2014 IEEE/IFIP Conference on Software Architecture

Industrial Implementation of a Documentation Framework for Architectural Decisions

Christian Manteuffel*, Dan Tofan*, Heiko Koziolek†, Thomas Goldschmidt† and Paris Avgeriou‡
*University of Groningen, The Netherlands, {c.manteuffel, d.c.tofan}@rug.nl
†ABB Corporate Research, Ladenburg, Germany, {heiko.koziolek, thomas.goldschmidt}@de.abb.com
‡University of Groningen, The Netherlands, paris@cs.rug.nl

***Abstract*—Architecture decisions are often not explicitly documented in practice but reside in the architect's mind as tacit knowledge, even though explicit capturing and documentation of architecture decisions has been associated with a multitude of benefits. As part of a research collaboration with ABB, we developed a tool to document architecture decisions. This tool is an add-in for Enterprise Architect and is an implementation of a viewpoint-based decision documentation framework. To validate the add-in, we conducted an exploratory case study with ABB architects. In the study, we assessed the status quo of architecture decision documentation, identified architects' expectations of the ideal decision documentation tool, and evaluated the new add-in. We found that although awareness of decision documentation is increasing at ABB, several barriers exist that limit the use of decisions in practice. Regarding their ideal tool, architects want a descriptive and efficient approach. Supplemental features like reporting or decision sharing are requested. The new add-in, was well-perceived by the architects. As a result of the study, we propose a clearer separation of problem, outcomes, and alternatives for the decision documentation framework.**

I. Introduction

The perspective of looking at software architecture as a set of architecture decisions is widely recognized [1], [2]. However, architecture decisions are often not explicitly documented in practice but reside in the architect's mind as tacit knowledge and are only implicit in the models that the architect creates [2], [3]; even though explicit capturing and documentation of architecture decisions has been associated with a multitude of benefits, such as avoiding knowledge vaporization, supporting change impact estimation, increasing system understanding, improving knowledge sharing, and facilitating architecture evaluation [2], [3], [4]. This gap between industry practice and the benefits demonstrated by research persists, although researchers have proposed many approaches to incorporate the documentation of architecture decisions in architecture practice [5].

However, multiple studies identified that these tools are often only applicable during certain stages of the architecting process and have been designed to only support a particular activity or only work under certain conditions [6], [7], [8]. In addition, van Heesch et al. [9] argued that the proposed approaches, i.e. decision templates, decision models, and annotations, "do not frame all concerns of all stakeholders in an adequate and useful manner". This makes the produced documentation only suitable for a small subset of stakeholders. Furthermore, existing decision documentation approaches do not integrate well with existing architecture documentation approaches (e.g. models, diagrams, views) [9].

In a research-industry cooperation with ABB, we developed an add-in for Sparx Systems' Enterprise Architect (EA) that allows to document architecture decisions in a user-friendly and efficient way. The add-in is based on a validated conceptual decision documentation framework, which proposes a set of viewpoints to holistically address stakeholder concerns and to provide support for all architecting phases. This framework has been described and extended by van Heesch et al. [9], [10]. By integrating a decision documentation tool into an existing modeling platform, architects can use the same tool that they use to create models, diagrams and views of the architecture, to document the decisions that constitutes these artifacts. This narrows the gap between architecture documentation and architecture decision documentation.

To ensure that our tool addresses the needs of software architects, it is important to understand the domain and company-specific characteristics and approaches toward software architecture and decision documentation. Therefore, we conducted an exploratory case study to assess the status quo of architecture decision documentation, to identify expectations of the ideal documentation approach, and to evaluate the advantages and limitations of the developed tool from the perspective of software architects. This paper reports on this exploratory case study.

The contribution of this paper is threefold. We describe the current approach and practices of decision documentation within a large company and contribute to the understanding of decision documentation practices in industry. We identified a set of key characteristics for architecture decision documentation (ADD) tools from the perspective of software architects. We present a tool to document architecture decisions, which is based on a validated theoretical framework and has been assessed in an industrial setting.

The rest of the paper is organized as follows. Section II introduces the developed add-in and the conceptual framework. Section III reports on the industrial case study. Section IV discusses the findings in relation to previous findings. Section V presents related research and tools. The paper ends with conclusions and directions for future research in Section VI.

II. Introducing the Add-in

To facilitate the documentation of architecture decisions in practice, we developed a new tool, which has been implemented as an add-in to Sparx System's Enterprise Architect. EA is a general-purpose model-driven UML tool that can be used to create various types of models to support the

978-1-4799-3412-6/14 $31.00 © 2014 IEEE
DOI 10.1109/WICSA.2014.32

architecting process. Enterprise Architect was chosen as target platform because it is widely-used in industry and because it is ABB's tool of choice for modeling aspects of their software architectures. The add-in is a concrete implementation of a conceptual framework for documenting architecture decisions. Although the conceptual framework has been validated in case studies [9], [10], there is no tool that fully implements the conceptual framework and is suitable for industrial use.

In Section II-A we elaborate on the conceptual framework while in Section II-B we describe the features and the functionality of the add-in

A. Conceptual Framework

The Decision Documentation Framework enables the creation of decision models that can capture implicit architectural knowledge [9]. The framework consists of five decision viewpoints, each satisfies different stakeholder concerns. In general, it is not necessary to document all views but only those that answer project-relevant concerns. The validation of the framework demonstrated that the effort to create the views is reasonable and has a low cost-benefit ratio. Moreover, it showed that the viewpoints are particularly useful for communicating decisions and reviewing architectures.

The framework shares information between the viewpoints via a common meta-model. This meta-model is aligned to the meta-model of the ISO/IEC/IEEE 42010 standard and therefore, the framework can be easily combined with other architectural viewpoints, such as Kruchten's 4+1 views[11].

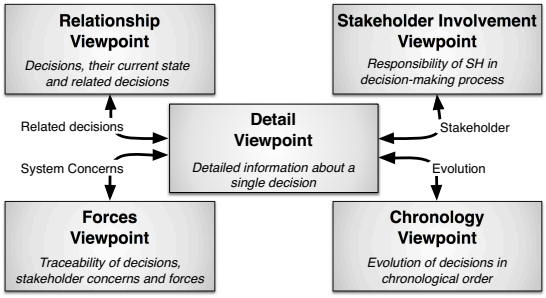

Fig. 1. The five viewpoints of the decision documentation framework.

In the following, we briefly present each viewpoint, starting with the *decision detail viewpoint*. The detail viewpoint presents an overview of one decision inspired by existing decision templates, similarly to Tyree and Ackerman [3]. As illustrated in Fig. 1, it aggregates information stored in the other four viewpoints, such as the evolution of the decision or related forces. While the detail viewpoint focusses on one particular decision, the other four viewpoints visualize multiple decisions to optimally frame related concerns [9].

The *decision relationship viewpoint* illustrates dependencies between decisions and their current state (e.g. idea, decided, rejected). Examples of relationships between decisions are *caused by*, *replaces* or *is alternative for*, e.g. MySQL replaces SQLite. The *stakeholder involvement viewpoint* shows the responsibilities of stakeholders in the decision-making process by relating stakeholder actions on decisions to concrete project phases, e.g. it identifies the stakeholder that proposed a certain alternative at a particular milestone. The *decision chronology viewpoint* is the only viewpoint with a temporal component, which captures the evolution of decisions throughout time. The viewpoint presents both, the individual evolution of a particular decision (e.g. from an *idea* to a stakeholder *approved* decision) and the chronology of all decisions (e.g. the decision to use MySQL has been decided after SQLite was rejected). The *forces viewpoint* relates architecture decisions to stakeholder concerns and to decision forces [10]. A force is any aspect that influenced the architect when making the decisions out of multiple alternatives, such as requirements or experience.

B. Features and Functionality of the Add-in

Implementing the framework as an add-in has the advantage that the tool can benefit from the existing functionality of the platform, such as creating diagrams, managing projects or having a central project repository. Additionally, the learning curve is lower since the users are already familiar with the platform. In order to benefit from these advantages, we carefully designed the add-in to meet the existing usage patterns and practices of Enterprise Architect and consistently integrated it into the existing user interface. For example, the add-in follows a model-based paradigm by having decisions as first class model elements, which implies that decisions are stored in a central model repository along with all other Enterprise Architect modeling elements, such as classes or components.

The add-in supports the following high-level use-cases: (i) Documentation of architecture decisions according to the five viewpoints from the conceptual framework. (ii) Tracing of decisions to other modeling elements within Enterprise Architect, such as components, classes or requirements. (iii) Export of decision views in Word, Excel and Powerpoint.

As illustrated in Fig. 2 the user-interface of Enterprise Architect is composed of three parts. The project browser (right) contains all elements and diagrams of one project in a hierarchical structure of packages and views. The diagram (center) is the workspace to model and to layout views. The toolbox (left) shows a list of elements and connectors, which can be dragged and dropped onto the diagram. The selection of elements in the toolbox depends on the type of the selected diagram. In this case, the toolbox is related to the decision relationship view. Hence, it contains decision elements and connectors to model relations between decisions.

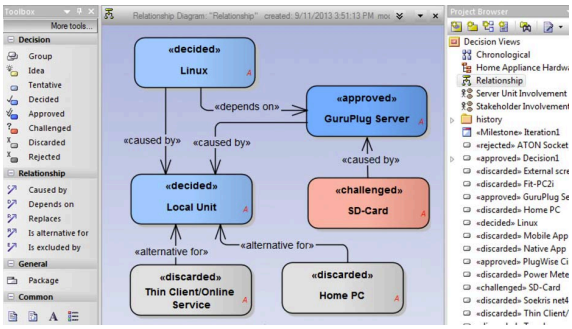

Fig. 2. A decision relationship view in Enterprise Architect with the main interface elements: the toolbox, the diagram and the project browser.

1) Decision Relationship Viewpoint: The decision relationship viewpoint allows to model dependencies between decisions. As illustrated in Fig. 2, a decision is represented by a rounded rectangle with the name of the decision as label. The state of the decision is shown as stereotype (i.e. ≪state≫) and is also encoded in the color of the decision to get a faster overview of the overall state of the decisions. The colors were selected carefully to express a semantic meaning. For example, *challenged* decisions are indicated in red to emphasize that this decision represents an unsolved problem and requires further discussion. *Discarded* or *rejected* decisions are displayed in an unobtrusive grey, while *ideas* resemble the bright yellow of a light bulb. *Tentative*, *decided* and *approved* decisions are visualized in a gradient from light blue to dark blue to illustrate the increasing certainty of a decision. A tentative state mean that a decision is seriously considered. A decided state reflects the current position of the architect, while an approved state means that the decision has been confirmed, e.g. during a review meeting.

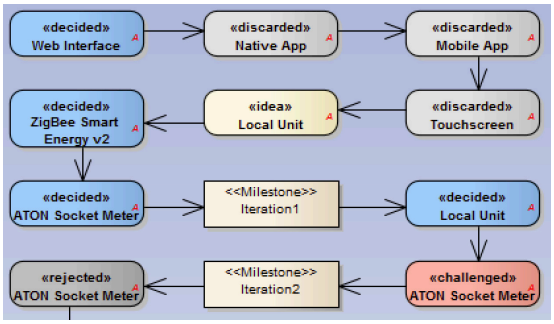

Fig. 3. A decision chronology view showing the evolution of decisions for two milestones.

2) Chronology Viewpoint: The decision chronology viewpoint shows the evolution of architecture decisions over time. It shows decisions that have been made, current set of decisions, the ordering of decisions, and obsolete decisions, as shown in Fig. 3. However, keeping track of every state change of a decision can be difficult. In order to reduce the effort required to create this view, the add-in keeps track of changes to automatically generate a chronology view.

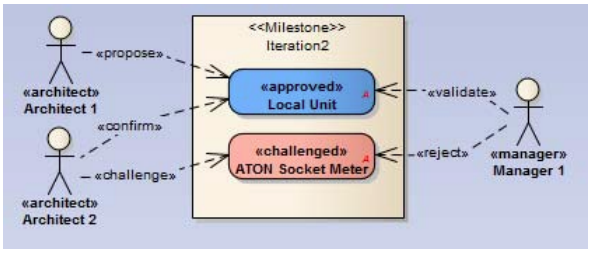

Fig. 4. A stakeholder involvement view showing the actions of two stakeholders (architect, manager) in the decision-making process.

3) Stakeholder Involvement Viewpoint: The stakeholder involvement viewpoint explains the responsibilities of specific stakeholders in the decision-making process. It shows decisions, actions, and stakeholders involved within one architecture iteration. The responsibilities can be modeled by relating decisions and stakeholders via special connectors (e.g. *proposed*, *validated* or *decided*), as illustrated in Fig. 4. This viewpoint supports a knowledge personalization strategy (i.e. *who-knows-what* instead of *what-is-known*).

	Concern	<<approved>> GuruPlug Server	<<discarded>> Soekris net4501	<<discarded>> Zotac ZBOX SD-ID12 Plus
ZigBee for commnication	Feature Support	Yes, via dongle	Yes, via dongle	Yes, via dongle
TR-1.2 Support 50 devices	Performance	+ (1.2 GHz C...	- - (133 MHz ...	++ (1.8 GHz C...
TR-1.3 WLAN Connectivity	Feature Support	Yes	Yes, but requi...	Yes
CF-1 Unobstrusive hardware	Marketing	++	-	+
CF-2 Target price	Marketing	+ 95EUR	- 143EUR	- - 175EUR

Fig. 5. A concrete decision forces view, showing forces and concerns, alternatives and ratings.

4) Forces Viewpoint: The decision-forces viewpoint makes relationships between decisions and decision forces explicit. This viewpoint identifies the impact of forces on decisions, conflicting influence of forces on a specific decision, rationale behind a decision, and to what concerns a decision pertains. As shown in Fig. 5, the decision forces viewpoint is a table that lists the forces on the vertical dimension and the decisions on the horizontal dimension. Just like the other viewpoints the table can be created via drag and drop.

5) Detail Viewpoint: The Detail viewpoint is the default dialog that is shown when a decision has been selected. The decision detail viewpoint combines the information shown in all other views by giving detailed information about a single architecture decision including its rationale. It uses a set of common attributes, such as the state of the decision, issue, alternatives, and related concerns. It also aggregates all incoming and outgoing relationships of a decision shown in the relationship viewpoint. The detail viewpoint addresses a large set of concerns, like conflicting impacts between decisions, dependencies, relations, the involvement of stakeholders, and the history of a specific decision. Fig. 6 shows the detail view of a decision.

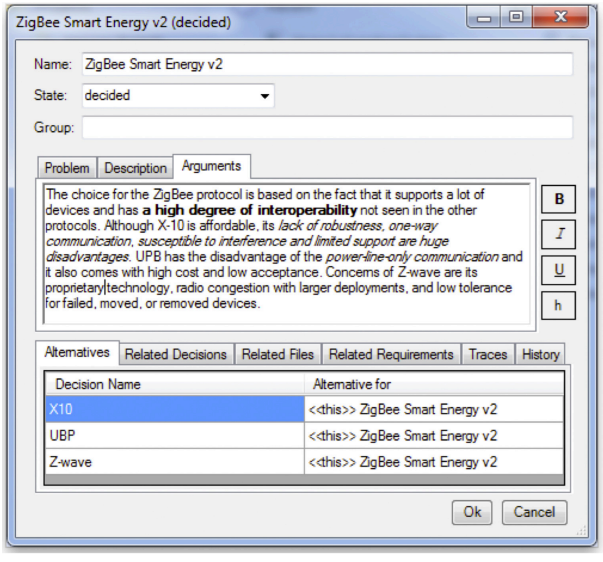

Fig. 6. A concrete decision detail view illustrating the arguments and alternatives of decision.

6) Traceability: The add-in provides a user-friendly way to create traces between decisions and other modeling elements. A trace can be created by adding the decisions in the same diagram as the component. Subsequently, a trace relationship between the two elements can be created by using the Trace-connector in the toolbox, as shown in Fig. 7. Even if the decision is removed from the diagram, the trace still persists. The tracing is bi-directional, so the user can navigate from decisions to design elements and vice versa through the context menu of the decision element. This allows to easily relate a decision to the underlying design model, e.g. a decision about a database to the actual data layer in the system.

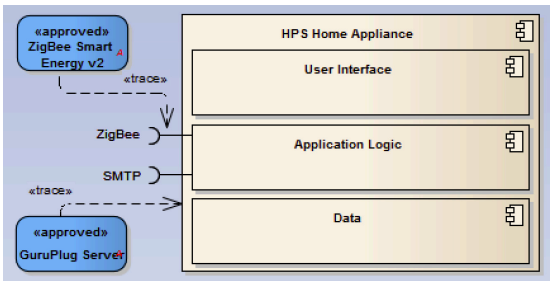

Fig. 7. Trace between decision and baseline component.

7) Reporting: Three types of documents can be generated out of the decision documentation: Word, PowerPoint, Excel. The Word and PowerPoint reports present all viewpoints, while the Excel report only contains forces viewpoints.

III. Case Study

The main research goal of our study is to analyze *the decision viewpoint add-in* for the purpose of *evaluation* with respect to *advantages and limitations* from the point of view of *software architects* in the context of the ABB company environment. This goal has been formulated according to the Goal Question Metric approach [12].

In order to assess the advantages and limitations of the add-in, it is necessary to compare the add-in to the current approaches of decision documentation within ABB but also to analyze to which extent the add-in matches the expectations of ABB software architects. Therefore, we need to understand how architecture decisions are currently documented, how they are used, and which challenges architects face during this activity. Only if we understand the status-quo of decision documentation at ABB, we can reliably assess if the add-in is an improvement. Furthermore, we need to understand what architects perceive as an ideal solution for tool-supported ADD, to compare to which extent the add-in fulfills these expectations. Based on our study goal (i.e. evaluate the decision viewpoint technology), we derived three research questions.

- RQ1 What is the status-quo of architectural decision documentation?
- RQ2 What is the ideal architecture decision documentation approach from the perspective of architects?
- RQ3 How do architects perceive the advantages and limitations of the add-in and the decision documentation framework?

In Section III-A we elaborate on the method, scope and units of analysis of the study. Section III-B reports on the background of the participants, while Section III-C describes the data collection procedure. In Section III-D we present the findings of the case study.

A. Method and Case

A case study was preferable over surveys or experiments, because the use of the add-in is tightly linked to the context of the software project, for which decisions are documented. The context-specific factors, such as complexity, domain, size or number of stakeholders, influence not only the quantity and quality of documented decisions but also the willingness of the architect to use such a tool. Therefore, we had to study the phenomena (i.e. the use of the add-in) in its natural context[13] The focus of this case study is a preliminary investigation of the add-in to seek new insights and to generate new ideas for our research, thus we conducted an exploratory study. We chose a single case, embedded design. In embedded case-studies "multiple units of analysis are studied within a case" and are an appropriate design for explorative case studies when there is no established theoretical framework[13]. We followed the guidelines for conducting case study research described by Runeson et al. [13].

Our concrete case is the documentation of software architecture decisions by five architects of different business units of a large industrial-automation company. Our unit of analysis is the decision documentation activity performed by individual architects. The object of our study is the developed add-in. The scope of our study is limited to the business units of ABB to which the participating architects belong.

B. Participants and Projects

All five participants can be classified as experienced software architects with substantial practical experience. The participants had between 10 and 24 years of experience in software development and three to 18 years of experience in the field of software architecture (average 11 years). The participants have experience in the the following domains, Information Systems, Enterprise Applications, Control Systems, Process Automation. Table I summarizes the software project that were reported during the study. The first column indicates the role(s) of the participant within a project. The second column is a brief description of the context of the project, while the third column reports about the number or frequency of documented architecture decisions within a project.

TABLE I. Role and projects of participating architects.

Role	Project	Number of decisions documented
Lead Architect, Project Manager	Corporate consulting project, internal client, one year duration	20-30 decisions
Software Architect	Process automation system, long-term project	Every 1-2 week a decision
System Architect	Industrial control systems, long-term	5-6 in three years
Lead Architect	Large corporate research project	15 in first six month
Software Architect, Project Leader	Industrial software architectures, two project with one year duration	First project 20 - 30, second project 30 - 40 decisions.

C. Data Collection

As recommended by Verner et al. [14], we used multiple sources of data. These include existing decision documentation of ABB, interviews with ABB architects, and direct observation of ABB architects using the add-in. The selected sources of data are commonly used in case studies [15]. Table II shows the mapping between selected sources of data and defined research questions.

TABLE II. MAPPING BETWEEN RESEARCH QUESTIONS AND SOURCES OF DATA.

	RQ1	RQ2	RQ3
Requirement elicitation interviews	×	×	
Existing decision document	×		
Pre-tool session interview	×	×	
Observations from tool session		×	×
Post-tool session interview			×

1) Requirement elicitation interviews: Eliciting requirements for the tool from software architects, helped us to understand both the status quo of AD documentation and the ideal tool. Due to the geographically dispersion of participants and researchers, the requirements elicitation interviews were conducted via telephone. We used semi-structured interviews to elicit requirements. We recorded the interview if the participant had agreed. If this was not the case, one of the researcher took extensive notes. One architect filled out the interview guide without being interviewed.

2) Existing decision documentation: Analyzing existing decision documentation helped us to understand the status quo of AD documentation at ABB. Moreover, it enabled us to collect details to corroborate information collected from other sources. The documentation that we received from ABB contained architectural design decisions for two systems and are based on real products of ABB. However, the decisions have been obfuscated to hide any product-specific information or component names. We investigated what kind of information has been documented (e.g. problem, solution, how many alternatives were considered, relationships, stakeholders, etc.). We also received templates that were used to document decisions in concrete projects, modeling guidelines, and Excel forms for evaluating alternatives against concerns.

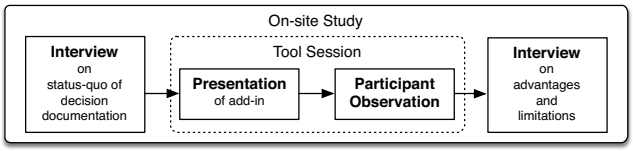

Fig. 8. Outline of a session with an individual software architect.

3) Tool sessions: Most of the data collection took place during a company-visit by three researchers. We conducted individual sessions with five architects. All sessions followed the same schema as illustrated in Fig. 8. We first interviewed the participant about the status-quo of decision documentation. The first interview was followed by a brief presentation about the add-in. Three of the participants used the tool to document decisions of a recent software project. At the end of each session, we conducted another interview about their experience and their opinion about the advantages and limitations of the add-in. Semi-structured interviews before and after the tool-sessions helped us to better understand the context in which the add-in is evaluated, e.g. information about the project, the team or the client. Each session was audio recorded and transcribed.

D. Analysis

We used constant comparison for data analysis, which is a systematic way to identify concepts in the data by comparing identified concepts to previous observations[16]. All interviews and the collected documentation were intensively studied and coded. "Coding in this context means attaching codes, or labels, to pieces of text which are relevant to a particular theme [...] of interest in the study[17]." We did not establish a set of codes before the analysis as the objectives of the analysis were not clear ahead of time. Instead, potential codes were announced during analysis and explained and discussed in a shared document. Responses of the participant that were related to one of the research questions were labeled. During the coding and constant comparison the researcher gets a better understanding of the data he is looking at [16]. The identified codes were grouped into related higher level concepts. We compared the collected codes and the related text passages and combined those that had commonalities. In our case, a concept is a representation of a pattern of behavior, a recurring statement or an expression of an opinion, e.g. about the usefulness of a certain viewpoint. After repeating step two and three multiple times, a set of concepts emerged. The concepts were intensively studied and discussed. Related concepts formed a category (cf. Table III).

In the following sub-sections we present the findings for each research question.

1) RQ1 - Status-Quo of Decision Documentation: All architects were familiar with the concept of architecture decisions and they documented them in their daily routine (**c1**). In fact, it was mentioned that the awareness for architecture decision documentation has increased over the past three years. However, the number of documented decisions varied and did not depend on the duration or the size of a project, e.g. one project had six decisions in three years while another had 40 in one year (cf. Table I).

The participants reported that they only document the *"major"* decisions of the system (**c2**). However, there was no common description of properties and characteristics that are suitable to determine whether a decision can be classified as *"major"* or not. Often they were just referred to as *"high-level"*, *"fundamental"* or *"big"* decisions. One mentioned that these are the decisions *"that are causing discussion between stakeholders and require common agreement"*.

There was no company-wide guideline for documenting decisions. However, the approach for decision-making was similar. At first, the problem was identified, followed by a preparation phase, in which the requirements were analyzed, stakeholder were interviewed, and the design space was explored (**c3**). After the preparation phase, the collected information was used as input for the decision. The final decision was eventually made either by a single architect or by a group, depending on the project. In some projects, the decisions were approved or reviewed during stakeholder meetings.

TABLE III. MAPPING BETWEEN CATEGORIES AND PARTICIPANTS.

	Architect 1†	Architect 2	Architect 3	Architect 4	Architect 5†	Ex. Documents
(**c1**) Familiar with architecture decisions	×	×	×	×	×	
(**c2**) Primarily documented "major" decisions	×	×	×			
(**c3**) Intensive-phase of preparation	×	×			×	
(**c4**) Less time required to document decisions	×	×		×	×	
(**c5**) Tools and templates used for decision documentation	×	×		×	×	×
(**c6**) Maintenance of documentation is a problem	×	×				
(**c7**) Different information and fields of decisions captured	×	×				×
(**c8**) No direct perceived benefits of ADD	×		×	×	×	
(**c9**) Decisions are used for communication	×	×	×	×	×	
(**c10**) Lean and flexible documentation	×	×	×	×	×	
(**c11**) Tool should not patronize user	×					
(**c12**) Limited time for documentation	×	×	×	×		
(**c13**) Compatibility with project-specific guidelines		×	×	×		
(**c14**) Integration into existing platforms	×	×	×	×	×	
(**c15**) Sharing and reusing decisions		×	×	×		
(**c16**) Stakeholder specific reports		×				×
(**c17**) Envisioned time savings		×			×	
(**c18**) Potential early adopter	×			×	×	
(**c19**) Viewpoints are good to separate concerns	×	×			×	
(**c20**) Forces views. useful for technology evaluation		×	×	×	×	
(**c21**) Limited features of forces viewpoint	×					
(**c22**) Limited benefits of stakeholder involvement viewpoint		×	×	×	×	
(**c23**) Accountability of stakeholder inv. viewp. is problematic	×	×	×			
(**c24**) Approved chronology viewpoint	×	×		×		
(**c25**) Effort vs. benefits of chronology viewpoint					×	
(**c26**) Ambiguous scope of chronology viewpoint	×		×			
(**c27**) Relationship viewpoint provides good overview		×	×	×	×	
(**c28**) Semantic of state, relations and traces unclear	×	×	×		×	
(**c29**) Detail viewpoint is very useful	×		×	×	×	
(**c30**) Integration into Enterprise Architect is good	×	×	×	×	×	
(**c31**) Approved traceability	×	×	×	×	×	
(**c32**) Problems with decision model	×	×	×	×	×	

† Did not participate in requirement elicitation.

It was reported that a lot of effort is spent on the preparations for those *"major"* decisions, which could take from one week to eight months. In contrast, the actual documentation of decisions does not require more than a day (**c4**) . The exception was one elaborated documentation approach, which included multiple iterations and reviews.

There was no agreement on how decisions should be documented or which tools should be used for documentation (**c5**). The following tools were mentioned: PowerPoint, Excel, Word, and Enterprise Architect. These tools are rather ineffective to manage decisions, hence maintaing decisions and keeping track of their progress was reported to be challenging (**c6**). This problem became evident in situations, when related requirements changed or new information about a technology became available. This shows a clear need for proper tool-support, which is illustrated by the following quote: *"Yes, in the absence of good support in tools like Enterprise Architect we fell back to Powerpoint tables."* Furthermore, in one case, decisions were documented as a collection of related documents in combination with a brief summary. The summary presented the outcome of the decision and listed the pros and cons.

Decisions were mostly documented according to decision templates, which were adapted to the specific needs of the project. The templates were inspired by Zimmermann's y-approach[1] or the ISO/IEC 42010 standard [18] (**c5**). This lead to different information being recorded. For example, one template, which was specifically tailored for technology evaluations, explicitly captured alternatives, stakeholders, risks, forces and a revision history of the decision (**c7**).

The documented decisions were primarily used to communicate decisions to non-technical stakeholders (**c9**). For example, decisions were presented on slides during meetings (e.g. architecture reviews). In some cases, decisions were also used to inform developers, but it was mentioned that they are usually only interested in the decision outcome rather than the design rationale.

We found that in projects that did not explicitly used documented decisions as basis for discussions, the direct benefits of decision documentation were perceived lower compared to those project where decisions were discussed with stakeholder, e.g. during review meetings (**c8**). Instead, the architects assume that explicitly documented decisions will become useful much later in the project life-cycle, e.g. to avoid knowledge vaporization or for system understanding. It was reported that due to time-pressure decisions were not continuously documented in these projects. In contrast, in those projects, where the client was actively participating in review meetings, the architect put a lot of emphasis on decision documentation and it was considered a positive return on investment. *"The feedback was very positive much more appreciated than pure component and connector documentation. [. . .] I talked about derived decisions and documented them in a consumable way. That made the review quite efficient."*

Summary: All studied business units document architecture decisions. However, several aspects limit the documentation, use and reuse of decision documentation in practice. Such aspects include a lack of tool-support, the absence of a standardized format and only little guidance on how to utilize decision documentation. We have seen that for some architects, the current approach of documenting decisions does not provide enough direct benefits for their project, compared to the required effort of documentation. The approach towards decision-making is comparable between the units and is characterized by an intensive phase of preparation.

2) RQ2 - Ideal AD Documentation Tool: All architects reported that lean and flexible documentation is important for them (**c10**). The tool should not enforce complete documentation but should offer the possibility to document only certain aspects of a decision. The architect should be able to choose how much documentation is really necessary and which information is critical to understand the rationale of a specific decision (**c11**) . This is supported by the finding that the participants only documented those decisions, which they considered to be important (**c2**).

[1] http://www.sei.cmu.edu/library/assets/presentations/zimmermann-saturn2012.pdf [Online accessed: 25/08/13]

Four of the architects mentioned that hard time constraints of the project limited the number of documented decisions. Because of this, documentation needs to be as efficient as possible (**c12**). Decision documentation approaches with complex semantics, rules and formalisms are not seen as beneficial since a lot of effort has to be spent to be correct and precise. One architect summarized this as follows: *"Do not give them too many options and too many things to do. Try to make them efficient in their work by minimizing the time they have to spent in creating a decent decision log"*. In this context, the participants also mentioned automation as a method to reduce required effort, e.g. by generating parts of the decision documentation out of existing documents. A tool should automatically fill in information that can be easily inferred from the context of the documentation activity, e.g. date and time, author, revision history, etc.

It was reported repeatedly that the decision documentation approach should be adaptable to corporate guidelines and existing development processes (**c13**) . Furthermore, the decision documentation tool should interface with tools that are used during architecting (**c14**), such as modeling tools, requirements management tools or document management systems. Being able to link decisions to existing design artifacts is seen as important, as documentation can be speed up by pointing to information instead of duplicating data, e.g. pointing to results of technology evaluations. Furthermore, linking decisions and other artifacts could reduce maintenance effort as changes can be synchronized with the decision documentation and thus would allow to perform sanity checks to keep the documentation up-to-date. For instance, a warning could be issued if a requirement has been changed, which is linked to a decision.

As indicated by (**c16**), the tool should be able to generate stakeholder-specific reports out of the stored decisions. This correlates with the fact that the main purpose of decision documentation within the observed units was stakeholder communication (**c9**). It was reported that a lot of effort is spent to create and optimize decision reports for non-technical stakeholders. The group of stakeholders usually do not require the level of technical detail that a developer would expect. As stated by one participant: *"Stakeholder are really on the other side, you have marketing people and management there and you really need to adapt to their mind"*. On the contrary, developers might not be interested in financial aspects of a decision.

We also identified a need to effectively share architectural knowledge across business units, in particularly the results of technology evaluations (**c15**). A lot of time is spent on evaluating technologies against certain criteria. It was mentioned that a shared repository of these evaluations could provide valuable insights on how other architects have assessed a technology and which factors and risks they have considered. *" So we can look at other investigations and how these are documented and what they did. If those were easy to find this would help."*

Summary: The following characteristics of the ideal ADD tool emerged during the interviews. It should be lean and flexible, efficient, and compatible with existing processes. Besides documenting decisions, the architects found it very important that the tool interfaces with existing tools, exports stakeholder-specific reports, and provides the possibility to share and reuse decisions.

3) RQ3 - Advantages and Limitations of the Add-in: We received very positive feedback about the add-in. Three of the architects stated that the presented tool would allow them to document decisions faster, compared to their current approach (**c17**). Three participants also mentioned that they would immediately start using a stable and complete version of this add-in (**c18**).

The approach to separate concerns in different decision viewpoints was widely accepted (**c19**). The importance of different perspectives is illustrated by the fact that there were different opinions about the usefulness of individual viewpoints. The same viewpoint was regarded as highly beneficial by some architects and as having no immediate benefit by others. However, each architect considered at least one of the five viewpoints as useful.

The *forces viewpoint* was seen as useful, since it resembles one of the templates used for technology evaluation. Thus, the viewpoint immediately seemed familiar and could be used to capture the results of these evaluations (**c20**). However, we also received comments regarding the limited capabilities compared to the Excel template. The participants were missing features like grouping, filtering or coloring to manipulate and analyze data (**c21**). One architect used a qualitative approach for evaluation (i.e. quality attribute scenarios) and thus wouldn't use the forces viewpoint.

The reception of the *stakeholder involvement viewpoint* was mixed. Four architects reported that they would probably not use this viewpoint in their current project, as they do not see an immediate value (**c22**). On the other hand, one architect saw the advantage *"of having a connection between project management and architecture decision-making"*, which he considered to be the advantage of this viewpoint. However, the accountability introduced by this viewpoint (**c23**) was seen as critical as it might create tension within the team.

The *chronological viewpoint* was regarded to be positive (**c24**). However, it was stated that the effort to create this viewpoint would not justify the insights gained from this viewpoint (**c25**). Hence, the possibility to automatically generate the viewpoint was highly appreciated. However, it was remarked that the viewpoint can get very large with an increasing number of decision and thus, it would become difficult to comprehend the evolution of decisions. It was suggested to reduce the size of the viewpoint by hiding, filtering or merging state changes that are less important (**c26**).

The *relationship viewpoint* was considered very positive (**c27**) as it provides a good overview of the decisions in the project. As indicated by (**c28**) the concept of decision states and the meaning of relations was not intuitively understood by the architects and requires some training. The ability to trace decisions with other modeling elements was highly appreciated by all architects (*"That's a dream coming true"*) (**c31**). Furthermore, the close integration with EA was praised as it helps them to get started (**c30**).

A general concern that was raised by all participants was that the add-in does not clearly separate between problem, alternatives and outcome of decisions (**c32**). The architects wanted to model those aspects of a decision individually, which would resemble their way of working and thinking.

Summary: The add-in was perceived very positively. Having different viewpoints for different concerns was appreciated, especially since there was no agreement on the usefulness of individual viewpoints. This finding is perfectly compatible with the decision framework and the tooling as it is up to the architect which viewpoints to use. Two suggestions for improvements could be identified: limit the scope of the chronology viewpoint and separating problem, alternatives and outcome in the decision model.

E. Limitations

The validity of a study is an indicator of the trustworthiness of the results and to what extent the results are true and not biased by the researcher's subjective point of view [13]. *Internal validity* does not apply because we do not examine causalities in our data [15].

Construct validity reflects to what extent the operational measures that are studied really represent what was intended to be studied. To ensure that our operational measures are suitable, we established a research protocol that was reviewed by two researchers with experience in conducting case study research. We used multiple sources of data (interviews, participant-observations and the analysis of work artifacts) to limit the effects of one interpretation of a single data source [13], this included interviews, participant-observations and the analysis of work artifacts. We also took different perspectives into account and investigated differences between projects and business units, which improves the strength of our conclusions.

External validity reflects the extent of generalizability of the findings, and to what extent the findings are of interest to other people outside the investigated case. Statistically representative samples can typically not be achieved in cases studies and thus, the emphasis is usually put on analytical generalization, which explains why the findings are representative for other cases with common characteristics [13].

The participants of the study were professional software architects with multiple-years of experience in architecting software systems. Moreover, ABB and the studied business units are representative for companies developing software intensive systems. Therefore, we argue that the subject population in the study is representative for the larger population of experienced software architects that have a similar approach to architecting and decision-making as the study participants. There is a risk that the cultural background influenced the results as all participants were German. However, since ABB is an international corporation and since the participants were working in multinational projects in Europe and Asia, we argue that the impact on the results is less substantial.

Additionally, we conducted two pilot-studies to train the interviewers and to identify problems with the questions[13]. In our study design we ensured that the interview about the current state of decision documentation is not biased by the demonstration of the add-in and the decision framework. With respect to our findings for RQ1 and RQ2, our findings are partly generalizable as they are supported by other studies that were conducted in different domains and with different subjects, e.g. [7], [19]. With respect to RQ3, external validity is limited as further replications with multiple-cases are necessary [15].

Reliability is concerned to what extent the operations of a study can be repeated. We addressed reliability by defining a rigor study protocol and interview guides [13]. We created a case study database that contains all collected data both raw and coded, so that the analysis can be verified and traced. We mitigated the risk of biasing our findings towards a positive outcome by asking open-ended questions and also asked the participant to motivate their answers. However, the risk of a researcher bias also affects analysis. Therefore, the analysis was conducted by two researchers and the findings were discussed with a third researcher, who did not participate in the analysis but in data collection, to get an unbiased perspective.

IV. Discussion

In the following sub-sections, we discuss the findings and compare them to existing literature.

A. Lack of Tool-Support and Standard Approach

The analysis of RQ1 showed that architecture decisions are documented within the selected business units, which indicates that the importance of ADD in the architecting process is increasingly recognized by practitioners. The finding is supported by a survey conducted by Tang et al. [19] in 2005 that showed that, contrary to popular believe, decisions are indeed documented in practice although not all aspects of a decision are captured, such as considered alternatives. Our findings showed that the information being recorded differs from project to project. Some architects documented decisions thoroughly with alternatives, justification and consequences, while others focussed solely on the outcome.

An explanation for that could be a missing standardized approach and the use of general purpose tools. In contrast to general purpose tools, a specialized tool can present the architect with a template and provide assistance, thus increasing the quality of decisions and providing a uniform format. Furthermore, general purpose tools are unsuitable for maintaining architecture decisions. This finding is supported by a recent study by Anvaari et al. [20] on decision making in the Norwegian electricity industry. In general, the lack of good tool support might discourage architects to capture knowledge in the first place.

B. Sharing and Reusing Decisions

We have seen in RQ1 that the preparation of a decision requires a lot of effort. This is aligned with the findings from van Heesch and Avgeriou [21] on the reasoning process of professional architects. Despite the effort that is being spent on these activities, we found that decisions are not shared within the company but are only consumed within the context of a project. However, reusing existing decisions as a basis for upcoming decisions could significantly reduce the effort of collecting and researching information. Of course, decisions cannot be applied without a critical reflection since the contexts of projects differ. Nevertheless, decisions can be used as a source of inspiration and information (e.g. "Which alternatives were considered?" or "How where they evaluated?"). Another aspect of sharing is the ability to make decisions available for stakeholders by exporting stakeholder-specific reports. We found that this is an important use-case for architects, since

decisions are often used for communication with stakeholders, e.g. during review meetings.

C. Perceived Benefits, Efficiency and Flexibility

RQ2 showed that there is only limited time available for documentation. This is supported by the study conducted by Tang et al. [19], who identified time-pressure as one of the main barriers for documenting decisions. However, not enough time for documenting decisions can be seen as an indicator that the architects did not see enough direct benefits to make decision documentation a higher priority task. Although, all participants reported long-term benefits of decision documentation, such as avoiding knowledge vaporization, the immediate benefits of documenting decisions were not that evident. Architects only mentioned direct benefits in those projects, in which decisions were immediately used for reviews or where the project focussed on extending the knowledge-base of the company, e.g. creating architecture prototypes of emerging technologies.

With regards to the important characteristics of the ideal ADD tool, we have seen that efficiency is very important. In general, the effort of documenting and maintaing decisions should be as low as possible, which can be achieved with dedicated tool-support and automation. In our case, generating viewpoints would be a possibility to increase the quality of documentation (i.e. completeness) and reduce the required effort.

We identified that a flexible approach is essential. The tool should not patronize its users by enforcing a certain way of documentation. The architect wants to decide when to document, what to document and how much documentation is necessary. This finding is supported by Farenhorst et al. [7], who emphasize that due to the creative nature of architecting, tools should be descriptive rather than being prescriptive.

D. Reinforced Findings about the Framework

One of the motivations of the *decision documentation framework* is that the architect can choose to document only a subset of viewpoints depending on the characteristics of the concrete project and the available time [9]. The analysis of RQ3 showed that this goal has been accomplished as the participants considered different viewpoints to be useful for their project. This emphasizes the importance of providing different perspectives on decisions as it is impossible to address all concerns in a single viewpoint.

Our study confirms several findings of prior studies about the framework. This includes that the stakeholders concordantly stated that the *relationship view* provides a good overview of the decisions made [9]. Furthermore, our study supports the presumption that the accountability introduced by the *stakeholder involvement viewpoint* might cause architects to neglect the use of this viewpoint [9].

E. Improvement of the Decision Model

A result of RQ3 is that a clearer separation between the problem, the alternatives and the outcome of a decision needs to be achieved. The fact that the problem and outcome are embodied in the same element caused difficulties for the participants to model their decisions in the add-in. We observed that the architects start with the identification of a problem, which requires a decision. After that, multiple alternatives are considered and evaluated. Eventually, one or a combination of the alternatives will be chosen. Since the model does not separate the outcome of a decision from the underlying problem and the considered alternatives, it does not adequately represent the architect's thought and work process.

V. Related Work

In the past ten years, different approaches emerged to support architectural knowledge management using architecture decisions as basic knowledge entities. For example, the *Knowledge Architect* tool suite is based on a common knowledge repository, which is accessed by clients to store and retrieve AK [22], e.g. a Word and Excel plugin to annotate AK in existing documents. Another example is the *Architecture Design Decision Support System (ADDSS)*, which is a web-based tool for storing, managing, and documenting architectural decisions along with candidate architectures [23]. The tool allows to visualize the evolution of AK over time by capturing AK and architecture in an iterative process. Furthermore, it supports requirements traceability and reporting.

Another web-based tool is the *Process-centric Architecture Knowledge Management Environment (PAKME)*, which is built on top of a open source groupware platform [24]. PAKME's features can be categorized into knowledge acquisition, knowledge maintenance, knowledge retrieval, and knowledge presentation [6]. Archium, proposed by Jansen et al. [1], takes a different approach by defining a component language extension for Java, which allows to model design decisions and relate those to components. This makes a single language act as an architectural knowledge repository and an implementation of a system.

The proposed tools were evaluated and compared by several studies. In 2010, Tang et al. conducted a study that compared five architectural knowledge (AK) management tools based on a framework, which represented how AK management is used in the architecture life-cycle [6]. They found that there is no tool that supports all activities of the architecture life-cycle, i.e. analysis, synthesis, evaluation, implementation, and maintenance. The results also show that the capability to provide different perspectives on AK is limited, as the tools do not support the concept of views and viewpoints. Furthermore, it was noted that knowledge sharing features are not fully supported by the tools yet. This finding is also supported by an earlier study conducted by Farenhorst et al. in 2007 [7]. In this study, the authors analyzed five tools from academia and industry from a knowledge sharing perspective. The results show that AK personalization strategies are not sufficiently supported and that the tools do not provide the possibility to generate stakeholder-specific content. The latter was also identified by Tang et al. [6]. In 2009, Liang and Avgeriou [8] surveyed nine existing tools to what extent they support the following categories of use cases: consuming AK, knowledge management, intelligent support, and producing AK. They found that the tools lack of the ability to identify stakeholders, to apply decisions to produce the design (synthesis), to evaluate AK, and to support the assessment design maturity.

VI. Conclusion

In this project we gained further insights into the challenges and the context of decision documentation and presented an add-in to document architecture decisions. The add-in was validated in an industrial case study, which has shown that it is very well received by practitioners and that the integration into Enterprise Architect is well done. For example, the ability to create traces between decisions and other modeling elements was regarded as beneficial. The theoretical foundation of the add-in, the decision documentation framework, ensured that the developed add-in is able to holistically address stakeholder concerns and provide support for the various architecting life-cycle phases. Hence, having different viewpoints was appreciated, giving the architect the freedom to choose which viewpoints to use. As suggested by this study, our add-in needs to be extended. Therefore, we propose a clearer separation of problem, outcomes, and alternatives for the decision documentation framework and we suggest to improve the scope of the chronology viewpoint.

The findings of the study showed that awareness of decision documentation is increasing at ABB, but still several barriers exist that limit the use of decisions in practice. Such aspects include a lack of tool-support, the absence of a standardized format and only little guidance on how to utilize decision documentation. Furthermore, we identified a set of characteristics that describe the architect's expectations of the ideal ADD tool.

The results of this project will help future tool development efforts to better understand the problems and the needs of practitioners with respect to decision documentation. Some of the questions that follow from the presented work are: (i) How can we further minimize the effort to document decisions? (ii) How can decisions sharing and reusing decisions support the decision-making process? (iii) How to integrate the separation of problem, alternatives and outcome into the decision documentation framework and the add-in? In the future, we continue to improve the add-in and we plan to conduct a second industrial study to empirically validate the add-in.

Acknowledgment

We would like to thank all participants of the case study. Especially, we would like to thank Spyros Ioakeimidis and Antonis Gkortzis for their contribution to the add-in as well as Zengyang Li and Werner Buck for piloting our study. This research has been partially sponsored by the ABB Software Research Grant Program and the ITEA2 project 11013 PROMES.

References

[1] A. Jansen and J. Bosch, "Software Architecture as a Set of Architectural Design Decisions," in *WICSA '05: Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture (WICSA'05*. IEEE Computer Society, Nov. 2005, pp. 109–120.

[2] P. Kruchten, R. Capilla, and J. C. Dueñas, "The Decision View's Role in Software Architecture Practice," *IEEE Software*, vol. 26, no. 2, Mar. 2009.

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

[4] J. S. van der Ven, A. Jansen, P. Avgeriou, and D. K. Hammer, "Using architectural decisions," in *Proceedings of the 2nd International Conference on the Quality of Software Architectures (QoSA)*. Karlsruhe University Press, 2006, short paper.

[5] M. Babar, T. Dingsyr, P. Lago, and H. van Vliet, *Software Architecture Knowledge Management: Theory and Practice*. Springer, 2009.

[6] A. Tang, P. Avgeriou, A. Jansen, R. Capilla, and M. A. Babar, "A comparative study of architecture knowledge management tools," *Journal of Systems and Software*, vol. 83, no. 3, pp. 352 – 370, 2010.

[7] R. Farenhorst, P. Lago, and H. Van Vliet, "Effective tool support for architectural knowledge sharing," in *Software Architecture*. Springer, 2007, pp. 123–138.

[8] P. Liang and P. Avgeriou, "Tools and technologies for architecture knowledge management," in *Software Architecture Knowledge Management*, M. Ali Babar, T. Dingsøyr, P. Lago, and H. van Vliet, Eds. Springer Berlin Heidelberg, 2009, pp. 91–111.

[9] U. van Heesch, P. Avgeriou, and R. Hilliard, "A documentation framework for architecture decisions," *Journal of Systems and Software*, vol. 85, no. 4, Apr. 2012.

[10] ——, "Forces on Architecture Decisions - A Viewpoint," in *Software Architecture (WICSA) and European Conference on Software Architecture (ECSA), 2012 Joint Working IEEE/IFIP Conference on*, 2012, pp. 101–110.

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

[12] V. R. Basili, G. Caldiera, and H. D. Rombach, "The goal question metric approach," *Encyclopedia of software engineering*, 1994.

[13] P. Runeson, M. Höst, A. Rainer, and B. Regnell, *Case Study Research in Software Engineering: Guidelines and Examples*. John Wiley and Sons, Inc., 2012.

[14] J. M. Verner, J. Sampson, V. Tosic, N. A. A. Bakar, and B. A. Kitchenham, "Guidelines for industrially-based multiple case studies in software engineering," in *Research Challenges in Information Science, 2009. RCIS 2009. Third International Conference on*, 2009, pp. 313–324.

[15] R. K. Yin, *Case Study Research: Design And Methods (Applied Social Research Methods)*. Sage Publications, Inc, 2008, vol. 5.

[16] S. Adolph, W. Hall, and P. Kruchten, "Using grounded theory to study the experience of software development," *Empirical Software Engineering*, vol. 16, no. 4, Aug. 2011.

[17] C. B. Seaman, "Qualitative Methods in Empirical Studies of Software Engineering," *IEEE Transactions on Software Engineering*, vol. 25, no. 4, pp. 557–572, July/August 1999.

[18] "ISO/IEC/IEEE 42010 systems and software engineering — architecture description," Jan 2010.

[19] A. Tang, M. Babar, I. Gorton, and J. Han, "A survey of the use and documentation of architecture design rationale," in *Software Architecture, 2005. WICSA 2005. 5th Working IEEE/IFIP Conference on*, 2005, pp. 89–98.

[20] M. Anvaari, R. Conradi, and L. Jaccheri, "Architectural decision-making in enterprises: Preliminary findings from an exploratory study in norwegian electricity industry," in *Software Architecture*. Springer Berlin Heidelberg, 2013, pp. 162–175.

[21] U. van Heesch and P. Avgeriou, "Mature architecting-a survey about the reasoning process of professional architects," in *Software Architecture (WICSA), 2011 9th Working IEEE/IFIP Conference on*. IEEE, 2011, pp. 260–269.

[22] P. Liang, A. Jansen, and P. Avgeriou, "Knowledge architect: A tool suite for managing software architecture knowledge," University of Groningen, Tech. Rep., 2009.

[23] R. Capilla, F. Nava, S. Pérez, and J. C. Dueñas, "A web-based tool for managing architectural design decisions," *SIGSOFT Softw. Eng. Notes*, vol. 31, no. 5, Sep. 2006.

[24] M. Babar and I. Gorton, "A tool for managing software architecture knowledge," in *Sharing and Reusing Architectural Knowledge - Architecture, Rationale, and Design Intent, 2007. SHARK/ADI '07: ICSE Workshops 2007. Second Workshop on*, 2007, pp. 11–11.