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/© 2015 Elsevier B.V. All rights reserved.

## 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].