The Journal of Systems and Software 84 (2011) 1424–1435

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

The lonesome architect

Johan F. Hoorn [a,*], Rik Farenhorst [b], Patricia Lago [c], Hans van Vliet [c]

[a] Center for Advanced Media Research Amsterdam, VU University Amsterdam, De Boelelaan 1081, Amsterdam The Netherlands
[b] DNV-CIBIT B.V., The Netherlands
[c] Department of Computer Science, VU University Amsterdam, The Netherlands

ARTICLE INFO

Article history:
Received 10 June 2010
Accepted 21 November 2010
Available online 2 December 2010

Keywords:
Software architecture
Knowledge sharing
Architecture needs
Support tools
Empirical research

ABSTRACT

Although the benefits are well-known and undisputed, sharing architectural knowledge is not something architects automatically do. In an attempt to better understand what architects really do and what kind of support they need for sharing knowledge, we have conducted large-scale survey research. The results of our study indicate that architects can be characterized as rather lonesome decision makers who mainly consume, but neglect documenting and actively sharing architectural knowledge. Acknowledging this nature of architects suggests ways to develop more effective support for architectural knowledge sharing.

1. Introduction

Much has been written on software architecture theory and practice. Many of us are familiar to existing best practices on describing architectures, methods to evaluate these architectures, and process guidelines to ensure that architecture development blends in nicely with the software development process. One of the core processes in software architecting is decision making. In this decision-making process the architects constantly need to balance all kinds of constraints, such as requirements and needs of the customer(s) and other stakeholders, technological and organizational limitations, existing expertise and experience, as well as organizational or architectural best-practices and standards. Based on sound judgment, architects are expected to take the best architectural design decisions and to communicate and motivate these decisions.

Over the past few years, an increasing attention is put on the role of decision making in the software architecting process (Jansen and Bosch, 2005; Kruchten et al., 2006; Tyree and Akerman, 2005). Consequently, the role of the architect in this process is a frequently recurring topic of discussion, and more and more researchers and practitioners deliberate on what a proper set of duties, skills, and knowledge of architects would be (Clements et al., 2007). In addition, researchers have proposed various tools to support the architect (Ali Babar and Gorton, 2007; Capilla et al., 2006; Farenhorst et al., 2007; Jansen et al., 2007), who is characterized as an all-round knowledge worker. Such a knowledge worker, as these researchers argue, would definitely benefit from specific support for managing design decisions, modeling architectural solutions, or related activities that can be automated. With such support relevant architectural knowledge is easier to share, which leads to more effective knowledge exchange and reuse and which prevents knowledge vaporization (Bosch, 2004).

Architects often reuse existing assets such as architectural patterns and styles. They negotiate with other stakeholders, or rely on their 'gut feeling'. Sharing support is crucial to prevent loss of architectural knowledge, to exchange experiences and ideas with colleagues, to (re)use architectural expertise and best practices and to train junior employees.

There are various approaches to support architects in sharing architectural knowledge. In practice, however, the adoption of these approaches is limited, perhaps due to a lack of alignment to the architecting process. Research showed that these approaches are often developed from a technological perspective alone (Farenhorst et al., 2007). To keep from falling into this so-called ICT trap (Huysman and de Wit, 2004), we wish to find out what support for architectural knowledge sharing fits the architecting process best. However, the field lacks clear insight into the connection between architecting activities and architectural knowledge sharing support.

The only way to understand what would really help architects is to capture what architects do and what kind of support they need during their main activities. To this end, we constructed a 'theory on architecting', based on a review of state-of-the-art literature and experiences gained during four years of case-study research

* Corresponding author.
E-mail addresses: jf.hoorn@camera.vu.nl (J.F. Hoorn), rik.farenhorst@dnv.com (R. Farenhorst), patricia@cs.vu.nl (P. Lago), hans@cs.vu.nl (H. van Vliet).

0164-1212/$ – see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2010.11.909

in a large software development organization (Farenhorst and van Vliet, 2009). Our theoretical framework consisted of a number of core architecting activities and a number of support methods that assist architects in sharing architectural knowledge during these activities. Further, we hypothesized the existence of a number of 'patterns', indicating that certain methods for sharing architectural knowledge are more supportive than others to architects involved in a specific activity. We believe that identifying such patterns helps the development and maturation of methods and tools for knowledge sharing.

To validate our framework, we conducted large-scale survey research in four IT organizations in the Netherlands. In total, 279 practicing architects were asked about their daily activities and how important they considered the various types of support for sharing architectural knowledge.

Our survey results showed that architects are complicated individuals. They make lots of architectural decisions but neglect their documentation. Codifying and subsequently sharing architectural knowledge is clearly not one of their most popular activities. While in itself not very surprising, one would expect that support to assist architects in this latter task is something they appreciate. The opposite is true. Architects rather stay in control and are not interested in automated or intelligent support. When it comes to consumption of architectural knowledge, however, support for effective retrieval of (stored) architectural knowledge is on the top of their wish list. This apparent contradiction suggests that architects are rather lonesome decision makers who prefer to spend their working life in splendid isolation.

The remainder of this paper is organized as follows. In Section 2, we outline our theory on architecting activities and support methods for architectural knowledge sharing. In Section 3, we elaborate upon the research methodology of our study. Section 4, presents the data analysis to validate our theory. In Section 5, we reflect on the results and discuss their implications. Section 6 concludes this paper with recommendations on architectural knowledge-sharing support.

This paper is an extended version of Farenhorst et al. (2009a). In this paper we provide more theoretical background, explicate and test more hypotheses, provide the statistical test values and explain our methodological considerations. In addition, we discuss more deeply the consequences of our results for theory and practice of sharing architecting knowledge, referring to a broader range of authors.

2. Theoretical framework

In a recent paper on sharing and reusing architectural knowledge it is argued that architectural knowledge consists of all the knowledge used and produced during architecting (Lago et al., 2008). Such architectural knowledge encompasses knowledge of the problem domain (e.g., architectural requirements, drivers, constraints), the solution domain (e.g., architectural tactics, patterns, styles), and also knowledge entities used in the architecting – or decision-making process – itself, such as design decisions, rationale, and the resulting architecture design (Farenhorst and de Boer, 2009). The broad scope of architectural knowledge calls for knowledge sharing support that is equally broad.

Based on experiences in earlier industrial case studies, combined with established literature on software architecture and knowledge management, we developed a theory to characterize what architects do and what they need for sharing architectural knowledge. In the next three sub-sections, we outline our theory on what kind of activities architects are involved in (Section 2.1), what kind of support they need (Section 2.2), and what patterns between support and activities exist (Section 2.3).

Table 1
Architecting activities.

A1: Communciation
I inform colleagues about the results of my work
My colleagues keep me up-to-date on the results of their work
I explain existing architectural principles to colleagues
During discussions I share my knowledge to colleagues
A2: Decision making
Before I take a decision I weigh the pros and cons of possible solutions
While taking decisions I meet the wishes of the stakeholders
I think about what impact my decisions have on the current architecture
I study the reasoning behind taken design decisions
A3: Quality assessment
I convince stakeholders in order to share the architectural vision
I convince stakeholders about the value of my architectural solution
Stakeholders approach me to discuss about architectural issues
I notify stakeholders about actions that deviate from the architecture
I check whether proposals from stakeholders are in line with the architecture
I judge whether architectural proposals could continue or be executed
A4: Documentation
I create documents that describe architectural solutions
I use (parts of) existing documents while creating new deliverables
I use templates to store architectural knowledge
I write vision documents to inform stakeholders
I write progress reports about the architecture to stakeholders
A5: Knowledge acquisition
I learn from my colleagues about architectural principles
Colleagues learn from me about architectural principles
I read (scientific/professional) literature on architecture
I keep my knowledge up-to-date by searching relevant information on Intranet or Internet
I expand my knowledge by visiting conferences and other events about architecture

2.1. Architecting activities

Our theory of architecting activities is depicted in Table 1 and consists of five categories: communication, decision making, quality assessment, documentation, and knowledge acquisition. For each activity we defined several sub-activities that are most indicative, which we tested in our online survey. The remainder of this section highlights some of the related work our theory was built on.

According to Philippe Kruchten, software architects should "make design choices, validate them, and capture them in various architecture related artifacts" (Kruchten, 2008). Doing all these things involves a lot of consensus *decision making* in which architects balance between quality attributes, stakeholder concerns, and requirements, and in which they try to apply architectural styles and patterns.

A study by Clements et al. (2007) on the duty, knowledge, and skills of architects confirms the above view of the primary tasks of architects. They found that architects frequently interact with stakeholders, are involved in organization and business related issues, but primarily guide the architecting process: "We realized, from our many interactions with practicing architects, that their job is far more complex than just making technical decisions, although clearly that remains their most essential single duty" (Clements et al., 2007).

In the decision-making process, architects *communicate* and share ideas with various stakeholders and collaborate in teams to find optimal solutions (Farenhorst et al., 2007). To keep track of all knowledge being created or shared in this process, architects maintain a backlog (Hofmeister et al., 2007). In this backlog, an overview of decisions, constraints, concerns, open issues, etc. are administered by the architect. As input to this process, architects need a vast body of architectural knowledge. According to Clements et al. (2007), this body of knowledge includes basic computer science knowledge, design knowledge in terms of knowledge of technologies and platforms as well as knowledge about organi-

zational context and management. Architects share this knowledge not only by talking to colleagues in their team but also by communicating with stakeholders outside the team.

That architecting is not only about technology is also illustrated by Bredemeyer's architect competency list (Bredemeyer and Malan, 2002). These competencies are classified in the categories technology, consulting, strategy, organizational politics, and leadership. Bredemeyer's vision is supported by Eeles (2006), who acknowledges that architects do not only do technical stuff ('they are a technical leader', 'they understand the development process') but that they also need a lot of soft skills: Architects 'have knowledge of the business domain', 'are good communicators', 'make decisions', 'are aware of organizational politics', and 'need to be effective negotiators'. Soft skills are also emphasized by van der Raadt et al. (2008), who studied how different stakeholders in the Enterprise Architecture domain perceived the enterprise-architecture function. Apart from technical knowledge or skills, enterprise architecture involves to a large extent governance, communication, vision, and collaboration (van der Raadt et al., 2008).

Another main architecting activity is to *document* architectural knowledge. This includes not only storing application-generic knowledge such as architectural principles, patterns, or styles, but also codifying application-specific knowledge such as design decisions and their rationale. Architectural knowledge may be stored in architectural descriptions or databases, modeled in reference architectures, or grouped in specific repositories.

During the 2006 and 2007 workshops on sharing and reusing architectural knowledge (SHARK), an explicit distinction was made between architecting as a product and as a process (Avgeriou et al., 2007; Lago and Avgeriou, 2006). During these sessions, a classification scheme for architecting as a process was constructed. This scheme adds two important architecting activities to the decision making, documenting, and communication activities: learning and assessing. With respect to learning, Clements et al. (2007) listed university or industrial courses as well as certification programs as possible sources of training and education for architects. Apart from these specific learning activities, architects in their everyday work are also constantly busy with *knowledge acquisition* activities to update their knowledge. By having discussions with colleagues or customers, reading fora on the internet, or by visiting seminars, workshops or conferences, they stay up-to-date and become acquainted with new trends, developments, or best practices.

The last main architecting activity we wish to define is *assessing the quality* of architectures. Assessing and reviewing architecture descriptions or related deliverables allow architects to control quality. They are ultimately responsible for the quality of the software systems. The design and evolution of an architecture allows them to achieve this goal. As part of the quality control, architects also declare certain rules and regulations for the projects they work on. Stakeholders need to adhere to these rules and architects may reject rule violations by overruling specific decisions made.

We collected the main architecting activities in Table 1: for each of the five activities we defined several sub activities, each of which was used as an item in our survey. Table 1 shows the activities and the items that should be most indicative for these activities, which were tested by our online survey.

2.2. Support methods

We also wanted to know the kind of support architects require with respect to sharing architectural knowledge. Effective support for sharing architectural knowledge is crucial, not only to assist individual architects, but also to improve the quality of architectural designs by leveraging all knowledge assets available in the organization. Table 2 shows our theory of support methods for sharing architectural knowledge (i.e. decision management, search efficiency, community building, intelligent advice, and knowledge management). The support methods listed in Table 2 served as indicative items on our survey.

Table 2
Support methods.

S1: Decision management
An overview of the most important architectural decisions
An overview of the relations between taken decisions
Templates for codification of architectural decisions
Insight into conflicts between architectural decisions
An overview of changes through time of certain decisions
A repository to store architectural decisions
S2: Search efficiency
Search methods for existing architectural guidelines
Retrieving all documentation related to a specific architectural subject
Search facilities for decisions within a specific project
Retrieving relevant information within a project
Overview of important events related to architecture
S3: Community building
A central system to hold discussions with stakeholders
Notify colleagues about relevant documentation
A central environment to collaborate with colleagues on arch, issues
Retrieve information about the expertise of colleagues
Acquire insight into which architectural projects colleagues worked on
S4: Intelligent advice
Concrete feedback during the writing process of arch, documentation
Specific advices on which architectural decisions need to be taken
Suggestions on how to use architectural guidelines in a specific project
Being notified about the availability of new architectural publications
Being sent an overview of the status of an architectural project
S5: Knowledge management
Automatic retrieval of architectural guidelines within projects
Simple methods to annotate architectural knowledge concepts in documents with meta-data
A automatically generated overview of open design issues in documents
A system that tracks overlap among codified architectural guidelines
Central maintenance of architectural guidelines and best practices

A large part of the architectural knowledge produced is related to the design decisions, so that explicit support for the *management of these decisions* is germane. Recently, various researchers have proposed tools that help managing concepts such as design decisions, rationale, and related architectural knowledge (Ali Babar and Gorton, 2007; Capilla et al., 2007; Jansen et al., 2007). These tools include specialized templates, overviews and storage facilities for architectural knowledge concepts. Such support for managing design decisions could help architects in decision making. While working on the 'backlog' of open issues and challenges they could organize their thoughts and structure the architectural knowledge they produced (Hofmeister et al., 2007).

In addition to assisting architects in producing architectural knowledge, support during the consumption of such knowledge is equally important. The need for a more balanced view on architectural knowledge sharing, in which support for both producing and consuming architectural knowledge is included, has been discussed before (Lago and Avgeriou, 2006). In a recent study it became clear that one of the main requirements that architects have is support in retrieving the right architectural knowledge at any time (Farenhorst et al., 2008). Support for *efficient search* of architectural knowledge will stimulate reuse (reusable assets are better accessible) and learning among architects (they can find what they need more easily).

Due to the size and complexity of most software systems, it is often infeasible for one architect to be responsible for everything alone. Consequently, the architect-role is often fulfilled by multiple collaborating architects. Liang et al. (2009) provide an introduction to how a collaborative architecting process would look like. To foster sharing of architectural knowledge between all these architects calls for support methods that focus on *community building*. Architects often do not know what colleagues of other teams work on, what their experience or expertise is, or what their interests are.

We argue that support for community building improves this situation and increases the amount of architectural knowledge shared. With the advent of 'Web 2.0', this is much easier to implement than in the past. The need for systems that enable such support is explicated in Farenhorst et al. (2007). Likewise, systems such as Wikis can be employed to improve networking by letting stakeholders share ideas, discuss progress, or collaboratively produce architectural knowledge in a variety of formats (Farenhorst and van Vliet, 2008).

Another main type of support in the architecting process relates to *intelligent advice* and support. Architects could benefit from intelligent support during almost all their core activities: (a) during production activities such as writing an architectural description, (b) directly after producing architectural knowledge (in terms of feedback on what they did), and (c) during reviewing and evaluation activities. Intelligent or pro-active support helps architects to leverage and share the architectural knowledge assets available by taking over certain tasks. This might result in a software assistant who thinks together with the practitioners and suggests ideas, challenges decisions, etc. Implementations of pro-active and intelligent assistants are not often seen in practice yet. In a research setting, examples start to emerge, such as provided by Garlan and Schmerl (2007) who designed a personal cognitive assistant called RADAR, which could help architects to accomplish their high-level goals, coordinating the use of multiple applications, automatically handling routine tasks, and, most importantly, adapting to the individual needs of a user over time. Another example is the Knowledge Architect tool suite that, among other things, assists architects in writing documentation and that can perform quantitative architectural analysis (Liang et al., 2009).

The last type of support that we envision is advanced *management of architectural knowledge* that includes refinement, enrichment, and smart overviews of such knowledge. Architects could benefit from (semi)-automatic interpretation of stored knowledge to enrich it. Text mining services (e.g., Fan et al., 2006) could be employed to automatically sift through existing architectural knowledge stored (e.g., in a database) looking for new patterns, define best practices, or locate trends. Based on the findings, additional meta-data could be generated by such a service and presented to the architect. Consequently, searches executed could deliver more in-depth results that can then be shared, because part of the interpretation of the 'raw' architectural knowledge has already been done.

2.3. Architectural knowledge sharing patterns

Tables 1 and 2 show the factors in our theory on architecting activities and support for architectural knowledge sharing. We hypothesized that certain support might be more appreciated during specific architecting activities. This means that certain 'patterns' exist that describe which type of support fits which type of architecting activity best.

We hypothesized that six patterns exist between activities and support, as depicted in Fig. 1. For each pattern, we formulated a hypothesis that was tested by our survey.

P1 Architects involved in architectural decision making (A2) benefit particularly from support for management of architectural decisions (S1). We posit that practitioners who are often involved in decision-making could benefit greatly from explicit support for working on a decision 'backlog', by having overviews of open issues, conflicting decisions, traces between requirements and solutions, etc.

P2 Architects involved in taking architectural decisions (A2) benefit particularly from support for intelligent advice (S4). We state that during the thought processes and trade-offs inherent to decision making, pro-active or just-in-time support might help architects greatly. This could include (automated) advice on which decisions to take in a specific situation, or tips about how to apply architectural patterns, styles, or tactics.

P3 Architects involved in documenting architectural knowledge (A4) benefit particularly from support for maintenance and overview of architectural knowledge (S5). Architects who are often involved in codifying knowledge in artifacts such as architectural descriptions are directly helped by templates, models, frameworks, etc. Easy access to other codification methods such as central repositories for guidelines also lower the threshold of production and facilitates the storing process.

P4 Architects involved in communication with colleagues or other stakeholders (A1) particularly benefit from support for community building (S3). We suspect that architects who are frequently involved in communication with colleagues or other stakeholders are more enthusiastic about community building, sharing knowledge, and networking, while using social media (e.g., wikis, blogs, twitter, people directories, yellow pages).

P5 Architects involved in quality assessments (A3) particularly benefit from support for efficient retrieval of (or searching for) architectural knowledge (S2). During reviews and assessments, quickly retrieving information about the status of a project, the set of principles applied, or the key architectural decisions taken, will greatly help architects. This also would lead to higher-quality results during the evaluation, because more accurate and complete architectural knowledge would be available.

P6 Architects involved in expanding their knowledge on architecture (A5) benefit particularly from support for the efficient retrieval of architectural knowledge (S2), making the learning process more quickly and accurately. Just-in-time architectural knowledge allows architects to quickly scan new domain knowledge, get an update on the status of a project, or read through the main project deliverables. Put differently, good support for knowledge retrieval helps practitioners to enrich and refine the required knowledge themselves, while combining inputs from different sources. On the whole, this probably improves knowledge internalization drastically.

3. Research methodology

Our survey consisted of a pilot study, followed by the main study. The pilot study provided a pre-test of the survey items as well as a first test of our theoretical framework. The main study was split into three parts: we surveyed the practitioners' architecting activities, support methods, and their prioritization of those methods.

In total 279 Dutch architects from four IT organizations were included into our study. Three organizations were international software development and consultancy firms in the public, finance, and industry domains. The fourth was a central Dutch governmental organization for the governance of ICT architectures in the public domain.

Our research methodology included all important elements required to properly design, administer, and analyze a survey (Kitchenham and Pfleeger, 2001–2002). For a complete discussion of our methodological approach, we refer to the technical report underlying this paper (Farenhorst et al., 2009b).

3.1. Pilot study

The theoretical framework presented in Section 2 is not something we built up overnight. An earlier theory, consisting of six categories of architecting activities and six different support methods, was presented in (Farenhorst and van Vliet, 2009). The main

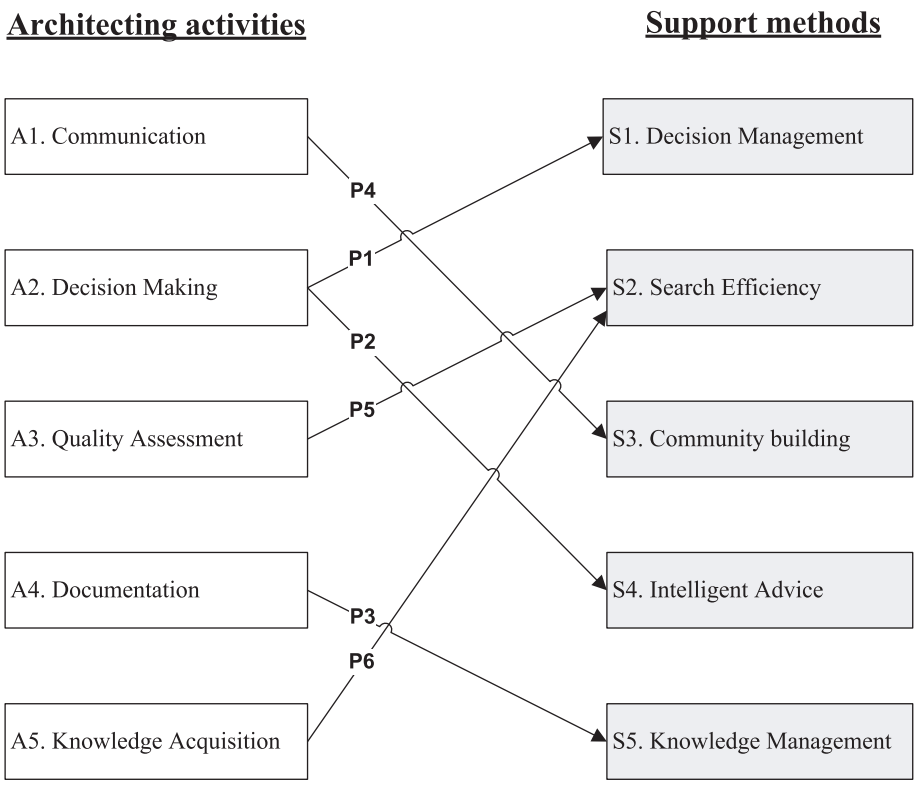

Fig. 1. Hypothesized patterns of architectural knowledge sharing support.

goal of the pilot study was to see whether this earlier theory was sufficiently strong for usage in the main study, and to improve it – or refactor – where necessary. To this end, we made constructs for all activities and support methods. Inspired by relevant literature and prior case studies (see Farenhorst and van Vliet, 2009, for more details), we formulated 10–15 statements (ie. 'items') for each of these constructs. After iteratively scrutinizing these items on clarity and redundancy, some were removed, leaving each construct between 5 and 10 items.

Because 10 items per construct would make the questionnaire of the main study too long, we decided to let a small percentage of the total sample ($N = 8$) pre-test all items in a pilot study. To test the internal consistency of the items, we performed reliability analyses on the item scores of the pilot questionnaire and selected the ones with the strongest internal consistency (ie. highest Cronbach's alpha).

To make sure that items did not indicate other activities or support methods than intended, we performed a factor analysis. This way, we could sharpen the definition of our constructs. For example, our earlier theory made a distinction between communication within a team (with colleagues) and communication outside the team (with other stakeholders). However, factor analysis of the pilot results indicated that this distinction was not made by the architects: The items of both constructs all mapped onto the same factor. Consequently, we merged the two item lists and labeled the new construct 'Communication'. This is why our final framework of architecting activities (Table 1) contains five categories and not six. In a similar fashion, we had to change our categorization of support methods by downscaling from six to five constructs: The items originally categorized as storage of architectural knowledge contaminated with decision management and knowledge management so that we collapsed the storage items over these two constructs. For each of the constructs, the five or six most indicative items were selected, as depicted in Table 2.

In addition, feedback from the architects in the pilot study was used to further optimize the survey. Examples are the layout of the questions and screens of the Web tool and the phrasing of the statements. We also improved the introductory text and the text of the invitation email.

3.2. Main study

The main study tested our theoretical framework as presented in Section 2. To properly answer what architects do and what support they need, the questionnaire of the main study consisted of three parts:

Part 1: Architecting activities. Each of the five architecting activities was treated as a separate scale, consisting of the (four to six) items as depicted in Table 1. Each item described a sub-activity (e.g., '*I inform colleagues about the results of my work*') and respondents replied to what extent that activity related to their daily work by scoring a 6-point Likert-type rating scale (1 = totally disagree, 6 = totally agree).

Part 2: Support methods. Each of the five support methods for sharing architectural knowledge was treated as a separate scale, consisting of the (five to six) items representing sub-methods, as depicted in Table 2. Respondents indicated on 6-point Likert-type rating scales to what degree these support methods were helpful in their daily work. A sample item was '*In my daily work, I benefit from an overview of the most important architectural decisions*'.

Part 3: Prioritization of support methods. The respondents prioritized the five types of support methods in order of importance for *one particular* activity alone. They ranked the five methods between 1 and 5, where a score of '1' corresponded to the highest ranked method and '5' corresponded to the lowest ranked method for that architecting activity. We made sure that no score could be given twice and that no missing values were allowed.

Note for the prioritization study (Part 3), that if we had opted for the survey approach most common in empirical software studies,

we would have presented all the architecting activities and support methods to the respondents up front and would have asked them to prioritize the methods for each activity without knowing their consensus on the definition of our terms. Our research design was more advanced in that we used Parts 1 and 2 to validate our ideas on activities and support methods. Only then, we reasoned, could we properly interpret the results of Part 3. For example, Part 1 showed that architects did not distinguish storage of architectural knowledge as a separate factor. Prioritization of this support method, then, would have been irrelevant even if it would have received systematically high or low rank-order numbers.

In addition, we split up the sample into 5 equally large homogeneous groups of architects. For these groups, Parts 1 and 2 of the questionnaire were identical. Part 3, however, contained but *one* prioritization task so to prevent bias and fatigue effects of repeating the same questions in five different settings (Hoorn et al., 2006). Each respondent thus prioritized the support methods for just one architecting activity. In Part 1 up to 3, all items were presented in random order, ensuring that each respondent received a different order of items so to keep from item-order or so-called halo effects. Further details about our approach can be found in Farenhorst et al. (2009b).

The questionnaire also contained sociodemographic items such as the respondent's job title (type of architect) and experience (number of work years). This allowed us to analyze the impact of these additional factors on the architecting activities respondents were involved in (Part 1), the support methods they appreciated (Part 2), and their prioritization of those support methods (Part 3).

To analyze the survey data, we used a range of analysis techniques, as discussed in-depth in Farenhorst et al. (2009b). The main procedure for analyzing our theory of architecting activities and support methods was *GLM-repeated measures*, which allowed us to not only evaluate the interaction effects between activities and support methods, but also the effect of covariates such as the number of work years. Multivariate tests (Pillai's trace) were run to assess the overall variance of effects on (combinations of) the dependent variables. To assess whether individual effects were significant, we used paired-samples *t*-tests. To further scrutinize significant effects of covariates such as number of work years, we looked into Pearson correlations. Finally, linear regression analyses were conducted to validate the hypothesized patterns for architectural knowledge sharing support.

4. Survey analysis

We conducted the main study in February 2009 at four IT organizations in parallel. In total, 271 architects were invited of whom 156 responded. From this group, 13 respondents were discarded in the analysis because their records were completely empty. One respondent was removed from the analysis because s/he produced scores of '1' alone. This left us with 142 respondents for the scale and factor analyses, which accounted for a response rate of 52.4%. Remaining missings were treated as subject-missing.

From the 142 respondents, 48 labeled themselves as Enterprise architects, 63 as IT or software architect, and the remainder (31) consisted of information analysts, infrastructure architects, and application designers. Their average architecture-related working experience was 6 years, 26 architects indicated less than 3 years of architecting experience, 89 architects 5 years or more. To assess whether our data were internally consistent and the items were indicative for the constructs in our framework, we conducted scale analyses. We assessed the psychometric quality of the 5 activity scales (communication, decision making, quality assessment, documentation, knowledge acquisition) and the 5 Support scales (decision management, search efficiency, community building, intelligent advice, knowledge management). Scales had between 4 and 6 items each. We tested whether items correlated sufficiently with their own scale by means of corrected item-total correlations and regular Cronbach's alpha (indicating reliability).

Table 3
Descriptive statistics for activities and support methods.

	Mean	Std. deviation	Correlation with	
			Work years	N
Work years (experience in architecture projects)	6 22	4 110		142
Architecting activities				
Communication	4.6155	.67423	.20*	132
Decision making	5.2087	.65285	.18*	131
Quality assessment	4.4484	.82204	.37**	128
Documentation	3.9847	.89052	.23**	131
Knowledge acquisition	4.6152	.74135	.15	136
Support methods				
Decision management	4.8374	.58188	.15	123
Search efficiency	4.8943	.60902	.09	123
Community building	4.5325	.74586	.10	123
Intelligent advice	4.5230	.64030	.10	123
Knowledge management	4.3033	.90077	.15	122

* $p < .05$.
** $p < .01$.

Item selection was a trade-off among several criteria. We wanted to establish as many items on a scale as possible with a minimum of 2, provided that Cronbach's alpha for a scale was ≥60 and corrected item-total correlations ≥20. In addition, the degree to which items did not correlate with other scales was tested with factor analysis, one time for the activities scales and one time for the support scales (PCA, 25 iterations Varimax rotation with Kaiser normalization, number of factors to be extracted: 5). The rotated component matrix showed that in 6 iterations, all five factors could be retrieved in the data, except for a limited number of items (Farenhorst et al., 2009b). We removed 8 items that correlated more strongly with another scale than with their own scale. Three more items were removed that correlated above .40 with other scales irrespective of high correlations with their own scale.

Thus the revised scales were submitted to reliability analyses again. Scale length was reduced to 3, 4, or 5 items. Regular Cronbach's alphas were between .72 and .90, which is reasonable to good. The scale knowledge acquisition had an alpha of .61, which is suspect but still acceptable.

4.1. What architects do

To find out what practicing architects actually do, we studied the data of Part 1 of our questionnaire. Across participants, we calculated the grand mean agreement scores to all items in a shortened scale. We ran a GLM-repeated measures for the 5-leveled within-subjects activities factor (communication, decision making, quality assessment, documentation, knowledge acquisition) on the grand mean agreement-scores with number of work years as covariate. According to the multivariate tests, the main effect of activities was significant with a considerable effect size (Pillai's Trace = .51, $F_{(4,121)} = 31.4$, $p = .000$, $\eta_p^2 = .51$).

As depicted in Table 3, the activity decision making received the highest agreement scores ($\mu = 5.21$, $\sigma = .65$) and documentation the lowest ($\mu = 3.98$, $\sigma = .89$). Analysis of variance showed that these results were not significantly different between the four participating organizations, nor did the type of architect impact this outcome. We did, however, find a significant correlation between the amount of work experience architects had and the activities they were involved in. Particularly, agreement to quality assessment and documentation increased significantly with the number of work years (see Farenhorst et al., 2009b). Paired-samples *t*-tests

showed that all differences among the activities were significant ($t > 2.7$, $p < .009$), except for the contrast between communication and knowledge acquisition ($t_{125} = -.38$, $p >> .05$).

Our survey results thus indicated that interestingly enough architects spent most of their time on making architectural decisions and least on documenting the results. This attitude creates a substantial risk of architectural knowledge vaporization (Bosch, 2004) and decreases the chances for effective reuse of architectural knowledge.

The effect of work years indicated that more experienced architects were more engaged in auditing activities and maybe related to this, spent more time on documenting their results. Over the years, the advantages of retrieving or reusing codified knowledge might have proven useful to the more experienced architects.

4.2. What architects need

To examine what kind of support methods architects desire, we studied the data of Part 2 of our questionnaire. Again, we ran a GLM-repeated measures for the 5-leveled within-subjects support method factor (decision management, search efficiency, community building, intelligent advice, and knowledge management) on the grand mean agreement-scores with work years as covariate. The main effect of support methods was significant with medium to low effect size (Pillai's Trace = .21, $F_{(4,117)} = 7.7$, $p = .000$, $\eta_p^2 = .21$).

As depicted in Table 3, search efficiency raised the highest agreement scores ($\mu = 4.89$, $\sigma = .61$) and knowledge management the lowest ($\mu = 4.30$, $\sigma = .90$). Again, analysis of variance showed no significant differences between types of architects or between architects from different organizations. The interaction between support methods and work years also was not significant. Search efficiency and decision management were equally agreed upon as the kind of support most needed, irrespective of work years ($F < 1$).

To further explore the main effect of support method, paired-samples t-tests showed that all differences among the kinds of support were significant ($t > 2.7$, $p < .008$), except for the contrast between decision management and search efficiency ($t_{122} = -.38$, $p >> .05$) and between community building and intelligent advice ($t_{122} = .15$, $p >> .05$). This means that decision management and search efficiency were equally agreed upon as the kind of support that was most needed, irrespective of number of work years.

In overviewing these results, we find it rather contradictory that although architects do not document much, they do wish for efficient support to retrieve useful – previously stored – knowledge. It seems that architects wish to use codified knowledge but refrain from codifying it in the first place. A proper balance between producing and consuming architectural knowledge, as deemed important in Lago and Avgeriou (2006), seems to be missing.

Another surprise related to architects' opinion about the various support methods is indicated by the low score for knowledge management. Considering that architects make a lot of decisions but are not that eager to properly document them, we expected that they would be more enthusiastic about automated support for these activities. One would say that easy annotation of architectural knowledge in documents would make the codification tasks a little less cumbersome. Yet, architects do not fancy such pro-active, or automated support. Perhaps they wish to remain in the driver's seat and do not fancy the idea that a system manipulates and processes architectural knowledge independently.

4.3. Preferences in support methods

We examined which support methods architects preferred by conducting two separate analyses: (a) we tested the patterns hypothesized in Fig. 1 by trying to retrieve them in the data of Parts 1 and of our questionnaire, and (b) we examined the ranking data of Part 3.

For our first round of analyses (a), we formulated six regression models in which the level of agreement to activities should predict the level of agreement to the kind of support needed (see Fig. 1): $A1 \rightarrow S3$, $A2 \rightarrow S1$, $A2 \rightarrow S4$, $A3 \rightarrow S2$, $A4 \rightarrow S5$, $A3 \,\&\, A5 \rightarrow S2$. In using linear regression analyses (method Enter), we evaluated the degree to which an activity explained the variance in agreement to the corresponding support method:

P1 Decision making was the predictor for decision management, which served as the dependent. Decision making accounted for a significant quantity of the variability in agreement to decision management ($R^2 = .063$, $R^2_{adj} = .056$, $F_{(1,121)} = 8.2$, $p = .005$). Decision making explained 25% of the variance in decision management (standardized $\beta = .25$, $t = 2.9$, $p = .005$).

This means that architects who made decisions were particularly in favor of decision management (P1). This is in line with our assumption that backlog management and effective management of design decisions makes it easier for architects to have a proper overview of the solution space, reusable assets, potential conflicts, stakeholder demands, etc.

P2 Decision making accounted for a significant quantity of the variability in agreement to intelligent advice ($R^2 = .035$, $R^2_{adj} = .027$, $F_{(1,121)} = 4.3$, $p = .039$). Decision making explained 19% of the variance in intelligent advice (standardized $\beta = .19$, $t = 2.1$, $p = .039$).

This means (P2) that intelligent advice such as specific advice on which design decision to take, or suggestions about which architectural guidelines or styles to use, was particularly helpful for architects who were busy taking design decisions.

P3 Documentation could not significantly account for the variability in agreement to knowledge management ($R^2 = .01$, $R^2_{adj} = .004$, $F_{(1,120)} = 1.5$, $p >> .05$). Therefore, this hypothesis was rejected.

P4 Communication served as the predictor of community building but could not significantly account for the variability in agreement to community building ($R^2 = .014$, $R^2_{adj} = .006$, $F_{(1,121)} = 1.8$, $p >> .05$). Therefore, this hypothesis was rejected.

P5 Quality assessment accounted for a significant quantity of the variability in agreement to search efficiency ($R^2 = .12$, $R^2_{adj} = .11$, $F_{(1,121)} = 16.2$, $p = .000$). Quality assessment explained 34% of the variance in search efficiency (standardized $\beta = .34$, $t = 4.0$, $p = .000$).

Retrieving architectural knowledge was important for architects, particularly when they were conducting audits or other quality evaluations (P5). Apparently, to decide about the quality of an architecture, it is important to quickly retrieve the standards, rules, etc. that need to be adhered to.

P5,P6 In a multiple linear regression, quality assessment and knowledge acquisition together accounted for a significant quantity of the variability in agreement to search efficiency ($R^2 = .13$, $R^2_{adj} = .12$, $F_{(2,120)} = 9.0$, $p = .000$). We also assessed the relative importance of quality assessment and knowledge acquisition in predicting the need for search efficiency. Quality assessment was most strongly related to search efficiency (30%) (standardized $\beta = .30$, $t = 3.2$, $p = .002$). Knowledge acquisition did not significantly add to this effect (standardized $\beta = .12$, $t = 1.3$, $p >> .05$). Supporting this conclusion was the height of the standardized Beta coefficient of quality assessment and the strength of the correlation between quality assessment and search efficiency, partialling out the effects of the other predictor ($r_{partial} = .28$, $r_{part} = .27$). Knowledge acquisition offered little or no addi-

tional predictive power beyond that contributed by the quality assessment measure.

Therefore, this hypothesis was accepted for quality assessment (P5) and refuted for knowledge acquisition (P6).

Linear regression analyses confirmed the hypotheses formulated for patterns P1, P2, and P5. By contrast, advanced knowledge management support was not particularly important to architects busy with documentation (P3). Community building was not particularly interesting to architects active in communication (P4) and search efficiency was not particularly appreciated by architects busy with knowledge acquisition (P6).

Some architects are more the decision making type (called Architectus Reloadus by Fowler, 2003) and those benefit mostly from management of architectural decisions (cf. pattern P1). Other architects who focus more on community building (Architectus Aryzus) benefit more from community building support. In organizations with a primary focus on *architecting,* i.e. the development of enterprise or software architectures using methods, tools, and methodologies (TOGAF, RUP), architects spend a great deal of time on creating solutions and making design decisions. Documenting these solutions is a main task to communicate solutions to the stakeholders or to simply finish a project by producing the necessary deliverables. For architecting-style organizations, the confirmation of pattern P1 and P2 show that practitioners are served by decision-management support and efficient knowledge-retrieval.

In organizations with a primary focus on *auditing*, i.e. assessing architectures on quality criteria or on specific requirements put forward by a customer, auditors usually evaluate (a version of) an architecture that is already designed. Therefore, these people are not involved in decision making and describing solutions. Instead, quality control is one of their primary activities, rendering pattern P5 applicable. A focus on availability and retrieval of architectural knowledge is most important in this case. Most likely, a strong focus on well-codified knowledge is suitable because templates, meta-data, and other ways of structuring knowledge in databases or repositories are most effective ways to make large quantities of architectural knowledge searchable. A Google-search kind of feature is what architects may benefit most from in this type of organization.

In organizations in which *learning and collaboration* are of primary interest, colleagues are invited to learn from each other, to share knowledge, or to pro-actively build up their expertise. Such organizations have a strong focus on communication and expanding people's knowledge. This pleas for the application of pattern P6, indicating that efficient retrieval of knowledge is crucial (codification), but creating social networks and communities in which knowledge is shared would be equally useful (personalization). Albeit for different reasons but similar to the 'architecting'-type of organization, a hybrid sharing strategy would be most welcome to this type of organization. The same holds for organizations that stimulate collaboration under the pressure of market conditions or due to their business model, including outsourcing work abroad, offshoring, or distributed development.

For our second round of analyses (b), we looked into the data of Part 3 of the survey, where the respondents ranked the five support methods. We were interested in the effects of architecture activities on the priority of support options.

Respondents ranked the support options for one activity alone, so that a 5-leveled between-subjects factor of activity-ranking condition could be created (communication condition, decision-making condition, quality-assessment condition, documentation condition, knowledge-acquisition condition). We ran a GLM-repeated measures for the between-subjects factor activities-ranking condition and the 5-leveled within-subjects factor of ranked support methods (ranked decision management, ranked search efficiency, ranked community building, ranked intelligent advice, and ranked knowledge management) on the mean rank numbers with work years as covariate. However, the main effect of work years was not significant ($F < 1$) and the interaction between ranked support methods and work years also was not significant ($F < 1$). Therefore, we reran the analysis but this time excluding work years as covariate.

Analysis of variance showed no significant differences between the five conditions over which the respondents had been divided. Apparently, architects prioritized the five support methods regardless of the activity they were assigned to. This meant that we could aggregate the results of the five groups by the 'Total' rows.

A new round of multivariate tests showed that the interaction of ranked support methods by activity-ranking condition was not significant ($F < 1$) and that the main effect of activity-ranking condition on the mean rank numbers also was not significant ($F < 1$). However, the main effect of ranked support methods did reach significance with a small effect size (Pillai's Trace = .12, $F_{(4,112)} = 3.8$, $p = .006$, $\eta_p^2 = .12$). Paired-samples t-tests showed that five out of 10 comparisons among the ranked support methods were significant, which means that the need for support was a sliding scale of priorities.

Keeping in mind that a lower rank-order number indicates higher priority, decision management ($\mu = 2.63$, $\sigma = 1.37$) had higher priority than intelligent advice, which had lowest priority ($\mu = 3.33$, $\sigma = 1.42$). The 'sliding scale' of priorities is a hierarchy of support methods that are more *or* less needed for architecting activities. In interpreting these results, we grouped the support methods as follows. Note that the $\cong$ signs mean that there was no significant difference between the mean rank numbers of two support methods, according to a t-test:

- Most needed: decision management ($\mu = 2.63$) $\cong$ search efficiency ($\mu = 2.86$)
- Needed: knowledge management ($\mu = 2.91$) $\cong$ community building ($\mu = 3.28$)
- Least needed: Intelligent advice ($\mu = 3.33$)

Please bear in mind that this ordering only tells us something about the relative value architects attribute to these support methods. Some methods can still have merit depending on a certain architecting activity that is undertaken. For example, the confirmation of pattern P2 showed that intelligent advice indeed may be helpful in decision-making activities.

5. Discussion

The survey results offer plenty of food for thought. Although we suggested plausible explanations for our findings, we were quite interested in the opinion of the practitioners themselves. Therefore, we asked the respondents of our survey to provide feedback on the results. We emailed them a bullet-wise highlight report, summarizing the main results as presented in the previous section. This email triggered several responses within days, from which we could distill a number of interesting quotes. Some of these quotes are included in the discussion below.

With respect to what architects do, the most striking result is that architects seem to make lots of architectural decisions, but – especially the less experienced ones – neglect documenting those decisions and their rationale. Most obvious reasons for this is a lack of time or interest. Often the benefits of documenting design rationale are only visible in the longer term; in the short term the most important thing is to meet the deadline, and move on to the next project. The lack of interest in long-term knowledge reuse is in line

with observations from Harrison et al. (2007), who found that architects miss real motivation to document and maintain architectural knowledge.

The absence of a clearly defined, visible process for architecting activities could also partly explain why architects omit to document much. If the role, responsibilities, and extent of the authority of architects were better defined, architects would become more accountable for their actions. Experience in several software projects shows that a well-defined, visible process (e.g, by using a charter) stimulates architects to document more knowledge (Kruchten, 1999).

Another reason for not documenting is that architects already possess tacit knowledge without having a direct need for making that knowledge explicit. This phenomenon is acknowledged by Kruchten et al. (2009), who also mention that as a consequence, architects cannot revisit or communicate decisions. Bosch (2004) acknowledges this problem as well, which he called knowledge vaporization. This process is to be prevented at all costs, because if the reasoning behind an architectural solution is lost or 'forgotten', it may cause design erosion.

Two more reasons for not documenting were provided by architects who responded to our feedback request:

> "Whether or not to document is a tough decision. Doing so could make yourself redundant in case questions are asked later on, and that is what worries architects most."
> "Many architects have a technical background and were programmers, designers or system maintainers before they became an architect. For these people, documenting has always been a pain, and this is a mindset they still have."

Our results are in line with a study on the architect's mindset by Clerc et al. (2007). They conducted a survey among architects in the Netherlands and found that the prevalent mindset of architects is 'to create and communicate' rather than 'to review and maintain' an architecture. In our study, decision making and communication scored highest and quality assessment and documentation scored lowest, which further emphasizes the bias towards short-term solutions rather than maintaining a body of knowledge that is useful for the longer term as well.

When looking at the prioritization of support methods, we were not surprised to see that managing architectural design decisions was found most important. This is in line with an increasing focus on design decisions in the software-architecture research-community, as reflected in the formulation of a decision view on software-architecture practice (Kruchten et al., 2009) and the adoption of central concepts as 'decision' and 'rationale' to the upcoming ISO/IEC 42010 standard for software architecture description (ISO/IEC, 2009).

The interesting thing, however, is that our results show that experienced architects differ slightly in their mindset because they do more on quality assessments and documentation than less experienced architects. Perhaps experienced architects value these activities more because they encountered more situations in which sufficient documentation proved to be useful. Another explanation could be that writing architectural documentation is easier if you are more experienced (e.g., formulating the rationale of decisions or drawing effective architectural views). Likewise, conducting architectural audits is something that is only possible for someone with sufficient experience in the field. This makes it easier to spot inconsistencies, conflicts, or missing qualities of the system.

As for the other support methods, we were surprised that 'knowledge-management support' received the lowest scores (cf. Table 3) and that 'intelligent advice' was ranked lowest by the architects. These two categories both assist architects in their routine knowledge-processing tasks by automatically retrieving and manipulating information, offering suggestions, etc. Particularly because architects appear to have little time or interest in documenting architectural knowledge, we had expected that all pro-active support would be more appreciated, and make certain basic tasks less cumbersome.

We suspect that architects are a bit frightened by the idea that smart automated tooling would take over their roles, although they acknowledge the positive effects (pattern P2 was significant). Architects want to sit in the driver's seat and have a tight control over all processes. That also explains why support methods that do not endanger this role (e.g., search efficiency and decision management) are clearly ranked higher. Another reason for the relative low rank of intelligent support could be the lack of effective implementations of such support. We heard from several architects that most intelligent tools and methods are still rather immature and add little to their daily work, but that more versatile, mature implementations would be welcomed:

> "I find most specialized tools to have too little functionality that supports my work as a software architect. As a result, I often resort to more generic tools such as MS Visio, spreadsheets, etc."

The fact that architects want to have control over their actions is further underlined by the fact that in the prioritization of support methods, community building is ranked fourth (out of five). We had expected the architects to be a bit more 'team-players', or at least more in favor of creating a 'community of architects' in which experts effectively exchange knowledge, discuss experiences, and build up a collective memory. Our study leaves us with the impression that practicing architects are rather 'lonesome'. They are not the community builders we had expected them to be. Instead, they act in splendid isolation. Several architects confirmed this, two of whom explained it as follows:

> "I recognize myself in the 'lonesome architect' characterization and find it actually quite logical. An architect is often busy working in the early stages of the software life cycle and of projects. Colleagues are busy helping to solve problems of other customers and are thus not around. Furthermore, there is often no guidance for the work you do as an architect because the systems you help design do not exist yet."
> "An architect usually considers his own judgment as most precise and valuable. Referring to others (humans or tools) is often only a strategic move."

Coming back to our main research question, our study provides valuable insights into what architects do and what support they need for sharing architectural knowledge. Architects are individual experts who consume substantial amounts of architectural knowledge but care less about actively sharing such knowledge. We conjecture that the best way to motivate these lonesome architects in sharing architectural knowledge involves non-intrusive support that offers various mechanisms to easily store, communicate, or manipulate knowledge in the architecting process. Based on the results of our survey, we postulate that such support should have two fundamental characteristics:

(1) *It respects the architect's autonomy*. We learned that architects like to control the processes they are involved in. Methods that support architectural knowledge sharing should respect this. Automated support that 'thinks' for the architect or tools that prescribe architects what to do are not going to work. What does work, however, are more person-centric descriptive tools that assist knowledge workers 'on the fly' during daily routines (Cormican and Dooley, 2007). Although in this survey, architects ranked intelligent support as least needed, feedback obtained suggests that such support could still be

valuable if implementations would be sufficiently versatile and mature.

(2) *It stimulates both production and consumption of architectural knowledge.* We saw that support for decision management and search efficiency were most needed according to practicing architects. Search efficiency support assists the autonomous, or lonesome, architect in retrieving relevant architectural knowledge when needed, and is thus oriented towards consumption of architectural knowledge. Decision management helps architects to manage their thought processes when taking (new) design decisions, solving conflicts, or codifying design rationale. This kind of support thus focuses mostly on the creation – or production – of architectural knowledge. Based on our survey results, we argue that mature support for sharing architectural knowledge should assist architects in both producing and consuming such knowledge.

A promising development in software architecture is the emergence of integrated platforms for sharing architectural knowledge that implement a range of 'use cases' related to production and consumption of knowledge. A state-of-the-art review of such platforms is provided by Liang and Avgeriou (2009). A main characteristic of some of these platforms is that they follow a hybrid strategy for architectural knowledge sharing (Farenhorst et al., 2008). As Lago et al. (2008) argued, such a strategy "support[s] the knowledge producers in efficiently documenting the knowledge and the consumers in using it", which would resolve the imbalance of producing and consuming knowledge we observed during our study.

The identification of architectural knowledge sharing patterns and subsequent development of appropriate methods and tools does not automatically lead to increased knowledge sharing behavior among architects. Lots of other influencing or motivational factors exist that induce architects to share knowledge (Gosh, 2004), including personal experience, skills and background, but also organizational context, culture and beliefs. This is acknowledged by Schneider and Modeling (2009), who argues that "It is a major misunderstanding to expect altruistic behavior. An expert or user needs reasons to provide input". Apart from offering effective support methods, organizations can do a lot to stimulate knowledge sharing. Some obvious options include setting up organizational activities (working in small competence teams, helping each other with producing deliverables) and communication activities (organize workshops and brainstorm sessions) in order to motivate architects to share knowledge with peers. Our study shows that real intrinsic motivation, however, probably involves all-round support that allows architects to stay in control of all their architectural knowledge assets.

5.1. Threats to validity

In this section, we list possible limitations to our study by discussing the internal validity, construct validity, and external validity following Kitchenham et al. (2002) and Perry and Porter (2000). Internal validity relates to the extent to which the design and analysis may have been compromised by the existence of confounding variables and other unexpected sources of bias. Construct validity means that the independent and dependent variables accurately modeled the abstract hypotheses. External validity relates to the extent to which the hypotheses captured the objectives of the research and the extent to which any conclusions can be generalized. In other words, external or ecological validity means that the study's results generalize to settings outside those of the current study.

We were happy with the rather high response rate of our survey (52%). Our questionnaire design and pilot study ensured that our items were unambiguous, to-the-point, and our hypotheses well-focused on the research question we wished to address. This made analysis of the survey data relatively straightforward and increased the chances for meaningful and significant results. With respect to *internal validity*, our survey design and procedures (cf. Farenhorst et al., 2009b) left little room for selection bias or confounding variables. For instance, participants were not able to go back to previous screens to change previous scores, test fatigue was prevented by splitting up the ranking items into different between-subject conditions, random presentation of items made sure that possible halo effects were smeared out over all items, and all items were phrased as concisely as possible and were controlled for in the pilot test.

To conform to *construct validity* we constructed our theory such (cf. Fig. 1) that it formed the basis for our survey items. During both the pilot study and the main study, we conducted scale analyses and factor analyses to assess whether our theorized concepts could be unambiguously retrieved in the data. In the pilot study, this helped us to refactor our original framework (as presented in Farenhorst and van Vliet, 2009) into the framework presented in this paper. Scale and factor analysis in the main study reconfirmed that indeed five factors of architecting activities and support methods should be distinguished, which indicates good construct validity.

With respect to *external validity*, we deem our results fairly generic. Our sample was relatively large and the fact that four different organizations participated, strongly decreased the chance for organizational bias. To obtain a fair reflection of a typical architecting process, multiple types of architects were included into our study such as software, IT, solution, enterprise, and infrastructure architects. However, even though the participating companies were international, our study only focused on practicing architects who worked in the Netherlands. Our study did not deal with possible cultural or educational factors in which Dutch architects may differ from their colleagues abroad.

6. Conclusions

Architects are complicated individuals. They make lots of architectural decisions but neglect to document them. Producing and subsequently sharing architectural knowledge is clearly not one of their most popular activities. For this reason, one would expect that supporting architects in this latter task is something they would appreciate but the opposite was true. Architects rather stay in control themselves and still need to be convinced of the value of current automated or intelligent support. When it comes to consumption of architectural knowledge, however, architects indicated that support for effective retrieval of (stored) architectural knowledge is on the top of their wish list. This imbalance in production and consumption shows that with respect to sharing architectural knowledge, architects are not the 'community builders' many expect them to be. Instead, they are rather lonesome decision makers who act in splendid isolation.

Our survey research helped unraveling what architects really do and what they need with respect to sharing architectural knowledge. The results act as a call for awareness to both researchers and practitioners. We found that effective support for sharing architectural knowledge should acknowledge 'the nature of the beast'. We therefore plead for descriptive, non-intrusive support that respects the architect's autonomy. Moreover, to be effective, this support should stimulate architects in both producing and consuming architectural knowledge. Integrated, 'all-round' tool environments that implement a variety of architectural knowledge-sharing use-cases seem promising because they could provide a solid balance between production and consumption of knowledge in the architecting process.

To round off, before implementing a range of support methods for architectural knowledge, it is advisable to scan the status

quo of an organization. Perhaps, our findings could become part of an assessment instrument such as the SEI is working on to gauge organizational architectural competence (Bass et al., 2008).

Acknowledgments

This research was 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. We thank Philippe Kruchten for providing valuable suggestions and comments.

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, at doi:10.1016/j.jss.2010.11.909.

References

Ali Babar, M., Gorton, I., 2007. 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.

Avgeriou, P., Kruchten, P., Lago, P., Grisham, P., Perry, D., 2007. Architectural knowledge rationale—issues, trends challenges. ACM SIGSOFT Software Engineering Notes 32 (4), 41–46.

Bass, L., Clements, P., Kazman, R., Klein, M., 2008. Evaluating the software architecture competence of organizations. In: 7th Working IEEE/IFIP Conference on Software Architecture, Vancouver, Canada.

Bosch, J., 2004. Software architecture: the next step. In: 1st European Workshop on Software Architectures (EWSA), LNCS vol. 3074. Springer, pp. 194–199.

Bredemeyer, V, Malan, R., 2002. The Role of the Software Architect. Technical Report White paper 12/10/04, Bredemeyer Consulting.

Capilla, R., Nava, F., Pérez, S., Dueñnas, J.C., 2006. A web-based tool for managing architectural design decisions. In: 1st ACM Workshop on SHaring ARchitectural Knowledge (SHARK), Torino, Italy.

Capilla, R., Nava1, F., Dueñas, J.C., Modeling, 2007. Documenting the evolution of architectural design decisions. In: 2nd Workshop on SHAring and Reusing architectural Knowledge—Architecture, rationale, and Design Intent (SHARK/ADI), Minneapolis, USA.

Clements, P., Kazman, R., Klein, M., Devesh, D., Reddy, S., Verma, P., 2007. The duties skills, and knowledge of software architects. In: 6th Working IEEE/IFIP Conference on Software Architecture (WICSA).

Clerc, V., Lago, P., van Vliet, H., 2007. The architect's mindset. In: 3rd International Conference on the Quality of Software-Architectures (QoSA), Boston, USA, pp. 231–249.

Cormican, K., Dooley, L., 2007. Knowledge sharing in a collaborative networked environment. Journal of Information & Knowledge Management 6 (2), 105–114.

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

Fan, W., Wallace, L., Rich, S., Zhang, Z., 2006. Tapping the power of text mining. Communications of the ACM 49 (9), 77–82.

Farenhorst, R., de Boer, R.C., 2009. Knowledge management in software architecture: state of the art. In: Ali Babar, M., Dingsøyr, T., Lago, P., van Vliet, H. (Eds.), Software Architecture Knowledge Management: Theory and Practice. Springer.

Farenhorst, R., Hoorn, J.F., Lago, P., van Vliet, H., 2009a. The lonesome architect. In: Kazman, R., Oquendo, F., Poort, E., Stafford, J. (Eds.), Proceedings 2009 Joint Working IEEE/IFP Conference on Software Architecture & European Conference on Software Architecture (WICSA/ECSA 2009, IEEE Computer Society, pp. 61–70.

Farenhorst, R., Hoorn, J.F., Lago, P., van Vliet, H., 2009b. What architects do and what they need to share knowledge. In: Technical Report IR-IMSE-003, VU University, Amsterdam, April.

Farenhorst, R., Izaks, R., Lago, P., van Vliet, H., 2008. A just-in-time architectural knowledge sharing portal. In: 7th Working IEEE/IFIP Conference on Software Architecture (WICSA), pp. 125–134.

Farenhorst, R., Lago, P., van Vliet, H., 2007. EAGLE: effective tool support for sharing architectural knowledge. International Journal of Cooperative Information Systems 16 (3-4), 413–437.

Farenhorst, R., van Vliet, H., 2008. Experiences with a Wiki to support architectural knowledge sharing. In: 3rd Workshop on Wikis for Software Engineering (Wikis4SE), Porto, Portugal.

Farenhorst, R., van Vliet, H., 2009. Understanding how to support architects in sharing knowledge. In: 4th Workshop on SHAring and Reusing architectural Knowledge (SHARK), Vancouver, Canada.

Fowler, M., 2003. Who needs an architect? IEEE Software, 2–4.

Garlan, D., Schmerl, B., 2007. The RADAR architecture for personal cognitive assistance. International Journal of Software Engineering and Knowledge Engineering 17 (2).

Gosh, T., 2004. Technical report, MIT Open Courseware, Sloan School of Management, Cambridge, MA. Creating incentives for knowledge sharing.

Harrison, N., Avgeriou, P., Zdun, U., 2007. Using patterns to capture architectural decisions. IEEE Software 24 (4), 38–45.

Hofmeister, C., Kruchten, P., Nord, R., Obbink, H., Ran, A., America, P., 2007. A general model of software architecture design derived from five industrial approaches. The Journal of Systems and Software 80 (1), 106–126.

Hoorn, J.F., Breuker, M.E., Kok, E., 2006. Shifts in foci and priorities different relevance of requirements to changing goals yields conflicting prioritizations and is viewpoint-dependent. Software Process Improvement and Practice 11, 465–485.

Huysman, M., de Wit, D., 2004. Practices of managing knowledge sharing: towards a second wave of knowledge management. Knowledge and Process Management 11 (2), 81–92.

ISO/IEC, 2009. Systems and software engineering—architecture description. Standard ISO/IEC 42010.

Jansen, A., Bosch, J., 2005. Software architecture as a set of architectural design decisions. In: 5th Working IEEE/IFIP Conference on Software Architecture (WICSA), Pittsburgh, USA, pp. 109–120.

Jansen, A., van der Ven, J.S., Avgeriou, P., Hammer, D.K., 2007. Tool support for architectural decisions. In: 6th Working IEEE/IFIP Conference on Software Architecture.

Kitchenham, B., Pfleeger, S., 2001–2002. Principles of Survey Research, Parts 1 to 6. Software Engineering Notes.

Kitchenham, B.A., Pfleeger, S.L., Hoaglin, D.C., Rosenberg, J., 2002. Preliminary guidelines for empirical research in software engineering. IEEE transactions on Software Engineering 28 (8).

Kruchten, P., 1999. The Architects—the software architecture team. In: 1st Working IFIP Conference on Software Architecture (WICSA), San Antonio, USA.

Kruchten, P., 2008. What do software architects really do? The Journal of Systems and Software 81, 2413–2416.

Kruchten, P., Capilla, R., Dueñnas, J.C., 2009. The Decision view's role in software architecture practice. IEEE Software 26 (2), 36–42.

Kruchten, P., Lago, P., van Vliet, H., 2006. Building up and reasoning about architectural knowledge. In: 2nd International Conference on the Quality of Software Architectures (QoSA), vol. 4214. Springer LNCS, pp. 39–47.

Lago, P., Avgeriou, P., 2006. 1st Workshop on SHaring and Reusing ARchitectural Knowledge. ACM SIGSOFT Software Engineering Notes 31 (5), 32–36.

Lago, P., Avgeriou, P., Capilla, R., Kruchten, P., 2008. Wishes and boundaries for a software architecture knowledge community. In: 7th Working IEEE/IFIP Conference on Software Architecture (WICSA), pp. 271–274.

Liang, P., Avgeriou, P., 2009. Tools and technologies for architectural knowledge management. In: Ali Babar, M., Dingsøyr, T., Lago, P, Vliet, H.v. (Eds.), Software Architecture Knowledge Management: Theory and Practice. Springer.

Liang, P., Jansen, A., Avgeriou, P., 2009. Collaborative software architecting through knowledge sharing. In: Mistrik, I. (Ed.), Collaborative Software Engineering. Springer.

Perry, D., Porter, A.A., Votta, L.V., 2000. Empirical studies of software engineering: a roadmap. In: Conference on The Future of Software Engineering, Limerick, Ireland, pp. 345–355.

Schneider, K., Modeling, 2009. Improving information flows in the development of large business applications. In: Ali Babar, M., Dingsøyr, T., Lago, P., Vliet, H.v. (Eds.), Software Architecture Knowledge Management: Theory and Practice. Springer.

Tyree, J., Akerman, A., 2005. Architecture decisions: demystifying architecture. IEEE Software 22 (2), 19–27.

van der Raadt, B., Schouten, S., van Vliet, H., 2008. Stakeholder perception of enterprise architecture. In: 2nd European Conference on Software Architecture (ECSA), LNCS vol. 5292. Springer, Paphos, Cyprus, pp. 19–34.

Dr. Johan F. Hoorn (D. Litt., D. Sc.) In being the managing director of the Center for Advanced Media Research Amsterdam (CAMeRA) at the VU University Amsterdam, senior associate professor Johan F. Hoorn works in an intensively multi-disciplinary and international research environment. Johan holds two PhD degrees, one in literature and one in computer science. Most of his work is cognitively oriented with a focus on creative technologies such as virtual characters, emotionally intelligent agents, and AI robots. Published papers range from author identification, virtual reality, genre rules, and metaphor to user studies, requirements engineering, and software architecture. He has a US patent pending regarding a user tracking system that utilizes Wii technology. Johan is the leading coordinator of the VU University's activities in the creative industries and one of the founders, executive board member, as well as director of Research and Education of THNK, the Amsterdam top school for creative leadership and entrepreneurship. Johan was a Burgen scholar of the Academia Europaea and recently, he was granted a Fellowship by the Lorentz Center of the Royal Dutch Academy of Arts and Sciences (KNAW) to develop a unified theory of creativity at the Netherlands Institute for Advanced Study in the Humanities and Social Sciences (NIAS).

Rik Farenhorst is a consultant and trainer in enterprise/IT architecture at DNV-CIBIT. He is involved with teaching several courses on enterprise/IT architecture and knowledge management, and helping various clients with issues and challenges

related to those topics. In 2009 Rik obtained his PhD degree at VU University Amsterdam on the topic of 'architectural knowledge management'. He is author of more than 20 scientific and popular publications on this topic.

Patricia Lago is associate professor at the VU University Amsterdam. Her research interests are in software- and service oriented architecture, architectural knowledge management, green IT and sustainable software engineering. Lago has a PhD in Control and Computer Engineering from Politecnico di Torino. She is Area Editor Service Orientation for the Elsevier Journal of Systems and Software. She is member of IEEE and ACM, and of the International Federation for Information Processing Working Group 2.10 on Software architecture.

Hans van Vliet is Professor in Software Engineering at the VU University Amsterdam, the Netherlands, since 1986. He got his PhD from the University of Amsterdam. His research interests include software architecture and empirical software engineering. Before joining the VU University, he worked as a researcher at the Centrum voor Wiskunde en Informatica (CWI, Amsterdam). He spent a year as a visiting researcher at the IBM Almaden Research Center in San Jose, California. He co-authored over 140 refereed articles. He is the author of "Software Engineering: Principles and Practice", Wiley (3rd Edition, 2008). He is a member of IFIP Working Group 2.10 on Software Architecture, and the Editor in Chief of the Journal of Systems and Software.