The Architect's Mindset

Viktor Clerc, Patricia Lago, and Hans van Vliet

Department of Computer Science
VU University , Amsterdam, the Netherlands
{viktor,patricia,hans}@cs.vu.nl

Abstract. Software architecture and software architecture practices become increasingly important for information systems since they enable reasoning on the design of the system. The concept of *architectural knowledge*, i.e. architectural design decisions and the resulting design, plays a pivotal role in architecture. In order to get the most out of architectural knowledge, we need insight into the ways in which architectural knowledge is used. Currently, we lack this insight. We performed survey-based research in the Netherlands to collect feedback on the importance of architectural knowledge for the daily work of practitioners in architecture. We present our findings using two perspectives: the architectural roles practitioners fulfill and the architecture level practitioners are engaged in. We use these perspectives to construct and reflect on the architect's mindset on architectural knowledge. This mindset of architects reveals an approach which is focused on 'to create and communicate' rather than 'to review and maintain' an architecture.

Keywords: software architecture, architectural knowledge.

1 Introduction

A software architecture is a transferable abstraction of a system and allows for communication of that system to different stakeholders [1]. Software architecture and software architecture practices are gaining importance since they enable reasoning on the design of the system and verifying quality attributes of a system at an early stage in the development cycle.

Rather than viewing the software architecture as a set of components and connectors, recent literature regards the software architecture as the set of architectural design decisions [1,2]. The collection of architectural design decisions and the resulting design together constitute *architectural knowledge* [3]. Besides providing insight into the current software architecture, architectural knowledge also caters for the 'why' of the software architecture, its rationale.

To get the most out of the architectural knowledge of information systems in general, we need to determine in what way different stakeholders use architectural knowledge. We term these typical uses *use cases for architectural knowledge*. Some of these use cases may depend on the roles that stakeholders fulfill or the architecture level stakeholders are engaged in. E.g. architects may favour other use cases than designers or technical specialists, and enterprise architecture practitioners may give priority to other use cases than software architecture

S. Overhage et al. (Eds.): QoSA 2007, LNCS 4880, pp. 231–249, 2007.

practitioners. Currently, we determine what information is particularly important for certain stakeholders, by using approaches and standards such as [4,5]. But we do lack insight into what intention these stakeholders have with architectural knowledge.

We have conducted a survey-based study to address the lack of insight into the importance of architectural knowledge. We designed a survey which includes use cases for architectural knowledge. These use cases are based on earlier work [3,6], experiences in industry, and our own experience.

This paper reports on the results of our study. We provide insight in the way practitioners in the Netherlands view and use architectural knowledge. In doing this, we reveal the mindset of practitioners with respect to the use of architectural knowledge by listing what uses are important for what roles and on what architecture levels.

Based on the survey results, we make the following observations. Architects regard the architecting process as a forward engineering discipline and do not see clear benefits of reflection, assessment, and change of the architecture. Yet, literature argues that these are precisely the intended benefits of architecture (e.g. [1]). Apparently, these intended benefits of architecture have not yet been firmly established in the mindset of architects nor transferred to practice. Furthermore, a forward decision-making process is reflected by the mindset of architects, but the value of managing the set of decisions ('architectural knowledge management') is not yet clear. Finally, the importance of stakeholder communication of the architecture is generally recognized.

The results of this research call for further knowledge transfer on the more innovative concept of viewing architecture as architectural decisions. Furthermore, it is important to quantify the benefits of this concept. At the same time, further research is needed into the foundation for the mindset to identify the activities needed to further establish the concept of architectural knowledge in the architect's mindset.

The remainder of this paper is organized as follows. Section 2 discusses related work in the field of architectural knowledge, design rationale, and architectural roles. Next, Sect. 3 describes the design of the research. Sections 4 and 5 present the findings and a discussion of their limitations. Section 6 reflects on the results and provides conclusions. Finally, Sect. 7 provides directions for future work.

2 Related Work

In recent work [7], the use and documentation of architecture design rationale has been analyzed. The survey reveals the view of the participants on several generic uses of design rationale. The study shows that although participants regard design rationale as important, they do not capture the rationale. The main reason for this is a lack of appropriate tools to support the architects. Furthermore, the survey shows that architects tend to focus on the positive aspects of architectural decisions and design rationale instead of looking for problems in

a specific architecture. We view design rationale as a specific subset of architectural knowledge [3]. We revert to the use cases for architectural knowledge, initially described in [3] and elaborated on in [6], and provide a detailed view on possible uses of architectural knowledge.

A template for architectural decisions is provided by [8]. This template describes a decision, its underlying assumptions, related requirements, and implications. The template is useful for organizing architectural knowledge but does not provide insight into the use of architectural knowledge. Multiple templates or tailoring of existing templates may be necessary to fully support architects in their use of architectural knowledge. Our work can provide input for this. Zimmermann et al. [9] report on a framework that can be useful in identifying, making, and enforcing architectural decisions *during* the architecting process.

The meaning of architecture in practice has been extensively described in [10]. Smolander describes four metaphors for architecture: 'architecture as blueprint', 'architecture as literature', 'architecture as language', and 'architecture as decision'. The metaphors explain the meaning of architecture in practice. This meaning may differ depending on the role practitioners fulfill or the architectural levels they are engaged in. We provide insight into the importance of use cases for architectural knowledge and relate this insight to the metaphors from [10]. Thus, we show what metaphors are accepted and in use by practitioners.

A good description of possible roles and activities of an architect is given in [11]. Examples include a visionary, technical consultant, decision maker, and coach. These different activities could be supported by appropriate use cases for architectural knowledge. Use cases for architectural knowledge that are deemed important for aforementioned roles enable the architect to effectively capture and reason about architectural knowledge. The other way around, the relevance of use cases for architectural knowledge for the practitioners could indicate to what extent the possible roles and activities from [11] in fact are established.

Clements et al. [12] performed a study on publicly available resources to identify the duties, skills, and knowledge of software architects. They show that the role of an architect implies more than only making technical decisions. Rather than focusing on the competences of architects, our study focuses on the use of architectural knowledge to support the architect in satisfying the competences.

3 Research Design

We aim to find out how practitioners engaged in architecture in the Netherlands view architectural knowledge. This helps us to construct the mindset of architects with respect to architectural knowledge. In order to reveal this mindset, we use a survey instrument. Survey instruments are widely used in software engineering research [13]. In survey-based research, a number of factors should be taken into account: the design of the survey, the selection of participants, and how to

control the response rate of the survey. The remainder of this section describes how we have addressed these factors.

3.1 Survey Design

The ultimate objective of the survey is to identify the way practitioners view and use architectural knowledge. We took the typical uses distilled from experienced practitioners within the GRIFFIN research project [3,6] as a starting point. We validated and augmented the list of use cases with the industrial partners in our research project and with our own experience. Next, we reformulated each use case into a clear, self-explaining statement on architectural knowledge. We allowed the survey participants to indicate the importance of each use case using a 5-point Likert scale [14] ranging from 'not important' to 'very important'.

We hypothesized that the importance of certain use cases may depend on the roles practitioners fulfill and on the level of architecture that practitioners are engaged in. We posed some demographic questions and specifically asked the participants to *a)* indicate the architectural roles they typically fulfill and *b)* indicate the relative amount of time they spend on certain architecture levels. Using the information on the roles and architecture levels, we constructed two different perspectives to analyze the way respondents perceive architectural knowledge.

As a first step, we conducted a pilot with the survey on a focused group consisting of selected employees of one of our industrial partners. We then developed a web-based survey for administration of the complete population.

3.2 Selection of Participants of the Survey

We needed to construct a representative subset of Dutch practitioners that play a key role in architecture. To come to this subset, we identified three dimensions by which we selected participating organizations: domain (e.g. banking, telecommunications, insurance, governmental), type (IT service providers, software development organizations), and market (commercial, non-commercial). Next, we selected organizations or platforms (such as *communities* of enterprise architects and embedded architects) in each of these dimensions and identified practitioners that play a key role in architecture at these organizations. This gives us confidence that we have selected a representative subset of Dutch practitioners to give us feedback on the use of architectural knowledge.

3.3 Control of the Survey

In order to keep control on the response rate, we directly contacted key practitioners at the organizations involved. We enquired their willingness to act as on-site representative for that organization. We sent these representatives the hyperlink to the on-line survey. The on-site representatives forwarded the hyperlink to knowledgeable colleagues and notified us of the total number of colleagues involved (*snowball sampling* [13]).

4 Findings

This section describes our findings. We first provide demographic information in Sect. 4.1, after which we discuss the two different perspectives for presenting the results as described in Sect. 3.1. Next, Sect. 4.4 describes how we clustered the use cases for architectural knowledge. After that, we present our main results: the way practitioners view and use architectural knowledge.

4.1 Demographic Information

We sent out the survey to 36 persons acting as on-site representatives. They forwarded the survey to 348 practitioners. In total, 384 practitioners were reached. We collected 143 responses, of which 107 were complete. This corresponds to a response rate of 27.86 %. We took these as a basis for our survey.

Of the total population, 213 practitioners are employed at one of the large IT service providers included in our survey. We discuss the overrepresentation of practitioners employed at IT service providers in Sect. 5. We received 75 responses from these practitioners, corresponding to a response rate of 35.21 %. The remaining 171 practitioners are employed at smaller IT consultancy firms or e.g. IT departments of banks, insurance organizations or governmental organizations. We received 32 responses, corresponding to a response rate of 18.71 %.

4.2 Architecture Levels

A list of definitions of architecture [15] shows that different views on architecture exist. Examples include a systems-oriented view and a view focusing on the information flow in or surrounding a software system. In our survey we used concise definitions from [16] for the so-called *architecture levels*:

software architecture the structure and relations of a software system.
systems architecture the architecture of a single system, taking into account both software and hardware.
information architecture the information needs and flows of business functions as they are identified.
enterprise architecture architecture at the level of an organizational unit or an organization as a whole.
process architecture a description of the processes running in or surrounding a software system.

Each practitioner indicated the amount of time spent on a certain level of architecture. To be able to compare the data collected, we normalized the total amount of time spent by each practitioner to 100 %. The relative amount of time spent on each level of architecture for all respondents is depicted in Fig. 1. We observed that a concrete architecture of a single system (i.e. 'software architecture' and 'systems architecture') receives more attention than company-wide architectures (i.e. 'information architecture', 'enterprise architecture', and

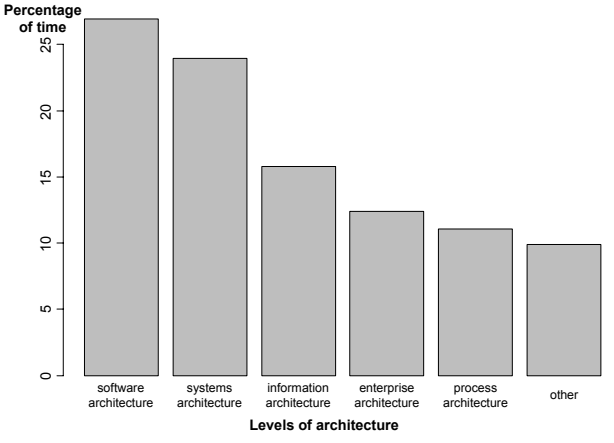

Fig. 1. The percentage of time spent by all respondents on architecture levels

'process architecture'). Other architecture levels that are less often practitioned, are grouped into the *Other* category in Fig. 1.

Since the population is relatively large, we grouped the respondents into clusters and based our analysis on the clusters instead of on the individual responses. Of course, architects may work on different architecture levels simultaneously. We wanted to see if these architecture levels are related based on the responses of the practitioners. E.g. if architects often work on two architecture levels simultaneously, we group these two levels. Moreover, this enables us to group architecture levels that have different names, but in fact appear to be closely related. In order to group similar or closely related architecture levels, we calculated the correlation between each of the architecture levels. This resulted in an n-dimensional space, n being the number of architecture levels. In order to plot the relative distances between the architecture levels, we reduced the number of dimensions to two using *classic multi-dimensional scaling* [17]. In order to assess the accuracy of the distance, we applied *k-means clustering* [18] to cluster architecture levels with a small distance. We observed that the optimal distribution of architecture levels in clusters occured when we used five clusters. We compared the clusters that appeared with *k-means clustering* with the distance plot provided by *classic multi-dimensional scaling*. The comparison revealed that we selected the right number of clusters and that we found the correct distribution of elements over the clusters. Consequently, the clusters contain elements that are different in nature and have overlap reduced to a minimum. We observed the elements in these clusters of architectural levels and labeled the clusters. The results are shown in Table 1.

Table 1 shows that distinct clusters provide for the relation of a software system and the hardware it runs on (*Systems Architecture*), the structure of a software system (*Software Architecture*), the structure of the organization or department using the software system (*Enterprise Architecture*), and the process and

Table 1. Clusters of architecture levels

Cluster label	Levels of architecture
Systems Architecture	systems architecture
Software Architecture	software architecture, management of architecture
Enterprise Architecture	enterprise architecture
Information and Process Architecture	information architecture, process architecture
Other	development coach, integration architecture, infrastructure architecture, service architecture, maintenance of architecture, solution architecture

information flow in or surrounding a software system (*Information and Process Architecture*).

Of the most significant architecture levels, only 'information architecture' and 'process architecture' are very often worked on by a single respondent simultaneously. Consequently, they fall into the same cluster. The remainder of the most significant architecture levels each fall into a distinct cluster.

Practitioners can potentially work on different levels of architecture simultaneously. In spite of that, apparently practitioners do not do this. They are specialized in working at one specific level of architecture only. Possible reasons for this are the different technical and interpersonal skill-sets required at each architecture level. For example, practitioners who mainly work on the level of *Systems Architecture* are concerned with CPU performance, interrupt levels, and other technical topics, whereas practitioners who mainly work on the level of *Process Architecture* are concerned with implications of decisions on working processes, which places less requirements on technical skills. Required interpersonal skills can vary at different architecture levels as well. As the topics that require to be communicated get less technical, the potential audience could grow. Consequently, the set of stakeholders with which to communicate grows from technology-oriented stakeholders to include more business-oriented stakeholders.

4.3 Architectural Roles

The participants indicated the architectural roles they typically fulfill. The survey contained a list of roles, including 'architect', 'reviewer of architecture', 'project manager', and 'developer'.

We repeated the same analysis as described in Sect. 4.2 to identify clusters of roles typically fulfilled by a single respondent. The optimal distribution of architectural roles in clusters occurred when we used five clusters. Again, we labeled the clusters. The results are listed in Table 2.

The clusters labeled *High-level* and *Low-level* show that, apparently, architectural roles are related based on level of abstraction with respect to a software system and practitioners work at one specific level of abstraction. Our results

Table 2. Clusters of architectural roles

Cluster label	Architectural roles
Communicator	architectural educator, project leader
Low-level	designer, developer, reviewer of source code
Specialist	consultant, technical specialist
High-level	architect, reviewer of architecture
Other	end user, lead architect, security consultant, other

also show that practitioners generally do not switch between the different levels. This contradicts with the view on the role of a software architect as an implementor [11]; according to our survey, architect do not design or implement that often. Furthermore, the roles 'architectural educator' and 'project leader' share a communication responsibility towards a variety of stakeholders. Consequently, we label this cluster *Communicator*.

4.4 Clustering the Use Cases

We listed the use cases for architectural knowledge from [6] and asked the practitioners to indicate the importance of each use case for their daily work. We used the answers of participants of the use cases to reveal underlying structure in the use cases. The structure would excavate similarities between use cases based on the answers and allow us to cluster the use cases accordingly.

First, we used principal components analysis [19] to identify the underlying structure in the use cases for architectural knowledge based on the respondents' answers. It turned out that no underlying structure could be found; the variance in the scores of the use cases was explained by one main principal component.

Since the principal components analysis did not lead to a clustering of the use cases, we next tried to cluster the use cases based on the purpose of the individual use cases. Most use cases for architectural knowledge could be clustered relatively easily. E.g. some use cases clearly dealt with stakeholders only. Consequently, we grouped these use cases into a single cluster. For some use cases, clustering was more difficult. These use cases could be grouped into multiple clusters (e.g. 'add an architectural decision' could point at a forward architecting approach, but at the same time assumes that a set of architectural decisions exists to which the new decision is added as well – see Table 3). We identified the most appropriate cluster for these use cases by analyzing the questionnaire results of the participants for these use cases. We compared the answers on a use case with the average of the answers for each candidate cluster. We assigned the use case to the cluster with the highest similarity in answers (see Sect. 4.5). The interpretation of the survey results also led to the cluster labels. Table 3 lists the resulting clusters of use cases for architectural knowledge.

The use case cluster *Architectural decision set* presupposes that a set of knowledge entities (i.e. architectural decisions) and relations between these knowledge

Table 3. Use cases for architectural knowledge

Use case cluster	Use cases
Architectural decision set	11. View the change of the architectural decisions over time 15. Recover architectural decisions 20. Identify incompleteness 22. Detect patterns of architectural decision dependencies 23. Check for superfluous architectural decisions 24. Cleanup the architecture
Assessment – reqs.→arch.→impl.	1. Check implementation against architectural decisions 5. Check correctness (i.e. architecture versus requirements) 18. Evaluate the impact of an architectural decision 19. Evaluate consistency 27. Get consequences of an architectural decision
Assessment – risk, trade-off analysis	4. Perform a review for a specific concern 16. Perform an incremental architectural review 17. Assess design maturity 21. Conduct a risk analysis 25. Conduct a trade-off analysis
Stakeholder-centric	2. Identify the subversive stakeholder 3. Identify key arch. decisions for a specific stakeholder 6. Identify affected stakeholders on change 7. Identify unresolved concerns for a specific stakeholder 8. Keep up-to-date 9. Inform affected stakeholders 26. Identify important architectural drivers
Forward Architecting	10. Retrieve an architectural decision 12. Add an architectural decision 13. Remove consequences of a cancelled decision 14. Reuse architectural decisions

entities exist (see [3] for a list of possible relations). The use cases in this cluster are aimed at managing that set. Several other use cases have to do with assessing or reviewing an architecture. Within this *Assessment* cluster, we distinguish between use cases that imply a forward-engineering approach to architecture (i.e. from requirements, to architecture, to implementation), and use cases that target at performing different kinds of analyses and reviews. The first set aims at verification of the architecting activities ("are we still on the right track?") whereas the second set aims at validation. Seven use cases form the cluster *Stakeholder-centric*. These use cases concern identification of stakeholders and communication of the architecture to specific stakeholders. The cluster *Forward Architecting*, finally, consists of use cases that create, request, reuse or remove architectural decisions.

4.5 Participants' Views on the Use Cases

Instead of elaborating on each of the 107 responses individually, we took the clusters of architecture levels and architectural roles as described in Sects. 4.2 and 4.3 as two perspectives for analyzing the survey results.

We built a structure to be used to identify the importance of a specific use case for architectural knowledge to a specfic cluster (of architectural roles or architecture levels) as follows. For each respondent i in a cluster with n respondents, we used the Likert scores ($score_i$). Using the relative contribution of the respondent to that cluster ($\%_i$), we calculated the weighted average as shown in (1):

$$score = \frac{\sum_{i=1}^{n} score_i \cdot \%_i}{\sum_{i=1}^{n} \%_i} \tag{1}$$

Next, we identified outliers and intermediate results by defining an upper and lower limit of importance: within the possible range of scores from 1 – 5[1] we regard a use case with a score of $\geqq 3.5$ as 'important' and a use case with a score of $\leqq 2.5$ as 'not important'. The results are listed in Table 4. Each row in Table 4 relates a cluster of use cases for architectural knowledge to both the clusters of architectural roles and the clusters of architecture levels. The importance of each use case cluster for each cluster of architectural roles and each cluster of architecture levels is provided. Important clusters are marked '(+)', not important clusters are marked '(–)'. Impartial results are not listed in the table. The findings are discussed below. An extensive discussion of their implications is given in Sect. 6.

The use cases for architectural knowledge within the cluster *Architectural decision set* assume that a set of architectural decisions is at the practitioner's disposal. In terms of the use cases, architecting thus boils down to managing and manipulating that set of architectural decisions. Table 4 shows that viewing architectural knowledge as a set of decisions has not been established at the *Software Architecture* and *Systems Architecture* levels. Furthermore, viewing the architecture as a set of decisions is regarded as not important for *Communicator* and *Specialist* roles. *High-level* and *Low-level* roles (i.e. 'architects' and 'designers'/'developers') deem these use cases neutral. Apparently, the view on architecture as a set of architectural decisions [2] and managing that set has not yet transferred to practice, nor is it of particular value to the practitioners.

The cluster labeled *Assessment – reqs.→arch.→impl.* covers traceability of architectural decisions to the actual implementation, the relation between decisions themselves, and from architectural decisions back to the requirements that have been set for the information system. Especially respondents who strongly contribute to the clusters *High-level*, *Low-level* and *Specialist* (see Table 2)

[1] 1 being not important, 5 being very important.

Table 4. Importance of use case clusters per cluster of architectural roles and cluster of architecture levels. (+) denotes importance, (–) denotes unimportance.

Use case cluster	Cluster of architectural roles	Cluster of architecture levels
Architectural decision set	(–) *Communicator* (–) *Specialist*	(–) *Software Architecture* (–) *Systems Architecture*
Assessment – reqs.→arch.→impl.	(+) *High-level* (+) *Low-level* (+) *Specialist*	(+) *All levels*
Assessment – risk, trade-off analysis	(–) *Specialist* (–) *Communicator* (–) *Low-level*	(–) *Software Architecture*
Stakeholder-centric	(+) *High-level* (–) *Communicator*	(+) *Enterprise Architecture* (+) *Process and Information Architecture*
Forward Architecting	(+) *High-level* (–) *Low-level*	(+) *All levels*

regard these use cases as important. These roles are the 'construction' roles with respect to architecture. This confirms our idea that practitioners involved in the construction of architectures have a need for traceability of architecture. The use cases in the cluster *Assessment – risk, trade-off analysis* are not regarded as important by the *High-level* cluster of architectural roles. Furthermore, especially practitioners engaged in *Software Architecture* regard the use cases in this cluster as not important.

A difference that exists between the two subclusters within *Assessment* could lie in the architect's mindset. The results of the cluster *Assessment – reqs.→arch. →impl.* reveal a mindset with a linear (i.e. non-iterative) approach to designing an architecture that satisfies the posed requirements and subsequently have the implementation satisfy the architecture. Use cases that offer traceability in this approach are regarded as important. The use cases in the cluster *Assessment – risk, trade-off analysis*, on the other hand, all are aimed at having an intermediate period of reflection to verify what risks apply, or what quality attributes could be affected by certain architectural decisions. These use cases are not directly related to either requirements or implementation.

In summary, in contrast to the literature stating that architecture offers a good means to assess the correctness and suitability of the desired solution (e.g. [1,11]), our results reveal architects regard the use cases for architectural knowledge in the *Assessment – risk, trade-off analysis* cluster as not particularly important. Literature points out that an architecture enables us to assess the design maturity, perform incremental, iterative design reviews, and periodically identify the largest risks pertaining to the architecture. Apparently, these benefits of architecture are not valued by our respondents, which is surprising.

Moreover, the use cases in the cluster *Assessment – risk, trade-off analysis* aim at finding possible problems in a certain architecture. Since practitioners

do not regard these use cases as important, we might infer that practitioners do not favour a period of reflection in which the current state of the architecture is explicitly tested. Yet, this is one of the main reasons stated in the literature for developing an architecture [1]. Apparently, these intended benefits of architecture have not yet been firmly established in the mindset of architects. The lack of value contributed to the intended benefits reveals a mindset of positiveness ("architects always take the right decisions"), which supports the findings of [7]. Respondents do not like to use architectural knowledge to identify potential weaknesses of their design.

A number of use cases for architectural knowledge are *Stakeholder-centric*. These use cases involve identifying stakeholders and communicating the architecture towards these stakeholders. Five out of the seven use cases in this cluster are regarded as important by the respondents. Especially the *High-level* role deems these use cases important. The remaining use cases 'identify affected stakeholders on change' and 'identify key architectural decisions for a specific stakeholder' are deemed neutral. Furthermore, stakeholder-centric use cases are regarded as more important at the architecture levels *Enterprise Architecture* and *Process and Information Architecture* than on the other levels. This confirms the general idea that the architecture levels *Enterprise Architecture* and *Process and Information Architecture* are suitable for communicating architecture to non-IT stakeholders. The other way around, practitioners engaged in *Software Architecture* and *Systems Architecture* do not regard communication of the architecture to stakeholders as important. Apparently, at these more technically oriented levels of architecture, practitioners mainly capture architectural decisions for themselves and not for communication to other stakeholders. This in itself is not bad, but reveals that different communication needs exist for different architecture levels.

Four use cases for architectural knowledge fall into the cluster *Forward Architecting*. When we regard the use cases in this cluster we see that 'adding an architectural decision' is deemed important at all architecture levels and by most architectural roles (only the *Specialist* role does not regard this use case as important). The use case 'remove consequences of a cancelled decision' is not deemed very important. We can identify two reasons for this. Firstly, this use case requires that a practitioner is able to cancel an architectural decision. Consequently, the practitioner should determine the decision that needs to be cancelled. This requires the practitioner to make a review iteration. Secondly, this use case does not directly contribute to the forward-engineering paradigm we identified when we analysed the *Assessment* use cases. Other use cases in this cluster, such as 'reuse architectural decisions' and 'retrieve an architectural decision' are deemed important by all architectural roles and at all architecture levels. These results show that the practitioners regard architectural decisions as an important asset to be reused in developing a specific architecture.

In addition to the results listed in Table 4, we make another observation. A difference exist with respect to the perceived importance of use cases between

the clusters *Communicator*, *Low-level*, and *Specialist* on the one side, and *High-Level* on the other side. The cluster *High-level* regards more clusters of use cases important than the other clusters. A possible reason lies in the fact that practitioners in the *High-level* cluster have a wider perspective on architecture and stakeholders involved, whereas practitioners in the other clusters have a more narrowed focus on architecture. This corresponds with the variety of roles and activities of a software architect listed in [11].

5 Threats to Validity

Our case study faced a number of threats. We list them similar to [13,20]. Our survey was targeted at practitioners in the Netherlands. By carefully selecting the participants for the survey, we have attempted to minimize a selection bias. Nevertheless, IT service providers are somewhat overrepresented in our population. Still, a comparison of the responses of practitioners employed at IT service providers and respondents employed at other organizations did not show significant differences.

We kept control on the population of practitioners we invited to participate in the survey. However, we do not have insight into the reasons why the non-respondents did not participate. We conjecture that these practitioners did not have enough time to administer the survey or could not relate the topic of the survey to their daily work. Although our survey satisfies the guidelines for the number of questions and maximum administration time as posed in [13], our results may suffer from a maturation effect, which means that the attitude of the participants towards the use cases in the survey changes during filling in the survey. On the one hand, use cases in the first half of the survey receive a more important rating than use cases in the second half. On the other hand, the second half does contain several use cases rated 'important'. Therefore, we have confidence that the maturation effect did not influence our results substantially.

It was not possible to obtain a structure in the use cases for architectural knowledge based on the practitioners' answers alone. Apparently, the survey answers varied too much to be used for structuring the use cases. A reason for this could be that our study is based on more recent definitions of architecture as made of a set of architectural decisions [2,3,21]. Some participants may regard architecture as a set of components and connectors and are not yet used to viewing architecture as a set of architectural decisions and rationale. Our approach, which uses a list of use cases for architectural knowledge, may have biased the results since the actual mindset of architects may require additional use cases or other approaches to be fully captured. We provide an architectural knowledge-oriented view towards the mindset.

To be able to reflect on the answers given, we identified a clustering based on the use cases for architectural knowledge alone and related the answers to these clusters. The resulting reflection in Sect. 6 is not only based on the clusters of use cases, but puts the survey results in a broader perspective.

6 Discussion and Conclusions

We conducted survey-based research on how the practitioners in software architecture in the Netherlands view and use architectural knowledge. Our results reveal the importance of certain use cases for architectural knowledge for the daily work of the practitioners. The individual results have been discussed in Sect. 4.5. This section reflects on these results and draws overall conclusions on the architect's mindset and the role of architectural knowledge in that mindset.

Figure 2 provides an overview of the results and depicts the major elements of the reflection. We approach architecture from two different perspectives. One perspective is focused on developing a solution, i.e. the architecture. The other perspective is focused on the underlying reason for that solution, i.e. architectural decisions and rationale. The clusters of use cases for architectural knowledge are depicted as package symbols. The +-mark or – -mark indicate the respondents' view on these clusters. We put the clusters in perspective by depicting the evolution between the different results that we identify in practice. By and large, widespread acceptance of architecture verification activities preceded architecture validation activities, such as performing risk or trade-off analyses. Similarly, viewing architecting as a forward decision-making process preceded managing the set of architectural decisions, i.e. 'architectural knowledge management'. Putting stakeholders central in architecture has been an important characteristic across time and perspectives. The remainder of this section describes our views as expressed in Fig. 2.

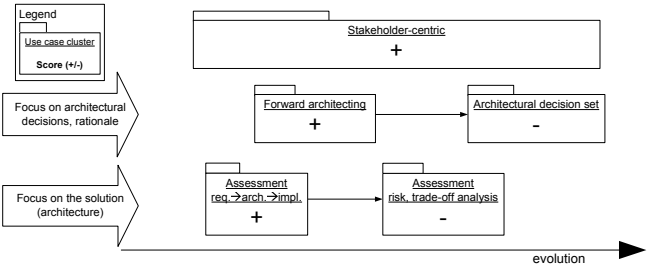

Fig. 2. Overview of the architect's mindset

Forward architecting – Architects regard taking architectural decisions and making these decisions explicit as important. Yet, architects tend to focus on only taking architectural decisions to end up with a correct software architecture for a specific problem. In taking these decisions, architects are supported by e.g. architectural patterns [22], which provide proven architectural solution fragments for certain problems, and by rationale tools such as gIBIS [23] and QOC [24]. We signal an ongoing tension between making architectural decisions

and capturing the underlying rationale and other context of these decisions; the time spent on capturing the context is not spent on making new architectural decisions. Consequently, adequate, lightweight tooling is necessary to lower the threshold for capturing the context. Despite the continual tension, progress has been made [25,26].

Architectural decision set – On a more generic level, architects do not regard the architecture as a set of architectural decisions. Although the concept of architectural decisions in itself has gained importance, the architect's mindset lacks focus on reflections on those decisions as building blocks for software architectures. These reflections allow for a step back to actually learn from architecture experiences. Furthermore, architects do not (yet) manage or manipulate that set of architectural decisions (i.e. use cases in the cluster *Architectural decision set*). A reason for this could be that more recent definitions of software architecture in terms of architectural decisions [2,3,21] are not yet completely transferred to practice. In addition, adequate tool support is necessary to fully exploit architectural knowledge as a set of architectural design decisions across architectures and domains.

Assessment – reqs.→arch.→impl. – Software development largely occurs via projects. Depending on the development approach chosen, the architecting phase can run in parallel during the lifetime of the project or the architecting phase is a distinct phase which leads to a deliverable – the architecture. Based on the results of this study, we conjecture that the latter is the case: the practitioners show an approach in which the architecture is delivered based on the requirements. After that, the implementation is checked against the architecture. Our experience shows that this verification phase often is not performed by architects. Architects, often experienced and relatively expensive resources, perhaps run off to other projects to run the architecting phase at that project. Consequently, they may not be offered the time to support the design and implementation phase.

Assessment – risk, trade-off analysis – Our study shows that methods and techniques to validate the architecture (such as the Architecture Tradeoff Analysis Method as described by [27], or their predecessors) are not embedded within the mindset of architects. A recent presentation on the topic of this paper given during the Dutch architecture conference revealed that when practitioners do deem performing a risk analysis important, they do not have clear what the role of architectural knowledge is in a risk analysis. Architectural knowledge may support to evaluate the impact of architectural decisions on the resulting architecture; it allows to (re-)consider alternative decisions as well. Apparently, this rather new view on architecture is not yet generally accepted. Education on viewing architecture as architectural decisions [10] as part of architectural knowledge could help overcome this.

Stakeholder-centric – Another benefit of architecture is that it enables communication among stakeholders [1]. Architecture thus can be regarded as a language to transfer the architect's opinions and views to those stakeholders. Most use cases in the cluster *Stakeholder-centric* rate high, which means that the

view of 'architecture as language' [10] is generally accepted. Communication of architecture to stakeholders is clearly established in the mindset of architects.

Our study shows that the mindset of architects is focused on delivering a solution and capturing the related architectural decisions. Consequently, we conjecture that a so-called *micro* view on software architecture largely is in place: architects are focused developing an architecture for a specific solution and (more and more) on capturing the architectural decisions and rationale for that solution. What lacks in the mindset of architects is a view that exceeds specific architectures but puts architectures in context by validating them, and the architectural decisions that led to them. When architects have a set of architectural decisions at their disposal, this offers the opportunity to interrelate architectural decisions taken in the past to identify learning opportunities for future architecting activities. We conjecture that this *macro* view may be achieved by applying initiatives that proved valuable in other disciplines, such as ontology engineering [28] onto the domain of (software) architecture.

In summary, the mindset of architects in the Netherlands reveals an approach which is focused on 'to create and communicate' rather than 'to review and maintain'. This reflects a general pattern as e.g. highlighted in [7]. Furthermore, architectural knowledge and the view of architecture as a set of architectural decisions has not yet transferred to industry. We see two possible approaches to embed the importance of architectural knowledge and design decisions in industry. Firstly, more knowledge transfer is needed on the concepts and intended benefits of this view. Secondly, it is necessary to collect more empirical data on these benefits in terms of throughput and cost to fully sustain the importance of architectural knowledge and architectural decisions.

7 Future Work

Our work describes the mindset of architects in the Netherlands. We provided several reasons for this mindset but acknowledge that additional research is needed on the foundation for this mindset. This additional research could focus on the activities needed to effectively establish the concept of architectural knowledge in the architect's mindset. The possible increase in understanding of architectural knowledge by architects may be monitored by using our survey instrument periodically. Moreover, we can compare the mindset of architects in the Netherlands with the mindset of architects at other countries or continents by reusing this survey.

We envision the use cases for architectural knowledge to define operations on a grid for architectural knowledge. We view this grid to support satisfying the need for architectural knowledge from different perspectives. De Boer et al. [29] define a model that lies at the basis for this knowledge grid and supports capturing architectural knowledge.

Within our research project, we are developing, notations, tools and associated methods to extract, represent and use architectural knowledge. This paper sheds

light onto the most important use cases for architectural knowledge from a practitioners' perspective. Although specialized tool support for the architects is still generally lacking, we use these results to develop tools for the most important use cases for architectural knowledge. In addition, we continue the work in our project to further embed the view of architectural knowledge and architectural decisions in practice.

Acknowledgement. We thank Joost Schalken for his support in the statistical analysis of the survey results and Rik Farenhorst and Remco C. de Boer for our discussions to clarify the survey results. This research has been partially sponsored by the Dutch Joint Academic and Commercial Quality Research & Development (Jacquard) program on Software Engineering Research via contract 638.001.406 GRIFFIN: a GRId For inFormatIoN about architectural knowledge.

References

1. Bass, L., Clements, P., Kazman, R.: Software Architecture in Practice, 2nd edn. SEI Series in Software Engineering. Addison-Wesley Pearson Education, Boston (2003)
2. Jansen, A., Bosch, J.: Software Architecture as a Set of Architectural Design Decisions. In: WICSA 2005. 5th Working IEEE/IFIP Conference on Software Architecture, Pittsburgh, Pennsylvania, pp. 109–120 (2005)
3. Kruchten, P., Lago, P., Van Vliet, H.: Building up and Reasoning about Architectural Knowledge. In: Hofmeister, C., Crnkovic, I., Reussner, R. (eds.) QoSA 2006. LNCS, vol. 4214, pp. 43–58. Springer, Heidelberg (2006)
4. Clements, P., Bachmann, F., Bass, L., Garlan, D., Ivers, J., Little, R., Nord, R., Stafford, J.: Documenting Software Architectures: Views and Beyond. SEI Series in Software Engineering. Addison-Wesley Professional, Reading (2003)
5. IEEE: IEEE Recommended Practice for Architectural Description of Software-Intensive Systems. Standard 1471-2000, IEEE (2000)
6. Van der Ven, J.S., Jansen, A., Avgeriou, P., Hammer, D.K.: Using Architectural Decisions. In: Hofmeister, C., Crnkovic, I., Reussner, R. (eds.) QoSA 2006. LNCS, vol. 4214, Springer, Heidelberg (2006)
7. Tang, A., Babar, M.A., Gorton, I., Han, J.: A Survey of the Use and Documentation of Architecture Design Rationale. In: WICSA 2005. 5th Working IEEE/IFIP Conference on Software Architecture, Pittsburgh, Pennsylvania, pp. 89–98 (2005)
8. Tyree, J., Akerman, A.: Architecture Decisions: Demystifying Architecture. IEEE Software 22(2), 19–27 (2005)
9. Zimmermann, O., Gschwind, T., Küster, J., Leymann, F., Schuster, N.: Reusable Architectural Decision Models for Enterprise Application Development. In: QoSA 2007. Third International Conference on the Quality of Software Architectures. LNCS, Springer, Heidelberg (2007)
10. Smolander, K.: Four Metaphors of Architecture in Software Organizations: Finding Out the Meaning of Architecture in Practice. In: ISESE 2002. 2002 International Symposium on Empirical Software Engineering, pp. 211–221 (2002)
11. Hofmeister, C., Nord, R., Soni, D.: Applied Software Architecture. Addison-Wesley Longman Publishing Co. Inc., Boston, MA, USA (2000)

12. Clements, P., Kazman, R., Klein, M., Devesh, D., Reddy, S., Verma, P.: The Duties, Skills, and Knowledge of Software Architects. In: WICSA 2007. 6th Working IEEE/IFIP Conference on Software Architecture, Mumbai, India, IEEE Computer Society Press, Los Alamitos (2007)
13. Kitchenham, B.A., Pfleeger, S.L.: Principles of Survey Research, Parts 1 to 6. SIGSOFT Software Engineering Notes (2001–2002)
14. Likert, R.: A Technique for the Measurement of Attitudes. Archives of Psychology 140 (1932)
15. SEI: Published Software Architecture Definitions (November 10th, 2006) `http://www.sei.cmu.edu/architecture/published_definitions.html`
16. Florijn, G., Clerc, V., Van Ekris, J., Koning, H., Leih, G., Maat, M., Niessink, F.: Softwarearchitectuur - Overzicht en Compendium (Dutch). Ten Hagen & Stam (2003)
17. Borg, I., Groenen, P.J.F.: Modern Multidimensional Scaling, 2nd edn. Statistics. Springer, New York (2005)
18. MacQueen, J.: Some Methods for Classification and Analysis of Multivariate Observations. In: Cam, L.M.L., Neyman, J. (eds.) The Fifth Berkeley Symposium on Mathematical Statistics and Probability. Statistics, vol. 1, pp. 281–297. University of California Press, Berkely, California (1967)
19. Anton, H.: Elementary Linear Algebra, 9th edn. John Wiley & Sons, Inc., Chichester (2005)
20. Kitchenham, B.A., Pfleeger, S.L., Pickard, L.M., Jones, P.W., Hoaglin, D.C., Emam, K.E., Rosenberg, J.: Preliminary Guidelines for Empirical Research in Software Engineering. IEEE Transactions on Software Engineering 28(8), 721–734 (2002)
21. Rozanski, N., Woods, E.: Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives. Addison Wesley Professional, Reading (2005)
22. Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M.: Pattern-Oriented Software Architecture, Volume 1: A System of Patterns. John Wiley & Sons, Inc., Chichester (1996)
23. Conklin, J., Begeman, M.L.: gIBIS: A Hypertext Tool for Exploratory Policy Discussion. ACM Transactions on Office Information Systems 6(4), 303–331 (1988)
24. MacLean, A., Young, R.M., Bellotti, V.M.E., Moran, T.P.: Questions, Options, and Criteria: Elements of Design Space Analysis. In: Moran, T.P., Carroll, J.M. (eds.) Design Rationale. Concepts, Techniques, and Use, pp. 53–105. Lawrence Erlbaum and Associates, Mahwah, NJ (1996)
25. Fischer, G., Lemke, A.C., McCall, R., Morch, A.I.: Making Argumentation Serve Design. In: Moran, T.P., Carroll, J.M. (eds.) Design Rationale. Concepts, Techniques, and Use, pp. 267–293. Lawrence Erlbaum and Associates, Mahwah, NJ (1996)
26. Conklin, J., Selvin, A., Buckingham-Shum, S.J., Sierhuis, M.: Facilitated Hypertext for Collective Sensemaking: 15 Years on from gIBIS. In: 12th ACM Conference on Hypertext and Hypermedia, Århus, Denmark, pp. 123–124. ACM Press, New York (2001)
27. Clements, P., Kazman, R., Klein, M.: Evaluating Software Architectures: Methods and Case Studies. SEI Series in Software Engineering. Addison-Wesley Professional, Boston (2001)

28. Gruber, T.R.: Towards Principles for the Design of Ontologies Used for Knowledge Sharing. In: Guarino, N., Poli, R. (eds.) Formal Ontology in Conceptual Analysis and Knowledge Representation, Kluwer Academic Publishers, Deventer, The Netherlands (1993)
29. De Boer, R.C., Farenhorst, R., Clerc, V., Van der Ven, J.S., Deckers, R., Lago, P., Van Vliet, H.: Structuring Software Architecture Project Memories. In: LSO 2006. The 8th International Workshop on Learning Software Organizations, Rio de Janeiro, Brazil, pp. 39–47 (2006)