Noname manuscript No.
(will be inserted by the editor)

Quality Attributes Use in Architecture Design Decision Methods: Research and Practice

Ioanna Lytra · Carlos Carrillo · Rafael Capilla · Uwe Zdun

Received: date / Accepted: date

Abstract Over the past ten years software architecture has been perceived as the result of a set of architecture design decisions rather than the elements that form part of the software design. As quality attributes are considered major drivers of the design process to achieve high quality systems, the design decisions that drive the selection and use of specific quality properties and vice versa are closely related. Consequently, quality attributes must play a role for decision making processes and be documented alongside with the decisions captured. Consequently, we conduct a systematic literature review to study the importance and impact of the relationships between quality attributes and architecture design decisions and to what extent existing architecture knowledge management methods and tools deal with the decisions that affect the quality of a system. We also report on the challenges and future research paths for architectural knowledge management methods and tools. Our results reveal important explicit relationships between both software artifacts, the role of uncertainty in decision making and empirical studies reporting the use of quality attributes in architecture knowledge management activities.

Ioanna Lytra
Software Architecture Group, University of Vienna, Austria
E-mail: ioanna.lytra@univie.ac.at

Carlos Carrillo
Department of Telematic and Electronic, Polytechnic University of Madrid, Spain
E-mail: carlos.carrillo@upm.es

Rafael Capilla
Department of Computer Science, Rey Juan Carlos University, Spain
E-mail: rafael.capilla@urjc.es

Uwe Zdun
Software Architecture Group, University of Vienna, Austria
E-mail: uwe.zdun@univie.ac.at

Keywords Quality Attributes · Architecture Design Decisions · Architectural Knowledge · Systematic Literature Review

1 Introduction

Since the early 2000s, the software architecture community perceives design decisions as first-class artifacts that should be captured alongside the standard software architecture documentation [23]. Today, many of the existing decision making approaches demand the capture and documentation of multiple alternatives that have to be evaluated during the development and evolution of the system. However, as making decisions often implies that competing requirements must be satisfied for different stakeholders' concerns, the evaluation of quality attributes (QAs) [8] in the architecture may trigger additional decisions that must also be evaluated. For many years, QAs [9] have been used in the architecture to describe the non-functional properties of systems and are primarily addressed in the early phases of the design activity. Although software architects make design decisions to depict the major functional parts in the design, we need to understand the impact of quality attributes in the architecture as well as important decision drivers [30, 54, 55].

Some authors [21, 47, 51] also highlight the role of design patterns, as a kind of architectural knowledge or proven design decisions, to address the quality requirements of a system. Although some of the existing architecture decision methods consider the role of the QAs implicitly or explicitly, satisfying the various trade-offs of QAs during any decision making process as well as the interplay of decisions and quality properties still remains a complex and challenging task.

Existing architecture design decision methods and tools have already been analyzed in other works but with a different focus and aim. For instance, Capilla et al. [13] provide an informal retrospective analysis of AK management research while Falessi et al. [19] examine various techniques for selecting design alternatives during decision making. Tofan et al. [46] conducted a systematic mapping study as an overview of existing architecture decision documentation approaches in which the authors discuss functional and non-functional requirements in the context of architecture decisions, but they do not analyze in detail the relationships between quality attributes and architectural decisions.

Although previous studies discuss the relationship between architectural design decisions (ADDs) and quality attributes (QAs), they do not provide a full perspective discussing dimensions such as uncertainty, dependency types between decisions, or QA trade-offs, and do not analyze in depth such characteristics we do in our study. Therefore, apart from analyzing different dimensions and challenges in the overlap of ADDs and QAs, we provide a deeper analysis classifying each selected paper according to different criteria.

To the best of our knowledge, the use and integration of QAs in existing AK management (AKM) methods and tools have been under-investigated so

far. Consequently, our main contribution in this research is the investigation of the role of QAs in existing AK approaches as well as the support of QAs in existing architecture decision methods and tools. We also provide an integrated view of the relationships between QAs and AK based on the three aforementioned dimensions and supported by a deeper analysis of approaches that report industry case studies.

The paper is structured as follows. In Section 2, we clarify the two key terms relevant to our SLR, architecture design decision and quality attribute.In Section 3 we present the research method we have applied to extract the relevant literature. In particular, we introduce the rationale of our literature review, discuss our search strategy and process as well as the selection criteria we applied, and we state the three research questions under investigation. We go into the details of the research's results in Section 4-6. Section 7 discusses our findings and future research directions, as well as the limitations of our study, and, finally, in Section 8, we draw the conclusions of our work.

2 Terminology and Conceptual Model

Architecture Design Decision (ADD): According to ISO/IEC/ 42010:2011 [2], an ADD affects various architectural elements, pertains to or raises concerns, and is justified by architecture rationale. Ven et al. [49] understand design decisions as a first-class artifact that couples rationale with software architecture. Approaches using templates for capturing ADDs [48] and are supported by existing meta-models [54] and tools [40, 44, 46] set the focus on architecture knowledge management.

Quality Requirement, quality attribute and quality property: According to ISO/IEC/IEEE 25010:2011 [1] a **quality requirement (QR)** is defined as "*a requirement that a software quality attribute be present in software*," while a **quality property (QP)** is understood as a "*measurable component of quality.*" In addition, **quality attribute (QA)** is defined by [9] as "*a measurable or testable property of a system that is used to indicate how well the system satisfies the needs of its stakeholders.*"

QAs are orthogonal to functionality [9] and the choice of a functionality in a design decision does not dictate the level of quality of such a functionality. Therefore, wheter ADDs are driving the selection of the qualities of the system or the other way round, QAs are seen as major decision drivers [9, 30, 54, 55].

In order to define a conceptual relationship between QAs and ADDs, we describe the following conceptual model depicted in Fig. 1, which illustrates the main entities used to create the significant design decisions that require quality properties as decision drivers or even incomplete knowledge to make a decision. The decisions can be used to select the most suitable qualities of interest for a particular system (e.g. security and availability are major concerns for banking systems) or the other way around, where quality attributes drive the selection of a particular decision (e.g. select a middleware with high performance). As

decisions are connected to other decisions, it is common that all the decisions made form a decisions network, and software architects can use this network to evaluate the impact of decision changes during evolution cycles. Finally, the set of decisions made should lead to a solution architecture that fulfills the requirements and constraints that motivated the decisions taken.

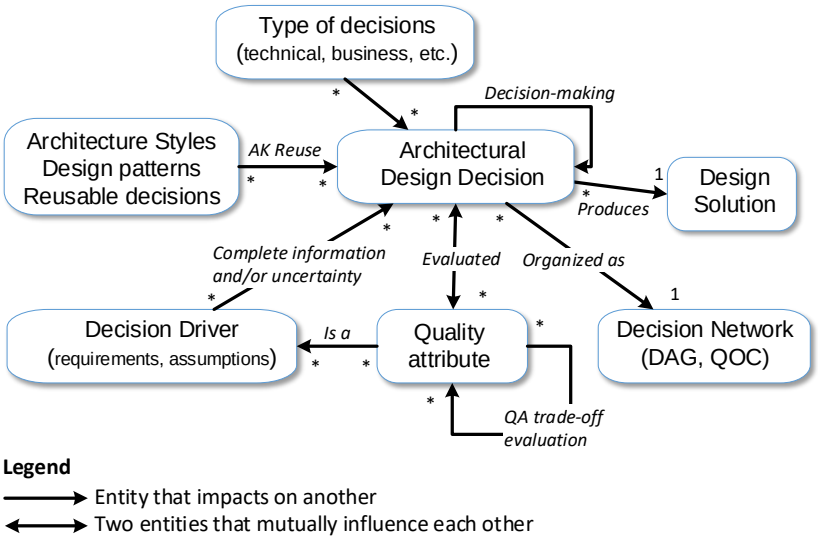

Fig. 1: Conceptual model showing the main concepts and relationships between design decisions and quality attributes. Decision networks can adopt different topologies like DAG (Directed Acyclic Graphs) or QOC (Question-Option-Criteria) models.

3 Review Method

Following the guidelines provided by Kitchenham et al. [26] we run a systematic review to investigate in depth the relationships and role of QAs and ADDs. According to Kitchenham et al. [25], systematic reviews comprise the following three phases: (1) planning i.e. stating the rationale of the review and editing of the review protocol, (2) execution i.e. conduction of the review according to the review protocol, and (3) reporting i.e. presentation of the review results. As we found that the topic relating QAs and ADDs in AKM approaches has been poorly treated, we perceived the need to analyze which research works better highlight the role of QAs and their relationships in existing AKM approaches, tools, and case studies in order to derive meaningful findings. We summarize the main objectives of our study below:

1. Review and better understand the relationships of QAs and ADDs.

2. Review and better understand the focus of existing AKM methods and tools, especially with regard to the challenges they address.
3. Find evidence about how the relationships between QAs and ADDs are modeled and documented.
4. Review and better understand implications for researchers and practitioners interested in AKM in the current state of the art.

3.1 Research Questions

In the context of the literature review, based on the aforementioned objectives, we considered the following research questions:

RQ1 *What is the role and use of QAs and their relationships to ADDs in existing AKM methods and tools?*
RQ2 *Which challenges related to managing QAs are frequently addressed by existing AKM methods and tools?*
RQ3 *What is the size and scope of industry case studies of existing AKM methods and tools with regard to QAs?*

In order to organize the results of the research questions, we describe the concepts and research questions in Fig. 2 to guide the reader through the results. For research question RQ1, we are interested in whether the QAs appear in the decision making process, in the documentation of the design decisions, or both. We also model different types of relationship between the decisions and the quality attributes (e.g. explicit, not explicit, etc.). The evaluation of ADDs using QAs can be supported in different ways such as the ones indicated in the values of the figure. Regarding research question RQ2, we first look for whether there is uncertainty using the QAs in existing architecture design decision methods. In addition, we searched for different types of dependency between the QAs and whether these appear explicitly as well as the trade-offs between QAs. Finally, research question RQ3 explores the size of the design space and scope of the different approaches using QAs and ADDs, and the evaluation method used. In the figure, We also explicitly model some interdependencies between different concepts belonging to the research questions. These are represented by solid arrows in Fig. 2. For instance, QA uncertainty from RQ2 has a clear impact on the evaluation of ADDs using QAs (cf., Research Question RQ1), as every time we evaluate a design decision we need to know if the qualities exhibit incomplete information.

3.2 Search Strategy

To conduct the systematic review, we queried four digital publication libraries: 1) the Association for Computing Machinery (ACM) Guide to Computing

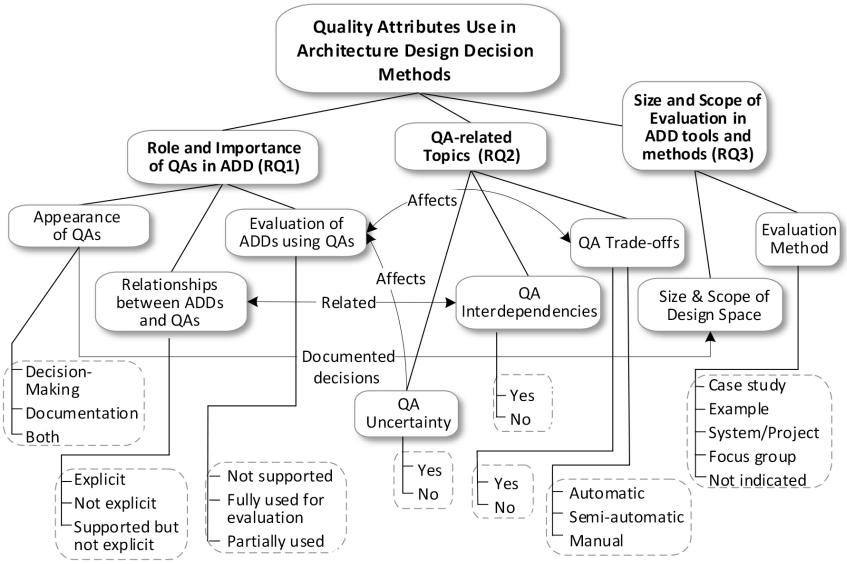

Fig. 2: Relationships between the main concepts and research questions that organize the results shown in the tables in Section 4-6

Literature[1], 2) the IEEE Xplore Digital Library[2], 3) DBLP[3], and 4) Springer[4]. The search string we used entails the most appropriate keywords related to the scope of this SLR, namely the terms "quality attributes", "non-functional requirements", "design decisions", and so on. The query we performed in the four digital libraries can be expressed as a Boolean formula:

> software architecture AND (quality attributes OR non-functional requirements OR nfr OR design decisions OR architecture knowledge OR architectural knowledge)

The queries performed on the libraries refer only to the titles, as additionally searching abstracts and content would produce an enormous amount of matches, unmanageable to be inspected manually. However, as Springer does not allow the filtering of only the titles, in this case we searched the overall contents of the articles. As our search strategy might miss useful citations, we used backward and forward snowballing [50] to identify additional candidate citations. In backward snowballing, the references of the selected articles are checked for useful publications, while in forward snowballing the publications that cite the selected articles are examined. In both cases, snowballing finishes

[1] http://dl.acm.org/
[2] http://ieeexplore.ieee.org/
[3] http://dblp.uni-trier.de/
[4] http://link.springer.com/

when a convergence is reached and no new articles can be found. Because different digital databases provide different search facilities, it was not possible to use one single search string in all the cases to identify all the relevant sources. We summarize in Table 1 the different queries we used in each digital library.

Digital Library	**Search String**
ACM DL	+("software architecture") + ("quality attributes" "non-functional requirements" "nfr" "design decisions" "architecture knowledge" "architectural knowledge")
IEEEXplore	("software architecture") AND ("quality attributes" OR "non-functional requirements" OR "nfr" OR "design decisions" OR "architecture knowledge" OR "architectural knowledge")
DBLP	architectur* quality|nfr|decision*|functional$
Springer	("software architecture") AND ("quality attributes" OR "non-functional requirements" OR "nfr" OR "design decisions" OR "architecture knowledge" OR "architectural knowledge")

Table 1: Search strings against four digital libraries

3.3 Selection Criteria

In order to extract the publications to be used as basis for answering our three research questions, we defined appropriate selection criteria. Inclusion/exclusion criteria were used to select publications focusing on methods and tools for ADD support and documentation with a focus on QAs, and to reject papers that focused on other forms of AKM or did not refer explicitly to ADDs or QAs. Therefore, the inclusion and exclusion criteria are as follows:

Inclusion Criteria (IC)

- **IC1:** The primary study is a peer-reviewed scientific paper that introduces a technique, method, or approach about AKM or ADDs using QAs.
- **IC2:** The primary study reports a case study about the use of QAs in architecture decision making or on an AKM tool.
- **IC3:** The primary study is the most recent version.
- **IC4:** The paper is written in English.

Exclusion Criteria (EC)

- **EC1:** The primary study does not deal with QAs in AKM explicitly or implicitly.
- **EC2:** The primary study does not consider the tandem QA-ADD, even if it belongs to the software architecture field.
- **EC3:** The primary study does not include an evaluation of QAs using ADDs.

EC4: The primary study is an older publication from the same authors about the same approach or a duplicated study.
EC5: The primary study is a short paper with less than five pages.
EC6: The primary study is a non-peer-reviewed publication.

3.4 Search Process and Selected Publications

The results of the queries we performed for the period between 01/2001 and 10/2018 reported the following numbers: ACM DL (1070 papers), IEEE (785 papers), DBLP (1509 papers), and Springer (2032 papers). In the case of the Springer database, we selected only those papers belonging to the software engineering discipline (i.e. "Software Engineering" keyword). In addition, as depicted in Fig. 3, we merged and aggregated the results from the four databases. Once we have removed the duplicates, we came up with 4571 papers. We used the title to filter out the research works according to the inclusion/exclusion criteria which resulted in 73 papers. According to [50], we carried out a snowballing process to find additional papers. Using this technique, we carried out two backward-forward iterations in the proposed four databases, except for Springer where we needed three iterations, and we found only four additional papers. This proves that the selection of the queries was precise enough in order to find the relevant papers.

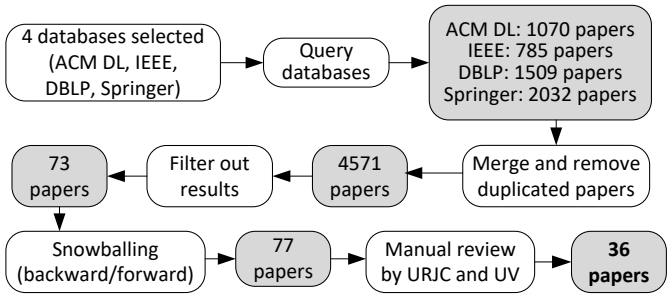

Fig. 3: Literature review steps and outcomes

After the iteration process, we manually filtered out the papers (77 papers) according to the inclusion/exclusion criteria, This was done by two of the paper's authors (Universidad Rey Juan Carlos - URJC), then a second time by the other two authors (University of Vienna - UV). We reviewed the contents of the papers in order to perform an accurate selection regarding those works relating ADDs and QAs. For instance, we did not consider the work of [27] on capturing software design decisions in our selection as the tool they introduce aims at annotating ADDs in software related artifacts rather than on architecture decision making and documentation, and does not consider the use of QAs. Other examples include the work of [42] which on the one hand discusses

the prioritization of QAs for different stakeholders' concerns, but on the other hand, it does not contain any explicit reference to ADDs. Apart from that, we excluded works that referred to the same approaches and/or tools and kept only the most recent or most detailed publications (e.g. journal or conference papers instead of workshop papers). The papers were selected based on careful reading of the abstracts, introduction, and conclusions but we read in depth all the 77 papers selected for manual review to ensure their compliance to the inclusion/exclusion criteria and we finally selected 36 papers. All the papers selected were reviewed by the four authors of this research work. As a result, we consolidated a list of 36 publications (four journal, 28 conference and four workshop publications) and we classified the selected publications into three focus areas according to our research questions.

Table 1 in the Appendix[5] lists the selected papers along with the publication year and place as well as the name of the method or tool used for architecture decision making and documentation. The majority of the publications (28) are from the years 2007–2018, which gives an indication of the increasing interest of the software architecture community in ADDs in recent years. In most of the cases, the proposals are accompanied by tools for software architects.

4 Role and Importance of QAs in ADD Tools and Methods (RQ1)

The importance of QAs in decision making in software architecture [9, 30, 54, 55] and the fact that good decisions affect the selection of which QAs are important for a system during architecture reviews [8] are widely recognized. In order to answer RQ1, in this section we discuss the role of QAs in AK approaches. More specifically, we will investigate at what point in time the QAs appear in the decision making process or are just documented, the explicit and implicit relationships between ADDs and QAs, how decisions are evaluated using QAs, and how QAs are used to decide on the best alternative. In the following subsections, we discuss in particular the selected publications with regard to the following aspects:

1. *Appearance of QAs*: We distinguish between tools that integrate QAs in decision making, in ADD documentation, or in both.
2. *Relationships between ADDs and QAs*: We investigate the relationships between QAs and ADDs and whether these are explicitly or implicitly documented. We distinguish between "not explicit" and "explicit" trace links support and identify those approaches that contain explicit links supported by some kind of evaluation method.
3. *Evaluation of decisions and qualities*: Some approaches use QAs to evaluate the best or the optimal alternative decisions. However, other research works focus on how to make design decisions in order to select one or several quality requirements that are more important for the architecture and hence, fulfill the quality requirements of a system.

[5] The Appendix is available as an online supplement to Springer Web site of this paper.

4.1 Appearance of QAs

It is of key importance to document the QAs that will drive the shape of the architecture and the decisions made during the architecting activity in order to prevent AK vaporization. In the primary studies, we observed three major trends regarding the "Appearance of QAs" (see RQ1). The values of the second column in Table 2 in the Appendix reflect these trends. The first trend is to document QAs in approaches using AK. The second trend is documenting QAs in the decision making process. Finally, the third set of research works document QAs in both documentation and decision making. In the first case, QAs are described explicitly in the documentation alongside the design decisions. This approach is taken by two of the papers examined excluding those that appear in both documentation and decision making. The second trend describes how the QAs are documented and used in the decision making process. Many of the works highlight the importance of QAs in the reasoning activity and how QAs act as major decision drivers (e.g. [5, 6, 31, 55]). In some of these works, QAs play a key role for architectural trade-off decision making that is sometimes carried out using multiple attributes [32] and constraints [5]. Different criteria and methods can be used to evaluate the role of QAs in the decision process based on e.g. using utility trees [6] and recommender systems [29]. Corresponding tool support can assist software architects in the decision making process using QAs for evaluating alternative decisions. A few tools provide automated support for managing the interactions between QAs and decisions such as the ArchiTech tool, an application of the Quark approach [5]. For reusable architectural decision models, CoCoADvISE provides automated support for quality-attribute-based architectural decision making [31]. In addition, Lopes Silva et al. introduce a tool to support the architecture design phase by recommending architectural styles given the quality attributes [29]. Finally, almost 14% of the approaches under study document both the QAs in ADD documentations while at the same time they describe how QAs influence the reasoning activity on ADDs. For instance, the STREAM-ADD approach [16] suggests to explicitly document ADDs alongside NFRs, and to use QAs during architectural refinements in structural and technology decisions. In this approach, NFRs are used with soft-goals to evaluate the set of alternatives using architectural styles and the rationale for them.

4.2 Relationships between ADDs and QAs

Regarding explicit and implicit relationships (or links) between ADDs and QAs, we explored the types of links between decisions and quality attributes. These QAs map to high-quality requirements [10] as an important quality concern a system demands. This is important for the software architecture documentation because it helps software maintainers to identify the trace links between both artifacts when decisions change, and to reevaluate the decisions if QAs are modified. From our results we identified the following types of links

between both artifacts: "not explicit", "explicit", and "supported but not explicit". For the sake of clarity, papers that categorize the relationship between ADDs and QAs as "not explicit" means that there is no direct and clear link described in a meta-model or via examples. Those works that we describe as containing "explicit" links between ADDs and QAs suggest an approach or method where QAs are used to support architecture decision making and in most cases such direct links are represented in a meta-model as well. Finally, works containing "supported but not explicit" relationship provide support for describing and evaluating the links between QAs and ADDs in an implicit way. Not all the research works selected provide explicit trace links between design decisions and quality attributes as these relationships are somehow hidden. In approaches categorized as "not explicit" [21, 35] the authors highlight that there is a relationship between ADDs and QAs but those links are not clearly described or represented. Such links are defined and used by tools like Software Architecture Warehouse (SAW), which handles collaborative decisions with a voting support facility [35]. Nevertheless, the authors did not explicitly document these trace links as the focus of their approach is different, but this does not mean the trace links do not exist in models supporting tools like SAW.

From the approaches that model and define explicit ADD-QA links, we can highlight the one introduced by [15]. In this work, the authors describe explicit connections in a meta-model linking decisions to design artifacts and quality attribute requirements, which helps support an automatic synthesis method of candidate architecture solutions. The decision-centric architecture design process uses the QAs to identify the issues that will lead to the issue solutions. These issue solutions will be used to synthesize the solution architecture from the decisions captured with other software engineering artifacts and the relationships among them. Other approaches [22, 52, 54] provide different but similar ways to relate several types of ADDs with QAs and they document such relationships explicitly with the use of meta-models.

Another way to relate ADDs with QAs is the use of architectural tactics – see for instance, [7, 34] and [24] – or by combining requirements goal models with component diagrams [41]. Finally, other works like CoCoADvISE [31] introduce a reusable architectural decision meta-model which integrates design solutions (i.e. ADDs) with decision drivers (i.e. QAs) and is subsequently used as a basis for generating questionnaires to support architectural decision making. This approach offers some degree of automation for representing the design decisions and QAs of interest, which are evaluated positively or negatively. More recent works like [14] describe explicit support of quality attributes to understand the ripple effect in decision making, but they do not provide explicit relationships between the quality attributes like in [38] or [39].

4.3 Evaluation of decisions and qualities

The final category suggests the following values to classify how QAs and ADDs are evaluated: "not supported", "partially used", and "fully used for evaluation"

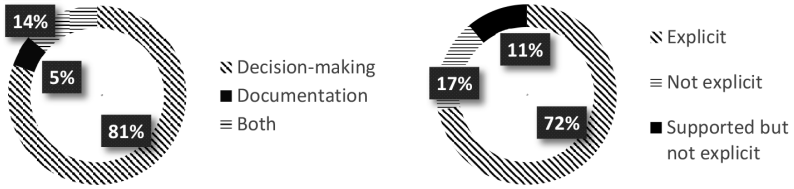

Fig. 4: Appearance of QAs in decision making, documentation, or both (left) and type of relationships between ADDs and QAs (right)

(see Table 2 of the Appendix for the detailed results). The first value (i.e. "not supported") refers to works where we could not find an evaluation of ADDs using QAs or the other way round. The second, "partially used", refers to those works where the evaluation is done only in one direction (e.g. QAs are used to evaluate ADDs) or the process is not described in enough detail. Finally, the last category, "fully used for evaluation", refers to those research works where the authors describe a method, approach, or tool supporting the evaluation and selection of architecture design decisions using quality attributes in both directions (compared to only one in the previous category).

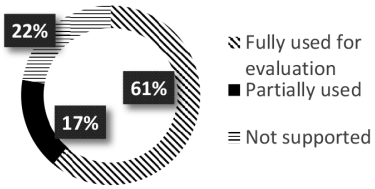

Fig. 5: Evaluation of ADDs using QAs and of QAs using ADDs

Design decisions are used to select which QAs are more suitable to improve the quality of the architecture. Alternatively, decisions can be evaluated based on qualities, and on the pros and cons of each decision. In the first case, we decide on the QAs that must be present in a design. In the second case, the pros and cons are used to select the best or the optimal design decisions (e.g. a higher level of security is needed, so a requirement decision about security needs to be improved and hence, the necessary mechanisms to achieve the desired quality must be added).

For instance, [52] do not indicate how they evaluate decisions using quality attributes and the other way around. In most of these approaches, decisions and QAs are captured, but the evaluation is not that explicit or explained. Some other research works do not support or indicate any of the aforementioned types of evaluations using ADDs and QAs (e.g. [4, 35, 36, 52, 55]).

If we focus on those works rated as "fully used for evaluation", the approach described by [22] uses the notion of forces. The decision forces make the rela-

tionships between design decisions and the factors that influence the decision makers explicit. Among these forces, we can find the quality attributes that are used to select the best decision alternatives in a competing manner. The authors use a table to connect these forces to other architecture views (e.g. view technology) and provide the explicit trace links between the forces and decisions, which are evaluated using a scale (i.e. ?, -, +, ++). We can use the forces to rank the QAs during the selection of different technologies or use the decisions to select the QAs that better suit different business goals.

Other works suggest a multi-attribute decision making approach to make decision trade-offs, evaluate the most suitable QAs implemented in the system [32], and estimate the impact of a decision in software quality [5]. The work presented by [11] highlights the role of design patterns, and their impact on good architecture design decisions and quality attributes is evaluated based on the selection of patterns. Most of the approaches categorized as "fully used for evaluation" put more emphasis on how decisions are selected based on a set of qualities that must be satisfied in a competing manner, while the rationale in the opposite direction (i.e. make decisions to select or reason on the best qualities that must be supported) is rarely seen, even if the trace links between both types of artifacts are explicit. In [33] the authors provide a catalogue of decisions for designing industrial IoT systems as they can be fully used for evaluation of different qualities.

5 QA-related Challenges Addressed by of ADD Tools and Methods (RQ2)

In this section, we discuss the QA-related challenges that are addressed in existing tools and methods for architecture decision making and documentation. In particular, after analyzing the selected articles of the literature survey, we identified the following main challenges that are addressed in several publications which are detailed in Table 3 of the Appendix.

1. *QA Uncertainty*: Uncertainty is caused by vague, incomplete, or imprecise information about QAs of design solutions and requirements. An approach that supports dealing with uncertainty provides means for expressing and/or resolving QA uncertainty.
2. *QA Interdependencies*: A decision on one QA may have an indirect impact on another QA as a consequence of such a decision. Apart from that, prioritization of QAs is often considered in architecture decision making.
3. *QA Trade-offs*: Making ADDs is essentially the result of making trade-offs between competing requirements and stakeholders' concerns. In case QA trade-offs are addressed, we are further interested in whether their resolution is performed manually or semi-automatically by software architects.

5.1 QA uncertainty

Uncertainty is caused, among others, by imprecise and incomplete knowledge, or requirements that make decision making and QA trade-offs difficult. Uncertainty describes a situation in which QAs are not exactly known and can not be precisely quantified, therefore, they are evaluated using a verbal scale. The task of resolving this kind of uncertainty during decision making is on the one hand complex and on the other hand, time-consuming. Even though the inherent uncertainty of QAs is explicit in a few cases, the resolution of this kind of uncertainty is not supported by the majority of the existing tools and methods. For instance, Zdun expresses uncertainty of quality goals using approximated scores (++ for very positive influence, + for positive influence, o for no influence, - for a negative influence, and -- for a very negative influence expected) [53]. Similarly, in our previous work [31] we described such relationships at a meta-model level for reusable architectural decision models in order to express a positive or negative impact of design solutions on QAs. Bode and Riebisch [11] use a point system to express the impact of architectural styles on quality properties with a scale going from -2 (strong negative) to 2 (strong positive), very similar to the verbal scale in [53]. The work of [39] provides a method to estimate uncertain parameters which are unknown during architecture design. We observe that a large majority of the 10 approaches under analysis which discuss uncertainty of QAs use either a verbal scale or a point system to express this uncertainty. Nonetheless, except for one paper (i.e. [37]) the works under study do not tackle resolving uncertainty during decision making. However, previous works have studied decision making in the architectural solution space under uncertainty by considering probability distributions of the parameters of an architecture model [28] or fuzzy values for the alternative solution properties [18].

5.2 QA interdependencies

The interactions between QAs are widely recognized in software architecture evaluation and a QA may have a *positive or negative impact* on other QAs [17]. For instance, in a particular system context, system security might come at the cost of availability, whereas in other system contexts availability is a subgoal of security and thus the two are associated positively [20]. In [31], QA interdependencies are described in a meta-model level, so a QA can be in *synergy with* or in *contradiction with* another QA. In another approach, the authors define subsets of QAs in the form of utility trees by distinguishing between quality attributes and quality factors [6] to establish links between the QAs and design decisions, while [11] also introduces a hierarchy of subcharacteristics and properties related to the quality goal "Evolvability" of software projects. Lopes Silva et al. express potential relationships between QAs, which can be conflictive (-), cooperative (+), or neutral (0) [29].

Another factor that may complicate QA evaluation is the resolution of priorities among the different qualities, as stated in [43]. For instance, preferences on QAs are given by comparing them pairwise and giving quantitative weights in the quality-driven approach for decision making by [3]. Also, [12] use prioritization of QAs using an ordinal scale to rank the design solutions with respect to the set of QAs of interest, and they define four types of criteria (i.e. ontocriteria, anticriteria, diacriteria, and pericriteria) based on classes of architectural design decisions. In the Quark approach described in [5], prioritization of QAs can be used by a decision inference system to provide a prioritized list of ADDs. Fig. 6 shows the percentage of the approaches under study which support uncertainty and QA interdependencies.

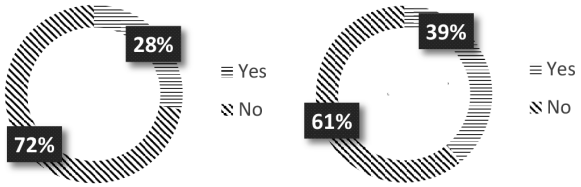

Fig. 6: Approaches supporting uncertainty (left) and interdependencies (right) of QAs

5.3 QA trade-offs

In our analysis we observed that 21 of the 36 approaches under study support quality attribute evaluation trade-offs during decision making. Fig. 7 shows the percentage of the selected approaches supporting QA trade-offs as well as the different types of trade-off automation.

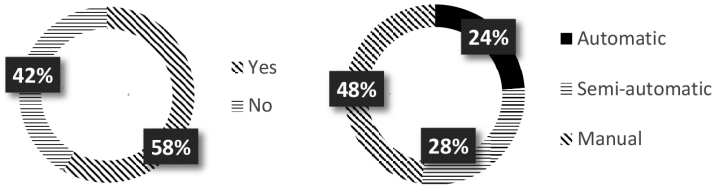

Fig. 7: Approaches supporting QA trade-offs (left) and different levels of trade-off automation (right)

While the largest number of the approaches support manual QA trade-offs (10 out of 21), only five approaches described an automatic process. In other works, the level of automation is not indicated. The ADD+ method [32]

supports automatic trade-offs between conflicting and incomplete quality scenarios and various stakeholders' concerns and preferences as a multi-attribute decision problem, in which the attributes represent the degree of satisfaction of conflicting quality scenarios. ArchDesigner considers making trade-offs a multi-attribute decision making problem and leverages the Analytic Hierarchy Process (AHP) to calculate value scores for design alternatives given their relative impact on QAs and relative stakeholders' preferences [3], while in [37] the authors perform multi-criteria decision analysis. The tool introduced by de Boer et al. can be used for automated trade-off analysis to rank the alternative solutions according to certain QA priorities [12].

Currently, many approaches integrate trade-offs in the architecture decision making process where quality scenarios are used to assist in making trade-offs, such as described in [6]. In addition, the ArchPad (RADM) method of [55] provides reusable pattern-based decision models, which entail the information for resolving requirements at different levels [54]. Also, [15] propose a method for assisting in the selection of and reasoning on architectural solutions, so architects can summarize the advantages and disadvantages of each architecture solution in order to make trade-offs. The STREAM-ADD approach supports manual trade-off analysis of alternatives considering the fulfillment and the priorities of softgoals and NFRs [16]. In other cases, like in the Software Architecture Warehouse (SAW tool) approach, trade-offs are made in a collaborative context [35] and stakeholders discuss the advantages and disadvantages of each design alternative until they reach a consensus. More sophisticated approaches use constraint satisfaction algorithms [5] and expert systems [29] to reason about QA trade-offs. Finally, the authors in [34] suggest two negotiation strategies using agents to support trade-off analysis. Fig. 8 summarizes the different methods used for making QA trade-offs based on the analysis of the 21 aforementioned approaches.

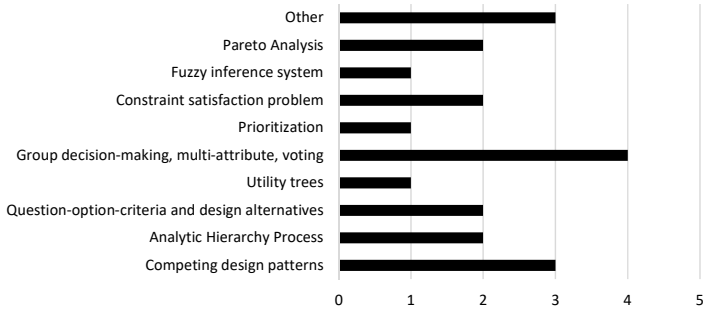

Fig. 8: Methods for making QA trade-offs and number of approaches employing these methods

6 Size and Scope of industry case studies in ADD tools and methods (RQ3)

The demonstration and evaluation of the approaches in real-life-scenarios and empirical studies the authors investigated validate many of the proposals, and they guide practitioners who need to integrate QAs in architecture decision making processes. The rationale for including the design space size and scope in this research relies on the importance for each industrial and academic case study about the decision capturing effort, dependencies between decisions and other software artifacts, type and nature of the decisions, number of alternatives considered, and the QAs used. Fig. 9 reports the size of several industry cases in terms of scenarios and design issues, architectural patterns and tactics used as well as the size of the decisions network.

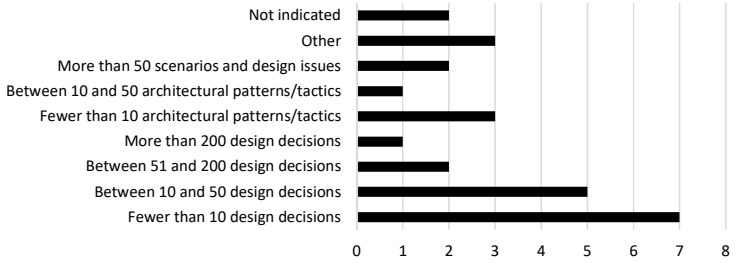

Fig. 9: Number of approaches with respect to the size and scope of reported industry cases studies

More specifically, a portion of the proposals uses a small number of decisions (<10), except in the case of [54, 55] where a big design space consisting of 300 reusable SOA-related decisions has been created. While the approach has been used in industrial projects, the validation of the corresponding QAs was not the focus of the validation in reusable ADDs. 19 proposals report a case study, six a motivating example and five of them a real project or system, such as the ones described in [43, 45, 54]. In other evaluation studies, industrial experts have participated in case studies (e.g. in [11, 31]).

In addition, few approaches (such as [21]) do not perform any kind of evaluation or they just document types of decisions ranging from those based on concrete scenarios to structural, behavioral, and technology decisions [16]. Also, the Software Architecture Warehouse [35] seems to be validated in various design workshops with students but this is poorly reported. The majority of the case studies report at least three QAs while only six publications do not report which QAs were used. The frequencies of the QAs used in the papers under study are shown in Table 2, as an indicator of the most relevant QAs used in decision making activities.

Quality Attribute	#	Quality Attribute	#	Quality Attribute	#
Performance	18	Testability	3	Evolvability	1
Security	12	Accuracy	2	Fault Tolerance	1
Cost	9	Flexibility	2	Functional Suitability	1
Reliability	8	Analyzability	1	Installability	1
Usability	8	Backupability	1	Latency	1
Maintainability	8	Capacity	1	Privacy	1
Modifiability	6	Completeness	1	Recoverability	1
Portability	5	Complexity	1	Reusability	1
Scalability	5	Compliance to standards	1	Traceability	1
Interoperability	4	Concurrency	1	Understandability	1
Availability	3	Data Completeness	1	Effort	1
Efficiency	3	Extensibility	1		

Table 2: Frequency of appearance of QAs

Regarding the types of studies, we can conclude that the majority of them have introduced a technique or a tool in their case studies and examples. While 42% report a case study, 25% of them are only examples but in the context of industrial projects. Nevertheless, in the majority of the studies, the design space size and scope, as well as the number of QAs that are systematically studied are rather low.

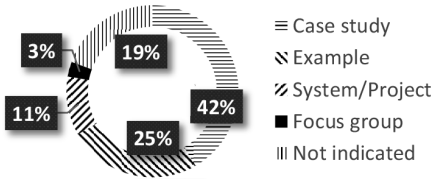

Fig. 10: Distribution of evaluation methods reported in analyzed ADD tools and methods

7 Discussion

In this literature review, we have studied the research works on QAs in ADD methods and tools over the past 17 years. Most of the papers examined investigate the role of QAs in architecture decision making approaches and how design decisions are used to select the most optimal QAs during architecture evaluation and design. The results also revealed that in the majority of cases, QAs are documented or used in a decision making process. Other findings show that a large majority of studies describe fully explicit relationships between QAs and ADDs, which highlights the importance of such trace links for documentation and knowledge capturing. With respect to the evaluation

of QAs and ADDs, about one fifth of the studies do not focus on or describe such an evaluation. This fact leads us to assume the difficulty of carrying out such an evaluation activity. The QA-related challenges used in existing ADD methods show that 58% of the research works deal with QA trade-offs, which is close to the number of works using QAs to evaluate the design decisions. However, only 28% and 39% of the research works address QA uncertainty and QA interdependencies respectively, an indication that these research areas are still immature. There is only one publication covering all three QA-related challenges (i.e. [38]). Another interesting aspect is the automation level of approaches using QAs: the results reported in half of the approaches show that 10 of them are "manual" and six use semi-automatic processes to calculate the priority of the QAs for decision making and trade-off evaluation. The findings have revealed that there is a mature set of baseline approaches for supporting QAs in AK methods and tools. However, it seems to be difficult to achieve more sophisticated automation, e.g. in QA-based architectural decision making.

7.1 Implications for researchers and practitioners

The size and scope of the industrial use cases analyzed show an early adoption of methods and tools that support the interplay between ADDs and QAs, but the maturity of the approaches and their adoption are in an early adoption phase. Furthermore, the traditional QA trade-off evaluation in software architecture can benefit from approaches relating the decisions with the qualities of the system, so that software architects can better understand the underpinning reasons for selecting the best quality properties of a system as well as how the selection of a quality property influences the decisions for technology selection, and use these results to train novice architects. In addition, industry practitioners provided a certain amount of evidence for successful use of ADD entangled with QAs in specific domains or application cases (e.g. SOA decisions, control systems, smart systems, financial IT systems), but the size of the decision set tends to be small, maybe because of the cost for capturing them.

At the same time, the size of the studies show that the techniques analyzed in this literature review can be applied with relatively low effort compared to the size of real software projects and no significant investment other than learning and simplistic tool development is needed to start out. Our analysis has also revealed that a large majority of the studies (85%) report evaluation of the presented approaches. Even though some of the studies present industry-related cases in the majority of the studies scientifically less rigorous evaluation approaches such as "non-industrial case studies", "example applications", and "motivating examples" have been used. In addition, the size of the decision sets in some of these industrial cases is sometimes low, which does not necessarily diminish the relevance of the evaluation approach adopted, as such decisions may belong to a subset of the architecture or system under study. Therefore,

more real examples where significant sets of decisions are made, based on stringent quality requirements, are needed.

Finally, one interesting finding for academics and professional software engineers is the frequencies of appearance of the QAs found in AKM approaches (see Table 2). Of about 40 QAs identified, only seven show frequencies of six or more. In particular, performance, security, cost, reliability, usability, maintainability, and modifiability exhibit the highest frequencies of appearance. These results might indicate that a reduced set of QAs seem to be more important or useful than others even for different domains and applications, which are at least in the scope of publications analyzed. The QAs with a higher frequency are more likely to be considered in relation to the ADDs than the QAs with a lower frequency, therefore, the most frequent QAs should be considered first by software engineers when designing software-intensive systems.

7.2 Limitations

The limitations of our study are the following: In many cases, we needed to interpret the implicit use of QAs in the ADD related approaches, as some of the results reflect our understanding given the missing or blurry specification. Therefore, the extraction and evaluation results of the related information may have led to inaccuracies. As many of the tools we discussed are not available online, we had to base our findings only on the reported results in the publications under study. There might also be some bias of the authors during the selection and classification of the candidate papers, as they have worked for years in the field and are active researchers in the area. We tried to mitigate this risk by reading and interpreting the primary studies independently in several iterations, and by double-checking each other. Also, the fact that the authors are based in different institutions and have different backgrounds helped to mitigate this risk. Finally, our systematic review may have inevitably missed some relevant tools and methods that were excluded during the search process and the implementation of our inclusion/exclusion criteria. We tried to mitigate that risk by querying and aggregating results from four databases (i.e. ACM DL, IEEEXplore, DBLP, Springer), by using general search terms (e.g. “software architecture”) and alternative terms (e.g. “quality attributes” and “non-functional requirements”), and by using forward and backward snowballing.

8 Conclusions and future trends

This paper reports a systematic literature review on QAs in architecture decision making and documentation approaches. While the findings of this literature review identify promising research and practice areas, there are still a number of factors that challenge a broader adoption of ADD methods using QAs. While the knowledge capturing problem and the relationships to

other software artifacts like requirements seems solved, the majority of the approaches examined cannot provide more automatic procedures to evaluate QAs using decisions and make decisions for the selection of the required QAs. Only some prioritization mechanisms for QAs and alternative decisions as well as the making of trade-offs can be partially automated. In addition, as QA interdependencies have been poorly discussed by ADD approaches, we provided additional insight motivating the importance of bi-directional links between ADDs and QAs to show that decisions can be motivated by changes in the quality attributes or the selection and evaluation of a particular quality is driven by a design decision. Thus, we also identified the relevant works supporting the explicit and documented trace links between both artifacts and which of these approaches use the trace links to evaluate QAs using decisions and the other way around, but also to understand how a change in a QA may affect other related QAs.

Another challenging area for knowledge sharing and collaborative approaches where decisions are not made by one single person is Group Decision Making (GDM), as well as how GDM approaches can be used in agile development contexts. Moreover, making decisions under uncertainty is another challenging area and green field to train software architects in making decisions with incomplete information. Finally, future research should provide better support and tools for attribute evaluation at decision making and for making architectural decisions sustainable over time.

Acknowledgments. This work was supported by: FFG (Austrian Research Promotion Agency) project DECO, no. 846707; FWF (Austrian Science Fund) project ADDCompliance: I 2885-N33; Spanish research network MCIU-AEI TIN2017-90664-REDT.

References

1. (2011) ISO/IEC 25010:2011 Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE). ISO/IEC 25010:2011 pp 1–34
2. (2011) ISO/IEC/IEEE 42010:2011 Systems and software engineering – Architecture description. ISO/IEC/IEEE 42010:2011 pp 1–46
3. Al-Naeem T, Gorton I, Babar MA, Rabhi F, Benatallah B (2005) A Quality-driven Systematic Approach for Architecting Distributed Software Applications. In: 27th International Conference on Software Engineering, ACM, New York, NY, USA, ICSE'05, pp 244–253
4. Alebrahim A, Hatebur D, Heisel M (2011) A method to derive software architectures from quality requirements. In: 18th Asia Pacific Software Engineering Conference, APSEC 2011, Ho Chi Minh, Vietnam, December 5-8, 2011, pp 322–330
5. Ameller D, Franch X (2014) Assisting software architects in architectural decision-making using Quark. CLEI electronic journal 17(3):1–20

6. Babar MA, Capilla R (2008) Capturing and Using Quality Attributes Knowledge in Software Architecture Evaluation Process. In: First International Workshop on Managing Requirements Knowledge, IEEE CS, pp 53–62
7. Bachmann F, Bass L, Klein M (2003) Moving from quality attribute requirements to architectural decisions. In: ICSE 2003 - Proceedings of 2nd International Software Requirements to Architectures Workshop, STRAW 2003, May 9, 2003, Portland, Oregon, USA, pp 122–129
8. Bachmann F, Bass L, Klein M, Shelton C (2005) Designing Software Architectures to Achieve Quality Attribute Requirements. Software, IEEE Proceedings 152(4):153–165
9. Bass L, Clements P, Kazman R (2012) Software Architecture in Practice (Third Edition). Pearson Education Inc., New Jersey, USA
10. Berntsson-Svensson R, Regnell B (2015) A Case Study Evaluation of the Guideline-Supported QUPER Model for Elicitation of Quality Requirements. In: Requirements Engineering: Foundation for Software Quality - 21st International Working Conference, REFSQ 2015, pp 230–246
11. Bode S, Riebisch M (2010) Impact evaluation for quality-oriented architectural decisions regarding evolvability. In: Software Architecture, 4th European Conference, ECSA 2010, Copenhagen, Denmark, August 23-26, 2010. Proceedings, pp 182–197
12. de Boer R, Lago P, Telea A, Van Vliet H (2009) Ontology-driven visualization of architectural design decisions. In: Software Architecture, European Conference on Software Architecture. WICSA/ECSA 2009. Joint Working IEEE/IFIP Conference on, Springer-Verlag, Berlin, Heidelberg, pp 51–60
13. Capilla R, Jansen A, Tang A, Avgeriou P, Babar MA (2016) 10 years of software architecture knowledge management: Practice and future. Journal of Systems and Software 116:191–205
14. Carrillo C, Capilla R (2018) Ripple effect to evaluate the impact of changes in architectural design decisions. In: Proceedings of the 12th European Conference on Software Architecture: Companion Proceedings, ECSA 2018, Madrid, Spain, September 24-28, 2018, pp 41:1–41:8
15. Cui X, Sun Y, Mei H (2008) Towards Automated Solution Synthesis and Rationale Capture in Decision-Centric Architecture Design. In: Proceedings of the Seventh Working IEEE/IFIP Conference on Software Architecture (WICSA), IEEE CS, Washington, DC, USA, pp 221–230
16. Dermeval D, Pimentel J, Silva C, Castro J, Santos E, Guedes G, Lucena M, Finkelstein A (2012) STREAM-ADD - Supporting the Documentation of Architectural Design Decisions in an Architecture Derivation Process. In: Proceedings of the 2012 IEEE 36$^{\text{th}}$ Annual Computer Software and Applications Conference, IEEE CS, Washington, DC, USA, COMPSAC'12, pp 602–611
17. Egyed A, Grünbacher P (2004) Identifying Requirements Conflicts and Cooperation: How Quality Attributes and Automated Traceability Can Help. IEEE Software 21(6):50–58

18. Esfahani N, Malek S, Razavi K (2013) Guidearch: guiding the exploration of architectural solution space under uncertainty. In: 35th International Conference on Software Engineering, ICSE '13, San Francisco, CA, USA, May 18-26, 2013, pp 43–52
19. Falessi D, Cantone G, Kazman R, Kruchten P (2011) Decision-making Techniques for Software Architecture Design: A Comparative Survey. ACM Computing Survey 43(4):33:1–33:28
20. Hafiz M, Adamczyk P, Johnson RE (2007) Organizing security patterns. IEEE software 24(4):52–60
21. Harrison NB, Avgeriou P, Zdun U (2007) Using Patterns to Capture Architectural Decisions. IEEE Software 24(4):38–45
22. van Heesch U, Avgeriou P, Hilliard R (2012) Forces on Architecture Decisions – A Viewpoint. In: Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture, WICSA/ECSA, IEEE CS, pp 101–110
23. Jansen A, Bosch J (2005) Software Architecture as a Set of Architectural Design Decisions. In: The 5th Working IEEE/IFIP Conference on Software Architecture (WICSA'05), IEEE Computer Society, pp 109–120
24. Kassab M, El-Boussaidi G, Mili H (2011) A quantitative evaluation of the impact of architectural patterns on quality requirements. In: Software Engineering Research, Management and Applications 2011 [selected papers from the 9th International Conference on Software Engineering Research, Management and Applications, SERA 2011, Baltimore, MD, USA, August 10-12, 2011]., pp 173–184
25. Kitchenham B, Charters S (2007) Guidelines for performing Systematic Literature. Tech. Rep. EBSE 2007-001, Keele University and Durham University Joint Report
26. Kitchenham B, Brereton O, Budgen D, Turner M, Bailey J, Linkman S (2009) Systematic literature reviews in software engineering: a systematic literature review. Information and Software Technology 1(51):7–15
27. Lee L, Kruchten P (2007) Capturing Software Architectural Design Decisions. In: 2007 Canadian Conference on Electrical and Computer Engineering, IEEE CS, pp 686–689
28. Letier E, Stefan D, Barr ET (2014) Uncertainty, risk, and information value in software requirements and architecture. In: 36th International Conference on Software Engineering, ICSE '14, Hyderabad, India - May 31 - June 07, 2014, pp 883–894
29. Lopes Silva IC, Brito PHS, dos S Neto BF, Costa E, Silva AA (2015) A decision-making tool to support architectural designs based on quality attributes. In: Proceedings of the 30th Annual ACM Symposium on Applied Computing, ACM, New York, NY, USA, SAC'15, pp 1457–1463
30. Lytra I, Sobernig S, Zdun U (2012) Architectural Decision Making for Service-Based Platform Integration: A Qualitative Multi-Method Study. In: Joint 10th Working IEEE/IFIP Conference on Software Architecture & 6th European Conference on Software Architecture (WICSA/ECSA), Helsinki, Finland, IEEE CS, pp 111–120

31. Lytra I, Engelbrecht G, Schall D, Zdun U (2015) Reusable architectural decision models for quality-driven decision support: A case study from a smart cities software ecosystem. In: Proceedings of the Third International Workshop on Software Engineering for Systems-of-Systems, IEEE Press, Piscataway, NJ, USA, SESoS'15, pp 37–43
32. Makki M, Bagheri E, Ghorbani AA (2008) Automating Architecture Trade-Off Decision Making through a Complex Multi-attribute Decision Process. In: 2nd European Conference on Software Architecture (ECSA), Springer, Lecture Notes in Computer Science, pp 264–272
33. Malakuti S, Goldschmidt T, Koziolek H (2018) A catalogue of architectural decisions for designing iiot systems. In: Software Architecture - 12th European Conference on Software Architecture, ECSA 2018, Madrid, Spain, September 24-28, 2018, Proceedings, pp 103–111
34. Monteserin A, Pace JAD, Gatti I, Schiaffino SN (2017) Agent negotiation techniques for improving quality-attribute architectural tradeoffs. In: Advances in Practical Applications of Cyber-Physical Multi-Agent Systems: The PAAMS Collection - 15th International Conference, PAAMS 2017, Porto, Portugal, June 21-23, 2017, Proceedings, pp 183–195
35. Nowak M, Pautasso C (2013) Team Situational Awareness and Architectural Decision Making with the Software Architecture Warehouse. In: Proceedings of the 7th European Conference on Software Architecture, Springer-Verlag, Berlin, Heidelberg, ECSA'13, pp 146–161
36. Rosa NS, Ribeiro-Justo GR, Cunha PRF (2001) A framework for building non-functional software architectures. In: Proceedings of the 2001 ACM Symposium on Applied Computing (SAC), March 11-14, 2001, Las Vegas, NV, USA, pp 141–147
37. Saadatmand M, Tahvili S (2015) A fuzzy decision support approach for model-based tradeoff analysis of non-functional requirements. In: 12th International Conference on Information Technology : New Generations, IEEE, pp 112–121
38. Schneider Y, Busch A, Koziolek A (2018) Using informal knowledge for improving software quality trade-off decisions. In: Software Architecture - 12th European Conference on Software Architecture, ECSA 2018, Madrid, Spain, September 24-28, 2018, Proceedings, pp 265–283
39. Sedaghatbaf A, Abdollahi Azgomi M (2018) Sqme: A framework for modelling and evaluation of software architecture quality attributes. Software and Systems Modeling pp 1–24
40. Shahin M, Liang P, Khayyambashi M (2009) Architectural design decision: Existing models and tools. In: Software Architecture, 2009 European Conference on Software Architecture. WICSA/ECSA 2009. Joint Working IEEE/IFIP Conference on, IEEE CS, pp 293–296
41. Shen L, Peng X, Zhao W (2012) Quality-driven self-adaptation: Bridging the gap between requirements and runtime architecture by design decision. In: 36th Annual IEEE Computer Software and Applications Conference, COMPSAC 2012, Izmir, Turkey, July 16-20, 2012, pp 185–194

42. Stoll P, Wall A, Norstrom C (2008) Guiding Architectural Decisions with the Influencing Factors Method. In: Proceedings of the Seventh Working IEEE/IFIP Conference on Software Architecture (WICSA), IEEE CS, Washington, DC, USA, WICSA'08, pp 179–188
43. Svahnberg M, Wohlin C, Lundberg L, Mattsson M (2003) A quality-driven decision-support method for identifying software architecture candidates. International Journal of Software Engineering and Knowledge Engineering 13(5):547–573
44. Tang A, Avgeriou P, Jansen A, Capilla R, Babar MA (2010) A comparative study of architecture knowledge management tools. Journal of Systems and Software 83(3):352–370
45. Tibermacine C, Fleurquin R, Sadou S (2006) On-Demand Quality-oriented Assistance in Component-based Software Evolution. In: 9th International Conference on Component-Based Software Engineering, Springer-Verlag, Berlin, Heidelberg, CBSE'06, pp 294–309
46. Tofan D, Galster M, Avgeriou P, Schuitema W (2014) Past and future of software architectural decisions – A systematic mapping study. Information and Software Technology 56(8):850–872
47. Ton That M, Sadou S, Oquendo F, Fleurquin R (2014) Preserving architectural decisions through architectural patterns. Automated Software Engineering Journal 22:1–41
48. Tyree J, Akerman A (2005) Architecture Decisions: Demystifying Architecture. IEEE Software 22(2):19–27
49. van der Ven J, Jansen A, Avgeriou P, Hammer D (2006) Using Architectural Decisions, Universitaet Karlsruhe, Fakultaet fuer Informatik, pp 1–10
50. Wohlin C (2014) Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, ACM, New York, NY, USA, EASE'14, pp 38:1–38:10
51. Wojcik R, Bachmann F, Bass L, Clements P, Merson P, Nord R, Wood B (2006) Attribute-Driven Design (ADD), version 2.0. Tech. Rep. CMU/SEI-2006-TR-023 ESC-TR-2006-023, Software Engineering Insttitute
52. Xu B, Huang Z, Wei O (2010) Making architectural decisions based on requirements: Analysis and combination of risk-based and quality attribute-based methods. Ubiquitous, Autonomic and Trusted Computing, Symposia and Workshops on pp 392–397
53. Zdun U (2007) Systematic Pattern Selection Using Pattern Language Grammars and Design Space Analysis. Software: Practice & Experience 37(9):983–1016
54. Zimmermann O, Gschwind T, Küster J, Leymann F, Schuster N (2007) Reusable Architectural Decision Models for Enterprise Application Development. In: 3rd International Conference on Quality of Software Architectures (QoSA), Springer, pp 15–32
55. Zimmermann O, Zdun U, Gschwind T, Leymann F (2008) Combining Pattern Languages and Reusable Architectural Decision Models into a

Comprehensive and Comprehensible Design Method. In: Proceedings 7th Work. IEEE/IFIP Conference Software Architecture, IEEE, pp 157–166