Prerequisites for Successful Architectural Knowledge Sharing

Rik Farenhorst, Patricia Lago, Hans van Vliet
Department of Computer Science
Vrije Universiteit Amsterdam
The Netherlands
{rik, patricia, hans}@cs.vu.nl

Abstract

Sharing knowledge pertaining to software architectures becomes increasingly important. If this knowledge is not explicitly stored or communicated, valuable knowledge dissipates. However, stakeholders will only share knowledge with each other if they are motivated to do so, or in other words if the necessary incentives are created. In this paper we identify three incentives for architectural knowledge sharing: the establishment of social ties, more efficient decision making, and knowledge internalization. Next, we discuss our experiences on how architectural knowledge is shared in a large software development organization. Based on these experiences we propose a set of prerequisites that need to be met to foster successful architectural knowledge sharing. The importance of these prerequisites is motivated by demonstrating that they create the identified incentives.

1. Introduction

The last decade and a half has seen a phenomenal growth of software architecture as a discipline [22]. There now is a universal recognition that software architecture is an indispensable part of software system development. Using an explicit software architecture allows for early stakeholder interaction, provides a basis for a work breakdown structure, and makes it possible to assess the quality of the system in an early stage [5]. Although considerable progress has been made in this area, few techniques exist to manage and share knowledge pertaining to software architectures.

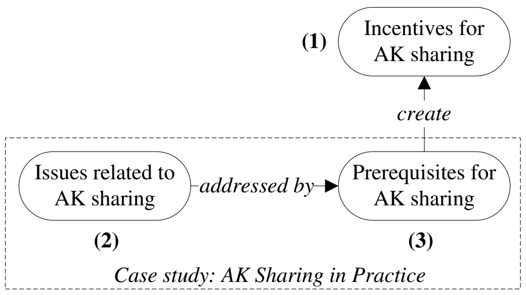
Various authors (e.g. [2, 6, 18]) address the notion of 'architectural knowledge' and provide a model of what this notion entails. Key elements in all these models are architectural design decisions and their rationale. A focus on such decisions allows capturing the process of *how* the software architecture was created and *why* it was done this way, instead of only emphasizing *what* it is comprised of.

Architectural design decisions embodied in the software architecture and the rationale for these decisions is often not explicitly stored or communicated. Therefore, valuable knowledge dissipates. This common problem in the architecting phase of software development is also known as knowledge vaporization [6].

Emerging trends such as globalization, offshoring, virtual organizations, and collaboration within closed and open communities further enhance the importance of sharing architectural knowledge. Stakeholders are often located at large distances from each other, which makes face-to-face exchange of knowledge hard. Nevertheless, to exchange ideas and share expertise these stakeholders have specific knowledge needs.

The knowledge needs of stakeholders in the architecting process can be addressed in various ways. However, implementing means for architectural knowledge sharing is definitely not only about choosing the right tool. Successful knowledge sharing can only be achieved if the necessary incentives are created. These incentives induce stakeholders to share valuable architectural knowledge, such as the major design decisions made, the underlying rationale for these decisions, and alternatives that were considered.

In Figure 1 our research contribution is visualized schematically. In this figure, the numbers between the brackets denote the main results presented in this paper. Based on a literature review on knowledge management principles, combined with our own experience in software architecture research, we **(1)** identify three incentives for architectural knowledge: the establishment of social ties, more efficient decision making, and knowledge internalization. In addition, to get insights in how architectural knowledge is shared in practice, we have conducted a case study in a large software development organization. Based on the results of this case study, we **(2)** identify a set of issues pertaining to sharing architectural knowledge. In order to address these issues, we **(3)** propose a set of prerequisites for architectural knowledge sharing. We argue that successful architectural knowledge sharing is only possible if



(1)
Incentives for AK sharing
create
Issues related to AK sharing
addressed by
Prerequisites for AK sharing
(2)
(3)
Case study: AK Sharing in Practice

Figure 1. Research Contribution

these prerequisites are met. We demonstrate this by showing that these prerequisites not only address the identified issues, but also create the identified incentives for sharing architectural knowledge. This demonstration indicates that these prerequisites are crucial to foster successful architectural knowledge sharing; not only in the context of the case study organization, but in general.

During the case study, our primary interest was in the architecting process and the mechanisms implemented to share architectural knowledge. We have modeled the architecting process of this organization using four different perspectives that focus on the design decisions taken, the architectural descriptions made, the roles and responsibilities of the stakeholders involved, and the architecting environment in which the process takes place, respectively. The advantages of modeling the organization's architecting process this way are twofold. First, it acts as a diagnosis instrument that allows us to identify the issues related to architectural knowledge sharing. Second, it serves as an implementation instrument that helps improving the organization's architecting process by addressing these issues.

The remainder of this paper is organized as follows. Section 2 discusses related work on knowledge management in the software architecture domain. In Section 3 a set of incentives for architectural knowledge sharing is identified. Section 4 describes the research methodology used during the case study. Sections 5 and 6 cover the results of this case study, and list a number of issues pertaining to architectural knowledge sharing. In Section 7 a set of prerequisites for architectural knowledge sharing is proposed, and it is demonstrated how these prerequisites create the necessary incentives for architectural knowledge sharing. Finally, Section 8 contains our conclusions and presents an outlook on future work.

2. Related work

In recent years, increasing attention has been paid in software architecture research to the notion of 'architectural knowledge' and how this knowledge, such as architectural design decisions and their rationale, can be successfully managed [6,24,26]. A growing number of researchers acknowledges that a software architecture can - or should - be viewed as the collection of architectural design decisions [17], or as the design decisions plus the resulting design [18]. Consequently, the architecting process is viewed as being primarily a decision making process.

In an overview of the maturation of the software architecture field [22], Shaw and Clements conclude with an outlook on future work in software architecture research. Promising topics mentioned include the organization of architectural knowledge to create reference materials, and the link of architectural design decisions to quality attributes. Both these goals are served in our recent work on constructing a domain model of architectural knowledge [12], that acts as a common frame of reference for architectural knowledge concepts.

Other recent software architecture research focuses on the traceability between architectural design decisions and stakeholder concerns [27], and the documentation of architecture design rationale [23]. In addition, some initiatives for managing or sharing architectural knowledge have been introduced. These include an architectural knowledge repository that provides an environment to capture and manage architectural knowledge and support the architecting process [2], and a tool that models the relationship between a software architecture and the design decisions embodied in it in detail [17].

Research pertaining to architectural knowledge benefits from related research fields, especially from knowledge management research. The influence from the knowledge management field in Software Engineering research is not new, as can be seen from examples such as the Experience Base construct [3] and organizational learning principles [9]. To further enhance the state of the art in software architecture, in this paper we draw on knowledge management principles in order to improve architectural knowledge sharing.

In recent literature additional warnings are presented for the fact that not all knowledge sharing implementations are successful automatically. In [13] several factors that make knowledge sharing difficult are listed, such as the fact that knowledge sharing is time consuming, and that people might not trust the knowledge management system. Another warning is that it is infeasible to strive for completeness. In [20] it is noted that it is impossible to create a tool to reason about all the issues that an architect would normally need to consider. Finally, we should be aware of the fact that a lot of the available knowledge cannot be made explicit at all, but instead remains tacit in the minds of people [21]. Sharing this tacit knowledge is very hard, as stated in [14].

The above warnings clearly indicate that successful architectural knowledge sharing involves more than just choosing the right software tool. We argue that successful architectural knowledge sharing can only be achieved if the necessary incentives are created. How these incentives can be created is the topic that is addressed in the remainder of this paper.

3. Incentives for AK Sharing

In the architecting phase of software development, a lot of implicit and explicit knowledge is required to take appropriate architectural design decisions. This knowledge is usually scattered across the organization. Various stakeholders, such as software developers, architects, project managers, and maintainers, all possess knowledge that can be of value when developing the software architecture. We assert that sharing this knowledge between these stakeholders leads to a number of 'intrinsic' benefits. If stakeholders would recognize these benefits, it would induce them to share valuable architectural knowledge. In this sense, these benefits act as incentives for architectural knowledge sharing.

Based on insights gained from related knowledge management literature, as well as our own experience in software architecture research, we have identified three incentives for architectural knowledge sharing. We argue that these incentives need to be created in order to promote successful architectural knowledge sharing. Below, they are elaborated in turn.

1. **Establishment of social ties.** Research has shown that if stakeholders trust each other and the 'competition' between them is minimized, the motivation for cooperation and sharing knowledge is higher [1]. Moreover, individuals will only participate willingly in knowledge exchange once they share a sense of identity or belonging with their colleagues [7]. Sharing knowledge with these colleagues may then further increase the trust between them, thereby improving relationships. Establishing social ties, either by increasing the trust or by minimizing the competition between stakeholders, therefore acts as an incentive for knowledge sharing.

   This incentive for knowledge sharing also applies to the software architecture domain. When typical stakeholders in the architecting process, such as project managers, developers, or architects, collaborate in the decision making process and exchange ideas or expertise, the social ties between them will become stronger.

2. **More efficient decision making.** According to [8] a group's performance increases when everyone in this group is informed of each other's expertise. This is because such knowledge allows groups to engage in joint brainstorming sessions in which group members are able to explore new ideas and discuss difficult issues.

   This increase in group performance is something architects in a software architecture environment benefit from. For software architects it is often very important to not only get feedback from colleague architects, but also from developers or maintainers when developing the software architecture. Developers may have a lot of expertise and their insights and experience can benefit the architects in an early stage. The same holds true for maintainers that may have gathered knowledge on the actual pros or cons of past decisions. Architectural knowledge sharing creates a feedback loop between architects and other stakeholders in the architecting process. Having such a feedback loop greatly improves the efficiency of the decision-making process and is therefore an incentive for architectural knowledge sharing.

3. **Knowledge internalization.** When actively sharing knowledge, people are able to communicate valuable experience with each other. This experience is then stored as tacit expert knowledge in their minds. This process is called knowledge internalization [8]. Knowledge internalization refers to the degree to which a recipient obtains ownership of, commitment to, and satisfaction with the transferred knowledge. When knowledge is fully internalized by a recipient, it becomes theirs. Consequently, tacit knowledge increases the quality of the work and makes the work go smoothly [14], and the more knowledge is shared, the more knowledge can be internalized by practitioners.

   Knowledge internalization greatly assists software architects in their daily practice. Having internalized valuable experiences helps them in avoiding suboptimal solutions, but also enables them to learn from mistakes, such as 'bad' design decisions or architectures that have been developed in the past but now cause headaches. As a result, the software architectures that are being developed are based on a set of decisions that is more carefully considered, and therefore have a higher overall quality.

4. Research Methodology

Within our research group we are developing notations, tools and associated methods to extract, represent and use architectural knowledge. Our research is conducted in a research consortium, in which four large organizations participate. Each of these organizations has specific interests in, or issues with successful managing and sharing knowledge pertaining to software architectures. In yearly research cycles we focus on a selection of these issues, conduct case studies to come to the core of the problem, and subsequently propose and develop solutions to this problem. After each research cycle we reflect on the insights gained and results obtained, in order to further improve our understanding of architectural knowledge.

In our latest research cycle, we aimed at investigating how architectural knowledge is shared in practice. To this end, we have conducted a case study in a large Dutch software development organization. Although this organization acknowledges the importance of software architectures, discussions with architects and developers indicated that the organization struggles with how software architecture development - using methods, techniques, and tools - can best be fit in the overall development process, and how architectural knowledge can best be shared and (re)used.

Our research methodology best resembles action research [4]. The essence of action research can be described as a two stage process, consisting of a diagnostic stage that involves a collaborative analysis of the current situation, followed by a therapeutic stage that covers collaborative change experiments to improve this situation. During the case study we were actively involved with stakeholders from the case study organization. Results of the diagnostic stage, in which we investigated how architectural knowledge is currently shared within the case study organization, are elaborated in Sections 5 and 6. Our suggestions for improvement are discussed in Section 7, but implementation thereof remains to be done.

5. Modeling the Architecting Process

We started our research in the case study organization with a diagnosis of how architectural knowledge was perceived in this organization. We used three main sources for this diagnosis: a questionnaire containing several use cases for architectural knowledge, a documentation study of standards, best practices and architectural descriptions, and finally a set of open interviews with various stakeholders of the architecting process.

Questionnaire. We used a questionnaire consisting of 27 potential use cases for architectural knowledge. A more elaborate description of these use cases is given in [25]. We asked 15 architects how important they consider these use cases, by letting them give scores on a scale from 1 to 5. We ranked the use cases by counting the total scores they received from the 15 architects. Among the five use cases that rank highest are three use cases that are clearly devoted to sharing architectural knowledge:

- *Take a design decision based on explicit concerns of a stakeholder.*
- *Explain to stakeholders (the impact of) a design decision that is taken.*
- *Keep stakeholders up-to-date on a certain (new or changed) design decision.*

This indicates that there is a clear interest in sharing architectural knowledge within this organization.

Documentation study. In order to retrieve more information on how architectural knowledge is currently shared, we examined available documentation such as architectural standards and best practices, example architecture documents, and related functional and technical system documentation. During this documentation study we gained initial insight into the terminology used and the kind of systems that are developed.

Interviews. We held 17 interviews with various kinds of stakeholders, among which were architects, developers, maintainers, project managers, and software testers. The interviews were open and focused on how the interviewees perceive architectural knowledge in the organization. Since the interviewees were asked to elaborate their answers the interviews provided us with detailed information on how architects and other stakeholders use architectural knowledge, what their knowledge need is, and what the current limitations are regarding architectural knowledge sharing.

Using the questionnaire, documentation study, and interviews as primary sources, we were able to collect an extensive amount of information about the architecting process of the case study organization. We have modeled the collected information using four perspectives. These four perspectives have been identified based on our domain model of architectural knowledge [12]:

1. The **architecting environment** perspective that targets the broad context in which the software architectures are developed.
2. The **decision making** perspective that puts the architectural design decisions and their rationale central.
3. The **architectural descriptions** perspective that focuses on the design artifacts that are constructed.
4. The **stakeholder roles and responsibilities** perspective that examines the stakeholders and their roles.

Viewing the case study organization from these multiple angles allowed us to retrieve information specific to important aspects of the architecting process, and to capture the recursive nature of this process, which is described in [15]. Other approaches tend to focus on either the decision making process, without explicitly looking at the process in which the decisions are being taken (e.g. [19] and its descendants), or on the design artifacts (e.g. [16]).

In this paper we use the observations made in [11] with respect to architectural knowledge sharing in the case study organization to determine a set of prerequisites that address the knowledge sharing issues. In the following subsections we summarize the findings by focusing on each of the four different perspectives in turn.

5.1. Architecting environment

Architect: "Even though it was clear that the system's performance was worse when using this type of database connections, the customer strictly referred to its reference architecture that forbids any alternatives. Further discussion on this issue was not possible."

The organization in which this case study has been performed has a large organization as its main customer. This customer organization has its own architecture department that has strict control over the system development. Based on this customer organization's reference architecture, constraints have been defined for individual systems. Often there is little room for deviating from these constraints.

Because of the existing systems that are already in place in the customer organization, little or no systems are developed based on a 'greenfield' situation. Often, new systems have to replace or extend an existing system. The case study organization has developed a knowledge repository to guide architects in creating an architecture, but this repository is mainly intended for 'greenfield' system development. The guidelines used by the repository do not comply with the reference architecture of the main customer. The customer organization acknowledges that if crucial architectural knowledge about the reference architecture is added to the repository, the quality and efficiency of decision making will increase. Nevertheless, as of yet no effort has been taken to implement this. Therefore, when using the repository, chances are that architectural solutions proposed are in conflict with the constraints defined by the customer.

One can argue that among the most interesting architectural knowledge worth sharing is that of the customer organization. The fact that this is not done in the current repository is one of the main reasons for architects not to use it. This was confirmed in interviews held with four different architects.

5.2. Decision making

Developer: "It is often the case that me and my colleagues take the major design decisions, while the architect's main function is that of a scribe. The architecture documentation he creates reflects the decisions we have taken."

We found out that currently in this organization the architects do not have overall control over the decision making process. Developers often possess more technical expertise and are more up-to-date with the functional and technical constraints posed by the customer. Therefore, they make most of the technical design decisions. In addition, since the customer organization dictates certain rules that have to be adhered to, already several decisions have been made at the start of the architecting process. Architects and developers maintained in the interviews that they have only limited freedom in taking design decisions. They decide mainly on more localized and less cross-cutting issues, such as the kind of development tools to be used.

Above observations are in contrast with the statement that the architect is taking the main design decisions [10]. Architects in this organization do not create much architectural knowledge to share with other stakeholders, but they do act as linking pin between the customer organization and the development teams. Architects are responsible for the overall quality of the system to be built, and therefore need to solve conflicting decisions, make the right tradeoffs, and check whether development teams comply to the architecture. Therefore, for them it is important that knowledge created within the customer organization or within the development teams is shared. Both project managers and architects confirmed in the interviews that this feedback loop would allow architects to build up expertise, gain practical insights, and define best practices for future software architectures.

5.3. Architectural descriptions

Maintainer: "Up until this interview I didn't know there existed an architecture description of the system."

Not all stakeholders agree with the level of detail that is to be used in the views of the architectural description. Developers and maintainers indicated in the interviews that they consider most of the views too high-level. Project managers on the other hand think the architectural descriptions are clear and nicely suited for non-technical people as well.

Architectural knowledge is not only found in the standard architecture documentation, but also scattered around in technical and functional design documents used by developers and maintainers. Many interviewees stressed the need for collaboration and sharing important information between architects, developers and maintainers.

Although some views in the architecture description are targeted at the maintainers, maintainers are unaware of the information that is described for them. In a meeting with five maintainers it became clear that none of them had seen the architecture description of the system they were maintaining before.

5.4. Stakeholder roles and responsibilities

Architect: ”Basically, the architecture description is finished as soon as the project manager and lead developers give their approval. After this, my job ends and I am no longer responsible for keeping the architectural description up-to-date.”

In the case study organization architects are not responsible for maintaining the architecture document after the system has been developed. Since no other ‘owners’ of this document exist, it soon becomes outdated. This is one of the reasons for the fact that architectural descriptions are not often read by developers and maintainers.

The architect does not fulfill a prescriptive role in the architecting process that allows him to take all major architectural design decisions, but rather a descriptive one in which he is the mediator between the customer organization and the technical stakeholders. However, the knowledge repository introduced in Section 5.1 is designed to be prescriptive, and supports architects in making decisions. Because of this mismatch, the repository turned out to be unpopular among architects and seldom used.

6. Issues related to AK Sharing

Based on the results described in the previous section, we can conclude that the available mechanisms for sharing architectural knowledge are not optimally implemented in the case study organization. From the case study results we have distilled six issues related to sharing architectural knowledge. Figure 2 depicts a model of the current architecting process of the case study organization. This model helps visualizing the issues by highlighting the five different stakeholders of the architecting process and how they communicate. The rectangular elements represent entities; the elliptical ones denote actions on or with these entities. Arrows indicate the creation of new instances of an entity. Actions and entities that are linked together can be read as sentences, e.g ”An architect takes an architectural design decision.”

1. No consistency between architecture and design documents. There is no alignment between the architecture descriptions and the functional design and technical design documents used by developers and maintainers. Because of the lack of alignment, valuable architectural knowledge might be dispersed in the organization without the architects knowing it. Consequently, it is hard to judge whether the architectural description conflicts with the design that is preferred by the developers. This lack of alignment is visualized in Figure 2 by the absence of a direct connection between the functional and technical design documents shown at the left part and the architectural description shown at the

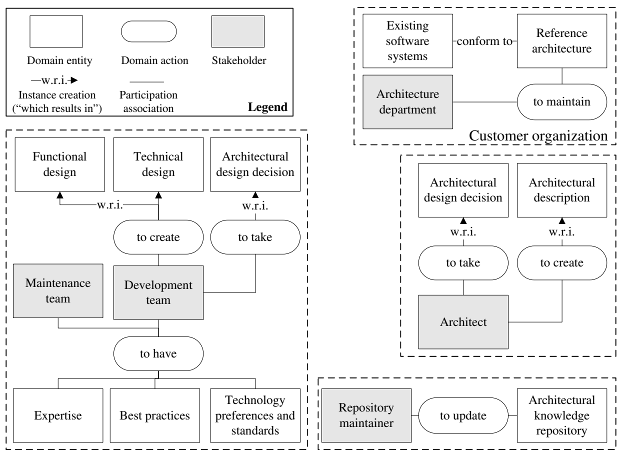


Figure 2. Current Architecting Process

right part. Architectural knowledge in a technical design document that needs to be shared therefore has to travel through two different stakeholders to reach the architecture description.

2. Communication overhead between stakeholders. Developers occasionally have to explain the architects technical decisions more than once. The reason for this is that decisions made in earlier meetings, including the rationale for these decisions, are not adequately stored in the architecture description. This knowledge sometimes dissipates quickly. Consequently, architects need to meet again with the developers to get this knowledge explicit at a later point in time. This problem of communication overhead is depicted in the right part of Figure 2 by the lack of explicit connection between design decisions taken by developers to the architectural description created by the architect.

3. No explicit collaboration with maintenance teams. Although maintainers are targeted in the architectural documentation, they are not involved as a stakeholder in the architecting process. No active discussions between architects and maintainers take place and the requirements of the maintenance teams are not taken into account during architecture development. The lack of a connection between the maintenance team and the architect in Figure 2 illustrates this problem.

4. No feedback from developers to architects. Developers sometimes wear the hat of the architect and also make design decisions. However, architects are not informed on the decisions made by the developers unless explicit meetings take place. There is no mechanism in place that allows developers to share what they are doing. Therefore, it is very difficult for the architect to find out what kind of technical issues are encountered or what detailed decisions are taken. The lack of feedback from developers to architects is reflected in the center of Figure 2 by the lack of communication between the development team and the architect.

5. No up-to-date knowledge from development teams in repository. The architectural knowledge repository contains little to no information on the 'best practices', technology preferences and standards, and expertise currently available at the development teams. Therefore, the repository is unable to advise architectural directions that match with the development processes. This problem is visualized in Figure 2 by the fact that design decisions made by the developers are disconnected from the knowledge repository.

6. No up-to-date knowledge from main customer in repository. The architectural knowledge repository also lacks up-to-date knowledge on the customer organization's reference architecture. Therefore, the repository cannot give architectural directions that automatically comply with the constraints posed by this reference architecture. This is illustrated in Figure 2 by the isolated reference architecture in the top right corner.

7. Prerequisites for Successful AK Sharing

The six issues identified in the previous section suggest that in the case study organization architectural knowledge sharing is immature. We have investigated how the architecting process of this organization needs to be changed in order to improve architectural knowledge sharing. We propose four prerequisites that have to be met in order to change the architecting process in such a way that all six architectural knowledge sharing issues are addressed. We argue that meeting these four prerequisites leads to an improved architecting process in which architectural knowledge is successfully shared. A conceptual outline of this improved process is depicted in Figure 3.

The four prerequisites are listed below. Two of them are illustrated in more detail in Figure 4 by showing the differences between the current and the improved architecting process. The thick boxes and lines in this figure highlight the domain entities and actions that relate to the issues in the current process and the solutions to these issues in the improved process.

1. **Alignment between design artifacts.** Architectural descriptions need to be aligned with other design documents. This can be done by enriching the architectural description with links to relevant (lower level) design documents, allowing developers or more technical stakeholders to more easily find their way in the set of documentation. Keeping these links up-to-date ensures that the architecture description always reflects the current state of the system, rendering it a good starting point for stakeholders that want to know something about the system. This prerequisite deals with issue #1: *'No consistency between architecture and design documents'*. In the left part of Figure 4 is visualized how this is done. At the top left of this figure the current situation is depicted. There is a clear distance between the architectural description used by the architect on the one hand, and the functional and technical design documents used by the development team and maintenance team on the other hand. The absence of a direct connection between these documents is highlighted by the thick dashed line. This line indicates that there is no check for consistency possible between these documents. In the improved situation, depicted in the lower left part of the figure, the alignment between design artifacts enables this consistency check.

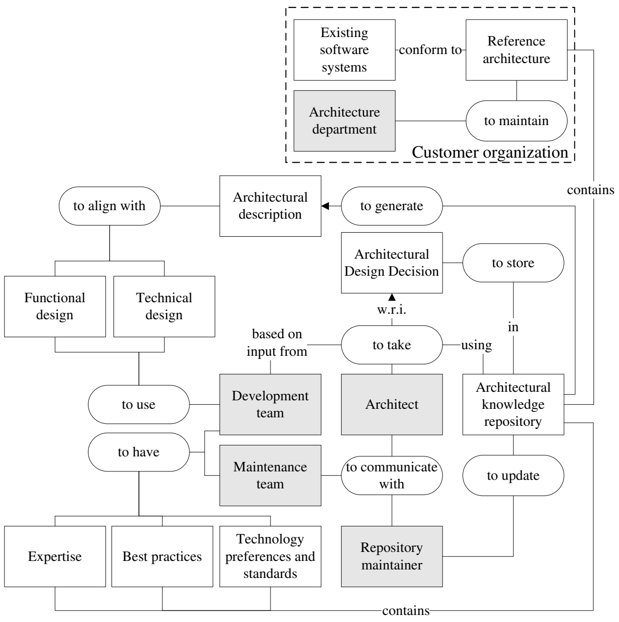


Figure 3. Improved Architecting Process

2. **Traceability between architectural decisions and descriptions.** If all architectural design decisions are documented using specific templates (e.g. using the one proposed in [24]), including considered alternatives and the rationale for the decisions, architectural descriptions provide a good summary of the decision-making process that leads to a certain architecture. Documenting the valuable architectural knowledge prevents its dissipation. As a result, communication between architects and other stakeholders, such as developers, will improve since the current state of the architecting process is known at all times. Consequently, discussions do not need to be held more often than necessary. This prerequisite therefore addresses issue #2: *'Communication overhead between*

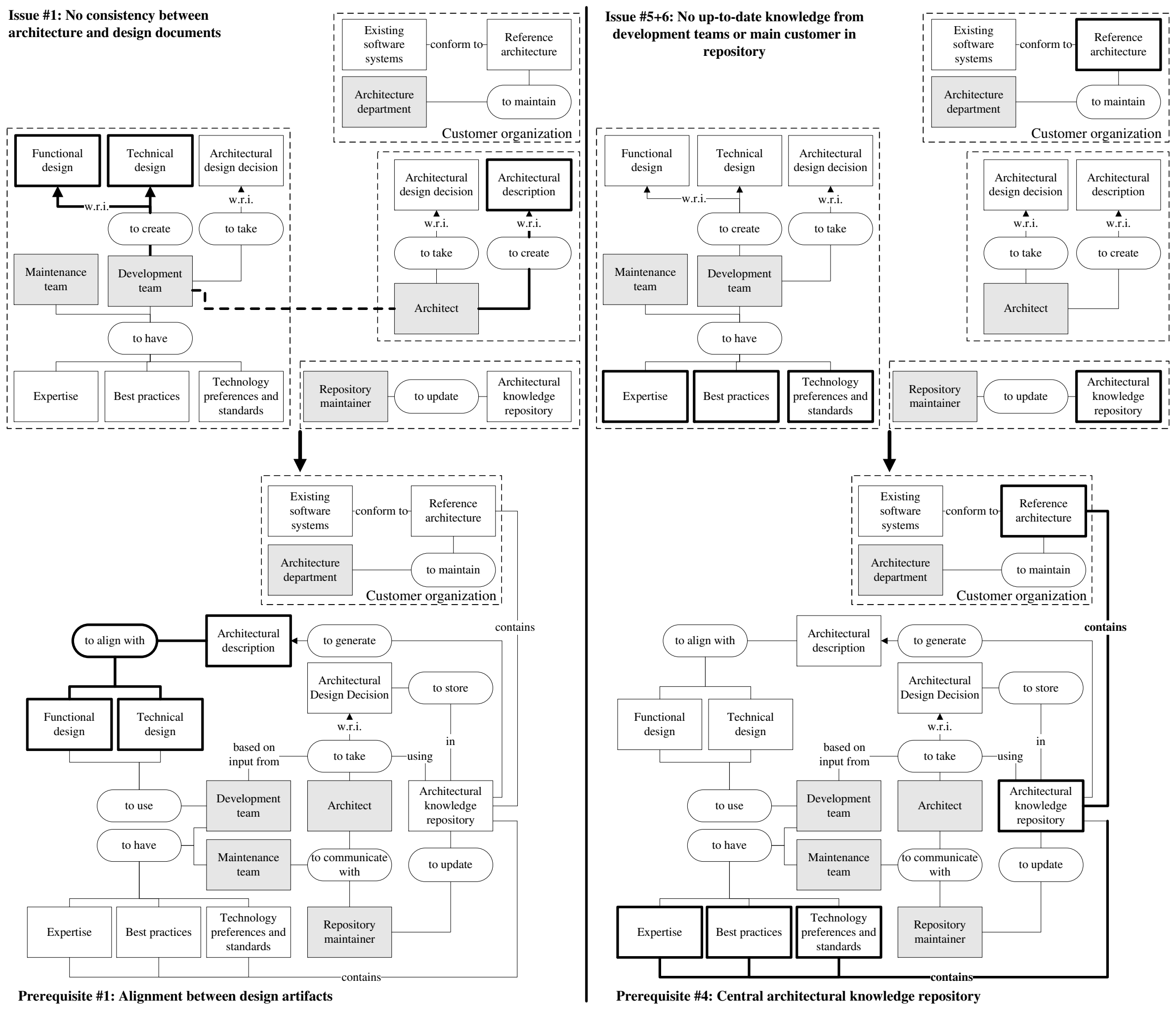

Issue #1: No consistency between architecture and design documents
Existing software systems
conform to
Reference architecture
Architecture department
to maintain
Customer organization
Functional design
Technical design
Architectural design decision
w.r.i.
to create
to take
Maintenance team
Development team
to have
Expertise
Best practices
Technology preferences and standards
Architectural design decision
Architectural description
w.r.i.
to take
to create
Architect
Repository maintainer
to update
Architectural knowledge repository
to align with
Architectural description
to generate
contains
Architectural Design Decision
to store
based on input from
to take
using
in
to use
Architect
to communicate with
to update
Repository maintainer
Prerequisite #1: Alignment between design artifacts
Issue #5+6: No up-to-date knowledge from development teams or main customer in repository
Prerequisite #4: Central architectural knowledge repository

Figure 4. Addressing Architectural Knowledge Sharing Issues

stakeholders'.

3. **Architects fulfill a central role.** The architects need to fulfill a central role in the architecting process. This guarantees better communication with all involved stakeholders through frequent formal and informal meetings, direct involvement of developers in decision-making, and better collaboration with the maintenance teams. This prerequisite addresses issues #3: *'No explicit collaboration with maintenance teams'* and #4: *'No feedback from developers to architects'*.

4. **Central architectural knowledge repository.** A central architectural knowledge repository allows for storing valuable input on the decision-making process from all stakeholders involved. This improvement is visualized in the right part of Figure 4. In the top right of this figure, the current situation is depicted. Here, the architectural knowledge repository is isolated and valuable knowledge, such as experience from developers or knowledge stored in the reference architecture within the customer organization, is not contained in this repository. In the improved situation this knowledge is stored in a 'central' architectural knowledge repository. The thick lines indicate that this repository is positioned as a central storage facility for architectural knowledge. This prerequisite therefore directly addresses issues #5: *'No up-to-date knowledge from development teams in repository'* and #6: *'No up-to-date knowledge from main customer in repository'*.

In the discussion of the four prerequisites for successful architectural knowledge sharing, we have indicated how the issues found in the case study organization are addressed.

The mapping between the six identified issues and the prerequisites that address these issues, is further illustrated in the left part of Figure 5.

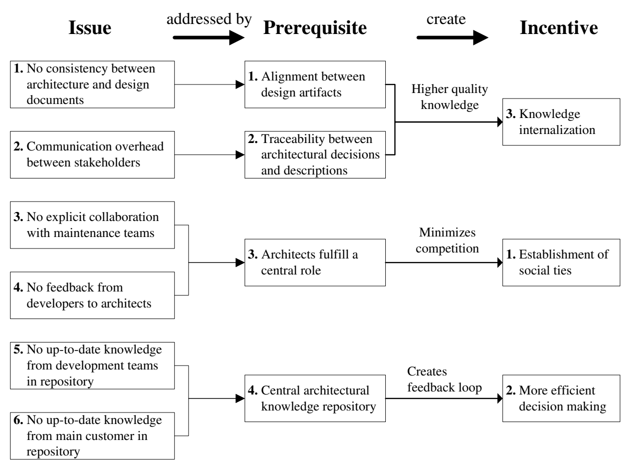

Issue
addressed by
Prerequisite
create
Incentive
1. No consistency between architecture and design documents
2. Communication overhead between stakeholders
3. No explicit collaboration with maintenance teams
4. No feedback from developers to architects
5. No up-to-date knowledge from development teams in repository
6. No up-to-date knowledge from main customer in repository
1. Alignment between design artifacts
2. Traceability between architectural decisions and descriptions
3. Architects fulfill a central role
4. Central architectural knowledge repository
Higher quality knowledge
Minimizes competition
Creates feedback loop
3. Knowledge internalization
1. Establishment of social ties
2. More efficient decision making

Figure 5. Mapping Prerequisites to the Identified Issues and Incentives

We argue that architectural knowledge sharing in practice significantly improves if all four of these prerequisites are met. In order to show the importance of these prerequisites, the right part of Figure 5 illustrates that these prerequisites create the identified incentives for sharing architectural knowledge. These incentives are created as follows. Ensuring *'alignment between design design artifacts'* and *'adding traceability between architectural decisions and descriptions'* both improve the quality of knowledge that is made explicit and subsequently shared. As described in Section 3, this positively affects the amount of knowledge that is internalized by stakeholders, hence creating the *'knowledge internalization'* incentive.

The incentive *'establishment of social ties'* is created by allowing the *'architects to fulfill a central role'* in the architecting process. The improved communication initiated by architects can create a bond between stakeholders that induces architectural knowledge sharing. The internal competition is minimized as architects stress the need for cooperation, and by working more closely together the trust between stakeholders also increases. More extensive knowledge sharing establishes social ties, as explained in Section 3.

Finally, implementing a *'central architectural knowledge repository'* enables a feedback loop between developers, maintainers and architects. All stakeholders use the central repository to monitor the status and progress of the architecture, and are able to participate in the decision making process by sharing expertise, best practices and common sense. This improves the overall group performance. As explained in Section 3 the existence of this feedback loop results in *'more efficient decision making'*.

In this section we have shown that the issues identified during the case study are all addressed if methods and means to share architectural knowledge meet the proposed prerequisites. Next to this, the fact that the four prerequisites map onto the three, context-independent, incentives identified in Section 3, denotes the value of these prerequisites; not only in the context of the case study organization, but in general. Sharing architectural knowledge is crucial in any architecting process; creating the necessary incentives motivates stakeholders to do so in an efficient and successful way.

8. Conclusions and Future Work

Sharing architectural knowledge is crucial to prevent this knowledge from dissipating. However, successful knowledge sharing can only be achieved if the necessary incentives are created. In this paper we have identified three incentives for architectural knowledge sharing. We argue that these incentives need to be created in order to promote successful architectural knowledge sharing.

We have examined how architectural knowledge is shared in a large software development organization. In this organization we identified several types of stakeholders with various interests, including a customer organization that poses constraints on the software architecture. Due to emerging trends such as globalization, offshoring, and virtual organizations, we believe that settings similar to the one found in our case study are not limited to the case study organization alone.

We have modeled the architecting process of the case study organization using four different perspectives. This allowed us to diagnose the issues related to architectural knowledge sharing. Next to acting as a diagnosis instrument, our modeling approach served as an implementation instrument to help us suggest how the architecting process of the case study organization should be improved. Together with the management of the case study organization it has been decided to put effort on implementing these improvements, by meeting the four identified prerequisites for successful architectural knowledge sharing.

Next to assisting the case study organization in improving their architecting process, we have demonstrated how the four prerequisites create the three identified incentives for architectural knowledge sharing. This demonstration indicates that these prerequisites are vital to foster successful architectural knowledge sharing; not only in the context of the case study organization, but in general.

Our ongoing research focuses on defining and implementing methods and tools for sharing architectural knowledge. On the long term, we envision a grid-like system that encompasses all kinds of services to manage and share architectural knowledge within and between organizations. The results of this paper indicate that in order to make this architectural knowledge platform a success, it is important that these services further enable the proposed prerequisites for architectural knowledge sharing.

Acknowledgment

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] L. Argote. *Organizational Learning : Creating, Retaining, and Transferring Knowledge*. Kluwer Academic, 1999.

[2] M. A. Babar, I. Gorton, and R. Jeffery. Toward a Framework for Capturing and Using Architecture Design Knowledge. Technical report unsw-cse-tr-0513, The University of New South Wales, June 2005.

[3] V. R. Basili, G. Caldiera, and D. H. Rombach. The Experience Factory. In *Encyclopedia of Software Engineering*, volume 2, pages 469–476. John Wiley & Sons Inc., 1994.

[4] R. L. Baskerville. Investigating Information Systems with Action Research. *Communications of the AIS*, 2(3es), 1999.

[5] L. Bass, P. Clements, and R. Kazman. *Software Architecture in Practice*. SEI Series in Software Engineering. Addison-Wesley Pearson Education, Boston, second edition, 2003.

[6] J. Bosch. Software Architecture: The Next Step. In F. Oquendo, B. Warboys, and R. Morrison, editors, *1st European Workshop on Software Architectures (EWSA)*, volume 3074, Lecture Notes in Computer Science, pages 194–199, St. Andrews, UK, 2004. Springer-Verlag.

[7] H. Bresman, J. Birkinshaw, and R. Nobel. Knowledge Transfer in International Acquisitions. *Journal of International Business Studies*, 30(3):439–462, 1999.

[8] J. Cummings. Knowledge Sharing: A Review of the Literature. Technical report, The World Bank Operations Evaluation Department, 2003.

[9] T. Dingsøyr, P. Lago, and H. van Vliet. Rationale Promotes Learning about Architectural Knowledge. In *8th International Workshop on Learning Software Organizations (LSO)*, pages 39–47, Rio de Janeiro, Brazil, 2006.

[10] P. Eeles. Characteristics of a Software Architect. Technical report, Available online: http://www-128.ibm.com/developerworks/rational/library/mar06 /eeles/index.html 2006.

[11] R. Farenhorst. Tailoring Knowledge Sharing to the Architecting Process. In *First ACM Workshop on SHaring ARchitectural Knowledge (SHARK)*, Torino, Italy, 2006.

[12] R. Farenhorst, R. C. de Boer, R. Deckers, P. Lago, and H. van Vliet. What's in Constructing a Domain Model for Sharing Architectural Knowledge? In *18th International Conference on Software Engineering and Knowledge Engineering (SEKE)*, San Francisco Bay, USA, 2006.

[13] T. Ghosh. Creating Incentives for Knowledge Sharing. Technical report, MIT Open Courseware, Sloan school of management, Cambridge, Massachusetts, USA, 2004.

[14] T. Haldin-Herrgard. Difficulties in Diffusion of Tacit Knowledge in Organizations. *Journal of Intellectual Capital*, 1(4):357–365, 2000.

[15] C. Hofmeister, P. Kruchten, R. L. Nord, H. Obbink, A. Ran, and P. America. Generalizing a Model of Software Architecture Design from Five Industrial Approaches. In *5th IEEE/IFIP Working Conference on Software Architecture (WICSA)*, pages 77–86, Pittsburgh, Pennsylvania, 2005.

[16] IEEE. IEEE Recommended Practice for Architectural Description of Software-Intensive Systems. Standard 1471-2000, IEEE, 2000.

[17] A. Jansen and J. Bosch. Software Architecture as a Set of Architectural Design Decisions. In *5th IEEE/IFIP Working Conference on Software Architecture (WICSA)*, pages 109–120, Pittsburgh, Pennsylvania, USA, 2005.

[18] P. Kruchten, P. Lago, and H. van Vliet. Building up and Reasoning about Architectural Knowledge. In *2nd International Conference on the Quality of Software Architectures (QoSA)*, LNCS, pages 43–58, 2006.

[19] W. Kunz and H. Rittel. Issues as Elements of Information Systems. Technical report, University of California, 1970.

[20] V. Lakshminarayanan, W. Liu, C. L. Chen, S. Easterbrook, and D. E. Perry. Software Architects in Practice. Report, Empirical Software Engineering Lab (ESEL), University of Texas, Austin, USA, October 2005.

[21] I. Nonaka and H. Takeuchi. *The Knowledge-Creating Company*. Oxford University Press, 1995.

[22] M. Shaw and P. Clements. The Golden Age of Software Architecture. *IEEE Software*, 23(2):31–39, 2006.

[23] A. Tang, M. A. Babar, I. Gorton, and J. Han. A Survey of the Use and Documentation of Architecture Design Rationale. In *5th IEEE/IFIP Working Conference on Software Architecture (WICSA)*, pages 89–98, Pittsburgh, Pennsylvania, USA, 2005.

[24] J. Tyree and A. Akerman. Architecture Decisions: Demystifying Architecture. *IEEE Software*, 22(2):19–27, 2005.

[25] J. S. van der Ven, A. Jansen, P. Avgeriou, and D. K. Hammer. Using Architectural Decisions. In C. Hofmeister, I. Crnkovic, and R. B. S. Reussner, editors, *Perspectives in Software Architecture Quality, 2nd International Conference on the Quality of Software Architectures (QoSA)*, Universität Karlsruhe, Facultät für Informatik, 2006.

[26] J. S. van der Ven, A. Jansen, J. Nijhuis, and J. Bosch. Design decisions: The Bridge between Rationale and Architecture. In A. Dutoit, editor, *Rationale Management in Software Engineering*, pages 329–346. Springer-Verlag, 2006.

[27] Z. Wang, K. Sherdil, and N. H. Madhavji. ACCA: An Architecture-centric Concern Analysis Method. In *5th IEEE/IFIP Working Conference on Software Architecture (WICSA)*, pages 99–108, Pittsburgh, Pennsylvania, USA, 2005.