EasyChair Preprint

№ 2741

# The Evolution of Architectural Decision Making as a Key Focus Area of Software Architecture Research: a Semi-Systematic Literature Study

Manoj Bhat, Klym Shumaiev, Uwe Hohenstein, Andreas Biesdorf and Florian Matthes



February 21, 2020

# The evolution of architectural decision making as a key focus area of software architecture research: A semi-systematic literature study

Manoj Bhat
*Corporate Technology*
*Siemens AG*
81739 München, Germany
manoj.mahabaleshwar@siemens.com

Klym Shumaiev
*Corporate Technology*
*Siemens AG*
81739 München, Germany
klym.shumaiev@siemens.com

Uwe Hohenstein
*Corporate Technology*
*Siemens AG*
81739 München, Germany
uwe.hohenstein@siemens.com

Andreas Biesdorf
*Corporate Technology*
*Siemens AG*
81739 München, Germany
andreas.biesdorf@siemens.com

Florian Matthes
*sebis*
*Technische Universität München*
85748 Garching, Germany
matthes@tum.de

***Abstract*—Literature review studies are essential and form the foundation for any type of research. They serve as the point of departure for those seeking to understand a research topic, as well as, helps research communities to reflect on the ideas, fundamentals, and approaches that have emerged, been acknowledged, and formed the state-of-the-art. In this paper, we present a semi-systematic literature review of 218 papers published over the last four decades that have contributed to a better understanding of architectural design decisions (ADDs). These publications cover various related topics including tool support for managing ADDs, human aspects in architectural decision making (ADM), and group decision making. The results of this paper should be treated as a getting-started guide for researchers who are entering the investigation phase of research on ADM. In this paper, the readers will find a brief description of the contributions made by the established research community over the years. Based on those insights, we recommend our readers to explore the publications and the topics in depth.**

***Index Terms*—software architecture, decision making, architectural design decisions, literature review**

## I. Introduction

Over the last years, the topic of architectural decision making (ADM) keeps receiving steady attention in the software architecture research community. Only from 2014 to 2018, we observe at least 14 publications every year being published at relevant scientific conferences and journals. The last extensive systematic mapping study on architectural design decisions (ADDs) by Tofan et al. [1] was published in 2014 which discussed publications from 2005 to 2011. This paper had a significant impact on the research community; being cited 59 times since then, it lays forth a research roadmap presenting future research topics for researchers and practitioners.

In our work, 6 years later, we revisit four decades of ADM research and summarize those publications that made ADM an established research area. We do it to support researchers and practitioners to navigate the growing body of knowledge, as well as, for them to reflect on the presented concepts and ideas related to ADDs, tool support for managing ADDs, ADM concepts, optimization-based ADM, and group decision making (GDM). We refer to our work presented in this paper as a getting-started guide to explore the existing work on ADM. We recommend our readers to pick their topic of interest within this paper, browse through the description of the papers corresponding to that topic, and read the selected papers to the full merit of those papers' scientific contributions. Furthermore, we strongly suggest our readers to refer to the recently published literature studies by Tang et al. on ADM [2], [3]. These publications not only provide future research directions but also share guidelines on how the future research in ADM should be conducted. In the subsequent sections, we present the literature review protocol used to identify the publications and then discuss the identified publications.

## II. Research methodology

Our literature review protocol adheres to the guidelines prescribed by Kitchenham and Charters [4]: (a) For the identification of research, we defined the inclusion and exclusion criteria. (b) For the selection of relevant publications, we described a search strategy. (c) Data extraction was performed based on the inclusion and exclusion criteria. (d) Finally, the collected publications were analyzed and summarized. We did not conduct (a) commissioning the review, (b) evaluating the protocol, and (c) evaluating the review report and these steps are also not considered mandatory in [4]; hence, we refer to this review as a semi-systematic literature review.

### A. Inclusion and exclusion criteria

For the inclusion criteria, we considered those papers that: (a) refer to software architecture or design and (b) discuss ADDs as the main topic (this could include architectural knowledge management (AKM), ADD models, and ADM. Since the search query was formulated using English terms, we restricted to those studies published in English. Furthermore, to retrieve all relevant studies, no lower boundary was set on the publication date. However, since the search queries were executed on 17.06.2019, that is considered as the upper boundary. The following list of exclusion criteria were considered: (a) publications describing conference/keynote, (b) publications discussing specific system's architecture (e.g., architecture of a decision support system), and (c) publications on specialized topics (e.g., requirements negotiation, modeling service-oriented process decisions, pattern-based approaches).

### B. Search strategy

We considered four digital libraries: ACM, IEEE Xplore, ScienceDirect, and Springer Link. These libraries include proceedings of the major software architecture conferences such as ICSE, WICSA, ICSA, ECSA, SHARK, MARCH and QoSA. To ensure the retrieval of all relevant publications, we formulated a broad search query based on the inclusion criteria. First, we wanted to include publications related to "software architecture" or "software design" (Q1). Second, we considered those publications that discuss "decision making" or "design decisions" (Q2). Given that both Q1 and Q2 had to be satisfied, they were combined using an AND clause. It should be noted that, typically, the keywords used in the search strategy are derived from the study's research questions. However, in our case, we conducted this study to position our past five years of work on managing ADDs and supporting ADM, which is published in a dissertation report [5].

The final search query was matched against publications' title, abstract, and keywords. Only for Springer Link, we could not match the query against the three publication fields. Instead, we had to perform a full-text search that resulted in a larger number (1,471 publications) of retrieved publications.

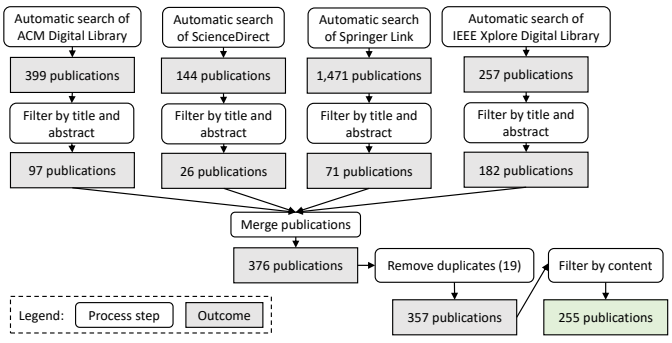


Fig. 1. Search process that resulted in 255 publications for this study

### C. Data extraction and synthesis

On executing the search query, we found 399, 144, 1,471, and 257 publications in ACM, ScienceDirect, Springer Link, and IEEE Xplore respectively. Next, each publication was filtered by reading the title and abstract according to the inclusion criteria. This filtering step resulted in 97, 26, 71, and 182 publications from ACM, ScienceDirect, Springer Link, and IEEE Xplore respectively. These 376 publications were merged into a new CSV file. Out of these 376 publications, we found 19 duplicate publications and were removed. For 357 publications, their full-text was thoroughly read by the first and second authors of this paper. During this step, those publications that were marked as deleted by both the reviewers as per the exclusion criteria were removed. As shown in Figure 1, the final step resulted in a total of 255 publications and were summarized.

### D. Overview of review results

Figure 2 shows the distribution of 255 publications according to their publication year. The first publication about design decisions was in 1980. Note that, in Figure 2, we see a steep increase in publications per year after 2004 (paradigm shift in software architecture [6]). Since 2004, researchers started to look at architecture not only as a composition of components and their relationships but instead began to reflect on how those components and relations come into existence. There has been a plethora of publications addressing various aspects of ADDs since 2004, e.g., models and tools for managing ADDs, ADM strategies, and factors influencing ADM. Today, we observe at least one research track dedicated to ADDs and ADM in architecture conferences such as ECSA and ICSA.

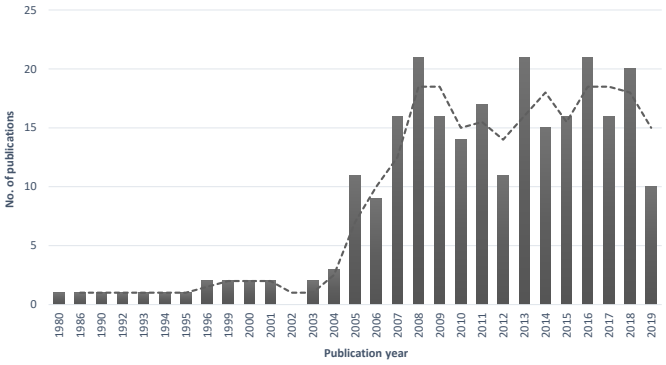


Fig. 2. Publications related to AKM, ADDs, and ADM over the past years

To analyze the identified 255 publications, we categorized them into: Architecture Knowledge Management (AKM), ADDs, formal ADD models, ADD tools, Architectural Design Rationale (ADR), ADM, and Group Decision Making (GDM) topics. One could argue that many of the publications cover more than one topic, GDM is part of ADM or managing ADDs is part of AKM. However, it should be noted that this categorization scheme was used only for the convenience of summarizing the results depending on the context and publication's theme. For instance, if a publication presented

an ADM strategy but specifically emphasized GDM, it was sorted into the GDM category. In this paper, especially due to space constraints, we do not discuss 37 publications that emphasize AKM and ADR and focus on the remaining 218 papers. However, the summary of those 37 papers are made available in a report [5]. Two papers in the references section, namely [5] and [4] are not part of our review. The replication package for the above review steps are available online[1].

## III. A SUMMARY OF IDENTIFIED PUBLICATIONS

In this section, we summarize the publications related to ADDs, formal models for ADDs, ADD tools, ADM, and finally, GDM. In each category, we have tried to discuss the publications in a chronological order. However, in some situations, we have also grouped them to fit the context.

### A. Architectural design decisions

The very first mention of design decisions was in 1980. During that time, software architecture, as a field of study, had not yet been well established, and the reference to design decisions was in the programming context. In [7], the author captured design decisions (e.g., replace recursion by iteration) using an applicability predicate (antecedent) and a predicate expressing the desired qualitative effect (consequent). Next, in 1990, the authors of [8] argued that design decisions could be *detected in the source code* using reverse engineering. Also, in the early 90s, when building software through model transformations was becoming popular, the authors of [9] proposed to map these models through design decisions. They presented the first tool (traceability support system) that allowed programmers to capture the problem, alternatives, decisions, rationale, and links between the source and the target models.

The authors of [10] used the Issue Based Information System (IBIS) for capturing design argumentation. They introduced a model to help developers answer questions such as "*Why certain decisions were made?*", "*What alternatives have been explored?*", and "*Will a change violate any constraint?*".

Kruchten in his seminal work [11] stated that "architecture documentation captures design decisions that must be respected to maintain the architectural integrity". To document those decisions, the authors of [12] proposed to use a tree structure; wherein, each node contains "concerns and constraints, decisions, new concerns raised by a decision, and references to other decisions". Kruchten highlighted that architects must be able to communicate ADDs to stakeholders and should consider others' decisions [13]. He further went on to say that an architect must "*make some decisions very quickly, based on experience and gut feelings rather than pure, thorough analysis*". [14] also suggests that the soft-skills and external influences are important in the socialization of decisions. Kruchten et al. argued that since the effort in capturing decisions outweighs the immediate benefits, we should *automate the collection of decisions and their rationale* [15].

Quality concerns are the drivers for ADDs [16], [17]. Kazman presented the Cost Benefit Analysis Method (CBAM) to model the costs and the benefits of ADDs. CBAM was further revised to incorporate an iterative process with a decision analysis framework [18]. [19] also investigates the economic perspective of ADDs and presents a "structured intuitive model for product line economics". Furthermore, to compute the savings in effort, [20] proposes a cost-benefit framework for ADM. By analyzing the differences in an architectural refactoring activity, this framework correlates the developer effort to the change in coupling.

The very first explicit mention of the statement "*architecture is a collection of design decisions that are expensive to change*" was in 2001 [21]. Alexander Ran argued that the "expensive to change" decisions are those decisions on which most other decisions depend. Jan Bosch, in 2004, also emphasized that "*architecture is a composition of ADDs*" and "*designing a system can be viewed as a decision process*" [22]. Jansen and Bosch pointed out that ADD tools must: (a) support multiple views, since ADDs affect various architectural elements (AEs) and (b) support the modification of ADDs [23]. However, given that ADDs are not explicitly documented, they proposed Architectural Design Decision Recovery Approach to recover ADDs by comparing different releases of a software [24]. Miesbauer and Weinreich [25] argued that the *non-existence or ban decisions are not captured* and some places for documenting ADDs include *code, meeting minutes, project diaries, issue trackers, and wikis*. Van der Ven and Bosch also showed that *ADDs are implicitly captured in commit messages* of code repositories [26]. To document ADDs made at the source code level, [27] presents Java annotations to mark operations of reused components and [28] proposes to use Java Doc comments to inject architectural information. Another *data source for reverse engineering ADDs is the bug reports of software systems* [29]. Hence, Bhat et al. applied machine learning (ML) techniques to automatically extract ADDs from issue management systems [30]. [31] also presents an automated approach to extract ADDs from information maintained in issues and commit messages. TREx [32] uses domain ontologies to annotate text with architectural topics and predefined rules in TREx tag statements as either design decisions or design structures. Soliman et al. proposed a search mechanism to retrieve architectural information from StackOverflow which is an online developer community [33].

The proposal to extend the 4+1 view model with the decision view was made in [34]. In that paper, the authors capture the following requirements for managing ADDs: (a) the system should have *multi-perspective and multi-user support*, (b) *ADDs should have visual representation* so that they can be easily understood, and (c) *ADDs in large systems should have some classifications (hierarchy, abstraction)*. ADDs can be classified into strategic and tactical decisions [35]. To capture such ADDs, "architectural decision description template" was proposed which includes concepts such as Issue, Decision, Assumptions, Constraints, and Related decisions [36]. Another method to capture ADDs is to use mind-maps. Using which, the internal structure of ADDs can be captured in the form of diagrams [37]. A template-based approach used in the context

[1]https://github.com/manoj5864/evolutionOfADM

of service-oriented design is presented in [38]. It contains basic concepts related to ADDs but does not capture relationships between ADDs and software artifacts. On the other hand, [39] shows how to link structural and technological decisions with requirements and the corresponding architectural models.

In 2007, Clerc et al. found that even though, representing architecture as a set of ADDs was not completely adopted in practice, the concept of ADDs had gained importance among practitioners [40]. These ADDs address system behavior, components' interaction, system deployment, and evolution a system [41]. To reduce the effort in capturing ADDs, [42] presents a three-step approach: flag, filter, and form. When architects encounter an ADD, they flag it; then, with filtering, they find relevant ADDs, and lastly, forming means to create an ADD entry based on the decision model.

Selecting an architectural pattern is an essential ADD [43]. ADD recovery can be improved by focusing on those ADDs related to the application of specific architectural patterns [44]. In [45], a pattern model is linked to the component meta-model and the selection of a pattern and the corresponding components implicitly captures that ADD. Similarly, design patterns can be associated with specific quality attributes and addressing a quality concern by selecting a design pattern forms an ADD [46]. [47] presents a concept called Tradeoff-oriented Development and emphasizes that the realization of ADDs are based on quality attributes trade-offs.

Heijstek et al. found that neither diagrams nor textual descriptions were significantly more efficient for conveying ADDs [48]. It should be noted that the diagrams that were used was either component or class diagrams. We believe that it would have been more interesting, if those authors had conducted similar studies with, for instance, decision graphs. Shahin et al. evaluated rationale visualizations of ADDs and showed the improvement in architects' understanding [49]. They indicated that "in comparison to less experienced participants, experienced ones benefited more from ADDs" [50].

Tofan et al. [1] presents a systematic mapping study from 2005 to 2011 and confirms the growing interest in ADDs.

*A.1. Formal representation of ADDs*

Jansen and Bosch [51] presented Archium to represent ADDs as first-class entities. In [52], the authors proposed an ontology that is composed of architectural assets, ADDs, and concerns. Since this was a position paper, not much details about the ontology have been presented. However, Kruchten et al. [53] presented a detailed ontology for AK representation that captures a taxonomy of ADDs and their relationships. [54] also presents a taxonomy of ADDs to support system's runtime behavior analysis using models. [55] proposes to model and capture ADDs in an ontology using OWL.

Despite the importance of ADDs, they remain tacit and are lost over time. Hence, the authors of [56] provided a vocabulary for operations that change the design: addComponent, addConnector, deleteQualityRequirement, and so on. Such operations can be versioned to preserve ADDs' history.

The AK models presented in [57]–[62] capture concepts and links between stakeholders, architecture significant requirements (ASRs), ADDs, alternatives, rationale, AEs, and other software artifacts. The authors of [63] extended the model presented by Zimmermann et al. [61] to include links between decision outcomes and software artifacts. The authors of [64] discussed two types of ADDs (recurring and project-specific) and their effects on requirements and their prioritization. An approach called ADVERT helps architects to document ADDs, their rationale, and links ADDs to requirements and AEs [65].

To capture the traceability from ADDs to AEs in UML diagrams, [66] proposes UML profiles for capturing ADDs and non-functional requirements (NFRs). SysML also has been extended to support the design rationale model [67]. [68] proposes specific decision types and links them with UML diagrams. The link between ADDs and AEs is explicitly discussed in the models presented in [69]–[72]. Using these links and a constraint graph, impact analysis can be performed to identify the effects of changing ADDs. For documenting ADDs, authors of [73] presented "Maps of Architectural Decisions" which includes concepts such as concern, connector, solution, and decision-maker. In a related publication [74], a Concept-Requirement-Decision tree is used to hierarchically organize architectural topics. Similarly, a design map is used in [75] for capturing ADDs and tracing ADDs to NFRs. With the focus on facilitating traceability from requirements to design, [76] links concepts from the goal-oriented modeling language to the concepts in an ADD model. The authors of [77], [78] captured links between NFRs, ADDs, and AEs in their traceability information models. They proposed to recover ADDs and their traceability links from documents by training a ML classifier. Since ADDs are interrelated, [79] shows the application of a ripple effect algorithm on ADD networks to estimate the effect of change in a decision on other ADDs as well as to assess stable and unstable ADDs.

Capilla and Babar [80] showed how the models of the PAKME and ADDSS can be merged to support Product Line Engineering (PLE). The integrated model captures links between ADDs and the variability rules in the feature model. Furthermore, the need to capture a related concept – *vulnerability* of ADDs along with the rationale is discussed in [81].

Tang and Van Vliet categorized design constraints into requirement-, quality requirement-, context-, and solution-related constraints [82], [83]. These constraints influence ADDs. Che and Perry presented a Triple View Model [84], [85] to capture the What (element view), Why (intent view), and How (constraint view) of ADDs.

Zimmermann presented an ADM framework for service-oriented architecture (SOA) which assists the selection of a design that supports runtime models [86]. In another paper [87], [88], Zimmermann et al. investigated the use of ADD models for microservice architectures. Since understanding ADDs' context is critical for recommendations and reuse, they emphasized the need for capturing decision context [89].

In a series of publications [90]–[92], Zdun et al. presented CoCoADvISE. In the context of software ecosystems, it uses a knowledge base (KB) to support ADM. The KB maintains patterns, tactics, quality attributes, decision drivers, ADDs,

and domain experts' feedback. CoCoADvISE uses the KB to generate recommendations and questionnaires for ADM.

Technology decisions are one of the most often made ADDs [124]. Soliman et al. extended the AK model to capture technology decisions [125]. The AK model was also extended in [126]–[128] to reflect ADDs' sustainability. [129] presents metrics to measure the AK quality of decision models. Shahin et al. [130] stated the following: (a) "ADD models have consensuses on capturing ADDs, rationale, constraints, and alternatives", (b) "Tools do not exist for all the ADD models, some use text templates", and (c) "ADD *personalization is a desired feature that is missing in many ADD tools*".

*A.2. Architectural design decisions - tools*

We found 19 tools that have been proposed over the years to manage ADDs and to support the ADM process. These 19 tools have been briefly summarized in Table I. We agree with Alexeeva et al. [131] who suggest the following reasons for the lack of tools' adoption: (a) the absence of *industrial applicability requirements*, (b) marginal support of brownfield development, (c) insufficient consideration of documentation *overhead*, (d) missing *perspective on the evolution ADDs*, and (e) lack of *integration with commercial tools*. However, with recent developments in lightweight bottom-up approaches, some of these challenges are being addressed.

### *B. Architectural decision making*

The first publication which highlights that architects follow Naturalistic Decision Making (NDM) was in 1996 [132]. In that work, the author states that: "ADDs are handled *intuitively* and alternatives for an ADD are influenced by *experiences* from earlier designs". For ADM, [133] presents "system architecture analysis" method to ensure the consideration of alternatives, their pros and cons, and analyses of ADDs.

Zannier and Maurer [134] presented the definitions of Rationalistic Decision Making (RDM) and NDM and introduced factors such as *expertise* and *mental modeling* affecting ADM. In their subsequent work [135], they highlighted that *NDM dominates RDM* and *in NDM, experience and intuition play a crucial role*. They also found that agile projects supported better communication and debate about alternatives [136]. They suggested that, if a design problem is well-structured, architects tend to use RDM, whereas if it is ill-structured, then NDM is preferred [137]. Zannier and Maurer concluded that *architects do not always strive for optimal alternative and do not always consider alternatives* [138]. However, they mentioned that "alternatives are considered more often in groups (during casual conversations)" and ADM is a highly cognitive and a social process. [139] reports that architects use their expertise to arrive at an ADD without performing extensive [mental] search for an optimal alternative. Tang et al. too discussed that *architects do not explore many options* and finalize their decisions as soon as they have *good enough* supporting reasons [140]. They found that capturing design rationale helps architects improve architecture quality, backtrack ADDs [141], and *avoid architectural assumptions (AAs)* [142]. Furthermore, even though the term AAs is not commonly used, architects and developers frequently make AAs during ADM [143], [144]. Weinreich et al. argued that "*education, experience, and biases*" influence ADM [145] and classified ADDs according to granularity, scope, and impact [146].

[147] is one of the very first works, that discusses the influence of specific biases in ADM. In this work, Wirfs-Brock indicated the presence of *confirmation and information biases in design discussions*. In her recent work [148], based on an industrial case study, she (a) reflects on GDM and factors such as decision scope and trust influencing ADM and (b) shares recommendations on how to manage ADDs. Zalewski et al. documented 12 cognitive biases that are prominent in ADM [149]. [150] also covers the aspect of cognitive biases in ADM and introduces RDM, NDM, and bounded rationality. [151] maps cognitive biases to the different phases of ADM and presents a bias catalog to document related biases, examples, and debiasing strategies. The following points were made by Tang and Van Vliet [150]: (a) "people are irrational in general", (b) "individuals' rationality is by what they already know, cognitive limitations, and the finite amount of time", and (c) "people minimize cognitive load and use intuition in ADM". [152] proposes research directions towards the use of intuition along with rationality in ADM. Tang suggested that (a) cognitive bias, (b) illogical reasoning, and (c) low-quality premises are the cause of design reasoning failures [153]. Documentation frameworks address some of those problems. [154] illustrates the benefits of the decision forces viewpoint [155] in an industrial setting.

Furthermore, [156] indicates that decision makers' experience affects projects' return of investment. That study also reveals that factors such as decision makers' role or if they code are irrelevant. With the focus on the source code, [157] argues that developers (a) make many decisions, (b) "rarely conceptualize their work as decisions between more than two options", and (c) have trouble remembering multiple options.

[158] provides an overview of Analytic Hierarchy Process (AHP), simple multi-attribute rating technique, utility theory, and weighted score method. However, [159] argues that *companies do not use well-known ADM processes* but rather use their own customized approaches. [160] also confirms that *architects do not follow any systematic ADM approach* but instead follow informal approaches. The authors of [161] also argue that there is no one-size-fits-all ADM process and hence, to help architects, they provide hints for selecting an ADM technique that fits the usage context.

An ADM framework comprising of macro and micro levels is presented in [162]. The macro-level is used to analyze the design strategy using problem and solution orientations. Whereas, the micro-level uses the decision mode and the decision strategy for analysis. The authors of [162] argued that ADM is a creative process which includes problem recognition and hypotheses testing. Frameworks like GRADE [163], [164] and ORION [165] support the selection of sourcing options.

To support ADM, [166] combines risk-based reasoning with the quality attribute model. First, risks drive the selection of an architectural style to meet the quality goals, and then, the

TABLE I
TOOLS FOR MANAGING ADDS

| ADD Tool | Short description | Pub. year |
|---|---|---|
| ADDSS [93]–[100] | Over four years, Capilla et al. proposed a web-based ADD management system that addressed use cases of recording, navigating, visualizing ADDs as well as supporting groups for managing ADDs. The tool was validated in through experiments in restricted environments. | 2006 - 2009 |
| ADD Visualization Tool [101], [102] | In this tool, the authors showed how ADDs can be visualized by investigating four aspects of ADDs visualizations and how they can influence ADDs exploration and analysis. These visualizations were validated by practitioners in a laboratory setting. | 2008 |
| DecisionStickies [103] | This tool supports three approaches of capturing ADDs: formal elicitation, top-down (from the existing SA documentation), and bottom-up (annotations in the source code). Given these options, users were able to choose the best suitable approach for their organizational context. Validation of this tool was performed during joint meetings with practitioners, where the feasibility of the approaches was demonstrated; however, the authors indicated that further work is required to enhance the tool. | 2008 |
| Decision Management Tool [104] | The integration of managing ADDs in the UML modeling environment with the intent to enforce a change in the model after new decisions are made is emphasized in this tool. The validation of the tool was not performed. | 2009 |
| ODV [105] | Uses table and matrix representations of ADDs and quality attributes to support architects in the early phase of a project (in particular, to perform trade-offs, impact, and if-then scenarios analysis). The tool was validated in a restricted environment. | 2009 |
| ADDDMS [106], [107] | Authors argue the importance of the customizability of tools for managing ADDs. In the proposed solution, architects in order to manage ADDs can utilize customizable templates, storage, and search functionalities. The validation of this tool was performed in laboratory settings. | 2010 |
| Rationale visualization [49], [108], [109] | The authors demonstrated an approach to visualize ADDs using the Compendium tool - "a visual environment for people to structure and record outcomes of collaborative work on wicked problems". | 2010 - 2011 |
| LISA Toolkit [110] | LISA automatically traces and visualizes ADDs made during the architectural and implementation phases within an environment (specifically, in Eclipse IDE). This tool was validated through action research by using it in three software engineering projects. | 2011 |
| RGT tool [111], [112] | The web-based tool that employees a repertory grid technique (borrowed from the knowledge engineering domain and originally coming from the personal construct psychological theory) to support architects in eliciting and analyzing ADDs and their alternatives. The approach itself was evaluated in an exploratory study [111], however, in [112], the authors do not share any details on how the tool itself was validated. | 2011 - 2014 |
| ADUAK [113] | ADUAK allows architects to capture and explore the selection of design patterns that can be applied in a project. The validation was performed by the authors themselves through the implementation of a flight reservation system. | 2012 |
| ArchiTech [114] | Repository-based tool intended to be used during the system design phase. ArchiTech supports architects in capturing ADDs and linking them to NFRs. Based on the links, the tool can generate a prioritized list of decisions that satisfy as much of architectural constraints as possible. In case the decision has been marked as "made", the system shows a list of concerns that should be taken into account by the architect. No validation was performed. | 2012 |
| DPS [115] | Authors demonstrated how a specific tool for video recording of design sessions can support architects in capturing ADDs. | 2012 |
| ADvISE [116] | ADvISE is an approach, as well as, an Eclipse-based tool that supports architects to reason about ADDs within their project (also, supports uncertainty using fuzzy logic). It uses the Questions, Options, and Criteria method. The applicability of the approach was demonstrated in the context of an industrial case study on service-based platform integration in the area of industry automation. | 2013 |
| Decision Architect [117] | A plugin for Sparx Enterprise Architect modeling tool to manage ADDs and its associated concepts. The plugin was validated through the application in pilot studies within the industry, which was followed by user interviews. | 2014 |
| DecDoc [118] | An Eclipse plugin to capture ADDs and link them to artifacts such as requirements, AEs, and code. The tool was tested on the data from an exemplary project. | 2016 |
| Ontology-based recommender [119] | Uses the DBpedia ontology to automatically identify AEs and to generate decision alternatives. The tool was validated using architecture documents from industrial projects. | 2017 |
| Quiver [120] | A web-based solution that maintains a KB of reference architectures and styles. No validation has been performed so far. | 2017 |
| EVA [121] | Supports the visualization of architecture evolution and helps architects to understand the impact of ADDs on architectural stability. The validation was performed using data from an open-source project. No empirical study has been conducted so far. | 2018 |
| ADeX [122], [123] | A bottom-up approach to automatically detect ADDs and provide ADM support. Automatically identifies experts who should be involved in ADM. The quantitative evaluation of the system components has been discussed in [30], [119], [122]. The qualitative validation of the tool was performed following the action research methodology in collaboration with an industry partner during the last four years [123]. | 2017 - 2019 |

goals help to choose architectural tactics. A framework called COMPonents using ArChitectural Tactics uses architectural tactics to search and select software components that meet NFRs [167]. In [168], the authors found that quality attributes strongly influence ADM and ADDs (in specific projects) had been made in small teams or by individuals. [169] proposes to automatically extract quality attributes from user stories so that those attributes can be used as a basis for ADM.

In the context of agile development, [170] presents a responsibility-driven architecture to understand when, how, and who should make ADDs. [171] explains how the role of an architect is changing as team player making ADDs only when necessary in agile projects.

A decision-centric design approach is discussed in [172]; wherein, an issue leads to multiple candidate solutions, selecting a solution reflects an ADD, and arguments for that solution become the rationale. Lytra et al. proposed a frequent-items set method to investigate which decision points may (not) coexist in the decision space [173].

[174] models ADM using BPMN activities. The activities include motivating an ADD, specifying alternatives, selecting an alternative collaboratively. Another process-based approach uses a tag-based traceability system to support collaboration among architects, notify them about changes, and include feedback loops to improve ADM [175].

Intuitive games using cards (representing constrain, assumption, risk, and tradeoff) are used to teach ADM [176], [177]. Since architects forget to reason about their ADDs [178], such card games could help to spark architects' reasoning process. Furthermore, tools like DVIA [179]–[181] and video wall [182] which support collaborative ADM and help to record and analyze architectural meeting discussions can aid architects to reflect on the past discussions.

An important aspect that should be considered by any ADM support system is that they should explicitly include reflective questions. The authors of [183] distinguish between two minds. Mind 1 reflects the design reasoning mind with the problem-solving mindset, whereas, Mind 2 is the reflective mind with a feedback mindset. ADM tools should include *mechanisms to trigger and reflect on Mind 1's activities*. Alternative to Mind 1 and Mind 2, Pretorius [184] proposes to evaluate the applicability of Kahneman's System 1 and System 2 modes in the context of ADM.

Tofan et al. [185] suggested the following research directions; there is a need for *addressing uncertainty in ADM*, *better approaches for GDM*, and *improved understanding of dependencies between ADDs and effort estimation*. [3] emphasizes on the human aspects in ADM and argues that even after gaining insights on the behavioral facets influencing ADM, there has not been a significant shift towards new practices. For a more detailed discussion on the factors influencing ADM, readers are directed to the work by Tang at al. [2].

*B.1. Optimization-based approaches*

AHP is one of the popularly used methods to choose an "optimal" design alternative [186], [187]. For example, AHP is used to select those components that meet quality criteria [188], [189]. First, all the goals are captured hierarchically, the criteria satisfying those objectives are specified, and the alternatives are listed. Second, at each level of the hierarchy, a decision table is created. Finally, a constraint solver prioritizes and selects an optimal alternative that meets the constraints or a search-based optimizer performs heuristic sampling to select an optimal option. [190] uses the hierarchical criterion structure based on the criteria importance theory. Since architects do not define all the necessary constraints, constraint solvers can be used to reduce infeasible options that are inconsistent with previous choices [191].

The design task is represented as a search problem in [192]. The relationships between ADDs are captured using superior and inferior relation types, and when a superior decision is under consideration, all its inferior decisions are treated jointly to identify inconsistencies. [193]–[196] consider ADM to be a Multiple-Criteria Decision-Making (MCDM) problem. By using occurrence probabilities for each combination of alternatives, the consequence of each alternative can be evaluated. A tool named RADAR uses a multicriteria optimization approach [197] to select an optimal alternative. A web-based tool called Decision buddy allows users to capture issues and alternatives, prioritize alternatives using a constraint solver, and finally approve the alternative that meets the criteria [198].

Fuzzy inference can also be used to select an optimal alternative that meets stakeholders' [quantified] goals [199], [200]. The Multicriteria decision aid method can be used to select an alternative while considering fuzziness in stakeholders' needs [201]. Esfahani et al. use a fuzzification approach to represent uncertainty as a triangular fuzzy value to support the exploration of the solution space under uncertainty [202].

A weighted-score approach is used to compare choices against decision criteria as well as to compare the options against ideal solutions [203]. Imran et al. also proposed a weighted-score approach for ranking and selecting the best architectural pattern [204]. A tool called DesignBots takes as input the system design and a weighted quality constraints list to generate alternatives for improving the design [205].

[206] presents a model-driven approach for the evaluation of ADDs while considering quality attributes. They claim that their system can be used to select an optimal decision when its impact on the quality is unclear. Another interesting approach uses MCMD to explicitly consider ethics in ADM [207]. Lastly, Shahbazian et al. present a search and simulation-based approach for understanding the impact of ADDs [208].

### C. Group Decision Making

Alali and Sillito [209] described that the motivation for collaboration include improving ADDs and sharing ADM effort. [210] shows that the majority of software teams make ADDs collaboratively and *prefer consultative ADM style* as it helps to consider the inputs from all team members.

An "ONTOlogy-based Group Decision Support System" allows architects to participate in the GDM process [211]. The system relies on a group argumentation model to support conflict resolution. Software Architecture Warehouse (SAW)

also uses an argumentation viewpoint [212]. In [213], the authors propose GADGET to help architects increase consensus during GDM. In GADGET, for a decision topic, alternatives and concerns are discussed, those alternatives are prioritized, and based on that, architects in a group aim to reach consensus.

The authors of [214] suggested to combine ADM with Scrum. They proposed GDM steps to be included in a sprint: (a) problem identification, (b) development of alternatives, (c) preference prioritization, and (d) reaching consensus.

To reach consensus in GDM, Smrithi and Muccini argued that instead of structured approaches, brainstorming is used in 70% of the companies [215]. They found that *conflicting decisions and misunderstanding of goals* were the challenges in GDM. They also discussed that the current methods do not suit GDM [216]. Methods should include stakeholders' preferences, rules indicating how those preferences should be considered, and mechanisms for conflict resolution. They suggested including GDM strategies into an architecting phase to not only capture ADDs but also to document the GDM factors that result in those ADDs [217]. Based on these observations, they extended the ADD model with concepts from GDM and organizational structures [218]. In their recent work, Smrithi and Muccini [219] showed that a standard way of ADM is less common and tools for ADM are rarely used (since the *quality is below satisfactory*). Interestingly, they also indicated that "despite the involvement of team members in discussions, the final decision is made by an individual".

Shumaiev et al. [220] showed that uncertainty expressions are used in the GDM process. However, they highlighted that those expressions are not only used to express uncertainty but also to trigger feedback, to indicate preference, for reassurance, and as a figure of speech. This finding raises concerns on the applicability of rule- or ML-based approaches to detect uncertainties in ADM.

## IV. Conclusion

In this work, we have briefly touched upon 218 publications that discuss topics related to ADM. Specifically, we looked at those publications that emphasize the need to capture ADDs, ADD models for AK representation, and tool support for managing ADDs. Next, we discussed ADM along with a few optimization-based approaches and followed it with GDM.

We observe that from 2004 to 2010, researchers focused on modeling ADDs, proposed tools to manage ADDs (mostly, top-down approaches). In the second wave, from 2010 to 2015, we saw the emergence of lightweight and bottom-up approaches that emphasized minimal documentation using ADD templates and automatic extraction of ADDs from different information sources. Researchers discussed the challenges with the "first-generation" tools and criticized their lack of adoption. In the recent years, while addressing the challenges of the existing ADD tools, researchers have shifted their focus towards addressing the social and psychological aspects of ADM. This covers various topics including NDM, bounded rationality, biases, uncertainties, AAs, and sustainability.

Given the detailed literature review protocol, as well as, a link to the replication package, researchers can extend this study beyond the time-frame (1980-2019) selected for the review presented in this paper. With the getting-started guide, our objective mainly focused on summarizing the selected publications, rather than to synthesis and make generalizability claims in the conclusion. We encourage the readers to treat the presented historical catalog of publications on ADDs and ADM as a study made from the perspective of the authors. Even though we have followed a systematic approach and at least two authors were involved at each phase of the study, due to a large number of publications under consideration, there is a possibility of human error in counting the publications or missing out on including the key contribution of some of the publications while ensuring the readability of this paper.

The fact that the publications summarized in our work include most of the publications discussed in previous literature study conducted by Tofan et al. [1] strengthens our study's construction validity. On the other hand, not considering a quasi-gold standard of publications (similar to the one used by Tofan et al. in [1]) poses a threat to validity and we cannot quantitatively analyze the relevant publications that we might have missed to consider. Hence, manually creating a quasi-gold standard of publications by methodologically executing a survey of researchers and practitioners working in the field might be beneficial. Furthermore, the utilization of such a new quasi-gold standard might consequently lead to the readjustment of the search strategy and the review process thereof, for instance, this might increase the number of the required search terms for reaching the final consolidated list of publications.

To the readers: use this work ***only as*** a getting-started guide to know about the existing work. Since this paper only presets a very short description for the publications, they should pick the publications of their interest and investigate it further.

## References


[1] D. Tofan, M. Galster, P. Avgeriou, and W. Schuitema, "Past and future of software architectural decisions–a systematic mapping study," *Information and Software Technology*, vol. 56, no. 8, pp. 850–872, 2014.

[2] A. Tang, M. Razavian, B. Paech, and T.-M. Hesse, "Human aspects in software architecture decision making: a literature review," in *ICSA*. IEEE, 2017, pp. 107–116.

[3] M. Razavian, B. Paech, and A. Tang, "Empirical research for software architecture decision making: An analysis," *Journal of Systems and Software*, vol. 149, pp. 360–381, 2019.

[4] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Keele University and Durham University Joint Report, Tech. Rep. EBSE 2007-001, 2007.

[5] M. Mahabaleshwar, "Tool support for architectural decision making in large software intensive projects," TUM, Germany, submitted-2019.

[6] R. Capilla, A. Jansen, A. Tang, P. Avgeriou, and M. A. Babar, "10 years of software architecture knowledge management: Practice and future," *Journal of Systems and Software*, vol. 116, pp. 191–205, 2016.

[7] M. Sintzoff, "Suggestions for composing and specifying program design decisions," in *International Symposium on Programming*. Springer, 1980, pp. 311–326.

[8] S. Rugaber, S. B. Ornburn, and R. J. LeBlanc, "Recognizing design decisions in programs," *IEEE Software*, vol. 7, no. 1, pp. 46–54, 1990.

[9] A. Cimitile, F. Lanubile, and G. Visaggio, "Traceability based on design decisions," in *ICSM*. IEEE, 1992, pp. 309–317.

[10] P. Chung and R. Goodwin, "Representing design history," in *Artificial Intelligence in Design'94*. Springer, 1994, pp. 735–752.

[11] P. B. Kruchten, "The 4+1 view model of architecture," *IEEE software*, vol. 12, no. 6, pp. 42–50, 1995.

[12] A. Ran and J. Kuusela, "Design decision trees," in *IWSSD*. IEEE Computer Society, 1996, p. 172.

[13] P. Kruchten, "The software architect," in *Software architecture*. Springer, 1999, pp. 565–583.

[14] J. Tyree, "Architectural design decisions session report," in *WICSA*. IEEE, 2005, pp. 285–286.

[15] P. Kruchten, P. Lago, H. Van Vliet, and T. Wolf, "Building up and exploiting architectural knowledge," in *WICSA*. IEEE, 2005, pp. 291–292.

[16] D. Ameller, C. Ayala, J. Cabot, and X. Franch, "Non-functional requirements in architectural decision making," *IEEE software*, vol. 30, no. 2, pp. 61–67, 2013.

[17] R. Kazman, J. Asundi, and M. Klein, "Quantifying the costs and benefits of architectural decisions," in *ICSE*. IEEE, 2001, pp. 297–306.

[18] M. Moore, R. Kazman, M. Klein, and J. Asundi, "Quantifying the value of architecture design decisions: lessons from the field," in *ICSE*. IEEE Computer Society, 2003, pp. 557–562.

[19] P. C. Clements, "An economic model for software architecture decisions," in *ESC*. IEEE Computer Society, 2007, p. 1.

[20] J. Carriere, R. Kazman, and I. Ozkaya, "A cost-benefit framework for making architectural decisions in a business context," in *ICSE*, vol. 2. IEEE, 2010, pp. 149–157.

[21] A. Ran, "Fundamental concepts for practical software architecture," *ACM SIGSOFT Software Engineering Notes*, vol. 26, no. 5, pp. 328–329, 2001.

[22] J. Bosch, "Software architecture: The next step," in *EWSA*. Springer, 2004, pp. 194–199.

[23] A. Jansen and J. Bosch, "Evaluation of tool support for architectural evolution," in *ASE*. IEEE Computer Society, 2004, pp. 375–378.

[24] A. Jansen, J. Bosch, and P. Avgeriou, "Documenting after the fact: Recovering architectural design decisions," *Journal of Systems and Software*, vol. 81, no. 4, pp. 536–557, 2008.

[25] C. Miesbauer and R. Weinreich, "Classification of design decisions–an expert survey in practice," in *ECSA*. Springer, 2013, pp. 130–145.

[26] J. S. van der Ven and J. Bosch, "Making the right decision - supporting architects with design decision data," in *ECSA*. Springer, 2013, pp. 176–183.

[27] A. Calvagna and E. Tramontana, "Delivering dependable reusable components by expressing and enforcing design decisions," in *COMPSACW*. IEEE, 2013, pp. 493–498.

[28] V. Bandara and I. Perera, "Identifying software architecture erosion through code comments," in *ICTer*. IEEE, 2018, pp. 62–69.

[29] A. J. Ko and P. K. Chilana, "Design, discussion, and dissent in open bug reports," in *iConference*. ACM, 2011, pp. 106–113.

[30] M. Bhat, K. Shumaiev, A. Biesdorf, U. Hohenstein, and F. Matthes, "Automatic extraction of design decisions from issue management systems: a machine learning based approach," in *ECSA*. Springer, 2017, pp. 138–154.

[31] A. Shahbazian, Y. K. Lee, D. Le, Y. Brun, and N. Medvidovic, "Recovering architectural design decisions," in *ICSA*. IEEE, 2018, pp. 95–9509.

[32] H. Astudillo, G. Valdés, and C. Becerra, "Empirical measurement of automated recovery of design decisions and structure," in *ANDESCON*. IEEE Computer Society, 2012, pp. 105–108.

[33] M. Soliman, A. R. Salama, M. Galster, O. Zimmermann, and M. Riebisch, "Improving the search for architecture knowledge in online developer communities," in *ICSA*. IEEE, 2018, pp. 186–18 609.

[34] J. C. Dueñas and R. Capilla, "The decision view of software architecture," in *EWSA*. Springer, 2005, pp. 222–230.

[35] A. H. Eden, "Strategic versus tactical design," in *HICSS*. IEEE, 2005, pp. 313a–313a.

[36] J. Tyree and A. Akerman, "Architecture decisions: Demystifying architecture," *IEEE software*, vol. 22, no. 2, pp. 19–27, 2005.

[37] A. Zalewski and M. Ludzia, "Diagrammatic modeling of architectural decisions," in *ECSA*. Springer, 2008, pp. 350–353.

[38] Q. Gu, P. Lago, and H. Van Vliet, "A template for soa design decision making in an educational setting," in *SEAA*. IEEE, 2010, pp. 175–182.

[39] D. Dermeval, J. Pimentel, C. Silva, J. Castro, E. Santos, G. Guedes, A. Finkelstein *et al.*, "Stream-add-supporting the documentation of architectural design decisions in an architecture derivation process," in *COMPSAC*. IEEE, 2012, pp. 602–611.

[40] V. Clerc, P. Lago, and H. Van Vliet, "The architect's mindset," in *QoSA*. Springer, 2007, pp. 231–249.

[41] N. Medvidovic and R. N. Taylor, "Software architecture: foundations, theory, and practice," in *ICSE*. ACM, 2010, pp. 471–472.

[42] L. Lee and P. Kruchten, "Capturing software architectural design decisions," in *CCECE*. IEEE, 2007, pp. 686–689.

[43] N. B. Harrison, P. Avgeriou, and U. Zdun, "Using patterns to capture architectural decisions," *IEEE software*, vol. 24, no. 4, 2007.

[44] U. van Heesch, P. Avgeriou, U. Zdun, and N. Harrison, "The supportive effect of patterns in architecture decision recovery-a controlled experiment," *Science of Computer Programming*, vol. 77, no. 5, 2012.

[45] M. T. T. That, S. Sadou, and F. Oquendo, "Using architectural patterns to define architectural decisions," in *WICSA-ECSA*. IEEE, 2012, pp. 196–200.

[46] T. Aslam, T. Rana, M. Batool, A. Naheed, and A. Andaleeb, "Quality based software architectural decision making," in *ComTech*. IEEE, 2019, pp. 114–119.

[47] T. Dürschmid, E. Kang, and D. Garlan, "Trade-off-oriented development: making quality attribute trade-offs first-class," in *ICSE: NIER*. IEEE Press, 2019, pp. 109–112.

[48] W. Heijstek, T. Kuhne, and M. R. Chaudron, "Experimental analysis of textual and graphical representations for software architecture design," in *ESEM*. IEEE, 2011, pp. 167–176.

[49] M. Shahin, P. Liang, and Z. Li, "Architectural design decision visualization for architecture design: preliminary results of a controlled experiment," in *ECSA-C*. ACM, 2011, p. 2.

[50] M. Shahin and P. Liang and Z. Li, "Do architectural design decisions improve the understanding of software architecture? two controlled experiments," in *ICPC*. ACM, 2014, pp. 3–13.

[51] A. Jansen and J. Bosch, "Software architecture as a set of architectural design decisions," in *WICSA*. IEEE, 2005, pp. 109–120.

[52] A. Akerman and J. Tyree, "Position on ontology-based architecture," in *WICSA*. IEEE, 2005, pp. 289–290.

[53] P. Kruchten, P. Lago, and H. Van Vliet, "Building up and reasoning about architectural knowledge," in *QoSA*. Springer, 2006, pp. 43–58.

[54] M. Szvetits and U. Zdun, "Architectural design decisions for systems supporting model-based analysis of runtime events: A qualitative multi-method study," in *ICSA*. IEEE, 2018, pp. 115–11 509.

[55] Y. C. Segura, N. S. Martínez, A. P. Fernández, and O. G. Baryolo, "Description and analysis of design decisions: An ontological approach," in *CITI*. Springer, 2018, pp. 174–185.

[56] M. L. Roldán, S. Gonnet, and H. Leone, "A model for capturing and tracing architectural designs," in *Advanced Software Engineering: Expanding the Frontiers of Software Technology*. Springer, 2006, pp. 16–31.

[57] R. C. De Boer, R. Farenhorst, P. Lago, H. Van Vliet, V. Clerc, and A. Jansen, "Architectural knowledge: Getting to the core," in *QoSA*. Springer, 2007, pp. 197–214.

[58] F. Gilson and V. Englebert, "Rationale, decisions and alternatives traceability for architecture design," in *ECSA-C*. ACM, 2011, p. 4.

[59] B. Orlic, R. Mak, I. David, and J. Lukkien, "Concepts and diagram elements for architectural knowledge management," in *ECSA-C*. ACM, 2011, p. 3.

[60] M. Bhat, K. Shumaiev, A. Biesdorf, U. Hohenstein, M. Hassel, and F. Matthes, "Meta-model based framework for architectural knowledge management," in *ECSAW*. ACM, 2016, p. 12.

[61] O. Zimmermann, T. Gschwind, J. Küster, F. Leymann, and N. Schuster, "Reusable architectural decision models for enterprise application development," in *QoSA*. Springer, 2007, pp. 15–32.

[62] R. Farenhorst, P. Lago, and H. Van Vliet, "Prerequisites for successful architectural knowledge sharing," in *ASWEC*. IEEE, 2007, pp. 27–38.

[63] S. Gerdes, S. Lehnert, and M. Riebisch, "Combining architectural design decisions and legacy system evolution," in *ECSA*. Springer, 2014, pp. 50–57.

[64] Z. Durdik, A. Koziolek, and R. H. Reussner, "How the understanding of the effects of design decisions informs requirements engineering," in *TwinPeaks*. IEEE, 2013, pp. 14–18.

[65] M. Konersmann, Z. Durdik, M. Goedicke, and R. H. Reussner, "Towards architecture-centric evolution of long-living systems (the advert approach)," in *QoSA*. ACM, 2013, pp. 163–168.

[66] L. Zhu and I. Gorton, "Uml profiles for design decisions and non-functional requirements," in *SHARK-ADI*. IEEE Computer Society, 2007, p. 8.

[67] S. De Wang, J. H. Liu, and C. Fu, "Sysml extension method supporting design rationale knowledge model," in *CDVE*. Springer, 2018, pp. 225–228.

[68] M. Küster, "Architecture-centric modeling of design decisions for validation and traceability," in *ECSA*. Springer, 2013, pp. 184–191.

[69] Y. Choi, H. Choi, and M. Oh, "An architectural design decision-centric approach to architectural evolution," in *ICACT*, vol. 1. IEEE, 2009, pp. 417–422.

[70] E. Navarro, C. E. Cuesta, and D. E. Perry, "Weaving a network of architectural knowledge," in *WICSA-ECSA*. IEEE, 2009, pp. 241–244.

[71] I. Lytra, H. Tran, and U. Zdun, "Constraint-based consistency checking between design decisions and component models for supporting software architecture evolution," in *CSMR*. IEEE, 2012, pp. 287–296.

[72] M. Soliman and M. Riebisch, "Modeling the interactions between decisions within software architecture knowledge," in *ECSA*. Springer, 2014, pp. 33–40.

[73] M. Szlenk, A. Zalewski, and S. Kijas, "Modelling architectural decisions under changing requirements," in *WICSA-ECSA*. IEEE, 2012, pp. 211–214.

[74] J. P. Ros and R. S. Sangwan, "A method for evidence-based architecture discovery," in *WICSA*. IEEE, 2011, pp. 342–345.

[75] A. Sawada, M. Noro, H.-M. Chang, Y. Hachisu, and A. Yoshida, "A design map for recording precise architecture decisions," in *APSEC*. IEEE, 2011, pp. 298–305.

[76] D. Dermeval, J. Castro, C. Silva, J. Pimentel, I. I. Bittencourt, P. Brito, E. Elias, T. Tenório, and A. Pedro, "On the use of metamodeling for relating requirements and architectural design decisions," in *SAC*. ACM, 2013, pp. 1278–1283.

[77] M. Mirakhorli and J. Cleland-Huang, "Transforming trace information in architectural documents into re-usable and effective traceability links," in *SHARK*. ACM, 2011, pp. 45–52.

[78] M. Mirakhorli, "Tracing architecturally significant requirements: a decision-centric approach," in *ICSA*. ACM, 2011, pp. 1126–1127.

[79] C. Carrillo and R. Capilla, "Ripple effect to evaluate the impact of changes in architectural design decisions." in *ECSA-C*, 2018, p. 41.

[80] R. Capilla and M. A. Babar, "On the role of architectural design decisions in software product line engineering," in *ECSA*. Springer, 2008, pp. 241–255.

[81] P. G. Avery and R. Hawkins, "Software design decision vulnerability analysis," 2014.

[82] A. Tang and H. Van Vliet, "Modeling constraints improves software architecture design reasoning," in *WICSA-ECSA*. IEEE, 2009, pp. 253–256.

[83] A. Tang and H. Van Vliet, "Software architecture design reasoning," in *Software Architecture Knowledge Management*. Springer, 2009, pp. 155–174.

[84] M. Che and D. E. Perry, "Scenario-based architectural design decisions documentation and evolution," in *ECBS*. IEEE, 2011, pp. 216–225.

[85] M. Che, "An approach to documenting and evolving architectural design decisions," in *ICSE*. IEEE Press, 2013, pp. 1373–1376.

[86] O. Zimmermann, "Architectural decisions as reusable design assets," *IEEE software*, vol. 28, no. 1, pp. 64–69, 2011.

[87] C. Carrillo, R. Capilla, O. Zimmermann, and U. Zdun, "Guidelines and metrics for configurable and sustainable architectural knowledge modelling," in *ECSAW*. ACM, 2015, p. 63.

[88] U. Zdun, M. Stocker, O. Zimmermann, C. Pautasso, and D. Lübke, "Guiding architectural decision making on quality aspects in microservice apis," in *ICSOC*. Springer, 2018, pp. 73–89.

[89] J. Carlson, E. Papatheocharous, and K. Petersen, "A context model for architectural decision support," in *MARCH*. IEEE, 2016, pp. 9–15.

[90] I. Lytra, P. Gaubatz, and U. Zdun, "Two controlled experiments on model-based architectural decision making," *Information and Software Technology*, vol. 63, pp. 58–75, 2015.

[91] S. Stevanetic, K. Plakidas, T. B. Ionescu, F. Li, D. Schall, and U. Zdun, "Tool support for the architectural design decisions in software ecosystems," in *ECSAW*. ACM, 2015, p. 45.

[92] I. Lytra, G. Engelbrecht, D. Schall, and U. Zdun, "Reusable architectural decision models for quality-driven decision support: A case study from a smart cities software ecosystem," in *SESoS*. IEEE Press, 2015, pp. 37–43.

[93] R. Capilla, F. Nava, S. Pérez, and J. C. Dueñas, "A web-based tool for managing architectural design decisions," *ACM SIGSOFT software engineering notes*, vol. 31, no. 5, p. 4, 2006.

[94] R. Capilla, F. Nava, J. Montes, and C. Carrillo, "Addss: architecture design decision support system tool," in *ASE*. IEEE Computer Society, 2008, pp. 487–488.

[95] R. Capilla and F. Nava, "Extending software architecting processes with decision-making activities," in *Balancing Agility and Formalism in Software Engineering*. Springer, 2008, pp. 182–195.

[96] R. Capilla, F. Nava, and J. C. Duenas, "Modeling and documenting the evolution of architectural design decisions," in *SHARK-ADI*. IEEE Computer Society, 2007, p. 9.

[97] R. Capilla, F. Nava, and A. Tang, "Attributes for characterizing the evolution of architectural design decisions," in *IWSE*. IEEE, 2007, pp. 15–22.

[98] F. Nava, R. Capilla, and J. C. Dueñas, "Processes for creating and exploiting architectural design decisions with tool support," in *ECSA*. Springer, 2007, pp. 321–324.

[99] R. Capilla, F. Nava, and C. Carrillo, "Effort estimation in capturing architectural knowledge," in *ASE*. IEEE Computer Society, 2008, pp. 208–217.

[100] R. Capilla, "Embedded design rationale in software architecture," in *WICSA-ECSA*. IEEE, 2009, pp. 305–308.

[101] L. Lee and P. Kruchten, "A tool to visualize architectural design decisions," in *QoSA*. Springer, 2008, pp. 43–54.

[102] L. Lee and P. Kruchten, "Visualizing software architectural design decisions," in *ECSA*. Springer, 2008, pp. 359–362.

[103] L. Lee and P. Kruchten, "Customizing the capture of software architectural design decisions," in *CCECE*. IEEE, 2008, pp. 000 693–000 698.

[104] P. Konemann, "Integrating decision management with uml modeling concepts and tools," in *WICSA-ECSA*. IEEE, 2009, pp. 297–300.

[105] R. C. De Boer, P. Lago, A. Telea, and H. Van Vliet, "Ontology-driven visualization of architectural design decisions," in *WICSA-ECSA*. IEEE, 2009, pp. 51–60.

[106] L. Chen and M. A. Babar, "Supporting customizable architectural design decision management," in *ECBS*. IEEE, 2010, pp. 232–240.

[107] L. Chen, M. A. Babar, and H. Liang, "Model-centered customizable architectural design decisions management," in *ASEC*. IEEE, 2010, pp. 23–32.

[108] M. Shahin, P. Liang, and M. R. Khayyambashi, "Rationale visualization of software architectural design decision using compendium," in *SAC*. ACM, 2010, pp. 2367–2368.

[109] M. Shahin and P. Liang and M. R. Khayyambashi, "Improving understandability of architecture design through visualization of architectural design decision," in *SHARK*. ACM, 2010, pp. 88–95.

[110] G. Buchgeher and R. Weinreich, "Automatic tracing of decisions to architecture and implementation," in *WICSA*. IEEE, 2011, pp. 46–55.

[111] D. Tofan, M. Galster, and P. Avgeriou, "Capturing tacit architectural knowledge using the repertory grid technique (nier track)," in *ICSE*. ACM, 2011, pp. 916–919.

[112] D. Tofan and M. Galster, "Capturing and making architectural decisions: an open source online tool," in *ECSAW*. ACM, 2014, p. 33.

[113] C. Dhaya and G. Zayaraz, "Development of multiple architectural designs using aduak," in *ICCSPA*. IEEE, 2012, pp. 93–97.

[114] D. Ameller, O. Collell, and X. Franch, "Architech: Tool support for nfr-guided architectural decision-making," in *RE*. IEEE, 2012, pp. 315–316.

[115] K. Nakakoji, Y. Yamamoto, N. Matsubara, and Y. Shirai, "Toward unweaving streams of thought for reflection in professional software design," *IEEE software*, vol. 29, no. 1, pp. 34–38, 2012.

[116] I. Lytra, H. Tran, and U. Zdun, "Supporting consistency between architectural design decisions and component models through reusable architectural knowledge transformations," in *ECSA*. Springer, 2013, pp. 224–239.

[117] C. Manteuffel, D. Tofan, H. Koziolek, T. Goldschmidt, and P. Avgeriou, "Industrial implementation of a documentation framework for architectural decisions," in *WICSA*. IEEE, 2014, pp. 225–234.

[118] T.-M. Hesse, A. Kuehlwein, and T. Roehm, "Decdoc: A tool for documenting design decisions collaboratively and incrementally," in *MARCH*. IEEE, 2016, pp. 30–37.

[119] M. Bhat, K. Shumaiev, A. Biesdorf, U. Hohenstein, M. Hassel, and F. Matthes, "An ontology-based approach for software architecture recommendations," in *AMCIS*, 2017. [Online]. Available: http://aisel.aisnet.org/amcis2017/SemanticsIS/Presentations/7

[120] A. Gopalakrishnan and A. C. Biswal, "Quiver an intelligent decision support system for software architecture and design," in *SmartTechCon*. IEEE, 2017, pp. 1286–1291.

[121] D. Nam, Y. K. Lee, and N. Medvidovic, "Eva: A tool for visualizing software architectural evolution," in *ICSE-C*. IEEE, 2018, pp. 53–56.

[122] M. Bhat, K. Shumaiev, K. Koch, U. Hohenstein, A. Biesdorf, and F. Matthes, "An expert recommendation system for design decision making: Who should be involved in making a design decision?" in *ICSA*. IEEE, 2018, pp. 85–8509.

[123] M. Bhat, C. Tinnes, K. Shumaiev, , A. Biesdorf, U. Hohenstein, and F. Matthes, "Adex: A tool for automatic curation of design decision knowledge for architectural decision recommendations," in *ICSA-C*. IEEE, 2019.

[124] T. D. LaToza, E. Shabani, and A. Van Der Hoek, "A study of architectural decision practices," in *CHASE*. IEEE, 2013, pp. 77–80.

[125] M. Soliman, M. Riebisch, and U. Zdun, "Enriching architecture knowledge with technology design decisions," in *WICSA*. IEEE, 2015, pp. 135–144.

[126] C. C. Venters, R. Capilla, S. Betz, B. Penzenstadler, T. Crick, S. Crouch, E. Y. Nakagawa, C. Becker, and C. Carrillo, "Software sustainability: Research and practice from a software architecture viewpoint," *Journal of Systems and Software*, 2017.

[127] U. Zdun, R. Capilla, H. Tran, and O. Zimmermann, "Sustainable architectural design decisions," *IEEE software*, vol. 30, no. 6, pp. 46–53, 2013.

[128] P. Lago, "Architecture design decision maps for software sustainability," in *ICSE: SEIS*. IEEE Press, 2019, pp. 61–64.

[129] M. Nowak and C. Pautasso, "Goals, questions and metrics for architectural decision models," in *SHARK*. ACM, 2011, pp. 21–28.

[130] M. Shahin, P. Liang, and M. R. Khayyambashi, "Architectural design decision: Existing models and tools," in *WICSA-ECSA*. IEEE, 2009, pp. 293–296.

[131] Z. Alexeeva, D. Perez-Palacin, and R. Mirandola, "Design decision documentation: A literature overview," in *ECSA*. Springer, 2016, pp. 84–101.

[132] R. Rauscher, "A design assistant for scheduling of design decisions," in *euromicro*. IEEE, 1996, p. 0088.

[133] L. Borrmann and F. N. Paulisch, "Software architecture at siemens: The challenges, our approaches, and some open issues," in *Software Architecture*. Springer, 1999, pp. 529–543.

[134] C. Zannier and F. Maurer, "A qualitative empirical evaluation of design decisions," in *ACM SIGSOFT Software Engineering Notes*, vol. 30, no. 4. ACM, 2005, pp. 1–7.

[135] C. Zannier and F. Maurer, "Foundations of agile decision making from agile mentors and developers," in *XP*. Springer, 2006, pp. 11–20.

[136] C. Zannier and F. Maurer, "Comparing decision making in agile and non-agile software organizations," in *XP*. Springer, 2007, pp. 1–8.

[137] C. Zannier, M. Chiasson, and F. Maurer, "A model of design decision making based on empirical results of interviews with software designers," *Information and Software Technology*, vol. 49, no. 6, pp. 637–653, 2007.

[138] C. Zannier and F. Maurer, "Social factors relevant to capturing design decisions," in *SHARK-ADI*. IEEE Computer Society, 2007, p. 1.

[139] S. T. Hassard, A. Blandford, and A. L. Cox, "Analogies in design decision-making," in *BCS-HCI*. British Computer Society, 2009, pp. 140–148.

[140] A. Tang and H. van Vliet, "Software designers satisfice," in *ECSA*. Springer, 2015, pp. 105–120.

[141] A. Tang, M. H. Tran, J. Han, and H. Van Vliet, "Design reasoning improves software design quality," in *QoSA*. Springer, 2008, pp. 28–42.

[142] A. Tang, A. Aleti, J. Burge, and H. van Vliet, "What makes software design effective?" *Design Studies*, vol. 31, no. 6, pp. 614–640, 2010.

[143] C. Yang, P. Liang, P. Avgeriou, U. Eliasson, R. Heldal, and P. Pelliccione, "Architectural assumptions and their management in industry–an exploratory study," in *ECSA*. Springer, 2017, pp. 191–207.

[144] Z. Xiong, P. Liang, C. Yang, and T. Liu, "Assumptions in oss development: An exploratory study through the hibernate developer mailing list," in *APSEC*. IEEE, 2018, pp. 455–464.

[145] I. Groher and R. Weinreich, "A study on architectural decision-making in context," in *WICSA*. IEEE, 2015, pp. 11–20.

[146] R. Weinreich, I. Groher, and C. Miesbauer, "An expert survey on kinds, influence factors and documentation of design decisions in practice," *Future Generation Computer Systems*, vol. 47, pp. 145–160, 2015.

[147] R. J. Wirfs-Brock, "Giving design advice," *IEEE Software*, vol. 24, no. 4, 2007.

[148] K. Power and R. Wirfs-Brock, "Understanding architecture decisions in context," in *ECSA*. Springer, 2018, pp. 284–299.

[149] A. Zalewski, K. Borowa, and A. Ratkowski, "On cognitive biases in architecture decision making," in *ECSA*. Springer, 2017, pp. 123–137.

[150] H. van Vliet and A. Tang, "Decision making in software architecture," *Journal of Systems and Software*, vol. 117, pp. 638–644, 2016.

[151] A. Manjunath, M. Bhat, K. Shumaiev, A. Biesdorf, and F. Matthes, "Decision making and cognitive biases in designing software architectures," in *ICSA-C*. IEEE, 2018, pp. 52–55.

[152] C. Pretorius, M. Razavian, K. Eling, and F. Langerak, "Towards a dual processing perspective of software architecture decision making," in *ICSA-C*. IEEE, 2018, pp. 48–51.

[153] A. Tang, "Software designers, are you biased?" in *SHARK*, 2011, pp. 1–8.

[154] J. Rueckert, A. Burger, H. Koziolek, T. Sivanthi, A. Moga, and C. Franke, "Architectural decision forces at work: experiences in an industrial consultancy setting," in *ESEC-FSE*. ACM, 2019, pp. 996–1005.

[155] U. Van Heesch, P. Avgeriou, and R. Hilliard, "Forces on architecture decisions-a viewpoint," in *WISCA*. IEEE, 2012, pp. 101–110.

[156] J. S. van der Ven and J. Bosch, "Busting software architecture beliefs: A survey on success factors in architecture decision making," in *SEAA*. IEEE, 2016, pp. 42–49.

[157] P. Ralph and E. Tempero, "Characteristics of decision-making during coding," in *EASE*. ACM, 2016, p. 34.

[158] J. Hutchinson and G. Kotonya, "A review of negotiation techniques in component based software engineering," in *SEAA*. IEEE, 2006, pp. 152–159.

[159] M. Anvaari, R. Conradi, and L. Jaccheri, "Architectural decision-making in enterprises: preliminary findings from an exploratory study in norwegian electricity industry," in *ECSA*. Springer, 2013, pp. 162–175.

[160] S. Dasanayake, J. Markkula, S. Aaramaa, and M. Oivo, "Software architecture decision-making practices and challenges: an industrial case study," in *ASEC*. IEEE, 2015, pp. 88–97.

[161] D. Falessi, G. Cantone, R. Kazman, and P. Kruchten, "Decision-making techniques for software architecture design: A comparative survey," *CSUR*, vol. 43, no. 4, pp. 1–28, 2011.

[162] H. Christiaans and R. A. Almendra, "Accessing decision-making in software design," *Design Studies*, vol. 31, no. 6, pp. 641–662, 2010.

[163] E. Papatheocharous, K. Petersen, A. Cicchetti, S. Sentilles, S. M. A. Shah, and T. Gorschek, "Decision support for choosing architectural assets in the development of software-intensive systems: The grade taxonomy," in *ECSAW*. ACM, 2015, p. 48.

[164] C. Wohlin, K. Wnuk, D. Smite, U. Franke, D. Badampudi, and A. Cicchetti, "Supporting strategic decision-making for selection of software assets," in *ICSOB*. Springer, 2016, pp. 1–15.

[165] A. Cicchetti, M. Borg, S. Sentilles, K. Wnuk, J. Carlson, and E. Papatheocharous, "Towards software assets origin selection supported by a knowledge repository," in *MARCH*. IEEE, 2016, pp. 22–29.

[166] B. Xu, Z. Huang, and O. Wei, "Making architectural decisions based on requirements: Analysis and combination of risk-based and quality attribute-based methods," in *UIC-ATC*. IEEE, 2010, pp. 392–397.

[167] G. Márquez and H. Astudillo, "Selecting components assemblies from non-functional requirements through tactics and scenarios," in *SCCC*. IEEE, 2016, pp. 1–11.

[168] N. B. Harrison, E. Gubler, and D. Skinner, "Architectural decision-making in open-source systems–preliminary observations," in *MARCH*. IEEE, 2016, pp. 16–21.

[169] F. Gilson, M. Galster, and F. Georis, "Extracting quality attributes from user stories for early architecture decision making," in *ICSA-C*. IEEE, 2019, pp. 129–136.

[170] S. Blair, R. Watt, and T. Cull, "Responsibility-driven architecture," *IEEE software*, vol. 27, no. 2, 2010.

[171] F. Heijenk, M. van den Berg, H. Leopold, H. van Vliet, and R. Slot, "Empirical insights into the evolving role of architects in decision-making in an agile context," in *ECSA*. Springer, 2018, pp. 247–264.

[172] X. Cui, Y. Sun, S. Xiao, and H. Mei, "Architecture design for the large-scale software-intensive systems: A decision-oriented approach and the experience," in *ICECCS*. IEEE, 2009, pp. 30–39.

[173] I. Lytra, S. Sobernig, and U. Zdun, "Architectural decision making for service-based platform integration: A qualitative multi-method study," in *WICSA-ECSA*. IEEE, 2012, pp. 111–120.

[174] G. Pedraza-Garcia, H. Astudillo, and D. Correal, "Modeling software architecture process with a decision-making approach," in *SCCC*. IEEE, 2014, pp. 1–6.

[175] A. Dragomir, H. Lichter, and T. Budau, "Systematic architectural decision management, a process-based approach," in *WICSA*. IEEE, 2014, pp. 255–258.

[176] C. Schriek, J. M. E. van der Werf, A. Tang, and F. Bex, "Software architecture design reasoning: A card game to help novice designers," in *ECSA*. Springer, 2016, pp. 22–38.

[177] R. C. De Boer, P. Lago, R. Verdecchia, and P. Kruchten, "Decidarch v2: An improved game to teach architecture design decision making," in *ICSA-C*. IEEE, 2019, pp. 153–157.

[178] M. Razavian, A. Tang, R. Capilla, and P. Lago, "In two minds: how reflections influence software design thinking," *Journal of Software: Evolution and Process*, vol. 28, no. 6, pp. 394–426, 2016.

[179] G. Pedraza-Garcia, H. Astudillo, and D. Correal, "Analysis of design meetings for understanding software architecture decisions," in *CLEI*. IEEE, 2014, pp. 1–10.

[180] G. Pedraza-García, H. Astudillo, and D. Correal, "Dvia: Understanding how software architects make decisions in design meetings," in *ECSAW*. ACM, 2015, p. 51.

[181] G. Pedraza-Garcia, H. Astudillo, and D. Correal, "An approach for software knowledge sharing based on architectural decisions," in *CLEI*. IEEE, 2016, pp. 1–10.

[182] J. M. E. van der Werf, R. de Feijter, F. Bex, and S. Brinkkemper, "Facilitating collaborative decision making with the software architecture video wall," in *ICSAW*. IEEE, 2017, pp. 137–140.

[183] M. Razavian, A. Tang, R. Capilla, and P. Lago, "Reflective approach for software design decision making," in *QRASA*. IEEE, 2016, p. 19.

[184] C. Pretorius, "Beyond reason: Uniting intuition and rationality in software architecture decision making," in *ICSA-C*. IEEE, 2019, pp. 275–282.

[185] D. Tofan, M. Galster, and P. Avgeriou, "Difficulty of architectural decisions–a survey with professional architects," in *ECSA*. Springer, 2013, pp. 192–199.

[186] A. Harchenko, I. Bodnarchuk, I. Halay, and V. Yatcyshyn, "The tool for design of software systems architecture," in *CADSM*. IEEE, 2013, pp. 138–139.

[187] A. Harchenko, I. Bodnarchuk, and I. Halay, "Decision support system of software architect," in *IDAACS*, vol. 1. IEEE, 2013, pp. 265–269.

[188] J. W. Cangussu, K. C. Cooper, and E. W. Wong, "Multi criteria selection of components using the analytic hierarchy process," in *CBSE*. Springer, 2006, pp. 67–81.

[189] N. Ernst, J. Klein, G. Mathew, and T. Menzies, "Using stakeholder preferences to make better architecture decisions," in *ICSAW*. IEEE, 2017, pp. 133–136.

[190] S. Orlov and A. Vishnyakov, "Decision making for the software architecture structure based on the criteria importance theory," *Procedia computer science*, vol. 104, pp. 27–34, 2017.

[191] A. Egyed and D. S. Wile, "Support for managing design-time decisions," *TSE*, no. 5, pp. 299–314, 2006.

[192] T. Al-Naeem, F. T. Dabous, F. A. Rabhi, and B. Benatallah, "Formulating the architectural design of enterprise applications as a search problem," in *ASWEC*. IEEE, 2005, pp. 282–291.

[193] M. Makki, E. Bagheri, and A. A. Ghorbani, "Automating architecture trade-off decision making through a complex multi-attribute decision process," in *ECSA*. Springer, 2008, pp. 264–272.

[194] F. H. Jabali, S. M. Sharafi, and K. Zamanifar, "A quantitative algorithm to select software architecture by tradeoff between quality attributes," *Procedia computer science*, vol. 3, pp. 1480–1484, 2011.

[195] M. Riebisch and S. Wohlfarth, "Introducing impact analysis for architectural decisions," in *ECBS*. IEEE, 2007, pp. 381–392.

[196] L. Grunske, "Identifying good architectural design alternatives with multi-objective optimization strategies," in *ICSE*. ACM, 2006, pp. 849–852.

[197] S. A. Busari, "Towards search-based modelling and analysis of requirements and architecture decisions," in *ASE*. IEEE, 2017, p. 1026.

[198] S. Gerdes, M. Soliman, and M. Riebisch, "Decision buddy: tool support for constraint-based design decisions during system evolution," in *FoSADA*. ACM, 2015, pp. 13–18.

[199] S. Moaven, J. Habibi, H. Ahmadi, and A. Kamandi, "A fuzzy model for solving architecture styles selection multi-criteria problem," in *EMS*. IEEE, 2008, pp. 388–393.

[200] S. Moaven and J. Habibi and H. Ahmadi and A. Kamandi, "A decision support system for software architecture-style selection," in *SERA*. IEEE, 2008, pp. 213–220.

[201] G. Zayaraz, S. Vijayalakshmi, and V. Vijayalakshmi, "Evaluation of software architectures using multicriteria fuzzy decision making technique," in *IAMA*. IEEE, 2009, pp. 1–5.

[202] N. Esfahani, S. Malek, and K. Razavi, "Guidearch: guiding the exploration of architectural solution space under uncertainty," in *ICSE*. IEEE Press, 2013, pp. 43–52.

[203] N. Upadhyay, "Sdmf: Systematic decision-making framework for evaluation of software architecture," *Procedia computer science*, vol. 91, pp. 599–608, 2016.

[204] M. A. Al Imran, S. P. Lee, and M. M. Ahsan, "Quality driven architectural solutions selection approach through measuring impact factors," in *ICECOS*. IEEE, 2017, pp. 131–136.

[205] J. A. Díaz-Pace and M. R. Campo, "Exploring alternative software architecture designs: a planning perspective," *IEEE Intelligent Systems*, vol. 23, no. 5, 2008.

[206] M. Scheerer, A. Busch, and A. Koziolek, "Automatic evaluation of complex design decisions in component-based software architectures," in *MEMOCODE*. ACM, 2017, pp. 67–76.

[207] G. Sapienza, G. Dodig-Crnkovic, and I. Crnkovic, "Inclusion of ethical aspects in multi-criteria decision analysis," in *MARCH*. IEEE, 2016, pp. 1–8.

[208] A. Shahbazian, Y. K. Lee, Y. Brun, and N. Medvidovic, "Making well-informed software design decisions," in *ICSE-C*. ACM, 2018, pp. 262–263.

[209] A. Alali and J. Sillito, "Motivations for collaboration in software design decision making," in *CHASE*. IEEE, 2013, pp. 129–132.

[210] S. Dasanayake, J. Markkula, S. Aaramaa, and M. Oivo, "An empirical study on collaborative architecture decision making in software teams," in *ECSA*. Springer, 2016, pp. 238–246.

[211] J. Chai and J. N. Liu, "An ontology-driven framework for supporting complex decision process," in *WAC*. IEEE, 2010, pp. 1–6.

[212] M. Nowak and C. Pautasso, "Team situational awareness and architectural decision making with the software architecture warehouse," in *ECSA*. Springer, 2013, pp. 146–161.

[213] D. Tofan, M. Galster, I. Lytra, P. Avgeriou, U. Zdun, M.-A. Fouche, R. De Boer, and F. Solms, "Empirical evaluation of a process to increase consensus in group architectural decision making," *IST*, vol. 72, pp. 31–47, 2016.

[214] S. V. F. Lopes and P. T. A. Junior, "Architectural design group decision-making in agile projects," in *ICSAW*. IEEE, 2017, pp. 210–215.

[215] V. S. Rekhav and H. Muccini, "A study on group decision-making in software architecture," in *WICSA*. IEEE, 2014, pp. 185–194.

[216] S. Rekha and H. Muccini, "Suitability of software architecture decision making methods for group decisions," in *ECSA*. Springer, 2014, p. 17.

[217] I. Malavolta, H. Muccini, and S. Rekha, "Enhancing architecture design decisions evolution with group decision making principles," in *SERENE*. Springer, 2014, pp. 9–23.

[218] H. Muccini, D. A. Tamburri, and V. S. Rekha, "On the social dimensions of architectural decisions," in *ECSA*. Springer, 2015, p. 137.

[219] S. Rekha and H. Muccini, "Group decision-making in software architecture: A study on industrial practices," *IST*, 2018.

[220] K. Shumaiev, M. Bhat, O. Klymenko, A. Biesdorf, U. Hohenstein, and F. Matthes, "Uncertainty expressions in software architecture group decision making: Explorative study," in *ECSA-C*. ACM, 2018, pp. 42:1–42:8.