ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/320081646

Managing Architectural Technical Debt: A unified model and systematic literature review

Article *in* Journal of Systems and Software · September 2017

DOI: 10.1016/j.jss.2017.09.025

CITATIONS
60

READS
918

3 authors:

Terese Besker
RISE Research Institutes of Sweden
27 PUBLICATIONS **354** CITATIONS
SEE PROFILE

Antonio Martini
University of Oslo
77 PUBLICATIONS **959** CITATIONS
SEE PROFILE

Jan Bosch
Chalmers University of Technology
541 PUBLICATIONS **16,189** CITATIONS
SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Managing Architectural Technical Debt View project

Architecting with Blockchain View project

All content following this page was uploaded by Terese Besker on 06 October 2017.

The user has requested enhancement of the downloaded file.

A systematic literature review and a unified model of ATD

Terese Besker
Computer Science and Engineering, Software Engineering
Chalmers University of Technology
Göteborg, Sweden
Besker@chalmers.se

Antonio Martini
Computer Science and Engineering, Software Engineering
Chalmers University of Technology
Göteborg, Sweden
Antonio.Martini@chalmers.se

Jan Bosch
Computer Science and Engineering, Software Engineering
Chalmers University of Technology
Göteborg, Sweden
Jan.Bosch@chalmers.se

***Abstract*— Fast software deliveries are hindered by high maintenance efforts due to internal quality issues and Technical Debt (TD) and specifically, Architectural Technical Debt (ATD) has received increased attention in the last few years. ATD has a significant influence and impact on system success and, left unchecked, it can cause expensive repercussions; it is, therefore, of maintenance and evolutionary importance to understand the basic underlying factors of ATD. Thus, with this as background, there is a need for a descriptive model to illustrate and explain the different ATD issues. The aim of this study is to synthesize and compile research efforts with the goal of creating new knowledge with a specific interest in the ATD field. The contribution of this paper is the presentation of a novel descriptive model, providing a comprehensive interpretation of the ATD phenomenon. This model categorizes the main characteristics of ATD and reveals their corresponding relations. The model is based on a systematic literature review (SLR) of currently recognized knowledge concerning ATD.**

Keywords—Systematic literature review; Architectural Technical Debt; Software Maintenance; Software Architecture

I. INTRODUCTION

Large scale software companies strive to increase their efficiency in each life-cycle phase, by reducing time and resources deployed by the development teams. To achieve this goal of delivering high-quality systems, the software architecture is essentially important, and should contribute to a minimal maintenance effort. Van Vliet [1] states that maintenance activities consume 50-70% of the total effort spent during a typical software project. Left unchecked, these maintenance activities can make the architecture diverging towards a suboptimal state, and possibly towards system obsolescence or failure.

Ward Cunningham [2] introduced the financial metaphor of Technical Debt (TD) to describe the need for recognizing the potential long-term and far-reaching negative effects of immature code, made during the software development life-cycle, which has to be repaid with interests in the long term.

The reason for taking on TD can be described as an unintentional consequence of accumulations of decision over time due to the context's evolution and the invisible aspects of natural software aging or, as in Cunningham's definition, in order to get new functionality running quickly.

During development of large scale systems, software architecture plays a significant important role [3] and consequently a vital part of the overall TD relates to sub-optimal architectural decisions and is regarded as Architecture Technical Debt (ATD) [4]. ATD is primarily incurred by architectural decisions with the consequence of immature architectural artifacts. ATD commonly refers to violations of best practices [5], or consistency and integrity constraints of the architectures, or implementation of immature architecture techniques.

This study focuses on different ATD categories, and their related effects, and thereby, become highly relevant when providing a platform for analyzing different architectural management strategies, solutions, and challenges. So far, we have not found any studies describing this issue in a unified way, which could facilitate the challenges of understanding and manage ATD in an overall context. To explore and understand these concerns in a more comprehensive context, a systematic literature review is conducted in the area of ATD with research questions focusing on the current knowledge regarding debt, interest, principal and existing challenges, and solutions in managing ATD.

The main contribution of this paper consists of two parts: First, it shows that there is not one unified, and overarching description or interpretation for ATD and, therefore, a 'state-of-the-art' review of significant issues is provided, concerning various ATD issues.

The second contribution of this article is a novel descriptive model that provides an overall understanding concerning knowledge currently of interest in the research area of ATD. The objective of this study has been achieved through employing a thorough Systematic Literature Review research method (SLR) [6].

The remainder of this paper is structured in seven sections, where the first section is a brief introduction and incentive for this paper. The second section introduces background information that is used during a discussion of the results. In the third section, the research method is described. The fourth section presents the result that is discussed in section five. Finally, in section six, threats to validity are presented, and the last section, seven, concludes the paper.

II. BACKGROUND AND RELATED WORK

In order to provide the reader with necessary information that is needed to better understand the remainder of the paper, this section provides a background to the ATD domain. In this broad view, we examine what constitutes ATD (debt, interest, and principal) and how it is managed (management processes, current challenges and analyzing support). This section also describes related work in the research area of ATD.

A. Background

ATD is a metaphor representing architectural technical issues, including the corresponding financial term debt, interest, and principal.

Interest is the most commonly used financial glossary used in TD research and Ampatzoglou et al. [7] define interest as: "*The additional effort that is needed to be spent on maintaining the software, because of its decayed design-time quality.*" Interest is the negative effect of the extra effort that has to be paid, due to the accumulated amount of *debt* in the system, such as executing manual processes that potentially could be automated or excessive effort spent on modifying unnecessarily complex code, performance problems due to lower resource usage by inefficient code [8], [9].

The *principal* is the cost of remediating planned software violations concerning TD. The aggregate of the principal can be computed as a combination of the number of violations, the hours to correct each violation, and the cost of labor [10].

The objectives of different software management activities performed during the *ATD Management* (ATDM) process include several different activities such as identification, measurement, prioritization, repayment, and monitoring [11].

Despite the significant need for supporting tools and methods for analyzing ATD, most of the available tools for analyzing TD focus on code level instead of architectural aspects [12]. These code focusing tools usually cannot provide indicative for architectural trade-off since they can cause misleading results [13].

One of the most challenging tasks encountered in the synthesis of ATD, is how to translate architectural complications or debt into economic consequences, in means of predicting financial implications based on estimated values. A second key challenge is an estimation that arrives at a strict probability without evidence-based historical data that can provide an accurate estimate at the outset.

B. Related work

Architecture plays a significant role in the development of large software systems and despite that ATD is commonly recognized as a major dimension of TD [8], there are no literature reviews, summarizing the research state of the art on the architectural aspect of TD, to the best of our knowledge. By studying the type of available literature reviews, four reviews (written in English) were found within the shared research area as this study. These studies differ, however, on the research questions, method, searching interval and primary focus, and the studies have different research goals. The most recent study was a systematic mapping review performed by Alves et al. [14], focusing, among other things, different types and useful indicators for detecting TD. Another significant related work in this research area is conducted by Ampatzoglou et al. [7] based on a comprehensive, systematic literature review with a primary focus on financial aspects of managing TD. This review also comprises a formation of a glossary of terms and a classification of financial approaches in TD management. Li et al. [15] published recently a mapping study (MS) with the primary focus on making a classification and thematic analysis on collected studies within the TD and TD management area. Finally, Tom et al. [8] published in 2013 a multivocal literature review (MLR) with the focus on the nature of TD and its implications for software development.

III. REVIEW METHOD

The following section describes the method used for conducting this SLR.

C. Systematic literature review method

The review procedure is based on guidelines by an established SLR method, described by Kitchenham [16]. The main rationale for undertaking a systematic review is to synthesize existing work, and that the review should be carried out in accordance with a predefined search strategy, which allows the search to be evaluated. The major advantage of using this method is that the result is provided by evidence, which is robust and transferable and that sources of variation can be studied further [16]. The method includes a review protocol (Fig. 1), which involves the phases: a) define research questions, b) define search process, c) define inclusion and exclusion criteria, d) define quality assessments, e) define data collection, f) define data analysis, and g) define deviations from protocol.

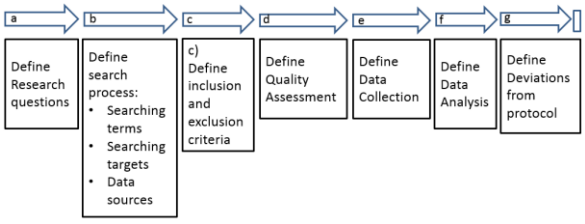

Fig. 1. Visualization of the overall review protocol

D. Research question

As mentioned in Section I, the goal of this review is to gain further knowledge on ATD regarding its constituents (principal, interest and debt) and its management (challenges and solutions). We, therefore, define the following two main Research Questions (RQs) and related six sub-questions. Each sub-question is related to an aspect of ATD, defined in Section II.

RQ1: **What is the existing knowledge concerning ATD in terms of debt, interest and principal?**

RQ1.1: What are the categories of ATD?

RQ1.2: What are the major negative effects caused by ATD?

RQ1.3: What refactoring strategies are mentioned for managing ATD?

RQ2: What are the existing challenges and solutions for managing ATD?

RQ2.1: Which ATDM activities are mentioned?

RQ2.2: Are there any specific challenges with ATD?

RQ2.3: Are there any analysis methods for detecting and/or evaluating ATD?

The first sub-question RQ1.1 focuses on different categories of ATD that will provide information about how debt is described in the publications. Studying, the outlined positive and negative effects caused by ATD in RQ1.2 will provide an understanding of the effects related to different quality attributes (QA), which will be synthesized as interest. Question RQ1.3 focus on the refactoring strategies revealed in publications, and this will be linked to the principal. The result obtained from studying which different ATD Management activities that are stated in the reviewed publication by utilization of results of RQ2.1 will answer which activities are used as notions in the publications. RQ2.2 focus on different kinds of challenges to the management of ATD, and RQ2.3 concentrates on supporting activities for detecting and evaluating the software architecture.

E. Search process

A searching strategy process should include: (1) defining a searching term (query), (2) defining the target for the searching term, and (3) select different data sources with the aim of identifying candidate publications [16].

Since "Architectural Technical Debt" is not a common expression in the title or the abstract as a concatenated word sequence, publications that both contain the words "technical debt" and architec* were studied.

The *search term* (query) contains the following keywords:

"technical debt" AND architec*

The searching terms were combined using a Boolean AND operator, which entails that publication needs to include both of the terms. To catch terms like "architectural" and/or "architecture", the asterisk character * is used, known as a wildcard, to match one or more inflected form of the searching term.

To increase the likelihood to find publications addressing architectural aspects of TD, the *target of the searching term* is defined to search in both title and abstract.

The *selection of data source*s involved automatic searching in six well-known digital libraries: the ACM Digital Library, IEEExplore, ScienceDirect, SpringerLink, Scopus, and Web of Science.

Additionally, in order to avoid overlooking important publications, we performed a hand-search in all the proceedings of a key conference on the subject: the International Workshop on Managing Technical Debt (MTD Workshop). The search was conducted in December 2015.

F. Inclusion and exclusion criteria

SLRs require explicit inclusion and exclusion criteria to assess the fitness of the content in each possible primary study with respect to the RQs.

The criteria should be based on the research questions, and be applied after the full text have been retrieved [16]. The inclusion and exclusion criteria that have been used in this study are listed in Table I:

TABLE I. INCLUSION AND EXCLUSION CRITERIA

Criteria	Assessment criteria
Inclusion	Publication should define or discuss architectural issues in the context of TD.
Inclusion	Publications published in journals, in conference proceedings, book chapters, and workshop proceedings were decided to be included.
Inclusion	Only publications written in English were included.
Exclusion	A publication that only mentions TD in an introductory statement and does not fully or partly focus on its architectural aspects.
Exclusion	Publication's full text is not available.
Exclusion	Publication where the full paper is not possible to locate.
Exclusion	For conferences and workshop proceedings, publications earlier than the year 2005 were excluded.

G. Quality assessment

To ensure that the selected publications comply with a certain quality, all publications went through a quality assessment process, to evaluate if they were of an adequate standard. In order to make this assessment, each publication´s content was evaluated using a set of questions in a checklist (Table II), where the answers were mapped according to the options on a ranking scale.

The questions (QA1-3) in the checklist, representing different assessment criteria with a focus on three diverse quality assessments pertaining the quality that needs to be considered when evaluating the publications identified in the review. The assessment criteria strive to appraise the quality of the publication (QA1), quality according to the findings and results (QA2), and relevant to an ATD aspect (QA3).

TABLE II. QUALITY ASSESSMENT CRITERIA

Question	Assessment criteria	Response option scale
QA1	Where was the research published?	Journal = 4 Conference = 3 Book = 2 Workshop =1
QA2	Did the publication provide clearly stated findings with credible results and justified conclusions?	Excellent = 3 Good = 2 Fair= 1 Poor = 0
QA3	Did the publication provide a valuable contribution to the review, in terms of the relevance of discussing ATD?	Excellent = 3 Good = 2 Fair= 1 Poor = 0

The scale of QA1 refers to a well-established standard where the Journals are ranked as having the highest reliability, followed by Conferences, Books, and Workshops. QA3 involves comparable requirements, which previously referred to an exclusion criteria, but this examination focuses on scaling the content of the remaining publications. The quality assessment scores are a heuristic only - to be used as a guide, where no publication is rejected on the basis of the quality assessment output.

H. Data collection

Data was collected using the form in Table III, including predefined Data Collection Variables. This enabled recording and tracking of full details of each surveyed publication.

TABLE III. DATA COLLECTION VARIABLES AND THEIR PURPOSE

Variable	Collected Information	Purpose
[DC1]	Author	Demographic characterization
[DC2]	Title	
[DC3]	Year	
[DC4]	Venue	
[DC5]	Quality assessment score	Data assessment
[DC6]	Categories of ATD	RQ1.1
[DC7]	Quality attributes/negative effects	RQ1.2
[DC8]	Refactoring Strategies	RQ1.3
[DC9]	Architectural TDM activities	RQ2.1
[DC10]	Challenges	RQ2.2
[DC11]	Analysis method	RQ2.3

The first data collection variables [DC1] – [DC4] are primarily due to the demographic characterization of the study. Variable [DC5] synthesizes the quality of the publications. The stated ATD categories in the research presented by variable [DC6] provides information when processing RQ1.1, likewise [DC7] reports the quality attributes and possible adverse effects when answering RQ1.2. To scrutinize different refactoring strategies mentioned in the research, the variable [DC8] is added to the data collection. Finally, to answer RQ2 and its sub-queries RQ2.1, RQ2.2 and RQ2.3 variables [DC9], [DC10] and [DC11] with a focus on different types of mentioned ATDM activities together with various challenges and monitoring/detecting methods are stated.

I. Data analysis

For an exploratory data analysis, the software tool Atlas.ti is used, allowing coding and visualization of qualitative information. Fig. 2 shows the outcome of the analysis process, described below: the graph is part of the overall analysis model (not completely displayed here for space reasons).

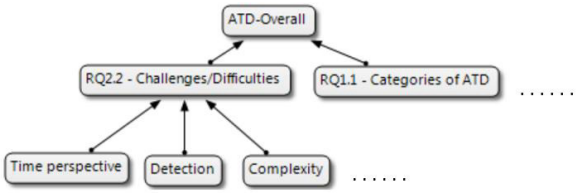

Fig. 2. The outcome of the analysis process

First, as reported in the data collection Section (III-H), the RQs generated DC variables (DC6-11 in Table III), which were used as an inductive coding scheme for the high-level categorization of the data (second level of codes in the Fig. 2). Then the data were inductively analyzed identifying several aspects: these aspects are visible as the bottom-level boxes in Fig. 2. In order to explain how this critical coding step was conducted, an example of a quotation from a paper mapped to a novel aspect and therefore to an RQ is reported.

The quote from [17]: *"Not being able to detect or address architectural decay in time incurs architecture debt that may result in a higher penalty"* was coded as *Time Perspective*, which is part of the challenges related to ATD management (RQ2.2).

During the analysis, we were explicitly observing cause and effect relations between the revealed aspects, which was introduced as relationships between the aspects. The complete result of this process is the *Unified Model* in Fig. 4, where all the aspects and their relationships are shown in a comprehensive way.

IV. RESULTS

The presentation of the results will initially focus on the overall results concerning the retrieval of the primary publications. Then, the result will concentrate on the selected publications, presenting the outcome of the data collection analysis, with respect to the investigated research questions.

A. Screening of retrieved publications

To screen out the most interesting and relevant publications for this review, a filtering technique based on five different stages has been used. Fig. 3 shows the filtering stages included in the searching process and the returned number of publications identified after each filtering stage.

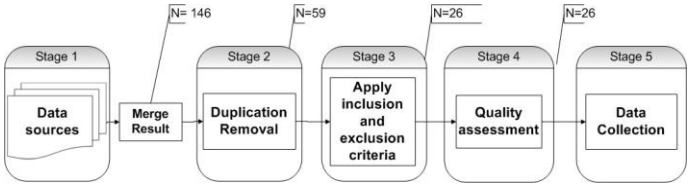

Fig. 3. Stages included in the selection process.

In the first stage, all publications (n = 146) from the different data sources, were retrieved and merged. The software Endnote facilitates of finding and removing duplicates, which resulted in a reduced number of publications (n = 59). Stage three was applied after the full text was retrieved, and the publication was checked using the inclusion and the exclusion criteria in Table I. As a result of this action, another 33 publications were excluded. In a fourth stage, the publication went through an assessment process, with the mission of assessing the quality of the publication. This process did not reduce the number of publication further; this stage serves mainly as a ranking of the publications. During the fifth stage, data was extracted from each of the 26 primary publications included in this SLR, according to the predefined data collection forms, in Table III. The examination of the selected primary publications showed that the most predominant type is a conference paper (88%) and only one (4%) publication was a journal. Two chapters from two books were included, and the most popular venue is the Managing Technical Debt Workshop.

B. Existing knowledge concerning ATD in terms of debt, interest and principal (RQ1)

RQ1 investigates how existing knowledge defines debt, interest, and principal, which are respectively represented by the three following aspects: (1) categories of ATD, (2) the negative effects caused by ATD, and (3) strategies for refactoring ATD.

1) Categories of ATD (RQ 1.1)

Architectural choices and design decisions have a great impact on the amount of ATD [18], [19], and can be incurred by either explicit or implicit architecture decisions [11] and be made either consciously or unconsciously [20]. These decisions affect several different categories of debt, and one of the main categories of ATD are architectural dependencies [4], [21] including module dependencies, external dependencies, external team dependencies [22]. Another category of ATD involves non-uniformity of patterns and policies where an i.e. violation of naming conventions and non-uniform design or architectural patterns are implemented [4]. Some authors add

code related issues as ATD variables, where lack of code documentation [22], code duplication [4] and overly complex code [23], [24], [25] are additional reasons for the emergence of ATD. Further, non-uniform management of integration with subsystem and resource [4] and different architectural decay instances [17] are also revealed as ATD categories. The lack of implementation or test of Quality Attributes (QA) or non-functional requirements are shown by [4] as a classification of ATD and [26] illustrates the problems with conflicting QA synergies as an important source of ATD.

2) Negative effects caused by ATD (RQ 1.2)

Fernández-Sánchez et al. [27] acknowledge that "*interest is the recurring cost of not eliminating a technical debt item over some period of time*" with the result of the negative effects and adverse impact on QAs. There is wide agreement in academic literature that some of the negative consequences of ATD can be linked to the effect it has on maintenance complications – "*Not being able to detect or address architectural decay in time incurs architecture debt that may result in a higher penalty in terms of quality and maintainability (interest) over time*" [17]. Li et al. [11], [28] mention particularly the QAs maintainability and evolvability, as immature architecture design artifacts. A system suffering from architectural drift or erosion will eventually develop some decay instances that negatively impact the system life-cycle properties, such as understandability, testability, extensibility, and reusability [17]. On the other hand, to proactively assume that changes will happen and in advance design for flexibility, often entails high costs and risk [27].

3) Refactoring strategies for managing ATD (RQ 1.3)

The concept of different refactoring strategies refers to, how to (i.e. if one refactoring must be done before another [29]), and if repaying the debt. A refactoring strategy typically involves decisions regarding continuing paying interest or by paying the principal by re-architecting and refactoring to reduce future interest payments [30]. In our previous paper, by Martini and Bosch [31], a description of a strategy with respect to minimizing risk for a development crisis is recommended - "*Partial refactoring is the best option for the maximization of refactoring. Thorough refactoring is not realistic. Drastic minimization of refactorings (No refactoring) leads to development crises often in the long run*." Based on economic considerations, it could be more profitable to delay the refactoring i.e. continue paying interest than to invest in refactoring to manage the debt and furthermore the team's capacity to perform the refactoring could be considered [27]. Fernández-Sánchez et al. [32] echo the notion stating, "*The basic way to perform a cost-benefit analysis is to compare the cost of removing technical debt (principal) with the benefit obtained.*" Several authors mention this decision-making as the main refactoring strategy and [33] reveal important aspects during this prioritization regarding lead time, maintenance costs, and risks. The refactoring decision making is enclosed by the problem of that costs are concrete and immediate whereas the benefits of refactoring are vague and long-term [34], and the benefits of refactoring have historically been difficult for architects to quantify or justify [34].

C. Existing challenges and solutions in managing ATD (RQ2)

The question RQ2 is separated into three top-level precedents, examining ATDM activities, related challenges, and analysis method for detecting or evaluating ATD.

1) Architectural TDM activities (RQ 2.1)

Nord et al. [35] describes the importance of this process as "*Development decisions, especially architectural ones, require active management and continuous quantitative analysis, as they incur implementation and rework cost to produce value"*. To fully manage and control a complete ATDM process, Li et al. [11] advocate the five different activities: identification, measurement, prioritization, repayment, and monitoring. However, while the concept of ATDM as an overall process is uncommonly described in the academic literature, several authors mention or focus on individual activities within an overarching process of ATDM. The initial activity within the ATDM process is to identify current ATD items within the software system [11], whereas this activity is crucial for a forthcoming successful management process. The following ATD measurement activity examines and estimates the cost and benefit, including the prediction of change scenarios influencing ATD items for interest [11]. The output of this activity is used as an input to the prioritization activity since prioritizing refactoring ATD items is an essential element when balancing the short-term value delivery and the long-term responsiveness where lead time, maintenance costs and risk are the variables that most influence the ATD effects [36]. The repayment activities involve refactoring, and can either be partly or fully resolved by making new or modifying existing architecture decisions [11] to reduce or mitigate the undesirable effects of ATD. The monitoring activity tracks ATD changes explicitly and consequently keeps all the ATD items of the system under control [11]. This activity is a complex process that endures over time, and [35] enforce that the repayment and the monitoring activities are a quite uncommon practice where existing techniques are lacking.

2) Challenges with ATD (RQ 2.2)

To get a rich picture and to reveal issues and risks early [12], it is of vital importance to understand the background of ATD problems. This highlights the need to understand the challenges to fully manage ATD in a more conscious and informed way. Among others, [17] discussed the challenges and were putting the time in perspective to ATD. [22] explain that architectural decisions take many years to evolve and are commonly made early in the software life-cycle and it is often invisible until very late in the life-cycle [37]. From a more technical perspective, critical issues relate to challenges of detection through standard testing [23] and ATD seldom yield observable behaviors to end users [20] whereas a higher number of dependencies decrease the comprehensibility of the system [38]. Yet another purported emergence of ATD is associated with architectural decision-making, where the decisions cross every possible communication link across the development and operations network, and loss of essential information is practically inevitable [39].

3) Analysis method for detecting or evaluating (RQ 2.3)

Using an analysis method may facilitate the evaluation and decision-making process, helping the prioritization of resources and efforts concerning refactoring strategies. The need of supporting tools for system monitoring and evaluating ATD using accurate metrics is a key issue and is not fully supported by any today available tools [22], [4]. The main goal of a continuous and iterative system monitoring of ATD is to capture and track the presence of ATD within a system [4], to provide early warnings to detect costs and risks [35], and to map architectural dependencies or pattern drift to decay [22].

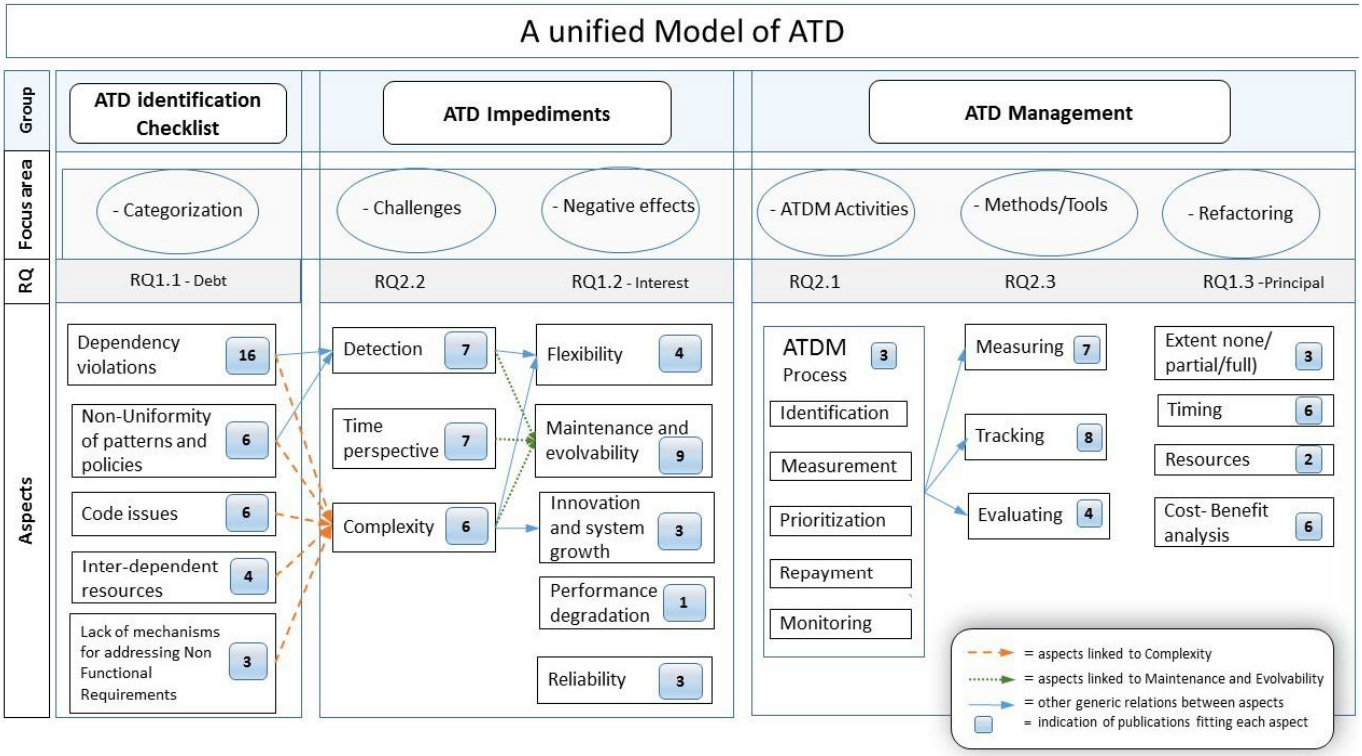

Fig. 4. The Unified Model of ATD

Ozkaya et al. [12] state further that most available metrics are at the code level and [35] describe that existing metrics for system quality visibility are insufficient and unproven.

D. Importance of ATD and a need for a unified model

There is wide agreement in the reviewed academic literature that ATD is of primary importance. Examples of such statements include *"We found that architectural decisions are the most important source of technical debt"* [22], *"Finally, the most encountered instances of technical debt are caused by architectural inadequacies"* [30], and *"Because architecture has such leverage within the overall development life cycle, strategic management of architectural debt is of primary importance"* [37].

It is observed from the result that ATD is described in a scattered and inconsistent way. Consequently, we conclude that to derive more value from the results concerning ATD and its effects, a holistic model depicting different views and their implications at hand is required. Therefore, we propose a novel descriptive model that provides an overall understanding of existing knowledge in the research area of ATD with the aim to provide a comprehensive interpretation of the ATD phenomenon.

E. A unified model for ATD management

The model in Fig. 4 graphically illustrates and synthesize the findings and their causal relationships. The structure of the model is inspired by Nickerson et al. [40] who recommend that a taxonomy of a model includes the criteria of conciseness, inclusiveness, comprehensiveness, and extendibility. The model furthermore, clarifies the different aspects of each research question and assembles relationships between them. As a help to get to deeper levels of observation, the model is divided into the vertical groups *ATD Identification Checklist*, *ATD Impediments*, and *ATD Management* and the horizontal cross-sections representing corresponding *Focus area*, *Research question* and *Aspects*. Within every aspect in Fig.4, there is a box with a number indicating how many papers that fit into each aspect. The references for each aspect are presented in Table IV. This information highlights the popularity of each categorization and contributes to the reader's knowledge base with useful and important information when creating a platform for understanding ATD.

The model's scaffolding allows practitioners to more easily leverage this study's outcome when analyzing the result derived from the RQs. The model's relationships are based on the systematic analysis of all surveyed publications, in which we have interpreted the result into the unified model illustrated by different types of arrows. Relationships between aspects are illustrated by different colored arrows.

The model highlights that Maintainance and Evolvability are vital challenges within ATD, due to all of the ATD challenges (green dotted arrows) are related to this negative effect, and furthermore, all of the ATD categories have an effect on Complexity (orange dashed arrows).

V. Discussion

This section discusses the findings and the implications for practitioners and academia. The result from the SLR indicates that there is an absence of a comprehensive descriptive model of ATD in the academic literature. For the academic and practitioner community, this descriptive model can support the process of more informed management of the software development life-cycle, with the goal of raising the system's success rate and lower the rate of negative consequences.

TABLE IV. REFERENCES FOR EACH ASPECT

Aspect	Reference
Dependency violations	[22],[32],[38],[34],[28],[41],[20],[33],[4],[31],[35],[17],[39],[5],[19],[21]
Non-Uniformity of patterns and policies	[27], [38], [11], [20], [5],[24]
Code issue	[23], [32], [33], [4],[35],[24]
Inter-dependent resources	[31], [26], [4], [17]
Lack of mechanisms for addressing None Functional Requirements	[26], [4], [41]
Detection	[17],[24],[25],[28],[41],[20],[21]
Time perspective	[32],[28],[41],[4],[5],[31],[35]
Complexity	[24],[37],[20],[12], [19],[21]
Flexibility	[27],[26],[32],[17]
Maintenance and evolvability	[24],[11],[28],[41],[20],[25],[4],[5],[31]
Innovation and system growth	[5],[32],[31]
Performance degradations	[23]
Reliability	[23],[26],[17]
ATDM Process	[41],[28],[11]
Measuring	[22],[24],[38],[34],[11],[41],[31]
Tracking	[22],[32],[34],[11],[28],[41],[4],[31]
Evaluating	[22],[34],[11],[31]
Extent none/ partial/full)	[4],[5],[31]
Timing	[24],[33],[4],[5],[31],[35]
Resources	[34],[31]
Cost- Benefit analysis	[34],[33],[5],[31],[35],[29]

A. ATD identification checklist (RQ1.1)

The research question RQ1.1 is addressed with an ATD identification checklist, in the form of a unified model (Fig.4). The current description in the literature of ATD is inconsistent and creates confusion both for practitioners and within academia. Therefore, we provide a comprehensive checklist of all aspects of ATD, reported in the literature. Researchers and practitioners can use this checklist as a universal reference for recognizing ATD. ATD can be categorized into different categories of debt (RQ1.1): dependency violation, code related issues, non-uniform usage of architectural policies, the lack of handling interdependent resources and lack of addressing non-functional requirements.

B. ATD impediments (RQ2.2 and RQ1.2)

In Fig. 4, we present all the ATD Impediments; both the challenges related to ATD and their associated negative effects. Researchers and practitioners can use this picture to evaluate and understand what problems might occur while dealing with ATD and the consequences if these challenges are left unattended. Major challenges (RQ2.2) include difficulty in ATD detection, challenges related to the time perspective and unmanageable architectural complexity. Often the mentioned complexity is related to code level (McCabe's cyclomatic complexity) but this metric is not related to architectural aspects of complexity, and therefore, we stress the need for further focus on this specific aspect.

These overall challenges can lead to severe and costly maintenance overhead that in the long run, can diminish the organization's ability to innovate and evolve. ATD can also result in restricted flexibility, decreased reliability and performance degradation (RQ1.2). Our findings show that challenges and effects are currently presented in a scattered way in literature, and, therefore, we provide a comprehensive presentation of ATD related challenges and effects in Fig. 4.

C. ATD management (RQ2.1, RQ2.3, and RQ1.3).

Although current literature stresses the importance of understanding ATD and its related consequences, we report the lack of guidelines on how to manage ATD successfully in practice.

We confirm that the generic activities recognized by [11] apply to ATD management. However, we report a lack of an overall process where these activities are fully integrated with each other. At the moment, they are mostly reported as single isolated activities. Two book chapters mention these activities within an overall process, but there is a need for further and stronger empirical evidence supporting a suitable and practical solution (RQ2.1).

The findings show that there is a compelling need for supporting tools and methods for system monitoring and evaluating ATD, but also shows that no software tools, covering the full spectrum of ATD are yet available (RQ2.3). For example, [11] propose an ATD identification method, but it is not clear to us how this method could be integrated with other methods and tools.

A refactoring strategy for paying the principal must be developed and implemented to successfully managing ATD and not allowing it to grow unbounded. Practitioners do in general lack strategies for refactoring, and, therefore, such activity might result in an ad-hoc process where the result is inadequate. We provide in this paper, the key dimensions that need to be taken into consideration when defining a refactoring strategy. Fig. 4 includes these dimensions and can assist practitioners and researchers when creating and studying refactoring strategies. A strategy needs to be flexible to adapt to changing requirements and take into account resources constraints, and forthcoming new features. An important aspect to consider while setting up a strategy is the amount of refactoring that should take place. If refactoring is overlooked, it can lead to a development crisis in the long run, and there are benefits of performing a partial refactoring (RQ1.3).

D. The unified model for ATD

As discussed earlier in the previous section, we found that the information about ATD is currently scattered in different publications and sometimes inconsistent. It is, therefore, difficult for researchers and practitioners to get a clear overview of ATD. This study provides a novel and comprehensive model that includes all the important aspects of the ATD phenomenon and their relationships (Fig. 4). The model will help academic and practitioner in interpreting ATD, recognizing its issues and understanding how to manage it.

The model presents different aspects of ATD and their relationships: this helps the practitioners in understanding how the aspects impact each other and assists the researchers in studying the connecting aspects together. For example, the orange dashed arrows in the model show that all revealed categories of ATD have a relation to the aspect of Complexity meaning that all instances of ATD increase the needed effort for software and system engineers to understand and manage the systems' interfaces and interconnections.

The green dotted arrows in the model show that all the Challenges of ATD have effects on system Maintenance and Evolvability meaning that *every ATD* item will have a negative impact on Maintenance and Evolvability.

E. Roadmap for future research on ATD

This roadmap provides clear targets for future research and concludes that future research roadmap should focus on:

- The findings show that there is a compelling need for supporting tools and methods for system monitoring and evaluating ATD. There is also a need for more studies about efficient refactoring strategies, taking different aspects into consideration.
- This study's result underlines the lack of guidelines on how to manage ATD successfully in practice, which needs to be addressed in future research.
- The indication of how many papers are fitting each aspect clearly shows which area that is covered by research and what aspects that need more investigation.

VI. THREATS TO VALIDITY

The result of this SLR may be affected by some threats to validity, such as:

A. Number of retrieved publications

A relatively limited number of publications were retrieved, with the logical consequences of that, the result of this review had limited publications to derive results from. This result could potentially have introduced subjective conclusions into the analysis or incorrect or missing relationships into the results.

B. Search string

During the search process, publications that did not include both the searching terms "technical debt" and architec* in the title or abstract of the publication was excluded from the review. This could imply that some publications could have been incorrectly excluded. We reason, nevertheless, that if a publication should mainly focus on the architectural aspects of TD, the word architec* should be explicitly mentioned. However, we are aware of that there are related terms that in many respects resemble the architectural aspects of TD.

C. Data extraction

To reduce the risk of subjectivity, during the classification and extraction phase, made by only one researcher, some publication was examined by at least two researchers to ensure that the returned publications were suitable, and equivalent analyzed.

D. The model of ATD

In the Model of ATD in Fig.4, other aspects, relationships, and associated impacts may exist but are omitted in this model, since they were not explicitly revealed in the review process and thus have not been further analyzed.

VII. CONCLUSIONS

In a software life-cycle perspective, it is of vital importance to understand and proactively manage ATD. Left unchecked, ATD can potentially stifle implementing new features and organizations may face expensive repercussions due to costly architectural maintenance complication. In order to synthesize and compile the current 'state-of-the-art' in the ATD field, we conducted a systematic literature review focusing on the research areas: ATD in terms of principal, interest, debt and related challenges and solutions for managing ATD.

In this study we have provided a new and comprehensive understanding and raised awareness about what challenges ATD are surrounded by and finally how ATD can be successfully managed. The findings showed that there is wide agreement in the reviewed literature that ATD is of primary importance. ATD is, however, surrounded by several challenges, and while numerous publications mention different isolated ATD management activities, there is an absence of, and a need for a thorough indicative ATDM process for the practitioner and academic communities, covering all these separate activities. Different ATD categories (as debt) can result in various negative consequences (as interest), requesting effective refactoring strategies (as principal). A refactoring strategy mainly refers to, how to, and if, and to what extend repaying the debt, should be formulated.

This research contributes to knowledge that addresses a current gap in understanding, where knowledge and research of ATD are non-unified and fragmented. One key contribution of this paper is our novel model of ATD. This model summarizes our findings and allows improved identification of ATD and associated negative consequences and corresponding ATDM activities. The model illustrates ATD, in a unified and comprehensive way by exploring different aspects and relationships, which are considered particularly valuable for managing and raising awareness about ATD.

The model reveals that all categories of ATD (as debt) are related to the challenge of Complexity and furthermore that all challenges are related to Maintenance and evolvability. This model can help several different stakeholders within the software life-cycle process to better and more informed, manage the software, with the goal of raising the system's success rate and lower the rate of negative consequences.

As future work, we plan to investigate this area further by means of expanding this review to include additional closely related research publications and by applying backward and forward snowballing methods. Also, it would be interesting to validate our model in an empirical context.

To further study the impact and the influence of the different aspects of the unified model of ATD, we are currently conducting empirical research with the aim of finding which aspects are the most hurtful for the software development life-cycle.

ACKNOWLEDGMENT

Many thanks to all the participants in our survey and interviews, as well as to our Software Center industrial partners.

REFERENCES

[1] H. Van Vliet, Software Engineering: Principles and Practice: John Wiley & Sons, 2008.

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

[3] P. Kruchten, R. L. Nord, and I. Ozkaya, “Technical Debt: From Metaphor to Theory and Practice,” Software, IEEE, vol. 29, no. 6, 2012, pp. 18-21.

[4] A. Martini and J. Bosch, "The Danger of Architectural Technical Debt: Contagious Debt and Vicious Circles," in Proceedings - 12th Working IEEE/IFIP Conference on Software Architecture, WICSA 2015, 2015, pp. 1-10.

[5] A. Martini, J. Bosch, and M. Chaudron, "Architecture Technical Debt: Understanding Causes and a Qualitative Model," in Software Engineering and Advanced Applications (SEAA), 2014 40th EUROMICRO Conference on, 2014, pp. 85-92.

[6] B. Kitchenham, O. Pearl Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, “Systematic literature reviews in software engineering – A systematic literature review,” Information and Software Technology, vol. 51, no. 1, 2009, pp. 7-15.

[7] A. Ampatzoglou, A. Ampatzoglou, A. Chatzigeorgiou, and P. Avgeriou, “The financial aspect of managing technical debt: A systematic literature review,” Information and Software Technology, vol. 64, 2015, pp. 52.

[8] E. Tom, A. Aurum, and R. Vidgen, “An exploration of technical debt,” Journal of Systems and Software, vol. 86, no. 6, 2013, pp. 1498-1516.

[9] B. Curtis, J. Sappidi, and A. Szynkarski, "Estimating the size, cost, and types of Technical Debt," in Managing Technical Debt (MTD), 2012 Third International Workshop on, 2012, pp. 49-53.

[10] B. Curtis, J. Sappidi, and A. Szynkarski, “Estimating the Principal of an Application's Technical Debt,” Software, IEEE, vol. 29, no. 6, 2012, pp. 34-42.

[11] Z. Li, P. Liang, and P. Avgeriou, "Architectural Debt Management in Value-Oriented Architecting," in Economics-Driven Software Architecture, ed, 2014, pp. 183-204.

[12] I. Ozkaya, R. L. Nord, H. Koziolek, and P. Avgeriou, "Second International Workshop on Software Architecture and Metrics (SAM 2015)," in Software Engineering (ICSE), 2015 IEEE/ACM 37th IEEE International Conference on, 2015, pp. 999-1000.

[13] I. Macia, J. Garcia, D. Popescu, A. Garcia, N. Medvidovic, and A. v. Staa, "Are automatically-detected code anomalies relevant to architectural modularity?: an exploratory analysis of evolving systems," presented at the Proceedings of the 11th annual international conference on Aspect-oriented Software Development, Potsdam, Germany, 2012.

[14] N. S. R. Alves, T. S. Mendes, M. G. de Mendonça, R. O. Spínola, F. Shull, and C. Seaman, “Identification and Management of Technical Debt: A Systematic Mapping Study,” Information and Software Technology, 2015.

[15] Z. Li, P. Avgeriou, and P. Liang, “A systematic mapping study on technical debt and its management,” Journal of Systems and Software, vol. 101, 2015, pp. 193-220.

[16] B. Kitchenham, “Procedures for performing systematic reviews,” Keele, UK, Keele University, vol. 33, no. 2004, 2004, pp. 1-26.

[17] M. Ran, J. Garcia, C. Yuanfang, and N. Medvidovic, "Mapping architectural decay instances to dependency models," in Managing Technical Debt (MTD), 2013 4th International Workshop on, 2013, pp. 39-46.

[18] N. A. Ernst, "On the role of requirements in understanding and managing technical debt," in Managing Technical Debt (MTD), 2012 Third International Workshop on, 2012, pp. 61-64.

[19] S. Bellomo, N. Ernst, R. Nord, and R. Kazman, "Toward design decisions to enable deployability: Empirical study of three projects reaching for the continuous delivery holy grail," in Proceedings - 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2014, 2014, pp. 702-707.

[20] Z. Li, P. Liang, P. Avgeriou, N. Guelfi, and A. Ampatzoglou, "An empirical investigation of modularity metrics for indicating architectural technical debt," presented at the Proceedings of the 10th international ACM Sigsoft conference on Quality of software architectures, Marcq-en-Bareul, France, 2014.

[21] J. Brondum and L. Zhu, "Visualising architectural dependencies," presented at the Proceedings of the Third International Workshop on Managing Technical Debt, Zurich, Switzerland, 2012.

[22] N. A. Ernst, S. Bellomo, I. Ozkaya, R. L. Nord, and I. Gorton, "Measure it? Manage it? Ignore it? software practitioners and technical debt," presented at the Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, Bergamo, Italy, 2015.

[23] B. Curtis, J. Sappidi, and A. Szynkarski, "Estimating the size, cost, and types of technical debt," in 2012 3rd International Workshop on Managing Technical Debt, MTD 2012 - Proceedings, 2012, pp. 49-53.

[24] F. A. Fontana, V. Ferme, and M. Zanoni, "Towards Assessing Software Architecture Quality by Exploiting Code Smell Relations," in Proceedings - 2nd International Workshop on Software Architecture and Metrics, SAM 2015, 2015, pp. 1-7.

[25] E. Ligu, A. Chatzigeorgiou, T. Chaikalis, and N. Ygeionomakis, "Identification of refused bequest code smells," in IEEE International Conference on Software Maintenance, ICSM, 2013, pp. 392-395.

[26] B. Boehm, "Architecture-Based Quality Attribute Synergies and Conflicts," in Proceedings - 2nd International Workshop on Software Architecture and Metrics, SAM 2015, 2015, pp. 29-34.

[27] C. Fernández-Sánchez, J. Díaz, J. Pérez, and J. Garbajosa, "Guiding flexibility investment in agile architecting," in Proceedings of the Annual Hawaii International Conference on System Sciences, 2014, pp. 4807-4816.

[28] Z. Li, P. Liang, and P. Avgeriou, "Architectural Technical Debt Identification Based on Architecture Decisions and Change Scenarios," in Proceedings - 12th Working IEEE/IFIP Conference on Software Architecture, WICSA 2015, 2015, pp. 65-74.

[29] K. Schmid, "A formal approach to technical debt decision making," presented at the Proceedings of the 9th international ACM Sigsoft conference on Quality of software architectures, Vancouver, British Columbia, Canada, 2013.

[30] J. Holvitie, V. Leppanen, and S. Hyrynsalmi, "Technical Debt and the Effect of Agile Software Development Practices on It - An Industry Practitioner Survey," in Managing Technical Debt (MTD), 2014 Sixth International Workshop on, 2014, pp. 35-42.

[31] A. Martini, J. Bosch, and M. Chaudron, “Investigating Architectural Technical Debt accumulation and refactoring over time: A multiple-case study,” Information and Software Technology, vol. 67, 2015, pp. 237-253.

[32] 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 Proceedings - 2nd International Workshop on Software Architecture and Metrics, SAM 2015, 2015, pp. 22-28.

[33] A. Martini and J. Bosch, "Towards Prioritizing Architecture Technical Debt: Information Needs of Architects and Product Owners," in Software Engineering and Advanced Applications (SEAA), 2015 41st Euromicro Conference on, 2015, pp. 422-429.

[34] R. Kazman, C. Yuanfang, M. Ran, F. Qiong, X. Lu, S. Haziyev, et al., "A Case Study in Locating the Architectural Roots of Technical Debt," in Software Engineering (ICSE), 2015 IEEE/ACM 37th IEEE International Conference on, 2015, pp. 179-188.

[35] R. L. Nord, I. Ozkaya, P. Kruchten, and M. Gonzalez-Rojas, "In search of a metric for managing architectural technical debt," in Proceedings of the 2012 Joint Working Conference on Software Architecture and 6th European Conference on Software Architecture, WICSA/ECSA 2012, 2012, pp. 91-100.

[36] A. Martini, L. Pareto, and J. Bosch, "Towards Introducing Agile Architecting in Large Companies: The CAFFEA Framework," in Agile Processes, in Software Engineering, and Extreme Programming. vol. 212, C. Lassenius, T. Dingsøyr, and M. Paasivaara, Eds., ed: Springer International Publishing, 2015, pp. 218-223.

[37] P. Kruchten, "Strategic management of technical debt: Tutorial synopsis," in Proceedings - International Conference on Quality Software, 2012, pp. 282-284.

[38] C. Izurieta, G. Rojas, and I. Griffith, "Preemptive Management of Model Driven Technical Debt for Improving Software Quality," presented at the Proceedings of the 11th International ACM SIGSOFT Conference on Quality of Software Architectures, Montréal, QC, Canada, 2015.

[39] D. A. Tamburri and E. D. Nitto, "When Software Architecture Leads to Social Debt," in Proceedings - 12th Working IEEE/IFIP Conference on Software Architecture, WICSA 2015, 2015, pp. 61-64.

[40] R. Nickerson, J. Muntermann, U. Varshney, and H. Isaac, "Taxonomy development in information systems: developing a taxonomy of mobile applications, in: 17th European Conference in Information Systems (ECIS ‘09), Italy," 2009, pp. 1138–1149.

[41] Z. Li, P. Liang, and P. Avgeriou, "Chapter 5 - Architecture viewpoints for documenting architectural technical debt," in Software Quality Assurance, I. M. S. A. G. Tekinerdogan, Ed., ed Boston: Morgan Kaufmann, 2016, pp. 85-132.

View publication stats