, and types of Technical Debt, *Third International Workshop on Managing Technical Debt (MTD'12)*, 2012, pp. 49-53. ⇒344

[22] R. R. de Almeida, R. do Nascimento Ribeiro, C. Treude, U. Kulesza, Business-Driven Technical Debt Prioritization: An Industrial Case Study, *2021 IEEE/ACM International Conference on Technical Debt (TechDebt)*, 2021, pp. 74-83. ⇒351

[23] M. A. de Freitas Farias, M. G. de Mendonça Neto, A. B. d. Silva, R. O. Spínola, A Contextualized Vocabulary Model for identifying technical debt on code comments, *2015 IEEE 7th International Workshop on Managing Technical Debt (MTD)*, 2015, pp. 25-32. ⇒346

[24] J. de Groot, A. Nugroho, T. Bäck, J. Visser, What is the value of your software?, *Third International Workshop on Managing Technical Debt (MTD'12)*, 2012, pp. 37-44. ⇒344

[25] M. di Biase, A. Rastogi, M. Bruntink, A. van Deursen, The Delta Maintainability Model: Measuring Maintainability of Fine-Grained Code Changes, *2019 IEEE/ACM International Conference on Technical Debt (TechDebt)*, 2019, pp. 113-122. ⇒349

[26] D. Falessi, R. Kazman, Worst Smells and Their Worst Reasons, *2021 IEEE/ACM International Conference on Technical Debt (TechDebt)*, 2021, pp. 45-54. ⇒351

[27] D. Falessi, P. Kruchten, R. L. Nord, I. Ozkaya, Technical debt at the crossroads of research and practice: report on the fifth international workshop on managing technical debt, 2014, *SIGSOFT Softw. Eng. Notes 39, 2*, 31–33. ⇒343, 345

[28] D. Falessi, A. Reichel, Towards an open-source tool for measuring and visualizing the interest of technical debt, *2015 IEEE 7th International Workshop on Managing Technical Debt (MTD)*, 2015, pp. 1-8. ⇒346

[29] D. Falessi, A. Voegele, Validating and prioritizing quality rules for managing technical debt: An industrial case study, *2015 IEEE 7th International Workshop on Managing Technical Debt (MTD)*, 2015, pp. 41-48. ⇒347

[30] F. A. Fontana, V. Ferme, S. Spinelli, Investigating the impact of code smells debt on quality code evaluation, *Third International Workshop on Managing Technical Debt (MTD'12)*, 2012, pp. 15-22. ⇒344

[31] F. A. Fontana, V. Ferme, M. Zanoni, R. Roveda, Towards a prioritization of code debt: A code smell Intensity Index, *2015 IEEE 7th International Workshop on Managing Technical Debt (MTD)*, 2015, pp. 16-24. ⇒346

[32] F. A. Fontana, R. Roveda, M. Zanoni, Technical Debt Indexes Provided by Tools: A Preliminary Discussion, *2016 IEEE 8th International Workshop on Managing Technical Debt (MTD)*, 2016, pp. 28-31. ⇒347

[33] I. Gat, J. D. Heintz, From assessment to reduction: how cutter consortium helps rein in millions of dollars in technical debt, *In Proceedings of the 2nd Workshop on Managing Technical Debt (MTD '11)*, 2011, Waikiki, Honolulu, HI, USA, 24–26. ⇒343

[34] R. Gomes, C. Siebra, G. Tonin, A. Cavalcanti, Fabio Q.B. da Silva, Andre L.M. Santos, R. Marques, An extraction method to collect data on defects and effort evolution in a constantly modified system, *In Proceedings of the 2nd Workshop on Managing Technical Debt (MTD '11)*, 2011, Waikiki, Honolulu, HI, USA, 27–30. ⇒343

[35] I. Griffith, D. Reimanis, C. Izurieta, Z. Codabux, A. Deo, B. Williams, The Correspondence Between Software Quality Models and Technical Debt Estimation Approaches, *Sixth International Workshop on Managing Technical Debt (MTD '14)*, 2014, pp. 19-26. ⇒345

[36] I. Khomyakov, Z. Makhmutov, R. Mirgalimova, A. Sillitti, 2019, Automated Measurement of Technical Debt: A Systematic Literature Review, *In Proceedings of the 21st International Conference on Enterprise Information Systems - Volume 2(ICEIS)*, 2019, pages 95-106. ⇒338

[37] J. R. Lahti, A. -P. Tuovinen, T. Mikkonen, Experiences on Managing Technical Debt with Code Smells and AntiPatterns, *2021 IEEE/ACM International Conference on Technical Debt (TechDebt)*, 2021, pp. 36-44. ⇒351

[38] V. Lenarduzzi, T. Besker, D. Taibi, A. Martini, F. A. Fontana, Technical Debt Prioritization: State of the Art. A Systematic Literature Review, *arXiv:1904.12538*. ⇒338

[39] V. Lenarduzzi, F. Lomio, H. Huttunen, D. Taibi, Are SonarQube Rules Inducing Bugs?, *IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 2020, pp. 501-511. ⇒336, 339

[40] V. Lenarduzzi, V. Mandić, A. Katin, D. Taibi, How long do Junior Developers take to Remove Technical Debt Items?, *In Proceedings of the 14th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement(ESEM '20)*, 2020, Article 30, 1–6. ⇒336, 338

[41] A. Lerina, L. Nardi, Investigating on the Impact of Software Clones on Technical Debt, *2019 IEEE/ACM International Conference on Technical Debt (TechDebt)*, 2019, pp. 108-112. ⇒349

[42] E. d. S. Maldonado, E. Shihab, Detecting and quantifying different types of self-admitted technical Debt, *2015 IEEE 7th International Workshop on Managing Technical Debt (MTD)*, 2015, pp. 9-15. ⇒346

[43] M. A. Al Mamun, C. Berger, J. Hansson, Explicating, Understanding, and Managing Technical Debt from Self-Driving Miniature Car Projects, *Sixth International Workshop on Managing Technical Debt (MTD '14)*, 2014, pp. 11-18. ⇒346

[44] D. Marcilio, R. Bonifácio, E. Monteiro, E. Canedo, W. Luz, G. Pinto, Are Static Analysis Violations Really Fixed? A Closer Look at Realistic Usage of SonarQube, *IEEE/ACM 27th International Conference on Program Comprehension (ICPC)*, 2019, pp. 209-219. ⇒336, 339

[45] J. Y. Monteith, J. D. McGregor, Exploring software supply chains from a technical debt perspective, *4th International Workshop on Managing Technical Debt (MTD '13)*, 2013, pp. 32-38. ⇒344

[46] A. Mori, G. Vale, M. Viggiato, J. Oliveira, E. Figueiredo, E. Cirilo, P. Jamshidi, C. Kastner, Evaluating Domain-Specific Metric Thresholds: An Empirical Study, *2018 IEEE/ACM International Conference on Technical Debt (TechDebt)*, 2018, pp. 41-50. ⇒348

[47] A. Nugroho, J. Visser, T. Kuipers, An empirical model of technical debt and interest, *In Proceedings of the 2nd Workshop on Managing Technical Debt (MTD '11)*, 2011, Waikiki, Honolulu, HI, USA, 1–8. ⇒343

[48] E. Parodi, S. Matalonga, D. Macchi, M. Solari, Comparing technical debt in student exercises using test driven development, test last and ad hoc programming, *2016 XLII Latin American Computing Conference (CLEI)*, 2016, pp. 1-10. ⇒336, 338

[49] I. Pigazzini, F. A. Fontana, V. Lenarduzzi, D. Taibi, Towards microservice smells detection, *In Proceedings of the 3rd International Conference on Technical Debt (TechDebt '20)*, 2020, ACM, New York, NY, USA, 92–97. ⇒350

[50] R. Plösch, J. Bräuer, M. Saft and C. Körner, Design Debt Prioritization: A Design Best Practice-Based Approach, *2018 IEEE/ACM International Conference on Technical Debt (TechDebt)*, 2018, pp. 95-104. ⇒348

[51] W. W. Royce, Managing the development of large software systems, *Proceedings of IEEE WESCON*, 1970, pp. 1–9. ⇒337

[52] N. Saarimäki, M. T. Baldassarre, V. Lenarduzzi and S. Romano, On the Accuracy of SonarQube Technical Debt Remediation Time, *45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 2019, pp. 317-324. ⇒336, 338

[53] V. Lenarduzzi, N. Saarimäki, D. Taibi, On the Diffuseness of Code Technical Debt in Java Projects of the Apache Ecosystem, *2019 IEEE/ACM International Conference on Technical Debt (TechDebt)*, 2019, pp. 98-107. ⇒349

[54] A. Shapochka, B. Omelayenko, Practical Technical Debt Discovery by Matching Patterns in Assessment Graph, *2016 IEEE 8th International Workshop on Managing Technical Debt (MTD)*, 2016, pp. 32-35. ⇒347

[55] V. Singh, W. Snipes, N. A. Kraft, A Framework for Estimating Interest on Technical Debt by Monitoring Developer Activity Related to Code Comprehension, *Sixth International Workshop on Managing Technical Debt (MTD '14)*, 2014, pp. 27-30. ⇒345

[56] A. Sundelin, J. Gonzalez-Huerta, K. Wnuk, The hidden cost of backward compatibility: when deprecation turns into technical debt - an experience report, *In Proceedings of the 3rd International Conference on Technical Debt (TechDebt '20)*, 2020, ACM, New York, NY, USA, 67–76. ⇒350

[57] K. Szabados and A. Kovács, Technical debt of standardized test software, *2015 IEEE 7th International Workshop on Managing Technical Debt (MTD)*, 2015, pp. 57-60. ⇒346

[58] R. Szalay, Á. Sinkovics, Z. Porkoláb, The Role of Implicit Conversions in Erroneous Function Argument Swapping in C++, *20th International Working Conference on Source Code Analysis and Manipulation (SCAM)*, 2020, pp. 203-214. ⇒336

[59] J. Tan, D. Feitosa, P. Avgeriou, An empirical study on self-fixed technical debt, *In Proceedings of the 3rd International Conference on Technical Debt (TechDebt '20)*, 2020, ACM, New York, NY, USA, 11–20. ⇒350

[60] A. Tornhill, Prioritize Technical Debt in Large-Scale Systems Using CodeScene, *2018 IEEE/ACM International Conference on Technical Debt (TechDebt)*, 2018, pp. 59-60. ⇒348

[61] P. Wang, J. Yang, L. Tan, R. Kroeger, J. David Morgenthaler, Generating precise dependencies for large software, *4th International Workshop on Managing Technical Debt (MTD '13)*, 2013, pp. 47-50. ⇒345

[62] N. Zazworka, C. Seaman, F. Shull, Prioritizing design debt investment opportunities, *In Proceedings of the 2nd Workshop on Managing Technical Debt (MTD '11)*, 2011, pp. 39–42. ⇒344

[63] ∗ ∗ ∗, *MTD '11: Proceedings of the 2nd Workshop on Managing Technical Debt*, 2011, ACM, New York, NY, USA. ⇒337, 338, 341, 343

[64] ∗ ∗ ∗, *MTD '12: Proceedings of the Third International Workshop on Managing Technical Debt*, 2012, IEEE Press ⇒337, 338, 341, 343, 344

[65] ∗ ∗ ∗, *MTD '13: Proceedings of the 4th International Workshop on Managing Technical Debt*, 2013, IEEE Press. ⇒337, 338, 341, 343, 344

[66] ∗ ∗ ∗, *MTD '14: Proceedings of the 2014 Sixth International Workshop on Managing Technical Debt*, 2014, IEEE Computer Society. ⇒ 337, 338, 341, 343, 345

[67] ∗ ∗ ∗, *2015 IEEE 7th International Workshop on Managing Technical Debt (MTD)*, 2015, IEEE Computer Society. ⇒337, 338, 341, 343, 346

[68] ∗ ∗ ∗, *2016 IEEE 8th International Workshop on Managing Technical Debt (MTD)*, 2016, IEEE Computer Society. ⇒337, 338, 341, 343, 347
[69] ∗∗∗, *2018 IEEE/ACM International Conference on Technical Debt (TechDebt)*, 2018, IEEE Computer Society. ⇒337, 338, 341, 343, 347
[70] ∗∗∗, *2019 IEEE/ACM International Conference on Technical Debt (TechDebt)*, 2019, IEEE Computer Society. ⇒337, 338, 341, 343, 348
[71] ∗∗∗, *TechDebt '20: Proceedings of the 3rd International Conference on Technical Debt*, 2020, ACM, New York, NY, USA. ⇒337, 338, 341, 343, 350
[72] ∗∗∗, *2021 IEEE/ACM International Conference on Technical Debt (TechDebt)*, 2021, IEEE Computer Society. ⇒337, 338, 341, 343, 350

Received: October 28, 2021 • Revised: December 14, 2021