The Journal of Systems and Software 101 (2015) 193–220

Contents lists available at ScienceDirect

The Journal of Systems and Software

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

A systematic mapping study on technical debt and its management

Zengyang Li[a,*], Paris Avgeriou[a], Peng Liang[b,c]

[a] Department of Mathematics and Computing Science, University of Groningen, Nijenborgh 9, 9747 AG Groningen, The Netherlands
[b] State Key Lab of Software Engineering, School of Computer, Wuhan University, Luojiashan, 430072 Wuhan, China
[c] Department of Computer Science, VU University Amsterdam, De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands

ARTICLE INFO

Article history:
Received 11 July 2014
Revised 10 December 2014
Accepted 10 December 2014
Available online 16 December 2014

Keywords:
Systematic mapping study
Technical debt
Technical debt management

ABSTRACT

Context: Technical debt (TD) is a metaphor reflecting technical compromises that can yield short-term benefit but may hurt the long-term health of a software system.
Objective: This work aims at collecting studies on TD and TD management (TDM), and making a classification and thematic analysis on these studies, to obtain a comprehensive understanding on the TD concept and an overview on the current state of research on TDM.
Method: A systematic mapping study was performed to identify and analyze research on TD and its management, covering publications between 1992 and 2013.
Results: Ninety-four studies were finally selected. TD was classified into 10 types, 8 TDM activities were identified, and 29 tools for TDM were collected.
Conclusions: The term "debt" has been used in different ways by different people, which leads to ambiguous interpretation of the term. Code-related TD and its management have gained the most attention. There is a need for more empirical studies with high-quality evidence on the whole TDM process and on the application of specific TDM approaches in industrial settings. Moreover, dedicated TDM tools are needed for managing various types of TD in the whole TDM process.

1. Introduction

Technical debt (TD) is a metaphor reflecting technical compromises that can yield short-term benefit but may hurt the long-term health of a software system. This metaphor was initially concerned with software implementation (i.e., at code level), but it has been gradually extended to software architecture, detailed design, and even documentation, requirements, and testing (Brown et al., 2010). Although the technical debt metaphor was proposed two decades ago, it has only received significant attention from researchers in the past few years.

TD can do both good and harm to a software project. TD that is intentionally incurred (to achieve some short-term benefit) can be fruitful (Allman, 2012) if the cost of the TD is kept visible and under control. In some cases, the development team may choose to take some TD in order to obtain business value. For instance, incurring TD can speed up the development of new features, thus helping the company move ahead of competition. On the other hand, TD can also be incurred unintentionally, meaning that the project manager and development team are not aware of the existence, location, and consequences of the TD. If left invisible and unresolved, TD can be accumulated incrementally, which in turn results in challenges for maintenance and evolution tasks.

Both intentional and unintentional TD (McConnell, 2008) should be managed in order to keep the accumulated TD under control (Lim et al., 2012). TD management (TDM) includes activities that prevent potential TD (both intentional and unintentional) from being incurred, as well as those activities that deal with the accumulated TD to make it visible and controllable, and to keep a balance between cost and value of the software project.

In order to systematically manage TD, it is necessary to have a clear and thorough understanding on the state of the art of TDM. Different methods and tools have been used, proposed, and developed for TDM, but it is not clear how these methods and tools map to TDM activities. Furthermore, although TD has gained significant attention over the past years, the researchers and practitioners in the TD community perceive the concept of TD in different ways, while ambiguities exist around the inevitable hype of TD. For example it is still unclear what can be classified as TD and what cannot in software development, the compromise of which system quality attributes is considered as TD, and what are the limits of the TD metaphor. Answering these basic questions on the TD concept would help researchers to advance the state of the art and practitioners to appraise and select techniques for TDM in their application context.

In this paper, we report the results of a systematic mapping study broadly examining the concept of TD and its management. A

* Corresponding author. Tel.: +31503633968.
E-mail address: zengyangli@gmail.com (Z. Li).

http://dx.doi.org/10.1016/j.jss.2014.12.027
0164-1212/© 2014 Elsevier Inc. All rights reserved.

systematic mapping study is a form of secondary study aiming to get a comprehensive overview on a certain research topic, to identify research gaps, and to collect evidence in order to direct future research (Kitchenham and Charters, 2007, Engström and Runeson, 2011). It allows all available studies in a domain to be analyzed at a high level thereby answering broad research questions regarding the current state of the research on a topic (Kitchenham and Charters, 2007). Another form of secondary study is a systematic literature review (SLR), which aims at identifying, evaluating, and interpreting all available studies to answer particular research questions, and requires more in-depth analysis (Kitchenham and Charters, 2007). We selected to conduct a mapping study instead of a SLR because the involved domain of TD is quite broad and we want to include all research literature (excluding gray literature) in the domain and classify it. Our focus is thus not on analyzing particular aspects of the involved domain, but on answering broad questions about the overall domain. This mapping study on TD and its management has the following objectives:

(1) To get a comprehensive understanding of the concept of "technical debt" in software development based on existing research work on TD;
(2) To get an overview of the current state of the research on TDM, including TDM activities, approaches, and tools;
(3) To identify promising directions for future research on TD and its management.

Despite the significant attention to the field of TD, our systematic mapping study indicates that there are no other secondary studies that comprehensively investigate the concept of TD and its management. One other significant work in this area, by Tom et al., reports on a study for understanding the dimensions of TD, the reasons for incurring TD, and the benefits and drawbacks of allowing TD to accrue (Tom et al., 2013). It involves a multivocal literature review (MLR) and is supplemented by interviews with software practitioners and academics to establish the boundaries of the TD phenomenon (Tom et al., 2013). The MLR is based on their previous SLR on TD (Tom et al., 2012), in the sense that the results of that SLR are combined into the MLR. Our mapping study and Tom et al.'s work are complementary to each other, in the following three aspects:

- **Objectives**. Our mapping study mainly aims to get (1) a comprehensive understanding on the concept of TD; (2) an overview of the current state of the research on TDM; (3) promising future research directions. In contrast, the work of Tom et al. (2013) focuses on the dimensions and causes of TD, and the benefits and drawbacks of allowing TD. The first objective of our study has a partial overlap with the study of Tom et al. (2013): the types of TD in our study are similar to the dimensions of TD in Tom et al.'s work. However, we collected more types of TD than the dimensions of TD in Tom et al.'s work, and we further classified the TD types into sub-types of TD. We also investigated several other aspects of TD that were not studied in Tom et al.'s work, including the studied and under-studied TD (sub-)types, TD-related notions, the compromised quality attributes when TD is incurred, and the limits of the TD metaphor. In contrast, Tom et al. looked into the reasons for TD, and the benefits and drawbacks of incurring TD, which were not investigated in our work. Thus, our mapping study and the work of Tom et al. are complementary to each other for the purpose of covering the whole research field of TD.
- **Methodology**. Both are secondary studies on the topic of TD. Our work applied a systematic mapping study method, while the work of Tom et al. used a SLR and MLR. The differences between the systematic mapping study and SLR methods are the type of research questions asked and analysis conducted on the literature review (Kitchenham and Charters, 2007). As aforementioned, a systematic mapping study provides demographics and classifications to answer broad research questions about a particular topic, while a SLR provides in-depth analysis to answer more specific research questions of the topic investigated (Kitchenham and Charters, 2007).
- **Primary studies**. First, our mapping study examined the research work published from 1992 to 2013, while the work of Tom et al. (2013) systematically checked the publications before 2011 (they conducted the SLR in 2011). Second, our mapping study only includes peer-reviewed publications as primary studies, while Tom et al. (2013) includes both peer-reviewed publications and web blogs and articles. Third, our mapping study selected 94 peer-reviewed primary studies, compared with 19 peer-reviewed primary studies in Tom et al. (2013); this can be partly explained by the large number of studies on TD published in the last two years as shown in Fig. 5. In addition, the study of Tom et al. also included around 35 web blogs and articles. In the work of Tom et al., they described that they also reviewed the papers published in the Managing Technical Debt (MTD) workshops in 2010–2012, and in the IEEE Software special issue on TD (November/December 2012), but they included these papers as additional sources instead of primary studies. Thus, the set of the primary studies in our mapping study is significantly different from that of Tom et al.'s.

The remainder of this paper is structured as follows: Section 2 describes the research questions of this systematic mapping study. The procedure of this mapping study is detailed in Section 3. Section 4 presents the synthesis results of the extracted data from the selected studies and answers the research questions. Section 5 discusses the mapping study results and their implications to researchers and practitioners. Section 6 discusses the threats to validity of this mapping study, and Section 7 presents the conclusions drawn in this mapping study.

2. Research questions

The goal of this study, described using the Goal-Question-Metric approach (Basili, 1992), is: to analyze *primary studies on TD* for the purpose of *getting a comprehensive understanding* with respect to the *TD concept and TDM*, from the point of view of *researchers and practitioners* in the context of *software development*.

To achieve the objectives presented in Section 1, this mapping study will answer the following research questions (RQs) classified into two categories below. The answers to these two categories of RQs can be linked to the objectives of this mapping study: a comprehensive understanding of the concept of TD (Category 1 of RQs), an overview of the current research on TDM (Category 2 of RQs), and promising future research directions on TD and TDM (Categories 1 and 2 of RQs).

(1) RQs on the TD concept
The RQs in this category concern the overall concept of TD. The answers to these RQs can provide us with a comprehensive understanding on TD.
RQ1: *What are the types of TD and what is not considered as TD?*
Rationale: A TD type refers to a specific category of TD (e.g., architectural, design, code) or a sub-category based on the cause of TD (e.g., architectural TD can be caused by architecture smells). TD can also be classified in other ways, such as strategic and non-strategic TD, or the TD quadrant (Fowler, 2009). However, in this mapping study we focus on the classification of TD according to the phases of the software development lifecycle, as this can help stakeholders in different roles (e.g., requirements engineer, architect, test engineer) become aware of what TD may be incurred during the development phases that they are involved. By answering this RQ, we can list the types of TD and potentially shed light on some conflicting viewpoints on these types. In addition, not all things that are detrimental