Naive Architecting - Understanding the Reasoning Process of Students

A Descriptive Survey

Uwe van Heesch[1,2] and Paris Avgeriou[1]

[1] University of Groningen, The Netherlands
[2] Fontys University of Applied Sciences Venlo, The Netherlands
uwe@vanheesch.net, paris@cs.rug.nl

Abstract. Software architecting entails making architecture decisions, which requires a lot of experience and expertise. Current literature contains several methods and processes to support architects with architecture design, documentation and evaluation but not with the design reasoning involved in decision-making. In order to derive a systematic reasoning process we need to understand the current state of practice and propose ways to improve it. In this paper we present the results of a survey that was conducted with undergraduate software engineering students, aiming to find out the innate reasoning process during architecting. The results of the survey are compared to the existing architecture literature in order to identify promising directions towards systematic reasoning processes.

1 Motivation

One of the responsibilities of software architects is to make decisions, which are usually called architectural decisions [4,12,29] and determine the overall structure and behavior of the system. Making architectural decisions involves understanding and addressing relevant requirements, business goals and issues, identifying and choosing among alternative solutions while adhering to constraints and mitigating risks. Architectural decisions form the basis for all other detailed decisions and are crucial for the success or failure of the whole project. This decision-making process is one of the major challenges during architecting since it requires a lot of experience and expertise by the architect.

Various methods exist to support software architects in their work. Hofmeister et al. derived a common model for architecture design from five industrial approaches [10], including the Rational Unified Process [21] and Attribute-Driven Design [3]. Other approaches deal with documenting the architecture in terms of multiple architectural views or with the help of architecture frameworks [2,7,16]. Furthermore, different methods exist to support the systematic evaluation of architectures [14,18,19]. More recently some approaches propose the documentation of the actual decisions as first-class entities by defining their attributes and relations [4,27]. However, all of these approaches deal with the core part of architecting: prioritizing architecturally significant requirements, selecting architecture patterns, styles and tactics, partitioning the system into components and connectors, assessing the design and documenting the result with architectural views, frameworks and architecture description languages. In contrast, there has been very little research on the reasoning part of the decision-making process; one can only find fragments about sound reasoning in the literature.

Recent work emphasizes the importance of design reasoning and design rationale [4,23]. Ideally a systematic reasoning process can shorten the gap between experienced and inexperienced architects: design reasoning can support designers step-by-step in making sound decisions and subsequently documenting the rationale behind them as first class entities. However, so far, architects are not trained on how to reason: making architectural decisions is often described as an ad-hoc creative process [5,32,33] that relies heavily on the personal experience and expertise of the architect. Research is required to explore the current state of practice in design reasoning and subsequently to find ways to enhance it.

Our work is towards this direction: investigating how the reasoning process takes place and identifying potential areas for improvement. This can be done either by studying beginners (bottom-up) or experienced architects (top-down). In the former case one can establish the baseline reasoning process that is based on common sense instead of experience. In the latter case one can discover best practices in successful architecting examples and synthesize them into an ideal reasoning process. Eventually one can propose an approach to close the gap between the baseline and the ideal process and package it appropriately to train current or future architects.

This paper deals with the former case; we leave the latter case as future work. In particular we have studied the most inexperienced subjects: students of software engineering. We asked 22 students to design an architecture for a large web application and carefully observed their reasoning during this process. After that, the students were interviewed about the way they thought and acted to come up with a software architecture. As a result, we identified the basic reasoning process of inexperienced designers, which we compared to established architecting processes in the literature in order to come up with promising directions for improvement.

The rest of this paper is organized as follows. Section 2 presents related Work. In Sect. 3, the design of the study is introduced. The next section presents an analysis of the results, which are interpreted in Sect. 5. The paper ends with conclusions and directions for further work.

2 Related Work

The survey presented in this paper is related to the software architecture research field, namely architecting processes, architecting practice in the industry and design reasoning.

Hofmeister et al. derive a general model of architecture design from five industrial approaches [10]. They identify the following common activities: *Ar-*

chitectural analysis is concerned with identifying architecturally significant requirements from architectural concerns and system contexts; *Architectural synthesis* is the activity of finding candidate solutions for architecturally significant requirements; Architectural evaluation makes sure that the candidate solutions are the right ones.

Jansen et al. specialize this generic model from the perspective of architectural decisions [13]. They describe the architecting process as a cycle of activities that are followed iteratively until the architecture is complete. In accordance with Hofmeister et al.'s categorization, in architectural analysis, the problem space is scoped down to problems that can be solved by single architectural decisions. Candidate decisions are proposed during architectural synthesis, while decisions are chosen during architectural evaluations, which also entails modifying and describing the architecture in multiple architectural views. In addition to Hofmeister et al.'s approach, which focuses mainly on architecting activities and artifacts, Jansen et al. indicate reasoning processes within the activities.

Various studies have attempted to define the role of software architects in the industry [8,9,17,31]. Clerc et al. have conducted survey-based research [8] to gain insights in the daily working processes of architecture practitioners. They found out that architecture use cases [28] concerning risk assessment and requirements trade-off analysis are not regarded as particularly important by the architects. In contrast, use cases concerned with requirements, architecture design and implementation, and the traceability among these were rated as important. The authors reckon that the architects' workflow follows a linear (i.e. non-iterative) approach to designing architecture that satisfies the requirements subsequently. In a different survey, Farenhorst et al. [9] describe that more experienced architects (in terms of working years) are more often involved in auditing activities and quality assurance. Kruchten defines the typical roles and responsibilities that architects should take in software projects [17]. Besides making architectural decisions, other central activities of architects include maintaining the architectural integrity, risk assessment and risk mitigation. Finally, Clements et al. compare duties, skills, and knowledge of software architects from the perspectives of literature, education and practice [31]. They found that architecture evaluation and analysis are regarded as less important in architecture practice, whereas knowledge of technologies and platforms, as well as technology-related duties are regarded more important in architecture practice than in the literature and education. We will revisit these results on architecting practice and relate them to our findings in Sect. 6.

The significance of design reasoning in software architecture has been recently emphasized. Tang and Lago describe design reasoning tactics [24] to support architects in structuring architectural problems and extracting design issues. In his previous work [23,25], Tang declares the importance of design reasoning and design rationale in the area of software architecture. It supports architects in making well-founded decisions and provides guidance to explore and manage the solution space. They state that the use of a reasoning approach significantly improves the quality of architectural design, especially for inexperienced architects [23].

3 Design of the Study

3.1 Goal

The goal of the study is to get insight into the innate reasoning that students follow while they are architecting. To make this goal more concrete we need to consider the fundamental reasoning activities that take place during the architecting process. As a reference architecting process, we use the one defined in our previous work [13], which explicitly takes into account the reasoning aspects and maps onto the process of Hofmeister et al. (see Sect. 2). We thus refine our research goal into the following three research questions:

RQ1: How do students scope and prioritize the problem space during architectural analysis?
RQ2: How do students propose solutions during architectural synthesis?
RQ3: How do students choose among solutions during architectural evaluation?

RQ1 is concerned with finding out how students scope and prioritize requirements and issues to define concrete problems that are small enough to be addressed by one architectural decision. RQ2 applies to finding candidate solutions based on the problems identified in the previous step. Finally, the aim of RQ3 is to discover how students make choices between the candidate solutions and how they evaluate their choices with respect to previously made decisions. It is noted that the requirements engineering activity, though closely related, is performed before the architecting process and is therefore out of the scope of this study (an initial set of requirements was made available to the students). Furthermore the activity of modifying and describing the architecture (see [13]) was omitted, because of time constraints in conducting the study.

3.2 Study Design and Execution

To find answers to the research questions, a descriptive survey [30] was conducted with students from the seventh semester, in a four-year software engineering programme of study at the Fontys University of Applied Science in Venlo, The Netherlands. At that time, the students had at least 3 years of OO-programming experience from small software development projects withing the study programme. Some of them had additional experience from side jobs. They had followed two lectures (three hours in total) specifically on software architecture. The following topics were covered in this course: the 4+1 architectural views [16], the recommended Practice for Architectural Description of Software-Intensive Systems [2], the concept of architectural decisions mainly using the template by Tyree and Akerman [27] and software architectural patterns [6]. In total, 22 students took part, who were divided into 11 pairs.

To produce an architecting experience, we asked the students to create a new software architecture of a non-trivial software system (later referred to as *phase one*). Right after that, the students were asked to fill in a questionnaire, in order to report about their individual architecting experiences (*phase two*). The questionnaire was designed and evaluated according to [20].

The architecting case used in phase one was a document describing architecturally relevant functional and non-functional requirements for an online selling platform comparable to Amazon.com [1]. The case study included requirements for user management, selling books, multimedia and other products, searching for products, notification of sellers and buyers. The non-functional requirements included interoperability, availability, performance and security. In total, nine functional and nine non-functional requirements were given. The students were explicitly allowed to supplement or modify the given requirements, for example because of specific trade-offs, if they could justify why.

The architecting activity (phase one) in the experiment took 60 minutes. The students were asked to make all necessary architectural decisions and to document the process of decision making in a mind map. The purpose of the mind map (created on flip charts) was to conserve as much reasoning and as many thoughts of the participants during the decision making process as possible. No architecting method was imposed on them, nor did they have knowledge about any existing systematic approach. The students were also asked to document design options and design decisions using a minimal template. Laptops with an internet connection were allowed to search for arbitrary information, e.g. to find design options like software patterns or technologies. To gather data in phase two of the experiment, a group-administered questionnaire (see [26]) was handed out to the students right after they finished phase one. The students used their documented decisions and the mind map as help to reflect on the process while answering the questions. The questionnaire 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 [26]. To mitigate the risk of ambiguous or poorly understood questions that comes along with questionnaires [20], an instructor explained the questions to the participants one by one. That way the students could clarify questions before answering. Table 1 shows a mapping of the questionnaire questions to the research questions formulated in Sect. 3.1. Some questions have a relation to more than one research question; in these cases, the bold-faced 'X' denotes the most relevant research question (except for Q16 and Q17 where all three research questions are relevant). Additionally, two more questions were asked, which do not directly map to the research questions: "Do you have the skills to design and program the given system?" (Q4) and "Are you confident that your decisions and the resulting design are sound?" (Q8).

The participation in the study was mandatory. The students received grades for the architecture documentation on the flip charts. It was clearly communicated to the students that the answers in the questionnaire in phase two were not taken into consideration for the grading. This issue will be further discussed in Sect. 5.1.

Table 1. Question Mapping

Code	**Question**	**RQ1**	**RQ2**	**RQ3**
Q1	Have you understood and considered the given requirements?	**X**		
Q2	Have you reasoned about the most challenging requirements?	**X**		
Q3	Have the quality attribute requirements played a prominent role during the design?	**X**		
Q5	Have you considered alternatives for the decisions you made?		**X**	
Q6	Have you relaxed requirements to have more design options?	X	**X**	
Q7	Have you thought about the pros and cons of each alternative that you have considered?			**X**
Q9	Have you preferred well-known solutions rather than searching for better alternatives?		**X**	
Q10	Have you sometimes made multiple decisions at the same time?		X	**X**
Q11	Have you rejected decisions?			**X**
Q12	Have you made trade-offs, while making decisions, between multiple requirements?	X		**X**
Q13	Have you come across dependencies between decisions?		X	**X**
Q14	How long did it take since you had a first architectural vision in mind?			**X**
Q15	Does the final architecture significantly differ from your initial vision?			**X**
Q16	How have you come from one decision to the next decision?	**X**	**X**	**X**
Q17	What has gone on in your head when you have thought about the architecture?	**X**	**X**	**X**

4 Analysis

We use descriptive statistics to visualize the collected data in the analysis. This section contains one subsection for every research question. Subsection 4.4 presents results concerning all three research questions. There are eleven valid datapoints for each question, one for each student pair.

4.1 RQ1 - Architectural Analysis

Questions Q1,Q2 and Q3 from the questionnaire are primarily related to the treatment of architecturally relevant requirements during architectural synthesis. Figure 1 shows a stacked bar chart presenting cumulative percentaged frequencies of answers to the respective questions. The vast majority of the participants ($> 90\%$) affirmed that they understood and considered the given requirements (Q1). The median answer was ‘affirmation’. The answers to question two, concerning the reasoning about the most challenging requirements, do not show a clear trend (Q2, median ‘neutral’). More than 80% affirmed that quality attribute requirements played a prominent role during the design (Q3, median ‘strong affirmation’).

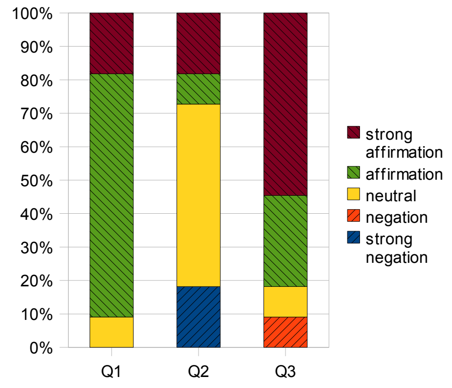

Fig. 1. Cumulative frequencies of answers to questions related to RQ1

4.2 RQ2 - Architectural Synthesis

Figure 2 shows the frequencies of answers to questions Q5,Q6 and Q9, related to finding candidate solutions during architectural synthesis (RQ2). More than 70% of the participants affirmed that they considered alternatives for the decisions they made (Q5, median 'affirmation'). The majority of participants did not relax requirements to have more design options (Q6, > 90%, median 'strong negation') and, without any negations, more than 70% of the participants affirmed that they preferred well-known solutions rather than searching for better alternatives (Q9, median 'affirmation').

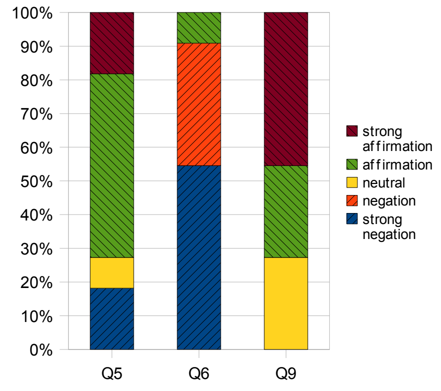

Fig. 2. Cumulative frequencies of answers to questions related to RQ2

4.3 RQ3 - Architectural Evaluation

Questions Q7, Q10-Q15 refer to the making of choices between candidate solutions and the evaluation of the choices with previously made decisions. Figure 3 shows the frequencies of answers. The answers to Q7, referring to the consideration of pros and cons of alternative solutions, show a clear tendency towards affirmation (median 'affirmation'). Q10, related to the making of multiple decisions at the same time, does not receive a clear result. Although the most frequent answer was 'affirmation', the median answer was 'neutral'. 100% of the participants negated the question about rejecting decisions (Q11, median 'strong negation'). The students also did not consciously make trade-offs between requirements (Q12, $> 60\%$, median 'negation'). The answers to question 13 concerning dependencies between decisions do not show a clear tendency (median 'neutral'). Question 14 did not have predefined answers. The participants were asked how long it took in minutes since they had a first architectural vision in mind (Q15 refers to this vision). On average, the participants took 13,36 minutes for a first vision. The standard deviation is 9,067 (min: 5min, max: 30min). Finally, without a single affirmative answer, more than 70% negated that the final architecture significantly differed from the initial architectural vision (Q15, median 'negation').

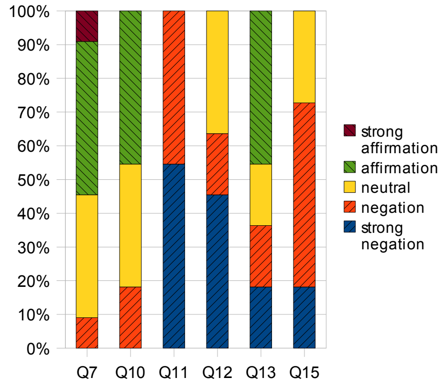

Fig. 3. Cumulative frequencies of answers to questions related to RQ3

4.4 Open questions concerning the whole architecting process

Besides structured questions, we asked the participants to answer two open questions (Q16 and Q17) that concern all three research questions.

In question 16, we asked the students to describe how they got from one decision to the next decision. Four of the pairs stated that they made decisions along the requirements (e.g. "Reading requirements one by one"). Two groups mentioned that they used common combinations of technologies as orientation in the decision making process (e.g. Spring as web framework, then Hibernate as object-relational mapper), one group explicitly stated that they first created a list of things to be decided and then made the decisions one by one.

In question 17, the students were asked to freely describe what went on in their heads when they thought about the architecture. The following workflow of decision making can be derived from the given answers: Analyze requirements, find candidate solutions based on own experience, search for alternative solutions, evaluate pros and cons of all candidate solutions, make decision. Exemplary verbatim answers are: "We started with own knowledge and experience, then we thought about alternatives and made pros and cons lists.", "We thought about what was necessary to fulfill requirements, we thought about known technologies, we tried to find some alternatives for these", "Based on the requirements we think about the decisions to take. Then we think of known solutions/technologies and research on further solutions. Finally we evaluate the different possibilities and make the decisions".

5 Interpretation

In this section, the behavior of the students is interpreted and compared to existing approaches in the architecture literature. The section is organized according to the three architecting activities. Findings on Q16 and Q17 that concern all three activities are mentioned where appropriate.

Architectural Analysis Architectural analysis involves articulating [10] and scoping down [13] architecturally significant requirements (ASR). The quality attribute requirements play a prominent role in this activity [3,11]. Usually the ASRs are further prioritized [13] to identify key issues or problematic requirements [11,21] that require special attention, because they are critical for the architecture. They sometimes become risks [21].

The analysis of the students' results showed that most of them intuitively followed these activities. They tried to understand and consider the ASRs and put emphasis on the quality attribute requirements. The only discrepancy is that many students did not identify the most challenging requirements, nor did they prioritize them. It is noticeable that the students do not seem to be aware of risks and consequently do nothing to mitigate them. However, two student pairs strongly affirmed that they did think about the most challenging requirements. A correlation analysis (Kendall's tau [22]) showed that students who affirmed the statement also had strong confidence in the soundness of their resulting designs (Q8) (corr.-coefficient 0,618, sig. 0,023), which allows the conclusion that risk assessment leads to higher confidence in the quality of the architecture.

Architectural Synthesis Architectural synthesis is the process of finding candidate architectural solutions that (partially) address the distilled ASRs [10,13].

This activity requires the architect to identify and distill relevant knowledge from own experience and external knowledge repositories [10,24], needed to create design solutions. To have more design options, it is sometimes advisable to relax requirements that put too many constraints on possible solutions [24].

In the study, the students affirmed that the identification of design options was driven by the requirements. However, they did not relax requirements to have more design options and they also declared that they preferred well known solutions in favor of unknown alternatives. Also, they do not seem to be aware of limitations and constraints that solutions impose on other decisions. Their answers to the open questions reflect that the requirements were used as a kind of checklist to ensure that all of them are covered by at least one solution without taking into account the relationships and dependencies between decisions. A similar behavior was also observed for practising architects by Clerc et al., who state that the architects' workflow follows a linear approach that satisfies the requirements sequentially [8].

Architectural Evaluation During architectural evaluation the candidate solutions are weighed against the ASRs [10] to make a design decision. Therefore, the pros and cons of each design option have to be considered [13,24]. Choosing solutions can entail making trade-offs [10,13] between requirements. This activity also involves identifying and documenting constraints that decisions impose on future decisions [3]. Evaluation further ensures that a decision does not violate previously made decisions. Therefore the architecture is regularly evaluated as a whole after a few iterations [11]. Some approaches emphasize the need of risk assessment during architectural evaluation [21,24] to ensure that no hidden assumptions or constraints behind decisions exist and to assess if additional risks are introduced by a decision.

The study shows strong deviation of the students' behavior from these activities. Although they weighted pros and cons for the design options, they did not consciously make trade-offs between requirements and also neglected to validate the decisions against each other. This explains why the students did not reject decisions. They do not seem to be aware of dependencies and relationships between architectural decisions. Only few students stated that they came across dependencies. In line with these observations, the students quickly came up with a first architectural vision ($13mins$) and did not significantly deviate from this vision any more. This is another indicator that students do not critically evaluate their decisions. This is not very surprising. As mentioned in Sect. 2, Clerc et al. [8] found out that even practising architects do not regard risk assessment and requirements trade-off analysis as particularly important.

Additionally, it was observed that no clear statement was made about the question if they made multiple decisions at the same time. Some students described that they used a kind of reference architecture they knew from comparable projects as a basis, others started from scratch and made decisions strictly sequentially. A correlation analysis (Kendall's tau [22]) showed that students who made multiple decisions at the same time also relaxed requirements to have more design options (corr.-coefficient 0,584, sig. 0,045).

5.1 Threats to validity

In this section, possible limitations of the study are presented by discussing internal validity, construct validity and external validity [15,22].

With respect to internal validity, the questionnaire design and the fact that an instructor verbally explained the questions before they were answered ensured that the questions were unambiguous and focused on the research questions. Furthermore, the fact that the study was done as a classroom assignment introduces a potential risk. The students received grades for the performance in phase one of the study. Although the questionnaires were not taken into consideration for the grading, some students might have tried to impress the lecturer by giving specific answers. This risk, however, is considered rather low: no evidence in favor of it could be found in the results; and it was not possible to determine which answer would be rated positively or negatively.

Concerning construct-validity, the fact that only one specific architecting experience was used as a basis for the study introduces the risk that the cause construct was under-represented. The architecting process could be different for other architecting case studies. In this study, the students already had experience building simple web applications. In totally unknown domains, they would have been forced to uncover design options they did not know before. However, the risk is regarded as rather low as working in unknown domains is unrealistic especially for inexperienced designers. It can further be assumed that the architecting process for the used system is representative for those of large and medium-size software projects. We also used multiple variables to cross-check the results concerning the research questions. The risk of researchers bias was mitigated for the most part, as the structured questions with pre-defined answers do not leave space for interpretation. However, some open questions do exist that were interpreted by the researchers.

With respect to external validity, the subject population in the study might not be representative for the larger population of inexperienced software architects. The participants of the study were undergraduate students in the last year of a software engineering study programme. Their state of knowledge is comparable to the lowest level of architecture knowledge that software architects in practise have. Thus, it can be assumed that this risk is mitigated.

The instrumentation used in phase one of the study might have been unrealistic or old-fashioned. This risk was mitigated by creating a working environment that corresponds to those of practicing architects. The students were allowed to use laptops with internet connections without any restrictions and they could discuss all issues with their partners. In real software projects however, additional constraints (e.g. time, cost, corporate culture, politics) exist that can hardly be simulated in a classroom environment.

6 Conclusions and Future Work

To gain insights into the innate reasoning processes of students during architectural design, we conducted a descriptive survey with software engineering students. The architecting process the students followed was compared to existing architecture practices in the literature.

The comparison showed that the students' activities during architectural analysis closely match with the activities advocated in existing architecture approaches. However, during architectural synthesis and architectural evaluation large discrepancies were observed. As pointed out, some of these were also observed in studies with professional architects, which leads to the conclusion that the problems do not only result from the low level of experience. To move towards a systematic reasoning process, we list the areas that need to be improved and invite the research community to work on providing the necessary methodological and tooling support:

– Prioritize requirements [13] and identify risks in terms of the most challenging requirements [11,21] that are hard to fulfill.
– Relax requirements to have more design options, where required [24].
– Search for alternatives, even if known solutions exists that seem to solve the design issue.
– Document why one option was chosen over another one [24] to ensure that design options were not only chosen because of personal bias towards known solutions.
– Reason about possible limitations and constraints that solutions impose on future decisions [3].
– Actively consider relationships and dependencies between decisions [11,12].
– Identify situations, in which decisions cannot satisfy two requirements at the same time. Try to find optimal trade-offs between the requirements [10,13,14].
– Determine constraints that decisions impose on future solutions [3].
– Assess and actively mitigate risks throughout the architecting cycle [17].

We hypothesize that systematic support in these areas can verifiably improve the reasoning process and we plan to conduct controlled experiment to test these hypotheses. As mentioned in the introduction, we also plan to study the reasoning practices of successful architects, in order to derive an ideal reasoning process and compare it with the aforementioned areas. However one major issue remains: finding and identifying suitable design options during architectural synthesis is a task that requires the combination of experience and personal design knowledge with new knowledge and unknown design solutions. This task is highly creative and dependent on personal skills and experience and we doubt whether it can be fully supported by systematic architecting approaches.

7 Acknowledgements

We would like to thank the students from the JEE course 2009 at the Fontys University of Applied Sciences Venlo for taking part in the study.

References

1. Amazon.com. http://www.amazon.com, 2010.
2. IEEE-Std-1471-2000. Recommended Practice for Architectural Description of Software-Intensive Systems. Technical report, IEEE, 2000.
3. L. Bass, P. Clements, and R. Kazman. *Software architecture in practice*. Pearson Education, 2003.
4. J. Bosch. Software architecture: The next step. *Lecture notes in computer science*, Springer, pages 194–199, 2004.
5. J. Bosch and P. Molin. Software architecture design: evaluation and transformation. In *IEEE Conference and Workshop on Engineering of Computer-Based Systems, 1999. Proceedings. ECBS'99*, pages 4–10, 1999.
6. F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal. *Pattern-Oriented Software Architecture, Volume 1: A System of Patterns*. John Wiley & Sons, Inc. New York, NY, USA, 1996.
7. P. Clements, D. Garlan, L. Bass, J. Stafford, R. Nord, J. Ivers, and R. Little. *Documenting software architectures: views and beyond*. Pearson Education, 2002.
8. V. Clerc, P. Lago, and H. Van Vliet. The Architect's Mindset. In *Software Architectures, Components, and Applications*, Springer, pages 231–249, 2007.
9. R. Farenhorst, J. F. Hoorn, P. Lago, and H. V. Vliet. The Lonesome Architect. In *Joint Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture (WICSA/ECSA 2009)*, Cambridge, UK, September 14-17, 2009.
10. C. Hofmeister, P. Kruchten, R. Nord, H. Obbink, A. Ran, and P. America. Generalizing a Model of Software Architecture Design from Five Industrial Approaches. In *Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture*, IEEE Computer Society, 2005.
11. C. Hofmeister, R. Nord, and D. Soni. *Applied Software Architecture*. Addison-Wesley Professional, 2009.
12. 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*, pages 109–120., 2005.
13. A. Jansen, J. Bosch, and P. Avgeriou. Documenting after the fact: Recovering architectural design decisions. *The Journal of Systems & Software*, Elsevier, 81(4):536–557, 2008.
14. Kazman, R., Klein, M., Barbacci, M., Longstaff, T., Lipson, H., Carriere, J.: The architecture tradeoff analysis method. In: ICECCS, Published by the IEEE Computer Society (1998)
15. B. Kitchenham, S. Pfleeger, L. Pickard, P. Jones, D. Hoaglin, K. El Emam, and J. Rosenberg. Preliminary guidelines for empirical research in software engineering. *IEEE Transactions on Software Engineering*, 28:721–734, August 2002.
16. P. Kruchten. The 4+ 1 view of architecture. *IEEE software*, 12(6):45–50, 1995.
17. P. Kruchten. What do software architects really do? *Journal of Systems and Software*, 81:2413–2416, 2008.
18. Kazman, R., Bass, L., Webb, M., Abowd, G.: SAAM: A method for analyzing the properties of software architectures. In: Proceedings of the 16th international conference on Software engineering, IEEE Computer Society Press (1994) 81–90

19. Williams, L., Smith, C.: PASA SM: a method for the performance assessment of software architectures. In: Proceedings of the 3rd International Workshop on Software and Performance, ACM (2002) 189
20. T. Lethbridge, S. Sim, and J. Singer. Studying software engineers: Data collection techniques for software field studies. *Empirical Software Engineering*, 10(3):311–341, 2005.
21. P. Kruchten. The Rational Unified Process An Introduction. *Addsion-Wesley Publishing Company*, 2000.
22. F. Shull, J. Singer, and D. Sjøberg. *Guide to Advanced Empirical Software Engineering*. Springer-Verlag New York, Inc. Secaucus, NJ, USA, 2007.
23. A. Tang, M. Babar, I. Gorton, and J. Han. A survey of architecture design rationale. *Journal of systems and software*, Elsevier, 79(12):1792–1804, 2006.
24. A. Tang and P. Lago. Notes on design reasoning tactics. Technical report, Swinburne University of Technology, 2009.
25. A. Tang, M. Tran, J. Han, and H. Vliet. Design reasoning improves software design quality. *QoSA 2008, LNCS*, 5281:28–42, 2008.
26. W. Trochim. *The Research Methods Knowledge Base.* Atomic Dog Publishing, 2001.
27. J. Tyree and a. Akerman. Architecture Decisions: Demystifying Architecture. *IEEE Software*, 22:19–27, 2005.
28. J. Van Der Ven, A. Jansen, P. Avgeriou, and D. Hammer. Using architectural decisions. *2nd International Conference on the Quality of Software Architectures (QoSA 2006)*, Västerås, Sweden, 2006
29. J. Van Der Ven, A. Jansen, J. Nijhuis, and J. Bosch. Design Decisions: The Bridge between Rationale and Architecture. *Rationale management in software engineering*, Springer, pages 329–348, 2006.
30. C. Wohlin, M. Host, and K. Henningsson. Empirical research methods in software engineering. *Empirical Methods and Studies in Software Engineering*, Springer, pages 145–165, 2003.
31. P. Clements, R. Kazman, M. Klein, D. Devesh, E. Reddy, P. Verma. The Duties, Skills, and Knowledge of Software Architects. *in Proceedings of the Sixth Working IEEE/IFIP Conference on Software Architecture*, 2007.
32. U. Zdun. Systematic pattern selection using pattern language grammars and design space analysis. *Software Practice and Experience*, John Wiley & Sons, Inc., 37(9):1016, 2007.
33. O. Zimmermann, U. Zdun, T. Gschwind, and F. Leymann. Combining Pattern Languages and Reusable Architectural Decision Models into a Comprehensive and Comprehensible Design Method. *Seventh Working IEEE/IFIP Conference on Software Architecture (WICSA 2008)*, pages 157–166, 2008.