Architectural Technical Debt Identification: the Research Landscape

Roberto Verdecchia*,†, Ivano Malavolta†, Patricia Lago†
*Gran Sasso Science Institute, L’Aquila, Italy - roberto.verdecchia@gssi.it
†Vrije Universiteit Amsterdam, The Netherlands - {i.malavolta | p.lago}@vu.nl

ABSTRACT

Architectural Technical Debt (ATD) regards sub-optimal design decisions that bring short-term benefits to the cost of long-term gradual deterioration of the quality of the architecture of a software system. The identification of ATD strongly influences the technical and economic sustainability of software systems and is attracting growing interest in the scientific community. During the years several approaches for ATD identification have been conceived, each of them addressing ATD from different perspectives and with heterogeneous characteristics.

In this paper we apply the systematic mapping study methodology for identifying, classifying, and evaluating the state of the art on ATD identification from the following three perspectives: publication trends, characteristics, and potential for industrial adoption. Specifically, starting from a set of 509 potentially relevant studies, we systematically selected 47 primary studies and analyzed them according to a rigorously-defined classification framework.

The analysis of the obtained results supports both researchers and practitioners by providing (i) an assessment of current research trends and gaps in ATD identification, (ii) a solid foundation for understanding existing (and future) research on ATD identification, and (iii) a rigorous evaluation of its potential for industrial adoption.

KEYWORDS

Technical debt, Software architecture, Systematic mapping study

1 INTRODUCTION

Technical debt (TD) is a term first coined in 1992 by Cunningham to indicate immature portions of code that, while potentially working fine, might be unmanageable when they grow in quantity [11]. From its original description, i.e. "not quite right code which we postpone making it right", the TD metaphor has been extended to encompass different types of debt and shortcomings of software development processes [5].

Architectural Technical Debt (ATD) is a specific type of TD with an impact at the software architecture level. Given that software architecture plays a crucial role in the implementation of software systems [19], software architecture design can potentially lead to the introduction of TD that has a high impact on many portions of the system. Therefore, it is crucial to keep track of the ATD of software systems during their entire lifecycle. Failing to identify ATD may cause the slow deterioration of the software architecture, potentially leading to obsolete systems or even run-time failures.

In general terms, ATD is referred to sub-optimal decisions taken at the architectural level, which usually result in the conceivement of immature architectural artifacts [9]. Such ill-suited architectural decisions can be of different types, e.g., they can be visible or invisible [18], or can be introduced deliberately or inadvertently [12]. Instances of ATD, referred to as "ATD items" (ATDI) occur when a design or construction choice is taken at architectural level as an expedient in the short term, but set up a technical context that can make future changes more costly or impossible [6]. ATD identification refers to the activity, carried out during or after the architecting process, aimed at detecting ATDIs in software-intensive systems [21]. During the years, various researches have investigated how ATDIs can be identified. The resulting techniques are heterogeneous in nature, considering different types of ATD and distinct strategies to undercover them. While some studies have been conducted to provide an overview of the state of the art on ATD in general [9, 10], none of these focused specifically on ATD identification.

The goal of this study is to fill this gap by providing an evidence-based overview of the existing ATD identification research landscape. This study has been carried out by adopting a well-established methodology called systematic mapping [15, 29], and we applied it on peer reviewed papers focusing on ATD identification in software-intensive systems. Through our systematic mapping process, we selected 47 primary studies among 509 potentially relevant studies, fitting at best a set of rigorously-defined inclusion and exclusion criteria. Then, we created a dedicated classification framework composed of 13 different parameters for comparing techniques for ATD identification, and we applied it to all primary studies. We analyzed and discussed the obtained data under three complementary perspectives: publication trends, characteristics, and potential for industrial adoption.

The main contributions of this study are the following:

- an objective map of the state of the art in ATD identification;
- a rigorously-built classification framework for past, present, and future techniques for ATD identification;
- an evaluation of publication trends, characteristics, and potential for industrial adoption of existing research on ATD identification;
- a discussion of the emerging research trends, patterns, and gaps, and their implications for future research on ATD identification.
The audience of this study is composed of both (i) researchers willing to contribute to the area of ATD identification (ii) practitioners willing to understand existing research on ATD identification and adopt the most appropriate solutions for their technical, business, and organizational needs.

The remainder of this paper is organized as follows. Section 2 discusses related work and compares it to this study. Section 3 presents the design of this study from a methodological perspective, whereas Sections 4, 5, and 6 discuss obtained results. Finally, Section 7 describes threats to validity and Section 8 closes the paper.

2 RELATED WORK

A number of secondary studies focusing on TD exist to date. In Table 1 we give a schematic overview of those studies and in the following we discuss each of them.

Table 1: Secondary studies on TD

<table>
<thead>
<tr>
<th>Secondary study title</th>
<th>Year</th>
<th>Focus</th>
<th>tStudies</th>
<th>Time frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tom et al. [20]</td>
<td>2013</td>
<td>TD</td>
<td>10</td>
<td>N/A</td>
</tr>
<tr>
<td>Li et al. [20]</td>
<td>2015</td>
<td>TD</td>
<td>84</td>
<td>1992-2013</td>
</tr>
<tr>
<td>This study</td>
<td>2018</td>
<td>ATD identification</td>
<td>67</td>
<td>2009-2017</td>
</tr>
</tbody>
</table>

Most of the secondary studies on TD consider it only from a general perspective (i.e., they are not specific to architectural TD). For example, Li et al. [20] consider a set of 94 primary studies, with the goal to provide a comprehensive understanding of the notion of TD and the related research activities. This was achieved by thematically classifying the primary studies into 10 coarse-grained TD types, among which ATD results to be the second most studied subject (together with test TD and design TD), appearing in 25 different researches.

In a similar research carried out by Alves et al. [2], the TD literature was inspected to characterize the types of TD, their indicators, management strategies, maturity level, and possible visualization techniques. The authors analyzed 100 studies published between 2010 to 2014, which resulted in the conception of a preliminary taxonomy of TD types and an overview of the current research trends of TD. While, as reported also in the research by Li et al. [20], ATD resulted to be the second most frequent TD type, no in-depth analysis was reported for such topic.

In an earlier research, Tom et al. [26] present a literature review considering 35 primary studies. The focus in this case is to identify the nature of TD and its impact on software development activities. From the results, the authors derived a theoretical framework illustrating the dimensions of TD, attributes, precedents and outcomes. As compared to the previously-mentioned literature studies, the focus of this research is to gain understanding of the current research activities related to TD; accordingly, it does not concentrate on any specific aspect of TD.

In their systematic literature review, Ampatzoglou et al. [4] considered a more detailed overview of a specific TD topic, namely the economic implications of TD. In their work, the authors analyzed 69 primary studies to understand how financial aspects are defined in the TD context and how these are related to various aspects of software engineering. Their results show that financial approaches for TD management lack consistency in their applications, as the same approach is utilized differently in different studies. In [4], ATD is considered exclusively from a financial point of view.

The secondary study of Besker et al. [9] is the closest to ours by focusing exclusively on ATD-related literature. The authors inspect 26 primary studies to conceive a descriptive model aimed to provide a comprehensive interpretation of the ATD phenomenon. Their model identifies the main characteristics of ATD in four clusters: ATD Identification, ATD Checklist, ATD Impediments, and ATD Management. Each cluster is further decomposed into Focus areas (e.g. Relevance, Challenges, Methods/Tools), which are characterized by different aspects (e.g. Methods/tools is composed of Measuring, Tracking and Evaluating). This study was extended in a later publication [10] presenting a more comprehensive investigation of the literature and an in-depth analysis of the results. Our study differs from theirs by zooming into a specific challenge they identify, namely ATD identification.

In conclusion, by inspecting the TD secondary studies, we can observe that none of the studies aims directly at the characterization of existing approaches for ATD identification. We therefore conducted this research, focusing exclusively on the research landscape of ATD identification, in order to fill this gap and complement the existing literature.

3 STUDY DESIGN

In this section we report the study design that was strictly followed while planning and conducting research reported in this paper. The study design was conceived by following a set of well-established guidelines for software engineering literature studies [29].

3.1 Research Goal

The research was designed with the goal of characterizing comprehensively the current state of the art of ATD identification research. More specifically, by following the Goal-Question-Metric approach [8], our goal can be formalized as follows:

- **Purpose**: Identify, classify, and evaluate
- **Issue**: publication trends, characteristics, and potential for industrial adoption
- **Object**: of existing techniques for ATD identification
- **Viewpoint**: from the researcher’s and practitioner’s point of view

3.2 Research Questions

From our research goal, we can derive the following three research questions underlying our study:

**RQ1**: What are the publication trends about techniques for ATD identification? By answering this research question we aim to assess the ongoing trends of scientific interest on ATD identification techniques in terms of publication frequency, most prominent venues where academics are publishing their results on the topic and most recurrent venue types.

**RQ2**: What are the characteristics of existing techniques for ATD identification? By answering this research question we aim at providing (i) a solid framework for examining and classifying existing (and future) research on ADT identification techniques, and (ii) an understanding of current research trends and gaps in the state of the art of such techniques.
3.3 Search and selection

The search and selection process was designed as a multi-stage process, as depicted in Figure 1. This enabled us to rigorously control on the number and characteristics of the studies considered during the various stages. A description of each process followed is provided in the reminder of this section.

Figure 1: Overview of the search and selection process

3.3.1 Initial search. To identify the initial set of studies we performed an automatic search query on one of the largest and most complete scientific database and indexing system, namely Google Scholar. We selected such digital library for the following reasons: (i) it resulted to provide the highest number of potentially relevant studies compared to other four relevant libraries (Scopus, ACM Digital Library, IEEE Explore, and Web of Science), (ii) as reported in the set of guidelines by Wholin et al. [28], the adoption of such indexer results to constitute a sound choice to identify the initial set of literature for snowballing processes, (iii) the query results could be automatically extracted from the indexer. The research query utilized was conceived to encompass as much relevant studies as possible and is as follows:

Listing 1: Search query

1 TITLE: (architecture OR architectural OR architect OR architecting OR TD OR “technical debt” OR ATD) AND (architecture OR architectural OR architect OR architecting) AND (“technical debt”) OR

The query selects studies containing either a keyword referring to “architecture” or “technical debt” and related acronyms in their title (Listing 1, lines 1-2). Additionally, the full-text of the studies must contain both on of the keywords referring to “architecture” and the phrase “technical debt” (Listing 1, lines 3-4). The exclusive presence of related acronyms, i.e. “TD” and “ATD” in the full-text is not considered as a valid hit. The considered timeframe end was delimited by when the query was first executed (November 2017), in order to avoid potential discrepancies of results due to different query execution times. The start date was not set, in order to a void the introduction of a potential bias, even if it could have been set to when the term “technical debt” was first referenced [11].

3.3.2 Impurity removal. From the initial execution of the search query, a number of elements resulted not to be research papers (e.g. patents, standards etc.). Such occurrences were manually removed from the initial set of potentially relevant studies.

3.3.3 Application of selection criteria. Subsequent to the impurity removal process, we filtered all the remaining research papers according to a set of rigorously defined selection criteria. A research paper was included in the set of primary studies exclusively if it satisfied all of our inclusion criteria and none of the exclusion ones. Several exclusion rounds were adoptive by utilizing an adaptive reading depth [22], in order to thoroughly examining the literature in a time-efficient and objective manner. The inclusion and exclusion criteria utilized were:

1) Studies focusing on TD identification in software-intensive systems. This inclusion criterion is utilized to select exclusively studies considering TD.

2) Studies focusing on the architecture of software-intensive systems. This inclusion criterion is utilized to filter out studies considering other levels of abstraction, such as specific code implementation details.

3) Studies presenting or using a technique aimed to the identification of ATD in software-intensive systems. With this inclusion criteria, we ensure that only papers discussing the identification of ATD are included.

4) Secondary or tertiary studies (e.g., systematic literature reviews, surveys, etc.). This exclusion criterion is adopted in order to exclude studies which do not report the desired level of detail of ATD identification techniques.

5) Studies that have not been published in English language, as their analysis would result to be too time consuming.

6) Studies that have not been peer reviewed, in order to ensure the high quality of the studies considered.

7) Duplicate papers or extensions of already included papers, in order to avoid possible threats to conclusion validity.

8) Papers that are not available, as we cannot inspect them.

3.3.4 Snowballing. In order to mitigate potential biases w.r.t. the construct validity of this study, the automatic search was complemented by a snowballing process [13]. Specifically, a closed recursive backward and forward snowballing activity was conducted [28]. During this process, researches either citing and cited by the ones selected in the previous stages were examined, in order to enlarge the set of potentially relevant studies.

3.3.5 Validation through Scoups. In order to further mitigate the potential threats to validity resulting from the selection of a specific digital library, we conducted an supplementary search query execution on Scopus. We chose this additional database as it is defined as the largest abstract and citation database of peer-reviewed literature [1]. From the query execution, most of the paper indexed resulted to be either primary studies or papers excluded in the previous selection stages. Only 3 new papers were identified. After the application of the selection criteria, all were discarded.

3.4 Data Extraction

The purpose of this process was to (i) create a classification framework for the primary studies and (ii) extract the data from each
primary study. The classification framework consists of three parts, each addressing one of the research questions (see Section 3.2).

3.4.1 Publication Trends (RQ1). In order to assess the trends of ATD identification research, three attributes were considered, namely publication year, publication type and publication venue.

3.4.2 Research Focus (RQ2). To characterize the focus of the primary studies a systematic keyworing process [23] was adopted to define some of the parameters of our comparison framework. This process is constituted of two distinct steps: (i) collection of keyword and concepts, i.e. the identification of keywords and concepts by inspecting the full-text of all primary studies, and the subsequent combination of these to clearly identify the context, nature, and contribution of the research (ii) keyword clustering, i.e. the clustering the identified keywords and concepts into categories, in order to build up a classification framework. The output of this stage will be the classification framework containing all the identified parameters (each of them having a specific type and possible values), representing a specific aspect of ATD identification. The parameters considered in order to answer RQ2 were: architectural level (keyworded), ATDI definition (keyworded), analysis type (keyworded), analysis input source (keyworded), temporal dimension, ATD resolution, and tool support. For a definition of each attribute we refer to Section 5.

3.4.3 Potential for industrial adoption (RQ3). To assess the potential for industrial adoption, four distinct facets were considered, namely: tool availability, industry involvement, rigor and industrial relevance. The data of the last two attributes were collected by adopting an ad-hoc data extraction process [14].

3.5 Data Synthesis
Through a data synthesis process we aggregated and summarized the data extracted from the primary studies [16, §6.5] in order to understand, analyze, and classify the landscape of ATD research. In particular, we adopted content analysis (to categorize and code the primary studies in broad thematic categories) in combination with narrative synthesis (used to describe details and interpret the findings resulting from the content analysis).

3.6 Study Replicability
In order to provide the ability to fully replicate the research, a replication package of the study is publicly available. The package includes (i) the protocol describing comprehensively the study design details (omitted in the paper due to space limitations), (ii) a detailed description of all the parameters composing the classification framework, (iii) the raw data extracted in each phase (iv) the scripts utilized for the data processing, and (v) the list of primary studies.

4 RESULTS - PUBLICATION TRENDS (RQ1)
In the reminder of this section we report the results of the analysis of publication year, type, and venue of each primary study.

4.1 Publication year
Figure 2 shows the number of primary studies appearing each year, clustered by venue type. Primary studies span from 2009 to 2017, i.e., the year in which the search query was executed. While the term TD was first coined in 1992, we can observe that several years passed before the TD metaphor was explicitly considered in software architecture. Interestingly, the paper published in 2009 (P41) does not explicitly refer to “ATD” but considers “architectural bad smells” instead.

![Figure 2: Primary studies publication year and venue type](image)

In general, a growing trend can be identified through the years, demonstrating the recent interest of researchers and practitioners in the subject. As an outlier, a drop of number of primary studies can be noticed for the year 2017. Such occurrence should be attributed to the fact that the search query used to select the primary studies was executed before the end of 2017, leading to partial results for this year. The reported publication trend is confirmed also in the recently published study by Besker et al. [10], focusing on the main ATD characteristics and relations among them.

4.2 Publication types
From the distribution depicted in Figure 2 we observe that the majority of primary studies is published in conferences (28/47), followed by a non-negligible number of workshops (12/47). Only a modest number of studies (7/47) is published in journals. We can conjecture that such trend is due to the relatively recent interest in ATD identification by the software engineering research community. We can expect a growth in the number of more scientifically-rewarding publication types (like journals) in the future. This conjecture is partially confirmed by the 3 journal publications only in 2017.

4.3 Publication Venues
Table 2 reports on the number of studies appearing in the most recurrent venues. There we can observe that ICSE is the most frequent venue (9/47) followed by MTD (6/47) and ICSA/WICSAs. The presence of MTD as second most recurrent venue highlights the importance that such workshop has in the TD research area. In general, due to the nature of the most recurrent venues, we can conjecture that the primary studies are characterized by being of high quality and have potentially high interest and resonance in the scientific community.
We uncovered four distinct and incremental levels of abstraction while source code classes (18/47) have been published in different venues, spanning different research areas like software maintenance and evolution (P14), software architecture erosion and architectural consistency (P3), and dependency and structure modeling (P27). This might indicate that the TD research community is still undergoing a consolidation phase.

Main findings (RQ1). ATD identification is attracting a growing scientific interest in the last years. The research landscape is quite fragmented, with ICSE, MTD, and ICSA as most targeted venues. So far, researchers mostly targeted conferences and workshops, but it is expected that journal publications will raise in the coming years.

5 RESULTS - RESEARCH FOCUS (RQ2)

Here we report the characteristics of ATD identification techniques as they emerged from our keywording process (see Section 3.4).

5.1 Level of abstraction

We uncovered four distinct and incremental levels of abstraction during our keywording process, namely: Source Code Classes, Source Code Files, Source Code Packages, and Components and Connectors.

As shown in Table 3, source code packages, defined as related files implementing the same functionality, are the most recurrent building blocks of the considered architectures (13/47). Components and Connectors are adopted by a similar number of studies (12/47), while Source Code Classes and Source Code Files are used in a lower number of studies.

The obtained results suggest that as of today there is no common agreement in the literature about which level of abstraction should be considered when dealing with ATD identification. We can conjecture that this phenomenon may be a consequence of the fact that (i) a unique and well-accepted definition of software architecture is also missing in the state of the art and (ii) the level of abstraction in the proposed techniques is strongly influenced by the types of ATD sources, such as system source code, its architecture documentation, etc. (see Section 5.4 for the details on ATD sources).

5.2 ATDI Definition

The keywording process resulted in four recurrent categories of ATDI (see Table 4): Dependency Violations among architectural components (27/47), Non-modularity (26/47), Compliance Violations (18/47) and Change Proneness (9/47). Dependency Violations describe architectural violations resulting from unfit dependencies among architectural components. Such type of ATDI is usually caused by unsound architectural design choices, incorrect implementation, or architectural deterioration. Non-modularity refers to the sub-optimal modularization of architectural components. Lack of modularity often causes small changes to propagate to other portions of a system, lowering the maintainability and evolvability of software systems. Compliance violations refer to the deviation w.r.t. a certain architectural pattern (e.g. model-view-controller) affecting the quality of the system. Change Proneness, instead, refers to architectural components that are modified with high frequency.

Table 4: ATDI definition

<table>
<thead>
<tr>
<th>ATDI keywording</th>
<th>#Studies</th>
<th>Studies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change Proneness</td>
<td>9</td>
<td>P1, P9, P14, P19, P20, P29, P32, P33, P40, P41, P42, P43, P44</td>
</tr>
<tr>
<td>Custom</td>
<td>18</td>
<td>P5, P8, P11, P16, P18, P19, P20, P22, P25, P28, P29, P30, P31, P32, P33, P34, P35, P41, P42, P44</td>
</tr>
</tbody>
</table>

In 25 primary studies more than one type of ATDI is used. In most of the cases this is due to the definition of more than one ATDI in the paper. For example, in P19 five different ATDIs related to change proneness, dependency violations, and non-modularity of source code files are defined. Additionally, a considerable number of papers (18/47) is based on a custom definition of ATDI that could not be mapped onto any specific category. Such occurrences focus on an ad-hoc definitions of ATD, e.g. lack of reusability (e.g., P9) or non-uniformity of package name patterns (e.g., P39).

From the high heterogeneity of the gathered data we can conclude that there is no common agreement on what defines an ATDI. We can hence conclude that, while different types of ATDI are required in order to comprehensively model ATD, the literature is still lacking in a comprehensive taxonomy for a sound classification of ATDIs.

5.3 Analysis Type

In Table 5 the most recurrent analysis types are reported. From the extracted data we can observe that a large number of analysis types are performed in the literature.

Table 5: Analysis Type

<table>
<thead>
<tr>
<th>Analysis Type</th>
<th>#Studies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static</td>
<td>33</td>
</tr>
<tr>
<td>Dynamic</td>
<td>32</td>
</tr>
<tr>
<td>Safety</td>
<td>22</td>
</tr>
<tr>
<td>Performance</td>
<td>18</td>
</tr>
<tr>
<td>Security</td>
<td>16</td>
</tr>
<tr>
<td>Reliability</td>
<td>10</td>
</tr>
<tr>
<td>Testability</td>
<td>5</td>
</tr>
</tbody>
</table>

A considerable number of primary studies (11/47) does not report explicitly the considered abstraction level. Those primary studies commonly rely on human knowledge (P6, P8, P12, P13, P29), self-admitted ATDI (P2, P16), and tools in which the architecture definition is implicit (P17, P35, P37, P43).
The majority of the ATD identification techniques are based on the identification of architectural antipatterns and smells among architectural components. Examples of identified architectural antipatterns include cyclic dependencies between architectural components and unstable Interfaces. In particular, such techniques usually entail (i) the identification of architectural components (usually achieved by clustering source code artifacts) and, (ii) the assessment of properties of/among the components, usually carried out through modularity analyses (19/47), dependency analyses (15/47), or a combination of the two.

Modularity analysis consists in assessing if system functionalities are separated into independent, self-contained modules [7]. Modularity resulted the second most recurrent type of analysis (19/47). Such approaches commonly entail the identification of architectural components and functional requirements of a software system, the evaluation of the modularization of components, and the eventual cost analysis of the rework cost required to carry out a refactoring process.

Dependency analysis is based on the evaluation of dependencies between architectural components in order to identify irregularities (e.g., circular dependencies). The use of Design Structure Matrices (DSM) for dependency analysis results to be quite widespread for ATD identification (P1, P10, P15, P19, P20, P22, P27, P30, P32, P33, P34, P35, P37, P38, P40, P42, P46).

In order to discover and understand ATD related phenomena, many studies rely on analyses of the evolution of software systems through time (16/47). Such approaches take as input historical data such as architectural documentation and version history. We observe that our primary studies adopt a heterogeneous set of approaches to identify ATD. For example, in P1 a technique based on locating co-changing files in order to identify "architectural roots" of systems is presented, while in P5 relates modularity metrics to the average number of modified components per commit. From the variety of approaches utilizing evolution analysis we can evidence the high potential that historical data has for ATD identification.

A relatively high number of studies considers the financial aspect of managing ATD through cost analyses (14/47). Such studies range from risk analyses (e.g., P34) to methods aiding decisions on if and when ATD should be refactored (e.g., P6). The presence of numerous studies focusing on cost implications of ATD is a meaningful indicator. In fact this demonstrates how the ATD problem is rooted in practice and furthermore shows that the importance of effectively understanding ATD issues and planning future maintenance activities is not neglected in current research activities.

Human knowledge based analyses are frequently adopted in the primary studies (10/47). Such types of analyses are conceived to identify ATDIs in systems by extracting human knowledge through structured- or unstructured processes such as surveys, interviews, and questionnaires. We can conjecture that the reason of the high adoption of human knowledge based analyses is twofold: (i) some types of ATDIs need insights that are not documented (e.g., rationale behind a design decision, P11), and (ii) human knowledge provides validation to complement (semi-)automated ATD identification analyses (e.g., gathering feedback by presenting to stakeholders a visualization of the automatically identified ATDIs, P21).

Analyses based on compliance checking assess the discrepancy between intended and implemented architecture, and the deterioration of the architecture in time [17]. In most of the cases such approaches are tool supported (P3, P7, P43, P47). One of the studies (P5) relies on an uncommon technique to discover architectural compliance in software systems, namely a genetic algorithm.

Change impact analysis is about the evaluation of different design alternatives and/or the calculation of the effort required to resolve or avoid ATD in the future [21]. In many cases this typology of studies involves also cost analyses (P10, P21, P22). Approaches considering different alternative results are not common in our dataset. This result shows that this is due to the fact that only few papers consider ATD resolution, as reported in Section 5.6. Few studies report analyses based on the object-oriented (OO) paradigm. In most of the cases such approaches adopt consolidated software metrics and transfer the gathered results to the architectural level. For example, in P26 architecturally relevant classes are identified through code smells and the change history of the classes, while in P44 architecture quality is assessed by analyzing code smells among related OO classes. In general, the presence of such studies indicates that a portion of ATD analyses is rooted in code analysis, highlighting the thin line that separates software architecture and the source code of the system in this research area.

Only few studies focus on ATD visualization techniques. This observation is confirmed by Alves et al. [3] in their secondary study on TD. This result shows that this research area is only marginally considered by current research activities and requires to be further explored, especially considering its potential efficacy in the communication of ATD issues.

A minority of analyses involve a manual classification processes. For example P40 presents a visualization approach to highlight ATD-prone dependencies, based on which architects can manually select the most significant to be considered for refactoring. We can conjecture that the low presence of such type of analysis is due to the high effort required to carry out manual processes and their relative low scalability, which makes difficult their application to large software systems.

Only two studies focusing on self-admitted ATD were found in the primary studies. Such studies rely on code and commit comments where developers "self-admittedly" identify ATDIs. Such technique is commonly used to identify code related TD [24].

---

Table 5: Analysis type

<table>
<thead>
<tr>
<th>Analysis keywording</th>
<th>#Studies</th>
<th>Studies</th>
</tr>
</thead>
</table>

---

We are aware that modularity and dependency analyses can be considered as particular types of architectural antipattern. Given their high frequency in the primary studies, we considered them as a separate categories.
low number of studies focusing on self-admitted ATD might indicate that it is still emerging and under-explored, or simply that architectural aspects of software systems are not frequently discussed in repositories (e.g. in commit messages, issue trackers).

5.4 Analysis Input

Our analysis identified five categories of ATD analysis inputs, namely: Source Code, Evolutionary Data, Architectural Documentation, Survey and Issue Tracker. As shown in Table 6, Source Code is the most recurrent analysis input (32/47). From this result we can observe that the majority of ATD identification techniques relies to a certain extent on the inspection of code-related properties. A smaller number of studies requires Evolutionary Data (16/47), such as commit history or measurements taken over time. Interestingly, two papers that utilize evolutionary data do not require source code as additional source, but rely on data extracted from human knowledge (P6), or use pre-existing architectural documentation (P10).

Table 6: Analysis input source

<table>
<thead>
<tr>
<th>Input source</th>
<th>#Studies</th>
<th>Studies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evolutionary Data</td>
<td>26</td>
<td>P1, P4, P5, P9, P10, P15, P16, P19, P20, P25, P26, P32, P34, P38, P42, P43</td>
</tr>
<tr>
<td>Human knowledge</td>
<td>10</td>
<td>P6, P8, P11, P12, P13, P18, P21, P23, P29, P36</td>
</tr>
<tr>
<td>Issue Tracker</td>
<td>9</td>
<td>P1, P15, P19, P20, P23, P25, P32, P35, P42</td>
</tr>
</tbody>
</table>

A lower number of studies takes Architectural Documentation as input (11/47). We conjecture that this is due to the difficulty to get access to industrial documentation, which is most often out of date, when available [25]. In line with the results reported in Section 5.3, 10 studies report approaches that require human knowledge as input source. Finally, a minority of studies takes Issue Trackers as additional input to source code (7/47). This latter typology of studies associate ATD to bug-related issues in source code. For example, in P20 the number of architectural flaws in a file is correlated to number of bugs in it, its change frequency, and the total amount of effort spent on it.

5.5 Temporal Dimension

Table 7 provides an overview of the papers taking into account the temporal dimension. In order to identify ATD, almost half of the papers considers the evolution of software systems through time (22/47). Here we can observe a discrepancy between the studies considering software evolution and the ones adopting Evolutionary Data as input (see Table 6). This difference is due to the studies relying on human knowledge extraction. In fact, those studies rely on human knowledge instead of evolutionary software artifacts in order to reason about the evolution of software systems.

Surprisingly, from the gathered data we can observe that the majority of the studies does not consider the temporal dimension. Nevertheless we have to remark that this aspect is analysis-specific, and hence not required per se in order to identify ATD in a system. For example, P2 considers self-admitted TD in source comments of a single software release, and hence can pinpoint ATD in source code without inspecting the entire version history of a system.

5.6 ATD Resolution

ATD resolution refers to refactoring strategies aimed to partially or completely remove identified ATDs. Table 8 gives an overview of which studies consider ATD resolution. From the results we can observe that only a limited number of studies reports ATD resolution strategies (15/47). This could be a symptom of the relative young age of the ATD research field, where most of the research effort is still devoted to processes aimed at understanding the ATD phenomenon, rather than at resolving the identified ATD.

Table 8: ATD resolution

<table>
<thead>
<tr>
<th>ATD resolution</th>
<th>#Studies</th>
<th>Studies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not considered</td>
<td>42</td>
<td>P2, P3, P7, P8, P9, P14, P15, P16, P17, P21, P22, P24, P28, P30, P32, P33, P34, P35, P37, P38, P40, P41, P42, P43, P44, P45, P46, P47</td>
</tr>
<tr>
<td>Considered</td>
<td>22</td>
<td>P1, P4, P5, P6, P10, P12, P13, P15, P18, P19, P20, P23, P25, P26, P27, P29, P32, P34, P36, P38, P42, P43</td>
</tr>
</tbody>
</table>

5.7 Tool Support

Finally, from the studies we extracted which tool were utilized in order to carry out the ATDI identification processes. This attribute is meaningful both for (i) researchers who want to conceive new tool-based ATD identification techniques and (ii) practitioners needing tools to get further insights in their projects. A comprehensive list of the most recurrent tools is reported in Table 9.

Table 9: Tool supported

<table>
<thead>
<tr>
<th>Tool supported</th>
<th>#Studies</th>
<th>Studies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Titan</td>
<td>10</td>
<td>P7, P18, P36, P47</td>
</tr>
<tr>
<td>Structure101</td>
<td>6</td>
<td>P1, P2, P4, P7, P10, P13, P14, P16, P17, P19, P20, P21</td>
</tr>
<tr>
<td>SonarQube</td>
<td>6</td>
<td>P8, P10, P12, P13, P18, P21, P23, P25</td>
</tr>
<tr>
<td>Arcan</td>
<td>8</td>
<td>P15, P19, P20, P23, P25, P32, P35, P42</td>
</tr>
<tr>
<td>CAST</td>
<td>10</td>
<td>P1, P7, P13, P15, P16, P17, P20, P23, P25, P26, P30</td>
</tr>
<tr>
<td>AMARIS</td>
<td>5</td>
<td>P4, P7, P18, P36, P47</td>
</tr>
<tr>
<td>CBLO</td>
<td>6</td>
<td>P4, P7, P10, P12, P13, P18, P21, P23, P25</td>
</tr>
<tr>
<td>Call Graph Extractor</td>
<td>5</td>
<td>P3, P4, P7, P10, P12, P13, P18, P21, P23, P25</td>
</tr>
<tr>
<td>Hotspot Detector</td>
<td>5</td>
<td>P3, P4, P7, P10, P12, P13, P18, P21, P23, P25</td>
</tr>
<tr>
<td>JHarc</td>
<td>5</td>
<td>P3, P4, P7, P10, P12, P13, P18, P21, P23, P25</td>
</tr>
<tr>
<td>SAJ</td>
<td>5</td>
<td>P3, P4, P7, P10, P12, P13, P18, P21, P23, P25</td>
</tr>
<tr>
<td>JPlasma</td>
<td>5</td>
<td>P3, P4, P7, P10, P12, P13, P18, P21, P23, P25</td>
</tr>
</tbody>
</table>

Titan results to be the most used tool in our primary studies. Conceived by Xiao et al. [30], Titan introduces a new architecture model referred to as “design rule space”, intended to capture both the architecture and the evolutionary structure of systems in order
Studies
35
P1, P2, P4, P6, P8, P10, P11, P12, P13, P15, P16,
As shown in Table 10, only a small number of studies is implemented
ATD identification approaches that require tool support. Neverthe-
the availability of a tool that implements a proposed ATD identifi-
cation approach is required in order to enable the efficient adoption
the studies for industrial adoption.

6 RESULTS - POTENTIAL FOR INDUSTRIAL ADOPTION (RQ3)
In this section we report on the results regarding the potential of
the studies for industrial adoption.

6.1 Tool Availability
The availability of a tool that implements a proposed ATD identifi-
cation approach is required in order to enable the efficient adoption
of such approach in industry.

Table 10: Tool availability

<table>
<thead>
<tr>
<th>Tool availability</th>
<th>#Studies</th>
<th>Studies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Available</td>
<td>32</td>
<td></td>
</tr>
</tbody>
</table>

As shown in Table 10, only a small number of studies is implemented
in an available tool. In particular, most of such studies makes use of
a novel tool created ad-hoc for the ATD identification described in
the paper or integrates tools that were already developed by the
authors. For example, in P5 two publicly available tools developed by
the authors are utilized: one to calculate modularity metrics (Mod-
ularityCalculator) and the other to calculate the average number of
modified components per commit (CommitAnalyzer).

From these results we evince that, while some approaches are
available in the form of a tool, this is not true for most of the studies
(35/47). This might indicate that (i) numerous researches provide
theoretical results and proof of concepts, and/or (ii) more effort is
needed to ease the application of ATD identification approaches.

6.2 Industry Involvement
In order to assess the involvement of industry in the research related
to ATD identification, we categorize the primary studies into three
partially overlapping categories, namely: academic, industrial and
mixed. A study is classified as academic if all authors are affiliated
to universities or research institutes, industrial if all the authors
are affiliated to industrial companies and, mixed if co-authors are
from both academia and industry. The distribution of the primary
studies according to this classification is reported in Figure 3.

![Figure 3: Distribution of industry involvement](image)

As shown in Figure 3 the majority of the researches were con-
ducted from an academic-only perspective (38/47), some studies
emerged from a mixed perspective (7/47) and industry-only studies
are rare (2/47). From these results we can observe that the research
topic is still addressed mostly from an academic perspective; more
partnerships between industry and academia are necessary in order
to enable beneficial knowledge exchanges and acquire a more com-
prehensive understanding of the problems and applicable solutions.

6.3 Rigor and Industrial Relevance
To gain further insights into the potential for industrial adoption
of the ATDI identification approaches, we evaluated the rigor and
industrial relevance of the studies. This process was carried out by
applying the well-defined classification model introduced by Ivars-
son et al. [14]. Accordingly, Rigor refers to the accuracy or exactness
of the research method used, and is subdivided into three categories:
Context, Study design and Validity. These categories assume values
“0”, “0.5”, and “1” reflecting the quality of their description. Industrial
relevance, instead, considers experimental Subjects, Context, Scale,
and Method, which assume values “0” or “1”.

6.3.1 Rigor. Figure 4 shows the distribution of the primary stud-
ies among the three rigor categories. We observe that the context
considered in the studies is in most of the cases reported but de-
scribed schematically (24/47). This indicates that potential imped-
iments could be encountered when contexts different from the ones
reported in the studies are considered. The study design is gener-
ally characterized by a medium or strong description (19/47 and

1http://structure101.com
2http://www.hello2morrow.com/products/sonargraph
23/47, respectively), enabling the reader to clearly understand the variables considered, the treatments adopted, etc. Concerning validity, a relatively low number of studies discusses threats to validity (10/47). In most of the studies validity is only mentioned (19/47) or fully neglected (18/47). This suggests that more effort should be spent in documenting the validity of the researched approaches, necessary to increase their potential for industrial adoption.

6.3.2 Industrial relevance. As shown in Figure 5, most of the studies consider representative industrial subjects (33/47) and utilize evaluation methods that could be efficiently applied in an industrial setting (37/47). From the data we can observe that, the fact that ATD is a problem rooted in practice is not neglected by researchers. In fact, the studies present approaches that can be easily applied in practice as well as industrial case studies utilized for their evaluation. Naturally, the size of the systems considered is in the majority of the cases of industrial scale, too (35/47).

Figure 6: Rigor and relevance of primary studies

On a less positive note, the context is not always representative of the intended usage of the researched approach, as evaluations are often preformed in an academic setting instead of an industrial one (19/47). This is reflected also in the high number of academic-only authors reported in Section 6.2.

6.3.3 Combined analysis of rigor and industrial relevance. By jointly considering the rigor and industrial relevance distributions of primary studies, we can sketch which future steps should be taken in order to increase the potential for industrial adoption of the research results in ATD identification. As illustrated in Figure 6, the practical roots of ATD research seem to deeply influence the industrial relevance of the primary studies, the vast majority of which has high cumulative scores for such attribute. A higher variability and lower scores can instead be noticed if rigor is considered. We can hence conclude that in future research more effort should be put into rigorously describing the context and the threats to validity, in order to increase rigor quality.

7 THREATS TO VALIDITY

In this section we discuss the threats to validity of our research. In general, in order to ensure the high quality of the data extracted, a well-defined research protocol was established before carrying out the data collection. In addition, all research activities were designed and carried out by adhering to a set of well-accepted guidelines for systematic mapping studies [15, 23, 29]. By formalizing such guidelines we established the protocol that was strictly followed throughout the study, as documented in Section 3 and further detailed in the provided replication package. In addition, in order to lower potential sources of bias, crucial considerations that emerged during the research were discussed jointly by all the authors. Despite adhering to a systematic literature review approach, potential threats to validity are still unavoidable, as discussed in the reminder of this section along with how we mitigated them.

External validity. The most significant threat to external validity consists of the potentially low representativeness of the primary studies. In order to mitigate this threat, we adopted for the identification of potentially relevant studies the most encompassing digital library\(^6\) (Google Scholar) and search query. Another threat to external validity is the adoption of a specific set of query keywords. To mitigate this threat the results of the automated search query were further extended by executing a backward and forward snowballing process. In order to have control over the quality of the

\(^6\)Selected after a preliminary execution of the search query on: Google Scholar, Scopus, IEEE Explore, ACM Digital Library, and Web of Science.
primary studies, we exclusively considered peer-reviewed papers, and hence excluded "grey literature", e.g. white papers, editorials, etc. We deem that this does not constitute an additional threat, as peer-review processes are a standard requirement of high quality publications. Finally, we utilized a set of well-defined inclusion and exclusion criteria, which rigorously guided our manual selection of the literature.

**Internal validity.** To mitigate potential threats to internal validity, we established a priori a rigorous research protocol that guided all the research activities. In addition, the classification framework utilized was defined iteratively by strictly adhering to the keywording process [23]. For the synthesis of the collected data, simple and well-established descriptive statistics were adopted. In addition, sanity tests on the extracted data were used by cross-analyzing different parameters of the established classification framework.

**Construct validity.** In order to ensure that the primary studies were suited to answer our research questions, we manually carried out the selection of primary studies according to a predefined set of well-documented inclusion and exclusion criteria. The results of such process were further expanded by conducting an additional iterative backward and forward snowballing process. In addition, as advised by Wholin et al. [29], a random sample of 10 studies were selected and analyzed independently by all 3 researchers in order to guarantee the alignment of the analyses.

**Conclusion validity.** Potential sources of bias resulting from the data extraction and analysis processes were mitigated by strictly adhering to an a priori defined protocol, explicitly conceived to gather the data required to answer our research questions. To further mitigate threats to conclusion validity, best practices from several well known guidelines for systematic literature reviews [15, 23, 29] were followed. These guidelines were strictly adhered to throughout the entirety of our research activities, and are thoroughly documented to ensure that our research approach is transparent and replicable.

### 8 CONCLUSIONS

This paper presents a systematic mapping study on ATD identification, the first and foremost building block of ATD management. Starting from 509 potentially relevant studies, we rigorously analyzed 47 primary studies via a classification framework dedicated to ATD identification. Our analysis provides a characterization for ATD identification techniques in terms of publication trends, their characteristics, and their potential for industrial adoption.

Our analysis also unveils a series of promising trajectories for future research on ATD, such as (i) the exploitation of the temporal dimension when identifying ATD, (ii) the related resolution of identified ATD, and (iii) a further industrial involvement when formulating, designing, and evaluating the ATD identification techniques.

In our future work, we are exploring various approaches to identify ATD in software-intensive systems. Among them, we are considering architecture compliance checking, e.g. in the context of Android applications [27].

### REFERENCES


