2011 Ninth Working IEEE/IFIP Conference on Software Architecture

Mature Architecting - A Survey about the Reasoning Process of Professional Architects

Uwe van Heesch
University of Groningen,
Fontys University of Applied Sciences Venlo
Venlo, The Netherlands
uwe@vanheesch.net

Paris Avgeriou
University of Groningen
Groningen, The Netherlands
paris@cs.rug.nl

***Abstract*— Architecting is to a large extent a decision-making process. While many approaches and tools exist to support architects during the various activities of architecting, little guidance exists to support the reasoning part of decision-making. This is partly due to our limited understanding of how professional architects make decisions. We report on findings of a survey that we have conducted with 53 industrial software architects to find out how they reason in real projects. The results of the survey are interpreted with respect to the industrial context and the architecture literature. We derive reasoning best practices that can support especially inexperienced architects with optimizing their decision-making process.**

I. Motivation

A software architecture is the result of a complex system of inter-dependent architectural design decisions [1], [2]. These decisions are made by architects who strive towards an optimal balance between the forces acting on the decisions, including financial and technical constraints. Architectural decisions are the corner stone for the whole software architecture and as such they are vital for the achievement of the system key drivers and goals.

Architecture decisions are made during the iterative and incremental process of architecting. Hofmeister et al. derived three general, recurring architecting activities, which are common in five industrial architecture approaches [3]: *Architectural analysis*, which is concerned with identifying architecturally significant requirements (ASR) from a set of architectural concerns and the business context; *architectural synthesis*, which concerns finding candidate solutions for the ASRs; and finally *architectural evaluation* in which decisions are made and validated against the architecture as a whole. These three activities are iteratively performed by moving back and forth between the problem and the solution space [4].

Various approaches have been proposed to support the three architecture design activities, and they are either concerned with the architecting process as a whole, or they focus on one of the three activities. Well known examples of the former category are the 5 processes used as reference in [3]: RUP, ADD, Siemens' 4 Views, BAPO and ASC. The latter category includes approaches for architecture evaluation like ATAM, SAAM, or CBAM [5]; approaches for architecture analysis like the goal-oriented paradigm (e.g. [6]); and various methods supporting architects in identifying candidate solutions during architectural synthesis, e.g. architectural patterns [7], styles [5] and reference architectures [8].

The aforementioned approaches, however, either ignore the reasoning process behind decision-making, or take design decisions into account only as input or output for individual architecture activities (ATAM for instance evaluates the role of design decisions in quality attribute scenarios). To the best of our knowledge, there is no holistic reasoning process that includes all three major architecture activities (analysis, synthesis and evaluation); nor can one be derived from the combination of multiple approaches, as the whole is more than the sum of the parts. In fact, with a few exceptions (e.g. [9], [10], [11]), very little research has been done on the reasoning part of decision-making so far.

Design reasoning is a logical process that designers follow when developing architectural solutions [11]. It applies to all three architecture activities and allows for systematic and disciplined decision making, based on argumentation instead of intuition. Furthermore, if the output of reasoning is documented, it can support stakeholders who were not involved in the decision making process to comprehend decisions and the resulting design. The lack of such reasoning processes, forces software architects to follow an ad-hoc, creative process [12], [13] relying heavily on their personal experience and expertise. As a consequence, rather inexperienced software architects go through a long and painful succession of sub-optimal decisions, before they can successfully reason about the design options and make informed, well-balanced trade-offs. Training practitioners to follow a systematic reasoning process could narrow the gap between expert architects and novice ones.

In our previous work, we started analyzing the reasoning process that inexperienced architects follow when they are architecting [14]. Our aim was to establish a baseline reasoning process that is based on common sense instead of experience. In this paper we present the results of a descriptive survey that we conducted with 53 industrial software architects from end-October 2010 until mid-January 2011. We investigate how experienced architects reason in the context of industrial projects and interpret the data according to industrial context and theory from the literature. Eventually we refine the findings and summarize them into a set of reasoning best practices that junior architects can use to improve their reasoning skills.

The rest of this paper is organized as follows. Section II presents related work. In Section III, the design of the study is introduced. The next section presents the analysis of the results, which are interpreted in Section V. The paper ends with conclusions and directions for future work.

II. Related Work

Our research is related to three areas within software architecture: architecting processes, architecting practice in the industry and design reasoning.

In order to study the reasoning process, we use the general model of architecture design by Hofmeister et al. as a reference process [3]. This model consists of three main architecture activities from industrial approaches, namely architectural analysis, architectural synthesis and architectural evaluation. Jansen et al. adopt the model to describe architecture activities from the perspective of architectural decision making [15]. They suggest that architectural decisions are the result of a decision-making process comprised of the activities defined in Hofmeister et al. 's general model. Our work is complementary to these approaches, as we explicitly focus on the reasoning process related to each of the architecture activities when making decisions.

The role and duties of software architects in the industry have been analyzed in multiple studies [16], [17], [18], [19]. Findings include, that risk assessment and architecture evaluation is not regarded very important by practicing architects and that architects mainly follow a non-iterative approach that subsequently satisfies requirements[16]. Fahrenhorst et al. refine those findings, stating that auditing and quality assurance activities are regarded more important with increasing years of experience [17]. Clements et al. suggest that evaluation and analysis are regarded less important in practice than in the literature [19]. In this study, we also observe the behavior of practicing architects in the context of industrial projects. However, the emphasis in the aforementioned papers is to find out what architects do, i.e. which activities they follow while they are architecting. In our study, we try to understand *how* architects perform the activities in order to derive reasoning practices.

As pointed out in Section I, little work has been done in the field of design reasoning in software architecture. Tang et al. look at design reasoning from a more general perspective, not only specific to software architecture and also take psychological aspects into consideration to explain human behavior during design activities [13]. In earlier work, they declared the importance of design reasoning in software architecture [11], [9], [20]. The results were used by Tang and Lago to describe an initial set of design reasoning tactics that can be used by software architects to improve their reasoning process [10]. Our work also emphasizes the importance of reasoning processes in software architecture. As opposed to Tang et al., who look at design reasoning from a very general, cognitive perspective, our aim is to understand and describe concrete reasoning practices within the three architecting activities found by Hofmeister et al. [3] that can be used as guidelines for inexperienced architects.

III. Design of the Study

A. Goal

The goal of this survey is to understand the reasoning process that industrial software engineering practitioners follow while they are architecting. To make the research goal concrete, we map the reasoning process onto the general model of architecture design by Hofmeister et al. [3]. The three activities in the model are iteratively performed by architects when making decisions. We aim at finding out the reasoning practices behind these activities, i.e. **how** each of the three activities is performed, which leads to the following research questions:

RQ1 : How do software architects scope and prioritize the problem space during architectural analysis?

RQ2 : How do software architects propose solutions during architectural synthesis?

RQ3 : How do software architects choose among solutions during architectural evaluation?

Research question one considers the involvement of architects in requirements engineering activities such as: requirements elicitation, evaluation of the importance and prioritization of quality attribute requirements and functional requirements and the definition of concrete problems that are small enough to be addressed in the architectural synthesis.

The aim of research question two is to find out how architects search for and choose design options based on the output of the architectural analysis.

Finally research question three applies to the assessment of candidate solutions and the evaluation of the architecture as a whole during architectural evaluation. The scope of this question includes architecture reviews and risk management.

B. Subjects and Sampling

The population under study are industrial software engineering practitioners, who have been working in the industry for at least five years and who have been responsible for software architectural design for at least two years. As an additional constraint, subjects were excluded from the study if their daily tasks do not include at least one of the following: requirements engineering, system architecture/design, or software design and specification. To evaluate if the subjects

Table I
QUESTIONS FOR SAMPLING

Question	Response Format
How many years have you been working as an IT professional?	Positive natural numbers including zero
How many years have you been working as a software architect / designer?	Positive natural numbers including zero
As an architect / designer, which of the following are your tasks?	Possible answers: • project management • requirements engineering • software architecture • software design and specification • test planning and design • reviewing / auditing • programming • others

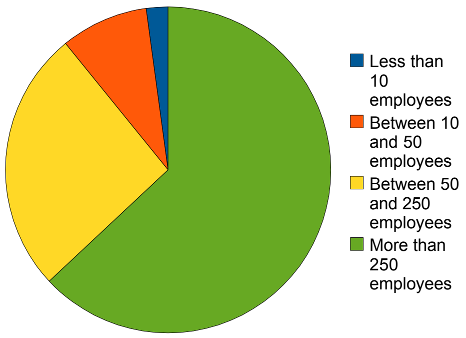

Figure 1. Number of employees in participating companies

fit into the target population, we asked the questions shown in Table I. To find appropriate subjects, we used chain referral sampling (also known as snowballing) [21]: the authors asked some individuals from their own network to forward the participation request to other professionals who fit the sampling requirements. In total, 53 people took part in the survey, out of which the results from seven people were excluded, because they did not satisfy the sampling requirements. On average, the remaining participants have worked 18.22 years in the IT-industry (min: 6, max: 35), and on average 10.59 years as a software architect/ designer (min: 4 , max: 30). For statistical means, we asked the participants to specify the numbers of employees in their companies using an interval scale ranging from *less than 10* to *more than 250 employees*. Figure 1 shows the distribution of answers. The majority of participants work in large companies.

C. Data Collection

To collect data, a web-based questionnaire was designed with questions that map to the defined research questions. Table II shows the questions from the questionnaire along with the response format and their relation to the research questions. If multiple research questions are concerned, the primary one is printed bold.

Using questionnaires, the subjects and researchers do not have to synchronize in time and place. Participants can fill them in, whenever they find time. A potential disadvantage of questionnaires is that in the case of ambiguous and poorly-phrased questions, there is no interviewer to explain the questions and make sure they are well understood. To mitigate this risk, Lethbridge et al. propose to pilot-test the questions and then re-design those questions that were interpreted wrongly [22]. We followed this advice and tested the questionnaire initially with one participant from the target population. Right after the questionnaire was filled in online, we had a video conference with the subject and asked him to explain how he understood every single question. After this, all questions that were poorly understood were re-designed and we provided additional help texts explaining the questions. Then we repeated the procedure with three additional participants from the target population until every question was explained back to us just the way we aimed it to be understood.

As described in the previous subsection, the URL of the questionnaire was sent to the participants by e-mail. It contained a mix of structured and un-structured questions. The structured questions had a five-point interval-level response format, also referred to as Likert-scale [23], whereas the unstructured questions requested numeric input or free-text.

In the questionnaire, we asked respondents to remember one specific software project they were involved in as a software architect and which is representative for the way they are working. The whole set of questions in the questionnaire referred only to this concrete project. To focus the participants on this project we asked them to estimate the project size and specify the domain of the project. The characteristics of the chosen projects are further described in Section IV. Furthermore we explicitly requested them to reflect upon their personal thoughts and their personal actions instead of describing their company policies or what the whole development team did. They were also asked to skip questions they did not understand.

IV. ANALYSIS

We use descriptive statistics and qualitative analysis to describe the collected data. This section is divided according to the research questions. As described in the study design, the participants were asked to reflect on one representative project they had worked on. Table III shows some characteristics of the chosen projects.

Table II
MAPPING OF QUESTIONS AND RESEARCH QUESTIONS

No	Question	Resp. Format	RQ1	RQ2	RQ3
Q1	How much were you involved in the requirements elicitation of the project?	Likert (Completely to Not at all)	X		
Q2	Have you understood the reasoning behind the requirements of the project?	Likert (Completely to Not at all)	X		
Q3	Compared to other influencing factors, how important were the requirements as input/motivation for your architectural decisions?	Likert (Very important to Unimportant)	X		
Q4	To what extent did you reflect on identifying which of the requirements were hardest to fulfill?	Likert (To the largest possible extend to Not at all)	X		**X**
Q5	How important were the functional requirements for your architectural design?	Likert (Very important to Unimportant)	X		
Q6	How important were the quality attribute requirements for your architectural design?	Likert (Very important to Unimportant)	X		
Q7	Have you searched for alternative design options, when making decisions?	Likert (Always to Never)		X	
Q8	Have you searched for alternative design options even if you already had a solution in mind?	Likert (Always to Never)		X	
Q9	Have you thought about the pros and cons of the design options you found?	Likert (Always to Never)		X	**X**
Q10	Have you preferred solutions that you are familiar with, in favor of others that you are not so familiar with?	Likert (Always to Never)		X	
Q11	Did you relax requirements?	Likert (Always to Never)	X		
Q12	How confident are you that the architectural decisions you made are sound?	Likert (Very confident to Not confident)			X
Q13	How often did you decide on multiple architectural solutions at the same time?	Likert (Always to Never)		X	**X**
Q14	How often did you withdraw solutions that you decided on earlier in the project?	Likert (Always to Never)			X
Q15	How often did you make trade-offs, while making decisions, between multiple requirements?	Likert (Always to Never)			X
Q16	How often did you come across dependencies between architectural solutions you decided on?	Likert (Always to Never)		**X**	X
Q17	How long did it take until you had a first vision of the overall software architecture in mind?	numeric (% of the whole proj. duration)	X	X	X
Q18	Does the final software architecture significantly differ from this initial vision?	Likert (Completely to Never)	X	X	X
Q19	What are the three most important things in decision making for you?	Open	X	X	X
Q20	How have you come from one decision to the next?	Open	X	X	X

Table III
CHARACTERISTICS OF THE CHOSEN PROJECTS

Variable	N	Value
Project size in SLOC	19	Min: 50K, Max: 15mill., Med: 400K
Project size in person-months	43	Min: 2, Max: 8000, Med: 150
No of architects involved	46	Min: 1, Max: 100, Med: 3
Domain of the project	46	Top six domains: Embedded systems (13.04%), Healthcare (13.04%), Transportation (13.04%), Enterprise Computing (10.87%), Realtime (10.87%), Telecommunication (10.87%)

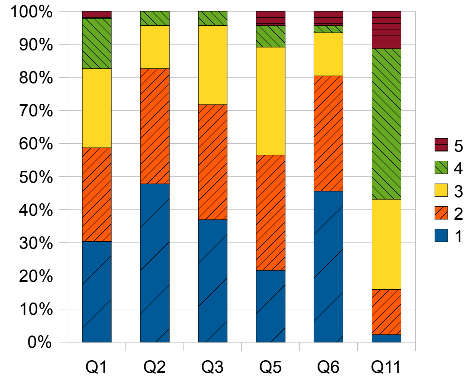

Figure 2. Cumulative frequencies of answers to questions related to RQ1

A. Analysis RQ1 - Architectural Analysis

Figure 2 shows a stacked bar chart with cumulative frequencies of answers to questions primarily related to RQ1. The colors and hatchings represent the answers to the Likert-scale questions. Depending on the concrete question ”1” stands for positive answers (*completely, very important, to the largest extend, always, or very confident*), while ”5” represents negative answers (*not at all, unimportant, never or not confident*). Please refer to Table II for the scalings of the respective questions.

Approximately 60% of the architects stated that they were involved either completely or a lot, in requirements elicitation (Q1). More than 80 % understood the reasoning behind requirements well (Q2). With more than 70% of affirmation, the requirements were regarded important for architecture decisions compared to other influencing factors like technology constraints, budget, or company culture (Q3). About 57% of the participants found the functional requirements important or very important (Q5). The quality attribute requirements were found important or very important by 81% of the respondents (Q6). Finally, the vast majority stated that they seldom or never relaxed requirements to have more design options (Q11).

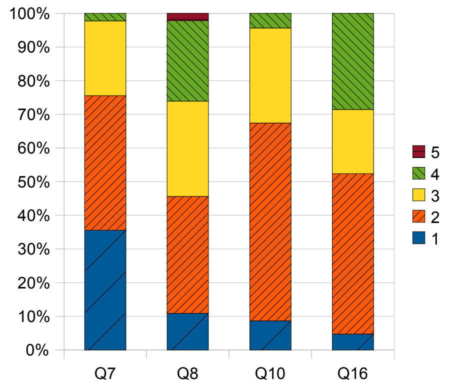

Figure 3. Cumulative frequencies of answers to questions related to RQ2

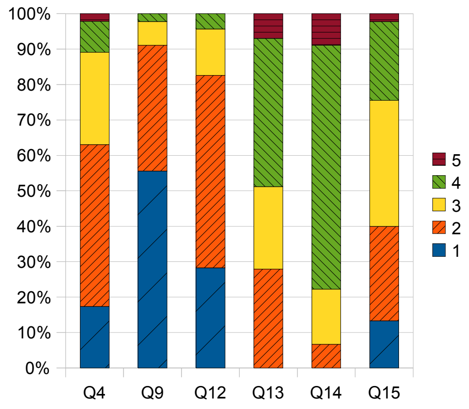

Figure 4. Cumulative frequencies of answers to questions related to RQ3

Apart from the structured questions, some answers to the open question Q19 are related to architectural analysis. The following procedure was used to analyze the open answers. We browsed the answers and searched for comments related to the research question. From the comments, we derived single concrete statements expressing what the respective participant answered. Finally, we counted the occurrences of the derived statements.

With respect to RQ1, the following statements were made. We only mention statements that were concordantly made by at least three participants. The numbers in brackets express the number of participants who made that comment: *Understand the problem domain* (12 times), *have well-defined requirements* (7 times), *consider non-technical requirements like time and resource limitations, political issues and return on investment* (7 times), *involve stakeholders in the decision making process* (7 times), *regard performance* (4 times), *consider functional- and non-functional requirements equally* (3 times), *negotiate and relax requirements* (3 times). In total, 48 comments were related to architectural analysis.

B. Analysis RQ2 - Architectural Synthesis

Figure 3 illustrates cumulative frequencies for Likert-scale questions related to architectural synthesis (RQ2).

With only one exception, all participants (74% plus 22% neutral) indicated that they usually search for alternative design options when making decisions (Q7). Significantly less participants (46%) search for alternative design options if they already have a suitable solution in mind (Q8). The respondents concordantly prefer well-known solutions in favor of unknown alternatives (Q10, 68% affirmation, 4% negation); more than 50% answered that they often come across dependencies between architectural solutions they decide on (Q16).

As for RQ1, we qualitatively analyzed the answers given to Q19 with respect to RQ2. The following statements were made by at least three participants: *Know the solution space* (7 times), *find multiple design options* (7 times), *discuss design options with colleagues* (5 times) and *choose the simplest design* (5 times). In total, 31 comments were related to architectural synthesis.

C. Analysis RQ3 - Architectural Evaluation

The results with respect to RQ3 are shown in Figure 4.

With roughly 10% of negation, more than 60% strongly reflected on identifying the most challenging requirements (Q4) and thought about the pros and cons for each of the considered design options (Q9). Likewise more than 80% had strong confidence in the soundness of their decisions (Q12). The question regarding the making of multiple decisions at the same time was answered less clearly. Approximately 50% negated the question, while less than 30% stated that they usually make multiple decision at the same time (Q13). With strong significance, the vast majority of the respondents (76%) did not withdraw decisions they decided on earlier in the project (Q14). Q15 does not show a clear tendency. The mode answer was neutral with a tendency towards affirmation (39% compared to 24% negation).

Concerning Q19, the following statements were made by at least three participants: *Understand pros and cons of each design option* (7 times), *validate decisions in reviews* (3 times). In total, 16 comments were related to evaluation.

D. Analysis of questions 17,18 and 20 in the questionnaire

Questions 17,18 and 20 in the questionnaire cannot be clearly assigned to a specific research question. We asked the participants how long it took relative to the whole duration of the architecture phase until they had a first vision of the architecture in mind (Q17), how much they derived from this initial vision after having completed the architecture (Q18) and how they came from one decision to the next (Q20).

As far as Q17 is concerned, 44 participants answered the question. In average it took the architects 17.2 % (min: 5%, max: 75%, med: 12.5 %) of the time spent on architecture to develop a first vision of the overall system. The same number of people answered question 18. The mode answer to this Likert-type question was "moderately" (39%), 11%

answered that the final architecture differed from the first vision completetly (2%) or a lot (9%). The remaining 46% answered, that the final architecture differed slightly (35%) or not at all (11%).

The open answers to question 20 describe the overall process that architects follow. For the qualitative analysis of the answers we use the same procedure as for Q19, i.e. we derive statements from the answers and count the occurrences of every statement.

The following statements were made by at least two participants. The number in brackets is the number of occurrences of the respective statement: *Requirements should be prioritized. The important ones should be regarded first* (6 times), *architecture is iteratively refined and improved* (6 times), *there is no specific order in decision making* (6 times), *the requirements guide the decision making process* (5 times), *some decisions have to be made in combination* (3 times), *sometimes candidate solutions should be prototyped to find the right one* (3 times), *some decisions have strong dependencies* (3 times), *decisions from other projects can guide the decision making process* (3 times), *the decision making process is driven by risks* (2 times).

V. Interpretation

In this section, we interpret the findings from the analysis. Specifically we interpret the architects' answers and compare them to existing approaches in the software architecture literature. The section is organized according to the three architecting activities.

A. Architectural Analysis

The purpose of architectural analysis is to define and scope the problems that have to be solved by the architecture [3], [15]. The outcome of this activity is a set of architecturally significant requirements that serve as input for the architectural synthesis.

The analysis of RQ1 showed that practicing architects are usually involved in the requirements elicitation of the project (Q1); this means that they do not just receive requirements and constraints as artifacts from requirements engineers, but they are actively involved in the communication with customers. This differs from architecture approaches in the literature, which generally assume that a set of requirements is given to the architects as input for the architectural design (see for instance ADD [5]).

The involvement in requirements elicitation partially explains the results of Q2: the vast majority of architects stated that they understood the reasoning behind requirements very well. The answers to Q19 also showed that architects find it very important to understand the problem domain and have well defined requirements. These statements were the most frequent answers to Q19, which allows the conclusion that a deep understanding of the requirements and the problem space is regarded as essential by most industrial architects.

Requirements are an important input factor for architectural decisions (Q3). This includes functional and quality attribute requirements (Q5, Q6), although the quality attribute requirements are clearly found more important than the functional requirements. Apart from the functional and quality attribute requirements the architects mentioned non-technical concerns like time and resource limitations, political issues and return on investment as important drivers for architectural decisions. This is comprehensible, as industrial practice is constrained by factors like budget and time limitations, development teams being experienced in specific technologies and customers who indicate the use of specific software systems, because of in-house software licenses. Consequently this means that an architecture that perfectly fulfills the functional and quality attribute requirements is not necessarily the right architecture in every organizational context. This finding stresses the need to document the rationale of design decisions, as decisions influenced by non-technical concerns may seem irrational or at least incomprehensible to stakeholders who are unfamiliar with those concerns.

Most of the architects answered that they seldom relax requirements (Q11). This sounds surprising in the first place as relaxing requirements that are too constraining would be a means to get more design options [10]. However, it is in accordance to the answers to Q1 and Q2: architects who are highly involved in the requirements engineering activity for a project gain deep knowledge about the problem space and have presumably already ensured that the requirements are not unrealistic or too challenging. Nevertheless, three architects mentioned that negotiating and relaxing requirements is one of the most important activities (Q19). A correlation analysis showed, that the three architects who made this comment were less involved in the requirements elicitation (med: 3 compared to med:2), understood the requirements worse than the average (med:3 compared to: med:2) and relaxed requirements more often than the average(med:3 compared to med:4). This means that architects who are less involved in the requirements elicitation process have to relax and negotiate requirements more often.

B. Architectural Synthesis

Architectural synthesis is the main activity in architectural design as it is concerned with identifying candidate solutions for the architecturally significant requirements [3]. Architects have to make use of their existing design knowledge or create new knowledge by consulting external knowledge repositories [3], [10] in order to find candidate solutions. Tang et al. suggest that creative design requires architects to refine and formulate the problem and solution space at the same time [13], in line with the "Twin Peaks" model [4]. This implies that architectural analysis and architectural synthesis are closely coupled activities.

The architects who took part in the study very frequently searched for multiple design options when making decisions (Q7). However this happens significantly less often if the architects already have a solution in mind (Q8). This might be due to the fact that searching for design options is an effort-intensive task, for which designers often do not afford the resources to perform adequately. Furthermore, designers need to search for design options on external knowledge repositories and choose candidate solutions based on unproven assumptions. In line with this finding architects prefer solutions they are familiar with instead of unfamiliar alternatives (Q10). It is less risky to select known solutions, even if they have known shortcomings, because these can be assessed and mitigated. Unfamiliar alternatives require substantial effort to reflect on and analyze, which is not always possible within the tight budget of a project.

The analysis of Q19 showed that architects find it very important to know the solution space and have multiple design options. This supports the finding by Cross [24] and Tang [11], who concordantly found that designers create better designs when they explicitly take multiple design options into consideration. In cases where the participants of our study did not have enough knowledge about the solution space to find candidate solutions on their own, they stated that they discuss design options with colleagues. They also emphasized that the simplest design for a problem should be chosen. This indicates that the size and complexity of system architecting is so overwhelming that simplicity of design solutions is of paramount importance to manage this complexity.

The answers to the open question Q20 and to the structured question 16 reflect that architects are aware of dependencies that exist between some of the decisions. Three architects suggest that dependencies have to be considered when finding candidate solutions. In addition some architects consider that certain dependencies are so significant, that the related decisions can only be made in combination. The analysis of dependencies between decisions is not supported in current architecting processes [3], but it has been discussed extensively within the Architecture Knowledge community [25].

C. Architectural Evaluation

During architectural evaluation, the candidate solutions from the synthesis are validated against the architecturally significant requirements [3]. This entails considering advantages and disadvantages of the candidates [15]. Some of the candidate solutions require trade-offs to be made between multiple requirements [3], [10]. Dependencies between decisions and constraints for future decisions should be analyzed and documented thoroughly [5]. Many architecture approaches regard risk assessment as integral part of architectural evaluation (see for instance [10] and [26]). Finally, the architecture as a whole should be evaluated regularly to make sure that decisions are consistent with each other, e.g. that older decisions do not harm constraints that came up after they were made.

With respect to the evaluation of candidate solutions, almost all architects stated that they usually think about the pros and cons of design options (Q9). Some emphasized the necessity for prototyping different candidate solutions before making a decision (Q20). They also have high confidence in the soundness of their design (Q12), which indicates that they have made an informed choice with respect to the pros and cons of the design options. It is noticeable that comparably few architects often decide on multiple decisions at the same time (Q13), although the majority of architects are aware of dependencies between decisions (Q16). A correlation analysis (Kendall's tau) did not show a significant correlation between Q13 and Q16 (corr.-coefficient 0,256, sig. 0,066), which means that architects who are aware of dependencies between decisions do not necessarily make more decisions in combination. This may also be due to the complexity of the various problems and their solutions; each design decision may be complex enough in its own right, making it difficult to take into account its dependent decisions.

Regarding Q14, the results are clear: architects seldom reject decisions they made before. This is in line with the findings of Tang et al., who suggest that designers are reluctant to change their minds [13]. This, however, could indicate that previously made architecture decisions are seldom revisited, i.e. the architecture is not validated at the end as a set of decisions. One comment to Q19 is a strong affirmation of this attitude: "once the decision was made it is not allowed to rediscuss it". This may be again due to the time and budget constraints of the projects: there is simply not enough time and resources to continue reflecting on past decisions; the architects need to consider them finalized and move on.

In general, architecture evaluation seems to be less important for practicing architects than the other two activities. This assumption is supported by the fact that only 16 out of 95 comments to Q19 (the most important thing in decision making) concerned architecture evaluation (31 for architectural synthesis, 48 for architectural analysis). Only three architects mentioned the necessity for reviews. Additionally to this finding, we observed that architects do not seem to pay particular attention to risks. In the answers to Q19, the word "risk" was not mentioned at all. In a survey with Dutch software architects, Clerc et al. also found that risk assessment was not regarded particularly important [16]. However, some of our results show that architects at least unconsciously perform risk mitigation, for instance by reflecting on identifying the requirements that are hardest to fulfill (Q4) and preferring well-known solutions in favor of unknown alternatives (Q10). In question 20, six architects explicitly answered that requirements should be prioritized and that the most important ones should be regarded first, which is also a means to minimize risks.

D. Overall architecting process

To understand the reasoning followed within the overall decision-making process, we interpret the findings from open question 20, in which we asked the participants to describe freely how they come from one decision to the next. One of the most frequent comments was that the prioritized requirements guide the decision making process. This, however, does not imply a sequential approach to decision making. Instead, many architects stated that architecture is iteratively refined and improved, which is in line with architecture approaches in the literature [3]. The iterative nature of architectural design is also indicated by the answers to Q17 and Q18. Architects rather quickly develop a first vision of the overall architecture ($<$20% of the time for the complete architecture phase, Q17) and then refine this vision until the architecture is complete without significantly deriving from the initial vision any more (Q18).

As opposed to the architects who used the requirements to imply the order of decisions, the same amount of architects reflected that there was no specific order in decision-making. This is an indication that the decision-making process follows an arbitrary reasoning path; we argue that further research should be conducted to provide practicing architects with effective methods and tools to structure their decision-making sequence.

Finally, it is a noticeable finding that only one respondent named a concrete architecture approach he followed (in his case the rational unified process). Thus the greatest part of the participants does not seem to follow one particular architecture approach from the literature; instead they at least partially adopt architecture activities to define their own customized approach to architecture.

E. Threats to validity

To describe the internal validity of empirical results, it is important to exclude or at least explain confounding variables and other sources of potential bias [27]. Surveys generally bare the risk of poorly controllable variables [28], at least if online questionnaires are used as data collection method. In such cases the only means to control variables is by exclusion or by randomization. In this study we used both: participants who were not sufficiently experienced in software architecture were excluded from the study, and other potential variables were randomized by using the snowballing as sampling technique. Other potential threats to internal validity (especially construct validity) in questionnaires are ambiguously and poorly-worded questions [22]. To mitigate this risk, we pilot-tested our questionnaire in multiple iterations until the respondents understanding of the questions matched our intentions (see Section III for more details).

An addition threat to internal validity is the fact that the answers to the open question Q19 (the three most important things in decision making) could have been influenced by the structured questions we asked before. However, the majority of the answers were complementary to the questions. Few of the answers indeed demonstrate such a correlation, but in these cases, the participants still had to make a choice that reflected their personal behavior.

An additional limitation of questionnaires is the uncertainty whether the participants answer truthfully. We tried to keep this risk low by ensuring the respondents that the participation is voluntary and that no data is gathered that would allow us to draw conclusions with respect to the identity of the respondent. Moreover, if people are not willing to be honest, they usually do not volunteer for such a survey. However, this risk can never be excluded totally.

External validity is the extent to which conclusions can be generalized and capture the objectives of the study [27]. It is primarily concerned with the representativeness of the sample for the target population [28]. The target population of this study is software architects, who have been working in the industry for at least five years and who have been responsible for software architectural design for at least two years. We assume that our findings concerning the reasoning process can be generalized to the population of architects who fit to these sampling criteria. However, one might argue that the reasoning process is not just influenced by the experience of the architect, but also by the characteristics of the software project (e.g. size and domain) and the culture of the company in which the project is done. The demographics of the participants demonstrate that they worked in a variety of application domains and companies, as discussed in the following two paragraphs.

The influence of the company culture is limited by the fact that multiple companies took part, which were not chosen by us directly. We know of at least eight different companies who took part in the study, because respondents from eight different organizations across Europe and the USA sent us e-mails after participating, to state their interest in obtaining the study results. Data about the domain and size of the project that the architects considered in the study was collected in the questionnaire. The average project size was 1441 person-months (1.4 mill SLOCs), which means that mainly large projects were regarded.

The domains of the project included software engineering (17%), embedded systems (13%), transportation (13%), healthcare (11%), realtime (11%), command and control (9%), enterprise computing (9%), telecommunication (9%), finance (8%), e-commerce (6%) and manufacturing (6%). Thus, a wide range of projects from different domains was covered. To understand the influence that the project domain had on the results, we correlated the domains with the dependent variables (Spearman's rho). At the significance level of 0.05 (2-tailed), the domains finance, transportation and healthcare showed correlations. Architects from the finance domain reflected less on identifying which of the requirements were hardest to fulfill (Q4, corr.-coeff: -.291, sig. 0.05), they spent less effort on searching for alternative design options, if they already had a solution in mind (Q8, corr.-coeff: -.306, sig. 0.039) and had less confidence in the soundness of their decisions (Q12, corr.-coeff.: -.303, sig. 0.041). Architects from the healthcare sector more often searched for alternative design options, if they already had a solution in mind (Q8, corr.-coeff: .307, sig. 0.038). In the transportation domain, architects reflected more on identifying which of the requirements were hardest to fulfill (Q4, corr.-coeff: .298, sig. 0.044) and also thought more about the pros and cons of design alternatives (Q9, corr.-coeff: .296, sig. 0.049). However, the fact that only few correlations were found shows that project domains seem to have no significant influence on the reasoning process.

VI. Conclusions and Future Work

We conducted a descriptive survey with industrial software architects from several companies and project domains to get insight in the reasoning process followed during architectural design. The results were interpreted according to the pragmatic constraints in the industry, as well as established architecting approaches in the literature. As explained in Section I, our aim is to define reasoning best practices guiding especially inexperienced architects in the three architectural activities. The following best practices were derived from our results:

- **Architectural Analysis:** A deep understanding of the requirements and the problem space is essential for successful architecting. If possible, architects should get involved in the requirements elicitation to gain a better understanding of the requirements and other architectural drivers like time and budget-constraints. If, for some reason, they cannot get involved in requirements gathering, they should make sure that requirements are not too constraining or unrealistic and eventually negotiate and relax them with the respective stakeholders. Requirements should be prioritized; the most important ones and the ones that are hardest to fulfill should be regarded first, as they bare potential risks. Requirements are an important part of the rationale behind architecture decisions and as such they should be documented adequately.
- **Architectural Synthesis:** It is advisable to search for multiple design options and get to know the solution space well when making decisions. In cases where time and budget is very limited it is sometimes practical to consider less design options, if the architect already has a working solution in mind that has proven itself in prior projects. In cases where multiple design options equally fit to the design problem at hand, it is less risky to stick to a solution the architect knows well. When weighing pros and cons of design options, a colleague can act as a sounding board to make sure that choices are informed and unbiased by personal preference. In cases where multiple decisions have strong dependencies, they can be discussed as a whole, i.e. the total of such strongly-dependent decisions can be treated as a single decision. Finally, as in other design disciplines, simplicity should be a key goal in software architecture; unnecessary complexity should be avoided.
- **Architectural Evaluation:** In architectural evaluation, candidate solutions must be validated against the ASRs to make a decision. In situations, in which a decision cannot satisfy two requirements at the same time, the optimal trade-off between those requirements has to be found. Prototyping design options or combinations of design options can help understanding solutions and provides additional rationale for informed choices. Apart from evaluating design options, the architecture should regularly be evaluated as a whole to ensure consistency between the decisions and to uncover hidden constraints. If this is not possible due to time and budget constraints, it should at least be done once at the end of the architecture phase. A thorough documentation of architecture decisions can reduce the effort needed for their evaluation.

There is one more best practice that spans through all three activities of architecture design and concerns the iterative refinement and improvement of an architecture. Architects should try to develop an overall vision of the complete architecture rather quickly, and then revisit the constituent parts of the vision to finalize the decisions. Decisions from comparable projects can serve as a starting point to develop the vision and can furthermore help to make sure that no important considerations were forgotten.

We are currently developing a documentation framework for architecture decisions to effectively support software engineers in the different activities of architectural design.

In our previous work we started analyzing the reasoning process of inexperienced software engineers [14]. We performed these studies with graduate students who have followed lectures specifically in software architecture and undergraduate students who have not had any software architecture education. This distinction is made to find out in how far software architecture education influences the way students reason about architecture. We assume that students who have had some kind of software architecture training adopt at least some of the practices and methods they were taught, while others are ignored. We plan to use these results and compare them with the findings presented in this article, in order to propose appropriate training material for inexperienced architects.

ACKNOWLEDGEMENTS

The authors would like to thank all respondents of the survey for their participation. Especially, we thank Philippe Kruchten, Antony Tang, Christian Dietrich and Kevin Erhardt for pilot-testing and discussing the questionnaire with us.

REFERENCES

[1] J. Ven, A. Jansen, J. Nijhuis, and J. Bosch, *Design decisions: The bridge between rationale and architecture*. Springer, 2006, pp. 329–348.

[2] A. Jansen and J. Bosch, "Software Architecture as a Set of Architectural Design Decisions," in *Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture*. IEEE Computer Society, 2005, pp. 109–120.

[3] C. Hofmeister, P. Kruchten, R. Nord, H. Obbink, A. Ran, and P. America, "A general model of software architecture design derived from five industrial approaches," *Journal of Systems and Software*, vol. 80, no. 1, pp. 106–126, 2007.

[4] B. Nuseibeh, "Weaving Together Requirements and Architectures," *Computer*, vol. 34, no. 3, pp. 115–117, 2001.

[5] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, 2nd ed. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 2003.

[6] A. Van Lamsweerde, "Goal-Oriented Requirements Engineering: A Guided Tour," in *Proceedings of the Fifth IEEE International Symposium on Requirements Engineering*. IEEE Computer Society, 2001, p. 249.

[7] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, *Pattern-oriented software architecture: a system of patterns*. John Wiley & Sons, Inc. New York, NY, USA, 1996.

[8] G. Muller, "CAFCR: A Multi-view Method for Embedded Systems Architecting. Balancing Genericity and Specificity," Ph.D. dissertation, Technische Universiteit Delft, 2004.

[9] A. Tang, M. Babar, I. Gorton, and J. Han, "A survey of architecture design rationale," *Journal of systems and software*, vol. 79, no. 12, pp. 1792–1804, 2006.

[10] A. Tang and P. Lago, "Notes on design reasoning tactics," Swinburne University of Technology, Tech. Rep., 2009.

[11] A. Tang, M. Tran, J. Han, and H. Vliet, "Design Reasoning Improves Software Design Quality," in *Proceedings of the 4th International Conference on Quality of Software-Architectures: Models and Architectures*. Springer-Verlag, 2008, pp. 28–42.

[12] F. Brooks, *The Design of Design: Essays from a Computer Scientist*. Addison-Wesley Professional, 2010.

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

[14] U. van Heesch and P. Avgeriou, "Naive architecting-understanding the reasoning process of students: a descriptive survey," in *Proceedings of the 4th European conference on Software architecture*. Springer-Verlag, 2010, pp. 24–37.

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

[16] V. Clerc, P. Lago, and H. van Vliet, "The architect's mindset," in *Proceedings of the Quality of software architectures 3rd international conference on Software architectures, components, and applications*. Springer-Verlag, 2007, pp. 231–249.

[17] J. Hoorn, R. Farenhorst, P. Lago, and H. van Vliet, "The Lonesome Architect," *Journal of Systems and Software*, vol. In Press, Corrected Proof, pp. –, 2010.

[18] P. Kruchten, "Controversy Corner: What do software architects really do?" *Journal of Systems and Software*, vol. 81, no. 12, pp. 2413–2416, 2008.

[19] P. Clements, R. Kazman, M. Klein, D. Devesh, S. Reddy, and P. Verma, "The Duties, Skills, and Knowledge of Software Architects," in *Proceedings of the Sixth Working IEEE/IFIP Conference on Software Architecture*. IEEE Computer Society, 2007, p. 20.

[20] W. Bu, A. Tang, and J. Han, "An analysis of decision-centric architectural design approaches," in *Proceedings of the 2009 ICSE Workshop on Sharing and Reusing Architectural Knowledge*. IEEE Computer Society, 2009, pp. 33–40.

[21] N. Mack, C. Woodsong, K. MacQueen, G. Guest, and E. Namey, *Qualitative research methods: A data collector's field guide*. FLI, 2005.

[22] T. Lethbridge, S. Sim, and J. Singer, "Studying Software Engineers: Data Collection Techniques for Software Field Studies," *Empirical Software Engineering*, vol. 10, no. 3, pp. 311–341, 2005.

[23] M. William and J. Donnelly, "The Research Methods Knowledge Base," *Mason, OH: Atomic Dog Publishing*, 2007.

[24] N. Cross, "Expertise in design: an overview," *Design Studies*, vol. 25, no. 5, pp. 427–441, 2004.

[25] A. Jansen, P. Avgeriou, and J. van der Ven, "Enriching software architecture documentation," *Journal of Systems and Software*, vol. 82, no. 8, pp. 1232–1248, 2009.

[26] P. Kruchten, "Tutorial: introduction to the rational unified process®," in *Proceedings of the 24th international conference on Software engineering*, 2002, p. 703.

[27] B. Kitchenham, S. Pfleeger, L. Pickard, P. Jones, D. Hoaglin, K. Emam, and J. Rosenberg, "Preliminary guidelines for empirical research in software engineering," *IEEE Transactions on Software Engineering*, vol. 28, no. 8, pp. 721–734, 2002.

[28] M. Ciolkowski, O. Laitenberger, S. Vegas, and S. Biffl, "Practical experiences in the design and conduct of surveys in empirical software engineering," *Empirical Methods and Studies in Software Engineering*, pp. 104–128, 2003.