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

Mrwan BenIdris, Hany Ammar, Dale Dzielski

West Virginia University– Morgantown, West Virginia USA

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 TD related 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 interest and future in TD.

KEYWORD


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 type of 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].

DOI:10.5121/ijsea.2018.9501
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 \((monetized = Principal)\). 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 RESEARCH QUESTIONS

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

<table>
<thead>
<tr>
<th>#</th>
<th>The exclusion criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Any paper’s full text is not available (such as abstracts)</td>
</tr>
<tr>
<td>2.</td>
<td>Any publication that is not English.</td>
</tr>
<tr>
<td>3.</td>
<td>Panel summary publication.</td>
</tr>
</tbody>
</table>

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 depended 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.

![Distribution of the empirical studies by digital library](image)

Fig. 1. Distribution of the empirical studies by digital library
As presented in Figure 2, the total number of selected empirical studies have nearly doubled from 2014 to 2016.

**Number of selected studies by publication year.**

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.

**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.
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.

![Design Debt, Code Debt, Architecture TD](image)

**Fig. 4.** Empirical studies Classification by type of TD

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 software history.

![Classification the studies by researchers’ investigation](image)

**Fig. 5.** Classification the studies by researchers’ investigation

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.
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.

Table 2: studies used more than 2 technical debt tools

<table>
<thead>
<tr>
<th>study #</th>
<th>Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>S15</td>
<td>FindBugs, Infusion, PMD, SonarQube and Understand</td>
</tr>
<tr>
<td>S30</td>
<td>CheckStyle, CodePro.AnalitiX, FindBugs, JoCoCo, PMD, SonarQube, and Squale</td>
</tr>
<tr>
<td>S37</td>
<td>CheckStyle, PMD, and TEDIOUS</td>
</tr>
<tr>
<td>S28</td>
<td>PMD, Puppet-Lint, and Puppeteer</td>
</tr>
<tr>
<td>S3</td>
<td>Coverity, FindBugs, FxCop, GDIHandle, GDIView, IBM Rational Purify, and Perfmon</td>
</tr>
<tr>
<td>S40</td>
<td>Seemeasurement, SonarQube, and Understand</td>
</tr>
<tr>
<td>S43</td>
<td>CLIQ, CodeWizard, and FindBugs</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>#</th>
<th>Shortcut</th>
<th>Shortcut meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DD</td>
<td>Database Debt</td>
</tr>
<tr>
<td>2</td>
<td>PerD</td>
<td>Performance Debt</td>
</tr>
<tr>
<td>3</td>
<td>UsbD</td>
<td>Usability Debt</td>
</tr>
<tr>
<td>4</td>
<td>ATD</td>
<td>Architecture Technical Debt</td>
</tr>
<tr>
<td>5</td>
<td>CD</td>
<td>Code Debt</td>
</tr>
<tr>
<td>6</td>
<td>DocD</td>
<td>Documentation Debt</td>
</tr>
<tr>
<td>7</td>
<td>VerD</td>
<td>Versioning Debt</td>
</tr>
<tr>
<td>8</td>
<td>BuD</td>
<td>Build Debt</td>
</tr>
<tr>
<td>9</td>
<td>DefD</td>
<td>Defect Debt</td>
</tr>
<tr>
<td>10</td>
<td>DD</td>
<td>Design Debt</td>
</tr>
<tr>
<td>11</td>
<td>TTD</td>
<td>Test Technical Debt</td>
</tr>
<tr>
<td>12</td>
<td>ReqD</td>
<td>Requirement Debt</td>
</tr>
<tr>
<td>13</td>
<td>InfD</td>
<td>Infrastructure Debt</td>
</tr>
<tr>
<td>14</td>
<td>TAD</td>
<td>Test Automation Debt</td>
</tr>
<tr>
<td>15</td>
<td>PD</td>
<td>People Debt</td>
</tr>
<tr>
<td>16</td>
<td>ProcD</td>
<td>Process Debt</td>
</tr>
<tr>
<td>17</td>
<td>ServD</td>
<td>Service Debt</td>
</tr>
</tbody>
</table>

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 (20142017) 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


APPENDIX


