# A Framework to Aid in Decision Making for Technical Debt Management

Carlos Fernández-Sánchez
CITSEM, Technical
University of Madrid, Spain
carlos.fernandez@upm.es

Juan Garbajosa
CITSEM, ETSISI, Technical
University of Madrid, Spain
jgs@eui.upm.es

Agustín Yagüe
CITSEM, ETSISI, Technical
University of Madrid, Spain
agustin.yague@upm.es

***Abstract*—Current technical debt management approaches mainly address specific types of technical debt. This paper introduces a framework to aid in decision making for technical debt management, and it includes those elements considered in technical debt management in the available literature, which are classified in three groups and mapped into three stakeholders' points of view. The research method was systematic mapping. In contrast to current approaches, the framework is not constrained by a concrete type of technical debt. Using this framework it will be possible to build specific models to assist in decision making for technical debt management.**

## I. Introduction

The total cost of software maintenance has increased from 35%-40% to nearly 90% of the total cost of software projects over the last few decades [1]. The greatest part of this effort is expended in adding new functionalities and adapting the software system to environmental changes. However, usually, systems are far from well-prepared for maintenance. Ordinarily, the focus on immediate completion and making decisions that offer benefits in the short term incur costs in the long term due to weaknesses that complicate maintenance activities [2]. These weaknesses include inconsistent design, cross-cutting concerns, complex classes, and coupling [3], which eventually require greater maintenance efforts.

Cunningham [4] introduced the term "technical debt" to describe the situation described above. Technical debt is a metaphor that refers to the consequences of weak software development. Technical debt can grow due to the inability of developers to develop high quality applications [5] or, intentionally, as a result of decisions that prioritize functionality over qualities. Moreover, technical debt can evolve due to circumstances out of the developers' control: for example, it can arise due to changes in the context of the application [6], and it clears automatically at the end of a system's life [7]. Some technical debt is inevitable in a world with finite resources [5]: it is possible, and necessary, to live with some technical debt. However, several studies have highlighted the negative impact of uncontrolled technical debt on software development [8], including effects on the developer morale, team velocity, and the quality of the products [5], [9]. The problem is that the benefits of refactoring are largely invisible, sometimes intangible, and long term, while the costs of refactoring activities are significant and immediate [10]. Therefore, it is not enough to estimate the technical debt of a project: it becomes necessary to manage it to be able to have convincing arguments regarding when and why to remove technical debt. Technical debt management consists of identifying the sources of extra costs in software maintenance and analyzing when it is profitable to invest effort into improving a software system [5]. If technical debt is not managed at all, the debt can become too high. Then, the company will be compelled to invest all of its efforts into keeping the system running, rather than increasing the value of the system by adding new capabilities [5], and this will seriously damage the company's profitability, risking the fulfillment of its strategic goals. In any case, from *a business point of view*, reducing technical debt is a good idea if, and only if, it leads to increased profitability [11]. Thus, decisions must be made regarding technical debt management. Questions that can be expected when managing the technical debt of a project can include the following: How much technical debt is acceptable? When is the right moment to reduce technical debt? We must consider the current literature to assess the extent to which current technical debt management approaches can answer these and other similar questions. In the current literature, technical debt management approaches currently address ad-hoc cases or concrete types of technical debt, for example [12] [13] [10], but few approaches analyze technical debt management from an overall holistic perspective. A model focused on just one issue is too narrow to support decision making. For this reason, the software engineering community needs a framework that integrates various existing approaches. A second issue, pointed out above, is that decision making depends on the stakeholder's interests. Also, various stakeholders (e.g. technical or business oriented) are involved in technical debt decision making

This paper addresses the first steps in the definition of a framework to aid in decision making for technical debt management. With this objective in mind, the following research questions will been addressed:

**RQ1**: What elements are taken into account to manage technical debt when making decisions in a software project?

**RQ2**: What elements are considered from the various stakeholders' points of view?

The contribution of this paper is a framework for technical debt management constructed using the elements identified via a systematic literature mapping. Elements are understood in this paper as the concepts used to implement technical debt

978-1-4673-7378-4/15/$31.00 © 2015 IEEE

management, regardless of the nature of these concepts. The elements are also classified using the previously mentioned stakeholders' points of view. Using this framework, it will be possible to build specific models to assist in technical debt management decision making, addressing specific development process model characteristics in the context of a set of strategic business goals.

The structure of the paper is as follows: Section II will discuss previous related studies. Section III will present the methodology and the processes used in this study. Section IV will describe the elements of the framework. Section V will present the mapping of the framework elements onto the stakeholders' points of view. Finally, threats to validity, and conclusions and future work will be presented in Section VI and Section VII, respectively.

## II. Related Work

There are several previous literature reviews regarding technical debt. Tom et al. [5] performed a multivocal literature review to create a taxonomy for the phenomenon of technical debt. Alves et al. [14] performed a systematic mapping to create an ontology about technical debt. Li et al. [15] performed a systematic mapping identifying activities to perform in technical debt management. Finally, Ampatzoglou et al. [16] performed a systematic literature review to study the financial aspects of technical debt. While these studies were focused on studying the technical debt phenomenon [5] [14] [16] or the activities used in performing in technical debt management [15], our perspective is different from these previous studies in that we are specifically interested in the elements required to manage technical debt. Therefore these other studies have been used as sources for identifying these elements. Finally, Falessi et al. [17] identified the requirements for the tools utilized to manage technical debt. We have extended their work by incorporating the ways in which other authors have considered these requirements.

## III. Methodology and Research Process

To identify the elements of technical debt management that have been addressed in the current literature, this study utilized systematic mapping. The systematic mapping was performed by following Petersen et al.'s guide [18]. Systematic mapping studies are designed to provide a broad overview of a research area and are consequently appropriate for the goal of this research, that is, to identify what elements must be taken into account to efficiently manage technical debt. All the steps in the process are described in the following sections.

### A. Research Questions

**RQ1**: What elements have been taken into account to manage technical debt when making decisions in a software project?

**RQ2**: What elements are considered from the various stakeholders' points of view?

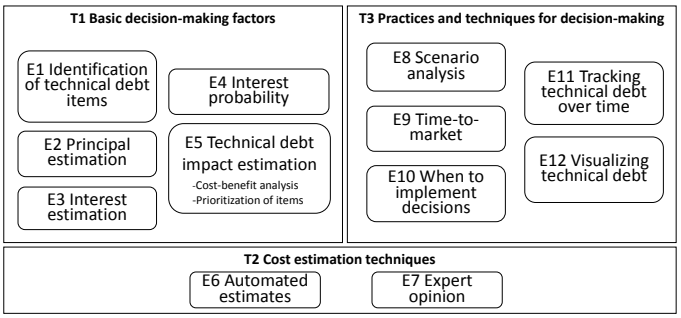


Fig. 1. Elements of the framework for technical debt management

### B. Conduct Search, Screen Papers, and Map

The steps described here were performed by two of the authors and discrepancies were resolved through agreement among all the authors. The term ***"technical debt"*** was used to search for papers about technical debt in the following digital repositories (using *full text* search when this option was available): IEEE Xplore, ACM, Scopus, ScienceDirect, Web of Science, and SpringerLink. The last search was performed on June 2nd, 2015. The total number of articles obtained (without duplicates) was 795. After the selection process was performed the number of papers selected was 51. To select the articles, the following inclusion and exclusion criteria were used:

- To be included, the paper had to describe parts, activities, tasks, elements, or considerations for technical debt management.
- Papers that failed to address technical debt management in detail were excluded
- To be included, the paper had to be published in a journal, conference proceedings, workshop proceedings, or book chapter (in some cases).
- Papers published as abstracts, call for workshops, tutorials, talks, or seminars were excluded.

The process used to identify the necessary elements for technical debt management has been adapted from the process described by Petersen et al. [18] for "keywording". The main difference is that in the present study, the process starts using the *full text* of the selected papers instead of the abstracts. The reason for this decision was that usually, the entire description of the technical debt management activity is not present in the abstract of the papers. The process was iterative. After a first classification was obtained, it was refined by contrasting the various elements found. As a result, the classification scheme was updated by binding similar elements together. This additional step was necessary because many times the same concept (or very similar concepts) is used in different contexts or with different names. Finally, the included papers have been classified using the various elements found (see Section IV) and the points of views (see Section V). Each paper may include several elements and/or points of view.

TABLE I
ELEMENTS TO SUPPORTING DECISION MAKING IN MANAGING TECHNICAL DEBT

| | Elements | Description | Sources |
|---|---|---|---|
| T1 Basic decision-making factors | E1 Identification of technical debt items | The sources of technical debt must be identified. | [6], [10], [12], [15], [16], [17], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38], [39], [40], [41], [42], [43], [44], [45], [46] |
| | E2 Principal estimation | The principal of a technical debt item is the cost that would have to be paid off to eliminate the technical debt item. | [5], [6], [10], [15], [16], [17], [19], [20], [22], [23], [24], [25], [28], [29], [30], [31], [33], [41], [42], [45], [46], [47], [48], [49], [50], [51] |
| | E3 Interest estimation | The interest of a technical debt item is the extra cost that must be paid over time if the technical debt item is not eliminated. | [5], [6], [10], [15], [16], [17], [19], [20], [22], [23], [24], [25], [28], [29], [30], [31], [33], [41], [42], [45], [46], [47], [48], [49], [50] |
| | E4 Interest probability estimation | The interest probability represents the likelihood of paying interest. | [6], [10], [15], [16], [17], [19], [23], [24], [25], [28], [29], [31], [42], [45], [48] |
| | E5 Technical debt impact estimation | Technical debt impact must be translated into economic consequences. This allows a company to perform cost-benefit analysis to identify the items with the most technical debt. | [5], [6], [7], [8], [10], [12], [13], [15], [16], [17], [19], [20], [21], [22], [23], [24], [25], [26], [27], [29], [31], [33], [35], [36], [38], [39], [41], [42], [45], [46], [47], [48], [50], [52], [53], [54], [55] |
| T2 Cost estimation techniques | E6 Automated estimates | To manage technical debt, it is desirable to have means of estimating variables in an automatic way. | [8], [10], [12], [17], [19], [20], [21], [27], [31], [35], [37], [38], [39], [43], [51] |
| | E7 Expert opinion | The expert opinion about a system will add information that it is not possible to obtain from available software information (code, documentation, etc.) | [10], [17], [19], [29], [31], [39], [45], [48] |
| T3 Practices and techniques for decision-making | E8 Scenario analysis | The output of the technical debt management must be in the form of scenario analysis that clarifies the various decisions to be made. | [6], [7], [10], [12], [17], [21], [25], [28], [29], [31], [35], [42], [48], [50], [53], [54], [56] |
| | E9 Time-to-market | When managing technical debt, the time needed to implement the decisions must be taken into account. | [5], [16], [17], [24], [28], [29], [41], [44], [53], [54], [55] |
| | E10 When to implement decisions | The implementation of a decision about technical debt can be made at different moments. These decisions include incurring technical debt or removing technical debt items. | [10], [16], [17], [19], [23], [34], [35], [45], [50], [54], [56], [57] |
| | E11 Tracking technical debt over time | It is necessary to track technical debt over time. | [6], [8], [12], [15], [16], [17], [21], [24], [25], [27], [35], [37], [45], [50], [53] |
| | E12 Visualizing technical debt | It is necessary to have mechanisms to show how technical debt is impacting the system. | [8], [10], [12], [15], [16], [21], [31], [32], [38], [58], [59] |
| List of all selected papers | | [5], [6], [7], [8], [10], [12], [13], [15], [16], [17], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38], [39], [40], [41], [42], [43], [44], [45], [46], [47], [48], [49], [50], [51], [52], [53], [54], [55], [56], [57], [58], [59] | |

## IV. ELEMENTS OF THE FRAMEWORK FOR TECHNICAL DEBT MANAGEMENT

This section addresses the first research question *What elements have been taken into account to manage technical debt when making decisions in a software project?* Figure 1 shows the identified elements and their type. Table I shows the sources that use or suggest the usage of specific elements for technical debt management. The analysis results in three types of elements. *T1, basic decision-making factors*, are focused on obtaining the necessary information about the system's technical debt. *T2, cost estimation techniques*, are focused on how a technical debt management model should be implemented. And, finally, *T3, practices and techniques for decision-making*, are focused on the considerations that must be taken into account, in addition to the technical debt estimations, to properly manage technical debt. The following subsections include a detailed explanation of the identified elements.

*E1 Identification of technical debt items.* To manage technical debt properly, some authors propose that the first step should be identifying the technical debt items [20] [19]. This step can include establishing a list of bad practices that create debt [12] [27] and identifying the potential kinds of technical debt from the sources of technical debt [17], as well as determining the part of the system that must be refactored [10]. Therefore, there are many potential sources of technical debt at any time in any system [6].

*E2 Principal estimation.* The principal of a technical debt item is the cost to be paid to eliminate such a technical debt item [5]. That is, the principal of a technical debt item is the effort required to remove a technical debt item by changing the software. Estimating the principal is one of the bases of technical debt management for most of the authors (see Table I). It is possible to estimate the principal as a function of three variables: the should-fix items with violations, the hours needed to fix each violation, and the cost of labor [20]. One important consideration is that technical debt is context dependent [31]. Therefore, solving a weakness can cost more or less depending on the project or even the subsystem within a project [31]. However, estimating the principal in terms of

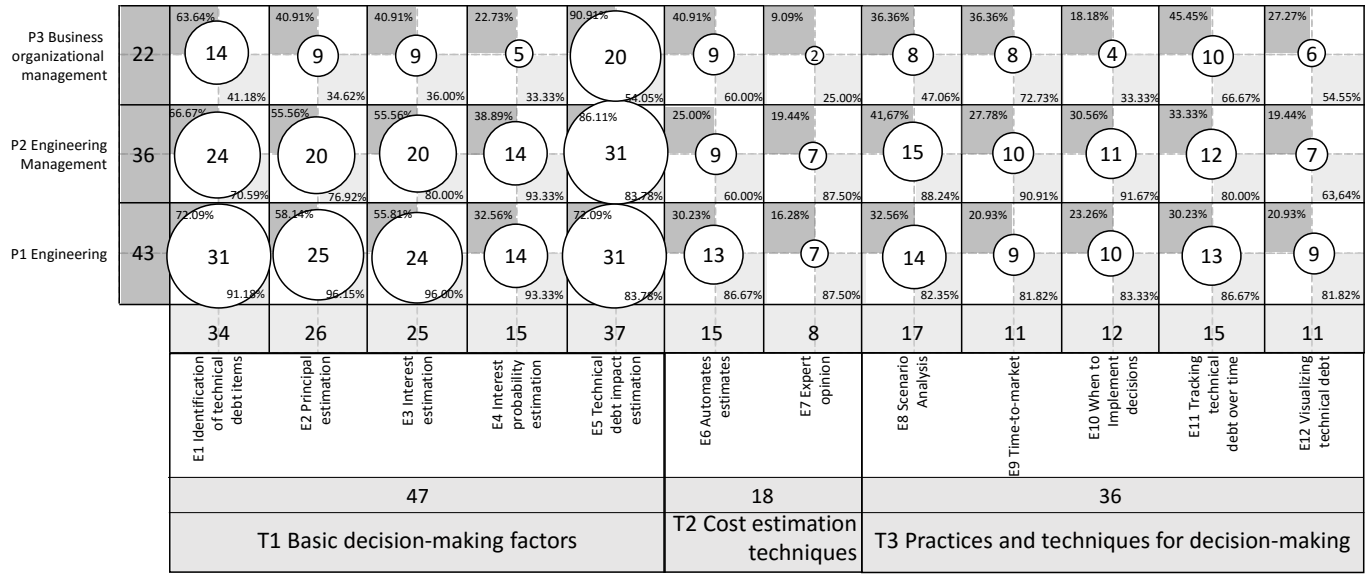


Fig. 2. Mapping of elements to support decision making in managing technical debt versus the engineering, engineering management, and business organizational management points of view. Each selected paper can include several elements and can be mapped onto more than one point of view. In each cell, the number in the center is the number of papers that identify an element from a specific stakeholder's point of view, the upper left percentage is the percentage of papers that identify the element (column) as specific to a point of view of (row), and the lower right percentage is the percentage of papers with the specific point of view (row) that identify the element (column). The first column shows the summary of the papers per stakeholder point of view, while in the bottom of the figure, there are summaries of papers per element and per type of element.

single values is difficult, and consequently, practitioners seem to think in ranges of values or best-case, worst-case, and most probable scenarios, rather than single values [17] [10].

*E3 Interest estimation.* The interest is the cost to be paid over time if a technical debt item is not eliminated. This can include the extra cost of modifying a component in need of refactoring as compared to the cost of modifying it after refactoring [6]. Much like the principal, estimating the interest is fundamental for most of the authors (see Table I). The interest can be seen in various ways. It is possible that depending on the projects and the contexts associated with them, the interest could be non-linear or have limits, maximums, or minimums [17]. As with the principal, one important consideration is that technical debt is context dependent [31]. Hence, the same detected weakness can imply more or fewer future costs depending on the project or even the subsystem within the project [31].

*E4 Interest probability estimation.* The interest has a probability of being paid [19] [17] [6]. This is because the interest must only be paid under some scenarios [34]: the probability of paying the interest will depend on the probability of the occurrence of future events [17] [25] [28]. For example, if a technical debt item implies that more effort is required in maintenance activities but that the item will not be changed over time, the interest does not need to be paid. Hence, in this example, the probability of changing this item will parallel the probability of interest. Therefore, the interest is determined by a set of relevant change scenarios [25]: the probable changes that will occur in the future. Those scenarios will include, for example, the addition of new functionalities, changes in the non-functional requirements, or the solution to problems or bugs in the system.

*E5 Technical debt impact estimation.* To manage technical debt, it is necessary to use models in which items are prioritized on the basis of theirs interests and principals [19]. The aim of ranking technical debt items is to identify which items should be resolved first, depending on the business's goals and preferences [25]. Decisions about technical debt should be made in terms of cost-benefit analysis [19] [17]. Thus, cost-benefit estimations are necessary to make decisions about removing technical debt [10]. The *business value* of conducting any activity lies in the difference between the cash flow stream of performing the activity and the cash flow stream of not performing it [11]. That is, a technical debt item must be removed when this turns out to be profitable. Identifying the items with the highest amount of technical debt and considering the cost of fixing them are the bases of cost-benefit analysis [19]. Usually, cost-benefit analysis will include several variables. The most obvious are the principal (the cost paid to remove a technical debt item) and the interest (the cost to be avoided by removing a technical debt item). More costs could be considered, such as time-to-market penalizations or quality issues, as well as benefits, for example, quality improvements, or customer satisfaction. Additional information that can be used includes the number of features that a software release will contain, the time required to deliver such a release, and the technical debt that can be generated due to forced quick development [53].

Any result of technical debt management should be reviewed in terms of its economic consequences [17], taking

into account business considerations [7]. That is, technical debt must be quantified [8]. It is not always easy to express technical debt using economic data. However, without expressing the economic consequences, it is difficult to quantify the impact of executing or not executing a decision. Tom et al. [5] classify technical debt costs into four types: morale, productivity, quality, and risk [5]. One problem is that the financial impact of technical debt is not always direct [5]. For example, technical debt can cause low morale on the development team resulting in systemic problems, such as developer turnover [5], [9]. Another issue is that it is necessary to balance rigor with the usability of the estimation method. A very complex and rigorous mathematical estimation model could have high precision in its prediction and high reliability in its results, but it could be unusable in practice due to its complexity when adapted to real, large projects [17]. Finally, some special situations must be considered in cost-benefit analysis. An example would be when a system is to be "retired" and therefore all of its technical debt is due to be removed [5]. Another issue is that organizations will probably not have adequate resources to fix all of the identified technical debt items [12]. To sum up, a complete cost-benefit analysis must take all factors into account. That is, it must consider not only the principal and interest but also the project constraints, including the deadline, budget, and impact [12], [25].

*E6 Automated estimates.* It is useful to obtain estimates automatically to manage technical debt, avoiding the negative impact of intrusiveness in the normal development process [8]. Making the collection of measures an extra step for developers, who are already overloaded, will compromise the success of technical debt management [31]. For this, the source code, a project's revision history, a project's bug history, and other similar data sources can be mined using automatic tools to obtain the necessary information to estimate technical debt automatically [17], [19] and to propose potential items for refactoring [10].

*E7 Expert opinion.* Together with automated estimates to manage technical debt, the opinions of the people that know the system deeply are required [17], [19]. An expert opinion about the system provides knowledge that cannot be obtained from available software information. This information can include new contracts to be signed, expected changes to be made, or new technologies to be adopted [48]. This will help the manager to effectively provide and apply information regarding issues such as uncertainty in the measures and judgments, the system's external context, and the knowledge of experts (project managers, architects, etc.) [10]. Finally, the goal of technical debt management methods and tools is to provide the necessary information to human decision-makers [31].

*E8 Scenario analysis.* Managing technical debt includes analyzing multiple potential scenarios [17]. By analyzing scenarios, managers can acquire information about the technical debt's impact if certain events occur in the future, as discussed in *E4 Interest probability estimation*. Various implementation possibilities can be analyzed to determine which one to implement. The goals of scenario analysis in technical debt management are as follows: *i)* to be able to set targets for debt and specify what level is acceptable for the project or organization [12] and *ii)* to identify the impact of non-fixed technical debt issues over multiple releases. This is performed by using change scenarios [25] [48]. Change scenarios represent the probable future changes that the system will accommodate and determine the amount of interest to be paid due to technical debt [28]. *iii)* To identify when it is profitable to either implement new functionality early (taking more technical debt due to the quick release) or when it is profitable to release functionality more slowly but with less technical debt [53]. The scenarios to be analyzed can include the various paths followed to deliver features in the release planning [54], that is, to analyze how much effort to invest in either refactoring, architecture design, or adding new features [6] by researching various release scenarios [56]. *iv)* The output of the technical debt management must be in the form of possible scenarios, along with the probability of their economic consequences [17]. With this information, the manager will be able to choose which decisions to implement with the highest probability, and he or she will also know the other possible alternatives and the system's technical debt evolutions. *v)* It is necessary to sketch and assess potential alternatives in terms of the benefits and costs to support choosing the most appropriate alternative for handling the technical debt [7]. This process includes testing various possible scenarios to analyze the impact of removing some technical debt items, that is, a what-if analysis [10] [31].

*E9 When to implement decisions.* Managing technical debt requires making the decision to either pay off the principal of technical debt items or continuing to pay the interest on such items. This element is greatly influenced by the constraints on and availability of the development team's resources. It is necessary to know when to carry this decision out [19] [17]. Some of the decisions may include when to implement a feature of a product, or when to refactor some part of the system to improve some of its qualities [54]. In this sense, these decisions will affect the release dates and planning [56].

*E10 Time-to-market.* One important constraint to consider is the time-to-market [17]. This element includes the resources and constraints involved in achieving a project goal on time. The solutions to be implemented could be useless if they cannot be implemented within a certain time. In some situations, it could be necessary to release a product by the deadline, without all of its expected characteristics. In certain environments, being the first to market is vital to obtain customers. In this situation, longer-term software problems are not important because they are not visible to product customers and these problems are not very important to the software company. In the long term, the problems will be relevant only if the product or the sponsors obtain customers in the short term [5]. To fully consider the cost and benefit of incurring technical debt it is necessary to take into account the *release planning* of the product under analysis [54]. To sum

up, a tradeoff between release characteristics and technical debt must be made to manage technical debt [53].

*E11 Tracking technical debt over time.* It is necessary to track technical debt over time [8]. This tracking requires methods to determine the level of technical debt and its evolution over time [6]. Monitoring technical debt consists of watching changes in the costs and benefits of unresolved technical debt items over time [25]. Monitoring technical debt frequently makes it possible to react quickly [12]. Thus, monitoring the evolution of the economic consequences of technical debt [17] is important. To analyze how the system will perform in the future, it is necessary to consider the time frame for such an analysis [48]. Due to the fact that technical debt implies a cost over time, the time frame will bind the analysis and enable a cost-benefit analysis. One threshold for the time frame could be the estimated date for retiring the software [53]. Other possible time frames could be obtained from the project release plan [24] or from the project roadmap.

*E12 Visualizing technical debt.* Having the means to see how technical debt is impacting the system or the development process is highly recommended [59]. Defining a visual language for the entire organization allows for fast and transparent communication between people and entities [12]. In this way, it is possible to see the relative impact of technical debt with regard to other activities [8]. However, the visualization technique must have the ability to summarize information for high-level analysis, and it must also include the possibility of analyzing technical debt at the lower levels, for example, at the subsystem or component level [31]. The visualization of technical debt is especially important in architectural technical debt [58]. Usually, this kind of technical debt implies several source code artifacts, for example, classes, configuration files, etc. Having mechanisms to see how these artifacts are grouped helps to clarify how technical debt is distributed over the system.

## V. Stakeholders' Points of View for Technical Debt Management

In this section the second research question is addressed: *What elements are considered from the various stakeholders' point of view?* Though it is obvious that various stakeholders are involved in technical debt management, which classification system to use is not evident. Clements et al. proposed, in [60], three stakeholders' point of view, namely (software) engineering, technical management, and organizational management, which were used in the first version of the work. More recently, in SWEBOK V3 [61], the terms engineering management, and business organizational are used. This reflects an evolution in the understanding of how software product development takes place. Therefore, the final list was as follows: (software) *P1 engineering*, *P2 engineering management*, and *P3 business organizational management*. *P1 engineering* involves processes such as software design and software construction, as well as, software architecting for some members of the community. *P2 engineering management* is comprised of process planning and monitoring, including measurement. Finally, *P3 business organizational management* involves organizational goals, business strategy, time horizons, risk factors, financial constraints, and tax considerations.

Figure 2 shows the mapping of the identified elements for technical debt management (see Section IV) and the stakeholders' points of view described above. The analysis reveals (see Figure 2) how most of the articles are focused on the *T1 basic decision-making factors*, i.e., on obtaining the necessary information about the system's technical debt. In this case, the preponderant point of view is *P1 engineering* (e.g., the creation of source code, designing, architecting, or testing) and *P2 engineering management* (e.g., planning, or product quality management).

Those papers included in *P1 engineering* and *P2 engineering management* strongly highlight all the identified technical debt management elements. Conversely, this is not the case for *P3 business organizational management*, apart from *E9 time-to-market*, *E11 tracking technical debt over time*, and *E6 automated estimates elements*. Actually, study [62] discusses how *E9 time-to-market* is hardly ever taken into account by the current technical debt management tools and methods. This is an interesting paradox and a serious issue: while *E9 time-to-market* is one of the least referenced/used/suggested *T3 practices and techniques for decision-making*, it is probably the most referenced cause of technical debt, and according to [5] one of its most relevant antecedents.

Most of the articles analyzed are focused on *T1 basic decision-making factors*. This may indicate that technical debt management is still in its initial phase and that the software engineering community lacks data estimation techniques, and metrics for the technical information normally extracted from source code, repositories, or tracking systems. While *T1 basic decision-making factors* seems to be conceptually close to the project *P1 engineering* activities, it happens that *T3 practices and techniques for decision-making* are not far from the decision making process and are therefore closer to *P2 engineering management* and *P3 business organizational management* than to *P1 engineering*. While *P1 engineering* is focused on finding project technical problems, *P2 engineering management* and *P3 business organizational management* extract information from sources such as strategic decisions about the architecture, product release dates, new contract signatures, budgets, or technologies provided by partners, which belong to the management and strategy areas. It is remarkable that *E5 Technical debt impact estimation* is the only highly suggested element, regardless of point of view. This somehow makes that this element, *E5 Technical debt impact estimation*, and by extension technical debt management, operates as a nexus between the three points of view.

The analysis of the *E8 scenario analysis* element reveals that it has the highest number of references within *T3 practice and technique for decision-making*. That is, it seems to operate as the point of connection of all the other *T3 practices and techniques for decision-making*. Finally, the *E6 automated estimates* and *E7 expert opinion* elements fit in the type *T2 cost estimation techniques*. These two elements are comple-

mentary: while *E6 automated estimates* refer to extracting the required data without disturbing the normal activity of developers, *E7 expert opinion* refers to using experts to fill in the information that cannot be automatically extracted and making decisions based on estimations.

## VI. Threats to Validity

This section addresses, following [63], potential biases and the actions taken to minimize their impact.

**Bias in identifying articles.** There are several issues that can affect article identification: the criteria of the reviewers and editors of the journals or conferences, industry-sponsored research in some areas, place of publication, biased indexing studies in literature databases, inadequate or incomplete searches, articles that are more often cited than others, and studies that generate multiple publications. In the present study, several different literature databases have been used to include the maximum number of sources and to minimize the impact of the above mentioned bias. Due to the complications involved in identifying when several publications are in fact results of the same study, no action has been taken to merge publications. This is a minor risk because few papers included in this study have been authored by the same researcher.

**Choosing study biases.** The selection process of papers was conducted following the steps provided by Petersen et al. [18]. To reduce possible bias, the inclusion/exclusion step was completed by two of the authors of this study. Discrepancies were resolved by agreements brokered among all the authors. The inclusion and exclusion criteria can also be influenced by the personal biases of the authors. The *a priori* definition of the criteria helps minimize this bias.

**Obtaining accurate data bias.** In a literature review, poor quality sources can lead to inaccurate conclusions. To mitigate this, the selected papers were published in journals, conference proceedings, or workshop proceedings with a peer-review process. To include all relevant studies about technical debt management, some book-chapters have been included. Because these are few in number compared to the other selected papers, the impact on the quality of the results is low. To mitigate the risk of the personal bias of the person who analyzes the papers, again, every paper was analyzed by two persons. Discrepancies were resolved through agreements among all the authors. The authors had to interpret the papers to classify them as being about engineering, engineering management, or business organizational management. This classification can be influenced by personal bias or by the information given in the papers, and this classification should be confirmed by subsequent studies, for example, by performing interviews with practitioners. Finally, this study does not consider the impact of the maturity of the analyzed papers. This can influence the reported results, and it could be investigated in the future.

## VII. Conclusions and Future Work

This paper has introduced a framework to aid in decision making for technical debt management. The framework includes those elements considered in technical debt management in the available literature, which were classified into groups (basic decision-making factors, cost estimation techniques, practices and techniques for decision-making), and mapped onto three stakeholders' points of view (engineering, engineering Management, business organizational management). The research method was systematic mapping [18]. In contrast to the vast majority of the current technical debt management approaches, the framework was not constrained by a concrete type of technical debt. Using this framework, it will be possible to build specific models to aid in decision making for technical debt management, considering specific development process models in strategic business goals. In the future, the framework will be extended by introducing the relationships among the elements to establish how one element affects or depends on the others. Another issue to be addressed will be to prioritize which elements are more relevant than others. The framework will be used to define technical debt management models for specific systems with two objectives: demonstrate how the framework works in practice and implement specific technical debt management models based on the integration of currently available tools for technical debt management.

## Acknowledgment

Work partially sponsored by MESC DPI2013-47450-C2-2-R (Spain). The authors are indebted to the anonymous reviewers for their useful and thoughtful comments.

## References

[1] A. Rashid, W. Wang, and D. Dorner, "Gauging the differences between expectation and systems support: The managerial approach of adaptive and perfective software maintenance," in *COINFO*, 2009.

[2] S. McConnell. (2007, November) Technical debt. [Online]. Available: http://www.construx.com/10x_Software_Development/Technical_Debt/

[3] A. Yamashita and L. Moonen, "To what extent can maintenance problems be predicted by code smell detection? an empirical study," *Information and Software Technology*, 2013.

[4] W. Cunningham, "The wycash portfolio management system," *SIGPLAN OOPS Mess.*, vol. 4, no. 2, pp. 29–30, Dec. 1992.

[5] E. Tom, A. Aurum, and R. Vidgen, "An exploration of technical debt," *Journal of Systems and Software*, vol. 86, no. 6, pp. 1498–1516, 2013.

[6] N. Brown, Y. Cai, Y. Guo, R. Kazman, M. Kim, P. Kruchten, E. Lim, A. MacCormack, R. Nord, I. Ozkaya, R. Sangwan, C. Seaman, K. Sullivan, and N. Zazworka, "Managing technical debt in software-reliant systems," in *FSE/SDP Workshop*, 2010.

[7] F. Buschmann, "To pay or not to pay technical debt," *Software, IEEE*, vol. 28, no. 6, pp. 29–31, Nov 2011.

[8] 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," in *International Workshop on Managing Technical Debt (MTD)*, 2013.

[9] L. Peters, "Technical debt: The ultimate antipattern," in *International Workshop on Managing Technical Debt (MTD)*, 2014.

[10] Y. Cai, R. Kazman, C. V. Silva, L. Xiao, and H.-M. Chen, "Chapter 6 - a decision-support system approach to economics-driven modularity evaluation," in *Economics-Driven Software Architecture*, 2014.

[11] S. Tockey, "Chapter 3 - aspects of software valuation," in *Economics-Driven Software Architecture*, I. Mistrik, R. Bahsoon, R. Kazman, and Y. Zhang, Eds. Boston: Morgan Kaufmann, 2014, pp. 37 – 58.

[12] J. Letouzey and M. Ilkiewicz, "Managing technical debt with the sqale method," *Software, IEEE*, vol. 29, no. 6, pp. 44–51, Nov 2012.

[13] Z. Codabux and B. Williams, "Managing technical debt: An industrial case study," in *International Workshop on Managing Technical Debt (MTD)*, 2013.

[14] N. S. R. Alves, L. F. Ribeiro, V. Caires, T. S. Mendes, and R. O. Spínola, "Towards an ontology of terms on technical debt," in *International Workshop on Managing Technical Debt (MTD)*, 2014.

[15] Z. Li, P. Avgeriou, and P. Liang, "A systematic mapping study on technical debt and its management," *J. Syst. Softw.*, vol. 101, no. C, pp. 193–220, Mar. 2015.

[16] A. Ampatzoglou, A. Ampatzoglou, A. Chatzigeorgiou, and P. Avgeriou, "The financial aspect of managing technical debt," *Inf. Softw. Technol.*, 2015.

[17] D. Falessi, M. Shaw, F. Shull, K. Mullen, and M. Keymind, "Practical considerations, challenges, and requirements of tool-support for managing technical debt," in *International Workshop on Managing Technical Debt (MTD)*, 2013.

[18] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering," in *EASE*, 2008.

[19] C. Seaman, Y. Guo, N. Zazworka, F. Shull, C. Izurieta, Y. Cai, and A. Vetro, "Using technical debt data in decision making: Potential decision approaches," in *MTD Workshop*, 2012.

[20] B. Curtis, J. Sappidi, and A. Szynkarski, "Estimating the principal of an application's technical debt," *Software, IEEE*, vol. 29, no. 6, pp. 34–42, Nov 2012.

[21] J.-L. Letouzey, "The sqale method for evaluating technical debt," in *International Workshop on Managing Technical Debt (MTD)*, June 2012, pp. 31–36.

[22] N. Zazworka, C. Seaman, and F. Shull, "Prioritizing design debt investment opportunities," in *MTD workshop*, 2011.

[23] Y. Guo and C. Seaman, "A portfolio approach to technical debt management," in *International Workshop on Managing Technical Debt (MTD)*, 2011.

[24] Y. Guo, C. Seaman, R. Gomes, A. Cavalcanti, G. Tonin, F. da Silva, A. Santos, and C. Siebra, "Tracking technical debt: An exploratory case study," in *ICSM*, 2011.

[25] Z. Li, P. Liang, and P. Avgeriou, "Chapter 9 - architectural debt management in value-oriented architecting," in *Economics-Driven Software Architecture*, I. Mistrik, R. Bahsoon, R. Kazman, and Y. Zhang, Eds. Boston: Morgan Kaufmann, 2014, pp. 183 – 204.

[26] E. Lim, N. Taksande, and C. Seaman, "A balancing act: What software practitioners have to say about technical debt," *Software, IEEE*, vol. 29, no. 6, pp. 22–27, Nov 2012.

[27] R. Marinescu, "Assessing technical debt by identifying design flaws in software systems," *IBM Journal of Research and Development*, vol. 56, no. 5, pp. 9:1–9:13, Sept 2012.

[28] K. Schmid, "On the limits of the technical debt metaphor some guidance on going beyond," in *MTD workshop*, 2013.

[29] ——, "A formal approach to technical debt decision making," in *QoSA*, 2013.

[30] N. Zazworka, A. Vetro, C. Izurieta, S. Wong, Y. Cai, C. Seaman, and F. Shull, "Comparing four approaches for technical debt identification," *Software Quality Journal*, vol. 22, no. 3, pp. 403–426, 2014.

[31] F. Shull, D. Falessi, C. Seaman, M. Diep, and L. Layman, "Technical debt: Showing the way for better transfer of empirical results," in *Perspectives on the Future of Software Engineering*, J. Münch and K. Schmid, Eds. Springer Berlin Heidelberg, 2013, pp. 179–190.

[32] R. Schwanke, L. Xiao, and Y. Cai, "Measuring architecture quality by structure plus history analysis," in *ICSE*, 2013.

[33] Z. Codabux, B. Williams, and N. Niu, "A quality assurance approach to technical debt," in *SERP*, 2014.

[34] P. Kruchten, R. Nord, and I. Ozkaya, "Technical debt: From metaphor to theory and practice," *Software, IEEE*, vol. 29, no. 6, pp. 18–21, Nov 2012.

[35] N. Ramasubbu and C. Kemerer, "Managing technical debt in enterprise software packages," *Software Engineering, IEEE Transactions on*, vol. 40, no. 8, pp. 758–772, Aug 2014.

[36] N. Ramasubbu, C. Kemerer, and C. Woodard, "Managing technical debt: Insights from recent empirical evidence," *Software, IEEE*, vol. 32, no. 2, pp. 22–25, Mar 2015.

[37] C. Siebra, A. Cavalcanti, F. Silva, A. Santos, and T. Gouveia, "Applying metrics to identify and monitor technical debt items during software evolution," in *ISSREW, 2014*, Nov 2014, pp. 92–95.

[38] G. Skourletopoulos, R. Bahsoon, C. Mavromoustakis, G. Mastorakis, and E. Pallis, "Predicting and quantifying the technical debt in cloud software engineering," in *CAMAD, 2014*, Dec 2014, pp. 36–40.

[39] D. Reimanis, C. Izurieta, R. Luhr, L. Xiao, Y. Cai, and G. Rudy, "A replication case study to measure the architectural quality of a commercial system," in *ESEM 2014*, 2014.

[40] Z. Li, P. Liang, P. Avgeriou, N. Guelfi, and A. Ampatzoglou, "An empirical investigation of modularity metrics for indicating architectural technical debt," in *QoSA 2014*, 2014.

[41] M. Naedele, R. Kazman, and Y. Cai, "Making the case for a "manufacturing execution system" for software development," *Commun. ACM*, 2014.

[42] C. Izurieta, G. Rojas, and I. Griffith, "Preemptive management of model driven technical debt for improving software quality," in *QoSA 2015*, 2015.

[43] H. Sneed, "Dealing with technical debt in agile development projects," *Lecture Notes in Business Information Processing*, 2014.

[44] J. Yli-Huumo, A. Maglyas, and K. Smolander, "The sources and approaches to management of technical debt: A case study of two product lines in a middle-size finnish software company," *Lecture Notes in Computer Science*, 2014.

[45] Y. Guo, R. Spínola, and C. Seaman, "Exploring the costs of technical debt management a case study," *Empirical Software Engineering*, 2014.

[46] A. Nugroho, J. Visser, and T. Kuipers, "An empirical model of technical debt and interest," in *International Workshop on Managing Technical Debt (MTD)*, 2011.

[47] J. de Groot, A. Nugroho, T. Back, and J. Visser, "What is the value of your software?" in *International Workshop on Managing Technical Debt (MTD)*, 2012.

[48] C. Fernández-Sánchez, J. Díaz, J. Pérez, and J. Garbajosa, "Guiding flexibility investment in agile architecting," in *HICSS*, 2014.

[49] V. Singh, W. Snipes, and N. A. Kraft, "A framework for estimating interest on technical debt by monitoring developer activity related to code comprehension," in *International Workshop on Managing Technical Debt (MTD)*, 2014.

[50] E. Alzaghoul and R. Bahsoon, "Evaluating technical debt in cloud-based architectures using real options," in *ASWEC 2014*, 2014.

[51] A. Mayr, R. Plosch, and C. Korner, "A benchmarking-based model for technical debt calculation," in *QSIC 2014*, 2014.

[52] E. Alzaghoul and R. Bahsoon, "Cloudmtd: Using real options to manage technical debt in cloud-based service selection," in *International Workshop on Managing Technical Debt (MTD)*, 2013.

[53] N. Ramasubbu and C. Kemerer, "Towards a model for optimizing technical debt in software products," in *International Workshop on Managing Technical Debt (MTD)*, 2013.

[54] R. Nord, I. Ozkaya, P. Kruchten, and M. Gonzalez-Rojas, "In search of a metric for managing architectural technical debt," in *WICSA/ECSA*, 2012.

[55] A. Martini, J. Bosch, and M. Chaudron, "Architecture technical debt: Understanding causes and a qualitative model," in *SEAA 2014*, 2014.

[56] J. Ho and G. Ruhe, "When-to-release decisions in consideration of technical debt," in *MTD workshop*, 2014.

[57] I. Griffith, H. Taffahi, C. Izurieta, and D. Claudio, "A simulation study of practical methods for technical debt management in agile software development," in *WSC 2014*, 2014.

[58] J. Brondum and L. Zhu, "Visualising architectural dependencies," in *International Workshop on Managing Technical Debt (MTD)*, 2012.

[59] J. Bohnet and J. Döllner, "Monitoring code quality and development activity by software maps," in *International Workshop on Managing Technical Debt (MTD)*, 2011.

[60] P. Clements and L. Northrop, *Software Product Lines: Practices and Patterns*. Addison-Wesley Professional, 2001.

[61] P. Bourque and R. E. Fairley, *Guide to the Software Engineering Body of Knowledge - SWEBOK v3.0*, 2014th ed., P. Bourque and R. E. Fairley, Eds. IEEE CS, 2014.

[62] C. Fernández-Sánchez, J. Garbajosa, C. Vidal, and A. Yagüe, "An analysis of techniques and methods for technical debt management: a reflection from the architecture perspective," in *SAM 2015*, 2015.

[63] A. C. Tricco, J. Tetzlaff, M. Sampson, D. Fergusson, E. Cogo, T. Horsley, and D. Moher, "Few systematic reviews exist documenting the extent of bias: a systematic review," *Journal of Clinical Epidemiology*, vol. 61, no. 5, pp. 422 – 434, 2008.