A Systematic Literature Review of Technical Debt Prioritization

Reem Alfayez
University of Southern California
King Saud University
alfayez@usc.edu

Wesam Alwehaibi
University of Southern California
walwehai@usc.edu

Robert Winn
University of Southern California
rwinn@usc.edu

Elaine Venson
University of Southern California
venson@usc.edu

Barry Boehm
University of Southern California
boehm@usc.edu

ABSTRACT

Repaying all technical debt (TD) present in a system may be unfeasible, as there is typically a shortage in the resources allocated for TD repayment. Therefore, TD prioritization is essential to best allocate such resources to determine which TD items are to be repaid first and which items are to be delayed until later releases. This study conducts a systematic literature review (SLR) to identify and analyze the currently researched TD prioritization approaches. The employed search strategy strove to achieve high completeness through the identification of a quasi-gold standard set, which was used to establish a search string to automatically retrieve papers from select research databases. The application of selection criteria, along with forward and backward snowballing, identified 24 TD prioritization approaches. The analysis of the identified approaches revealed a scarcity of approaches that account for cost, value, and resources constraint and a lack of industry evaluation. Furthermore, this SLR unveils potential gaps in the current TD prioritization research, which future research may explore.

CCS CONCEPTS

• **Software and its engineering → Maintaining software**.

KEYWORDS

software, technical debt, prioritization, software management, software maintenance

ACM Reference Format:
Reem Alfayez, Wesam Alwehaibi, Robert Winn, Elaine Venson, and Barry Boehm. 2020. A Systematic Literature Review of Technical Debt Prioritization . In *International Conference on Technical Debt (TechDebt '20), October 8–9, 2020, Bozeman, MT, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3387906.3388630

1 INTRODUCTION

Accumulating technical debt (TD) in a software system can result in an inflexible and uncontrollable system and a decrease in software team morale. Large amounts of TD may cause the inability to incorporate new features without compromising existing ones. Eventually, the developers of a system will not be able to make changes without the introduction, on average, of at least one bug [36]. A Stack Overflow survey found that 29.6% of participants considered a fragile code base as their primary challenge in the workplace [1]. A report published jointly by Evans Data Corp, CIA Factbook, and Stripe research revealed that an estimated worth of $85 billion in opportunity cost is lost, annually, worldwide as a result of developers' time spent on "bad code." Furthermore, the report found that the average developer spends more than 13 hours a week repaying TD. Nearly 80% of the participants believe that repaying TD hurts their morale, and over half of the participants believe that TD hinders their productivity. Lastly, nearly two-thirds of the participants called for a clear TD prioritization [2].

TD prioritization is the process of deciding which TD items are to be repaid first, and which items are to be delayed until later releases [33]. A naive approach would attempt to repay all TD existing in a system. However, this may be unviable, due to the usual limited allocated resources for TD repayment [10, 21, 25]. The scarcity in the allocated resources results from the innate nature of TD. Repaying TD does not necessarily have a direct impact on the external behavior of a system, and the effect is only limited to its internal quality [21]. As a result, both software and business teams prefer to spend available resources on implementing new features or fixing bugs [25]. With limited resources at their disposal, individuals may struggle to decide which TD items can be repaid to achieve the highest value, as there is typically a trade-off between the value achieved from repaying a TD item and its cost [25]. The complexity of TD prioritization grows exponentially with the number of TD items, as each item can be included or excluded from a repayment activity. This makes finding an optimal solution that maximizes value and satisfies resources constraint in large problems hard [28].

Despite these challenges, researchers have developed multiple TD prioritization approaches. Even though the software engineering community has put in a great effort into understanding the current status of TD and technical debt management (TDM), to the best of our knowledge, the existing literature relating to TD lacks a study that summarizes the current literature regarding TD prioritization and analyzes the aspects that this study considers. This study aims to analyze the existing TD prioritization approaches by the means of a systematic literature review (SLR) [29]. Specifically, this SLR aims to identify studies that propose TD prioritization approaches, summarize and evaluate prioritization techniques employed by such approaches, and synthesize the findings. The analysis involves identifying the steps required in each approach,

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
TechDebt '20, October 8–9, 2020, Bozeman, MT, USA
© 2020 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-7960-1/20/05…$15.00
https://doi.org/10.1145/3387906.3388630

the type of TD addressed, the decision factor categories considered, the essential artifacts, type of human involvement required, and how and in which setting each approach was evaluated. Additionally, the analysis identifies the prioritization techniques that these approaches are based on and point out some of their limitations.

Having a synthesised reference of TD prioritization is beneficial for practitioners and researchers alike. Practitioners can refer to this reference to gain a better understanding of the identified TD prioritization approaches. For example, they can identify which approaches address the TD type in hand and whether an approach accounts for resources constraint. The review can be utilized by researchers to identify gaps that the research community can contribute to fulfill. Additionally, the review highlights a few of the limitations within the current TD prioritization approaches, which individuals can seek to avoid when developing new approaches.

2 RESEARCH METHOD

This study aims to identify and analyze the current TD prioritization approaches and techniques through the means of a systematic literature review (SLR). The SLR's steps are detailed in this section.

2.1 Research Questions

This section presents the research questions that were formulated to achieve this SLR's goal. The research questions are as follows:

RQ1: What are the current TD prioritization approaches?
- **RQ1.1:** Which types of TD are addressed by the TD prioritization approaches?
- **RQ1.2:** What decision factors are the TD prioritization approaches based on?
- **RQ1.3:** What software artifacts do the TD prioritization approaches depend on?
- **RQ1.4:** What type of human involvement is required in the TD prioritization approaches?
- **RQ1.5:** How are the TD prioritization approaches evaluated?

RQ2: What prioritization techniques are utilized by the identified TD prioritization approaches?
- **RQ2.1:** What are the limitations of the identified prioritization techniques?

2.2 Search Strategy

The search strategy conducted in this SLR is a combination of a manual and automated search. The concept of a quasi-gold standard was employed to assess the completeness of the search, as proposed by Zhang et al. [57]. This dual search method improves the rigor of the search process [57]. The search strategy for both the manual and automated search was constrained to published literature from 1992 to 2018 because the TD metaphor was only introduced in 1992 [19], and the search was conducted in early 2019.

The International Workshop on Managing Technical Debt (MTD) and the International Conference on Technical Debt (TechDebt) were selected as basis venues for the manual search, due to their specialization in TD. The manual search and the application of the selection criteria, which will be defined in the following section, resulted in a set of 10 studies that constitutes the quasi-gold standard. The completeness of the search was evaluated through comparing studies in the quasi-gold standard set against studies identified by the automated search. This comparison was carried out through an application of the sensitivity and precision metrics [29, 57].

To design the search string, a subjective approach was followed [57]. Trial experiments were conducted to derive the string, and the candidate search string was evaluated against the quasi-gold standard. The experiment was begun using the term "debt". However, this produced an overwhelming number of results. The majority of these results reference "debt" in a financial or economic context. Consequently, the search string was tailored to "technical debt".

To evaluate the performance of the "technical debt" search string, an automated search was conducted on the two venues that were used in the manual search: MTD and TechDebt. From the automated search, a total number of 129 studies were retrieved. From these 129 studies, a total number of 10 relevant studies were identified by applying the selection criteria, which will be defined in the following section. The sensitivity and precision of the search string were then calculated. The "technical debt" search string resulted in 100% sensitivity and 7.75% precision. Though an ideal search string has 100% precision and 100% sensitivity, obtaining perfect values is challenging due to a trade-off between the two metrics. In the context of an SLR, a high sensitivity search string is more desirable than a high precision one [57]. Therefore, "technical debt" was selected, as the search string, since it has a perfect sensitivity.

Following the recommendations of [29, 57], the selected databases for the automated search are as follows: ACM Digital Library, Google Scholar, IEEE Xplore, Inspec, ScienceDirect, Scopus, Springer, and Web of Science. For Google Scholar, only the top 1,000 results were included. The automated search was performed using the "technical debt" search string in the eight databases listed above. This resulted in a total number of 2,613 papers.

2.3 Study Selection

This section summarizes the selection criteria established to identify relevant studies and the application procedure of these criteria.

2.3.1 Selection Criteria. The selection criteria were derived from these previous secondary studies: [9–12, 21, 33, 41]. This ensured that this SLR follows the standards of the TD community. The inclusion criteria was limited to one criterion: A study must propose a TD prioritization approach. A study will be disqualified if it meets any of the following exclusion criteria.

- Studies that are not published in English.
- Studies that were inaccessible (i.e., the full text was not available).
- Non-paper submissions, such as keynotes, or non-peer reviewed.
- Duplicates of an included study; the most comprehensive version reporting the study was selected.
- Studies published prior to 1992 or after 2018.
- Studies that decide between repaying TD items or implementing new features.
- Studies that prioritize the refactoring steps of a TD item.
- Studies that categorize TD items based on their types or relevancy to a specific software attribute.

2.3.2 Selection Procedure. Figure 1 summarizes the selection procedure. Omitting duplicates resulted in 1,550 papers that were independently reviewed by two authors by reading the title and the abstract of each study to determine its relevance to the subject

matter. The reading step was followed by a joint meeting between the two authors in which the results were discussed. Disagreements were resolved by independently reading the full papers and holding a discussion to compare the results. If the disagreement persists, a conservative approach was followed by including the paper under disagreement. These steps resulted in a 27 papers that were fully read independently by three authors. Afterward, discussions among the three authors were held to discuss the results. The discussions resulted in the exclusion of seven papers, due to a lack of well defined TD prioritization approaches. The resulting 20 papers were passed to the snowballing (i.e., citation analysis) step. Forward and backward snowballing were conducted on the selected set of papers, while following the guidelines in [54]. The aforementioned selection procedure was repeated to determine the relevancy of papers during the snowballing process. Two iterations of snowballing were conducted. The first iteration resulted in three papers. Performing snowballing on these three papers did not result in any new papers. Consequently, the search was concluded with a total of 23 papers.

Figure 1: Selection procedure summary

2.4 Data Extraction and Synthesis

The data extraction procedure involved the three authors, independently, reading the full papers and tabulating, in a spreadsheet, how each study addresses each research question. Data was validated by holding joint meetings to discuss the results. A thematic analysis [18] was employed to identify, analyze, and report patterns within the included studies. An integrative approach was applied to categorize the findings. Both a start list of categories (deductive approach) and the development of new categories along the way (inductive approach) were utilized, following the guidelines in [18]. The initial list of categories was derived from the research questions.

3 RESULTS

This section presents and discusses the findings of this SLR. The results are based on the set of 23 studies obtained using the aforementioned search and selection processes.

3.1 RQ1: What are the current TD prioritization approaches?

In this SLR, a TD prioritization approach refers to a systematic methodology, that is based on a prioritization technique, with the intent of prioritizing a group of TD items for repayment, revolving around a set of predefined goals. Table 1 summarizes the 23 selected studies. The interest toward TD prioritization prior to 2012 was little to none, but has significantly increased in recent years. The first approach to prioritize TD was only published as recently as 2011, and 34.78% of the studies were published in 2018.

A total of 24 TD prioritization approaches were identified in the 23 selected studies. This imbalance in the number of approaches and the number of studies is due to study S23 which proposes four approaches. One of these approaches has been discussed in more detail in a previous study, denoted by S9. Another one was detailed in a recent study, indicated as S2. Therefore, these duplicate approaches were included only once in the total count of TD prioritization approaches. For each study, we identified the proposed prioritization approach and summarized the specific process of each approach. Due to space limitation, we provide the summary in [3].

Table 1: Selected studies and their reference numbers

ID	Author (Year)	Reference
S1	Schmid (2013)	[43]
S2	Guo & Seaman (2011)	[24]
S3	Ribeiro et al (2017)	[40]
S4	Almeida et al (2018)	[20]
S5	Sae-Lim et al (2018)	[42]
S6	Snipes et al (2012)	[45]
S7	Plösch et al (2018)	[37]
S8	Guimaraes et al (2018)	[23]
S9	Guo et al (2016)	[25]
S10	Aldaeej & Seaman (2018)	[8]
S11	Vidal et al (2015)	[53]
S12	Letouzey & Ilkiewicz (2012)	[32]
S13	Choudhary & Singh (2016)	[15]
S14	Mensah et al (2018)	[34]
S15	Tornhill (2018)	[49]
S16	Zazworka et al (2011)	[56]
S17	Albarak & Bahsoon (2018)	[7]
S18	Codabux & Williams (2016)	[17]
S19	Akbarinasaji (2015)	[6]
S20	Fontana et al (2015)	[22]
S21	Harun & Lichter (2015)	[26]
S22	Abad & Ruhe (2015)	[4]
S23	Seaman et al (2012)	[44]

3.2 RQ1.1: Which types of TD are addressed by the TD prioritization approaches?

The identified approaches vary in terms of the TD types that they address. Of the identified approaches, 70.83% address a specific type of TD and 29.17% are applicable to any TD type. As Table 2 summarizes and Figure 2 displays, 33.33% of the approaches address code TD, 16.67% address design TD, and 12.50% address defect TD. The remaining four TD types, which are self-admitted technical debt (SATD), database normalization TD, requirement TD, and architectural TD, were each respectively only found to be addressed by one approach. The summary above may be helpful for practitioners when deciding which approach to use. Additionally, the summary can be used to assess the applicability of one of the approaches to a different TD type when developing new TD prioritization approaches.

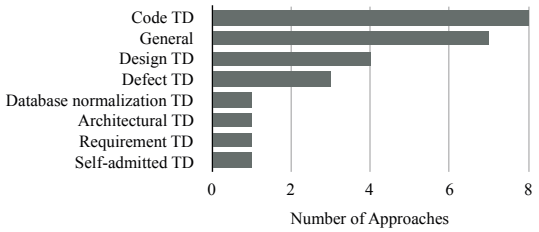

Figure 2: Breakdown of approaches based on TD type

Table 2: TD types addressed, decision factors considered, value estimation methods, and cost estimation methods

Study	Type	Value	Cost	Constraint	Value estimation (suggestions)	Cost estimation (suggestions)
S1	General	●	●	○	Not specified	Not specified
S2	General	●	●	●	Not specified (expert knowledge, historical data, program analysis)	Not specified (expert knowledge, historical data, program analysis)
S3	General	●	●	○	Expert knowledge	Expert knowledge
S4	General	●	○	○	Expert knowledge	N/A
S5	Code TD	●	○	○	Calculation model based on impact analysis using information retrieval, mining software repository, or dynamic analysis	N/A
S6	Defect TD	●	●	○	Expert knowledge	Expert knowledge
S7	Design TD	●	●	●	A combination of calculation model and expert knowledge	Expert knowledge
S8	Code TD	●	○	○	Voting criterion based on component, concern, and scenario criteria calculated using expert knowledge and JSpIRIT	N/A
S9	General	●	●	●	Not specified (any cost estimation such as cost estimation model or expert knowledge)	Not specified (any cost estimation such as cost estimation model or expert knowledge)
S10	Defect TD	●	●	○	Prediction model using historical data	Prediction model using historical data
S11	Code TD	●	●	○	JSpIRIT predefined criteria or practitioner's criteria	Not specified
S12	Code TD	●	●	○	Predefined quality characteristic based on expert knowledge	Calculation model based on expert knowledge
S13	Architectural TD	●	○	○	Calculation model based on historical data, architecture design, and severity of the class	N/A
S14	Self-admitted TD	●	●	○	Prediction model	Calculation model based on the prediction model result
S15	Code TD	●	○	○	Prediction model based on technical and social factors	N/A
S16	Design TD	●	●	○	Calculation model based on code repository and issue tracking system data	Calculation model based on the comparison of the calculated metrics to their respective thresholds
S17	Database normalization TD	●	○	○	Calculation model based on risk of data inconsistency and expected return from decreasing I/O cost	N/A
S18	Code TD and design TD	●	●	○	Prediction model result in addition to suggesting other potential criteria (expert knowledge, historical data, or dependency analysis)	Not specified (dependency analysis, expert knowledge, or historical data)
S19	Defect TD	●	●	●	Not specified	Not specified
S20	Code TD	●	○	○	Calculation model based on predefined software metrics	N/A
S21	Code TD and design TD	●	●	○	Calculation model based on change likelihood using historical data, defect likelihood using historical data, expert knowledge, and impact using dependency analysis	Calculation model using SonarQube, which is based on expert knowledge
S22	Requirement TD	●	●	○	Historical data	Not specified
S23:						
S23.1	General	●	●	○	Not specified	Not specified
S23.2	General	●	●	○	Not specified	Not specified

3.3 RQ1.2: What decision factors are the TD prioritization approaches based on?

There is a multitude of decision factors to consider when deciding whether to repay a TD item or delay it until next release. Ribeiro et al. [41] identified 14 unique decision factors that influence the decision of TD repayment. These decision factors were classified into three factor categories: value, cost, and resources constraint. Moreover, these categories are referred to as the overarching decision factors that are depended on by prioritization approaches. A TD prioritization approach may utilize one of the defined categories or a combination of them. Table 2 displays which decision factor categories are considered in each approach, where filled circles represent the considered factors. The analysis revealed that all the identified TD prioritization approaches consider value in the prioritization process. However, only 70.83% of the approaches consider cost. Additionally, a mere of 16.67% of the approaches consider resources constraint. As Figure 3 illustrates, 29.16% of the approaches rely solely on value, 54.17% of the approaches employ value and cost, and only 16.67% of the approaches utilize a combination of value, cost, and resources constraint. This indicated the lack of approaches that account for all three factors in TD prioritization.

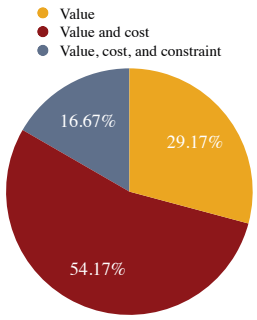

Figure 3: Approaches breakdown based on categories

Measuring value and cost varied among the identified approaches, as summarized in Table 2. The majority of the approaches (i.e., 70.83%) provided guidelines regarding value estimations. Of the approaches that specified value estimation methods, 35.29% rely on calculation models, 23.53% utilize prediction models, and 17.65% depend on experts' knowledge to estimate value. Additionally, 23.53% of the approaches utilize a combination of calculation models and experts' knowledge in their value estimation. A few of the TD prioritization approaches (i.e., 12.50 %) did not specify a value estimation method, but suggested several methods, such as experts' knowledge, that could be chosen at the practitioner's discretion, which are denoted in Table 2 in parentheses. Four (i.e., 16.67 %) of the approaches did not provide a specific method, nor any suggestions. In regards to cost, of the approaches that considered cost as a decision factor, 47.06% specified how to estimate cost. Of these approaches that specifically included how to estimate cost, the most applied cost estimation methods are as follows: experts' knowledge (37.50 %), a combination of calculation models and experts' knowledge (25.0 %), calculation models (12.50 %), prediction models (12.50 %), and a combination of calculation and prediction models (12.50 %). Additionally, 17.65 % of the approaches that considered cost did not provide what specific method to utilize when estimating cost and only provided some suggestions. On the contrary, 35.29% of these approaches did not provide any guidelines regarding cost estimation. The lack of concise methods to estimate value and cost of TD items suggests that the value and cost of a TD item are context-dependent, which was also noted in [9, 10, 12, 21, 33, 41].

3.4 RQ1.3: What software artifacts do the TD prioritization approaches depend on?

Software artifacts provide a wealth of information that can aid in TD prioritization [23, 42]. In the context of this SLR, a software artifact is considered indispensable to an approach if the approach will not work without the presence of the software artifact. For example, assume that a study's TD prioritization approach requires the value of each TD item and that the study suggested using an issue tracking system or a static analysis tool to estimate the value. In this case, it is concluded that the study's TD prioritization approach does not have any software artifact dependencies, as any value estimation method will suffice. However, assume now that a given study's approach requires the cost of each TD item and that the study explicitly specifies that historical releases data is used to compute the cost of each TD item. Then historical releases data would be considered as an essential software artifact of which the provided approach requires to be successful in TD prioritization. Table 3 summarizes the software artifact dependencies of each TD prioritization approach. Many of the approaches (i.e., 66.67%) depend on a single software artifact, while some rely on multiple artifacts. 58.33% of the approaches are reliant on a system's source code to prioritize TD. Issue tracking systems are the second most common software artifact dependency, with 25.0% of the total TD prioritization approaches requiring it to work as intended. Additionally, 12.50% of the approaches make use of historical releases data. Code repository and multiple releases data are core requirements in 12.5% and 8.33% of the approaches, respectively. Furthermore, one approach requires the use of architecture blueprints and another requires database monitoring system. Identifying the software artifacts that are essential to each TD prioritization approach can aid practitioners in determining which approaches are suitable for their individual needs and available resources. For example, a practitioner might refrain from using an approach that requires multiple releases given that the system in hand is a one release system.

3.5 RQ1.4: What type of human involvement is required in the TD prioritization approaches?

Human involvement can provide invaluable insights during TD prioritization [23, 25, 42]. It may be essential to identify necessary information, which can be overlooked or impossible to measure automatically, such as understanding the business value of a TD item [25]. However, human involvement can also be burdening, tedious, and not necessary in some situations, such as identifying the dependencies of a TD item [23, 42]. As the number of TD items increases, the effort required to prioritize these items increases as well. Human involvement should be minimized, as much as possible, to lower the workload on the associated software engineers. Some of approaches identified in this SLR developed automated means to obtain TD item information and aid in the prioritization process. However, other approaches demand human involvement to determine TD items' priorities. The approaches were categorized based on the level of human involvement as: **None:** The approach is fully-automated and does not require any human intervention. **Minor:** The approach is semi-automated and only requires human involvement during the setup phase, such as defining thresholds and assigning weights to various criteria. Once the set up is complete, human involvement is not required, and the prioritization of TD can take place without a human's presence. **Major:** The approach depends on a human's continual presence to carry out the prioritization. If a human is absent, the approach will not work as intended. As Table 3 summarizes, most of the approaches (i.e., 41.67%) are fully-automated. Minor human involvement is required by 29.17% of the approaches. Similarly, 29.17% of the approaches require continual human participation during the prioritization. A practitioner might find this summary useful to anticipate the required human effort when utilizing an approach. Furthermore, researchers can utilize this summary to find opportunities to fully automate an approach such as utilizing a tool to identify dependencies instead of relying on humans.

3.6 RQ1.5: How are the TD prioritization approaches evaluated?

Evaluation is the process of measuring the effectiveness or performance of an approach in achieving its goals [16]. TD prioritization approaches might be challenging to evaluate objectively, due to the difficulty that lies within the replication of settings and the attempt to compare the effectiveness of a proposed approach to another [9, 21, 43]. Nonetheless, as Table 3 illustrates, 62.50% of the identified approaches were evaluated. The table also presents each evaluation method and its respective setting, in which the evaluation was conducted, in parentheses. 37.50% of the approaches were evaluated using a case study. Additionally, 20.83% of the approaches were evaluated through conducting an experiment, and only one

Table 3: Software artifact dependencies, human involvement level, and evaluation method for each approach

Study	Required software artifacts	Human involvement	Evaluation (setting)
S1	None	Minor	None
S2	None	None	None
S3	None	Major	Case study (academic)
S4	None	Major	Case study (industry)
S5	Issue tracking system and source code	Major	Experiment (industry)
S6	Source code	Major	Case study (industry)
S7	Source code	Minor	Case study (open-source)
S8	Architecture blueprints and source code	Major	Experiment (open-source)
S9	None	Major	Case study (industry)
S10	Historical releases data, issue tracking system, multiple releases data, and source code	None	None
S11	Source code	Minor	Case study (industry)
S12	Source code	Minor	Case study (industry)
S13	Historical releases data, multiple releases data, and source code	None	Experiment (open-source)
S14	Source code	Minor	Experiment (open-source)
S15	Historical releases data and source code	None	Case study (open-source)
S16	Code repository, issue tracking system, and source code	None	Feasibility study (industry)
S17	Database monitoring system	None	Case study (open-source)
S18	Code repository, issue tracking system, and source code	Minor	None
S19	Issue tracking system	None	None
S20	Source code	None	Experiment (open-source)
S21	Code repository, issue tracking system, and source code	Major	None
S22	None	None	None
S23:			
S23.1	None	Minor	None
S23.2	None	None	None

approach utilized a feasibility study to assess its applicability. The evaluations were conducted in various settings. Of the evaluated approaches, 46.67% were evaluated, respectively, in the industry and using Open Source Software (OSS) systems, and one approach was evaluated in an academic setting. Based on these results, it can be seen that some approaches were not evaluated by any means and other approaches were not evaluated in the industry. This lack of industry evaluations harms the ability to predict the performance of such approaches in an industry setting.

3.7 RQ2: What prioritization techniques are utilized by the identified TD prioritization approaches?

Each TD prioritization approach, while unique, is based on a core concept, which is defined as a prioritization technique. A prioritization technique is an overarching strategy with the aim of assigning values to distinct prioritization objects that allow the establishment of a relative order between said objects in the set [55]. In this SLR, the objects are the TD items needed to be prioritized. This research question aims to identify and define prioritization techniques, so that they can be utilized and built upon by the software engineering community during TD prioritization. When designing a new TD prioritization approach, individuals should begin by selecting which prioritization technique best fits their situation. Once a prioritization technique has been selected, a TD prioritization approach should be designed around the core ideals of the given technique.

By examining the selected studies, 10 unique prioritization techniques were identified. Table 4 displays these techniques, their definitions, their limitations, and studies that incorporated them. The most utilized techniques are as follows from greatest to least: Cost-Benefit Analysis (CBA), Ranking, Predictive Analytics, Real-Options Analysis (ROA), Analytic Hierarchy Process (AHP), Modern Portfolio Theory (MPT), Weighted Sum Model (WSM), Business Process Management (BPM), Reinforcement Learning, and Software Quality Assessment Based on Lifecycle Expectations (SQALE).

3.8 RQ2.1: What are the limitations of the identified prioritization techniques?

Each prioritization technique has its own drawbacks. Being aware of such drawbacks can be beneficial for practitioners and researchers alike. When deciding whether to utilize a given technique, one must be aware of the potential drawbacks of said technique to avoid potential consequences. Once such drawbacks have been considered, users can decide whether to continue with the technique or to switch to a prioritization technique more suited to their scenario. This research question aims to provide more background on the various existing prioritization techniques. The limitations of the existing prioritization techniques are provided in Table 4. It is important to note that the identified limitations are not comprehensive and there may exist other limitations for the identified prioritization techniques that were failed to be included since the list is based solely on the acquired literature pool. The identified limitations have been further categorized into groups, which can be seen in Table 4 in parentheses.

The categorization groups help identify which type of limitation is associated with which prioritization technique. In summary, these categories are defined as follows: **Applicability**: A few of the TD prioritization approaches revolve around the use of a few core concepts, which might be difficult to equate in a software engineering scope. For example, ROA is conducted through the use of a variety of measures and metrics, which are all not viable for use with TD [46]. ROA can be equated partly through the use of the liquidity of a given item. Calculating the liquidity and other similar factors of a TD item can be very difficult or not possible in some cases [46]. **Communication among stakeholders**: A few prioritization techniques are dependent on the ability of all relevant stakeholders to understand the importance and impact of each TD

Table 4: TD prioritization techniques, limitations, and corresponding studies

Technique	Description	Limitations	Studies
Analytic Hierarchy Process (AHP)	Considers multiple criteria and generates weights for each criterion based on pair-wise comparisons. A score is assigned, based on the pair-wise comparisons, to each criterion and TD item, which are then used to calculate the relative importance of each TD item [44]	• (Error Proneness & Scalability) Challenge in identifying ideal criteria and weights, and the quality of the results depends on these weights [52] • (Computational Complexity & Scalability) Time and effort consuming to complete pairwise comparisons in large hierarchies [52] • (Rank Updates) Difficulty in adding a new TD item as all the analysis must be redone [52] • (Rank Updates) Rank reversal may occur after the addition of an "inferior" alternative [30]	S18, S23.1
Business Process Management (BPM)	Utilizes BPM in TD prioritization by conducting focus groups and various interviews with technical and business stakeholders in order to account for the business perspective in TD prioritization [20]	• (Communication among Stakeholders) Depends on business and technical expert knowledge [20] • (Communication among Stakeholders) Lack of a mutual understanding between technical and business stakeholders can lead to issues that may influence and misdiagnose the level of urgency and criticality of a given TD item [20] • (Scalability) Can be time consuming for large amounts of TD [20]	S4
Cost-Benefit Analysis (CBA)	Assigns a numerical value to the cost and the benefit of TD items to prioritize them by weighing their potential benefits against costs [25]	• (Error Proneness) Difficulty in obtaining accurate estimates for benefit and cost, especially for qualitative benefits, and the quality of the results is highly dependant on the quality of cost and value estimates [35]	S1, S6, S9, S16, S21
Modern Portfolio Theory (MPT)	Employs the use of financial MPT techniques to prioritize TD by assembling a portfolio of assets that maximizes the expected return for a given level of risk [39, 44]	• (Computational Complexity & Rank Updates) The approach depends on the risk measures, return measures, and correlations, which might be challenging and complicated to measure for large amounts of TD [39] • (Error Proneness) Requires past historical data to accurately assess return and risk, which can lead to problems if new circumstances occur that were not present in the past [39]	S2, S17
Predictive analytics	Uses data mining, predictive modeling, statistical, and machine learning techniques to analyze current and historical TD data, which are then used to identify patterns that aid in prioritizing TD items [17]	• (Error Proneness) Require significant human effort to build training data set and finding categorization patterns. The training data set is used to train and build the model, and the validity of obtained results is highly dependent on the quality of the data [13] • (Scalability) The models and categorization patterns might be software system or team dependant [13]	S14, S15, S18
Ranking	Develops a calculation model to generate a value that aids in TD prioritization through sorting or threshold-based approaches [5, 15]	• (Loss of Information) The final rank may not display the relative difference between ranked TD items [38] • (Rank Updates) Adding a new TD item may require a redo of the analysis, since these calculation models are, generally, based on the TD set [23, 42]	S5, S7, S8, S13, S20
Real-Options Analysis (ROA)	Employs financial ROA techniques to value and prioritize TD items, while considering the possibility to adjust them in the future [44]	• (Error Proneness) Expected results are subject to error based on the accuracy of the calculation of value, cost, and other option pricing criteria [14] • (Applicability) Some real options' assumptions (e.g., liquidity) are not applicable in the software engineering field [46]	S10, S22, S23.2
Reinforcement learning	Develops an optimal policy that determines TD repayment actions, which aim to maximize a reward [6, 27]	• (Error Proneness & Computational Complexity) Depending on how long the agent is run, the results may vary due to the innate nature of reinforcement learning when it comes to exploitation versus exploration trade-off [27] • (Scalability) Many reinforcement learning techniques struggle to properly scale and accurately solve large problems [27] • (Rank Updates) Requires rerunning the model when a new TD item is added to the preexisting pool of TD items [6]	S19
Software Quality Assessment Based on Lifecycle Expectations (SQALE)	Uses predefined, modifiable rules and measures to identify, quantify, and categorize TD using several indicators that can aid in TD prioritization [31]	• (Error Proneness) Challenges arise in identifying "right code" and accurately mapping each quality requirement into one of the SQALE's quality model characteristics [31]	S12
Weighted Sum Model (WSM)	A multi-criteria decision analysis method for determining the importance of a TD item by summing the weights of importance of the defined criteria [52, 53]	• (Error Proneness & Scalability) Criteria need to be additive, and difficulty arises when dealing with criteria that have different units of measurement [50] • (Error Proneness) Difficulty in determining the ideal weight for each criterion, and the results are highly sensitive to these weights [50]	S3, S11

item. Without the input of relevant stakeholders or if such stakeholders are uninformed, the given technique is doomed to output inaccurate results [5]. A communication among stakeholders limitation would be encountered when utilizing the BPM prioritization technique. BPM requires the technical stakeholders and the business stakeholders to jointly evaluate TD. In the case that the various stakeholders are unable to communicate or meet with each other, BPM will be unusable [20]. **Computational complexity**: The cost of applying a technique to prioritize TD items, which is measured by the number of operations required, the amount of memory used, the amount of time required, and the order in which the memory is used [5]. One example of a technique with a high computational complexity would be reinforcement learning. Due to the innate nature of reinforcement learning, results have a degree of variance depending on how long the agent is run [27]. **Error proneness**: A few existing prioritization techniques are prone to errors. When using this type of technique, small changes in variables or the accuracy of estimates can substantially alter the results achieved [5].

Due to this restriction, one must be meticulous when employing a prioritization technique that depends on estimates, such as CBA that is reliant on the accuracy of the cost and value estimation [35]. **Loss of information**: An ordered list of TD items is occasionally displayed as the result of a few prioritization techniques, such as ranking. However, this ranking, while effective at displaying the prioritization results, can run into problems when attempting to convey the individual impact of each TD item. It can be difficult to view a ranked list and fully understand the scope of each TD item. This potential loss of information includes the cost of each TD item, the impact of each item on a given system, and more [5]. An example of this limitation is the use of ranking as a prioritization technique. The resulting ranked list only displays the relative importance of each TD item and leaves out all other information [38]. **Rank updates**: The ability to automatically update ranks anytime a TD item is added or removed from the prioritized list of items is considered a rank update. A rank update limitation is when a provided prioritization technique's results are subject to change if an additional item is added or removed. With the inclusion or exclusion of an item, the selected prioritization technique must be rerun to generate a new list of rankings. Furthermore, the entire prioritized list of TD items is subject to change, which can lead to inaccurate results and frustration [5]. MPT can have a rank update limitation, as the model must identify the relationships a new TD item has with all of the preexisting items, which can result in a completely new portfolio [39]. **Scalability**: Describes the capacity of a technique to handle large amounts of TD. Techniques that become unmanageable and require plenty of resources, such as effort, when dealing with large amounts of TD are considered unscalable [5]. An example of such a technique is AHP, where TD items are prioritized based on n(n-1)/2 pairwise comparisons. AHP is unscalable when dealing with many TD items or many criteria, as the effort and time required to compute all the comparisons become unrealistic [52].

4 THREATS TO VALIDITY

The validity of this SLR is subject to construct, internal, and external threats. This section discusses each threat and its mitigation strategy based on the guidelines in [29].

4.1 Construct Validity

Construct validity is concerned with the design of a study. Primarily, construct validity focuses on the adequacy of a study's design to address its research questions [29]. Potential threats to the construct validity of this SLR, primarily, encompass the search strategy and study selection. Pilot trials were conducted to select the appropriate search string to mitigate construct threats in the search string. The search string that resulted in the most relative number of results and that was adopted by previous TD secondary studies [10, 11, 21] was selected. Additionally, the quasi-gold standard was applied to assess the completeness of the search strategy and quality of the search string. To mitigate threats related to the study selection strategy, the strategy was based on the software engineering systematic review guidelines [29] and previous secondary studies on TD.

4.2 Internal Validity

Internal validity is concerned with the conducts of the study. In the context of secondary studies, potential threats to a study's internal validity include weakness in one's data synthesis and extraction methods [29]. To mitigate internal validity threats during the study selection process, Kitchenham et al.'s [29] guidelines were followed to construct the search strategy. Two authors independently decided which papers should be included by reading the title and abstract of each paper and applying the predetermined selection criteria to each study. The authors followed a conservative strategy, meaning that discrepancies between the authors' analysis were resolved by the authors reading the entirety of each study and discussing the findings. All studies where a consensus was not achieved were included to strengthen the internal validity of this research. The lack of quality assessment of the included studies is another threat to the internal validity of our study. We did not conform with the guidelines mentioned above and did not apply such an assessment, due to the small number of studies regarding TD prioritization. If we were to apply such an assessment, the assessment might result in the omission of a large portion of our included studies which would lead to difficulties when answering our research questions and understanding the current approaches and techniques for TD prioritization. Kitchenham et al. 's [29] guidelines were followed during data extraction as well. Three authors independently read and extracted the data from each study selected and discussed the results through the means of collective discussions.

4.3 External Validity

External validity is concerned with the generalizability of a study's findings. In the context of secondary studies, it involves the range of primary studies covered [29]. To eliminate any threats to this aspect of validity, the standard software engineering databases were considered during the search [29]. The search scope was also complemented with Google Scholar, which is believed to be the "most comprehensive" source of literature available [11]. Moreover, multiple iterations of forward and backward snowballing were performed to expand the search scope [29]. Though the aim was to cover a representative body of current TD prioritization literature, the findings may not be generalizable to TD prioritization studies outside of this search scope. It must be noted that this study's conclusions are only applicable to the studies that were assessed.

5 RELATED WORK

This section presents the contributions of the software engineering community to understand the current TD and TDM literature.

Tom et al. [47] were the first to conduct an SLR on TD. The study's aim is to establish boundaries of the TD metaphor and develop a comprehensive theoretical framework to facilitate future research. To build upon their results and expand their search scope, Tom et al. [48] performed a multivocal literature review (MLR), supplemented by interviews with 11 software practitioners. The study summarized its findings in a theoretical framework that provides a comprehensive understanding of TD from academic and industry perspectives.

Unfortunately, in 2014, there were many ambiguities present in the use of TD as a term and how to manage it. Li et al. [33] aimed

to address this gap of knowledge and conducted a systematic mapping review (SMR) with the objective to acquire a comprehensive, general understanding of TD and TDM and to display promising future research directions. To address the current strategies, at the time, that have been proposed to identify or manage TD in software projects, Alves et al. [9] conducted an SMR that focuses on defining different TD types, identifying indicators of TD existence, identifying TDM strategies, understanding the maturity level of each identification and management technique, and identifying the available TD visualization techniques. Fernández-Sánchez et al. [21] performed an SMR, with the focus being on the elements required to manage TD effectively. The authors summarized the elements that are considered in TDM, in general, and categorized them into three categories: essential decision factors, cost estimation techniques, and techniques for decision-making. Ribeiro et al. [41] identified the decision criteria utilized in the repayment of TD by conducting an SMR. The SMR included all studies that explored decision making criteria in its theory, practice, or approach, without any distinction between these types of studies. As a result, the authors provided a summary of 14 decision criteria used in the TD repayment process.

Lenarduzzi et al's ongoing review [51], which is currently available as a preprint, also aims to analyze the current TD prioritization approaches. Though the review and this study focus on the same topic, this study and Lenarduzzi et al's review are significantly different and each draw unique conclusions. It should be noted that the two studies have differing resultant literature pools, due to differences in the search strategies employed. Contrary to this SLR, [51] has an expanded definition of TD prioritization, which includes deciding between repaying TD items or implementing new features. The authors analyze publications in regards to different aspects of TD prioritization: whether an approach is a one-time activity or a continuous process, whether an approach decides between TD items or repaying TD items instead of implementing new features, and identifying tools utilized in TD prioritization. Though this study's focus is also on TD prioritization, the utilized definition of TD prioritization in this study concentrates on deciding which TD items are to be repaid among a set of TD items. In contrast to [51], this study summarizes the software artifact dependencies, type of required human involvement, and the evaluation approach and settings of each identified approach. Additionally, this SLR identifies the prioritization techniques that each approach is based on and highlights a few of the limitations of each technique. Both reviews analyze the TD type addressed by each approach and identify the decision factor categories considered in each approach. However, as opposed to 53 unique factors being categorized into 11 categories, this study identifies whether an approach accounts for value, cost, or resources constraints. Moreover, this study summarizes how value and cost are estimated in each approach.

Some secondary studies have focused on specific aspects of TD and TDM. To understand the financial aspects of TDM, Ampatzoglou et al. [10] conducted an SLR to identify the financial approaches utilized in TDM and create a glossary of financial terms and definitions related to TDM in the software engineering field. Focusing on TD in the context of agile development, Behutiye et al. [12] conducted an SLR aiming to analyze and synthesize TD, its causes and consequences, and various TDM strategies in agile context. Becker et al. [11] conducted an SLR that concentrates on the trade-off decisions across time in TDM. The review reveals that only nine studies explicitly used empirical methods to study specific trade-off decisions, and that these studies have different notions of time and none of them account for inter-temporal choices.

6 CONCLUSION

The study of TD and specifically its prioritization is a relatively new topic of research. It is important to consolidate the current research on such a topic in order to spur new growth in the field and allow practitioners to learn about its current state. This study conducts an SLR of the current TD literature to identify the existing TD prioritization approaches and the prioritization techniques that these approaches utilize. Several research questions were formulated with this goal in mind regarding what current TD prioritization approaches exist, types of TD addressed, decision factor categories considered, software artifact dependencies, required human involvement, and their evaluation approaches. Additionally, this SLR explores the topic of prioritization techniques and such techniques' limitations. The SLR identified 24 unique TD prioritization approaches, which employed the use of 10 unique prioritization techniques. The SLR found that many of the approaches do not consider resources constraint and only four of the approaches considered all three decision factor categories of value, cost, and resources constraint in their TD prioritization process. Additionally, the SLR revealed that the evaluations of a few approaches were lacking. Of the identified approaches, 37.5% were not evaluated in any manner, and the majority of the evaluated approaches were not evaluated in industry settings. This lack of industry evaluations diminishes the credibility of the approaches.

This SLR allows individuals to view the current TD prioritization approaches and several characteristics of such approaches. Practitioners can utilize this SLR as a reference to gain an overview of existing TD prioritization approaches. Additionally, TD researchers might find this review beneficial when developing new TD prioritization approaches. Based on the results of this SLR, more research needs to be conducted in reference to creating TD prioritization approaches that consider cost, value, and resource constraints, in addition to including an evaluation method in an industry setting. Furthermore, this SLR demonstrates that there is no one concise method to estimate value and cost of TD items, and these estimates are context-dependent. Consequently, new approaches should be flexible enough to incorporate various value and cost estimation methods.

7 ACKNOWLEDGMENT

This material is based upon work supported in part by the U.S. Department of Defense through the Systems Engineering Research Center (SERC) under Contract H98230-08-D-0171. SERC is a federally funded University Affiliated Research Center managed by Stevens Institute of Technology.

REFERENCES

[1] 2016. *Developer Survey Results 2016.* Retrieved December 27, 2018 from https://insights.stackoverflow.com/survey/2016/

[2] 2018. *The Developer Coefficient Software engineering efficiency and its $3 trillion impact on global GDP.* Retrieved December 27, 2018 from https://stripe.com/files/reports/the-developer-coefficient.pdf

[3] 2020. *Summary of Technical Debt Prioritization Approaches*. Retrieved March 10, 2020 from https://drive.google.com/file/d/11XRpmkxbKaOyiwGrfm1UAWF0sqJJyn41/view?usp=sharing

[4] Zahra Shakeri Hossein Abad and Guenther Ruhe. 2015. Using real options to manage technical debt in requirements engineering. In *2015 IEEE 23rd International Requirements Engineering Conference (RE)*. IEEE.

[5] Philip Achimugu, Ali Selamat, Roliana Ibrahim, and Mohd Naz'ri Mahrin. 2014. A systematic literature review of software requirements prioritization research. *Information and software technology* (2014).

[6] Shirin Akbarinasaji. 2015. Toward measuring defect debt and developing a recommender system for their prioritization. In *Proceedings of the 13th International Doctoral Symposium on Empirical Software Engineering*. 15–20.

[7] Mashel Albarak and Rami Bahsoon. 2018. Prioritizing technical debt in database normalization using portfolio theory and data quality metrics. In *Proceedings of the 2018 International Conference on Technical Debt*. ACM, 31–40.

[8] Abdullah Aldaeej and Carolyn Seaman. 2018. From Lasagna to Spaghetti: A Decision Model to Manage Defect Debt. In *2018 IEEE/ACM International Conference on Technical Debt (TechDebt)*. IEEE, 67–71.

[9] Nicolli SR Alves, Thiago S Mendes, Manoel G de Mendonça, Rodrigo O Spínola, Forrest Shull, and Carolyn Seaman. 2016. Identification and management of technical debt: A systematic mapping study. *Information and Software Technology* 70 (2016), 100–121.

[10] Areti Ampatzoglou, Apostolos Ampatzoglou, Alexander Chatzigeorgiou, and Paris Avgeriou. 2015. The financial aspect of managing technical debt: A systematic literature review. *Information and Software Technology* (2015).

[11] Christoph Becker, Ruzanna Chitchyan, Stefanie Betz, and Curtis McCord. 2018. Trade-off decisions across time in technical debt management: a systematic literature review. In *Proceedings of the 2018 International Conference on Technical Debt*. ACM, 85–94.

[12] Woubshet Nema Behutiye, Pilar Rodríguez, Markku Oivo, and Ayşe Tosun. 2017. Analyzing the concept of technical debt in the context of agile software development: A systematic literature review. *Information and Software Technology* 82 (2017), 139–158.

[13] Christian Bird, Tim Menzies, and Thomas Zimmermann. 2015. *The art and science of analyzing software data*. Elsevier.

[14] Edward H Bowman and Gary T Moskowitz. 2001. Real options analysis and strategic decision making. *Organization science* (2001).

[15] Aabha Choudhary and Paramvir Singh. 2016. Minimizing Refactoring Effort through Prioritization of Classes based on Historical, Architectural and Code Smell Information.. In *QuASoQ/TDA@ APSEC*. 76–79.

[16] Alan Clarke. 1999. *Evaluation research: An introduction to principles, methods and practice*. Sage.

[17] Zadia Codabux and Byron J Williams. 2016. Technical debt prioritization using predictive analytics. In *Proceedings of the 38th International Conference on Software Engineering Companion*. ACM.

[18] Daniela S Cruzes and Tore Dyba. 2011. Recommended steps for thematic synthesis in software engineering. In *2011 International Symposium on Empirical Software Engineering and Measurement*. IEEE.

[19] Ward Cunningham. 1993. The WyCash portfolio management system. *ACM SIGPLAN OOPS Messenger* 4, 2 (1993), 29–30.

[20] Rodrigo Rebouças de Almeida, Uirá Kulesza, Christoph Treude, Aliandro Higino Guedes Lima, et al. 2018. Aligning Technical Debt Prioritization with Business Objectives: A Multiple-Case Study. In *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE.

[21] Carlos Fernández-Sánchez, Juan Garbajosa, Agustín Yagüe, and Jennifer Perez. 2017. Identification and analysis of the elements required to manage technical debt by means of a systematic mapping study. *Journal of Systems and Software* 124 (2017), 22–38.

[22] Francesca Arcelli Fontana, Vincenzo Ferme, Marco Zanoni, and Riccardo Roveda. [n.d.]. Towards a prioritization of code debt: A code smell intensity index. In *2015 IEEE 7th International Workshop on Managing Technical Debt*. IEEE.

[23] Everton Guimaraes, S Vidal, A Garcia, JA Diaz Pace, and Claudia Marcos. [n.d.]. Exploring architecture blueprints for prioritizing critical code anomalies: Experiences and tool support. *Software: Practice and Experience* ([n. d.]).

[24] Yuepu Guo and Carolyn Seaman. 2011. A portfolio approach to technical debt management. In *Proceedings of the 2nd Workshop on Managing Technical Debt*.

[25] Yuepu Guo, Rodrigo Oliveira Spínola, and Carolyn Seaman. 2016. Exploring the costs of technical debt management–a case study. *Empirical Software Engineering* 21, 1 (2016), 159–182.

[26] Muhammad Firdaus Harun and Horst Lichter. 2015. Towards a technical debt-management framework based on cost-benefit analysis. In *Proceedings of the 10 th International Conference on Software Engineering Advances*.

[27] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. 1996. Reinforcement learning: A survey. *Journal of artificial intelligence research* (1996).

[28] Hans Kellerer, Ulrich Pferschy, and D. Pisinger. 2011. *Knapsack problems*. Springer.

[29] Barbara Ann Kitchenham, David Budgen, and Pearl Brereton. 2015. *Evidence-based software engineering and systematic reviews*. CRC press.

[30] Edouard Kujawski. 2003. Multi-Criteria Decision Analysis: Limitations, Pitfalls, and Practical Difficulties. In *INCOSE International Symposium*, Vol. 13. Wiley Online Library, 1169–1176.

[31] Jean-Louis Letouzey. [n.d.]. The SQALE method for evaluating technical debt. In *2012 Third International Workshop on Managing Technical Debt*. IEEE.

[32] Jean-Louis Letouzey and Michel Ilkiewicz. 2012. Managing technical debt with the sqale method. *IEEE software* 29, 6 (2012), 44–51.

[33] Zengyang Li, Paris Avgeriou, and Peng Liang. [n.d.]. A systematic mapping study on technical debt and its management. *Journal of Systems and Software* ([n. d.]).

[34] Solomon Mensah, Jacky Keung, Jeffery Svajlenko, Kwabena Ebo Bennin, and Qing Mi. 2018. On the value of a prioritization scheme for resolving Self-admitted technical debt. *Journal of Systems and Software* 135 (2018), 37–54.

[35] John L Moore. 1995. Cost-benefit analysis: Issues in its use in regulation. Congressional Research Service, Library of Congress.

[36] David Lorge Parnas. 1994. Software aging. In *Proceedings of the 16th international conference on Software engineering*.

[37] Reinhold Plösch, Johannes Bräuer, Matthias Saft, and Christian Körner. 2018. Design debt prioritization: a design best practice-based approach. In *2018 IEEE/ACM International Conference on Technical Debt (TechDebt)*. IEEE.

[38] Muhammad Ramzan, M Arfan Jaffar, and Arshad Ali Shahid. 2011. Value based intelligent requirement prioritization (VIRP): expert driven fuzzy logic based prioritization technique. *International Journal Of Innovative Computing, Information And Control* 7, 3 (2011), 1017–1038.

[39] Frank K Reilly and Keith C Brown. 2002. *Investment analysis and portfolio management*.

[40] Leilane Ferreira Ribeiro, Nicolli Souza Rios Alves, Manoel Gomes de Mendonca Neto, and Rodrigo Oliveira Spínola. [n.d.]. A Strategy Based on Multiple Decision Criteria to Support Technical Debt Management. In *2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE.

[41] Leilane Ferreira Ribeiro, Mário André de Freitas Farias, Manoel G Mendonça, and Rodrigo Oliveira Spínola. 2016. Decision Criteria for the Payment of Technical Debt in Software Projects: A Systematic Mapping Study.. In *ICEIS (1)*.

[42] Natthawute Sae-Lim, Shinpei Hayashi, and Motoshi Saeki. 2018. Context-based approach to prioritize code smells for prefactoring. *Journal of Software: Evolution and Process* 30, 6 (2018), e1886.

[43] Klaus Schmid. 2013. A formal approach to technical debt decision making. In *Proceedings of the 9th international ACM Sigsoft conference on Quality of software architectures*. ACM, 153–162.

[44] Carolyn Seaman, Yuepu Guo, Nico Zazworka, Forrest Shull, Clemente Izurieta, Yuanfang Cai, and Antonio Vetrò. 2012. Using technical debt data in decision making: Potential decision approaches. In *2012 Third International Workshop on Managing Technical Debt (MTD)*. IEEE, 45–48.

[45] Will Snipes, Brian Robinson, Yuepu Guo, and Carolyn Seaman. 2012. Defining the decision factors for managing defects: a technical debt perspective. In *2012 Third International Workshop on Managing Technical Debt (MTD)*. IEEE, 54–60.

[46] Paul P Tallon, Robert J Kauffman, Henry C Lucas, Andrew B Whinston, and Kevin Zhu. 2002. Using real options analysis for evaluating uncertain investments in information technology: Insights from the ICIS 2001 debate. *Communications of the Association for Information Systems* (2002).

[47] Edith Tom, Aybuke Aurum, and Richard Vidgen. 2012. A consolidated understanding of technical debt. (2012).

[48] Edith Tom, Aybuke Aurum, and Richard Vidgen. 2013. An exploration of technical debt. *Journal of Systems and Software* 86, 6 (2013), 1498–1516.

[49] Adam Tornhill. 2018. Prioritize technical debt in large-scale systems using codescene. In *Proceedings of the 2018 International Conference on Technical Debt*.

[50] Evangelos Triantaphyllou. 2000. Multi-criteria decision making methods: A comparative study. In *Multi-criteria decision making methods: A comparative study*. Springer, 5–21.

[51] Valentina Lenarduzzia,Terese Beskerb, Davide Taibia, Antonio Martinic, Francesca Arcelli Fontana. [n.d.]. Technical Debt Prioritization: State of the Art. A Systematic Literature Review. https://arxiv.org/pdf/1904.12538.pdf.

[52] Mark Velasquez and Patrick T Hester. [n.d.]. An analysis of multi-criteria decision making methods. *International Journal of Operations Research* ([n. d.]).

[53] Santiago Vidal, Hernan Vazquez, J Andres Diaz-Pace, Claudia Marcos, Alessandro Garcia, and Willian Oizumi. 2015. JSpIRIT: a flexible tool for the analysis of code smells. In *2015 34th International Conference of the Chilean Computer Science Society (SCCC)*. IEEE, 1–6.

[54] Claes Wohlin. 2014. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proceedings of the 18th international conference on evaluation and assessment in software engineering*. Citeseer, 38.

[55] Claes Wohlin et al. 2005. *Engineering and managing software requirements*. Springer Science & Business Media.

[56] Nico Zazworka, Carolyn Seaman, and Forrest Shull. 2011. Prioritizing design debt investment opportunities. In *Proceedings of the 2nd Workshop on Managing Technical Debt*. ACM, 39–42.

[57] He Zhang, Muhammad Ali Babar, and Paolo Tell. 2011. Identifying relevant studies in software engineering. *Information and Software Technology* (2011).