# ARCHITECTURAL KNOWLEDGE SHARING (AKS) APPROACHES: A SURVEY RESEARCH

**[1]Reza Meimandi Parizi, [2]Abdul Azim Abdul Ghani**

[1]Department of Information System, UPM, Serdang, Malaysia-43400

[2]Assoc. Prof., Department of Information System, UPM, Serdang, Malaysia-43400

E-mail: parizi@fsktm.upm.edu.my , azim@fsktm.upm.edu.my

## ABSTRACT

Research on software architecture from different perspectives has been done for several years. However, Architectural Knowledge (AK), a relatively new field, has gained its increasing interest among the community. On this regard, various topics devoted to architectural knowledge, such as reusing, sharing, managing, and communicating are being studied. Among them, AK sharing brings new effective challenges and issues not present when studying other topics in architectural knowledge. Therefore, this paper surveys the current researches on AK sharing (pertaining to software architecture), the approaches, models that are being proposed, and issues that arise when sharing different AKs by different parties. By making survey on AK sharing approaches, a better understanding of these approaches and issues related in this area is provided so that it can be a penetrative resource for a fast training and educating in this area of SE. In addition, conclusion about current state of research in this area and future trends for AK sharing is identified.

**Keywords:** Architectural Knowledge Sharing (AKS), Software Architecture, Knowledge Sharing, Architectural Knowledge (AK)

## 1. INTRODUCTION

Architectural knowledge captures precious knowledge that is worth sharing. The models, tools designed for this purpose, sometimes, fail to facilitate such knowledge exchange. Moreover, sharing knowledge, pertaining to software architecture, is not easy to achieve, in particular in distributed and global projects. Some companies that participate in virtual communities like inner or open-source communities are starting to realize the challenges of sharing the architectural knowledge between the communities [2]. The reason for this matter can be viewed *(1)* the people involved in the architecting process (who own the knowledge) often do not document it. For the most part, existing notational and documentation approaches to software architecture typically focus on the components and connectors and fail to document the design decisions that resulted in the architecture as well as the organizational, process and business rationale underlying the design decisions. This results in high maintenance cost, high degrees of design erosion and lack of information and documentation of relevant architectural knowledge [4]. *(2)* Or even when that architectural knowledge is documented, it is often not sufficiently shared within the organization: the knowledge is not disseminated to the appropriate stakeholders; the recipients of knowledge do not use it in their own tasks either intentionally or because there is no provision in the processes [3].

Due to the aforementioned issues, AK sharing brings interesting questions such as, what is the relevant architectural knowledge we want (and can) share? [2] Which can be further refined into: Is it possible to define a shared body of knowledge about software architecture? Is it possible to standardize the meta-models for architecture knowledge through a generic core meta-model? How to bridge the gaps between the different architecture knowledge meta-models of various organizations? And etc, or how can we share architectural knowledge? [2] This can be detailed as: How can we deliver or make accessible the right knowledge to the right person at any given point in time? How can we realize the necessary knowledge management strategies? Can we build a common knowledge base for a web community? Etc. In our opinion, the questions and issues regarding AK sharing are very important to be addressed when dealing with knowledge exchange. Thus, by surveying, writing summaries, and analyzing current research on AK sharing, we can get a better idea of these approaches, models and issues related in AK sharing. This paper is aimed to explore the current research being done on AK sharing, survey suggested approaches, models, and consider the issues that arise in sharing of such AK.

The rest of the paper is organized as follows: Section 2 presents the background on Architectural Knowledge (AK) and AKS requirements and prerequisites; Section 3 describes the surveyed works on AKS as well as their associated strengths and issues; finally, section 4 reports the conclusion and future work on this area of research.

## 2. BACKGROUND

This section provides the reader with information on architectural knowledge and its definitions as well as, elaborating on AK sharing and its aspects, which help a better understanding of the rest of the paper.

### 2.1. Architectural Knowledge (AK)

The subject of architectural knowledge is complex and covers many issues, both general and domain specific. It is truly a multi-disciplinary domain within software engineering and knowledge engineering [5].

In general, there are two ways to share knowledge: *person-to-person* and *person-to-artifact* [7]. And two main kinds of knowledge: *factual knowledge* and *expertise*. Factual knowledge relates to the things that you know, and pieces of knowledge that you can use. Architectural knowledge is factual knowledge, consisting of facts that can be recorded and disseminated. Explicit representation of architectural knowledge is essential in software development because it facilitates building and evolving quality systems [10, 11, 8, 9].

#### 2.1.1. AK definition

There appears to be no commonly accepted definition of what architectural knowledge entails [1] so that different works and/ or authors define dissimilar definition of the same term. This may cause misunderstanding and it is hard to find out if these, indeed, denote the same concept, which consequently influences the sharing architectural knowledge among the parities. Therefore, in this work in order to avoid this and give the reader a better understanding of the all aspects of AK definition the result of a recent systematic review of AK definition [1] is presented, where they synthesize all the definitions of the AK from different literature in order to give a clear definition of what architectural knowledge entails. These definitions are listed in Table 1, ordered chronologically by publication date.

From the Table 1, it becomes clear that a prefect definition of architectural knowledge that everyone being in agreement on is still not found. Therefore, it is recommended, researchers to be specific and concrete in defining the semantic of their AK to get over this lack, which consequently helps community to work on a common realization of the term, where the confusion and ambiguity is avoided.

### 2.2. AK Sharing

Sharing knowledge of software architectures becomes more and more important. However, in order to define suitable sharing mechanisms for architectural knowledge, insight into the software organization and its architecting process in particular, is required. If this is not investigated properly, chances are that methods and tools used to guide stakeholders in the architecting process do not match with people’s everyday work. A consequence of this might be a low acceptance rate for these methods and tools, and a poor return on investment, which highlight the need of AK sharing. Therefore, a successful architectural knowledge sharing is only possible when the sharing mechanisms are tailored to the architecting process [6].

#### 2.2.1. Issues related to AK sharing

The issues presented are based on the results of a recent case study in [12], which are briefly explained as follows:

- *No consistency between architecture and design documents:* There is no alignment between the architecture descriptions and the functional design and technical design documents used by developers and maintainers. Because of the lack of alignment, valuable architectural knowledge might be dispersed in the organization without the architects knowing it. Consequently, it is hard to judge whether the architectural description conflicts with the design that is preferred by the developers.
- *Communication overhead between stakeholders:* Developers occasionally have to explain the architect’s technical decisions more than once.

The reason for this is that decisions made in earlier meetings, including the rationale for these decisions, are not adequately stored in the architecture description. This knowledge sometimes dissipates quickly. Consequently, architects need to meet again with the developers to get this knowledge explicit at a later point in time.

- *No explicit collaboration with maintenance teams:* Although maintainers are targeted in the architectural documentation, they are not involved as a stakeholder in the architecting process. No active discussions between architects and maintainers take place and the requirements of the maintenance teams are not taken into account during architecture development.
- *No feedback from developers to architects:* Developers sometimes wear the hat of the architect and also make design decisions. However, architects are not informed on the decisions made by the developers unless explicit meetings take place. There is no mechanism in place that allows developers to share what they are doing. Therefore, it is very difficult for the architect to find out what kind of technical issues are encountered or what detailed decisions are taken.
- *No up-to-date knowledge from development teams in repository:* The architectural knowledge repository contains little to no information on the 'best practices', technology preferences and standards, and expertise currently available at the development teams. Therefore, the repository is unable to advise architectural directions that match with the development processes.
- *No up-to-date knowledge from main customer in repository:* The architectural knowledge repository also lacks up-to-date knowledge on the customer organization's reference architecture. Therefore, the repository cannot give architectural directions that automatically comply with the constraints posed by this reference architecture.

### 2.2.2. Prerequisites for successful AK sharing

Implementing means for architectural knowledge sharing is definitely not only about choosing the right tool [12]. Successful knowledge sharing can only be achieved if the necessary incentives are created. These incentives induce stakeholders to share valuable architectural knowledge, such as the major design decisions made, the underlying rationale for these decisions, and alternatives that were considered. Refer to [12] for the explanation of these incentives and the way that they were identified

In order to address the aforementioned issues, section 2.2.1, we present a set of prerequisites for architectural knowledge sharing based on the work in [24]. They argue that successful architectural knowledge sharing is only possible if these prerequisites are met. The four prerequisites are listed below:

- *Alignment between design artifacts:* Architectural descriptions need to be aligned with other design documents. This can be done by enriching the architectural description with links to relevant (lower level) design documents, allowing developers or more technical stakeholders to more easily find their way in the set of documentation. This prerequisite deals with issue: 'No consistency between architecture and design documents'.
- *Traceability between architectural decisions and Descriptions:* If all architectural design decisions are documented using specific templates, including considered alternatives and the rationale for the decisions; architectural descriptions provide a good summary of the decision-making process that leads to certain architecture. As a result, communication between architects and other stakeholders, such as developers, will improve since the current state of the architecting process is known at all times. This prerequisite therefore addresses issue: 'Communication overhead between stakeholders'.
- *Architects fulfill a central role:* The architects need to fulfill a central role in the architecting process. This guarantees better communication with all involved stakeholders through frequent formal and informal meetings, direct involvement of developers in decision-making and better collaboration with the maintenance teams. This prerequisite addresses issues: 'No explicit collaboration with maintenance teams' and 'No feedback from developers to architects'.
- *Central architectural knowledge repository:* A central architectural knowledge repository allows for storing valuable input on the decision-making process from all stakeholders involved. This prerequisite therefore directly addresses issues: 'No up-to-date knowledge from development teams in repository' and 'No up-to-date knowledge from main customer in repository'.

### 2.2.3. Requirements for AK sharing

There are five main requirements identified in [15] for an architectural knowledge sharing environment, which are briefly explained in the following:

- *Integration*: This requirement is considered most important by the architects is that a tool environment should offer a central point of access to the various types of functionality available. This central point of access should be both attractive and intuitive.
- *Project view*: This requirement gives support for a project view that enables management of project-specific architectural knowledge. The main advantage of such a project view is that it offers a central point of access to easily search all architectural knowledge related to a particular project. For stakeholders that join a project at a later point in time, such a central point of access is helpful to quickly become acquainted with the ins and outs of the project. In addition, it contains information about the project stakeholders. This information may include standard personal contact information, but also more architectural knowledge related content such as expertise areas of people
- *Manage documentation:* Related to the previous requirement is support for managing documentation. The difference with the project view is that the scope may be (much) broader, including all sorts of company documents.
- *Community building:* In contrast with the need for document management is the architects' wish to support building a community within their department. Consequently, requirements in this category include support for discussions and sharing expertise, but also overviews of 'who knows what' and 'who is doing what' in the organization. Finally, the ability to share news and events with colleagues would further add to the community feeling.
- *Constructing architecture descriptions:* The last main requirement relates to one of the primary deliverables of the architects: architecture descriptions. These documents usually contain a variety of architectural knowledge, and usually take multiple days or weeks to construct. All sort of automated support during the process of making well founded decisions, followed by reflecting these decisions in the architecture description is highly appreciated [15].

The aforementioned requirements are must for a desired approach to architectural knowledge sharing. However, the reason for presenting the given requirements and prerequisites is to give the reader the first impression of AKS problems as well as being used as comparison criteria in our future work to survey the effectiveness of the proposed approaches in section 3.

## 3. APPROACHES ON ARCHITECTURAL KNOWLEDGE SHARING

Since AK is a relatively new attention; only a few works/ approaches for sharing the architectural knowledge have been proposed. Furthermore, we thoroughly looked at the current literatures, domain analysis, regarding AKS. We identified four general categories of AK sharing mechanisms, which are (1) AK sharing through model, (2) AK sharing using pattern structure, (3) AK sharing via workspace, and (4) AK sharing through portal. On this regard, all the related works about AKS were further classified to these main categories in which each category may encompass more than one proposed approach.

However, to survey the current state of the art and practice in this field, this section summarizes the surveyed works on AKS and its variants, where the mechanism as well as strengths and drawbacks/ issues associated with each approach is identified as well.

### 3.1. AK Sharing Through Model

In this category we found a variety of approaches/ works comparing to the other categories. In the following, these approaches are presented.

#### 3.1.1. Approach I

Preliminary work on AK sharing in a knowledge grid through models has been carried out by de Boer *et al*. [21]. They propose a model of architectural knowledge that acts as a common frame of reference and enables architectural knowledge sharing, which is known as *Griffin core model*. The core model of architectural knowledge is depicted in Figure 1. The core model leaves room for the use of different architecture description methods, including IEEE-1471. On the other hand, two perspectives can be viewed on the core model, namely data integration and service integration. For data integration the core model becomes a reference model for sharing architectural knowledge. For service integration, it provides the means to integrate the services that a grid infrastructure may provide. Refer to [21] for a more detailed information about core model.

In order to use the given model in architectural knowledge sharing, of course in grid-setting, the model is deemed as an instantiation of the knowledge grid discussed in [22]. The basic idea is sketched in Figure 2. The model depicted in the center is the core model of what concerns architectural knowledge. Organization-specific models provide a specialization hereof. Refer to [21] for information on envision of architectural knowledge sharing in a grid-setting.

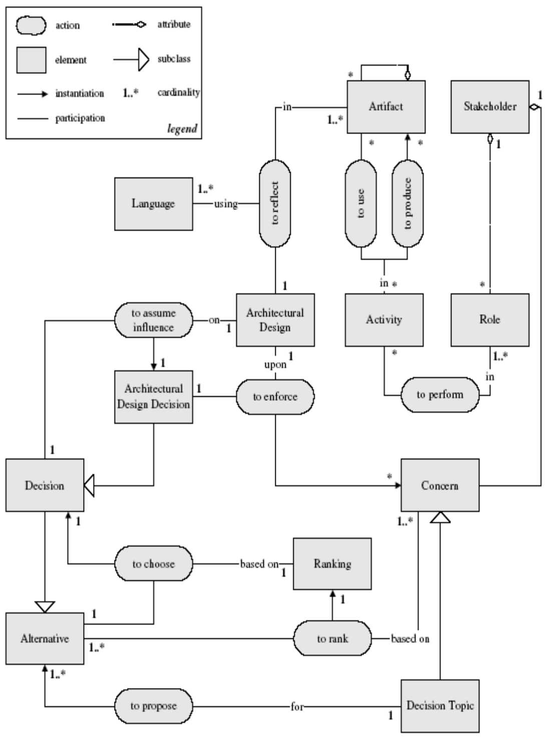

Figure 1. Core Model of Architectural Knowledge [21]

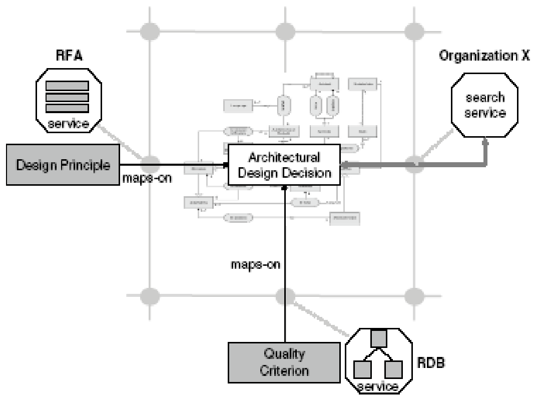

Figure 2. Sharing Architectural Knowledge in a grid-setting [21]

### 3.1.2. Strengths and issues (Approach I)

Having a core model of architectural knowledge has a number of advantages, *(1)* it defines, from a data integration perspective, a vocabulary for architectural knowledge: the minimal set of common notions that is needed when architectural knowledge has to be made explicit. Terminology, processes, and concerns particular to a specific organization or domain can be expressed in terms of core model concepts. Roughly speaking, the organization-specific terminology lies like a shell around the core model, which helps organization to define their own methodologies. *(2)* With a shared core model it becomes easier to agree on a common terminology, which makes terminological misunderstanding to be avoided.

The issues with this approach are, the detailed and complete mapping relationship between Griffin Core Model and shell model is not presented, and some relationships between the elements in core model are not well defined. This can hamper the effective implementation of AK grid, and further development in AK sharing and management. Besides, this model does not allow multiple stakeholders sharing the same concern. These issues later were addressed in [23].

### 3.1.3. Approach II

This approach [23] is a refinement of the Griffin core model in the sense that it overcomes the problems, as described in section 3.1.2, associated with the original model as well as, argues a UML class diagram as a better means for representation of core model of AK sharing.

The motivational arguments deal with the question of why UML is better for the representation of AK core model. On this regard, the advantages of doing so are presented in [23] as well. However, the *refined Griffin core model* in UML as shown in Figure 3 can be regarded as a model mapping based on Griffin core model in [21] from UML perspective. In this model, all the 12 entity concepts in the Griffin core model are inherited, and some of the actions as the relationship between concepts are used but some action are renamed in order to make it as simple as possible for easy understanding. For example, the "perform" action between Role and Activity entity is mapped as a "perform" relationship between Role and Activity concept in the refined Griffin core model in UML. Refer to [23] for more information on transformation from UML model to OWL mode and case of concept mapping.

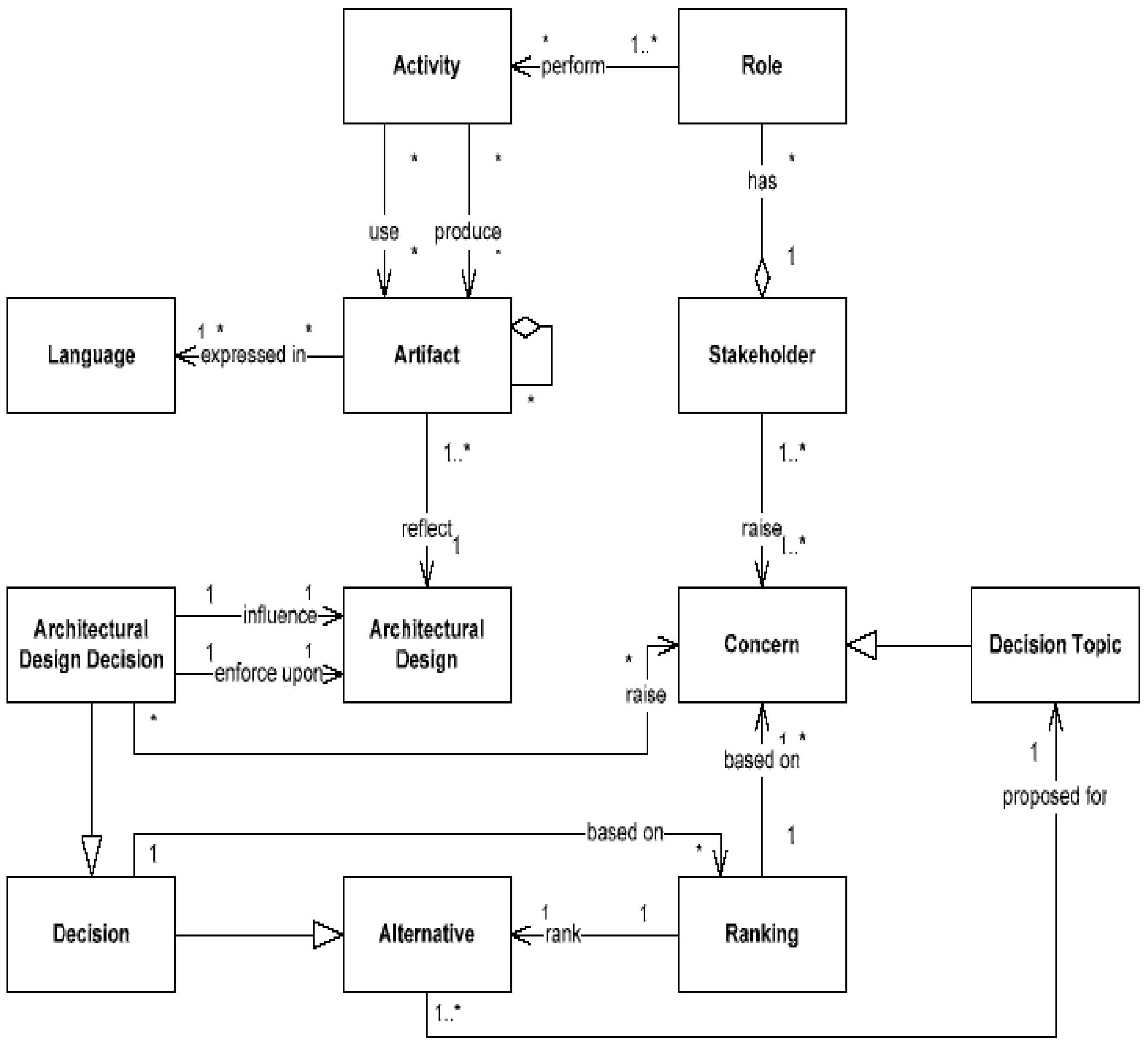

*Figure 3. Refined Griffin Core Model represented in UML [23]*

### 3.1.4. Strengths and issues (Approach II)

This approach gives a clearer understanding about how concept mapping on AK works and which can be a base for the evaluation of instance mapping quality on AK. Besides, it enables smooth AK sharing and management since the UML is the best language, could be, for the common understanding, especially among the stakeholders.

Some issues regarding this approach are still open to be addressed, such as: how to minimize the change impact of core model modification, and the model mapping between core model and the other domain model. How to define the mapping quality evaluation process in instance level, and what is implicated relationship between the quality prediction in the model level and practical mapping result in instance level. The issues regarding quality evaluation have recently been addressed in [24, 25].

### 3.1.5. Approach III

This approach is built on strategy that it looks at the AK sharing through a knowledge grid, where AK is captured in domain-specific models [24]. The use of such models allows different organizations, departments, or even persons to express their AK using their own concepts in the AK grid. Using the mappings between these models, all AK is transparently shared among the interested stakeholders, as the AK is expressed for each person in terms of his or her own domain model(s). To implement this strategy, two different sub-approaches are proposed in [24]. The first approach is a *direct mapping* approach, in which all models are directly mapped onto each other. The second approach is an *indirect mapping* approach, in which all the models map onto one central model. This central model acts a mediator between the different domain models. On this regard, AK sharing is generalized in two levels: the conceptual level and the instance level. At the conceptual level, an AK model defines the *concepts* and *relationships* that a particular organization, department, project, or person uses. At the instance level, the *instances* of the aforementioned concepts and relationships exist. Therefore, with the direct mapping approach, one defines mapping relationships from a source AK model to a target AK model directly. The source AK model is the model in which the instances to be translated are defined. The target AK model is the language in which one would like to use the AK. An indirect mapping approach defines mapping relationships from a source AK model to a target AK model through a central model, which acts as a mediator.

### 3.1.6. Strengths and issues (Approach III)

One of the strengths associated with this approach is that it solves the problem of local domain model, in which different stakeholders can share AK through their own domain models. However, the tradeoff between the cost and the quality of the AK sharing when using these two sub-approaches is still an interesting challenge and ongoing research. Recently, these issues have been, to some extent, addressed in [24, 25].

## 3.2. AK Sharing Using Pattern Structure

Patterns have proven to be a good way to codify knowledge in the field of software design, where the knowledge can be shared through them. Functional design patterns (FDP) are patterns that describe recurring functionality of applications [18].

This approach is hypothesized that extending Functional Design Patterns with the explicit representation of the problem space in terms of conflicting conditions (forces and tensions) has the potential to improve sharing of architectural knowledge. To this end, this approach utilize a novel sort of patterns called *casual patterns* [17] as units to share knowledge, which are a combination of problem-description in terms of forces, tensions and tradeoffs with solution, as depicted in Figure 4.

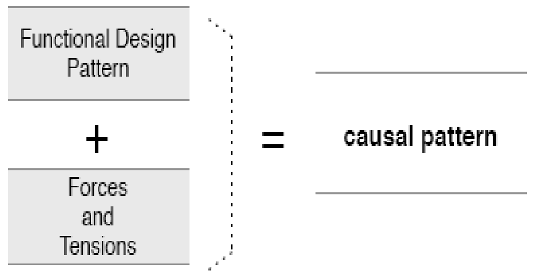

*Figure 4. Putting together functional design patterns and forces and tensions [17]*

Causal patterns are novel because no previous research presented the idea of including forces and tensions into patterns explicitly. Moreover, template of such casual pattern can be viewed in [17].

This mechanism asserts that putting together a functional design pattern and the explicit representation of forces and tensions, the author of the pattern communicates critical conditions that architects using the pattern should be aware of. In this way, the architect is able to better reason about the outcome and tradeoffs of applying the pattern. More specifically, causal patterns improve sharing of architectural knowledge because they make explicit the rationale behind a design decision. Therefore, according to the experts, causal patterns can be used to communicate the “why” of a decision, which in turn support architects in a more efficient sharing of AK.

The word ‘efficient’ may raise this question that why these casual patterns are called efficient? To answer this question, the authors in [17] use the ontology and their reasoning proposed in [19] to discuss the effectiveness of causal patterns, where it showed that causal patterns cover the relevant issues supported by the ontology, viz., kinds of architectural design decisions, attributes of architectural design decisions, and relationships between architectural design decisions.

### 3.2.1. Strengths and issues

This approach, casual patterns, brings some advantages, *(1)* providing the rationale behind repetitive architectural decisions: When architects miss the reason why an important decision was made, architects spend much time in rediscovering the reason behind the old architectural decision. Unaware of this decision, architects are afraid to implement changes. Causal patterns seem to solve this disadvantage of patterns. *(2)* Causal patterns are crystallized pieces of understanding: It does not mean that architects copy solutions when they use causal patterns. It means that causal patterns are a tool to sanitize dynamics of the environment and share them in time.

A main issue with this approach is how to disseminate such patterns in which they can be transferred and used. Therefore, research needs to be done to find a method, in a maintainable way, to address this issue.

### 3.3. AK Sharing Via Workspace

*Workspaces* are virtual spaces, which are conceived as a logical counterpart of physical spaces and are based on physical metaphors [13, 14]. Like physical spaces, these workspaces are expected to make all the required objects, tools, people, and guidelines available along with all the necessary communication channels and coordination mechanisms. A workspace is expected to create opportunities for the users to turn them into a place of collaboration as in a virtual world it is not the spatial features of a space that matters the most, rather what the users of such a space can do within it and that is what turns such a space into a place. That is why it is vital that a virtual space provides its users with an opportunity to turn it into a place for collaboration.

Figure 5 shows the structure of the mechanism of knowledge-sharing workspace along with its major components, which was designed in [16] to support the different activities (such as design, evaluation, and documentation) of the software architecture process in the context of global software development. All the objects of the workspace are related with one another according to a meta-model of collaboration presented in [14]. Actors assuming organizational roles collaborate with other roles within same workspace or in another workspace to perform designated actions on artifacts to achieve desired organizational goals [13]. Artifacts are either inputs for or outputs from different actions. Roles are attached to a particular workspace and are unique within that workspace; more than one workspace may have the same roles attached to them. For example, a manager role may be attached to the workspaces designed to support the plan architecture evaluation, and prepare and manage results activities of the proposed process. Refer to [16] for the detailed explanation of the components.

#### 3.3.1. Strengths and issues

This approach enables the scalability of the needs of distributed and virtual teams in the context of global software development, which are not addressed in the most the tools, for capturing and sharing AK, designed based on the traditional workflow paradigm. Instead, it uses an electronic workspace paradigm for sharing AK. Besides, these workspaces allow participants to share knowledge in manners increasingly being used by different communities. Participants of such workspaces can produce or consume AK using codification strategy and/or easily consult known experts and/or peers wherever needed using personalization strategy. Moreover, teams can easily change the workspace objects and participants as the work progresses and teams evolve.

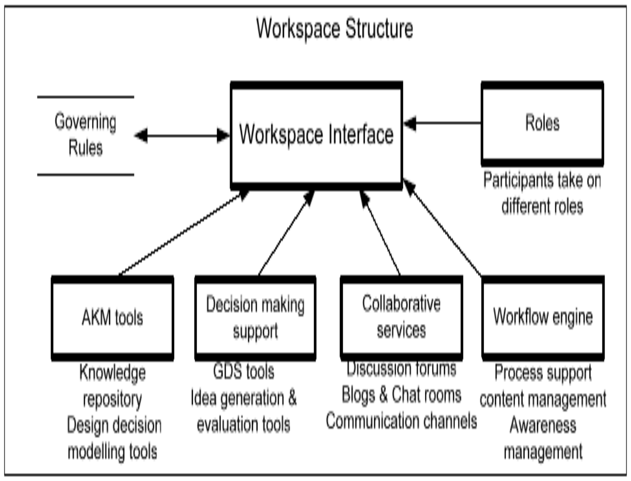

*Figure 5. The structure of a knowledge-sharing workspace and its major components [16]*

One issue with this approach is the immaturity of that to be applied in practical architecture evaluation process. Therefore, research needs to be conducted to experiment the given approach before it can be employed in the practice.

### 3.4. AK Sharing Through Portal

The design and implementation of this approach stems from questions what architects' specific architectural knowledge sharing needs are, and how best to fulfill these needs. In response to this questions work in [15] has discovered that architects are especially in need for *'Just-In-Time (JIT) architectural knowledge'*, which is defined as access to and delivery of the right architectural knowledge, to the right person, at any given point in time. Such architectural knowledge may include updates on major decisions made or discussions held, but also contact information or expertise of important stakeholders.

Therefore, to fulfill the above needs, this approach utilizes a web-based architectural knowledge sharing portal [15]. This portal harbors various types of architectural knowledge, which can be easily retrieved using a number of integrated codification and personalization techniques.

The portal is in essence a client server system; a web browser communicates with an Apache web server. Asynchronous communication between the client and server is applied whenever possible, to foster the speed and usability of the application. All architectural knowledge is stored in a relational database. Additional meta-data is stored with this architectural knowledge to make retrieval easier. For the client side of the portal, a suitable open source framework: Portaneo is selected, a Rich Internet Application. Portaneo is highly modifiable, has a flexible plugin system –making the portal highly extensible – and, above all, is free. These characteristics make it a better choice compared to existing commercial software such as Microsoft Sharepoint, because with Portaneo experiment more easily can be done with the portal, while using little resources. For a more detailed discussion about the portal's architecture and its features refer to [20].

Furthermore, the portal has three main features, as follows:

- *Integrated functionality:* The portal offers a central access point to various types of functionality by means of a start page.
- *Stakeholder-specific content:* The portal offers an intuitive and attractive user interface. Since architects are already familiar with web pages, navigating the portal is easy.
- *Notifications and subscriptions:* The portal has a built-in subscription and notification system. Architects can subscribe to specific architectural knowledge topics (e.g. a topic of a discussion forum) or artifacts (e.g. a document).

The above three features together ensure that [15] portal offers support for what is defined as JIT architectural knowledge. The integrated functionality provides access to *'the right architectural knowledge'*. The support for stakeholder-specific content ensures that *'the right person'* finds what he wants. Finally, the subscription and notification mechanisms allow architects to stay up-to-date by delivering the relevant architectural knowledge to them when needed.

### 3.4.1. Strengths and issues

The portal's integrated functionality is one the strengths that supports architects in their decision-making process, by providing easy access to the right architectural knowledge at any given point in time (support for JIT). Furthermore, the given approach also supports personalization( besides codification) techniques such as collaboration using portal's discussion board for sharing architectural knowledge because it helps stakeholders to find each other through the portal, which in turn can be considered as a good first step to create a real 'community of architects'.

One issue with respect to this approach is the low satisfaction of its best practices repository, where architectural knowledge is codified in predefined formats, and could be retrieved for various purposes. Because, adding or modifying the best practices is time-consuming and error-prone, the costs for keeping the content up-to-date outweigh the benefits as well. Therefore, research needs to be carried to make the repository more intelligent, better maintainable, and better-looking.

## 4. CONCLUSION AND FUTURE WORK

In this paper an overall introduction to the problems of AK sharing is given. Background and basic terminology on architectural knowledge and architectural knowledge sharing is presented. A summary of surveyed works on sharing approaches, including mechanism, strengths, and issues, for AK is provided as well. Finally, conclusion and future work are included, which addresses the interesting research trends in connection with this field.

Based on the information that was presented in the overall paper, the following conclusions can be drawn:

- Although AK is relatively new, there is plenty of research on AKS, which shows the increasing interest on architectural knowledge and sharing within the software engineering and architecture community. However, much of this research is being conducted towards problem analysis, software architecture, and implementation techniques.
- Based on the survey shown among AKS approaches and the fact that these sharing approaches are improved from the issues brought up on previously AK approaches, at the moment and to the best of our knowledge and understanding of the surveyed AKS approaches, it is not easy to select the best approach for AKS, as with AK sharing itself. But the AK sharing approach via workspace proposed by [16] shows to be one of the most effective sharing approach of AK at the present time.

Future works for this study include surveying more AK sharing approaches and incorporate new AKS approaches that are in the process of research. Besides, (as immediate future works) the effectiveness of proposed AKS approaches will be compared and analyzed in terms of their ability to fulfill different kind of requirements and prerequisites of AKS ,as were described in sections 2.2.2 and 2.2.3. More specifically, those requirements will be used as comparison criteria towards determining an effective mechanism of AKS in global software development. Furthermore, the investigation of Aspect-Oriented paradigm as mediator connector in indirected sharing AK model approach, where AK is viewed as a *process*, will be considered.

## 5. ACKNOWLEDGMENT

The authors would like to thank Assistant Professor Dr.Paris Avgeriou, University of Groningen, and The Netherlands, who provided them the opportunity to come up with this study.

## 6. REFERENCES

[1] R.C. de Boer, R. Farenhorst , "In Search of Architectural Knowledge", *In Proceedings of the 3rd international workshop on Sharing and reusing architectural knowledge (SHARK08)*, ACM, Leipzig, Germany, pp. 71-78, 2008.

[2] P. Lago, P. Avgeriou, R. Capilla, and P. Kruchten, "Wishes and Boundaries for a Software Architecture Knowledge Community", *In Seventh Working IEEE/IFIP Conference on Software Architecture*, IEEE computer society, pp.271-274, 2008.

[3] N. B. Harrison, P. Avgeriou, and U. Zdun. "Using patterns to capture architectural decisions", *IEEE Software*, pp.38-45, 2007.

[4] P. Lago, P. Avgeriou, "First Wrokshop on Sharing and Reusing Architectural knowledge", *SIGSOFT Software Engineering Notes 31(5)*, ACM, pp.32-36, 2006.

[5] P.Avgeriou, P.Kruchten, P.Lago, P. Grisham, and D. Perry, "Sharing and Reusing Architectural Knowledge - Architecture, Rationale, and Design Intent", *In 29th International Conference on Software Engineering (ICSE'07 Companion)*, IEEE computer society, 2007.

[6] R. Farenhorst, "Tailoring Knowledge Sharing to the Architecting Process", *SHARK'06* , ACM, Torino, Italy, 2006.

[7] M. T. Hansen, N. Nohria, and T. Tierney, "Whats your strategy for managing knowledge?", *Harvard Business Review*, pp. 106–116, 1999.

[8] A. Jansen, J. van der Ven, P. Avgeriou, and D. Hammer, "Tool support for architectural decisions", *In 6th Working IEEE/IFIP Conference on Software Architecture (WICSA)*, 2007.

[9] P. Kruchten, P. Lago, and H. van Vliet, "Building up and reasoning about architectural knowledge", *In 2nd International Conference on the Quality of Software Architectures*, 2006.

[10] M. A. Babar, R. C. de Boer, T. Dingsøyr, and R. Farenhorst, "Architectural knowledge management strategies: Approaches in research and industry", *In Second Workshop on SHAring and Reusing architectural Knowledge Architecture, Rationale, and Design Intent-TOC (ICSEW'07)*, IEEE Computer Society, Minneapolis, USA, 2007.

[11] R. de Boer, R. Farenhorst, J. van der Ven, V. Clerc, R. Deckers, P. Lago, and H. van Vliet, "Structuring software architecture project memories", *In 8th International Workshop on Learning Software Organizations (LSO)*, 2006

[12] R. Farenhorst, P.Lago, H. van Vliet, "Prerequisites for Successful Architectural Knowledge Sharing", *In proceedings of the 2007 Australian Software Engineering Conference (ASWEC'07)*, IEEE computer society, 2007.

[13] I. Hawryszkiewycz, "Designing the Networked Enterprise", Artech House, Boston, USA, 1997.

[14] R.P. Biuk-Aghai, I.T. Hawryszkiewyez, "Analysis of Virtual Workspaces", *Proceedings of the Database Applications in Non-Traditional Environments*, 1999.

[15] R.Farenhorst, R. Izaks, P. Lago, H. van Vliet, "A Just-In-Time Architectural Knowledge Sharing Portal", *Seventh Working IEEE/IFIP Conference on Software Architecture*, IEEE computer society, 2008.

[16] M. A. Babar, "The Application of Knowledge-Sharing Workspace Paradigm for Software Architecture Processes", *SHARK'08*, ACM, Leipzig, Germany, 2008.

[17] M. L. Ponisio, K.Sikkel, E. Vermeulen, "Structures to Effectively Share Architectural Knowledge", *Proceedings of the IASTED International Conference on Software Engineering as part of the 26th IASTED International Multi-Conference on Applied Informatics*, ACTA PRESS, Innsbruck, Austria, pp.192-199, 2008.

[18] J. Snijders, Functional design patterns, Master Thesis, Utrecht University, 2004.

[19] P. Kruchten, “An ontology of architectural design decisions in software intensive systems”, *In 2nd Groningen Workshop on Software Variability*, pp. 54–61, 2004.

[20] R. Farenhorst, P. Lago, and H. van Vliet, “EAGLE: Effective Tool Support for Sharing Architectural Knowledge”, Under submission.

[21] R.C de Boer, R. Farenhorst, P. Lago, H. van Vliet, V.Clerc, and A .Jansen, “Architectural knowledge: getting to the core”, *In Proceedings of 3rd International Conference on the Quality of Software-Architectures (QoSA)*, Boston, USA. pp. 197-214, 2007.

[22] H. Zhuge, “The Knowledge Grid”, *World Scientific Publishing Co.*, Singapore, 2004.

[23] P. Liang, A.Jansen, and P. Avgeriou, “Refinement to Griffin Core Model and Model Mapping for Architectural Knowledge Sharing”, RUG-SEARCH-07-L01, 2007.

[24] P. Liang, A. Jansen, and P. Avgeriou,” Sharing architecture knowledge through models: quality and Cost”, *The Knowledge Engineering Review*, Cambridge University Press, 2008.

[25] P.Liang, A. Jansen, and P. Avgeriou,“ Selecting a High-Quality Central Model for Sharing Architectural Knowledge”, *The Eighth International Conference on Quality Software*, IEEE computer society, pp.357-365, 2008.

[26] A. Ran and J. Kuusela,” Design Decision Trees”, *In 8th International Workshop on Software Specification and Design*, pp. 172–175, 1996.

[27] E. Carayannis and J. Coleman, “ Creative System Design Methodologies: the Case of Complex Technical Systems”, Technovation, pp.831–840, 2005

[28] S. Chen, “Task Partitioning in New Product Development Teams: A Knowledge and Learning Perspective”, *Journal of Engineering and Technology Management*, pp.291–314, 2005.

[29] P. Kruchten, P. Lago, H. van Vliet, and T. Wolf, “ Building up and Exploiting Architectural Knowledge”, *In 5th Working IEEE/IFIP Conference on Software Architecture (WICSA)*, Pittsburgh, Pennsylvania, USA, pp. 291–292, 2005.

[30] P. Kruchten, H. Obbink, and J. Stafford, “The Past, Present, and Future for Software Architecture”, *IEEE Software*, pp.22–30, 2006.

[31] M. A. Babar, I. Gorton, and B. Kitchenham,” A Framework for Supporting Architecture Knowledge and Rationale Management”, *In Rationale Management in Software Engineering*, pp. 237–254. 2006.

[32] L. Lee and P. Kruchten, “Capturing Software Architectural Design Decisions”, *In P. Kruchten, editor,Canadian Conference on Electrical and Computer Engineering (CCECE)*, pp. 686–689, 2007.

[33] R. C. de Boer , H. van Vliet, “Constructing a Reading Guide for Software Product Audits”, *In 6th Working IEEE/IFIP Conference on Software Architecture (WICSA)*, Mumbai, India, pp. 11–20 2007.

[34] R. Farenhorst, P. Lago, and H. van Vliet, “ Effective Tool Support for Architectural Knowledge Sharing”, *In 1st European Conference on Software Architecture (ECSA)*, Aranjuez (Madrid), Spain, pp. 123–138, 2007.

[35] I. Habli and T. Kelly, “Capturing and Replaying Architectural Knowledge through Derivational Analogy” , *In Second Workshop on Sharing and Reusing Architectural Knowledge - Architecture, Rationale, and Design Intent, (SHARK/ADI)*, Minneapolis, USA, 2007.

[36] M. A. Babar and I. Gorton, “ A Tool for Managing Software Architecture Knowledge”, *In Second Workshop on Sharing and Reusing Architectural Knowledge - Architecture, Rationale, and Design Intent, (SHARK/ADI)*, 2007.

[37] R. Bahsoon, “ Defining Dependable Dynamic Data-Driven Software Architectures”, *In IEEE International Conference on Information Reuse and Integration, (IRI)*, pp. 691–694, 2007.

*Table 1. Definitions of 'architectural knowledge'*

| | |
|---|---|
| Ran and Kuusela (1996) [26] | To avoid replication when representing variations and alternatives DDT structures architectural knowledge hierarchically into fine-grain elements we call design decisions. |
| Carayannis and Coleman (2005) [27] | The architectural innovation is dependent on the system designers' knowledge of the components in the system and their knowledge of the configuration of the components. Henderson and Clark (as cited in Afuah, 1998) show the knowledge as Component Knowledge (CK) and the latter Architectural knowledge (AK). |
| Chen (2005) [28] | A distinction that is particularly significant in the product innovation context is the distinction between component-specific knowledge and "architectural" knowledge (Henderson and Clark, 1990). Component knowledge is knowledge that concerns a particular aspect of an organization's product, process or operation. Architectural knowledge, on the other hand, relates to the various ways in which the components are integrated and linked together into a complete system. |
| Kruchten et al. (2005) [29] | Architectural knowledge consists of architecture design as well as the design decisions, assumptions, context, and other factors that together determine why a particular solution is the way it is. |
| Kruchten et al. (2006) [9] | Architectural Knowledge = Design Decisions + Design, derived from 'Architectural knowledge consists of architecture design as well as the design decisions, assumptions, context, and other factors that together determine why a particular solution is the way it is.' |
| Kruchten et al. (2006) [30] | Some researchers are looking into architectural knowledge – that is, architectural design decisions and their rationale. |
| Babar et al. (2006) [31] | We propose a framework for managing design rationale to improve the quality of architecture process and artifacts. This framework consists of techniques for capturing design rationale, and approach to distill and document architectural information from patterns, and a data model to characterize architectural constructs, their attributes and relationships. These collectively comprise Architectural Design Knowledge (ADK) to support the architecting process. |
| SHARK workshop (2006,2007) [4, 5] | Architectural Knowledge (AK) is defined as the integrated representation of the software architecture of a software-intensive system or family of systems along with architectural decisions and their rationale external influence and the development environment. |
| Lee and Kruchten (2007) [32] | Software architectural knowledge is composed of the design and the set of decisions made to arrive at the design. |
| De Boer and Van Vliet (2007) [33] | Following a recent trend in software architecture research we refer to the collection of architectural design decisions and the resulting architectural design as 'architectural knowledge'. |
| Farenhorst et al. (2007) [34] | [..] not only the architecture design itself is important to capture, but also the knowledge pertaining to it. Often, this so-called architectural knowledge is defined as the set of design decisions, including the rationale for these decisions, together with the resulting architectural design. |
| Habli and Kelly (2007) [35] | Architectural Knowledge = {drivers, decisions, analysis} |
| Babar and Gorton (2007) [36] | [The knowledge management component] provides services to store, retrieve, and update artifacts that make up architectural knowledge |
| Bahsoon (2007) [37] | We anticipate the architectural knowledge to constitute architectural artifacts such as deployable components and associated specification of what the components provide and require, quality requirements, scenarios corresponding to specific dependability requirements, and possibly dependable styles and patterns. |