

Experiences with a Wiki to Support Architectural Knowledge Sharing

Rik Farenhorst
Department of Computer Science
VU University Amsterdam
the Netherlands
rik@cs.vu.nl

Hans van Vliet
Department of Computer Science
VU University Amsterdam
the Netherlands
hans@cs.vu.nl

ABSTRACT

Wikis are becoming increasingly popular knowledge management systems in both software engineering research and practice. In this experience paper we explore the applicability of wikis in the software architecting process, a process which is knowledge-intensive in nature. To this end we report on our experiences with creating and using a wiki environment in the architecture department of a large organization. Next we compare this wiki to existing tools for architectural knowledge management. It turns out that whereas existing tools are stronger in codifying specific architectural knowledge concepts, a wiki prevails as versatile and all-round knowledge management environment. To assist researchers and practitioners in setting up such an environment, we conclude this paper by proposing a number of dos and don'ts for wiki usage in the software architecting process.

1. INTRODUCTION

In the last decade, research and industry have primarily considered a software architecture as a high-level design captured in sets of components and connectors [3], which can be represented using various viewpoints [6]. In recent years, there has been an increasing awareness that not only the architecture design itself is important to capture, but also the knowledge pertaining to it, which is often referred to as 'architectural knowledge'. This awareness has made architectural knowledge management an increasingly important topic in software architecture research.

Wikis are becoming increasingly popular knowledge management systems in both software engineering research and practice, indicated by a number of recent papers about this subject. Bachmann and Merson describe their experiences with applying a wiki environment for supporting architecture documentation [2]. Schuster et al. propose a wiki-based tool to document architectural design decisions [21].

In an attempt to more precisely define the potential merit of wikis in software architecting, in this paper we define four

main characteristics of software architecting. Based on this characterization we analyze what contribution a wiki could offer to a typical architecting process. To place our analysis results in context we compare the strengths and weaknesses of a wiki to some existing tools that support architectural knowledge management.

We have experimented with creating and using a wiki environment in the architecture department of NPK, a large Dutch software development organization. By working together with, and interviewing several architects of NPK we were able to acquire a solid understanding of a wiki's potential in the software architecting process. To assist researchers and practitioners in setting up such an environment, we conclude this paper by sharing our experiences, which have been reflected in a number of dos and don'ts for wiki usage in the software architecting process.

2. SOFTWARE ARCHITECTING

Software architects are typical knowledge workers. They are considered the 'spider-in-the-web' of the software development process. While communicating with both business and technical stakeholders, they need to negotiate and find a balance between these stakeholders' needs and concerns, take the important architectural design decisions, reuse existing architectural styles and patterns, evaluate and review architectural solutions and create architectural standards and guidelines.

In the next subsection we characterize software architecting by focusing on respectively collaboration, decision making, distributed development and reusable assets. This characterization, which is based upon earlier research [12], sheds light on what software architecting entails and what knowledge needs to be managed in this process.

2.1 Software Architecture Needs

With respect to how the software architecture field developed over the past few years, four trends can be observed:

Increased collaboration. In recent years, software projects have increased considerably in size and so have the number of stakeholders working on these projects. In these projects collaboration takes place between various (types of) stakeholders. Architectural discussions are held between large groups of stakeholders, who vary in background and expertise. Consequently, the knowledge flows in the architecting process are diverse and frequent, which calls for proper support to enable efficient communication and collaboration between team members.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Wiki4SE September 8, 2008, Porto, Portugal.

Copyright 200X ACM X-XXXXX-XX-X/XX/XX ...\$5.00.

Focus on decision making. Architecting is a typical consensus decision making process. In order to take major architectural decisions, reaching consensus between the main stakeholders is vital. Architectural decisions can have a relatively large impact on the final system, because they constrain lower-level design issues. This trait makes these decisions often irreversible, making the need to formulate and communicate a proper rationale for these decisions even higher. Therefore, architects spend significant amount of time in negotiating and discussing (both verbally and by means of documentation) with the parties involved, and in making tradeoffs between various design alternatives to reach consensus. During this process the architect also needs to guarantee that decisions taken do not conflict with company regulations, reference architectures or other standards and rules.

Distributed development. Architecting is often done by architects who are spread over multiple teams in one organization. When the organization has multiple sites, online communication replaces traditional coffee room talks, which are not an option anymore to share valuable knowledge. Global Software Development (GSD) affects the way the architecting process is organized even further. Architectural knowledge management needs to take into account the challenges that arise as a result of the geographical, temporal, and socio-cultural distance innate to GSD [8].

Need for reusable assets. With the maturation of the software architecture field in both research and practice, architectural solutions (e.g. patterns and styles) are better formulated, using well-documented architectural views become common practice, and architectural standards and principles are more frequently embraced by the stakeholders. Consequently, architects increasingly benefit from such reusable assets; they do not always have to reinvent the wheel, but instead can learn from prior experience. In order to reuse such architectural knowledge, retrieving the right knowledge at the right time is considered crucial [11].

2.2 Architectural Knowledge Needs

Stakeholders in the software architecting process produce and consume much knowledge. The characteristics of software architecting we discussed in Section 2 call for proper knowledge management facilities to support practitioners in their daily work. In this section we elaborate in more detail upon the knowledge needs of the stakeholders involved by focusing on ‘architectural knowledge’ and ‘non-architectural knowledge’, respectively.

Architectural knowledge. Two types of architectural knowledge exist: application-specific architectural knowledge and application-generic architectural knowledge [17]. Application-specific architectural knowledge is important knowledge to understand why an architecture for a software system is the way it is. It is often defined as the set of design decisions [4, 13], including the rationale for these decisions [23], together with the resulting architectural design [16]. In this paper we follow this definition:

$$\text{Architectural Knowledge} = \{\text{architectural design decisions} + \text{rationale}\} + \text{architectural design}$$

Architectural design decisions and their rationale can be modeled in various ways. Kruchten proposes an ontology of design decisions in which certain properties, relationships

Concern	The customer requests high maintainability of the system, so we need to find a way to handle complexity.
Alternative(s)	- Client-server architecture - 3-tier architecture
Ranking criterion	- maintainability of the system
Architectural design decision	
Identifier	#1
Description	The application will be based on a 3-tier architecture that consists of a user interface tier, a business logic tier, and a database tier.
Status	This decision has been approved by the project manager
Relationship(s)	This decision is related to decision #3 “MySQL database”
Rationale	A 3-tier approach leads to clean code because the user interface is not allowed to access the data directly. Also the use of design patterns is stimulated when using a 3-tier architecture.

Table 1: Example Architectural Design Decision

and states of a decision are codified [15]. Tyree and Akerman propose a template in which they store all related information of a decision in tables [22], such as concerns, alternatives, constraints, etc. Based on a core model of architectural knowledge [9] we have devised a template for storing architectural design decisions, which is closely related to Tyree and Akerman’s one. An example decision based on this template is depicted in Table 1.

The resulting architectural design is often codified using a number of architectural views, which are constructed using notations and structures defined in an associated viewpoint [6]. These views are often modeled in specialized tools such as architecture design tools, or in drawing tools such as MS Visio. Architecture documents usually aggregate the views with the necessary explanation, plus some additional insights in the main architectural decisions taken.

Application-generic architectural knowledge is knowledge that is often tacit [19] and built up by the architect as experience, domain knowledge or expertise. It includes architectural styles, patterns, and tactics, and knowledge about when and how to use specific technologies, tools and methods. Application-generic architectural knowledge enables practitioners in producing or consuming application-specific architectural knowledge (e.g. taking a high-quality architectural decision based on a proper rationale) and is therefore also very important, as confirmed by research of Clements et al. about the duties, skills and knowledge of architects [7].

Non-architectural knowledge. Apart from architectural knowledge needs, stakeholders in the architecting process also require additional non-architectural knowledge to perform their daily tasks, just like other IT professionals do. This non-architectural knowledge is very broad but equally crucial for the architecting process as architectural knowledge is. Non-architectural knowledge includes process management information, information about the organization and the context and domain, information about the business processes of the internal and customer organization, knowledge about both architectural and non-architectural standards (e.g. TOGAF, Sabanes-Oxley), and knowledge about the current status in the organization (e.g. ‘who is doing what’ or ‘who knows what’).

2.3 Architectural Knowledge Sharing Support

All the knowledge discussed in Section 2.2 needs to be managed and shared to effectively assist stakeholders in their daily work. In recent years, various tools have been proposed that specifically manage architectural knowledge such as design decisions, and rationale. These tools include ADDSS [5],

DGA DDR [10], PAKME [1], and Archium [14]

Capilla et al. have proposed a web-based tool, ADDSS, for recording and managing architectural design decisions [5]. Falessi et al. have devised a specific design decision rationale documentation technique, which is driven by the decision goals and design alternatives available [10]. Ali Babar et al. have proposed a Process-based Architecture Knowledge Management Environment (PAKME) that allows storing application-generic architectural knowledge (such as general scenarios, patterns, and quality attributes), and project specific architecture knowledge (such as concrete scenarios, contextualized patterns, and quality factors) [1]. Archium is a tool environment proposed by Jansen et al. aimed at establishing and maintaining traceability between design decision models and the software architecture design [14].

All the aforementioned tools have been designed for modeling, retrieving and editing architectural design decisions, rationale and related architectural concepts. However, support for collaboration, decision making and distributed development is not always strongly supported by these tools, which raises the question whether they are suitable as overall knowledge management platform in software architecting.

3. A WIKI IN THE ARCHITECTING PROCESS: A CASE STUDY

Ever since the advent of Wikipedia, the interest in wiki-based systems is growing steadily. Numerous wiki software packages have been proposed and commercial organizations are increasingly interested in setting up collaborative workspaces where people meet, produce and consume knowledge. Within the software engineering community the interests in wikis is growing as well, evidenced among others by the birth of the Wiki4SE workshop series.

Over the last half year, we have experimented with a wiki environment in the architecture department of NPK, a large Dutch software development organization. The wiki is built using the Confluence Enterprise wiki software¹. In the next four subsections we elaborate upon our experiences by focusing on the context in which the wiki was used, the adoption strategy we followed, and how architectural and non-architectural knowledge was managed by the wiki, respectively.

3.1 Context

The experiments were conducted at the architecture department of NPK. Architects of this department work on architectural solutions for local and national government. The department consists of around 30 enterprise and IT architects, who work on various projects at customer organizations and internal projects. At least once a week the architects visit the department's office for meetings and sharing experience and ideas with colleagues. Often just coffee room talks and small (often informal) meetings were used to this end, but also some repositories and an intranet website exist in which architectural knowledge is stored. None of the architects had been using a wiki environment before, so apart from being familiar to Wikipedia, they were new to this type of knowledge management platform.

The choice for the Confluence Enterprise wiki had been taken jointly by us (the researchers) and the management of the architecture department. Prior to our experiments

the unit manager had asked one architect to study available wiki environments (e.g. MediaWiki, XWiki, wiki support in MS Sharepoint 2007) for their suitability. In a comparison that was made Confluence turned out to be the most promising wiki environment, mostly due to its 'rich text editing' features and the fact that a Confluence wiki is highly customizable using a plugin-system. Compared to e.g. MediaWiki (the environment on which Wikipedia is hosted) in Confluence wiki pages can be written in a way similar to MS Word (a tool frequently used by architects). With respect to the integration, Confluence made it possible to link from the wiki to existing Sharepoint repositories full of corporate information.

By choosing Confluence we ensured that the wiki was not becoming yet another stand-alone knowledge management environment, like several smaller systems that already existed in the architecture department of NPK. From earlier research we know that stand-alone tools might become difficult to maintain, are hard to keep up-to-date and confuse practitioners, who have difficulty determining which one to use for what purpose and where to find what architectural knowledge [11].

3.2 Adoption strategy

In order to increase the adoption of the wiki in NPK, we decided to start with generating "critical mass" on the wiki with three architects, before showing the wiki to the whole team. By following this Critical Mass pattern [18] we automatically used additional wiki patterns as well, such as the Champion pattern (one of us acted as wiki champion), the Moderator pattern (the three architects were appointed Moderators of specific parts of the wiki), and the FAQ pattern (we made a page with frequently asked questions to help wiki users with basic functionality and features). More useful information on these (and other) patterns can be found in [18] or online².

After working for two months in a core team, which consisted of the champion and three moderators, the wiki had been filled with initial content, the layout had been changed to match corporate styles, and the functionality and quality of Confluence had been thoroughly tested. A screenshot of the wiki's starting page is depicted in Figure 1.

The next step was informing the other colleagues of the wiki's existence. To this end an email was sent among the team members with the wiki's address, login details and some general introduction to the wiki's aim and scope. This introduction email resulted in an immediate boost of wiki activity. Several architects created a personal wiki page within a day, and started adding comments to existing content on the wiki. Others uploaded documents to the wiki in an attempt to share their recent work to the group. It is fair to say that the enthusiasm was wide-spread.

The enthusiasm of the wiki reached other departments within NPK as well. Several colleagues from other departments came by for a demonstration of its capabilities, and became excited as well. We provided several of them with their own 'wiki spaces' so that they could further experiment or play with the wiki themselves. After half a year management of NPK noticed the wiki's popularity and decided to acquire a company-wide license and to place the maintenance of the wiki under the responsibility of the cen-

¹<http://www.atlassian.com/software/confluence/>

²<http://www.wikipatterns.com>

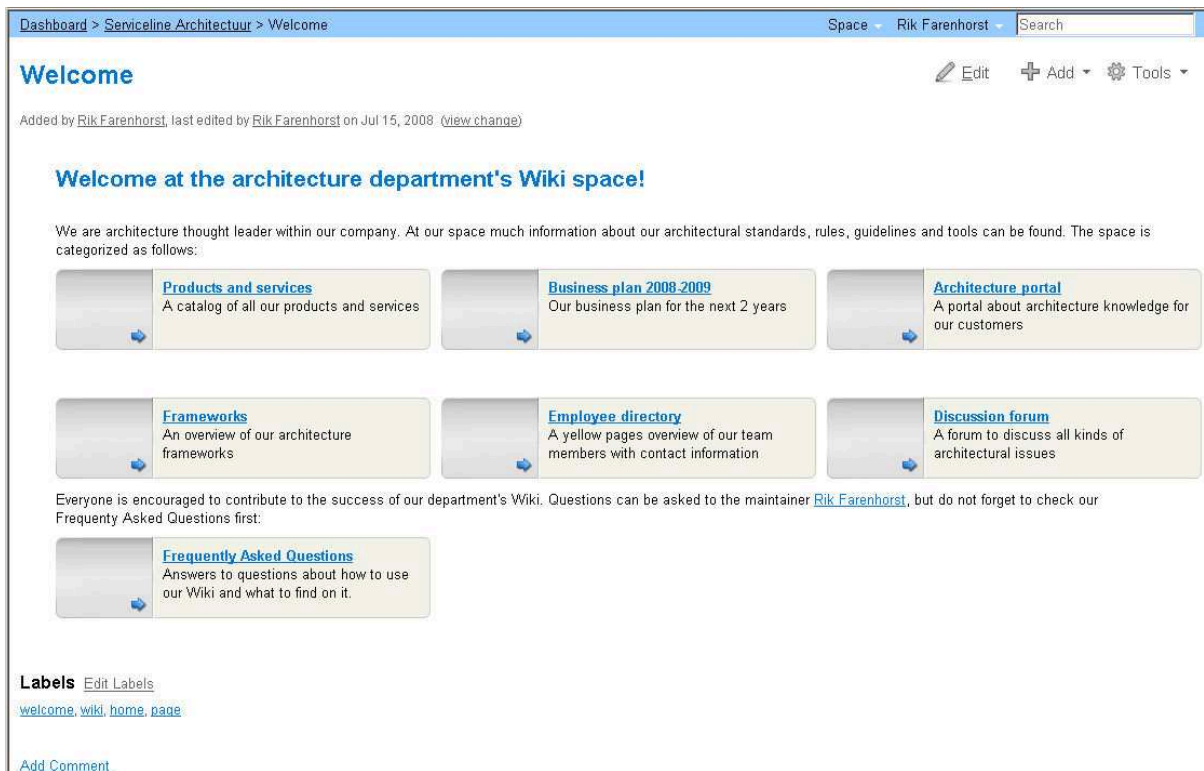


Figure 1: Screenshot of the start page of NPK's Wiki

tral IT department. As a result in the near future the wiki will be used in a more professional way throughout NPK.

3.3 Managing architectural knowledge

We have used the wiki for storing and managing both architectural knowledge and non-architectural knowledge. Parts of the wiki were filled with reusable architectural principles and rules (one per wiki page) after which an overview page was generated using the 'table of contents' plugin. The same approach has been used for architectural knowledge concerned with architectural technologies and patterns.

It should be noted, though, that the architects had some trouble with structuring the architectural knowledge in a suitable way. Being used to writing documents and storing these in repositories with some meta-data, it took a mindset change before some architects understood that storing the knowledge in text-format directly in wiki pages, and labeling these pages for retrieval purposes is all there is to it in a wiki environment. The knowledge entities can be edited directly in wiki-markup, and the documents are redundant, although they might be archived (i.e. added as attachment) for users who wish to read more about the topic.

Unfortunately, the approach sketched above was not that easy. Adding meta-data such as specific properties of architectural decisions was quite cumbersome. Confluence does not provide much support for this. It does, however, offer some plugins that address this topic. One of these is a plugin that allows connections to SQL databases, but we found that this connection is read-only. Consequently, one cannot update database tables from within the wiki pages, which limits the integration with relational databases significantly.

There is also a meta-data plugin in Confluence, but this offers only limited functionality compared to what specific databases based on their own data model could offer. Storing architectural design decisions using a format depicted in Table 1 was therefore not an easy task.

As a solution to model architectural knowledge in the wiki we are currently devising templates to be used by the architects to further codify the architectural knowledge concepts in a uniform way and to ensure that this knowledge is sufficiently enriched with meta-data that is stored with it. Further experimentation is needed to see if these templates offer sufficient guidance to the architects. We also plan to investigate whether we could 'borrow' best practices from the semantic wiki community (see e.g. [20]) to handle architectural knowledge concepts in a more mature way.

Another nuisance when working with the wiki was the support for images. Architects often use diagrams and models to depict their architectural views [6]. They sometimes use specialized modeling tools to this end, but often also construct diagrams in MS Word or Visio. When storing architectural documentation in wiki pages the architects found that each image needs to be added as separate attachment and that editing images from within Confluence is not possible. Since architectural views are update frequently – based on stakeholder discussions and negotiations – this is a drawback when using the wiki as main storage place for architectural deliverables.

3.4 Managing non-architectural knowledge

To our surprise one of the main strengths of the wiki was not its support for managing architectural knowledge, but

how it offers a central storage place for non-architectural knowledge that architects produce and consume during their daily activities. Architects put meeting minutes on the wiki, but also a holiday roster, and a list of the team members' birthdays. They also constructed a 'yellow pages' system by creating a wiki page for each team member, including contact information, current tasks and activities, and interests and expertise. From what we observed it is exactly these 'related' knowledge pages that made the wiki fun to use and to visit regularly. It attracted the architects to the platform, it decreased the emails being sent with attachments significantly, and it even started some healthy competition between a few architects, who checked the wiki's activity monitor on a daily basis to see who had the honor of being the 'most active contributor'.

4. WIKI'S FOR ARCHITECTURAL KNOWLEDGE SHARING?

Based on our experiences with using a wiki in NPK, we are able to indicate the main strengths and weaknesses of this tool for its use in the architecting process. In order to determine whether wikis are suitable to share architectural knowledge, in the next subsection we compare wikis to the state-of-the-art architectural knowledge management tools introduced in Section 2.3. For this comparison we zoom in on the four main characteristics of the architecting process discussed in Section 2. This allows us to better define the relative strengths and weaknesses of a wiki environment when used as architectural knowledge management platform.

In subsection 4.2 we list a number of dos and don'ts for using wiki's in the software architecture domain. During our experimentation we found that several personal and organizational factors impact the success of a wiki in the software architecting process either positively or negatively. Researchers or practitioners who wish to set up a wiki environment themselves can use our dos and don'ts to increase the chance of success.

4.1 Comparison of AK management tools

As discussed in Section 2, in the architecting process:

1. A lot of *collaboration and communication* takes place due to the large software development teams and the projects these teams work on.
2. *Consensus decision making* is one of the primary activities that stakeholders spend their time on.
3. *Online communication* becomes more and more important due to distributed development which inhibits traditional coffee room talks.
4. *Retrieving architectural assets* is crucial to enable reuse and to deliver high-quality architectural solutions.

Below we compare some architectural knowledge-specific tools (ADDSS, PAKME, DGA DDR, and Archium) with the Confluence wiki by zooming in on these characteristics of architecting.

When taking a close look at the architectural knowledge management tools introduced in Section 2.3 we see that these offer specific support for codifying and retrieving architectural knowledge concepts. In Archium, ADDSS and DGA DDR the main concepts are design decisions, whereas

in PAKME also patterns and styles are included. All of these tools use tailored databases to store and edit the knowledge and use an underlying data model to define the knowledge they focus on. Retrieving architectural knowledge assets is therefore well supported by these tools. As we have described in Section 3 the wiki does not have strong support for modeling architectural knowledge concepts out-of-the box, which makes it harder to retrieve such knowledge using the wiki than when using one of the AK-specific tools.

When looking at the support for online communication, the wiki stands out. Current versions of ADDSS, DGA DDR, PAKME and Archium run locally on a pc and therefore cannot be collectively used by practitioners working on different sites. The wiki on the other hand is web-based and aimed at targeting communities. It supports concurrent editing of content, version management, and users can easily provide comments on wiki pages created by others.

As described in Section 3, a major strength of the wiki is its support for managing non-architectural knowledge, such as meeting minutes, plannings, deadlines, progress reports, etc. This non-architectural knowledge can be stored in the various wiki pages, and can be very supportive for stakeholders involved in consensus decision making. They could use the wiki as central archiving environment for all information used in software architecture projects. This makes the wiki a much more 'all-round' tool than the AK-specific tools, which offer only codification techniques for architectural knowledge concepts.

Another drawback of the AK-specific tools is that they fall short when it comes to supporting collaboration. In the architecting process architects typically work on designing and describing or communicating architecture solutions. For the description part the wiki offers a nice groupware platform where architects can collaboratively produce new information, review each others work, etc. And since architecture projects are growing larger and larger, involving more and more stakeholders, groupware support is much welcomed. In addition, the architectural knowledge-specific tools tend to have a rather high learning curve, whereas the wiki's usability resembles both text-editors and websites, to which architects are already familiar.

4.2 Dos and don'ts for Wikis in the architecting process

If we summarize the comparison described in Section 4.1 we see that the wiki is strong in its support for collaboration, non-architectural knowledge and online communication, whereas the architectural knowledge-specific tools offer more thorough support for the retrieval of reusable architectural knowledge assets. Based on these results we conjecture that a wiki can make a fine architectural knowledge management environment.

Keep in mind though that a successful wiki is something that takes time to set up. In this section we provide some dos and don'ts when setting up a wiki environment in the architecting process. These dos and don'ts have been defined based on 1) multiple interviews and discussions with the architects at NPK who were most active on the department's wiki, and 2) a review of available literature on wiki patterns [18], based on which we analyzed which ones are most significant in the software architecting process.

1. [**Do**] **start small**. Define a startup period in which preparations on the wiki environment begin. Produce

an initial set of wiki content with a small group of people, who use this startup period to become familiar with the wiki system and the ins-and-outs of wiki markup. Appoint one or more wiki champions who will promote the wiki to colleagues and assign moderators who will safeguard the quality of the knowledge stored in the wiki. Make sure that the champions and moderators have a clear idea on their responsibilities. Only when all the above has been arranged, the wiki should be announced to the whole team or department.

2. **[Do] allow and stimulate all types of knowledge.** Do not restrict the wiki to only very specialized architectural knowledge such as design decisions, architectural principles or standards. As discussed, a wiki is particularly strong in managing non-architectural knowledge, so make sure that users have sufficient freedom to manage such knowledge in the wiki as well.
3. **[Do] strive for integration.** To increase the chance that users consider the wiki an integral part of their daily work, integration with existing systems such as file shares, a Sharepoint site, etc. is crucial. Ideally the wiki is positioned as core system for codifying both non-architectural and architectural knowledge, and when necessary, links and traces to additional systems are made.
4. **[Don't] impose too much structure.** It might be very tempting to introduce lots of templates, rules and guidelines for users to follow or use. However, one of the wiki principles is that a wiki should become a self-organizing system in which users incrementally improve the quality and correct mistakes. Introducing templates, e.g. to model AK concepts is a good thing to do, but do not overdo it, since it might kill the 'fun' of working with a wiki.
5. **[Don't] restrict access.** Try to keep the wiki contents open to as many users as possible instead of restricting access to specific stakeholders depending on their role. By granting access to as much architectural knowledge as possible, reuse is stimulated and the chance of users correcting or improving knowledge grows. Classified information defies the wiki philosophy of collaborative editing, so if access to specific AK needs to be restricted, consider storing it on other systems instead of on the wiki.

Please note that four of the five dos and don'ts above are not necessarily 'architecting-specific', and could well apply to other domains as well. The one that stands out, though, is the fact that the scope of the wiki should encompass both architectural knowledge and non-architectural knowledge. This permits the wiki to become an all-round knowledge management environment that supports the architecting process in a more mature way compared to the other tools.

5. CONCLUSIONS

Software architecting is very knowledge-intensive process, which benefits from architectural knowledge management support. In this paper we have explored the applicability of wikis to offer such support. To this end we have set up and

experimented with a wiki environment at the architecture department of a large software development organization.

Based on our experiments we conjecture that wikis can be very supportive to the architecting process. They could be employed as all-round environments, offering strong support for collaboration, online communication, and consensus decision making. Nevertheless, to guarantee the wiki's success, a proper strategy must be chosen for setting up and adoption of the wiki in an organization. To assist both researchers and practitioners in this task, we have established a number of dos and don'ts for wiki usage in the architecting process.

The most apparent limitation of the wiki we found is that codification of architectural knowledge is less straightforward than with using state-of-the-art tools that are explicitly constructed for this task, and requires additional support by means of plugins or templates. This drawback, however, is outweighed by the wiki's versatility to manage all sorts of non-architectural knowledge related to the architectural knowledge produced, as well as its support for integration with other tools, intuitive interface and low learning curve.

Acknowledgements

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 in-FormatIoN about architectural knowledge.

6. REFERENCES

- [1] M. Ali Babar and I. Gorton. A Tool for Managing Software Architecture Knowledge. In *2nd Workshop on SHaring and Reusing architectural Knowledge - Architecture, rationale, and Design Intent (SHARK/ADI)*, Minneapolis, USA, 2007.
- [2] F. Bachmann and P. Merson. Experience Using the Web-Based Tool Wiki for Architecture Documentation. Technical Report CMU/SEI-2005-TN-041, Carnegie Mellon University, Software Engineering Institute, 2005.
- [3] L. Bass, P. Clements, and R. Kazman. *Software Architecture in Practice*. Addison-Wesley, Boston, second edition, 2003.
- [4] J. Bosch. Software Architecture: The Next Step. In *1st European Workshop on Software Architectures (EWSA)*, pages 194–199, St. Andrews, UK, 2004.
- [5] R. Capilla, F. Nava, S. Pérez, and J. C. Dueñas. A Web-based Tool for Managing Architectural Design Decisions. In *1st ACM Workshop on SHaring ARchitectural Knowledge (SHARK)*, Torino, Italy, 2006.
- [6] P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, R. Nord, and J. Stafford. *Documenting Software Architectures: Views and Beyond*. Addison Wesley, 2002.
- [7] P. Clements, R. Kazman, M. Klein, D. Devesh, S. Reddy, and P. Verma. The Duties, Skills, and Knowledge of Software Architects. In *6th Working IEEE/IFIP Conference on Software Architecture (WICSA)*, 2007.

- [8] V. Clerc. Towards Architectural Knowledge Management Practices for Global Software Development. In *3rd Workshop on SHaring and Reusing architectural Knowledge (SHARK)*, Leipzig, Germany, 2008.
- [9] R. C. de Boer, R. Farenhorst, P. Lago, H. van Vliet, V. Clerc, and A. Jansen. Architectural Knowledge: Getting to the Core. In *3rd International Conference on the Quality of Software-Architectures (QoSA)*, Boston, USA, 2007.
- [10] D. Falessi, M. Becker, and G. Cantone. Design Decision Rationale: Experiences and Steps Ahead Towards Systematic Use. In *1st ACM Workshop on SHaring ARchitectural Knowledge (SHARK)*, Torino, Italy, 2006.
- [11] R. Farenhorst, R. Izaks, P. Lago, and H. van Vliet. A Just-In-Time Architectural Knowledge Sharing Portal. In *7th Working IEEE/IFIP Conference on Software Architecture (WICSA)*, 2008.
- [12] R. Farenhorst, P. Lago, and H. van Vliet. Effective Tool Support for Architectural Knowledge Sharing. In *1st European Conference on Software Architecture (ECSA)*, Madrid, Spain, 2007.
- [13] A. Jansen and J. Bosch. Software Architecture as a Set of Architectural Design Decisions. In *5th Working IEEE/IFIP Conference on Software Architecture (WICSA)*, pages 109–120, Pittsburgh, USA, 2005.
- [14] A. Jansen, J. S. van der Ven, P. Avgeriou, and D. K. Hammer. Tool Support for Architectural Decisions. In *6th Working IEEE/IFIP Conference on Software Architecture*, Mumbai, India, 2007.
- [15] P. Kruchten. An Ontology of Architectural Design Decisions in Software-Intensive Systems. In *2nd Groningen Workshop on Software Variability Management*, Groningen, The Netherlands, 2004.
- [16] P. Kruchten, P. Lago, and H. van Vliet. Building up and Reasoning about Architectural Knowledge. In C. Hofmeister, I. Crnkovic, and R. Reussner, editors, *2nd International Conference on the Quality of Software Architectures (QoSA)*, volume 4214 Springer LNCS, pages 39–47, Stockholm, Sweden, 2006.
- [17] P. Lago and P. Avgeriou. 1st Workshop on SHaring and Reusing ARchitectural Knowledge, Final Workshop Report. *ACM SIGSOFT Software Engineering Notes*, 31(5):32–36, 2006.
- [18] S. Mader. *Wiki Patterns*. Wiley, 2007.
- [19] I. Nonaka and H. Takeuchi. *The Knowledge-Creating Company*. Oxford University Press, 1995.
- [20] S. Schaffert, F. Bry, J. Baumeister, and M. Kiesel. Semantic Wikis. *IEEE Software*, 25(4):8–11, 2008.
- [21] N. Schuster, O. Zimmermann, and C. Pautasso. ADkwik: Web 2.0 Collaboration System for Architectural Decision Engineering. In *19th International Conference on Software Engineering & Knowledge Engineering (SEKE)*, Skokie, USA, 2007.
- [22] J. Tyree and A. Akerman. Architecture Decisions: Demystifying Architecture. *IEEE Software*, 22(2):19–27, 2005.
- [23] J. S. van der Ven, A. Jansen, J. Nijhuis, and J. Bosch. Design decisions: The Bridge between Rationale and Architecture. In *Rationale Management in Software Engineering*, pages 329–346. Springer-Verlag, 2006.