Information and Software Technology 64 (2015) 52–73

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

The financial aspect of managing technical debt: A systematic literature review

Areti Ampatzoglou [a,b], Apostolos Ampatzoglou [a,*], Alexander Chatzigeorgiou [b], Paris Avgeriou [a]

[a] Department of Mathematics and Computer Science, University of Groningen, Netherlands
[b] Department of Applied Informatics, University of Macedonia, Thessaloniki, Greece

ARTICLE INFO

Article history:
Received 27 October 2014
Received in revised form 31 March 2015
Accepted 1 April 2015
Available online 11 April 2015

Keywords:
Technical debt
Financial debt
Financial terms
Systematic literature review

ABSTRACT

Context: Technical debt is a software engineering metaphor, referring to the eventual financial consequences of trade-offs between shrinking product time to market and poorly specifying, or implementing a software product, throughout all development phases. Based on its inter-disciplinary nature, i.e. software engineering and economics, research on managing technical debt should be balanced between software engineering and economic theories.
Objective: The aim of this study is to analyze research efforts on technical debt, by focusing on their financial aspect. Specifically, the analysis is carried out with respect to: (a) how financial aspects are defined in the context of technical debt and (b) how they relate to the underlying software engineering concepts.
Method: In order to achieve the abovementioned goals, we employed a standard method for SLRs and applied it on studies retrieved from seven general-scope digital libraries. In total we selected 69 studies relevant to the financial aspect of technical debt.
Results: The most common financial terms that are used in technical debt research are principal and interest, whereas the financial approaches that have been more frequently applied for managing technical debt are real options, portfolio management, cost/benefit analysis and value-based analysis. However, the application of such approaches lacks consistency, i.e., the same approach is differently applied in different studies, and in some cases lacks a clear mapping between financial and software engineering concepts.
Conclusion: The results are expected to prove beneficial for the communication between technical managers and project managers, in the sense that they will provide a common vocabulary, and will help in setting up quality-related goals, during software development. To achieve this we introduce: (a) a glossary of terms and (b) a classification scheme for financial approaches used for managing technical debt. Based on these, we have been able to underline interesting implications for researchers and practitioners.

Contents

* Corresponding author.
E-mail addresses: areti.ampatzoglou@rug.nl (A. Ampatzoglou), a.ampatzoglou@rug.nl (A. Ampatzoglou), achat@uom.gr (A. Chatzigeorgiou), paris@cs.rug.nl (P. Avgeriou).

http://dx.doi.org/10.1016/j.infsof.2015.04.001
0950-5849/

1. Introduction

Maintenance is one of the most effort-intensive activities in the software lifecycle. It is estimated that maintenance activities consume 50–75% of the total effort spent during the complete lifecycle of a typical software project [36]. From all maintenance activities,[1] it is reasonable to assume that requests for changes concerning the addition of functionality, the adaptation to new environments, the enhancement of run-time qualities [3], and the correction of errors, are hard to neglect or defer to a future iteration. On the contrary, changes that are not directly related to the external behavior of the system but relate to design-time qualities [3], are often postponed or neglected, in order to shrink product time to market and reduce short-term costs. However, software systems are by definition highly evolving products, whose design-time quality will gradually decay [31], and therefore deferring such maintenance activities (e.g., refactorings, resolution of bad smells, reverse engineering) might have a significant impact on several design-time qualities (e.g., maintainability, comprehensibility, reusability, etc.). This strategy leads to the creation of a financial overhead due to degraded quality, originally termed by Cunningham as *technical debt* [8].

Technical debt (TD) is a metaphor that is used to draw an analogy between financial debt as defined in economics and the situation in which an organization decides to produce immature software artifacts (e.g. designs or source code), in order to deliver the product to market within a shorter time period [8]. TD is accumulated during all development phases, i.e. requirements analysis, architectural/detailed design, and implementation, and therefore should be monitored and handled during the complete software lifecycle [23]. In practice, technical debt is sometimes desirable (e.g., in cases when companies opt for investing on a different product rather than producing a new one with optimum design-time quality), whereas the complete repayment of TD is considered unrealistic [12]. However, since the accumulation of technical debt may severely hinder the maintainability of the software [38], it should be continuously monitored and managed.

One of the most prevalent characteristics of technical debt is its interdisciplinary nature, since it combines elements from both financial and software engineering theory. Although this nature may lead to additional challenges (resulting from the gap between software and financial perspectives) it can potentially help the *communication between both practitioners and researchers.* Concerning practitioners we refer to the communication gap between technical stakeholders (software engineers, architects, testers, etc.) and project managers. On the one hand, project managers are interested in concepts like value, cost, benefit, debt, principal (i.e. capital), and interest. Moreover, they are not particularly focused on the design-time quality of intermediate artifacts during the software development lifecycle, since it is only indirectly associated with their basic goals as stakeholders in the software development lifecycle, i.e. increase benefit, decrease production cost, shrink time to market, etc. On the other hand, technical stakeholders are usually more focused on design-time quality. Enhanced design-time quality in terms of e.g. maintainability, reusability and understandability, eases the post-production activities of development teams and decreases the effort needed for the development of similar projects. However, in practice, high-level budget and effort allocation decisions are taken by project managers. Thus, in order for maintenance activities to be approved, technical stakeholders should communicate their benefits to project managers. In this type of communication, the terminology of technical debt can prove beneficial. *Practitioners' experience suggests* that *using economics-based terminology* and approaches like real-options, cost/benefit analysis, and portfolio management, bridges the gap between software engineers and managers and *facilitates the negotiation of trade-offs* of quality for quicker product delivery [11].

Despite the communication benefits that Technical Debt may bring, due to its aforementioned interdisciplinary nature, the *body of knowledge* on the subject can be rather difficult to comprehend in-depth, since it requires expertise or at least experience from

[1] We deliberately avoid the use of software maintenance types, such as those defined by ISO/IEC 14764-2006 [18] (i.e., corrective, adaptive, perfective, preventive), because these types are interpreted differently by different researchers, causing a naming ambiguity [7,15,34].

two rather diverse scientific domains (i.e., software engineering and economics). Therefore, researchers with financial background are often not sufficiently familiar with terms like refactorings and source code analysis, whereas software engineering researchers are usually not experts in applying methods like real-options and portfolio management. Consequently, in many cases, research on the subject: (a) uses *ambiguous terminology* and *sometimes misuses terms*, (b) *is not balanced* between the economical and the software engineering aspects, or in other words it is not truly interdisciplinary. For example, when a purely software engineering team with limited expertize on economics, apply the real-options theory, there is an increased possibility that the method is not perfectly applied.

Furthermore, the current literature in TD lacks a study summarizing the state of the art and practice with respect to financial approaches used in TD. This makes it difficult for practitioners to search for such approaches, and evaluate them for fitness of purpose and applicability for the particular problem at hand. Even when they manage to select an appropriate approach, applying such an approach can be challenging, since most of them lack an established way of mapping their original (from the financial domain) inputs and outputs to software engineering technologies.[2] For example, in real options analysis the improvement probability factor is required as an input to the method [32], but how is this probability calculated in the context of technical debt and what software engineering technologies should be used for assessing it, and how? Moreover, the lack of a state of the art also hinders researchers in understanding the overall picture of what has been studied by this community, and what is still missing.

In this study we will analyze existing literature, by employing the Systematic Literature Review (SLR) method [20] to tackle the two aforementioned problems, i.e. the communication gap among researchers and among practitioners, and the lack of a study summarizing the state of the art and practice on financial approaches for technical debt management. Therefore, this study has two goals:

(goal-a)	Introducing a glossary of financial terms and definitions related to technical debt management.
(goal-b)	Classifying financial approaches (originating from traditional or software economics) used in technical debt management.

By referring to Technical Debt Management (TDM) in this study, we are interested in all TDM activities as proposed by Li et al. [24], i.e., (a) identification, (b) measurement, (c) prioritization, (d) repayment, and (e) monitoring. However, since the focus of the study is on the financial perspective of technical debt, special attention will be given to the measurement (or quantification) activity. We note that due to the nature of technical debt research, we have broadened the primary study search space to generic computer science literature, rather than focusing on software engineering literature only.

The rest of the paper is organized as follows: in Section 2, we present related work, i.e. other studies that aim at providing an overview of the research state of the art on the domain of technical debt. In Section 3, we provide some background information on financial concepts that will be used during the presentation and discussion of the results. Next, in Section 4, we present the protocol of this systematic literature review. In Section 5 we present the raw results of this study and answer the research questions, whereas in Section 6 we discuss the basic contributions of this study w.r.t. *goal-a* and *goal-b*. Finally, in Section 7 we present threats to validity, and in Section 8 we conclude the paper.

[2] By the term 'software engineering technologies' we refer to: (a) any element of the software engineering process (e.g., tools, artifacts, activities [22]), (b) quality attributes, and (c) well-established approaches that aim at the improvement of design-time qualities (e.g., patterns, refactorings, etc.).

2. Related work

Although the financial aspect of technical debt is a research topic that has attracted the attention of both academia and industry, to the best of our knowledge there are no literature reviews or mapping studies (either systematic or not), summarizing the research state of the art on the subject. The only two studies that have been identified as related work to this paper, are a Multi-vocal Literature Review (MLR) by Tom et al. published in 2013 [35] and a Mapping Study (MS) by Li et al. published in 2015 [25]. The main points of deviation of our work, compared to these studies, are:

- the use of a different research method (SLR vs. MS vs. MLR), and
- the different focus of the research goals (financial aspects of technical debt vs. software engineering aspects of technical debt).

The main difference between a Multi-vocal Literature Review (MLR) and a Systematic Literature Review (SLR) or a Mapping Study (MS) is the fact that, while SLRs and MSs use as input only academic peer reviewed articles, in MLRs, *grey literature,* such as blogs, white papers and webpages, is also considered as input. We note that despite the differences in the used research methods, the results of both types of studies are valid; however, through a different perspective. Consequently, concerning technical debt, the multi-vocal results [35] are expected to provide indications on how the concept is exploited by both researchers and practitioners; although it goes without saying that using such not peer-reviewed experience reports or position articles raises significant reliability and validity issues [30]. On the other hand, the results of an SLR or an MS could provide an established body of knowledge, focusing only on research contributions. However, comparing the research goals and designs of MS and SLR, the following differences can be identified:

- SLRs aim at providing a complete, detailed and fair synthesis of evidence related to a topic of interest, whereas MSs aim at providing an overview of a research area to assess the quantity of evidence [21];
- the research objectives for MSs are usually high level and include issues, such as identification of research sub-topics and used research, and classification of research themes. On the other hand, SLRs usually synthesize data in a more detailed way [21].

Additionally, the goal of the three studies is completely different as well. More specifically, the studies of Tom et al. [35] and Li et al. [25] aimed at understanding the nature of technical debt and its implications for software development, whereas our study is focused on the financial perspective of technical debt. However, although the research questions of the three studies do not overlap, it is expected that discussing the results will provide possibilities for synthesizing the outcomes of the three secondary studies.

3. Background information

In this section we provide background information on financial debt i.e. definitions, related terms and approaches. More specifically, we provide: (a) basic definitions on financial debt that are

expected to be investigated in the related literature on technical debt, (b) broader financial terms, related to interest theory, and (c) some fundamental strategies for handling financial debt. This section aims at supporting readers with a software engineering background to become familiar with some basic financial concepts that will be needed for understanding the results of this SLR. All terms discussed in this section will be used either during the presentation of results or in the discussion of directions for future research.

3.1. Basic financial debt terms

Debt is a term used to describe the amount of money owed by one party (*debtor* or *borrower*) to another party (*creditor* or *lender*). The certain amount of money derives from a *loan*, which denotes that the money has been lent by the creditor to the debtor for a specific period of time. The obligation of the debtor is to repay a larger sum of money to the creditor at the end of that period [28]. The original amount of money borrowed is called the *principal*, while the additional amount paid back constitutes the *interest*. To accomplish this agreement, the borrower issues a *debt instrument*, which specifies the type of the loan and obliges the debtor to repay it at the *maturity date* (date of repayment). Economists consider the interest to be a fee that is charged for the use of money, i.e. the creditor is being remunerated for the undertaken risk of losing the principal and for the lost benefit that another *investment* of the money would have yielded (*opportunity cost*). On the other hand, the debtor (individual, company or government) is in need of money, in order to proceed with an investment or to cover temporary lack of *liquidity*, therefore decides to pay the interest in the future.

Interest is calculated as a percentage of the principal, usually on an annual basis, namely *interest rate*. Interest rate is dependent upon systemic financial factors (e.g. *quantity of money* and *price level* [14]) and the debtor's *risk rating* [28]. If the debtor is assessed as a low-risk partner, a lower interest rate shall be applied. On the other hand, if the borrower is considered to be of high-risk, a higher interest rate will be charged (see Section 3.3, on risk management). Interest can be discriminated into two common types (simple and compound), according to the way it is calculated. When interest is calculated in every period on the amount of the principal, it is called *simple interest*. In this case, the total amount of interest increases linearly. On the other hand, *compound interest* occurs when it is calculated on the principal plus the accumulated interest. Thus, the total amount of interest increases exponentially [19].

3.2. Broader financial terms (related to investments and interest theory)

In order to elaborate on concepts of debt and loan we present some additional terms that are considered important for financial transactions. More specifically, *liabilities* and *assets* are terms used in both financial and accounting literature and practice and refer to elements that constitute obligations or create economic value for their owners respectively. Moreover, in order to compare the value of a sum of money or a debt instrument through different periods of time, the concepts of *present value* (*PV*) and *future value* (*FV*) are used. So as to assess the effectiveness of an investment, the *Return on Investment (ROI)* ratio is used. A high level of ROI indicates an efficient investment. *Financial leverage* is perceived as the use of *loanable funds*, i.e. money derived from loans by a company, in order to invest in actions that will result in profit. Specifically:

- An asset could be a company's plant, mechanical equipment or money. On the other hand, a company's liabilities can consist of its loans, accounts payable or any other company's obligations. Referring to finance, every debt instrument is a liability for the debtor and at the same time it is an asset for the creditor [28].
- Present Value or Present Discounted Value (PV) refers to the current worth of an amount of money paid or received in the future. Present value is calculated as in (1), where r represents the given rate of interest and n stands for the number of periods [28]

$$PV = \frac{FV}{(1+r)^n} \tag{1}$$

- Future Value *(FV)* or *time value of money* refers to the value of a present sum of money in a specific time the future [19]. According to Kellison [19], future value is calculated as:

$$FV = Principal * (1+r)^n \tag{2}$$

- Financial leverage increases the undertaken risk; nevertheless it is generally acceptable and sometimes preferable over other forms of financing by shareholders, as it increases the market value of the *shares* [9]. The degree of a firm's leverage expresses its dependence on long term debt. This dependency can be shown by the debt to equity ratio that is the percentage of debt on the company's equity.
- *Return on Investment (ROI)* is defined as a ratio that measures the return of an investment divided by the invested capital. More specifically, ROI is defined in (3), where Net Profit is the total gains of the investment minus the cost of the investment [13]. Besides the evaluation of a single investment, ROI ratio can be used to compare different investments.

$$ROI(\%) = \frac{Net\ Profit}{Cost\ of\ Investment} \times 100 \tag{3}$$

3.3. Managing debt strategies

Decision making on undertaking an investment or on taking a loan so as to perform an investment should consider the risks and the cost of the specific financial transaction. Thus, in order to make the most profitable decision, economists usually apply the *Cost/Benefit Analysis* (CBA), i.e. a technique that compares the cost of the realization of a decision to the expected benefit of this action. The implementation of CBA method should take into account all the possible expected costs until the completion of the project, such as cost of acquisition, man hours needed for implementation, operational costs etc.

As mentioned above, risk is another factor that should be considered during the decision making on the realization of a financial transaction. The risk concerning a financial transaction may occur due to *interest rate fluctuations* or to other factors involving the two parties of the transaction [28], e.g. the probability of the debtor to *bankrupt*. In order to accomplish their goals, companies should apply *risk management* strategies. More precisely, they should: (a) examine the possible sources of their risk, (b) evaluate the probability of its occurrence, and (c) implement methodologies for measuring the credibility of their counterparties [9].

A specific technique that is used for reducing or eliminating financial risk and reducing market uncertainty is *hedging* [28]. Hedging involves the engagement in a financial transaction in order to counterbalance the risk undertaken by another transaction. For example, hedging may offset a *long position* (that is buying an asset) by taking an additional *short position* (that is selling an asset). In literature, a field that offers great opportunities for hedging is *financial derivatives market* [28]. Financial derivatives can be defined as financial products whose values are linked to the values of previously issued instruments or other variables [16,28].

A common form of financial derivatives contract is an *Option* contract, which gives the owner the right to buy (*call option*) or sell (*put option*) a specific financial product at a specified price (*strike*

price or *exercise price*) in the future. The purchaser of an option is not obligated to exercise the right of buying or selling, on or until the expiration date. On the other hand, the seller of the option is obligated to sell or buy the financial instrument, if the owner decides to do so. In order to acquire the option to buy or sell a product at a specified price, the purchaser pays an amount called a *premium* [28].

Option Pricing Theory has been widely used during the last decades in order to evaluate opportunities in non-financial or "real" investments, such as land, buildings, plant expansion or multi-stage R&D [6,16]. *Real Option Analysis* (ROA) is extensively applied to assess IT investment risk. By equating managerial flexibility with real options, it permits the inclusion of uncertainty in the valuation model [4].

Finally, the total amount of assets and liabilities of a company or individual is called a *portfolio*. Portfolio management is the technique of managing the assets and liabilities in order to achieve a satisfactory benefit. The aim of a successful portfolio management is to achieve the highest possible profit and at the same time to minimize the risk of the undertaken financial transaction [32].

4. Review methodology

This section presents the protocol of this systematic literature review. A protocol constitutes a pre-determined plan that describes research questions and how the systematic literature review will be conducted. The review protocol includes six activities, namely [20]: (a) define research objectives and questions, (b) define search procedure (search terms and resources), (c) define study selection procedure and inclusion/exclusion criteria, (d) define study quality assessment, (e) define data extraction strategy, and (f) define synthesis of the extracted data.

4.1. Research objectives and research questions

We reformulate the goals of this study, using the Goal-Question-Metric format [2], as follows: *analyze* existing literature on the financial aspects of technical debt *for the purpose* of characterization and evaluation, with respect to terminology (see goal-a in Section 1) and the employed financial approaches (see goal-b in Section 1), from the *point of view of* researchers and practitioners. In order to more systematically investigate the aforementioned goal, our study sets three research questions, which are listed here and explained right below:

- **RQ_1:** *Which are the most common financial terms used in the context of technical debt management?*
- **RQ_2:** *Which financial approaches have been applied in technical debt management?*
  - **$RQ_{2.1}$:** *Which financial approaches have been applied for measuring technical debt, in terms of financial units (e.g., man-months or US$)?*
  - **$RQ_{2.2}$:** *Which financial approaches have been applied for identifying, prioritizing, repaying and monitoring technical debt?*
- **RQ_3:** *Which software engineering technologies have been used by financial approaches in the context of technical debt management?*

The two goals stated in Section 1, i.e. glossary of terms (goal-a) and classification of financial approaches in TDM (goal-b), are achieved in two steps. First, we collect data (according to the process in Section 4.5) in order to answer each research question and present them in Section 5. Second, we further analyze the collected data in order to achieve the stated goals, as presented in Section 6. Specifically, for the first goal we will utilize the results of RQ_1 (they will provide a list of terms that are used in technical debt research) and the background information presented in Section 3. Specifically, we use the descriptions of terms in financial theory, as presented in Section 3, in order to fine-tune the definition of the glossary terms. Additionally, for the second goal, we will synthesize the results of RQ_2 (that will provide a list of financial approaches) and RQ_3 (that will provide us information about the employed software engineering technologies), which in turn are used as inputs when applying the financial approaches in technical debt management. Consequently, the classification synthesizes the set of financial approaches with the software technologies they use as input. We note that due to our special interest on TD measuring approaches we have set a separate research question concerning measurement ($RQ_{2.1}$), whereas other managing activities are investigated through $RQ_{2.2}$.

4.2. Search process

The search procedure aimed at the identification of candidate primary studies, which would be either included or excluded from the final set of the primary studies. The search plan involved automated search into seven well-known digital libraries (ACM Digital Library, IEEExplore, ScienceDirect, SpringerLink, Scopus, Web of Science, and Google Scholar), and a manual search for string calibration and validation of the process accuracy. To cross check the results we obtained from the automatic search and to create a valid search string, we searched a small number of venues manually, similarly to determining a "quasi-gold" standard, as proposed by Zhang and Babar [39]. Venues for the manual search were determined based on their probability to publish research in the context of software engineering and technical debt. Therefore, the manual search was limited to the following[3]:

- International Workshop on Managing Technical Debt (MTD Workshop).
- IEEE Software (SW).
- Journal of Software Quality (SQJ).

When manually searching the venues, we considered title, keywords, and (if necessary) abstract, and compared the results with those of the automated search, so as to estimate if we were missing any studies. The results from the automated search included all studies found from applying the "quasi-gold" standard. In the automated search and by taking into account the limited number of articles on the subject, as well as the natural connection of the term 'debt' to economics, we believe that the search string should consist of one string, i.e. "*technical debt*". The searching completed on September 2013.

4.3. Article filtering phases

The papers that were selected as candidate primary studies in the review should be relevant to technical debt, and more specifically to its economic perspective. In line with Dyba and Dingsoyr [10], there are three stages of filtering the article set to produce the primary study data set:

1. The search process, described in Section 4.2, returned a set of candidate primary studies.

[3] MTD has been selected since it is the most specialized venue for technical debt research, whereas SW and SQJ have been selected, since they are highly reputed software engineering journals and the authors are aware of both journals having recently published a number of articles on TD.

2. The article set went through a manual inspection of each article's title, and abstract/conclusions. The inclusion/exclusion criteria that have been used in every stage are listed below:
   Inclusion Criteria:
   - Study should be related to financial aspects of technical debt. In order for a study to be related to financial aspects of technical debt it should: (a) apply a financial method in technical debt management, or (b) measure financial debt in terms of some currency, e.g., US$ (i.e., studies that provide indicators of technical debt – measures that are correlated to technical debt, but do not quantify in money or effort – are excluded), or (c) define/discuss a financial term in the context of technical debt.

   Exclusion Criteria:
   - Study only mentions technical debt in an introductory statement and does not focus on its calculation, definition, economical aspects, etc.
   - Study is an editorial, keynote, opinion, tutorial, workshop summary report, poster, or panel. Such papers have been excluded either due to their small size or due to the fact that such articles are usually not peer-reviewed.
   - Study's full text is not available
3. The qualified articles went through the same inclusion/exclusion criteria, by taking into account the full text of the articles.

Every article selection phase has been handled by the first author and possible doubts have been resolved by a discussion between all authors.

4.4. Quality assessment

All selected papers, which are included in the review, have gone through a quality assessment process, in order to investigate if they were of adequate standard, so as to be included in this review. Thus, each study has been evaluated against a set of questions with respect to the used method, and the quality of the reporting. As the basis for our primary study quality assessment, we used the following instrument:

1. Are the aims and objectives clearly reported [10]? (0: no, 0.5 partially, 1.0 yes)
2. Is the study's focus or main focus on TD [10]? (0: no, 0.5 partially, 1.0 yes)
3. How much evidence supports the results of the study [1] (0.0–1.0 points)?
   - Level 0: No evidence (0.0).
   - Level 1: Evidence obtained from demonstration or working toy examples (0.2).
   - Level 2: Evidence obtained from expert opinions or observations (0.4).
   - Level 3: Evidence obtained from academic studies, e.g. lab experiments (0.6).
   - Level 4: Evidence obtained from industrial studies, e.g., case studies (0.8).
   - Level 5: Evidence obtained from industrial practice (1.0).
4. Is there a clear statement on the TDM activity that study focuses? (0: no, 1.0 yes)
5. Is the study sufficiently using both the financial and the software engineering aspect of TD? (0: none, 0.5 only one, 1.0 both)

The first two questions concerning the aims and focus of the examined primary studies are based on the work by Dyba and Dingsoyr [10]. The third question evaluates the empirical rigor and industrial relevance of the studies based on the classification by Alves et al. [1]. The final two questions aim at assessing the fitness of the candidate primary study for answering our research questions. The complete evaluation of the studies is presented in Appendix D. The inclusion threshold has been set to a total score of 2.0, because questions 3, 4, and 5, might be assigned a value of zero in a paper that approaches technical debt management from a theoretical perspective. For such papers, we require at least that the goals and focus on TDM are explicit.

Table 1
Data synthesis methods.

Research question	Variables used	Synthesis method
RQ_1	[A10]	Frequency tables for [A10]
RQ_2	[A10] [A11]	Cross-tabulation for [A10], [A11]
	[A12]	Pie chart for [A12]
	[A14] [A15]	Discussion of [A14] and [A15]
RQ_3	[A13]	Frequency tables for [A13]

4.5. Data collection

During the data collection phase, we collected data on a set of variables that describe each primary study. Data collection has also been handled by the first author and has been validated by the second author. If both reviewers assigned the same value to one variable, this value would be assigned to the variable without further discussion. In any other case, a discussion among the authors would result in consensus about the value to be assigned. We have not applied an agreement measure as the number of researchers involved in the review is not significantly large. However, all conflicts have been recorded. For every study, we have extracted the following data:

[A1] Author
[A2] Year
[A3] Title
[A4] Source
[A5] Venue
[A6] Type of Paper (conference / journal)
[A7] Keywords
[A8] Quality Score
[A9] Focus of the research (financial, software engineering, both, neither[4])
[A10] Financial Terms
[A11] Ways of Measuring Technical Debt
[A12] Ways of Managing Technical Debt
[A13] Software Engineering Technologies
[A14] Limitations
[A15] TDM Activity [24]

4.6. Data analysis

The mapping between variables and research questions is provided in Table 1, accompanied by the analysis methods used on the data. Variables [A1]–[A9] are not listed in this table, since they are not used for answering any specific questions, but for demographic characterization of the study.

Specifically, for RQ_1, will be investigated through a frequency table of the identified financial terms. To identify if there are any dominant approaches for quantifying specific financial terms ($RQ_{2.1}$), we will cross-tabulate financial terms found in the selected primary studies against TD measuring approaches. Additionally,

[4] Without in-depth analysis of either the financial or the software engineering aspect.

Table 2
Primary studies per Digital Libraries.

Digital Library	Step 1	Step 2	Step 3
ACM Digital Library	72	19	14
IEEE Xplore	136	29	23
SpringerLink	46	4	1
ScienceDirect	33	5	2
Scopus	107	12	10
Web of Science	16	6	0
Google Scholar	763	34	19
Total	1173	103	69

we will produce a pie chart on the ways of managing technical debt, to visualize the frequency of their application ($RQ_{2.2}$). Finally, to answer RQ_3, we will create a frequency table for identifying the software engineering terms that are more relevant to TDM research.

5. Results

After conducting the article searching and filtering phases, as described in Sections 4.2 and 4.3, we have concluded in the inclusion of 69 primary studies. In Table 2, for each investigated data source, we present the number of papers that have been returned as candidate primary studies (step 1), the number of papers qualified after primary study selection on the basis of title and abstract (step 2), and the final number of primary studies (step 3). On the completion of these phases we examined the quality of each article, based on the guidelines of Section 4.4, and all articles have qualified.

Table 3 illustrates the frequency of specific venues from which relevant articles have been retrieved. The most popular venue in our final set of primary studies in the Managing Technical Debt Workshop (MTD), followed by Cutter IT Journal and IEEE Software. The interdisciplinary nature of technical debt is reflected in the diversity of venues in which primary studies with reference to the financial aspect of technical debt have been identified. Specifically, we retrieved 27 studies from TD-specific venues, 25 studies from software engineering (SE) venues, 8 studies from Information Technology (IT) venues, and 9 studies from generic (GEN) venues such as Communications of the ACM and IBM Journal of Research & Development.

As described in Section 1, technical debt is an interdisciplinary domain, which should ideally deal with aspects of both economics and software engineering. To this end, it is interesting to examine how far the selected studies cover either the economics or the software perspective aspect or both, through the following classification:

- *emphasis on the software engineering aspect of TD* – the study exploits software engineering technologies for managing TD, and uses only basic terminology for referring to the financial aspect of technical debt (see Section 3); An example is the work of Chin et al. [P9] that quantifies the interest of TD on the basis of the number of design smells, without using any financial approaches.
- *emphasis on the financial aspect of TD* – the study adopts a financial approach for handling technical debt (see Section 3), but without an explicit relationship to software engineering technologies; An example is the work of Alzaghoul and Bahsoon [P2] that assesses the future and present value of the amount of technical debt using real options, without providing a clear mapping between the required inputs of real options and characteristics of the examined software.

Table 3
Primary studies per venue.

Venue	Type	Frequency	Nature
Managing Technical Debt	Workshop	27 (40.2%)	TD
Cutter IT Journal	Journal	7 (10.1%)	IT
IEEE Software	Journal	7 (10.1%)	SE
Agile Conference	Conference	2 (2.9%)	SE
Software Quality Journal	Journal	2 (2.9%)	SE
Advances in Computers	Journal	1 (1.4%)	GEN
Agile Processes in Software Engineering and Extreme Programming	Conference	1 (1.4%)	SE
API Design for C++	Book	1 (1.4%)	GEN
Communications of the ACM	Journal	1 (1.4%)	GEN
Workshops of the Computer Software and Applications Conference	Workshop	1 (1.4%)	GEN
Hawaii Int. Conf. on System Sciences	Conference	1 (1.4%)	GEN
IBM Journal of Research and Development	Journal	1 (1.4%)	GEN
Int. Journal of Advancements in Computing Technology	Journal	1 (1.4%)	GEN
Int. Conf. on Building and Exploring Web Based Environments	Conference	1 (1.4%)	GEN
Int. Conf. on Evaluation and Assessment in Software Engineering	Conference	1 (1.4%)	SE
Int. Conf. on Software Engineering	Conference	1 (1.4%)	SE
Int. Conf. on Software Engineering Advances	Conference	1 (1.4%)	SE
Int. Conf. on Software Engineering and Mobile Application Modelling and Development	Conference	1 (1.4%)	SE
Int. Conf. on Software Maintenance	Conference	1 (1.4%)	SE
Int. Conf. on Software Testing, Verification and Validation	Conference	1 (1.4%)	SE
IT Professional	Journal	1 (1.4%)	IT
Object Oriented Programming Systems Languages and Applications	Conference	1 (1.4%)	SE
Procedia Computer Science	Journal	1 (1.4%)	GEN
Proceedings of the Third C* Conf. on Computer Science and Software Engineering	Conference	1 (1.4%)	SE
Quality of Software Architectures	Conference	1 (1.4%)	SE
SIGSOFT Software Engineering Notes	Journal	1 (1.4%)	SE
Symposium on Empirical Software Engineering and Measurement	Symposium	1 (1.4%)	SE
Working Conference on Software Architecture	Conference	1 (1.4%)	SE
Workshop on Future of Software Engineering Research	Workshop	1 (1.4%)	SE

- *emphasis on both software engineering and financial aspects of TD* – the study adopts a financial approach for managing technical debt (see Section 3), and provides a detailed analysis on how software engineering technologies can be used; An example is the work of Seaman et al. [P53] that suggests the management of TD using Cost-Benefit analysis, Portfolio Management and Real Options considering as inputs design smells which are present in software.
- *emphasis on neither software engineering nor financial aspects of TD* – the study is using only the basic terms for referring to the financial aspect of technical debt (see Section 3), and does not exploit software engineering technologies. An example is the article by Rooney [P47] that discusses the need for a new metaphor of technical debt claiming that the metaphor is inappropriate for systems built with modern development approaches such as Agile practices.

The results indicated that 56% of the studies that examined technical debt emphasize on the SE aspect of technical debt, without specific focus on any financial aspect (i.e. other than the basic terms discussed in Section 3). Next, in about 22% of the studies the two aspects were balanced, i.e. employed financial approaches for managing technical debt, by using some software engineering

Table 4
Financial terms used in technical debt research.

Term	Frequency	Percent (%)	Term	Frequency	Percent (%)
Interest	29	42.03	Compound interest	2	2.90
Principal	17	24.64	Benefit	2	2.90
Cost	8	11.59	Bankruptcy	2	2.90
Repayment	7	10.14	Future value	2	2.90
Return on investment (ROI)	7	10.14	By-product	1	1.45
Asset	5	7.25	Total cost of ownership (TCO)	1	1.45
Investment	4	5.80	Depreciation	1	1.45
Value	4	5.80	Effort	1	1.45
Risk	4	5.80	Financial leverage	1	1.45
Present value	3	4.35	Cash flow	1	1.45
Productivity	3	4.35	Hedging	1	1.45
Opportunity cost	3	4.35	Loan shark	1	1.45
Business value	3	4.35	Opportunity benefit	1	1.45
Option	3	4.35	Voice of market	1	1.45
Liability	3	4.35	Savings	1	1.45
Interest rate	2	2.90	Value-added	1	1.45
Revenue	2	2.90	Voice of business	1	1.45
Capital	2	2.90	Voice of customer	1	1.45
Net present value	2	2.90			

technologies. However, we note that the level of detail in one of the two parts might lag compared to the other (see discussion on Section 5.2). Finally, research efforts that mainly emphasize on the use of financial approaches are rather limited, i.e. about 10% of the primary studies, while the rest 12% of the studies can be classified as having limited emphasis on both the software engineering and the financial aspect of TD. The rest of this section presents the results of data collection and analysis, organized by research questions.

5.1. Financial terms related to technical debt management (RQ_1)

By focusing on the terms used to describe the financial aspect of TD, the results are summarized in Table 4. For each term, we list the frequency (i.e., the number of distinct studies in which the term is being used), and the corresponding percentage over the total number of primary studies. We note that one paper, might employ more than one terms; in this case all relevant terms have been recorded. However, for a paper to be counted, the corresponding term should form an integral part of its discussion or methodology.

We observe that *interest* and *principal* are the most frequently used ones. Additionally, we can observe that in many cases, conceptually similar terms may not be used in the primary studies under a uniform term. For example: the terms *value* and *business value* are both used in different studies with similar meaning. We also note that some of the basic financial terms presented in Section 3 (i.e., debtor and creditor, debt instrument, maturity date, and liquidity) are not investigated in technical debt literature. This raises the question whether these terms have been deliberately left out or whether they can be applied in the technical debt domain. For example, can *technical debt have a maturity date* or *does liquidity constitute a factor on the management of technical debt?*

Although we acknowledge the fact that some of these terms might have already been used in related research areas such as software economics (e.g., net present value of software development strategies [12], valuation of software refactorings [27], etc.), their results have not yet been highly exploited by the technical debt community. To this end, we suggest that the research community of technical debt might benefit from reusing research results from other related domains (some examples for measuring technical debt by using software economics [5] are presented in Section 5.2).

5.2. Financial approaches for managing technical debt (RQ_2)

According to Li et al. [24], technical debt management, can be decomposed to five activities: identifying, measuring, prioritizing, repaying, and monitoring TD. Since measurement forms the basis for any financial handing of TD, we separately discuss financial approaches w.r.t. measurement (see $RQ_{2.1}$), and group the discussion on all other TDM activities (see $RQ_{2.2}$).

5.2.1. Financial approaches for measuring technical debt

In the table of Appendix B,[5] for each financial approach related to technical debt measuring we provide: (a) a pointer to the corresponding primary studies, (b) a brief description of the used method, (c) the software engineering technologies that they use as input, (d) the measured financial terms, (e) the used financial approaches, and (f) the TDM activity that they refer to. As financial approaches we consider all the approaches that have been presented in Section 3, under *Managing Debt Strategies*. In case a study only used software engineering tools or methods to quantify technical debt (i.e. the approach is not financial), it has been classified as based on *Software Economics*. For example, a study would be characterized as using software economics for measuring technical debt if:

- it considers the number of bad smells as an estimate of accumulated technical debt, and then valuate this number in terms of money that have to be spent for resolving them; or
- it uses software effort/cost estimation methods (size, cyclomatic complexity, etc.) for assessing the effort/cost required to enhance particular system design-time qualities (e.g., maintainability) of a software component.

To provide an overview, we synthesize the data from Appendix B through cross-tabulating the measured financial terms and the corresponding approaches, as presented in Table 5. Similarly to Table 4, frequency refers to the number of papers, in which the corresponding terms are measured. From Table 5, we can observe that most financial terms have been quantified through software economics methods (73.1% or 19 out of 26), whereas the most frequently quantified terms are *amount of debt* (*i.e. principal + interest amount*) and *interest*.

[5] Table has been moved to Appendix B to improve the readability of the paper.

Table 5
Measuring technical debt approaches.

Measured term	Frequency	Measuring approach			
		Portfolio management	Real options	Software economics	Value based
Amount of debt	9	0	1	7	1
Compound Interest	1	0	0	1	0
Interest	6	1	0	5	0
Repayment	1	0	0	1	0
Principal	4	1	0	3	0
ROI	1	0	0	0	1
Simple Interest	1	0	0	1	0
Future Value	1	0	1	0	0
Present Value	1	0	1	0	0
Total	26	2	3	19	2

Based on the inherent financial nature of TD, an approach that measures technical debt is expected to calculate the amount of technical debt (or related terms) as values in money or other similar units. By further focusing on the level of granularity of the items that these approaches receive as inputs, we can identify two basic trends:

- Approaches that receive ***coarse-grained inputs,*** i.e., inputs that have to be estimated based on a number of factors. For example, in this category we classify approaches that use effort as input, in the sense that effort spent on several activities (e.g., identification of code smells and application of refactorings) is related to the size of the artifacts, the number of bad smells, the intensity of the bad smells, the urgency to solve them (based on artifacts history), etc.
- Approaches that receive ***fine-grained inputs,*** i.e., inputs that can be estimated directly without taking other factors into account. For example, in this category we classify approaches that use as input the number of code smells, in the sense that the number of code smells can be measured directly from the code.

Approaches receiving coarse-grained inputs for measuring technical debt usually originate from economics (traditional or software economics). In the domain of cloud-computing, Alzaghoul and Bahsoon define the amount of debt as the *additional effort needed to provide scalability to a SOA* [P2]. On the other hand, concerning traditional software development, de Groot et al. [P22], Nord et al. [P41], Nugroho et al. [P42], and Stochel et al. [P60] define the amount of debt as the rework cost that is needed to improve the levels of quality. Building on that, Schmid [P50] specifies that in order to quantify TD, you need to assess the effort needed to refactor a software product so as to have zero debt. In a similar context, interest is defined as the difference in the amount of technical debt. Chin et al. [P9] define simple interest as the *ongoing product maintenance cost* and compound interest as the *increase in the amount of technical debt over time*.

Concerning approaches that receive fine-grained inputs, Guo and Seaman [P23] and Guo et al. [P24] use historical data on effort, so as to estimate the effort needed for bug-fixing or increasing test coverage, as indications of technical debt. Curtis et al. [P12] count code violations and classify them based on their severity (high, moderate and low), and then calculate the total time needed for fixing these violations. Finally, Santos et al. [P49] use cohesion and coupling metrics, code duplications, lack of comments, coding rules violation, potential bugs, the absence of unit tests, etc. for assessing code technical debt.

It becomes clear that all methods used for measuring TD that are solely based on software economics, are not related to any approach related to financial debt. On the other hand, all studies that apply financial approaches to technical debt measurement appear to be more balanced. Specifically, Alzaghoul and Bahsoon [P2] apply Real Options Analysis and use a two-step binomial tree approach in order to quantify technical debt, in the case of web services substitution. Technical debt is considered to be the difference between the option's value and the cost of switching to a new web service. The value of the system, the option price, the option value, and the probability coefficient are derived through functions, using values based on historical data. However, the use of interest rate (r) in the model is not clearly defined.

In addition, Guo and Seaman [P23] measure technical debt principal and expected interest amount in person-days as the effort needed to resolve a technical debt item and the extra effort required to complete the task later, respectively. In order to fit their measurements in a Portfolio Management model, they consider TD items as assets and they also determine the likely net benefit of the item and the risk that the item will not produce benefit. The likely net benefit is considered to be the expected asset return and is defined as the principal minus the expected interest amount. On the other hand, in Portfolio Management theory, the risk is considered to be represented by the variance of return. The authors suggest that, in the case of TD, this is equal to the interest standard deviation. All estimations are based on historical effort, usage, change, and defect data.

Finally, Stochel et al. [P60] propose a three-layered model and define codebase/design debt, architectural debt and portfolio debt. They propose the SonarQube approach and the Wisdom of Crowds technique for the measurement of design debt. Regarding architecture debt, they propose that its estimation should be based on the cost of expected changes in the architecture. At portfolio layer, technical debt is defined as the variation of the cost of change in order to achieve the highest ROI.

5.2.2. Financial approaches for identifying, prioritizing, repaying and monitoring technical debt

In contrast to approaches that have been introduced for measuring technical debt, in which only limited studies use financial approaches while quantifying technical debt, in the rest of the management activities, we observe that the financial aspect is more evident. In the table of Appendix C, we present the financial approaches that have been proposed for managing technical debt. Specifically, we provide: (a) a pointer to the corresponding primary studies, (b) a brief description of the used method, (c) the software engineering technologies that they use as input, (d) the used financial approaches, and (e) the TDM activity that they refer to. The results in Appendix C are synthesized and visualized in the pie chart of Fig. 1.

In addition, we also investigated the financial approaches that have been used for managing technical debt in terms of: (a) the software engineering technologies that have been used as input/output of these approaches (if applicable), and (b) the potential omissions or assumptions that researchers have done while adopting the financial approach in the technical debt domain. In the next paragraphs, we present more details about the three most popular approaches for managing technical debt: cost/benefit analysis, real options analysis and portfolio management.

5.2.2.1. Cost/benefit analysis. Regarding cost/benefit analysis theories that have been transformed so as to fit the management of technical debt, we were able to identify six relevant studies. Buschmann [P8] presents some industrial case studies, indirectly referring to cost/benefit analysis by presenting real-world examples on the advantages and disadvantages of repaying technical

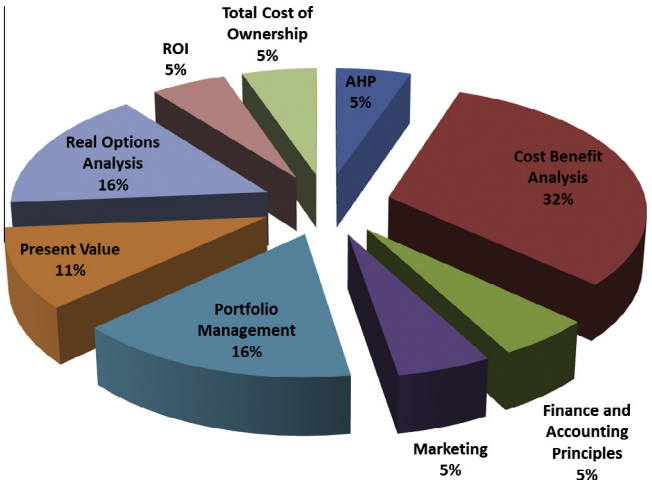

Fig. 1. Managing technical debt approaches.

debt, without however using the original terminology implied by the financial method, or presenting software engineering approaches to assess the cost and benefits of technical debt repayment. On the contrary, Guo et al. [P24] and Seaman et al. [P53] have employed a cost/benefit analysis in a way in which costs have been related to interest amount (calculated based on interest probability and changeability analysis) and benefit has been related to the principal. However, the amount of the principal is assumed to be known, and is not calculated in the paper. To support these claims the authors present a small size case study.

In addition, Schmid [P50] presents a theoretical framework that enables the application of cost/benefit analysis during software evolution. Initially he suggests that cost is related to development effort, to product time-to-market, and to customer benefits. However, in the rest of the paper he only focuses on development costs, by taking into account the cost of refactorings as part of iteration costs. As benefit of repaying technical debt he characterizes saved cost by technical debt item, by iteration. For calculating cost and benefit, Schmid is based on existing software cost estimation models, and clearly notes that the accuracy of his model cannot be higher than the accuracy of the used cost estimation model [P50]. Furthermore, Snipes et al. [P58] propose the use of cost/benefit analysis for making decisions on bug fixing prioritization. In this study, the authors propose a method for calculating the cost of each defect fixing activity (based on type, condition and change proneness), and the expected benefit from fixing the defect (based on severity, existence of work around, etc.). The specific application of the method is sound, since it uses the cost/benefit analysis ratio, based on well-defined parameters [P58]. Finally, Zazworka et al. [P66] implement a cost/benefit matrix in order to identify and prioritize technical debt items (identified God classes [26]) refactoring. The cost of each refactoring opportunity is assessed by using detection strategy metrics (e.g. Weighted Method Count – WMC, Tight Class Cohesion – TCC and Access To Foreign Data – ATFD), whereas the benefit is quantified by class change and defect proneness.

5.2.2.2. Real options. Next, concerning studies that employ the real options theory, Alzaghoul and Bahsoon [P2] describe asset value as the quality of service (e.g. scalability and availability), and option price for leaf nodes as the difference between system value and maintenance cost. On the other hand, the probability coefficient of improvement, the expected pay-off when the value goes up, the expected pay-off when the value goes down, and the interest are not directly associated with software technologies. However, the authors state that such input parameters can be assessed by stakeholders input, historical data, or be based on existing valuation methods. In contrast to the previously well described application of the real option theory, Power [P43] characterizes the time that development teams spent on paying off technical debt as an option and investigates its impact on team's capacity and velocity, without however defining any of the aforementioned input variables or applying the real options theory in practice. Finally, Seaman et al. [P53] also described that paying-off technical debt (e.g. through refactorings) can be managed as an investment with short-term cost (e.g. resources, effort on applying refactorings, etc.) that can be valuated through the decrease of technical debt in the long term. However, in this short paper the major focus of the authors was not to present in detail how a financial approach could be applied, but to debate on the fitness of several financial methods and their relationship to technical debt.

5.2.2.3. Portfolio management. With respect to studies that attempt to manage technical debt through portfolio management theory, we have been able to identify three related papers. Guo and Seaman [P23] and Seaman et al. [P53], use technical debt items as assets and populate the portfolio of the software development organization with them. In an iterative way, each item is assessed and is either preserved or excluded from the portfolio. In each iteration, the method suggests excluding from the portfolio technical debt assets that should be repaid. The decision to repay an item or not is based on risk assessment. The risk assessment is being performed based on the principal, estimated interest rate, interest standard deviation and dependencies between technical debt elements. Finally, Stochel et al. [P60] suggest that the debt portfolio is layered into process debt, architecture debt, and design/source code debt portfolio. Each one of these distinct portfolio technical debt items, should be individually assessed (e.g. for architecture technical debt by using ATAM, and for source code debt by using methods/tools like the Wisdom of Crowds).

5.3. Software engineering technologies used by financial approaches in TDM (RQ_3)

Table 6 presents the frequency of software engineering technologies that have been used in financial approaches in technical debt management. We note that as software engineering technologies we collectively refer to any element in the backbone of the software development process according to RUP [22] (i.e., tools, software artifacts, development activities/phases), as well as in quality assessment/improvement process (e.g., quality attributes, quality measurements, refactoring opportunities, etc.). Such elements can provide information to be fed as input to the applied financial approaches. For example, Schmid [P50] applied the cost/benefit analysis, to decide whether technical debt items should be paid off, considering as input the refactoring cost, the relative technical debt per refactoring and per evolution step, and also the probability that a particular evolution step will be performed. According to the aforementioned definition of software engineering technologies, in Table 6, we organize the data in the following categories: *tools, artifacts, phases/activities, quality attributes* and *quality assurance-related terms* (e.g., quality models). We note that terms that fit two categories (e.g., design can refer to both be *design activity* and the *design artifacts*), have been classified to both classes.

6. Discussion

The results of the study provide useful insights towards the two goals set in Section 1, the glossary of terms and the classification of

Table 6
Software engineering technologies used in technical debt research.

Group	Term	#	Percent (%)	Group	Term	#	Percent (%)
Tools	SonarQube	6	8.51	Phases /Activities	Implementation	17	24.28
	AIP	2	2.90		Static analysis	15	21.42
	CodeVizard	2	2.90		Testing	11	15.71
	FindBugs	2	2.90		Maintenance	11	15.71
	CLIO	1	1.45		Design[a]	9	12.85
	DebtFlag	1	1.45		Architecture[a]	6	8.51
	Microsoft Tree Mapper (tool)	1	1.45		Agile development	6	8.51
Artifacts	Source code	17	24.28	Phases /Activities (cont.)	Refactoring	6	8.51
	Design[a]	9	12.85		Requirements[a]	4	5.71
	Architecture[a]	6	8.51		Evolution analysis	2	2.90
	Requirements[a]	4	5.71		Dynamic analysis	2	2.90
	Code smells	3	4.28		Project management	2	2.90
	Design smells	3	4.28		Acceptance testing	1	1.45
	Design patterns	3	4.28		ATAM	1	1.45
	Incomplete documentation	1	1.45		Commonality-variability analysis	1	1.45
Quality attributes	Maintainability	11	15.71	Quality assurance	Quality dashboard	2	2.90
	Defect-density	10	14.28		SIG/TUV	2	2.90
	Functionality	2	2.90		SQALE	2	2.90
	Reusability	1	1.45		ISO/IEC 25010	1	1.45
	Performance	1	1.45		ISO/IEC 9126	1	1.45
					COCOMO	1	1.45

[a] These terms are duplicate in the table.

approaches. In this section we synthesize the abovementioned results into the two envisioned goals: a glossary of well-defined descriptions of all financial terms that have until now been connected to technical debt (see Section 6.1 – related to goal-a); a classification schema of the application of financial approaches in technical debt research (see Section 6.2 – related to goal-b).

6.1. Technical debt financial glossary

In this section we provide a glossary of the financial terminology that is used in technical debt research, presented in Table 4. In order to compose the glossary, we have used the terms and their meanings from the primary studies. We derived the definition for each term, by synthesizing the way that these terms are used in several studies. In cases of conflicting definitions (explicit or implied) the terms are defined based on our own perception of the way that these concepts could prove beneficial for technical debt management (see Fig. 2). We also performed a merging of terms that were similar in meaning. From the glossary of Fig. 2, we have intentionally excluded the following terms, because their meaning in TD is completely equivalent to their meaning on financial debt: benefit, cash flow, cost, depreciation, hedging, investment, loan shark, option, opportunity cost, productivity, savings, voice of business, voice of customer, voice of market, and total cost of ownership. Finally, we note that the original (i.e., financial) definitions of terms have been omitted, although they might hold in the TD context. For example, the software product per se is an asset for the company, since the company can sell it and make value from it; however, this value is outside the context of TD.

Next, all terms of the glossary are discussed in detail, and graphically illustrated in Figs. 3a–3c. The intention of these Figures is not to quantify the terms, but to provide a graphical context in order to facilitate their understanding. To further enhance the readability of the discussion, we group the terms, based on if they are applicable to:

- the pre-deployment phase when technical debt is accumulated – Fig. 3a;
- the post-deployment phase, considering changes related to the implementation of new requirements – Fig. 3b;
- the post-deployment phase, considering changes related to the improvement of system design-time qualities – Fig. 3c.

We note that the values in all axes of Figs. 3a–3c are illustrative and do not originate from any empirical data, or reference. In addition, the Figures illustrate individual artifacts as items that are subject to technical debt (see for example [P23], [P53], [P60], and [P66]); however, TDM can also be performed at system-level (e.g., see for example [P2]). The depictions and explanations are equally applicable to system and artifact level.

In Fig. 3a, we assume a software system that is composed of 7 artifacts (e.g. software components). Artifacts 2 and 3 have been developed on the desired levels of design-time quality, whereas in all other artifacts several compromises have been made ($\delta_{quality}$). The immature software artifacts are named *technical debt items* or *liabilities*. While developing the aforementioned artifacts the development team spent less effort than it was required in the optimal case resulting in $\delta_{quality}$. The effort that is required to address this difference in the levels of quality is termed *principal* (or *capital*) of the TD. Principal can be considered as an *asset* for the company, since it can be invested in any other activity due to *financial leverage*. Such activities can be the development of *by-products* or *decreased time to market*. The value of such by-products divided by the principal represents the *return of investment (ROI)* of investing the principal into other activities.

In Fig. 3b, the *y*-axis refers to effort instead of quality. We suppose that the same system, after its deployment, requires the addition of a feature. For simplicity, we assume that this feature is global and the same effort needs to be spent in every component (artifact). However, in the artifacts where technical debt is accumulated (technical debt items), additional effort (δ_{effort}) is required because of their deteriorated design-time quality (e.g. low maintainability, bugs that are found during extension, incomplete documentation that lead to low understandability). This additional effort is the *interest* that the development team has to pay, due to the accumulated amount of debt. If the interest becomes so high that maintenance is not financially feasible or beneficial, the project becomes *bankrupted*.

In Fig. 3c, we consider the evolution of a system with accumulated technical debt (accumulated technical debt is calculated as: *principal* + *interest*). The two series in the line chart represent the evolution of the system without any repayment activity (red[6] line

[6] For interpretation of color in Figs. 3c and 4, the reader is referred to the web version of this article.

Asset:	*1. The effort that the development team saves by producing immature artifacts while accumulating technical debt.*
	2. Tools or approaches that lead to the decrease of technical debt.
Bankruptcy:	*A software engineering project could be considered bankrupt, in case that it fails to survive/evolve (either being cancelled or forced to be re-written from scratch) due to large maintenance costs, caused by the accumulation of technical debt.*
By-product:	*An additional software product that can be produced with the effort that has been saved from producing immature artifacts while accumulating technical debt.*
Financial Leverage:	*The ability of software companies to produce by-products, decrease time to market, etc. by accumulating technical debt.*
Future Value:	*The value that an action (e.g. undertaking technical debt) performed now will have after a given time period.*
Interest:	*The additional effort that is needed to be spent on maintaining the software, because of its decayed design-time quality.*
Liability:	*TD items, i.e. artifacts that have not been developed with the optimal design-time quality and thus subject to improvement.*
Present Value:	*The value that an action (e.g. a repayment activity) to be performed in the future has now.*
Principal:	*The effort that is required to address the difference between the current and the optimal level of design-time quality, in an immature software artifact or the complete software system (syn: capital).*
Repayment:	*The amount of effort spent on improving design-time quality. This effort will decrease the effort needed for future maintenance tasks.*
Risk:	*The probability or threat that the technical debt items are accumulated in a design hotspot. Immature artifacts in such places of the system can hinder a product's viability.*
ROI:	*The ratio of (1) the additional amount of money that is earned by bringing the product earlier into the market or (2) the additional amount of money that has been earned from the company by investing the effort of the principal in an activity different than the improvement of design-time quality, over the principal.*
Value-Added:	*The additional value that repayment of technical debt would bring to the software product.*

Fig. 2. Technical debt financial glossary.

– inherent) and with one repayment activity performed in revision-4 (blue line – with repayment). We suppose that the system starts with an amount of debt that increases over time (interest rate is represented by the slope of each line, e.g., $\theta 1$ and $\theta 2$). As we observe, in the case of Fig. 3c, the interest rate is not *stable*, but *floating*, since the slope of the red line is increasing over time (i.e., $\theta 2 > \theta 1$ and $\varphi 2 > \varphi 1$). This increase is expected, since the design-time quality of decayed projects, deteriorates quicker than the quality of better-designed products (the poor getting poorer). Therefore, the *TD risk* for low design-time quality products is higher than the TD risk of high design-time quality products. For this particular example, we assume that the interest rate increases, not because of the introduction of new flaws or bugs, but due to the increased difficulty of resolving existing problems, which in turn is caused by the gradual increase in size and functionality. Due to the *repayment* actions in revision-4, some artifacts' design-time quality is increased, i.e. technical debt is decreased (δ_{TD1}). The effort that is spent during this repayment activity is the *value of repayment*. Furthermore, because of the floating rate of the interest (it increases in revision-3 (TD = 50) and in revision-6 (TD = 75)), we can see that in revision-7 the value of repayment increases, as illustrated by the distance between the two lines ($\delta_{TD2} > \delta_{TD1}$). The value of the repayment performed in revision-4, is named *future value of repayment*, in the timestamp of revision-7. Assuming that present time is revision-2 and that a future repayment activity (e.g. the one performed in revision-4) should be valuated at present conditions, one can assess at revision-2 the *present value of the repayment* to be performed in revision-4. Finally, according to de Groot et al. software value is, among

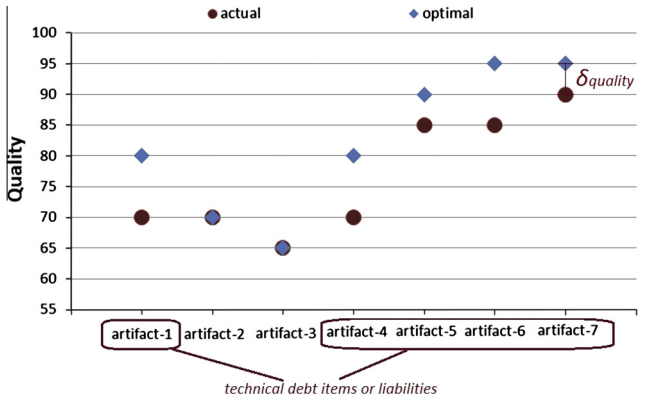

Fig. 3a. Pre-deployment phase.

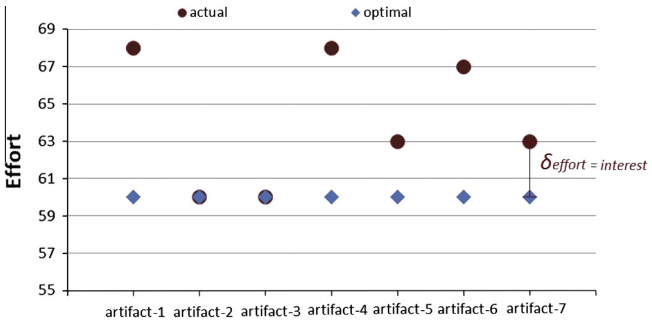

Fig. 3b. Post-deployment changes to implement new requirements.

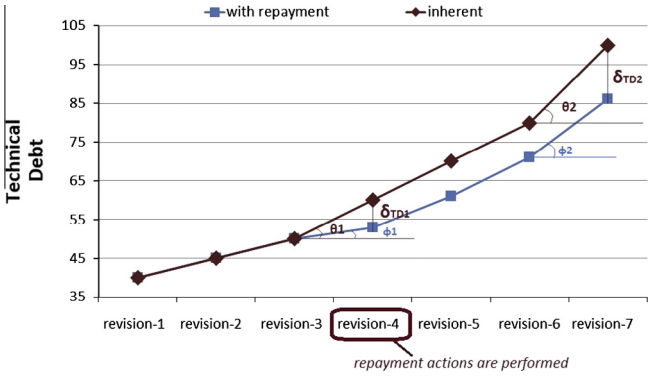

Fig. 3c. Post-deployment changes to improve design-time quality.

others (e.g., volume/functionality), related to product quality (i.e., inherent characteristics of the software) [P22]. Therefore, the enhancement of design-time quality that is achieved through the repayment on revision-4 represents the *value-added* of TD repayment.

6.2. Classification scheme for financial approaches used for managing technical debt

In this section we present a classification scheme for techniques on managing technical debt (see Fig. 4). The classification has been built based on the raw data of this SLR (presented on Appendices B and C). For readability reasons, while developing the classification schema we preferred not to list the primary studies that should be mapped to each edge. In any case, the interested reader can find all relevant information in Appendices B and C. According to Nickerson et al., the most common paradigm for building classification schemas for information systems is the three-level indicators model, which is based on both empirical and deductive approaches [29]. By applying this model, we: (a) examined the objects (i.e. studies), (b) we identified general distinguishing characteristics of the objects, and (c) we grouped their characteristics so as to create our classification schema [29]. Specifically, in step (b) we identified three *characteristics* that will constitute the three levels of the proposed schema:

- the 1st level of the schema represents the existing activities within technical debt management [24] – extracted variable: [A15] – last column of Appendices B and C;
- the 2nd level represents the proposed financial techniques – extracted variables: [A11] and [A12] – 5th column of Appendix B and 4th column of Appendix C;
- the 3rd level corresponds to the used software engineering technologies – extracted variable: [A13] – included in the 3rd column of Appendices B and C.

In addition, in the proposed schema the popularity of each technique in level 2, and of each software engineering technology in level 3, can be deduced by the number of edges reaching the corresponding node, respectively. More specifically, the lines connecting the categories of level 1 to the approaches of level 2 and the latter to the technologies of level 3 indicate which elements are connected to each other in the primary studies. In case more than one edges connect two nodes, a multiplicity symbol is used. For example, the *cost/benefit analysis* has been encountered in 3 studies discussing the *repayment* of technical debt. To improve the readability of Fig. 4, edges leaving from the same financial approach (at level 2) are grouped through a common line style.

Moreover, we evaluate each technique of level 2, w.r.t. the cross-examination of software engineering technologies and financial approaches. Specifically, the number of arrows under the names of each financial approach denotes how many times it is applied, regardless of the technical debt management activity (e.g., *value-based* approaches have been applied 3 times in TD research). Approaches marked with upwards-pointing green arrows are adequately applied by focusing on both financial and software engineering aspects; approaches marked with downwards-pointing red arrows miss one aspect (either financial or software engineering); those marked with a horizontal yellow arrow present a mapping between software engineering and financial aspects, but their explanations are not detailed. For example, the *cost/benefit* analysis has been adequately applied in 4 studies, by relating financial and software engineering aspects, in 1 study both aspects are discussed but valuation of software activities are not thoroughly discussed, while in 1 study the authors have focused only on the financial aspect of technical debt. As evaluation for the proposed classification schema, we attempted to validate the four desirable attributes for successful classification schemas, as proposed by Nickerson et al. [29]. According to Nickerson et al., a successful classification should ensure:

- ***conciseness:*** contain a limited number of dimensions or a limited number of characteristics in each dimension [29];
- ***inclusiveness:*** contain dimensions and characteristics to be of interest [29];
- ***comprehensiveness:*** provide for classification of all current objects within the domain under consideration [29]; and
- ***extendibility:*** allow for additional dimensions and new characteristics within a dimension when new types of objects appear [29].

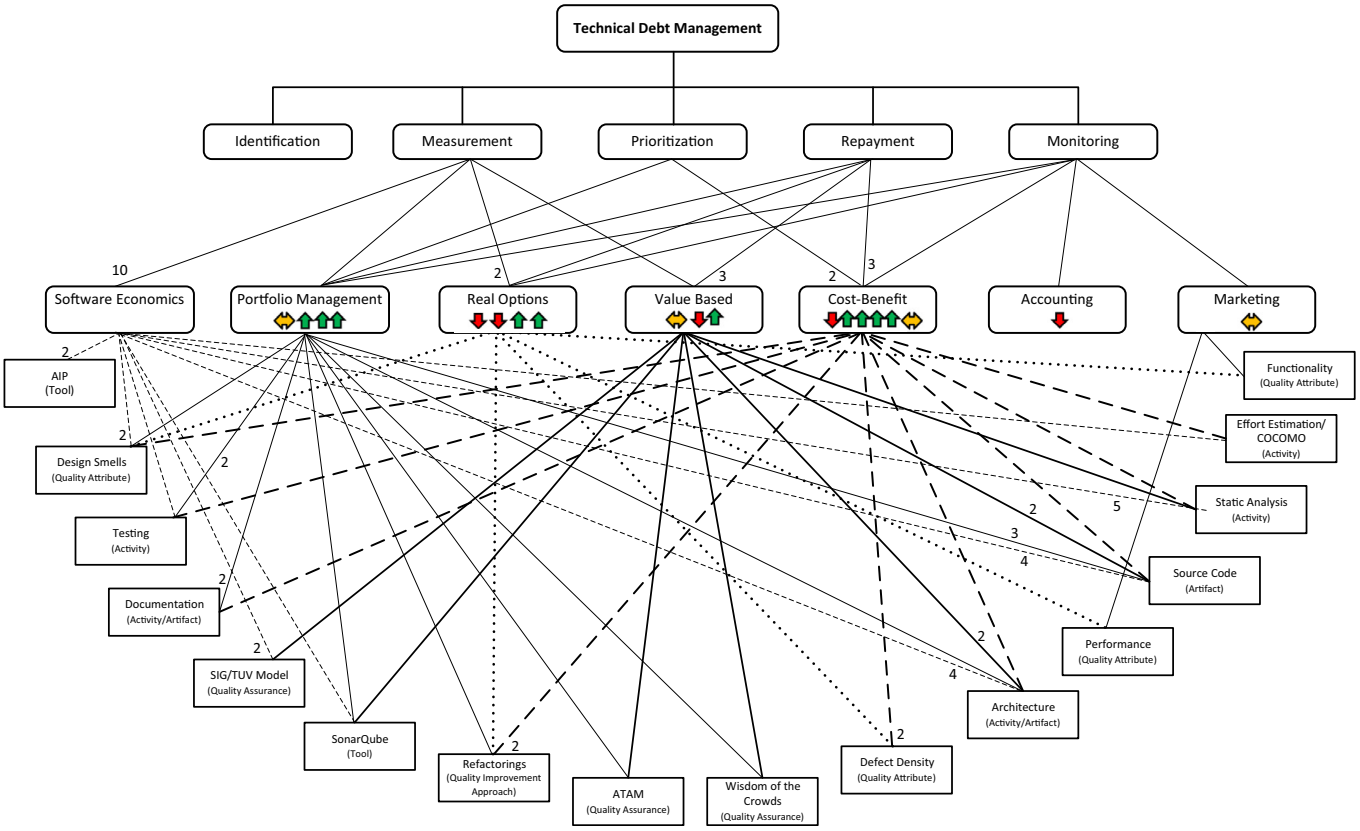

Fig. 4. Financial approaches classification schema.

We note that conciseness can sometimes be compromised in favor of inclusiveness. Therefore, we selected a reasonable depth for our classification schema (i.e., 3) and a small amount of characteristics in the first two dimensions (i.e., 5 and 6 respectively), so as to provide a *concise* schema that will be understandable and applicable in practice. On the other hand, we preferred to include a large amount of classes in the third level, so as to be as *inclusive* as possible with respect to software engineering technologies. Additionally, the *comprehensiveness* of the proposed schema is guaranteed by the fact that we were able to classify each study in exactly one category. Finally, *extendibility* is related to the effort required to extend the schema: adding classes is an easy extension, while adding levels can be far more complicated since it can result in changing the existing mapping between classes and levels. In this sense, we believe that the proposed three levels of the schema (TDM activities, financial approaches, and software engineering technologies) are sufficiently generic to reflect a large body of research in TD without requiring additional levels. On the other hand, we expect additional classes to be added, representing novel financial approaches and software engineering technologies that will be proposed in future research.

From the first level of the schema (i.e., TDM activities) we can identify classes, i.e. research topics that need further investigation. First, we can observe that technical debt identification is not connected to any financial approach, whereas research on prioritizing is also neglected, compared to monitoring, repaying and measuring technical debt. Repaying technical debt is in most cases explored through cost/benefit analysis, which however sometimes is not adequately applied (see Section 5.3 for details). Additionally, real options, although quite popular as a technique for managing technical debt, they: (a) seem to be used without a clear mapping of financial and software engineering technologies and (b) are in many cases applied in an artificial rather than in a real-world setting. Finally, concerning the application of financial approaches on TDM, we observe that researchers do not uniformly map software engineering technologies to the inputs required by the financial methods. For example, in the cost/benefit analysis, researchers use a variety of software engineering technologies, which indicates that there are many alternatives that can be used as inputs (e.g., refactoring opportunities at the architectural level, smells at the source code level, quality metrics, etc.).

6.3. Implications for researchers and practitioners

Based on the expected contributions (see Section 1), we believe that the reported results can prove beneficial to both researchers and practitioners. Firstly, we encourage software practitioners:

- to use the "correct" financial terms while communicating their technical debt concerns, to managers. Under this perspective, software engineers are expected to find the technical debt glossary a useful tool that will enhance communication issues;
- to use the classification for guidance on how each technique can be applied and for which management activity. For example, measuring is usually done with software engineering methods, prioritizing and monitoring with portfolio management, and repaying with cost/benefit analysis;
- to use the classification for mapping the inputs and outputs of the financial approaches to software engineering technologies. For example the classification suggests that prioritization of technical debt items can be performed through cost/benefit analysis, applied at both source code and architecture level. The benefit can be quantified by measuring the improvement resulting by structural changes (e.g. refactoring) or defect

elimination (debugging). In this example, design-time quality of the product and number of defects are mapped to the inputs of the financial approach.

Secondly, regarding researchers, the results of the study indicate that there are specific issues in technical debt research that need further attention:

- at this point the use of financial terms is rather ambiguous. Therefore, we prompt researchers to "correctly" use financial terms (as close to their original definition as possible), and clearly describe any possible deviations from them;
- technical debt research is interdisciplinary. Therefore, researchers should present both aspects (financial and software engineering) and make the connection between the two visible. In addition, although, we acknowledge that in most cases it is inevitable to focus on one aspect, the omission of the other should be discouraged;
- the application of a financial method on software is not an easy task. Most software engineers lack in-depth knowledge of financial approaches, and should spent more time on understanding the method and apply it as appropriately as possible, by clearly explaining the mapping of terms of one discipline on the terms of the other;
- there are specific financial aspects (i.e., terms) of technical debt that have not been quantified or taken into account in technical debt research (compare results of Table 4 to those of Fig. 2). However, these terms are transferable to technical debt research, and by using them, a new range of approaches becomes applicable to technical research;
- the research domain of technical debt management is in need of a common reference point from a terminology perspective. For example, in this study we used and extended the classification of technical debt activities proposed by Li et al. [24]. However, in other studies, such references appear to lack. To this end, we present a glossary that can be used as a starting point for the construction of widely accepted list of terms, and method classification.

7. Threats to validity

In this section, we discuss possible threats to the validity of our study. To the best of our knowledge there is no established categorization of threats to validity for secondary studies, in contrast to other types of empirical research, e.g. case studies [33], or experiments [37]. Therefore, we organize this section in four major categories: (a) threats to identification of primary studies, (b) threats to data extraction, (c) threats to generalization of results, and (d) threats to conclusions.

7.1. Threats to identification of primary studies

Threats to the identification of primary studies deal with possible limitations of the article search process that can lead to missing related literature. In our search process, any study that does not mention the word "technical debt" in the title, abstract or keywords of the article has been excluded from the primary studies set. So, a number of articles that deal with technical debt might have been omitted. However, we believe that papers that focus on technical debt would most probably explicitly state it in their titles, abstracts or keywords. Other papers that use the term in the full text are probably just making a brief mention to it, without leading to loss of valuable data for our research.

7.2. Threats to data extraction

Threats to data extraction deal with possible problems that might arise in the data collection phase. The most common threat during this phase is the subjectivity of the researcher who performs the data collection. In order to mitigate this threat we developed a data collection validation mechanism that involved at least two researchers, as it is presented in Section 4.5. In addition to that, especially for extracting the data for variables [A11] and [A12] (i.e. ways of measuring and managing technical debt), the data extractors were advised to try to use the classification scheme proposed by Li et al. [24], so as to avoid the use of synonyms and ease the data analysis phase.

7.3. Threats to generalization

Threats to generalization refer to threats that occur while generalizing the results from our sample to the population. In this SLR, we identified three such threats. First, the results on the evaluation of the applicability of financial approaches on technical debt only represent the way that they are used in the primary studies that we have explored, and do not capture the applicability of the approaches in general. Second, both the classification schema and the glossary are only representing the current status of technical debt research, and therefore are prone to change when additional primary studies will be published. Third, although the discussions presented in this study (i.e., the glossary of terms, the classification schema) are heavily dependent on the raw results of this SLR (i.e., a plethora of published primary studies), we acknowledge that subjective decisions (e.g., resolution of conflicting term definitions, merging of terms) have to be validated in practice. It should be noted that since about one fourth of the primary studies originate from a particular workshop (i.e., MTD), the results might overrepresent the research of this particular community. This a rather expected outcome, in the sense that MTD is devoted to studies on technical debt, and TD is a relatively new area of research.

7.4. Threats to conclusions

Threats to conclusions validity are factors that can lead to incorrect conclusions, either by identifying incorrect relationships, or by missing existing relationships. In this category, we have been able to identify two possible threats to validity. First, we measure research intensity with the number of studies and not with the volume of research or information they provide. However, such an attempt would introduce subjective criteria into the analysis of the results. For this reason, we preferred to use an objective (directly measurable) criterion for characterizing the intensity of research. Second, we evaluate the completeness of the approach based on if the study uses sufficiently both aspects of software engineering research (i.e. financial and software engineering) and not a method for level of evidence [1] or rigor [17]. The reason for preferring this type of assessment in favor of evaluating level of evidence or rigor was the relevance of such an evaluation to the goals of this study, i.e. to investigate if the financial aspect of technical debt is neglected against the software engineering one.

8. Conclusions

This paper aims at summarizing the research state of the art on technical debt management, emphasizing on the financial aspect of the metaphor. The main research questions that the systematic literature review answers are: (a) which financial terms are used in the field of technical debt, (b) which financial approaches have been used for technical debt measurement, (c) which financial

approaches have been used in technical debt management, and (d) which software technologies have been applied in financial approaches of technical debt management.

As an answer to the aforementioned questions, we introduce a glossary of financial terminology and a classification schema of the financial approaches used in technical debt management. On the one hand, the glossary incorporates the most important financial terms used in technical debt research and provides definitions that can be used by the technical debt community. On the other hand, the classification schema consists of three levels; the upper one constitutes of the existing categories of technical debt management [24], the middle one represents the financial techniques used, and the lower level depicts the used software engineering technologies, methods or tools. Additionally, the schema provides details on the popularity of each technique, term, method or tool, plus an evaluation of the techniques of level 2, based on their limitations when they are used in technical debt research.

By incorporating the abovementioned glossary of terms, software practitioners are expected to improve their communication with non-technical managers. Moreover, they can employ the proposed classification schema as a tool that will help them apply the suggested techniques for every TD management task. Finally, the results of our research suggest that researchers should attempt to pay further attention to issues such as the interdisciplinary nature of technical debt, the appropriate application of financial approaches to technical debt research and the possibility of applying new approaches to technical debt research, based on aspects that have not yet been quantified.

Appendix A. Papers included in the review

[P1] E. Allman, "Managing technical debt", *Communication*, ACM, 55 (5), pp. 50–55, May 2012.

[P2] E. Alzaghoul and R. Bahsoon, "CloudMTD: Using real options to manage technical debt in cloud-based service selection", *4th International Workshop on Managing Technical Debt (MTD '13)*, IEEE Computer Society, pp. 55–62, 18–26 May 2013, San Francisco, USA.

[P3] B. Bartonand and C. Sterling, "Manage Project Portfolios More Effectively by Including Software Debt in the Decision Process", *Cutter IT Journal*, October 2010.

[P4] E. Bavani, "Distributed Agile Testing and Technical Debt", *Software*, IEEE Computer Society, 29 (6), pp. 28–33, November/December 2012.

[P5] J. Bohnet and J. Dollner, "Monitoring code quality and development activity by software maps", *2nd Workshop on Managing Technical Debt (MTD '11)*, ACM, pp. 9–16, Hawaii, USA, 21–28 May 2011.

[P6] J. Brondum and L. Zhu, "Visualizing architectural dependencies", *3rd International Workshop on Managing Technical Debt (MTD '12)*, IEEE Computer Society, pp. 7–14, Zurich, Switzerland, 2–9 December 2012.

[P7] N. Brown, Y.Cai, Y.Guo, R.Kazman, M. Kim, P. Kruchten, E. Lim, A. McCormack, R. Nord, I.Ozkaya, R.Sangwan, C. Seaman, K. Sullivan, and N.Zazworka, "Managing technical debt in software-reliant systems", *Proceedings of the FSE/SDP workshop on Future of software engineering research*, ACM, pp. 47–52, New Mexico, USA, 7–8 November 2010.

[P8] F. Buschmann, "To Pay or Not to Pay Technical Debt", *Software*, IEEE Computer Society, 28 (6), pp. 29–31, November/December 2011.

[P9] S. Chin, E. Huddleston, W. Bodwell, and I. Gat, "The Economics of Technical Debt", *Cutter IT Journal*, October 2010.

[P10] Z. Codabux and B. Williams, "Managing technical debt: An industrial case study", *4th International Workshop on Managing Technical Debt (MTD '13)*, IEEE Computer Society, pp. 8–15, 18–26 May 2013, San Francisco, USA.

[P11] P. Conroy, "Technical Debt: Where Are the Shareholders' Interests?", *Software*, IEEE Computer Society, 29 (6), p. 88, November/December 2012.

[P12] B. Curtis, J. Sappidi, and A. Szynkarski, "Estimating the Principal of an Application's Technical Debt", *Software*, IEEE Computer Society 29 (6), pp. 34–42, November/December 2012.

[P13] 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)*, IEEE Computer Society, pp. 49–53, Zurich, Switzerland, 2–9 December 2012.

[P14] J. Davis and T. Andersen, "Surviving the Economic Downturn", *Agile Conference 2009 (AGILE '09)*, IEEE Computer Society, pp. 245–250, Chicago, USA, 24–28 August 2009.

[P15] R. J. Eisenberg, "A threshold based approach to technical debt", *ACM SIGSOFT Software Engineering Notes*, ACM, 37 (2), pp. 1–6, March 2012.

[P16] N. Ernst, "On the role of requirements in understanding and managing technical debt", *3rd International Workshop on Managing Technical Debt (MTD '12)*, IEEE Computer Society, pp. 61–64, Zurich, Switzerland, 2–9 December 2012.

[P17] D. Falessi, M. Shaw, F. Shull, K. Mullen, and M. Keymind, "Practical considerations challenges and requirements of tool-support for managing technical debt", *4th International Workshop on Managing Technical Debt (MTD '13)*, IEEE Computer Society, pp. 16–19, San Francisco, USA, 18–26 May 2013.

[P18] Gat and J. D.Heintz, "From assessment to reduction: how cutter consortium helps rein in millions of dollars in technical debt", *2nd International Workshop on Managing Technical Debt (MTD '11)*, ACM, pp. 24–26, Hawaii, USA, 21–28 May 2011.

[P19] J.M. Golden, "Transformation Patterns for Curing the Human Causes of Technical Debt", *Cutter IT Journal*, October 2010.

[P20] R. Gomes, C. Siebra, G. Tonin, A. Cavalcanti, F.Q. da Silva, A.L. Santos, and R. Marques, "An extraction method to collect data on defects and effort evolution in a constantly modified system", *2nd International Workshop on Managing Technical Debt*, ACM, pp. 27–30, Hawaii, USA, 21–28 May 2011, ACM, New York, NY, USA, 2011.

[P21] D. R. Greening, "Release Duration and Enterprise Agility", *46th Hawaii International Conference on System Sciences (HICSS-46)*, IEEE Computer Society, pp. 4835–4841, Hawaii, USA, 7–10 January 2013.

[P22] J. de Groot, A. Nugroho, T. Back, and J. Visser, "What is the value of your software?", *3rd International Workshop on Managing Technical Debt (MTD '12)*, IEEE Computer Society, pp. 37–44, Zurich, Switzerland, 2–9 December 2012.

[P23] Y. Guo and C. Seaman, "A portfolio approach to technical debt management", *2nd International Workshop on Managing Technical Debt*, ACM, pp. 31–34, Hawaii, USA, 21–28 May 2011.

[P24] Y. Guo, C. Seaman, R. Gomes, A. Cavalcanti, G. Tonin, F. da Silva, A.L. Santos, and C. Siebra, "Tracking technical debt - An exploratory case study", *27th International Conference on Software Maintenance (ICSM '11)*, IEEE Computer Society, pp. 528–531, Williamsburg, Virginia, USA, 25 September - 1 October 2011.

[P25] J. D. Heintz, "Modernizing the DeLorean System: Comparing Actual and Predicted Results of a Technical Debt Reduction Project", *Cutter IT Journal*, October 2010.

[P26] J. Holvitie and V. Leppanen, "DebtFlag: Technical debt management with a development environment integrated tool", *4th International Workshop on Managing Technical Debt (MTD '13)*, IEEE Computer Society, pp. 20–27, 18–26 May 2013, San Francisco, USA.

[P27] C. Izurieta and J. M.Bieman, "A multiple case study of design pattern decay grime and rot in evolving software systems", *Software Quality Journal*, SpringerLink, 21 (2), pp. 289–323, June 2013.

[P28] M. Kaiser and G. Royse, "Selling the Investment to Pay Down Technical Debt: The Code Christmas Tree", *Agile Conference 2011 (AGILE '11)*, IEEE Computer Society, pp. 175–180, Utah, USA, 8–12 August 2011.

[P29] T. Klinger, P.Tarr, P. Wagstromand, and C. Williams, "An enterprise perspective on technical debt", *2nd International Workshop on Managing Technical Debt*, ACM, pp. 35–38, Hawaii, USA, 21–28 May 2011.

[P30] S. Koolmanojwongand, and J.A. Lane, "Enablers and Inhibitors of Expediting Systems Engineering", *11th Annual Conference on Systems Engineering Research (CSER '13)*, Procedia Computer Science, Elsevier, 16, pp. 483–491, Atlanta, USA, 19–22 March 2013.

[P31] V. Krishna and A. Basu, "Minimizing Technical Debt: Developer's viewpoint", *International Conference on Software Engineering and Mobile Application Modelling and Development (ICSEMA '12)*, IEEE Computer Society, pp. 1–5, Chennai, India, 19–21 December 2012.

[P32] P. Kruchten, R. Nord, and I. Ozkaya, "Technical Debt: From Metaphor to Theory and Practice", *Software*, IEEE Computer Society 29 (6), pp. 18–21, November/December 2012.

[P33] O. Ktata and G. Lévesque, "Designing and implementing a measurement program for Scrum teams: what do agile developers really need and want?", *3rd C∗ Conference on Computer Science and Software Engineering (C^3S^2E '10)*, ACM, pp. 101–107, Montreal, Canada, 19–20 May 2010.

[P34] J. Letouzey and M. Ilkiewicz, "Managing Technical Debt with the SQALE Method", *Software*, IEEE Computer Society 29 (6), pp. 44–51, November/December 2012.

[P35] J. L. Letouzey, "The sqale method for evaluating technical debt", *3rd International Workshop on Managing Technical Debt (MTD '12)*, IEEE Computer Society, pp. 31–36, Zurich, Switzerland, 2–9 December 2012.

[P36] E. Lim, N. Taksande, and C. Seaman, "A Balancing Act: What Software Practitioners Have to Say about Technical Debt", *Software*, IEEE Computer Society 29 (6), pp. 22–27, November/December 2012.

[P37] R. Marinescu, "Assessing technical debt by identifying design flaws in software systems", *Journal of Research and Development*, IBM, 56 (5), pp. 1–13, September/October 2012.

[P38] J. D. McGregor, J. Monteith, and J. Zhang, "Technical debt aggregation in ecosystems", *3rd International Workshop on Managing Technical Debt (MTD '12)*, IEEE Computer Society, pp. 27–30, Zurich, Switzerland, 2–9 December 2012.

[P39] J. Monteith and J. McGregor, "Exploring software supply chains from a technical debt perspective", *4th International Workshop on Managing Technical Debt (MTD '13)*, IEEE Computer Society, pp. 32–38, 18–26 May 2013, San Francisco, USA.

[P40] J. Morgenthaler, M.Gridnev, R. Sauciuc, and S. Bhansali, "Searching for build debt: Experiences managing technical debt at Google", *3rd International Workshop on Managing Technical Debt (MTD '12)*, IEEE Computer Society, pp. 1–6, Zurich, Switzerland, 2–9 December 2012.

[P41] R. Nord, I. Ozkaya, P. Kruchten, and M. Gonzalez-Rojas, "In Search of a Metric for Managing Architectural Technical Debt", 2012 *Joint Working IEEE/IFIP Conference on Software Architecture (WICSA) and European Conference on Software Architecture (ECSA)*, IEEE Computer Society, pp. 91–100, Helsinki, Finland, 20–24 August 2012.

[P42] Nugroho, J. Visser, and T. Kuipers, "An empirical model of technical debt and interest", *2nd International Workshop on Managing Technical Debt (MTD' 11)*, ACM, pp. 1–8, Hawaii, USA, 21–28 May 2011.

[P43] K. Power, "Understanding the impact of technical debt on the capacity and velocity of teams and organizations: Viewing team and organization capacity as a portfolio of real options", *4th International Workshop on Managing Technical Debt (MTD '13)*, IEEE Computer Society, pp. 28–31, San Francisco, USA, 18–26 May 2013.

[P44] K. Pugh, "The Risks of Acceptance Test Debt," *Cutter IT Journal*, October 2010.

[P45] N. Ramasubbu and C. Kemerer, "Towards a model for optimizing technical debt in software products", *4th International Workshop on Managing Technical Debt (MTD '13)*, IEEE Computer Society, pp. 51–54, San Francisco, USA, 18–26 May 2013.

[P46] M. Reddy, "Chapter 4 - Design", Morgan Kaufmann, Boston, 2011, pp. 105–150.

[P47] D. Rooney, "Technical Debt: Challenging the Metaphor", *Cutter IT Journal*, October 2010.

[P48] Y. Rubin, S. Kallner, N. Guy, and G. Shachor, "Restraining Technical Debt when Developing Large-Scale Ajax Applications", *1st International Conference on Building and Exploring Web Based Environments (WEB' 13)*, IARIA XPS Press, pp. 13–18, Seville, Spain, 27 January–1 February 2013.

[P49] P. S. M. dos Santos, A.Varella, C. R.Dantas and D. B. Borges, "Visualizing and Managing Technical Debt in Agile Development: An Experience Report", *Lecture Notes in Business Information Processing*, Springer, 149, pp. 121–134, 2013

[P50] K. Schmid, "A formal approach to technical debt decision making", *9th International Conference on Quality of Software Architectures (QoSA' 13)*, ACM, pp. 153–162, Vancouver, Canada, 17–21 June 2013.

[P51] K. Schmid, "On the limits of the technical debt metaphor some guidance on going beyond", *4th International Workshop on Managing Technical Debt (MTD '13)*, IEEE Computer Society, pp. 63–66, San Francisco, USA, 18–26 May 2013.

[P52] C. Seaman and Y. Guo, "Measuring and monitoring technical debt", *Advances in Computers*, Elsevier, 82, pp. 25–46, 2011.

[P53] C. Seaman, Y.Guo, N. Zazworka, F. Shull, C. Izurieta, Y. Cai and A. Vetró, "Using technical debt data in decision making: Potential decision approaches", *3rd International Workshop on Managing Technical Debt (MTD' 12)*, IEEE Computer Society, pp. 45–48, Zurich, Switzerland, 5 June 2012.

[P54] C. Shafer, "Infrastructure Debt: Revisiting the Foundation", *Cutter IT Journal*, 2010.

[P55] S. Shah, M. Torchiano, A. Vetró, and M. Morisio, "Exploratory testing as a source of testing technical debt", *IT Professional*, IEEE Computer Society, 16 (3), pp. 44–51, March 2013.

[P56] T. Sharma, "Quantifying Quality of Software Design to Measure the Impact of Refactoring", *36th Annual Computer Software and Applications Conference Workshops (COMPSACW' 12)*, IEEE Computer Society, pp. 266–271, Izmir, Turkey, 16–20 July 2012.

[P57] C. S. Siebra, G. S. Tonin, F. Q. Silva, R. G. Oliveira, A. L. Junior, R. C. Miranda, and A. L. Santos, "Managing technical debt in practice: an industrial report", *6th International Symposium*

on Empirical Software Engineering and Measurement (ESEM' 12), ACM, pp. 247–250, Lund, Sweden, 19–20 September 2012.

[P58] W. Snipes, B. Robinson, Y. Guo and C. Seaman, "Defining the decision factors for managing defects: A technical debt perspective", *3rd International Workshop on Managing Technical Debt (MTD' 12)*, IEEE Computer Society, pp. 56–60, Zurich, Switzerland, 5 June 2012.

[P59] R. Spinola, N.Zazworka, A. Vetró, C. Seaman, and F. Shull, "Investigating technical debt folklore: Shedding some light on technical debt opinion", *4th International Workshop on Managing Technical Debt (MTD '13)*, IEEE Computer Society, pp. 1–7, San Francisco, USA, 18–26 May 2013.

[P60] M. G. Stochel, M. R. Wawrowski, and M. Rabiej, "Value-Based Technical Debt Model and Its Application", *7th International Conference on Software Engineering Advances (ICSEA' 12)*, XPert Publishing Service, pp. 205–212, Lisbon, Portugal, 18–23 November 2012.

[P61] T. Theodoropoulos, M. Hofbergand D. Kern, "Technical debt from the stakeholder perspective", *2nd International Workshop on Managing Technical Debt (MTD' 11)*, ACM, pp. 43–46, Hawaii, USA, 21–28 May 2011.

[P62] Vetró, "Using automatic static analysis to identify technical debt", *34th International Conference on Software Engineering (ICSE' 14)*, IEEE Computer Society, pp. 1613–1615, Zurich, Switzerland, 2–9 June 2012.

[P63] K. Wiklund, S. Eldh, D. Sundmark and K. Lundqvist, "Technical Debt in Test Automation", *5th International Conference on Software Testing, Verification and Validation (ICST' 12)*, IEEE Computer Society, pp. 887–892, Montreal, Canada, 17–21 April 2012.

[P64] R. J. Wirfs-Brock, "Skills for the agile designer: seeing shaping and discussing design ideas", *25th International Conference Companion on Object Oriented Programming Systems Languages and Applications*, ACM, pp. 323–326, Nevada, USA, 17–21 October 2010.

[P65] J. Xuan, Y. Hu and H. Jiang, "Debt-Prone Bugs: Technical Debt in Software Maintenance", *International Journal of Advancements in Computing Technology*, 4 (19), pp. 453–461, 2012.

[P66] N. Zazworka, C. Seaman, and F. Shull, "Prioritizing design debt investment opportunities", *2nd International Workshop on Managing Technical Debt (MTD' 11)*, ACM, pp. 39–42, Hawaii, USA, 21–28 May 2011.

[P67] N. Zazworka, M. Shaw, F. Shull, and C. Seaman, "Investigating the impact of design debt on software quality", *2nd International Workshop on Managing Technical Debt (MTD' 11)*, ACM, pp. 17–23, Hawaii, USA, 21–28 May 2011.

[P68] N. Zazworka, R. O. Spanola, A. Vetró, F. Shull and C. Seaman, "A case study on effectively identifying technical debt", *17th International Conference on Evaluation and Assessment in Software Engineering*, ACM, pp. 42–47, Porto Galinhas, Brazil, 14–16 April 2013.

[P69] N. Zazworka, A. Vetró, C. Izurieta, S. Wong, Y.Cai, C. Seaman and F. Shull, "Comparing four approaches for technical debt identification", *Software Quality Journal*, Springer, 22 (3), pp. 403 – 426, September 2014.

Appendix B. Collected data for $RQ_{2.1}$

Study	Brief approach description	Software eng. technologies	Financial terms	Financial approach	TDM activity
[P2]	The study focuses on the TD created by web services substitution, based on Real Options. Using a two-step binomial tree approach, it quantifies TD and identifies the time period during which TD is cleared out	–	Amount of debt Present Value Future Value	Real Options	Measurement
[P9]	The article calculates TD as a function of simple and compound interest, and the expected years of software's active development and maintenance. Simple interest can be determined by ongoing product maintenance costs, while compound interest rate may be measured by the increase in technical debt over time	Static Analysis Source Code Testing Design Smells	Simple interest Compound Interest Amount of debt	Software Economics	Measurement
[P12]	This article proposes the estimation of Technical Debt principal as a function of the number of should-fix violations in an application, the hours needed to fix each violation and the labor cost	AIP Tool Static Analysis Architecture Source Code	Principal	Software Economics	Measurement
[P13]	This paper calculates the principal of TD with a formula that takes into consideration three variables—the number of must-fix problems in an application, the time required to fix each problem, and the cost for fixing a problem	AIP Tool Static Analysis Architecture Source Code	Principal	Software Economics	Measurement
[P22]	TD equals Repair Effort that is the effort needed to improve the quality of a system to the ideal	SIG/TÜV Model	Amount of debt Interest	Software Economics	Measurement

(*continued on next page*)

Appendix B. (*continued*)

Study	Brief approach description	Software eng. technologies	Financial terms	Financial approach	TDM activity
	level. Additionally, it calculates Repair Effort as a function of 3 variables: an estimated percentage of LOC that needs to be changed in order to reach an ideal level of quality, the effort required to rebuild the system from scratch and a ratio discount for code volume that does not need to be rebuilt				
[P23]	The paper suggests the quantification of Technical Debt principal by the effort required to resolve Technical Debt Items (i.e. source code that needs to be modified, test cases that need to be exercised, documentation that needs to be updated, etc.). TD interest is related to interest standard deviation (based on Portfolio Management Theory), that is the probability of this interest to incur. All three parameters are estimated using historical data	Source Code Testing Documentation	Principal Interest	Portfolio Management	Measurement
[P24]	The paper presents a case study where Technical Debt principal, interest amount and interest probability are estimated based on historical effort data collected from the project documentation	COCOMO Static Analysis	Principal Interest	Software Economics	Measurement
[P41]	The paper assesses the amount of TD and interest through a model that computes the rework cost associated with each new architectural element implemented in every new release	Architecture	Amount of debt Interest	Software Economics	Measurement
[P42]	This paper defines the amount of TD as the Rework Effort and calculates it as a function of Rework Function, Rebuild Effort and Refactoring Adjustment. TD interest is defined as the difference of Maintenance Effort between a particular quality and the ideal quality level	SIG/TÜV Model Static Analysis Source Code	Amount of debt Interest Net Present Value	Software Economics	Measurement
[P49]	The paper reports a case where technical debt is quantified in terms of effort needed to fix it, by the use of SonarQube tool	SonarQube Tool	Amount of debt	Software Economics	Measurement
[P50]	The study provides formalization for the quantification of TD. TD is measured in terms of evolution costs and is defined as the difference between additional development cost introduced in an evolution and the respective cost required in an optimal implementation	Architecture	Amount of debt	Software Economics	Measurement
[P57]	In a case study, the paper estimates debt, interest and debt payment as the difference in effort between two or more alternative decisions	–	Amount of debt Interest Payment	Software Economics	Measurement
[P60]	The study proposes a three-layered model aligned with a typical software product development lifecycle (based on P-Diagram approach on risk assessment)	SonarQube Tool Wisdom of Crowds ATAM Architecture Source Code	Amount of debt ROI	Value-based	Measurement

Appendix C. Collected Data for $RQ_{2.2}$

Study	Brief approach description	Software eng. technologies	Financial approach	TDM activity
[P2]	The study uses a Binomial Option – Based Approach to manage Technical Debt. Binomial trees are built in order to indicate if and when an alternative can clear out Technical Debt and add value	–	Real Option	Repayment
[P8]	The article proposes the consideration of Technical Debt's financial benefits and business advantages as well as of the costs of servicing or repaying the debt in order to manage it efficiently	–	Cost/Benefit	Repayment
[P11]	The article suggests that TD should be handled as financial obligation and should be recorded on the firm's accounting books	–	Accounting	Monitoring
[P23]	This study proposes TDM based on Portfolio Management and treats TD items as assets. For each item, it is needed to determine if it is better to keep it or to pay it off. A risk level is set and the model generates the optimal portfolio (A) of the TD items. The items not chosen for the new portfolio (A') are those that need to be paid off	Source Code Testing Documentation	Portfolio Management	Repayment
[P24]	This paper applies cost/benefit analysis on a case study in order to demonstrate how adequate Technical Debt management can save a firm from high cost decisions	COCOMO Static Analysis Documentation	Cost/Benefit	Monitoring
[P32]	The article proposes Net Present Value theory as the most promising financial method to apply on Technical Debt management and decision making	–	Value-based	Repayment
[P41]	The paper suggests that Technical Debt management should take into consideration both value and cost of the project and also include elements of Total Cost of Ownership Management in order to enable efficient decision making on repaying Technical Debt	Architecture	Value-based	Repayment
[P42]	Using a case study, the paper defines the repair effort to improve the quality of a system as an investment. Thus it proposes the assessment of the worthiness of such an investment by looing at its ROI or Net Present Value	SIG/TÜV Model Static Analysis Source Code	ROI Net Present Value	Repayment
[P43]	This paper proposes the consideration of the development's team capacity as an investment portfolio that can be managed as a Real Options portfolio. Technical Debt reduction should be a significant investment in the team's portfolio	Defect-Density Functionality Performance	Real Options	Monitoring
[P45]	The paper provides a decision framework to assess the overall benefit (or loss) of accumulating technical debt, based on platform-based product development and customer adoption curve	Functionality	Marketing	Monitoring
[P50]	The study proposes a decision making approach on paying off technical debt items, according to the refactoring cost, the relative technical debt per refactoring and per evolution step and, finally, the probability that the evolution step will be performed	Architecture	Cost/Benefit	Repayment
[P53]	The paper suggests the management of Technical Debt by the use of Cost – Benefit analysis, Analytic Hierarchy Process, Portfolio approach and Options theory	Refactoring Design Smells	Cost/Benefit Portfolio Management Real Options	Prioritization Prioritization Repayment
[P58]	The paper identifies different kinds of costs associated with handling defects and investigates the criteria according to which a defect is decided to be fixed. Based on these findings, it proposes a cost – benefit analysis for decision making on fixing the defects	Defect-Density Testing	Cost/Benefit	Repayment
[P60]	By using Architecture Tradeoff Analysis Method (ATAM) and portfolio management, the paper proposes a model of Decision Making at the architecture level. Profitability aspect needs to be taken into account when managing; organization aims at optimal ROI	SonarQube Tool Wisdom of Crowds ATAM Architecture Source Code	Portfolio Management	Monitoring

Appendix C. (*continued*)

Study	Brief approach description	Software eng. technologies	Financial approach	TDM activity
[P66]	This paper implements a cost – benefit approach in order to identify technical debt items that should be refactored first, by ranking the value and interest of design debt caused by god classes	Refactoring Design Smells Defect-Density	Cost/ Benefit	Prioritization

Appendix D. Primary Studies Quality Assessment

Study	Question score					Total	Study	Question score					Total
	1	2	3	4	5			1	2	3	4	5	
[P1]	1.00	1.00	0.00	0.00	0.00	2.0	[P36]	1.00	1.00	0.80	0.00	0.00	2.8
[P2]	1.00	1.00	0.20	1.00	0.50	3.7	[P37]	1.00	1.00	0.80	0.00	0.50	3.3
[P3]	1.00	0.50	0.00	0.00	0.50	2.0	[P38]	1.00	1.00	0.00	0.00	0.00	2.0
[P4]	1.00	1.00	0.40	0.00	0.50	2.9	[P39]	1.00	1.00	0.60	0.00	0.50	3.1
[P5]	1.00	1.00	0.80	0.00	0.50	3.3	[P40]	1.00	1.00	1.00	0.00	0.50	3.5
[P6]	1.00	1.00	0.60	0.00	0.50	3.1	[P41]	1.00	1.00	0.20	1.00	1.00	4.2
[P7]	1.00	1.00	0.40	0.00	0.50	2.9	[P42]	1.00	1.00	0.80	1.00	1.00	4.8
[P8]	1.00	1.00	0.00	1.00	0.50	3.5	[P43]	1.00	1.00	0.20	1.00	1.00	4.2
[P9]	1.00	1.00	0.00	1.00	1.00	4.0	[P44]	1.00	1.00	0.00	0.00	0.50	2.5
[P10]	1.00	1.00	0.80	0.00	0.50	3.3	[P45]	1.00	1.00	0.00	1.00	1.00	4.0
[P11]	1.00	1.00	0.00	1.00	0.50	3.5	[P46]	1.00	1.00	0.60	0.00	0.50	3.1
[P12]	1.00	1.00	0.80	1.00	1.00	4.8	[P47]	1.00	1.00	0.00	0.00	0.00	2.0
[P13]	1.00	1.00	0.80	1.00	1.00	4.8	[P48]	1.00	1.00	0.60	0.00	0.50	3.1
[P14]	1.00	0.50	1.00	0.00	0.50	3	[P49]	1.00	1.00	0.80	1.00	1.00	4.8
[P15]	1.00	1.00	0.20	0.00	0.50	2.7	[P50]	1.00	1.00	0.20	1.00	1.00	4.2
[P16]	1.00	1.00	0.00	0.00	0.50	2.5	[P51]	1.00	1.00	0.00	0.00	0.00	2.0
[P17]	1.00	1.00	0.40	0.00	0.00	2.4	[P52]	1.00	1.00	0.20	0.00	0.50	2.7
[P18]	1.00	1.00	0.80	0.00	0.50	3.3	[P53]	1.00	1.00	0.00	1.00	1.00	4.0
[P19]	1.00	1.00	1.00	0.00	0.50	3.5	[P54]	1.00	1.00	0.00	0.00	0.50	2.5
[P20]	1.00	1.00	0.80	0.00	0.50	3.3	[P55]	1.00	1.00	0.00	0.00	0.50	2.5
[P21]	1.00	1.00	0.00	0.00	0.50	2.5	[P56]	1.00	0.50	0.60	0.00	0.50	2.6
[P22]	1.00	1.00	0.80	1.00	1.00	4.8	[P57]	1.00	1.00	1.00	1.00	0.50	4.5
[P23]	1.00	1.00	0.00	1.00	0.50	3.5	[P58]	1.00	1.00	0.80	1.00	1.00	4.8
[P24]	1.00	1.00	0.80	1.00	1.00	4.8	[P59]	1.00	1.00	0.60	0.00	0.00	2.6
[P25]	1.00	1.00	0.80	0.00	0.50	3.3	[P60]	1.00	1.00	0.20	1.00	1.00	4.2
[P26]	1.00	1.00	0.00	0.00	0.50	2.5	[P61]	1.00	1.00	0.00	0.00	0.50	2.5
[P27]	1.00	0.50	0.80	0.00	0.50	2.8	[P62]	1.00	1.00	0.60	0.00	0.50	3.1
[P28]	1.00	1.00	0.80	0.00	0.50	3.3	[P63]	1.00	1.00	0.80	0.00	0.50	3.3
[P29]	1.00	1.00	0.40	0.00	0.50	2.9	[P64]	1.00	0.50	0.40	0.00	0.50	2.4
[P30]	1.00	0.50	0.00	0.00	0.50	2.0	[P65]	1.00	1.00	0.60	0.00	0.50	3.1
[P31]	1.00	1.00	1.00	0.00	0.50	3.5	[P66]	1.00	1.00	0.20	0.00	0.50	2.7
[P32]	1.00	1.00	0.40	1.00	0.50	3.9	[P67]	1.00	1.00	0.80	0.00	0.00	2.8
[P33]	1.00	1.00	0.00	0.00	0.50	2.5	[P68]	1.00	1.00	0.80	1.00	1.00	4.8
[P34]	1.00	1.00	0.20	0.00	0.50	2.7	[P69]	1.00	1.00	0.60	0.00	0.50	3.1
[P35]	1.00	1.00	0.00	0.00	0.50	2.5							

References

[1] V. Alves, N. Niu, C. Alves, G. Valenca, Requirements engineering for software product lines: a systematic literature review, Inf. Softw. Technol., Elsev. 52 (8) (2010) 806–820.

[2] V. Basili, G. Caldiera, D. Rombach, The Goal Question Metric Approach, Encyclopedia of Software Engineering, John Wiley & Sons, 1994.

[3] L. Bass, P. Clements, R. Kazman, Software Architecture in Practice, second ed., Addison-Wesley Professional, 2003.

[4] M. Benaroch, Managing information technology investment risk: a real options perspective, J. Manage. Inf. Syst., ACM 19 (2) (October 2002) 43–84.

[5] B.W. Boehm, K.J. Sullivan, Software economics: a roadmap, in: 22nd International Conference on the Software Engineering (ICSE '00), ACM, pp. 319–343, Limerick, Ireland, 4–11 June 2000.

[6] A. Borison, Real options analysis: where are the emperor's clothes?, J Appl. Corp. Finan., Wiley & Sons 17 (2) (2005) 17–31.

[7] N. Chapin, J.E. Hale, K. Khan, J.F. Ramil, W.-G. Tan, Types of software evolution and software maintenance, J. Softw. Maint. Evol.: Res. Pract., Wiley & Sons 13 (1) (2001) 3–30.

[8] W. Cunningham, The WyCash portfolio management system, in: 7th International Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA '92), Vancouver, Canada, 5–10 October 1992, pp. 29–30.

[9] D. Duffie, K.J. Singleton, Credit Risk Pricing, Measurement, and Management, Princeton University Press, 2003.

[10] T. Dyba, T. Dingsoyr, Empirical studies of agile software development: a systematic review, Inf. Softw. Technol., Elsev. 50 (9) (2008) 833–859.

[11] R. Eisenberg, Management of technical debt: a lockheed martin experience report, in: 5th International Workshop on Managing Technical Debt (MTD' 13), Baltimore, USA, 9 October 2013.

[12] H. Erdogmus, Comparative evaluation of software development strategies based on Net Present Value, in: 1st Workshop on Economics Driven Software Engineering Research (EDSER' 99), Los Angeles, USA, 17 May 1999.

[13] P.W. Farris, N.T. Bendle, P.E. Pfeifer, D.J. Reibstein, Marketing Metrics: The Definitive Guide to Measuring Marketing Performance, second ed., Wharton School Publishing, 2010.
[14] M. Friedman, Factors affecting the level of interest rate, in: Conference on Savings and Residential Financing, USLL, Chicago, USA, 1968, pp. 11–27.
[15] P. Grubb, A. Takang, Software Maintenance: Concepts and Practice, second ed., World Scientific Publishing Company, 2003.
[16] T. Hull, Options, Futures and Other Derivatives, seventh ed., Pearson Prentice Hall, 2009.
[17] M. Ivarsson, T. Gorschek, A method for evaluating rigor and industrial relevance of technology evaluations, Emp. Softw. Eng., Spring. 16 (3) (June 2011) 365–395.
[18] ISO/IEC/IEEE 14764-2006 Standard for Software Engineering – Software Life Cycle Processes – Maintenance, IEEE Computer Society, 18 September 2006.
[19] S. Kellison, The Theory of Interest, third ed., McGraw-Hill, 2008.
[20] B. Kitchenham, O.P. Brereton, D. Budgen, M. Turner, J. Bailey, S. Linkman, Systematic literature reviews in software engineering – a systematic literature review, Inf. Softw. Technol., Elsev. 51 (1) (2009) 7–15.
[21] B.A. Kitchenham, D. Budgen, O.P. Brereton, Using mapping studies as the basis for further research: a participant–observer case study, Inf. Softw. Technol., Elsev. 53 (6) (2011) 638–651.
[22] P. Kruchten, The Rational Unified Process: An Introduction, second ed., Addison-Wesley Longman Publishing, 2000.
[23] P. Kruchten, R.L. Nord, I. Ozkaya, Technical debt: from metaphor to theory and practice, Software, IEEE Computer Society 29 (6) (2012) 18–21.
[24] Z. Li, P. Liang, P. Avgeriou, Architectural Debt Management in Value-oriented Architecting, Economics-Driven Software Architecture, Elsevier, 2014, pp. 183–204.
[25] Z. Li, P. Avgeriou, P. Liang, A systematic mapping study on technical debt and its management, J. Syst. Softw., Elsev. 101 (2015) 193–220.
[26] R.C Martin, Agile Software Development: Principles, Patterns, and Practices, Prentice Hall PTR, 2003.
[27] A. Mavridis, A. Ampatzoglou, I. Stamelos, P. Sfetsos, I. Deligiannis, Selecting refactorings: an option based approach, in: 8th International Conference on Quality of Information and Communications Technology (QUATIC' 12), IEEE Computer Society, Lisbon, Portugal, 3–6 September 2012, pp. 272–277.
[28] F. Mishkin, S. Eakins, Financial Markets and Institutions, seventh ed., Pearson Prentice Hall, 2012.
[29] R. Nickerson, J. Muntermann, U. Varshney, H. Isaac, Taxonomy development in information systems: developing a taxonomy of mobile applications, in: 17th European Conference in Information Systems (ECIS '09), Italy, 8–10 June 2009, pp. 1138–1149.
[30] R.T. Ogawa, B. Malen, Towards rigor in reviews of multivocal literatures: applying the exploratory case study method, Rev. Educ. Res., SAGE Publ. 61 (3) (1991) 265–286 (Fall 1991).
[31] D.L Parnas, Software Aging, 6th International Conference on Software Engineering (ICSE'94), 16–21 May 1994, IEEE Computer Society, Sorrento, Italy, 1994, pp. 279–287.
[32] F. Reilly, K. Brown, Investment Analysis and Portfolio Management, 10th ed., South-Western Cengage Learning, 2012.
[33] P. Runeson, M. Host, A. Rainer, B. Regnell, Case Study Research in Software Engineering: Guidelines and Examples, John Wiley & Sons, 2012.
[34] I. Sommerville, Software Engineering, ninth ed., Addison-Wesley, 2010.
[35] E. Tom, A. Aurum, R. Vidgen, An exploration of technical debt, J. Syst. Softw., Elsevier 86 (6) (2013) 1498–1516.
[36] H. van Vliet, Software Engineering: Principles and Practice, John Wiley & Sons, 2008.
[37] C. Wohlin, P. Runeson, M. Host, M.C Ohlsson, B. Regnell, A. Wesslen, Experimentation in Software Engineering: An Introduction, Kluwer, 2000.
[38] N. Zazworka, M. Shaw, F. Shull, C. Seaman, Investigating the impact of design debt on software quality, in: 2nd Workshop on Managing Technical Debt (MTD '11), ACM, Hawaii, USA, 21–28 May 2011, pp. 17–23.
[39] H. Zhang, M.A. Babar, On searching relevant studies in software engineering, in: 14th International Conference on Evaluation and Assessment in Software Engineering (EASE' 10), 12–13 April 2010, Keele, UK, pp. 1–10.