Organizing a Software Architecture Body of Knowledge
—Summary of the 5th SHARK Workshop, at ICSE 2010

Patricia Lago
VU University Amsterdam
The Netherlands
patricia@cs.vu.nl

Paris Avgeriou
University of Groningen
The Netherlands
paris@cs.rug.nl

Philippe Kruchten
University of British Columbia
Vancouver, Canada
pbk@ece.ubc.ca

DOI: 10.1145/1838687.1838696 http://doi.acm.org/ 10.1145/1838687.1838696

Abstract

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. A fifth workshop on *Sharing and Reusing Architectural Knowledge* (SHARK) was held jointly with ICSE 2010 in Cape Town, South Africa. The theme of this workshop was the organization of a body of knowledge for software architecture knowledge management. It featured thirteen research position statements and three working groups that discussed on focused topics. This report summarizes the results of the discussions we held, and suggests some topics for future research.

1. Introduction

Software architecture plays an important role in managing the complex interactions and dependencies between stakeholders and serves as a reference artifact that can be used by stakeholders to share knowledge about the design of a system. Architecture also facilitates early analysis of the system, especially with respect to quality attributes and maintainability of the system. Current approaches of software architecting focus heavily on documenting components and connectors and fail to document the design decisions that *produced* the architecture – as well as the organizational, process and business rationale underlying those design decisions. This lack of relevant architectural knowledge and documentation can negatively impact maintenance costs and lead to architectural erosion and mismatch. The SHARK workshop series focuses on current approaches and trends that tackle this problem: methods, languages, and tools that can be used to extract, represent, share, apply, and re-use architectural knowledge.

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.

In this fifth edition of SHARK [1], we've proposed to the SHARK community to discuss and contribute on how to (re)organize and codify a Body of Knowledge of the WICSA community (WICSA BOK). This BOK is partially available through Software Architecture Portal [2] and www.wicsa.net, but it needs to be reorganized and unified.

2. Working Group Discussions

Out of 25 submissions, the workshop accepted 13 research and position papers[2] for inclusion in the proceedings. The papers were presented in short position statements and were classified into the following 4 categories: Modeling and visualizing architectural decisions; Creation of AK, using business goal, or rationale; Using AK for supporting the evolution of systems; Tools for sharing AK; all corresponding to various entries of the BOK. The presentations[3] of the accepted papers provided the basis for further dialogue among the workshop participants in several working group sessions. The topics selected for further discussion were:

1. Content of the BOK: what kind of AK should be stored in the repository

[2] Papers accepted for the SHARK 2010 workshop are available through the ACM Digital Library

[3] PDF versions of the presentation slides are available on the SHARK wiki at: www.cs.rug.nl/shark/

2. Support for the BOK: Web 2.0 and social networking techniques to support AK sharing
3. Integration: Integrating AK in various repositories

The following sections elaborate on the results of the discussions in each of the working groups.

2.1. Content of the BOK

The group first brainstormed about the type of information a BOK of AK should cover. We could reach consensus on the following preliminary observations, which we propose for further reflection. Notwithstanding the potential value of the knowledge codified in current web repositories (portals, blogs, catalogs, etc.), these repositories are not very much used. Possible reasons are: (1) they have too much of an academic flavor (practitioners do not perceive the direct usefulness for their goals, [3]); (2) some repositories like the WICSA wiki (wwwp.dnsalias.org/wiki) mix up BOK and conference information, which does not help reuse nor maintenance; (3) the field reached sufficient maturity to offer reusable knowledge. We should skip the academic discussion about definitions and dependencies among knowledge elements; rather, we need to identify the major usage scenarios [4] for academics and practitioners, and reach out for our potential users.

Along these lines, three main aspects have been identified as crucial for the BOK ontology under discussion: the function covered by the members of the community, or more in general the potential users of the BOK (i.e. role); the issues they expect to get answered by consulting the BOK (i.e. questions); and their ultimate (personal or professional) objective (i.e. goal). These three aspects have been used to define the BOK entry points for the community, as reflected in Table 1.

Roles	Questions	Goal
Architect in Business (experienced)	Know existing SA (set of SPLs)	Integrate new platform
Architect in Training, Architect in Business	Know of non-functional / architecture-relevant requirements in existing SA	Select the "best" SA
Architect in Training	Find examples of AK "x"	Learn
Architect in Business	Find examples of AK "x" (e.g. techniques, past AK)	Communicate
Communication Architect	How to communicate to my boss?	Communicate
Students (PhD, …)	Generic AK	Learn
Students	Context-specific AK	Experiment with

Table 1: RQC dimensions of the AK BOK

At this stage the group did further investigate the notion of AK (object of discussion of the SHARK community since its inception). We asked ourselves if we could agree on the gross types of AK a BOK should cover, the so-called AK "x" (as sketched in Table 1). We defined AK "x" as being either:

- Product (e.g. business goals): this corresponds to "WHAT to produce"
- Technique (e.g. how to represent something) : this corresponds to "HOW" to produce
- Enriching, co-product (e.g. rationale) : this corresponds to "WHY"
- Process (e.g. practices, skills) : this corresponds to "WHAT to do, WHO"

In summary, the above classification of what a knowledge element (AK "x") entails provides the framework for filling the BOK. We think that this should be pragmatic, essential, simple. In line with this idea, we envisage each **BOK element** as described by five core elements: stakeholder, description, example, author, (optional) application domain.

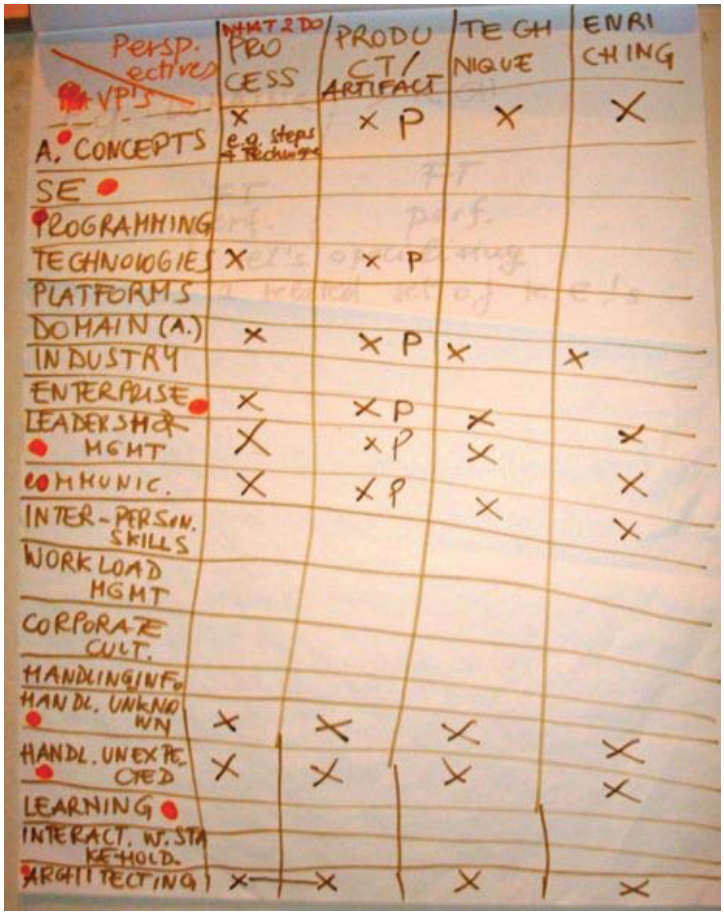

Figure 1: Subject areas or viewpoints

To jump-start the population of the BOK, we followed the classification given above and identified the core subject areas or viewpoints. We are happy to share with the community this exercise (see Figure 1) even if this is clearly just the result of a brainstorming session. Due to time constraints only some subject areas (identified subjectively as the most interesting for the attendees) were selected for discussion, of which only a subset could be actually filled. Interestingly enough, we all agreed that everything we should codify boils down to this list. This would mean that a BOK could cover this list as its ontological structure.

The last discussion topic the subgroup addressed was: we have already a history of not-so-successful repositories. What can we do to ensure success? What can we learn from the 'Web 2.0+' wave? Major selling points are twofold:

1. The repository should include various **feedback mechanisms** for users to vote on the value of a repository entry, to review entries, and it should offer statistics about actual access, usage, etc.
2. The repository should offer a **return on investment** for academics and practitioners to invest time and effort in sharing AK and in maintaining the BOK. We do not think that building a BOK easily results in high quality, mature publications (ROI for academics) or in industrial revenues (for practitioners). Nevertheless, if the appropriate mechanisms are in place, important returns can be gained in terms of: reputation, competition, self-motivation,

benchmarking.

2.2. Support: Web 2.0 and social networking techniques to support AK sharing

The subgroup looked at the tools and techniques from software engineering and from Web 2.0/social networking that could support the concept of a BOK. The key functions this set of tool should provide are:

- Entry and editing of AK, plus its evolution (CRUD)
- Visualization of AK
- Search through AK, by topics, people, keywords, etc.
- Storage and access control of AK
- Metrics and quality of AK elements: completeness, usefulness, usage, "ratings", …
- Integration of multiple sources.

A BOK support tool could be organized as shown in fig. 2. A layer is focused on people, active participants, "ghosts" or participants not actively involved in the BOK anymore, or roles. A lower layer organizes the information, with pointers to the actual source of information (documents, code, models, etc.), some of which could be cached. It was felt that wiki-like functionality, combined with versioning capability like GIT, would be best to deliver the functions envisaged with minimal effort.

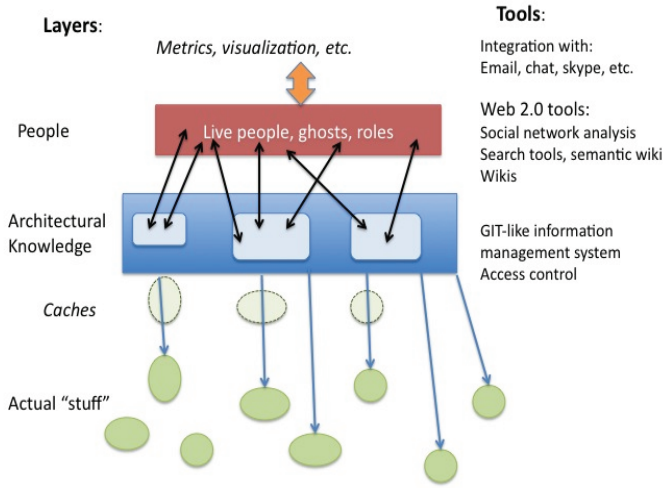

Figure 2: Tool support for an AK BOK

2. 3. Integrating AK in various repositories

The discussion started with a simple cataloguing of the different types of knowledge that the participants have "in stock" within their own business environment or research group. Our intention was to be pragmatic: we wished to explore what is already available and can be reused without expensive rework and even more so without following the elaborate templates [5] proposed by the community. This would allow the members of the community to start sharing knowledge instances with each other and experiment potential usages. This exercise resulted in the following catalogue:

1. Different types of software patterns and particular, architecture patterns, and evolution patterns (solutions for migrating software structures to satisfy new requirements). Besides the content of the patterns per se (application-generic knowledge) participants have available documented decisions of applying such patterns (application-specific knowledge)
2. Rationale, i.e. the reasons for making specific design decisions. This was accompanied by a argument ontology, that allows architects to reuse the arguments (e.g. that apply to
3. Models of different types, e.g. in Architecture Description Languages or in the Unified Modeling Language. This is the most commonly found form of AK.
4. Architecture views and their associated in terms of the IEEE 1471 std.
5. Artifacts that belong to the problem space, such as architectural-significant requirements, concerns of different stakeholders, business goals, quality attributes etc.
6. Constraints imposed by stakeholders, chosen technologies, the environment etc. The difference between constraints and requirements or concerns is of course a matter of debate.
7. Technologies that are considered or have been applied. The associated AK with technologies comprises the design decisions they entail, their consequences on the system's requirements, the structures or patterns they contain etc.
8. Traceability information between different types of artifacts across the software engineering lifecycle
9. Assumptions that have been recorded in order to enable better communication and decision-making.

Having this list of available knowledge, the group then attempted to prioritize on possible combinations: which pairs of the aforementioned types of knowledge would be more fruitful to integrate in terms of providing some value to stakeholders? The discussion yielded the following combinations (the list itself is not prioritized):

- Patterns and the models that are derived by instantiating the former.
- Business goals and the requirements derived by refining the former.
- Patterns and the rationale for applying them, usually in terms of impact on the system requirements
- Patterns and the requirements they are mean to satisfy
- Pattern and the decisions that are made by applying them

It is apparent that patterns are an important type of architecture knowledge that can be readily integrated with other types [6]. The main reason is that patterns already come in documented format and can be easily associated with other artifacts from both the problem and the solution space.

We subsequently tried to look at the AK integration from the opposite point of view: if we have established the integration of knowledge, how can we used it to improve the current state of the practice? We came up with the following list of use cases.

1. Architecture decision-making as the overarching knowledge production activity can be enhanced by integrating any of the AK types mentioned above.
2. Educating practitioners and providing them access to the right information at the right time. In particular integrating assumptions with the rest of the AK types would be of great help here.
3. Making the implications of decisions explicit and measurable could particularly benefit by integrating patterns and rationale.
4. Providing the right traceability and consistency between the different knowledge types and especially between models and design decisions.
5. Coverage of the problem space by the proposed solution and mitigating the risk to omit important requirements can be

significantly enhanced by integrating requirements with the rest of the AK types.

6. Improving the timing that decisions are made can be achieved by associating rationale with the rest of the AK types.
7. Conveying the right AK to the various stakeholders in order for them to check that their concerns are addressed, can be enhanced by relating requirements with decisions and the subsequent models as well as by mapping concerns to architecture views.

As a final point in our discussion, we reflected upon the potential research directions in the topic of integrating AK. A first promising research question is how to ensure the coverage of concerns or architecture-significant requirements by the set of design decisions, patterns, models. This is the typical traceability problem between problem and solution space, but it has never been researched from the AK point of view. A second research area would be to improve the consistency between different architecture views, e.g. the view of design decisions and the component-connector view. Again there are different approaches here but the semantics of different views are sometimes too apart to facilitate consistency checking.

Acknowledgments

We thank to all those who have participated in the organization of this workshop, particularly to the program committee, which was comprised of:

- Ademar Aguiar, University of Porto, Portugal
- Pierre America, Philips Research, The Netherlands
- M. Ali Babar, IT University of Copenhagen, Denmark
- Remco de Boer, ArchiXL, The Netherlands
- Jan Bosch, Intuit, Mountain View, California
- Rafael Capilla, Universidad Rey Juan Carlos, Spain
- Torgeir Dingsoyr, Sintef, Trondheim, Norway
- Peter Eeles, IBM, United Kingdom
- Rik Farenhorst, DNV Cibit, The Netherlands
- Jon Hall, Open University, UK
- Trevor Harrison, University of South Australia, Australia
- Rich Hilliard, independent consultant, USA
- Anton Jansen, ABB research, Sweden
- Ivan Mistrik, independent consultant, Germany
- Henry Muccini, University of L'Aquila, Italy
- Eltjo Poort, Logica, The Netherlands
- Antony Tang, Swinburne University of Technology, Australia
- Hans van Vliet, VU University Amsterdam, The Netherlands
- Uwe Zdun, University of Vienna, Austria
- Olaf Zimmermann, IBM Research, Switzerland

References

[1] SHARK web site: www.cs.rug.nl/~paris/SHARK2010

[2] Software Architecture Portal: www.softwarearchitectureportal.org

[3] Farenhorst, R., Izaks, R., Lago, P., and van Vliet, H. A Just-In-Time Architectural Knowledge Sharing, IEEE/IFIP Working Conference on Software Architecture (WICSA), IEEE CS, 2008, 125-134

[4] Lago, P., et al., The GRIFFIN Collaborative Virtual Community for Architectural Knowledge Management, in Mistrik, I.; van der Hoek, A.; Grundy, J. & Whitehead, J. (eds.), Collaborative Software Engineering, Springer, 2010, 195-218

[5] Jansen, A., Avgeriou, P., and van der Ven, J., Enriching Software Architecture Documentation. Journal of Systems and Software, Vol. 82(8), August 2009, Elsevier, 1232-1248.

[6] Harrison, N., Avgeriou, P., and Zdun, U. Architecture Patterns as Mechanisms for Capturing Architectural Decisions, IEEE Software, July-August 2007, IEEE Computer Society Press.