





# What Rationales Drive Architectural Decisions? An Empirical Inquiry

Klara Borowa<sup>(✉)</sup> , Rafał Lewanczyk, Klaudia Stpiczyńska,  
Patrik Stradomski, and Andrzej Zalewski 

Warsaw University of Technology, Institute of Control and Computation Engineering,  
Warsaw, Poland

[klara.borowa@pw.edu.pl](mailto:klara.borowa@pw.edu.pl)

**Abstract.** Architectural decision-making is a crucial concern for researchers and practitioners alike. There is a rationale behind every architectural decision that motivates an architect to choose one architectural solution out of a set of options. This study aims to identify which categories of rationale most frequently impact architectural decisions and investigates why these are important to practitioners. Our research comprises two steps of empirical inquiry: a questionnaire (63 participants) and 13 interviews. As a result, we obtained a set of rationales that motivated architects' decisions in practice. Out of them, we extracted a list of software quality attributes that practitioners were the most concerned about. We found that, overall, architects prefer to choose solutions which are familiar to them or that guarantee fast software implementation. Mid-career architects (5 to 15 years of experience) are more open to new solutions than senior and junior practitioners. Additionally, we found that most practitioners are not concerned about the quality attributes of compatibility and portability due to modern software development practices, such as the prevalence of using specific standards and virtualisation/containerization.

**Keywords:** Software Architecture · Architectural decision-making · Rationale · Software Quality Attributes

## 1 Introduction

Understanding software architecture as a set of architectural decisions (ADs) [11] draws our attention to the motivation underlying these decisions and - this way - the entire architecture. Design rationale, which is a component of ADs [25], consists of the knowledge and reasoning justifying design decisions [20].

The research on factors (including rationales) [15] that shape architectural decisions in practice is rather scarce and seems still far from being mature. The most recent papers by Weinreich et al. [23], Miesbauer et al. [14] and Tang et al. [20] that explore the motivations underlying practitioners' ADs are at least eight years old. These works are continued in more recent studies that investigate

what software quality attributes (QAs) are discussed when choosing architectural patterns [4] and what technology features drive technology design decisions [19].

As the software development landscape changes rapidly, the general purpose of this study is to discover what rationales, and why, currently drive ADs in practice. Such results importantly extend our knowledge and understanding of architectural decision-making (ADM)

by allowing researchers to focus their efforts on improving ADM on the basis of current needs and practices of architects. Additionally, we put an emphasis on QAs since they are a rationale subset that has been of major interest for researchers [2, 4, 5].

Such an aim is expressed by the following research questions:

- RQ1: What rationales most frequently influence architectural decisions?
- RQ2: Which software quality attributes are usually prioritised during architectural decision-making?
- RQ3: Why do practitioners prioritise these rationales?

In order to investigate the above problems we performed a two-phase inquiry. Firstly, we gathered data through a questionnaire. We obtained answers from 63 practitioners. Then, we presented the questionnaire's results to 13 practitioners during interviews. As a result of the questionnaire, we created a list of rationales (including quality attributes as given in ISO 25010 [10]) that practitioners of various experience levels (beginners, mid-career and experts) consider essential. As a result of the interviews, we found out that, depending on experience level, practitioners tend to prioritise different architectural options.

The rest of the paper has been organised as follows: Sect. 2 presents related work, Sect. 3 contains details about our research process and Sect. 4 the study's results. We discuss our findings in Sect. 5, present the threats to validity in Sect. 6 and conclude in Sect. 7.

## 2 Related Work

The notion that software architecture is a set of design decisions [11] has heavily impacted the field of software architecture [2]. To enable better decision-making, researchers have explored such areas as: human factors in ADM [15], AD models [25], mining AK [5], curating AK [3], tools supporting decision-making [13], techniques that can aid designers in the decision-making process [16, 21]

and ADM rationale [20].

Numerous aspects make ADM an extremely challenging process. The traditional decision-making process, which includes listing all possible alternatives and their attributes, is impractical for software design decisions [7] because of the number of possible architectural solutions. Furthermore, practitioners can be overwhelmed by the time and effort required to find architectural information [8]. Additionally, an entirely rational design-making process is impossible as long as it depends on human beings, that are impacted by various human factors [15].

While there exist general guidelines [21] and various tools [2] for ADM, empirical research on ADM factors is scarce [15]. On the topic of the practitioners' rationale behind design decisions, several studies must be acknowledged. Firstly, the study of Tang et al. [20], reporting the results of a survey on practitioners' approach to architectural rationale. Researchers had practitioners choose the importance of generic rationales and optionally allowed participants to provide their own rationales. As a result, a list of 12 rationales indicated by practitioners was made. This study's results were later expanded by Miesbauer et al. [14] and Weinreich et al. [23] who performed interview-based studies through which the list was expanded to include 18 rationales in total. Soliman et al. [19] researched what technology features impacted technology design decisions. Bi et al. [4] took a different approach and researched which ISO 25010 software quality attributes [10] were most often discussed in the context of architectural patterns on the StackOverflow platform.

We found no recent empirical research focusing widely on ADM rationale more recent than eight years ago. As software technology evolves rapidly, the rationales could also change.

Additionally, we found no studies on how rationales depend on architects' professional experience, which we believe could be relevant since junior and senior architects find different aspects of ADM challenging [22].

### 3 Method

Our research comprises two phases: questionnaire and interviews. The purpose of the questionnaire was to gather a larger sample of data that would enable us to answer RQ1 and RQ2. The interviews let us delve deeper into the meaning and implications of the questionnaire's results (RQ3). Another reason for using two data-gathering methods was to achieve so-called 'methodological triangulation' [17], which helps to strengthen the validity of our findings. The overview of the study process is presented in Fig. 1. The questionnaire questions, a summary of questionnaire results, the interview plan, and interview coding details are available online [6].

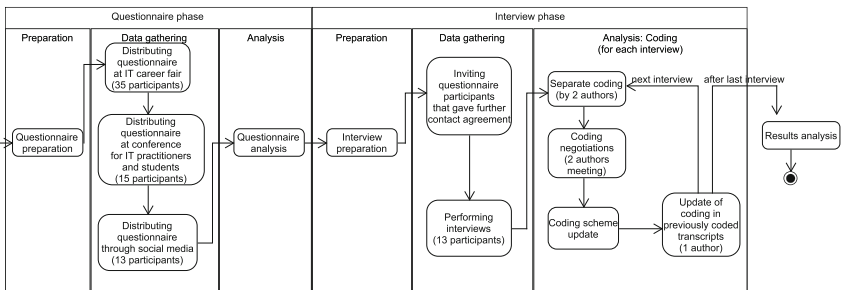


Fig. 1. Study phases

### 3.1 Questionnaire: Data-Gathering

The questionnaire's [6] design was simplistic in order to avoid discouraging practitioners from taking part and to avoid biasing the results by suggesting any specific answers. The questionnaire was divided into four main sections:

1. Participant data: age, gender, education, years of experience in software development, role in the company, company size, company domain.
2. An open-ended question to provide a maximum of three most often used rationales for architectural decisions, according to the participant's personal experience.
3. An open-ended question to provide a maximum of three most often used rationales for architectural decisions by the participant's colleagues. We asked this question to investigate if the participants believed that other practitioners have different priorities from them.
4. An optional section containing the option to provide an email and give consent for further contact from the researchers.

In order to obtain samples for the study, we distributed the questionnaires in three different locations:

1. During a 3-day long IT career fair at our faculty, where representatives of over 50 companies were present. We approached each stall and gave a physical copy of the questionnaire to the practitioners that were advertising their companies. We obtained 35 completed questionnaires at this event.
2. During an IT conference for practitioners and students, where representatives from over 60 companies were present. We used the same strategy as the one during the career fair and obtained 15 additional completed questionnaires.
3. We made the questionnaire available online and posted it on our personal social media accounts; this led to additional information from 13 participants.

In total, we obtained data from 63 participants. A summary of the participants' demographic data is presented in Fig. 2, and their employers' companies' domain and size in Fig. 3.

### 3.2 Questionnaire: Analysis

To analyse the questionnaire, we performed the following actions:

1. We divided the participants into the following groups: beginners (under five years of experience), mid-career (5 to 14 years of experience), and experienced (15 or more years of experience) practitioners.
2. We extracted the answers about the participants' as well as their colleagues' rationales and analysed them separately.
3. For each of the six combinations of the above groups (participants' experience level and their own/colleagues' rationales) separately, we classified the rationales (even if they were worded differently) into categories. When applicable,

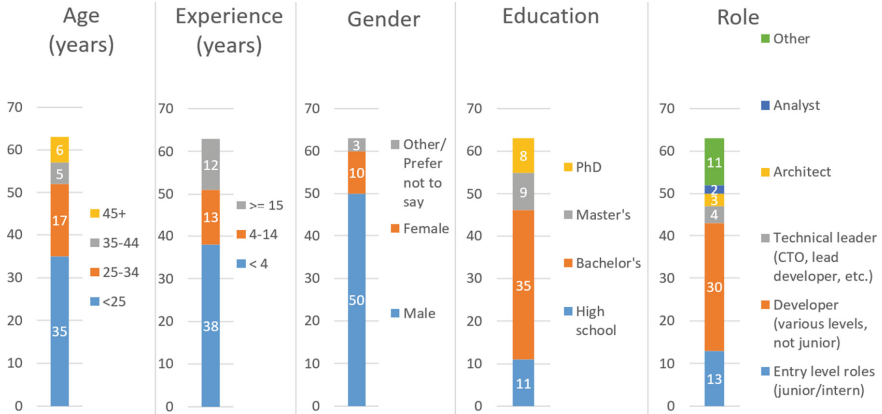


Fig. 2. Questionnaire participants

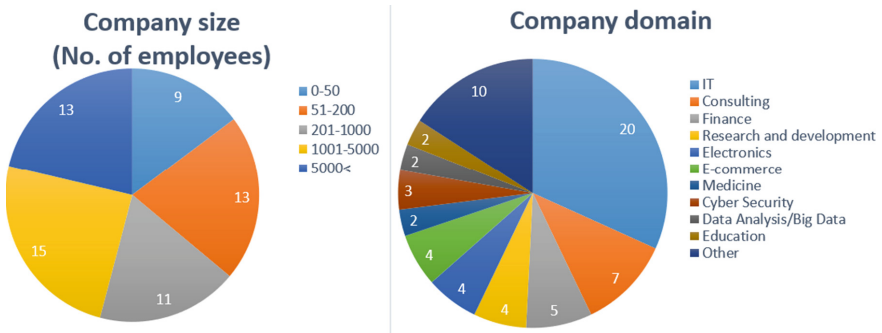


Fig. 3. Questionnaire participants companies

we used the ISO/IEC 25010 [10] software quality attributes as the rationale categories. We grouped rationales into categories, since participants often used different words to explain the same factors influencing their decision-making. A rationale category groups rationales that are similar to such a degree that we found them almost indistinguishable. For example, we categorised all of the following as “Time/Deadlines”: “time that we will waste on it; how much time there is to do it; time available to create the software; Number of hours required to write the functionality; time-consumption of making the solution; time-consuming; deadline to deliver the project; time available; time”. When rationales were only related to each other, like for example “Documentation” and “Maintainability”, we did not categorise them together. Table 3 summarises the questionnaire analysis results.

### 3.3 Interviews: Data Gathering

Based on questionnaire data analysis, when creating the interview plan [6], we focused on the following categories of observations:

1. The rationales common for 20% of the participants of each professional experience level.
2. Quality attributes of generally low interest to the architects, namely, attributes mentioned by fewer than 5% of all the participants.
3. Cases in which answers varied among architects of different experience levels. For example, some rationales were over the 20% cutoff score in one group but not in all of them.

We presented the results from the questionnaire in which the above cases occurred to the interviewees. Then, we asked them about the reasons behind the observed level of importance of these rationales for specific architects' experience groups.

All 13 interviewees were recruited from the questionnaire participants. We invited to a follow-up interview all participants that consented to a follow-up interview in the questionnaire. Table 1 presents the overview of the interviewees' characteristics.

**Table 1.** Interview participants

No.	Gender	Age (years)	Experience (years)	Education	Role	Company size (employees)	Company domain
1	Male	23	1	Bachelor's	Software Engineer	1001–5000	Infrastructure monitoring
2	Male	22	1	Bachelor's	C++ Developer	51–200	Power Engineering
3	Male	45	22	PhD	Company owner	0–50	IT, Data Science
4	Male	23	1	Bachelor's	Pythin Backend Developer	51–200	Software House
5	Male	22	1	High School	Junior Developer	1001–5000	E-commerce
6	Male	23	3	Bachelor's	Junior Java Developer	over 5000	Consulting
7	Male	24	4	Bachelor's	Software Engineer	51–200	Finance
8	Male	31	5	Master's	Software Developer	1001–5000	Electronics
9	Male	45	20	PhD	Architect	over 5000	Commerce
10	Female	25	3	Master's	NLP Engineer	over 5000	R&D
11	Male	41	20	PhD	CTO	201–1000	Finance
12	Male	28	5	High School	Senior Testing Engineer	201–1000	Videogame development
13	Male	32	6	Master's	Senior Software Engineering Manager	over 5000	FMCG

### 3.4 Interviews: Analysis

The interview recordings have been transcribed. Then we coded the transcripts by following the subsequent steps:

1. Two separate authors coded the same transcript using the descriptive coding method [18]. This means that segments of the transcripts, which contained a relevant piece of information, were labelled with a code that described its type of content. We started with an empty list of codes, to avoid biasing the results towards our own ideas, and allowed the codes to emerge during the coding process.
2. Both coding authors met to negotiate their coding [9] — they made changes to the coding until reaching a unanimous consensus.
3. An updated list of codes was created as a result of the coding meeting.
4. One of the authors re-coded previously coded transcripts with new codes if they emerged during the current analysis step.
5. The above steps were repeated for each interview transcript.

**Table 2.** Codes

Code	Description	Number of occurrences	Number of interviews where code occurred
EX	Perspective/performed tasks change with the developer's experience	58	13
CLNT	Recognising client's needs, focusing on the client's benefit.	31	12
EASY	Participant mentions how important ease of use for development/maintenance is in the project	28	13
FUT	Thinking about what effects the choice will have for the project	29	12
D	Focusing on the deadline/ how much time something will take	23	9
FAM	Choosing something based on one's familiarity with it	25	9
IMP	The rationale was omitted because it is 'obviously' important	18	9
EMP	Thinking how the choice will impact other people	15	10
CR	Focusing on personal growth	15	8
OUTDATED	The rationale does not require much thought because it is handled by newer technology	13	8
NEG	The participant disagrees with other practitioners' opinions (from the questionnaire)	11	7
EDU	Described behaviour is an effect of education	10	7
RARE	The participant considers something as niche or unimportant	9	5

Codes are summarised in Table 2. After coding all transcripts, we analysed and discussed the coded segments to draw conclusions.

## 4 Results

Table 3 presents the questionnaire results. As explained in Sect. 3.3, we consider the rationale as important to a given group of architects if it was indicated but at least 20% of them. Additionally, we focused on software quality attributes that were mentioned by less than 5% of the participants and the variation in rationale prioritisation in different groups of participants.

### 4.1 RQ1 & RQ2: Most Frequent Rationales and Prioritised Software Quality Attributes

The rationales that most frequently occurred in the questionnaires (over 20% of participants) were:

1. “**Ease of use for development**” was the dominant rationale for almost all groups of participants. Over 40% of the beginner and expert groups believed

**Table 3.** Questionnaire results. ISO/IEC 25010 quality attributes are marked by a bold font.

No	Rationale category	Sum		Beginners		Mid-career		Experienced	
		Participants	Colleagues	Participants	Colleagues	Participants	Colleagues	Participants	Colleagues
1	Ease of use for development	23	11	16	7	2	0	5	4
2	<b>Maintainability</b>	15	2	12	1	2	1	1	0
3	<b>Performance</b>	14	6	13	6	0	0	1	0
4	Prior knowledge/experience	14	14	11	9	1	2	2	3
5	Time/deadline	12	8	10	6	1	0	1	2
6	<b>Reliability</b>	10	4	6	3	2	1	2	0
7	Development Project Environment	9	2	4	1	3	1	2	0
8	Cost	8	9	5	7	1	0	2	2
9	Popularity	8	8	7	5	0	1	1	2
10	Scalability	7	3	4	3	2	0	1	0
11	Business/customer requirements	7	5	4	4	1	0	2	1
12	Documentation	6	4	6	4	0	0	0	0
13	<b>Usability</b>	5	0	3	0	2	0	0	0
14	<b>Security</b>	5	2	3	2	2	0	0	0
15	Aesthetics/UX	5	2	1	1	2	0	2	1
16	Fit with existing systems/project	5	7	4	4	0	1	1	2
17	Decision-making methodology	5	4	0	0	2	1	3	3
18	Testability (simplicity of writing tests)	4	0	3	0	0	0	1	0
19	Level of complexity of the problem/system	4	1	4	1	0	0	0	0
20	Expertise of more experienced colleagues	4	1	4	1	0	0	0	0
21	<b>Functional Suitability</b>	3	1	2	1	0	0	1	0
22	Availability of packages	3	0	1	0	0	0	2	0
23	Team members' preferences	3	4	2	4	0	0	1	0
24	<b>Portability</b>	2	2	2	1	0	0	0	1
25	System life expectancy	2	0	0	0	0	0	2	0
26	I want to add new skill to my resume	2	1	2	1	0	0	0	0
27	<b>Compatibility</b>	1	2	0	0	0	0	1	2
28	Return on Investment (ROI)	1	1	1	1	0	0	0	0
29	Market expectations	1	0	1	0	0	0	0	0
30	Available human resources/money	0	4	0	2	0	2	0	0
31	Bus factor	0	1	0	1	0	0	0	0
32	“It works so I should use it”	0	1	0	1	0	0	0	0
33	My colleagues have the same rationales as me	0	19	0	13	0	4	0	2



that it was important. However, this was not the case for mid-career practitioners, where only 15% mentioned this rationale.

2. The quality attribute of **“Maintainability”** was the second most often indicated rationale, which was mentioned by 24% of the participants. This was due to the beginners’ insistence that this rationale is important (30% of them), though it was not similarly prioritised by mid-career practitioners (15%) and experts (8%).
3. Both the quality attributes of **“Performance”** and **“Prior knowledge/experience”** were mentioned by the same number of practitioners overall (22%). **“Performance”**, similarly to **“Maintainability”**, was important to beginners (33%) but not to mid-career (0%) and to expert practitioners (8%). **“Prior knowledge/experience”** of the solution, in the same way as **“Ease of use for development”**, was prioritised by both beginners and experts (over 20% in both groups) but not by mid-career practitioners (only 8%).

Rationales that were overall mentioned by less than 20% of the participants but were important for a particular group of practitioners (over 20% of that group):

1. **“Time/deadline”** is a rationale that was mentioned by 26% of beginners but less often by mid-career and expert practitioners (8% in both groups).
2. **“Development Project Environment”**, which refers to various aspects of management and organisation of development project (e.g. company standards, client specifics) or current possibilities (available technologies), was important to mid-career practitioners (23%) but less so to beginners (10%) and experts (16%).
3. A **“decision-making methodology”** was by experts (25%) but only a few mid-career practitioners (8%) and no beginners.

Three software quality attributes were mentioned by less than 5% of the participants: **Compatibility** (1 participant), **Portability** (2 participants) and **Functional Stability** (3 participants).

Finally, when asked about their colleagues’ rationales, most participants wrote unprompted in their questionnaires that **their colleagues are motivated by the same rationales as they are themselves** (30%). These were not cases of copying the same answers from one question to another but literally writing a statement about one’s colleagues. This answer dominated the beginner (33%) and mid-career (31%) groups but occurred less frequently in the expert group (17%).

## 4.2 RQ3: Rationales’ Origins

By analysing the interviews, we found a key set of rationales’ origins. Some rationales and rationale origins may slightly overlap (e.g. “Time/deadlines” rationale and “fear of deadlines” rationale origin). This was the case when participants listed both a rationale in the questionnaire and a rationale’s origin in the

interviews. The rationale's origins include (number of code occurrences over-all/number of interviews where code occurred):

1. **Practitioner's experience(58/13)**: The primary origin of the practitioners' rationales were their previous experiences. Beginners had limited experience, and to avoid the risk of not performing their duties efficiently, they preferred the solutions which they had used previously - because of that, "Ease of use for development" and "Prior knowledge" turned out to be the prevailing rationale for them. As one of the participants stated: "(...) [junior developers] are such fresh people, it is certainly much more convenient. Because, well, since it's easy to learn something [how to use a solution], it's easy to reach the right level quite quickly."

Experts with significant experience also prioritised these rationales but for different reasons - they already had knowledge that they were confident in, so they did not feel the need to try new solutions and leave their comfort zone, e.g. "Maybe more experienced people who worked a long time with a certain technology change it less often than people who are just entering the IT market (...), but feel comfortable with certain technologies and have been comfortable working with them for many years."

The exception to this effect were mid-career practitioners who were most likely to possess the knowledge and willingness to discover new solutions. As a participant said: "Maybe the moderately experienced people are neither those very experienced people who have been working in a particular technology for a longer period of time, but those who change it more often and maybe they see that it is not that difficult, they are used to changing technologies."

The practitioner's experience influence was also crucial for choosing the "Time/deadline" and "decision-making methodology" rationales. Beginners feared the possible consequences of missing a deadline more than other practitioners. Hence, they indicated the "Time/deadline" rationale more frequently than more experienced architects, e.g. "People with more experience are more assertive when it comes to deadlines and are able to say 'no' when they know that it is simply impossible to do something in a certain time, and those with less experience may also not be so sure that this is the moment that it is worth saying 'no' and not doing something, they are afraid of the deadline."

However, using a "decision-making methodology" as their rationale's foundation was only possible to experienced practitioners due to their greater knowledge, e.g. "(...) we [the architects] are just getting used to such methodologies, acquiring them, so we will only use them after some time."

2. **Client focus (31/12)**: Various rationales originated from the endeavour to meet the client's needs. Practitioners often prioritised "Ease of use for development" and "Time/deadline" rationales because they strived to deliver new functionalities to the client as soon as possible, e.g. "(...) recently there has been a lot of emphasis on time to market and deadlines for implementing individual functionalities, which are usually short."

Similarly, the “Development Project Environment” had to be considered to satisfy the client’s needs. Even if two projects appeared to be the same, the environment often made a difference in its development. As one participant stated: “Otherwise, seemingly the project sounds the same, but in practice, the client often wants something completely different than the previous one.” Additionally, “Performance” was seen generally as a key software quality attribute from the client’s perspective, since weak software performance (software freezes, long waiting times, etc.) was seen as very problematic to the clients, e.g. “(...) usually the performance of the system is related to the comfort of use, so it seems to me that this is also the reason why performance is an important criterion.”

3. **Making one’s life “easy” (28/13):** Generally, practitioners choose solutions that they believed would make their work as effortless as possible. This was not only related to the “Ease of use for development” rationale but also “Prior knowledge” (the source of information about what is “easy”) and “Maintainability” (minimisation of future work). As one participant stated: “Some developers are lazy, which means that solutions that are easier to maintain often scale easier and perhaps require less work or less mental effort to add a new feature or to fix a bug.”
4. **Thinking of the project’s future (29/12):** In general, practitioners were aware of the software life-cycle and knew that “Maintainability” could impact the amount of effort they would have to put into maintaining the system in the future. However, “Ease of use for development” was also a rationale impacted by this factor. Practitioners believed that if it is easy to use a given solution, it will also be easier to find, hire and train new employees that would work on the project in the future, e.g. “(...)the ease of training new employees to work, whenever the software is easier to develop and is based on popular technology or the code is transparent, it is easier to introduce someone new here.”
5. **Fear of deadlines (23/9):** The fear of missing a deadline had a major impact on beginner practitioners. This was not the case for mid-career and expert practitioners since they already had experiences with missed deadlines in their careers and had the capacity to imagine how such a situation could be handled. For example: “I think it’s because the more experienced ones, I also know that this is how managers and programmers work, as well as project managers, that they know that this deadline is set with some reserve.”
6. **Familiarity with a particular solution(25/9):** Prior experience with a particular solution was the main source of architectural knowledge. Since it is rarely possible to explore all the possible alternatives, prior experiences are the primary source of information, e.g. “Architecture, all engineering, in general, is based on experience, and experience means things that we brokne in previous designs, in previous products. And on this experience, which looks so negative, but is nevertheless building our knowledge, we base what we create in the future.”.

7. **“Obviousness” (18/9)**: In the case of the “Functional Stability” quality attribute, some practitioners expressed the opinion that the importance of this rationale is simply obvious, and as such, there was no need to mention it in the questionnaire, e.g. “It’s [Functional Stability] also so mundane and part of such day-to-day work that maybe we don’t tie it to the architecture.”.
8. **Empathy (15/10)**: “Ease of use for development” and “Maintainability” were often prioritised because of the practitioners’ awareness that their colleagues will have to maintain and further expand a system in the future, e.g. “It should be done in such a way that I would not hurt myself or that it would not be painful for my colleagues to maintain. I see in this perhaps some form of empathy.”.
9. **Personal growth (15/8)**: Mid-career practitioners did not prioritise “Ease of use for development” and “Prior knowledge” rationales, as beginners and experts did. Our participants pointed out that mid-career practitioners are in a specific professional situation where they can already feel confident in their basic knowledge (unlike beginners) but strive to learn about new solutions to further develop their careers (unlike experts). As one participant stated: “(...) resume driven development, i.e. we choose those technologies that will look nice in the CV, or that will make us learn something.”.
10. **New technology handles the problem (11/7)**: In the case of the “Compatibility”, and “Portability” quality attributes, practitioners believed that new technologies already solved most problems related to these rationales. In the case of “Compatibility”, currently, existing standards are widely used, and compatibility problems are rare. As a participant stated: “(...) because everything is somehow compatible with each other, only a matter of certain calling some services(...)”.  
Similarly, the widespread use of virtualisation and containerisation solved most problems with “Portability”, as a participant stated: “(...) because practically everything can be uploaded, containerized”.
11. **Practitioner’s education(10/7)**: “Performance” was stated to be a rationale prioritised by beginner practitioners that recently finished their degrees in a field related to Software Engineering. This was due to the focus on the use of optimal data structures and algorithms during their studies, e.g. “(...) during studies and in earlier educational programming, a lot of emphasis was placed on making these solutions work quickly. I even had one subject where we were judged on how many minutes it took to run a program, so it stuck in my head a bit.”.
12. **Perception of the quality attribute as unimportant(9/5)**: Some participants stated that in the case of the projects that they worked on, “Compatibility” and “Portability” quality attributes were not important. For example, the project was targeted to work on a very specific platform, as the participant stated: “(...)projects are created, for specific hardware or for specific platforms, not multi-platform solutions.”

## 5 Discussion

Two top rationales that were not quality attributes were “Ease of use for development” and “Prior knowledge/experience”. This result is similar to the findings of Miesbauer et al. [14] and Weinreich et al. [23] who found that the most influential rationale was “Personal experience/Preferences”. This implies that the current trend of researching human factors in ADM [2,15] is appropriate for further understanding and improving ADM. To be more specific, it seems that practitioners prioritise minimising their own and their colleagues’ workload, both in the short and the long term. This fits with the principle of “Simplicity – the art of maximising the amount of work not done” [1] from Agile software development. However, if done inappropriately, this can lead to consequences such as incurring architectural technical debt [12].

The quality attributes of “Maintainability” and “Performance” were perceived as the most important out of the set of ISO 25010 software quality attributes [10]. This matches the findings of Bi et al. [4] who found these to be the most often discussed quality attributes in the context of architectural patterns. We further explain this phenomenon since we found that beginner practitioners emphasise these rationales more than experts. In the case of “Maintainability”, it seems that they wanted to avoid their own future workload, which may be perceived as an intimidating perspective. In the case of “Performance”, beginners followed the knowledge acquired during their formal education and the emphasis of scholars on algorithmic efficiency.

Additionally, we found that practitioners in general do not put an emphasis on the quality attributes of “Portability” and “Compatibility”. Modern technologies deliver solutions that well-address both these issues. In the case of “Portability”, there are many efficient tools that resolve such problems: virtualisation, containerisation or frameworks for building multi-platform applications. Furthermore, in some fields (like developing console video games), the hardware on which the software will be run can be accurately predicted. Challenges with “Compatibility” have been overcome mostly through the standardisation of the technologies used by practitioners; for example, in the case of web applications, a REST API between the front-end and back-end layers is a predictable solution that most would choose by default.

Finally, we discovered that depending on experience level, practitioners have a significantly different mindset when it comes to ADM. Beginners are greatly influenced by a fear of the unknown: they fear that it would be too hard to develop the software, or to maintain it later, to learn new solutions during the projects, and the consequences of unmet deadlines. Experts experience less fear of deadlines but put an emphasis on ease of development to make their colleagues’ work easier and feel comfortable with their current practices. They were also the only group to use any decision-making methodologies, which they found natural if they gained enough knowledge. Lastly, mid-career practitioners are the most open to learning about new solutions and attempting not to use ones that are not considered “easy”, to create bespoke solutions that would fit their clients the best.

## 6 Threats to Validity

In this Section we describe three main kinds of threats to validity [17]:

**Construct Validity** To find the participants' rationales for architectural decisions, the possible methods of enquiry are either methods based on self-reporting or observation of the participants' work. We have chosen self-reporting methods (questionnaires and interviews) since that enabled us to obtain data from a greater number of practitioners. However, it is still possible that the participants' actual rationale may differ from those that they reported. For example, they may be impacted by cognitive biases [24] that they are not aware of.

**Internal Validity** To maximise the internal validity of our findings, the coding of the transcripts was always done independently by two authors. Then, both discussed the coding until they unanimously agreed on all codes. This was done to minimise the impact of the researcher's bias on the findings. However, it is possible that factors that we did not consider could play a role in practitioners' approach to decision-making, such as their company's size or domain.

**External Validity** We used convenience sampling since it is an extreme challenge to obtain a random generalisable sample of software practitioners. However, we strived to overcome this by providing data source triangulation [17]: we searched for participants from three different sources (two in-person events and one on social media). This resulted in a varied group of participants.

Though, worth noting is that the sample may be biased towards less experienced practitioners, due to the majority of participants having less than 4 years of professional experience. Additionally, since our results partially match results from previous studies [4, 14, 23], it seems that our sample was big enough to give us outcomes also noticeable to other researchers.

## 7 Conclusion

In this study, we performed a mixed methods two-step empirical inquiry into the practitioners' rationale behind their architectural decisions. The three main contributions of this study are as follows:

1. A list of the most impactful rationales that influence practitioners' architectural decision-making;
2. An exploration of these rationales' origin;
3. The finding of how a practitioner's experience has a significant impact on how they make architectural decisions.

Future research could employ different research techniques to further confirm or disconfirm our findings. A survey on a random generalisable sample would be beneficial, as well as observational studies on practitioners that would explore their decision-making in real-time. In accordance to our findings, since experience level seems to be a major factor shaping who architects make their decisions researchers should take it into account during future research on ADM.

Practitioners could benefit from our study by understanding better the way they and their colleagues develop software architectures. The observation on the influence of experience on ADM should also be reflected in shaping a team's structure, e.g. it would be prudent to focus on having mid-career (between 5 and 14 years of experience) practitioners in their teams when working on an innovative project.

## References

1. Beck, K., et al.: Principles behind the Agile Manifesto (2001). <https://agilemanifesto.org/principles.html>
2. Bhat, M., Shumaiev, K., Hohenstein, U., Biesdorf, A., Matthes, F.: The evolution of architectural decision making as a key focus area of software architecture research: a semi-systematic literature study. In: 2020 IEEE International Conference on Software Architecture (ICSA), pp. 69–80. IEEE (2020)
3. Bhat, M., Tinnes, C., Shumaiev, K., Biesdorf, A., Hohenstein, U., Matthes, F.: ADeX: a tool for automatic curation of design decision knowledge for architectural decision recommendations. In: 2019 IEEE International Conference on Software Architecture Companion (ICSA-C), pp. 158–161. IEEE (2019)
4. Bi, T., Liang, P., Tang, A.: Architecture patterns, quality attributes, and design contexts: how developers design with them. In: Proceedings - Asia-Pacific Software Engineering Conference, APSEC 2018-Decem(61472286), pp. 49–58 (2018)
5. Bi, T., Liang, P., Tang, A., Xia, X.: Mining architecture tactics and quality attributes knowledge in stack overflow. *J. Syst. Softw.* (May) (2021)
6. Borowa, K., Lewanczyk, R., Stpicyńska, K., Stradomski, P., Zalewski, A.: What rationales drive architectural decisions? An empirical inquiry - Additional material (May 2023). <https://doi.org/10.5281/zenodo.7946764>
7. Burge, J.E.: Design rationale: Researching under uncertainty. *Artif. Intell. Eng. Design, Anal. Manuf.: AIEDAM* **22**(4), 311–324 (2008)
8. De Dieu, M.J., Liang, P., Shahin, M.: How do developers search for architectural information? An industrial survey. In: Proceedings - IEEE 19th International Conference on Software Architecture, ICSA 2022 (December 2021), pp. 58–68 (2022)
9. Garrison, D.R., Cleveland-Innes, M., Koole, M., Kappelman, J.: Revisiting methodological issues in transcript analysis: negotiated coding and reliability. *Internet Higher Educ.* **9**(1), 1–8 (2006)
10. ISO/IEC 25010: ISO/IEC 25010:2011, systems and software engineering - systems and software quality requirements and evaluation (square) - system and software quality models (2011)
11. Jansen, A., Bosch, J.: Software architecture as a set of architectural design decisions. In: 5th Working IEEE/IFIP Conference on Software Architecture (WICSA'05), pp. 109–120. IEEE (2005)
12. Kruchten, P., Nord, R., Ozkaya, I.: *Managing Technical Debt*. Addison-Wesley Professional (2019)
13. Liu, M.X., et al: Unakite: scaffolding developers' decision-making using the web. In: Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology, pp. 67–80 (2019)
14. Miesbauer, C., Weinreich, R.: Classification of design decisions – an expert survey in practice. In: Drira, K. (ed.) *ECSA 2013. LNCS*, vol. 7957, pp. 130–145. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-39031-9\\_12](https://doi.org/10.1007/978-3-642-39031-9_12)

15. Razavian, M., Paech, B., Tang, A.: Empirical research for software architecture decision making: an analysis. *J. Syst. Softw.* **149**, 360–381 (2019)
16. Razavian, M., Tang, A., Capilla, R., Lago, P.: Reflective approach for software design decision making. *Proceedings - 1st Workshop on Qualitative Reasoning about Software Architectures, QRASA 2016* pp. 19–26 (2016)
17. Runeson, P., Höst, M., Rainer, A., Regnell, B.: Case study research in software engineering: guidelines and examples (2012). <https://www.wiley.com>
18. Saldaña, J.: *The coding manual for qualitative researchers*, pp. 1–440 (2021)
19. Soliman, M., Riebisch, M., Zdun, U.: enriching architecture knowledge with technology design decisions. In: *Proceedings - 12th Working IEEE/IFIP Conference on Software Architecture, WICSA 2015*, pp. 135–144 (2015)
20. Tang, A., Babar, M.A., Gorton, I., Han, J.: A survey of architecture design rationale. *J. Syst. Softw.* **79**(12), 1792–1804 (2006)
21. Tang, A., Kazman, R.: Decision-making principles for better software design decisions. *IEEE Softw.* **38**(6), 98–102 (2021)
22. Tofan, D., Galster, M., Avgeriou, P.: Difficulty of architectural decisions - a survey with professional architects. In: Drira, K. (ed.) *Softw. Architect.*, pp. 192–199. Springer, Berlin Heidelberg (2013). [https://doi.org/10.1007/978-3-642-39031-9\\_17](https://doi.org/10.1007/978-3-642-39031-9_17)
23. Weinreich, R., Groher, I., Miesbauer, C.: An expert survey on kinds, influence factors and documentation of design decisions in practice. *Future Gener. Comput. Syst.* **47**, 145–160 (2015)
24. Zalewski, A., Borowa, K., Ratkowski, A.: On cognitive biases in architecture decision making. In: Lopes, A., de Lemos, R. (eds.) *Softw. Architect.*, pp. 123–137. Springer International Publishing, Cham (2017). [https://doi.org/10.1007/978-3-319-65831-5\\_9](https://doi.org/10.1007/978-3-319-65831-5_9)
25. Zimmermann, O., Koehler, J., Leymann, F., Polley, R., Schuster, N.: Managing architectural decision models with dependency relations, integrity constraints, and production rules. *J. Syst. Softw.* **82**(8), 1249–1267 (2009)