Information and Software Technology 70 (2016) 100–121

Contents lists available at ScienceDirect

Information and Software Technology

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

Identification and management of technical debt: A systematic mapping study

Nicolli S.R. Alves[b], Thiago S. Mendes[a,d], Manoel G. de Mendonça[a], Rodrigo O. Spínola[a,b,*], Forrest Shull[e], Carolyn Seaman[c]

[a] Fraunhofer Project Center for Software and Systems Engineering, Federal University of Bahia (UFBA), Salvador, Bahia, Brazil
[b] Graduate Program in Systems and Computer, Salvador University, Salvador, Bahia, Brazil
[c] Department of Information Systems, University of Maryland Baltimore County, Baltimore, MD, USA
[d] Information Technology Department, Federal Institute of Bahia (IFBA), Santo Amaro, Bahia, Brazil
[e] Carnegie Mellon University, Software Engineering Institute, Arlington, VA, USA

ARTICLE INFO

Article history:
Received 11 March 2015
Revised 24 October 2015
Accepted 24 October 2015
Available online 1 November 2015

Keywords:
Technical debt
Software maintenance
Software engineering
Systematic mapping

ABSTRACT

Context: The technical debt metaphor describes the effect of immature artifacts on software maintenance that bring a short-term benefit to the project in terms of increased productivity and lower cost, but that may have to be paid off with interest later. Much research has been performed to propose mechanisms to identify debt and decide the most appropriate moment to pay it off. It is important to investigate the current state of the art in order to provide both researchers and practitioners with information that enables further research activities as well as technical debt management in practice.
Objective: This paper has the following goals: to characterize the types of technical debt, identify indicators that can be used to find technical debt, identify management strategies, understand the maturity level of each proposal, and identify what visualization techniques have been proposed to support technical debt identification and management activities.
Method: A systematic mapping study was performed based on a set of three research questions. In total, 100 studies, dated from 2010 to 2014, were evaluated.
Results: We proposed an initial taxonomy of technical debt types, created a list of indicators that have been proposed to identify technical debt, identified the existing management strategies, and analyzed the current state of art on technical debt, identifying topics where new research efforts can be invested.
Conclusion: The results of this mapping study can help to identify points that still require further investigation in technical debt research.

1. Introduction

The technical debt (TD) metaphor was first mentioned by Ward Cunningham in 1992 [7]. His definition, "not-quite-right code", remains the most commonly cited, but it has been extended to refer to those internal software development tasks chosen to be delayed, but that run a risk of causing future problems if not done eventually. Thus, it describes the debt that the development team incurs when it opts for an easy or quick approach to implement in the short term, but with a greater possibility of a negative long-term impact.

Debt can refer to any aspect of the software that we know is inappropriate, but do not have time to fix at the moment, such as outdated/missing documentation, planned testing that is not executed, overly complex code that needs to be restructured or refactored, and known defects that remain uncorrected. The result of these immature artifacts is observed in unexpected delays in carrying out necessary modifications, and in difficulties meeting the established quality criteria of the project [Spínola et al., 2013] [Zazworka et al., 2013].

TD is usually incurred in software projects when there is a need to choose between maintaining the quality standards of the system, and putting the software to work in the shortest possible time, using minimal resources. These TD "items", or instances, may have to be paid with interest later in the project. Translating this metaphor into a tractable model for analysis, we identify the following variables:

- The *principal* on the debt refers to the cost to eliminate the debt (i.e. the effort required to complete the task);

* Corresponding author at: Graduate Program in Systems and Computer, alvador University, Salvador, Bahia, Brazil. Tel.: +55 71 3330 4630.
E-mail addresses: nicollirioss@gmail.com (N.S.R. Alves), thiagomendes@dcc.ufba.br (T.S. Mendes), manoel.mendonca@ufba.br (M.G. de Mendonça), rodrigo.spinola@fpc.ufba.br, rodrigo.spinola@pro.unifacs.br (R.O. Spínola), fjshull@sei.cmu.edu (F. Shull), cseaman@umbc.edu (C. Seaman).

http://dx.doi.org/10.1016/j.infsof.2015.10.008
0950-5849/© 2015 Elsevier B.V. All rights reserved.

- The *interest amount* is the potential penalty in terms of increased effort and decreased productivity that will have to be paid in the future as a result of not completing these tasks in the present [Seaman and Guo, 2011], including the extra cost of paying off the debt later, as compared to earlier;
- It is also necessary to consider the *interest probability*, because TD will not always bring negative impacts on future project activities. For example, the higher the probability that the artifact that contains the debt will undergo maintenance, the higher the probability that the interest will negatively impact the project.

To illustrate the aforementioned variables, we can imagine a scenario where a software product, over time, becomes highly coupled and contains many redundant modules. Reducing the coupling and cleaning up the code constitutes the *principal* on this debt. Although the software may be functioning properly, any addition of new functionalities may be time consuming and require extra effort to deal with the coupling or redundancy issues. The probability that extra effort will be required is the *interest probability*, while the amount of extra effort that is likely is the *interest amount*. Although such design decisions do no harm in the current stage, or may even have benefits such as reduced design time, these immature artifacts can be seen as a type of debt that may burden software maintenance in the future.

Despite similarities between terms and concepts that are used, technical debt is not the same as financial debt. The major difference is that the interest associated with technical debt may or may not need to be paid off [9]. By incurring technical debt, software managers can trade off software quality against productivity. If on one side maintenance time or cost is reduced in the short term (which is the main advantage of incurring technical debt), on the other side, this advantage is achieved at the cost of extra work in the future [9]. Therefore, software managers have to balance the costs and benefits of technical debt and make informed decisions on when and what technical debt should be paid off [Lim et al. 2012].

In order to ensure productivity in the short term and at the same time monitor the progress of the project so that incurred debt doesn't impede the development of the project, TD management techniques have started to be developed [Seaman et al., 2012]. These techniques are generally concerned with identifying and monitoring TD items (instances of technical debt) so that they are explicit and are paid at the right time.

But even before we can effectively work on the management of debt, we need to know what types of debt can be incurred, how they can be identified, and what strategies can be used to manage it. Although technical debt is being increasingly discussed, as reported by trends.google.com indicating that over the last seven years more and more Google users have been searching for the term “technical debt”), it is still difficult to have a broad understanding of the area because the information about it is still spread out in the technical literature.

Beyond a general investigation of technical debt identification and management techniques, we focus in particular on software visualization techniques. Software visualization techniques have been used in software engineering as a possible solution to the task of software systems understanding. Software visualization uses visual aids to facilitate software comprehension [14]. While it seems clear that tools that have been found useful for software comprehension should be highly useful in the identification and management of technical debt, it is still not clear how visualization techniques can support TD related activities. Thus, in this study, we specifically examine the literature that suggests ways that this might be done.

In this context, this work presents a systematic mapping of the literature, conducted to address the following high-level research question: “***What are the strategies that have been proposed to identify or manage TD in software projects?***”. The following complementing research questions were derived from the main question:

- **(Q1)** What are the types of TD?
- **(Q2)** What are the strategies proposed to identify TD?
  - **(Q2.1)** Which empirical evaluations have been performed?
  - **(Q2.2)** Which artifacts and data sources have been proposed to identify the TD?
  - **(Q2.3)** Which software visualization techniques have been proposed to identify TD?
- **(Q3)** What strategies have been proposed for the management of TD?
  - **(Q3.1)** Which empirical evaluations have been performed?
  - **(Q3.2)** Which software visualization techniques have been proposed to manage TD?

By answering these questions, in this study, we have identified the types of TD, the indicators of their existence in projects, and the strategies that have been developed for the management of this debt. Further, we assess the degree of maturity of the existing proposals through an analysis of the empirical evaluations that have been carried out. In addition, we also investigated how software visualization capabilities have been used to support the identification and management of TD by identifying which visual metaphors have been proposed and what platforms are being used to show the different types of debt. These results contribute to the evolution of the TD Landscape [Izurieta et al., 2012].

We believe that the results of the study presented in this paper will be beneficial for both researchers and practitioners. For the research community, this mapping will provide information about the current status of TD research, as well as topics that require further investigation. For practitioners, the paper shows the types of TD currently considered, as well as strategies for their identification and management. Professionals may use this information as a basis for adapting and developing strategies to control the TD in their projects.

Besides this introduction, this paper has seven other sections. Section 2 discusses some related work. In Section 3, the methodology used in this work is presented. Section 4 presents our implementation of the research methodology, including the process of defining the addressed research questions, the study selection process, and the classification scheme we used. Next, in Section 5, the results of the systematic mapping are shown. Section 6 discusses the results, compares them to related work, and presents implications for practitioners and researchers. Section 7 presents the threats to the validity of the study. Finally, Section 8 presents the conclusions of this work and directions for further research.

2. Related work

Technical debt has been increasingly investigated in recent years. An indicator of this trend is the existence of five other secondary studies in the area. In this section, we will discuss in chronological order the goals and results of each study.

Tom et al. (2012) presented, to the best of our knowledge, the first secondary study in the area. By performing a systematic review, Tom et al. intended to provide a consolidated understanding of the TD phenomenon (research questions: *What are the elements of technical debt? Why does technical debt arise?*), to reflect this consolidated understanding in the form of a theoretical framework, and discuss the positive and negative outcomes of TD (research question: *What are the benefits and drawbacks of allowing technical debt to accrue?).* According to the authors, the resulting theoretical framework portrayed a holistic view of TD that incorporates a set of precedents and outcomes, as well as the phenomenon itself (behaviors, metaphors, and elements). In another secondary study in the area, Villar and Matalonga [19] performed an initial mapping of the area by answering the following research questions: