INVESTIGATE,IDENTIFY AND ESTIMATE THE TECHNICAL DEBT: A SYSTEMATIC MAPPING STUDY

Mrwan BenIdris, Hany Ammar, Dale Dzielski

West Virginia University– Morgantown, West Virginia USA

Mobenidris@mix.wvu.edu, Hany.Ammar@mail.wvu.edu, Dale.Dzielski@mail.wvu.edu

ABSTRACT

Context: Technical Debt (TD) is a metaphor that refers to short-term solutions in software development that may affect the cost to the software development life cycle. Objective: To explore and understand TDrelated to the software industry as well as an overview on the current state of TD research. Forty-three TD empirical studies were collected for classification and analyzation. Goals: Classify TD types, find the indicators used to detect TD, find the estimators used to quantify the TD, evaluate how researchers investigate TD. Method: By performing a systematic mapping study to identify and analyze the TD empirical studies which published between 2014 and 2017. Results: We present the most common indicators and evaluators to identify and evaluate the TD, and we gathered thirteen types of TD. We showed some ways to investigate the TD, and used tools in the selected studies. Conclusion: The outcome of our systematic mapping study can help researchers to identify interestand future in TD.

KEYWORD

Software Engineering, Technical Debt, Systematic Mapping Review, Technical Debt Indicator and Estimator.

1. INTRODUCTION

In 1992, Ward Cunningham was the first one who introduced the concept of TD. He said that "*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*" [22]. Software developers often face the challenge of delivering software products under tight schedules while trying to keep the quality up to standard [19]. If the developers focus only on requirements and take shortcuts, TD can occur. The shortcut taken by a decision reflect the TD metaphor and this decision may affect the software in the long-term. However, many types of TD can occur during the software lifecycle [1]. To identify a particular type of TD, it is necessary to know the causes of that typeof TD and the indicator[8]. Before we manage the TD, the first is to identify it in the software [7].

In many empirical studies selected in this study, the source code was used to identify the TD. However, the code comments also help developers to understand the code source [11, 12]. That reason leads some researchers to use code comments to identify Self-Admitted Technical Debt (SATD) [8, 9, 10]. In these 43 empirical studies, approximately 48 tools were used. Although

many of these tools are open source, some researchers develop their own approaches, method, and tools to identify, evaluate and manage TD [13, 14, 15, 16, 17].

To calculate TD we need to estimate the cost of fixing the debt (*principal*) and the future cost if the debt is not fixed (*interest*). There is no exact way to estimate the TD. However, TD Principal is calculated as a function of three variables: the number of the problems needed to solve, the time and the cost required to fix each problem [18]. And according to Alves et al. [4], the principal of TD is related to the effort and accompanying cost to eliminate the debt from a given system or artifact. In S18, the writers built a model based on metrics that can predict the monetized assessments (monetized = *Principle*). In S6, the authors measured the Principal of Defect Debt and employed KNN regression for predicting the value of fixing time (principal). S16 introduced a method called Goal-Questions-Metrics (GQM) which helps in quantifying the TD coupled with a specific release in order to be able to give more accurate release dates. The *interest* is inherently difficult to estimate or measure [20]. In S31, the developer activity logs were used as a measure of program comprehension effort (an indicator of TD interest payments). In S8, the authors estimated the TD breaking point (Accumulated interest, at some point, becomes greater than the effort required to repay the initial amount of TD) to support decision making.

Many tools, irrespective of their use of metrics, are available as open source to compute TD. An example is SonarQube, which uses the SQALE method for TD estimation. In the selected empirical studies around 48 tools were used. However, some researchers preferred to develop their own tools. For instance, in S28 the authors develop a tool called Designite. According to the developers, "Designite is a software design quality assessment tool that detects a wide variety of design smells and computes various metrics at different granularities. It provides a simple and interactive implementation of DSM (Dependency Structure Matrix) to help analyze the dependencies among the source code entities" [14].

2. RELATED WORK

Despite the fact that TD is a concerned research area, we found just five mapping studies or literature reviews related to TD. Li et al. [5] published a systematic mapping study that collected 75 TD studies which were published between 1992 and 2013. After they classified and analyzed these studies, they found ten TD types and eight TD management activities such as TD identification, repayment, prevention, communication, and monitoring. They also collected twenty-nine TD management tools.

Tom et al. [3] gathered the TD studies that published before 2011 [5] and performed semi-structured interviews in parallel with a multivocal literature review by focusing on the dimensions and causes of TD. Besides that, the authors concentrated on the benefits and drawbacks of allowing TD. The paper was published in 2013.

In 2015, a related systematic literature review was conducted by Ampatzoglou et al. [2]. The authors focused on the financial aspects of TD. They found that *principal* and *interest* are the most common financial terms used in TD research. They also found that the real options, portfolio management, cost/benefit analysis and value-based analysis are the most frequently financial strategies applied. They also presented a glossary of terms and a classification scheme for financial approaches used for managing TD.

In 2016, systematic mapping study was conducted to investigate strategies proposed to identify and manage TD in the software lifecycle by Alves et. al [4]. They focused on detecting the TD by using different types and valuable indicators. The authors collected 100 studies which were

published between 2010 and 2014 and provided a list of TD types, proposed TD manager, and software visualization techniques to identify and manage the TD.

In 2017, by concentrating on the Architecture Technical Debt (ADT) regarding principal, interest and management, Besker el at [6] gathered the TD studies that published before December 2015. The authors developed a novel ATD model for managing and raising awareness about ATD. The model explores different aspects and relationships to illustrate ATD in a unified and comprehensive way.

There are currently no systematic mappings focused on studying TD in the empirical studies between 2014 and 2017. Consequently, in our study, we collected 43 empirical studies from four databases to investigate and analyze them. The studies were published between 2014 and 2017

3. GOALS AND RESEAREACH QUETIONS

The purpose of our study is to explore and understand TD by identifying and analyzing the empirical studies published from January 2014 to December 2017 in TD. We plan to conduct a systematic mapping study of TD by systematically reviewing TD empirical studies published. Based on our research goal, we formulated four research questions

RQ1: What are the types of TD in the selected studies?
RQ2: How did researchers investigate the TD in the selected studies?
RQ3: What are the methods used to detect and estimate the TD in the selected studies?
RQ4: What are the tools used in the selected studies and which tools are the most commonly used by researchers?

4. SYSTEMATIC MAPPING EXECUTION

In order to get an overview of the research on TD, a systematic mapping study is carried out. In this study, six steps are performed to search for publications.

1. We determined the scope of this study. The four electronic Databases: IEEE Xplore, ACM Digital Library, Springer Link, Science Direct and the period: from January 2014 to December 2017 is our study scope.

2. We used “technical debt” as a search string in the abstract to find the publication that related to the TD.

3. We applied the exclusion criteria which is shown in Table 1.

Table 1:The Exclusion Criteria

#	*The exclusion criteria*
1.	*Any paper's full text is not available. (such as abstracts)*
2.	*Any publication that is not English.*
3.	*Panel summary publication.*

4.We merged the results from the four databases and removed the duplicates.

5.We filtered the publications by reading the abstract, and we excluded any unrelated paper or any study that dependet on just the developer or student such as surveys. Included, however, were articles that generated debate regarding inclusion amongst the authors of this review.

6.After we read the full text for all publications, the final result was 43 empirical studies.

5. EMPIRICAL STUDIES METADATA

Number of selected studies by digital library

As shown in Figure 1, the majority of the chosen empirical studies came from IEEE Xplore DB. One reason is there are five publications duplicated between ACM and IEE Xplore, so we deleted them from ACM and kept them with IEEE Xplore.

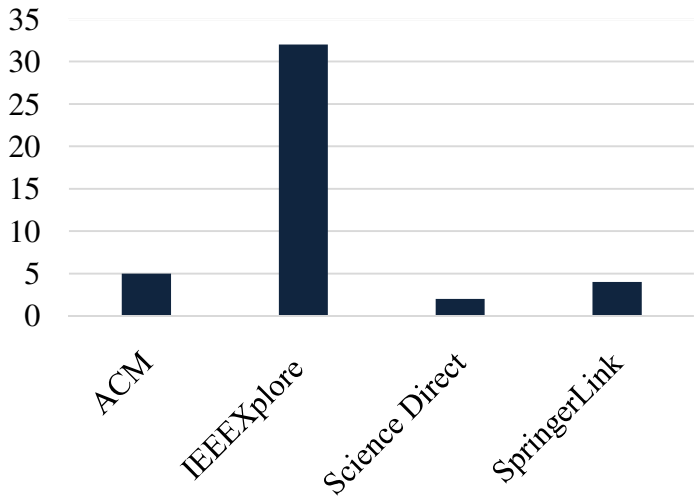


Fig. 1. Distribution of the empirical studiesby digital library

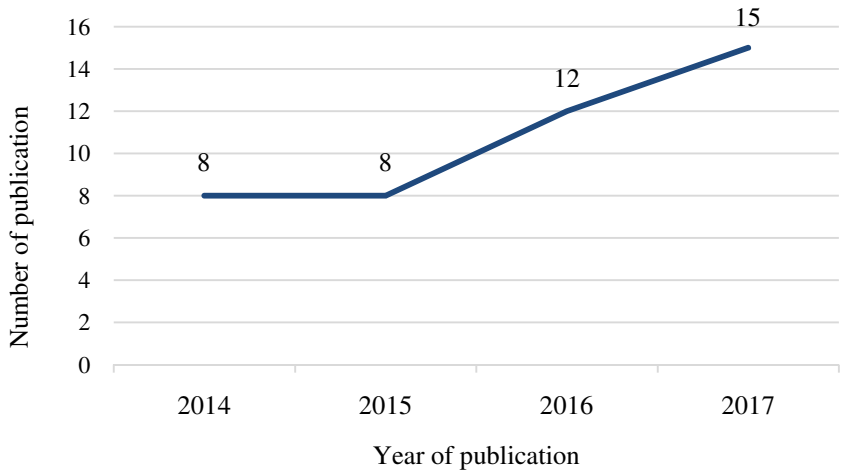


Fig. 2. Distribution of the empirical studies by year of publication

Number of selected studies by publication year.

As presented in Figure 2, the total number of selected empirical studies have nearly doubled from 2014 to 2016

Number of selected studies by number of citation

As shown in Figure 3, study number 43 (S43) has around 55 citations in four years, but study number 17 (S17) has 44 citations in just three years.

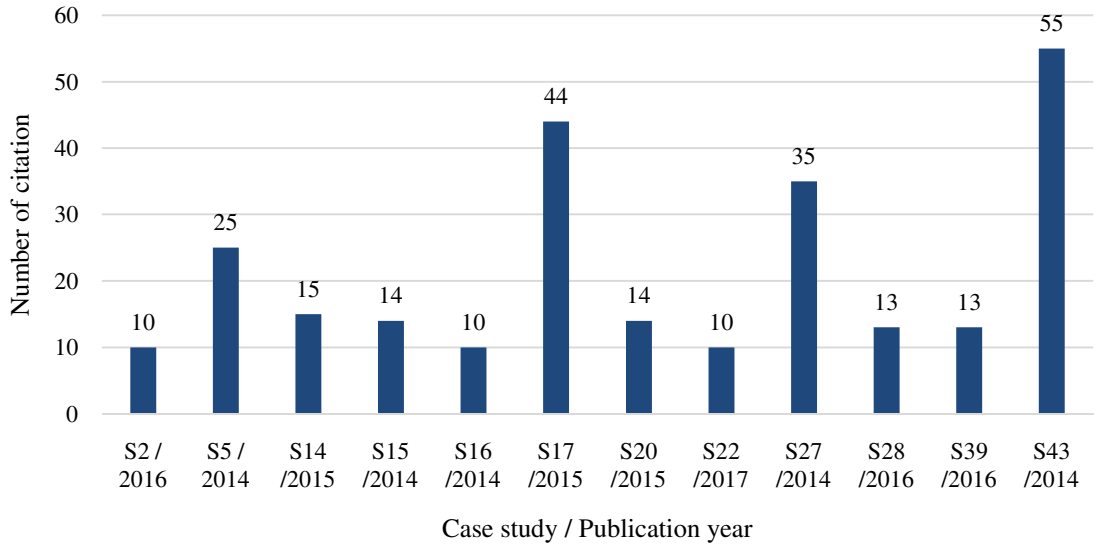


Fig. 3. Distribution of the empirical studies by number of citation (> 9 citations)

6. CLASSIFICATION SCHEME

We have five categories of classification that will allow for a better understanding of the TD and help us to answer four research questions: TD types, researchers' investigation, TD indicator, TD evaluator, and used Tools.

6.1. Empirical studies Classification by type of TD

A large number of TD types were selected for the selected empirical studies. We found 13 types. However, as shown in Figure 4 Code Debt, Design Debt and Architecture Debt were the most frequent types in these studies.

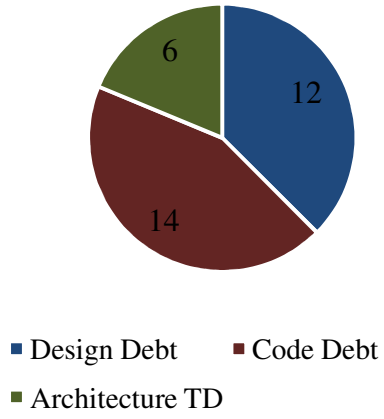


Fig. 4. Empirical studies Classification by type of TD

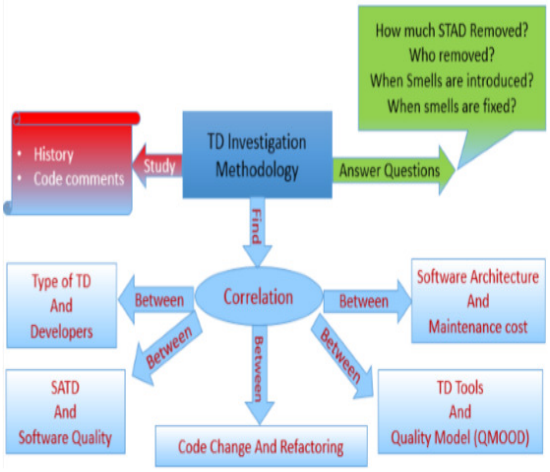


Fig. 5. Classification the studies by researchers' investigation

6.2 Empirical studies Classification by researchers' investigation

Figure 5 shows what the researchers tried to investigate in these selected case studies. The authors explored how much SATD is removed and who removed it. They also investigated the relationship between SATD and software quality. They examine the relationship between the quality model (QMOOD) and the different TD tools. They also find the correlation between software architecture and maintenance cost. They also study the the software history.

6.3. Empirical studies Classification by TD Estimator

As shown in Figure 6, the effort was the most-common estimator and was used by 6 empirical studies. The Violation and the Quality was the second most estimator used.

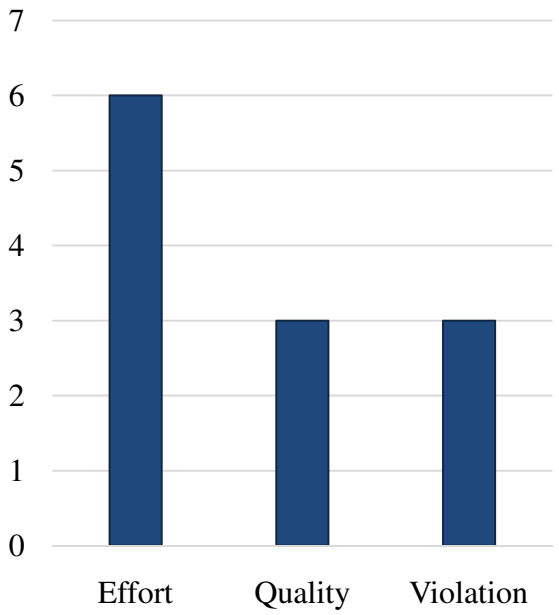


Fig. 6. The most used TD Estimators

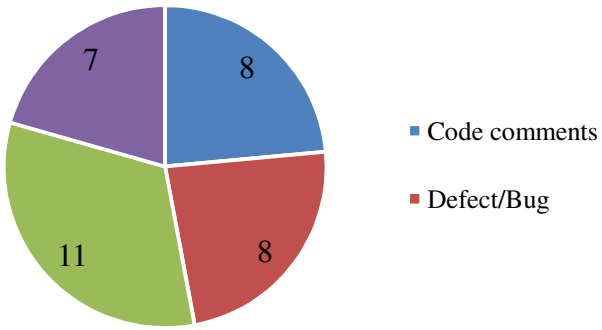


Fig. 7. The most used TD Indicators

6.4 Empirical studies Classification by TD Indicator

Figure 7 presents the TD indicators. It shows that the most used TD indicator is Smells and the second most commonly used one are Code Comments and Defect/Bug.

6.5 Empirical studies Classification by amount of used tools

Many tools were used in the selected studies, but some tools are not for TD. However, as shown in Figure 8, SonarQube was the most used TD tool from 48 observed tools which used in the selected empirical studies. Despite that many of these tools are open source, some researchers developed their own tools. Many empirical studies used one tool, but some studies used more than two tools. For instance, S15 used five tools which are FindBugs, Infusion, PMD, SonarQube and Understand. Table 2 presents details about empirical studies that used more than two TD tools presented on Table 2.

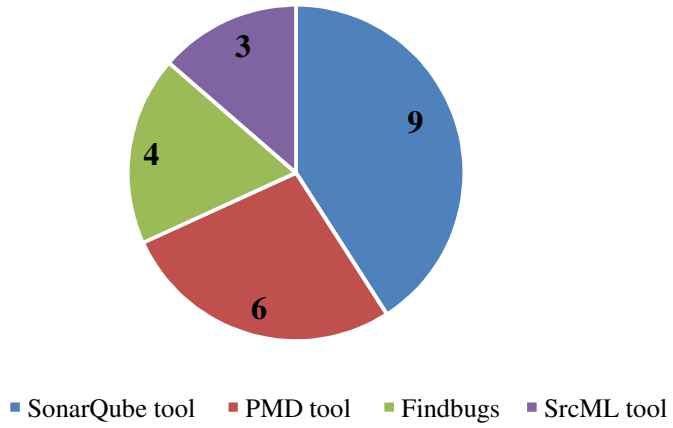


Fig. 8. The most used tools

Table 2 : studies used more than 2 technical debt tools

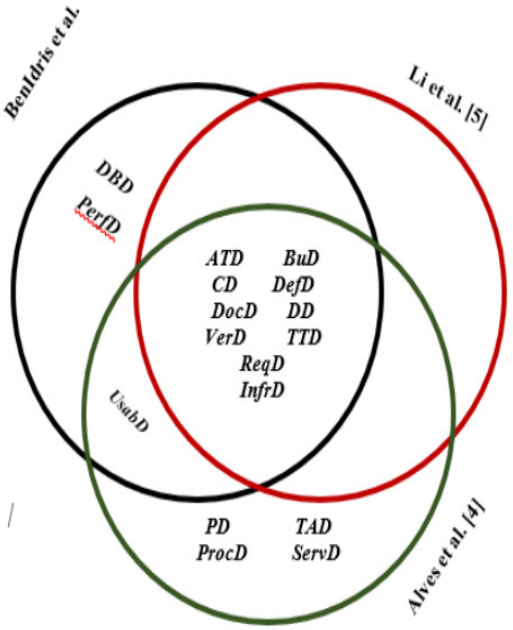
study #	Tools
S15	*FindBugs*, *Infusion*, *PMD*, *SonarQube* and *Understand*
S30	*CheckStyle*, *CodeProAnahtiX*, *FindBugs*, *JoCoCo*, *PMD*, *SonarQube*, and *Squale*
S37	*CheckStyle*,PMD, and *TEDIOUS*
S28	*PMD*, *Pppet-Lint*, and *Puppeteer*
S3	*Coverity*,*FindBugs*,*FxCop*,*GDIHandle*,*GDIView*, *IBMRationalPurify* , and *Perfmon*
S40	*Seerenemeaurement*, *SonarQube*, and *Understand*
S43	*CLIO*, *Codewizard*, and *FindBugs*

7. RESULTS AND COMPARISON

In this section, we present our results and compare them with the other five related studies to determine if there is overlap between the results.

TD TYPES (RQ1)

We discovered around 13 types of TD which are: Architecture Debt, Build Debt, Code Debt, Design Debt, Defect Debt, Database Debt, Documentation Debt, Infrastructure Debt, Performance Debt, Requirement Debt, Test Debt, Usability Debt, and Versioning Debt.
In comparison, Li et al. [5] found ten TD types, while Alves et al. [4] found 15 kinds of TD. Figure 9 illustrates the interference results for Li, Alves and our results. The key of terms of the TD shortcut is presented in Table 3.



#	Shortcut	Shortcut meaning
1	DBD	Database Debt
2	PerfD	Performance Debt
3	UsabD	Usability Debt
4	ATD	Architecture Technical Debt
5	CD	Code Debt
6	DocD	Documentation Debt
7	VerD	Versioning Debt
8	BuD	Build Debt
9	DefD	Defect Debt
10	DD	Design Debt
11	TTD	Test Technical Debt
12	ReqD	Requirement Debt
13	InfrD	Infrastructure Debt
14	TAD	Test Automation Debt
15	PD	People Debt
16	ProcD	Process Debt
17	ServD	Service Debt

Fig. 9. The interference results for Li, Alves and our results

TD INVESTIGATORS (RQ2)

In our study, we found that the researchers investigate the TD in various ways. They investigated the distribution of the TD among the developers then tried to find the types of violations caused by each developer. They also looked at the relationship between developers' maturity and the tendency to accumulate TD. Others investigated the characteristics of the project, quality model, removed debt, code changes, smells, and project history. Li classified the Technical Dept Type first as a tree. Each type was categorized into sub-types based on the causes of the TD type. These sub-types are what were investigated in Li's study. However, many investigators that were mentioned in Li were also similar to ours.

TD INDICATORS AND ESTIMATORS (RQ3)

Alves et. al [4] and we found that Smells was the most commonly used indicator. We found Code Comments and Defect/Bug was the second most commonly used while Alves found the

Automatic Static Analysis (ASA) and Software Architecture Issues was the second most commonly used.

We found that some authors developed prediction tool to measure the principle. Others used linear regression or real data. S15 used three estimators which are Detected Violation, Quality and equation based on Gaudin[20].

However, the most common estimator was Effort. Violation and Quality were the second most common estimator. Ampatzoglou et al. [2] mentioned in "Appendix B." also identified Effort as the first estimator.

USED TOOLS (RQ4)

In our study, we found around 48 tools used by these 43 empirical studies. SonarQube was the most used TD tool. Ampatzoglou et al. [2] mentioned about seven tools while Li et al. [5] found 29 tools for TD.

8. THREATS TO VALIDITY

In this section, we address possible threats to the validity that may be affected our Systematic Mapping Study. Our research questions do not cover all TD area. Our domain cover just four years (2014 - 2017) and four databases. This domain explicitly does not include all the TD empirical studies. Our search string, "Technical Debt", in the title, abstract, or keywords obviously excluded studies that do not use the term of "technical debt" explicitly but use the name of a specific form of technical debt, such as, design debt, code smells debt [5].

9. CONCLUSIONS

In this mapping study, we examined relevant studies in four databases and selected 43 empirical studies. Our objective was to find the Technical Debt Types, discover the TD indicators and estimators, and recognize the methods and tools used to investigate, indicate, and quantify Technical Debt from the selected empirical studies. In this study, our research questions' answers were collected. This data was summarized and analyzed to draw our main conclusions which are compressed in the following points: (1) The number of the published TD empirical studies has been significantly increasing from 2014 to 2017; (2) Forty-eight tools were identified from the selected empirical studies, and SonarQube was the most used tool; (3) In some empirical studies, the authors used more than three tools to investigate the TD. Others develop new tools and compared their results to open tools; (4) Special attention was paid to study SATD throughout the code comments; (5) Smells was the most applied as indicators of Technical Debt.

REFERENCES

[1] N. S. R. Alves, L. F. Ribeiro, V. Caires, T. S. Mendes, and R. O Spínola, "Towards an Ontology of Terms on Technical Debt." in 6rd International Workshop on Managing Technical Debt, Victoria, BC, Canada, 2014. DOI: 10.1109/MTD.2014.9.

[2] A. Ampatzoglou, A. Chatzigeorgiou, and P. Avgeriou, "The financial aspect of managing technical debt: A systematic literature review," Inf. Softw. Technol., vol. 64, pp. 52–73, Aug. 2015.

[3] Tom, Edith, AybüKe Aurum, and Richard Vidgen. "An exploration of technical debt." Journal of Systems and Software 86.6 (2013): 1498-1516.

[4] Alves, Nicolli SR, et al. "Identification and management of technical debt: A systematic mapping study." Information and Software Technology 70 (2016): 100-121.

[5] Z. Li, P. Avgeriou, P. Liang, "A systematic mapping study on technical debt and its management", J. Syst. Softw., Elsev. 101 (2015) 193–220.

[6] Besker, Terese, Antonio Martini, and Jan Bosch. "Managing architectural technical debt: A unified model and systematic literature review." Journal of Systems and Software 135 (2018): 1-16.

[7] Guo, Y., Spínola, R.O., Seaman, C.: Exploring the costs of technical debt management – a case study. Empir. Softw. Eng. 1, 1–24 (2014)

[8] Bavota, Gabriele, and Barbara Russo. "A large-scale empirical study on self-admitted technical debt." Mining Software Repositories (MSR), 2016 IEEE/ACM 13th Working Conference on. IEEE, 2016.

[9] A. Potdar and E. Shihab, "An Exploratory Study on Self-Admitted Technical Debt," 2014 IEEE Int. Conf. Softw. Maint. Evol., pp. 91– 100, 2014.

[10] de Freitas Farias, Mário André, et al. "A Contextualized Vocabulary Model for identifying technical debt on code comments." Managing Technical Debt (MTD), 2015 IEEE 7th International Workshop on. IEEE, 2015.

[11] S. C. B. Souza, N. Anquetil, and K. M. Oliveira, "Which documentation for software maintenance?," J. Brazilian Comput. Soc., vol. 12, no. 3, pp. 31–44, 2006.

[12] Steidl, Daniela, Benjamin Hummel, and Elmar Juergens. "Quality analysis of source code comments." Program Comprehension (ICPC), 2013 IEEE 21st International Conference on. IEEE, 2013.

[13] Martini, Antonio, and Jan Bosch. "An empirically developed method to aid decisions on architectural technical debt refactoring: AnaConDebt." Software Engineering Companion (ICSE-C), IEEE/ACM International Conference on. IEEE, 2016.

[14] Sharma, Tushar, Marios Fragkoulis, and Diomidis Spinellis. "Does your configuration code smell?." Mining Software Repositories (MSR), 2016 IEEE/ACM 13th Working Conference on. IEEE, 2016.

[15] Sharma, Tushar, Pratibha Mishra, and Rohit Tiwari. "Designite-A Software Design Quality Assessment Tool." Bringing Architectural Design Thinking Into Developers' Daily Activities (BRIDGE), IEEE/ACM International Workshop on. IEEE, 2016

[16] Trumler, Wolfgang, and Frances Paulisch. "How "Specification by Example" and Test-Driven Development Help to Avoid Technial Debt." Managing Technical Debt (MTD), 2016 IEEE 8th International Workshop on. IEEE, 2016.

[17] Ciolkowski, Marcus, et al. "Lessons Learned from the ProDebt Research Project on Planning Technical Debt Strategically." International Conference on Product-Focused Software Process Improvement. Springer, Cham, 2017.

[18] B. Curtis, J. Sappidi, and A. Szynkarski, "Estimating the size, cost, and types of Technical Debt," 3rd International Workshop on Managing Technical Debt (MTD '12), Zurich, Switzerland, 2012, pp. 49 – 53

[19] Guo, Yuepu, Rodrigo Oliveira Spínola, and Carolyn Seaman. "Exploring the costs of technical debt management–a case study." Empirical Software Engineering 21.1 (2016): 159-182.

[20] Zazworka, Nico, et al. "Comparing four approaches for technical debt identification." Software Quality Journal 22.3 (2014): 403-426.

[21] O. Gaudin, "Evaluate your technical debt with Sonar," Sonar, Jun, 2009.

[22] Cunningham, Ward. "The WyCash portfolio management system." ACM SIGPLAN OOPS Messenger 4.2 (1993): 29-30.

APPENDIX

[S1] Amanatidis, Theodoros, et al. "Who is Producing More Technical Debt? A Personalized Assessment of TD Principal." (2017).

[S2] Bavota, Gabriele, and Barbara Russo. "A large-scale empirical study on self-admitted technical debt." Proceedings of the 13th International Conference on Mining Software Repositories. ACM, 2016.

[S4] Gupta, Rajeev Kumar, et al. "Pragmatic Approach for Managing Technical Debt in Legacy Software Project." Proceedings of the 9th India Software Engineering Conference. ACM, 2016.

[S5] He, Xiao, et al. "Technical debt in MDE: a case study on GMF/EMF-based projects." Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems. ACM, 2016.

[S6] Akbarinasaji, Shirin, Ayse Basar Bener, and Atakan Erdem. "Measuring the principal of defect debt." Proceedings of the 5th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering. ACM, 2016.

[S7] Ampatzoglou, Areti, et al. "The Perception of Technical Debt in the Embedded Systems Domain: An Industrial Case Study." Managing Technical Debt (MTD), 2016 IEEE 8th International Workshop on. IEEE, 2016.

[S8] Chatzigeorgiou, Alexander, et al. "Estimating the breaking point for technical debt." Managing Technical Debt (MTD), 2015 IEEE 7th International Workshop on. IEEE, 2015.

[S9] De Freitas Farias, Mário André, et al. "A Contextualized Vocabulary Model for identifying technical debt on code comments." Managing Technical Debt (MTD), 2015 IEEE 7th International Workshop on. IEEE, 2015.

[S10] De Jesus, Jandisson Soares, and Ana Cristina Vieira de Melo. "Technical Debt and the Software Project Characteristics. A Repository-Based Exploratory Analysis." Business Informatics (CBI), 2017 IEEE 19th Conference on. Vol. 1. IEEE, 2017.

[S11] Delplanque, Julien, et al. "CodeCritics applied to database schema: Challenges and first results." Software Analysis, Evolution and Reengineering (SANER), 2017 IEEE 24th International Conference on. IEEE, 2017.

[S12] Eliasson, Ulf, et al. "Identifying and visualizing Architectural Debt and its efficiency interest in the automotive domain: A case study." Managing Technical Debt (MTD), 2015 IEEE 7th International Workshop on. IEEE, 2015.

[S13] Falessi, Davide, Barbara Russo, and Kathleen Mullen. "What if I Had No Smells?." Empirical Software Engineering and Measurement (ESEM), 2017 ACM/IEEE International Symposium on. IEEE, 2017.

[S14] Fontana, Francesca Arcelli, et al. "Towards a prioritization of code debt: A code smell intensity index." Managing Technical Debt (MTD), 2015 IEEE 7th International Workshop on. IEEE, 2015.

[S15] Griffith, Isaac, et al. "The correspondence between software quality models and technical debt estimation approaches." Managing Technical Debt (MTD), 2014 Sixth International Workshop on. IEEE, 2014.

[S16] Ho, Trong Tan, and Guenther Ruhe. "When-to-release decisions in consideration of technical debt." Managing Technical Debt (MTD), 2014 Sixth International Workshop on. IEEE, 2014.

[S17] Kazman, Rick, et al. "A case study in locating the architectural roots of technical debt." Proceedings of the 37th International Conference on Software Engineering-Volume 2. IEEE Press, 2015.

[S18] Kosti, Makrina Viola, et al. "Technical Debt Principal Assessment Through Structural Metrics." Software Engineering and Advanced Applications (SEAA), 2017 43rd Euromicro Conference on. IEEE, 2017.

[S19] Ludwig, Jeremy, Steven Xu, and Frederick Webber. "Compiling Static Software Metrics for Reliability and Maintainability from GitHub Repositories."

[S20] Maldonado, Everton da S., and Emad Shihab. "Detecting and quantifying different types of self-admitted technical debt." Managing Technical Debt (MTD), 2015 IEEE 7th International Workshop on. IEEE, 2015.

[S21] Maldonado, Everton da S., et al. "An empirical study on the removal of self-admitted technical debt." Software Maintenance and Evolution (ICSME), 2017 IEEE International Conference on. IEEE, 2017.

[S22] Maldonado, Everton, Emad Shihab, and Nikolaos Tsantalis. "Using natural language processing to automatically detect self-admitted technical debt." IEEE Transactions on Software Engineering (2017).

[S23] Martini, Antonio, and Jan Bosch. "An empirically developed method to aid decisions on architectural technical debt refactoring: Anacondebt." Software Engineering Companion (ICSE-C), IEEE/ACM International Conference on. IEEE, 2016.

[S24] Mayr, Alois, Reinhold Plösch, and Christian Körner. "A benchmarking-based model for technical debt calculation." Quality Software (QSIC), 2014 14th International Conference on. IEEE, 2014.

[S25] Oliveira, Frederico, Alfredo Goldman, and Viviane Santos. "Managing technical debt in software projects using scrum: An action research." Agile Conference (AGILE), 2015. IEEE, 2015.

[S26] Palomba, Fabio, et al. "An exploratory study on the relationship between changes and refactoring." Proceedings of the 25th International Conference on Program Comprehension. IEEE Press, 2017.

[S27] Potdar, Aniket, and Emad Shihab. "An exploratory study on self-admitted technical debt." Software Maintenance and Evolution (ICSME), 2014 IEEE International Conference on. IEEE, 2014.

[S28] Sharma, Tushar, Marios Fragkoulis, and Diomidis Spinellis. "Does your configuration code smell?." Mining Software Repositories (MSR), 2016 IEEE/ACM 13th Working Conference on. IEEE, 2016.

[S29] Sharma, Tushar, Pratibha Mishra, and Rohit Tiwari. "Designite-A Software Design Quality Assessment Tool." Bringing Architectural Design Thinking Into Developers' Daily Activities (BRIDGE), IEEE/ACM International Workshop on. IEEE, 2016.

[S30] Siebra, Clauirton A., et al. "Applying metrics to identify and monitor technical debt items during software evolution." Software Reliability Engineering Workshops (ISSREW), 2014 IEEE International Symposium on. IEEE, 2014.

[S31] Singh, Vallary, et al. "A case study of program comprehension effort and technical debt estimations." Program Comprehension (ICPC), 2016 IEEE 24th International Conference on. IEEE, 2016.

[S32] Trumler, Wolfgang, and Frances Paulisch. "How "Specification by Example" and Test-Driven Development Help to Avoid Technial Debt." Managing Technical Debt (MTD), 2016 IEEE 8th International Workshop on. IEEE, 2016.

[S33] Tufano, Michele, et al. "When and Why Your Code Starts to Smell Bad (and Whether the Smells Go Away)." IEEE Transactions on Software Engineering (2017)

[S34] Venkatasubramanyam, Radhika D., Shrinath Gupta, and Umesh Uppili. "Assessing the Effectiveness of Static Analysis through Defect Correlation Analysis." Global Software Engineering (ICGSE), 2015 IEEE 10th International Conference on. IEEE, 2015.

[S35] Weber, Jens H., et al. "Managing technical debt in database schemas of critical software." Managing Technical Debt (MTD), 2014 Sixth International Workshop on. IEEE, 2014.

[S36] Wehaibi, Sultan, Emad Shihab, and Latifa Guerrouj. "Examining the impact of self-admitted technical debt on software quality." Software Analysis, Evolution, and Reengineering (SANER), 2016 IEEE 23rd International Conference on. Vol. 1. IEEE, 2016.

[S37] Zampetti, Fiorella, et al. "Recommending when Design Technical Debt Should be Self-Admitted." Software Maintenance and Evolution (ICSME), 2017 IEEE International Conference on. IEEE, 2017.

[S38] Amanatidis, Theodoros, Alexander Chatzigeorgiou, and Apostolos Ampatzoglou. "The relation between technical debt and corrective maintenance in PHP web applications." Information and Software Technology (2017).

[S39] MacCormack, Alan, and Daniel J. Sturtevant. "Technical debt and system architecture: the impact of coupling on defect-related activity." Journal of Systems and Software 120 (2016): 170-182.

[S40] Ciolkowski, Marcus, et al. "Lessons Learned from the ProDebt Research Project on Planning Technical Debt Strategically." International Conference on Product-Focused Software Process Improvement. Springer, Cham, 2017.

[S41] Digkas, Georgios, et al. "The Evolution of Technical Debt in the Apache Ecosystem." European Conference on Software Architecture. Springer, Cham, 2017.

[S42] Huang, Qiao, et al. "Identifying self-admitted technical debt in open source projects using text mining." Empirical Software Engineering (2017): 1-34.

[S43] Zazworka, Nico, et al. "Comparing four approaches for technical debt identification." Software Quality Journal 22.3 (2014): 403-426.