Architectural Design Decision: Existing Models and Tools

Mojtaba Shahin[1], Peng Liang[2], Mohammad Reza Khayyambashi[3]
[1]Department of Computer Engineering and IT, Sheikh Bahaei University, Iran
[2]Department of Mathematics and Computing Science, University of Groningen, The Netherlands
[3]Faculty of Engineering, Department of Computer Engineering, University of Isfahan, Iran
[1]mojtabashahin@gmail.com, [2]liangp@cs.rug.nl, [3]m.r.khayyambashi@eng.ui.ac.ir

Abstract

In the field of software architecture, there has been a paradigm shift from describing the outcome of architecting process mostly described by component and connector (know-what) to documenting architectural design decisions and their rationale (know-how) which leads to the production of an architecture. This paradigm shift results in emergence of various models and related tools for capturing, managing and sharing architectural design decisions and their rationale explicitly. This paper analyzes existing architectural design decisions models and provides a criteria-based comparison on tools that support these models. The major contribution of this paper is twofold: to show that all of these models have a consensus on capturing the essence of an architectural design decision; and to clarify the major difference among the tools and show what desired features are missing in these tools.

1. Introduction

Software architecture has an important role to mange complicated interactions between stakeholders of software-intensive systems in order to balance all kinds of constraints [1]. The architecting process can be considered as a decision making process through which the appropriate decisions must be made at the right time [7]. Current methods for the documentation of architecture concentrate on components and connectors [1], which causes problems such as expensive system evolution, lack of stakeholders communication and limited reusability of architecture [3,6]. Architecture as a set of architectural design decisions (ADD) was proposed to address these issues [4]. The ADD, assumptions, and architectural design integratedly make a concept called architectural knowledge (AK) [2], which has been a controversial issue in software architecture community recently. Nevertheless, capturing and managing AK through a systematic method can definitely improve the architectural capability of organizations, and also promote the interaction between stakeholders [10].

Practitioners and researchers have made great efforts to develop models and related tools to capture, manage and share ADD explicitly [12]. Each model has its own strong and weak points. These models are similar to each other and use different terms for describing the identical concept, which leads to the problem of terminological misunderstanding across organizations, which use different models to document their ADDs [5]. In this paper we analyze and compare 9 ADD models and related tools to express their similarities and dissimilarities. It shows that these models have consensus on capturing design decisions, their rationale, constraints on decisions and alternatives of decisions. We use 6 criteria, which are based on the desired properties introduced in [7], to compare the tools. The tools comparison demonstrates the major difference among the tools and what desired features are still missing in these tools. This work provides an initial reference for practitioners to evaluate and select appropriate ADD tool based on their architecting needs, context and challenges, and promote the ADD employment in architecting practices.

In the reminder of this paper, the ADD models are compared systematically in Section 2. Section 3 compares related tools that support the ADD models analyzed in section 2. We conclude our work with future work directions in Section 4.

2. Comparison of Existing ADD Models

In [12], we surveyed 9 ADD models and related tools. In Table 1 these models are compared so that their similarities and dissimilarities are clarified. The elements of each model are located in one row which expresses similar or identical concept; however they may use different terminologies. Two types of ADD concepts are classified: *Decision*, *Constraint*, *Solution* and *Rationale* are elements that all the ADD models have consensus on, and they are named major elements. *Problem*, *Group*, *Status*, *Dependency*, *Artifact*, *Consequence*, *Stakeholder* and *Phase*/*Iteration* are elements without consensus, called minor elements.

2.1 Major Elements

Decision. A *Decision* in Tyree' template, is equal to the concept *Architectural Design Decision* of core model and ADDSS, the *Solution* in pattern model, the *Architectural Decision* in SOAD, the *Design Decision* in AREL, and *Architecture Decision* in DAMSAK.

978-1-4244-4985-9/09/$25.00 ©2009 IEEE

Constraint. All the concepts like *Assumption*, *Constraint*, *Requirement*/*Defect*/*Risk*, *Context*, *Concern*, *Decision drivers and Quality Attribute,* and *Architectural Driver* state the limitations which influence architectural design and the choice of alternatives. These concepts are also equal to *Architecturally Significant Requirements* (ASR) and *Scenario* in DAMSAK.

Solution. The concept of *Position* in Tyree's template expresses alternatives for solving the design issue. This concept equals *architectural design decisions* in Kruchten's Ontology with the status of *Idea* or *Tentative*. This concept is equal to *Solution variants* in pattern model and also the *Potential Solution* in Archium and *Design option* in DAMSAK.

Rationale. *Argument* in Tyree's template expresses the "why" of the decision made which is equal to *Rationale* in pattern model, Kruchten's Ontology and DAMSAK and also to concepts *pros* and *cons* in Archium, and *justification* in SOAD model. In AREL, this concept is captured by *Qualitative* and *Quantitative design rationale*. The core model lacks explicit concept for recording the rationale of an ADD, but the rationale can be discovered by following the *decision loop* [5], in which the *Concern* and *Decision Topic* lead to the ranking of the *Alternatives* so that the *Decision* is chosen conditionally.

2.2 Minor Elements

Problem. The *Issue* concept in the Tyree's template equals the *Problem* in pattern model, *Problem Statement* in SOAD model, *Decision Topic* in core model, *Problem*, *Motivation* and *Cause* in Archium, *Requirement* in ADDSS, *Design Issue* in AREL, which all express stakeholders' problem to be solved and also the origins that motivated the decisions.

Group. The *Group* concept in Tyree's template is equal to the concept of *Category*, *ADtopic* and *Category of decision* in Kruchten's Ontology, pattern model, SOAD and ADDSS respectively. These concepts are all considered as a group of architectural decisions related with certain characteristic.

Status. The *Status* concept, expressing the evolution process of an ADD, equals the concept of *State* in Kruchten's Ontology, *Status* in SOAD and ADDSS model. As pattern is a type of general knowledge, it does not need the trait of *status*. There is no *status* concept in other ADD models, e.g. core model, while *Status* can be added as a property of *Decision*.

Dependency. The *Related Decision* concept in Tyree's template describes the dependencies between decisions. This concept is captured in Kruchten's Ontology by *Relationship*, in pattern model by *Related Pattern*, in SOAD model by *Dependency*, in ADDSS by *Dependencies* and in DASMAK by *Relationship*. In core model, these relationships are manifested in the *decision loop*. AREL can not capture the dependency between ADDs. It captures the dependency from architectural driver to decision, and from decision to design outcome.

Artifact. Besides relationships between decisions, Kruchten names several relationships with *external artifacts*. Design decisions can *trace from* technical artifacts upstream: *requirements* and *defects* (i.e. *Concerns* in core model), and *trace to* technical artifacts downstream: *design* and *implementation artifacts* (i.e. *Artifacts* in core model [5]). They are also traceable to management artifacts, such as *risks* and *plans* (again *Concerns*). Kruchten notes that it is useful to track which portions of the system are *not compliant with* some design decisions. In core model, this non-compliance corresponds to a reflection in *Artifacts* of an *Architectural Design* upon which some ADDs have not yet been enforced.

Consequence. A pattern's consequences state the results of applying a pattern. This concept corresponds to *Implications* concept in Tyree's template and *Consequence* in Archium. We can also get this concept in core model via composition of *Architectural Design Decision* and *Concern* concepts [8].

Stakeholder. The *Stakeholders* concept in ADDSS contains the list of stakeholders interested in particular view, and hence in a particular subset of design decisions. This concept corresponds to *Stakeholder* concept in core model and DASMAK.

Phase/**Iteration**. Architecture is constructed in an iterative process. In ADDSS and SOAD model, The *Iteration* and *Phase* element are used to record the architectural iteration/phase which the decisions belong to.

	Tyree Template	Kruchten's Ontology	Core Model	Pattern Model	SOAD	Archium	ADDSS	AREL	DAMSAK
Major Elements	Decision	Design Decision	Architectural Design Decision	Solution	Architectural Decision	Decision	Architectural Design Decision	Design Decision	Architecture Decision
	Assumptions, Constraints	Requirement, Defect, Risk	Concern	Context, Force	Decision drivers	Design rules, Design constraints	Constraints, Assumptions, Quality Attributes	Architectural Driver	Architecturally significant requirements, Quality Attributes, Scenario
	Positions	Idea, Tentative	Alternative	Solution Variants	AD Alternative	Potential Solutions	Alternative Decisions	Alternative Design	Design option
	Argument	Rationale	decision loop	Rationale	Rationale (Pros, Cons,	Rationale (Pros, Cons)	Rationale (Pros, Cons)	Qualitative and Quantitative	Rationale

					Justification)			Rationale	
Minor Elements	Issue		Decision Topic	Problem	Problem Statement	Problem (motivation, cause)	Requirements	Design Issue	
	Group	Category		Category	AD Topic		Category of Decisions		
	Status	State			Status		Status		
	Related Decision	Relationships	decision loop	Related Pattern	Dependency		Dependencies		Relationship
	Related Artifact	trace from/to	to reflect					forward and backward tracing	
	Artifact	Technical Artifact	Artifact		AD Outcome			Design element	
	Resulting Context, Consequence			Implications		Consequences			
	Stakeholder		Stakeholder				Stakeholders		Stakeholder
					Phase		Iteration		

Table 1. A Comparison of Existing ADD Models

This panoramic comparison shows that in many cases the ADD models employ different terms to express similar or identical concept. These models do have some common features: (1) treat architectural design as a decision making process; (2) all these models have consensus on capturing and documenting *rationale*, *constraints* and *alternatives* of decisions. However, different models have different coverage for capturing minor ADD elements.

3. Comparison of Tools Supporting the Existing ADD Models

In Table 1, the existing ADD models are compared with each other, and the similarities and dissimilarities among them were clarified. It is identified that these models have no consensus on minor ADD elements. The differences in naming, capturing and using minor ADD elements result in a substantial difference among the tools, which have been implemented based on various ADD models. Meanwhile, not all the ADD models are supported by tools. For example, the Tyree's template has no tool support. Practitioners use a text template for capturing and documenting ADD with Tyree's template. As mentioned in Section 1, we surveyed (available) tools about the 9 ADD models in [12]. In this section, we use 6 criteria to compare these tools. In Table 2, the tools that support the Kruchten's Ontology, core model, SOAD, Archium, ADDSS, AREL and DAMSAK are compared in details.

The desired properties proposed by Farenhorst et al. for ADD tools [7] are used as criteria to compare these tools, including stakeholder-specific content, easy manipulation of content, descriptive in nature, support for ADD codification, support for ADD personalization, and support for collaboration. In Table 2, the tools are compared with each other by desired properties, in which "+" means "support", and "-" represents "no support".

Stakeholder-specific content: Archium is designed for single user to create and maintain traceability between the decisions and the architectural design [6]. In ADDSS 2.0, multi-perspective is supported by *Stakeholders* and *Architectural Views* concept. In AD_{kwik}, it is possible to have multiple users with various roles and this tool provides "search function" for retrieving required information for specific stakeholders. In a large project, multiple stakeholders are involved in various architecting activities, and typically manage and maintain their part of the relevant ADD. The *Knowledge hub* feature is addressed in KA tool for gathering all the AK (including ADD) in one resource, and provides an interface for all involved stakeholders to mange and evolve it.

Easy manipulation of content: The language used in Archium suggests explicit support for addition and modification of ADDs. In ADDSS 2.0, user can easily modify the attributes of a design decision, which is not a simple task in ADDSS 1.0. KA tool suite, AD_{kwik}, Kruchten's Ontology tool and AREL tool support this property as well. In PAKME tool, a maintenance service provides various features to modify, delete and instantiate different ADD artifacts.

Descriptive in nature and ADD codification: All the tools have the character of "descriptive in nature". As a consequence, all these tools do not prescribe how architects should manage ADDs, for example, by offering an abundance of predefined models, guidelines or templates. This is also related to the fact that all the tools employ the codification approach for ADD management [11]. In codification strategy users are able to codify and store explicit ADD in document, (UML) models, or databases.

ADD personalization: Personalization means that users communicate the ADD with human instead of knowledge stored in a repository. Most of ADD tools do not support ADD personalization. PAKME is the only tool that supports personalization as it not only provides access to codified AK, but also identifies the source of knowledge [9]. That means it can also support a hybrid strategy (codification and personalization) for managing AK and ADDs.

Collaboration: AD_{kwik} and KA tool suite are designed and implemented as a collaboration system. These tools allow the stakeholders to collaborate with

each other. This feature is also supported in PAKME because PAKME is built on an open source groupware platform, which provides collaboration features by content management, project management etc.

Table 2 provides an overview of ADD tools, whose comparison result is neither complete nor perfect. The objective of this comparison is to provide an initial reference for ADD practitioners and researchers to evaluate and select appropriate ADD tool based on their architecting needs, context and challenges, and promote the ADD employment in architecting practices.

	Desired Properties					
Tools Supporting Existing ADD Models	Stakeholder specific content	Easy manipulation of content	Descriptive in nature	Support for ADD codification	Support for ADD personalization	Support for collaboration
Kruchten's Ontology Tool	-	+	+	+	-	-
Knowledge Architect (KA)	+	+	+	+	-	+
AD_{kwik}	+	+	+	+	-	+
Archium Tool	-	+	+	+	-	-
ADDSS 2.0	+	+	+	+	-	-
AREL Tool	-	+	+	+	-	-
PAKME Tool	-	+	+	+	+	+

Table 2. A Comparison of Tools Supporting Existing ADD Models

4. Conclusions and Future Work

The attention of software architecture community has changed over recent years and results in an increasing interest in ADD as one of key elements of the architecting process. The aforementioned change brings the models and related tools to capture, store, manage and share the AK, especially ADDs. We analyze and compare the existing ADD models and related tools in this paper. The main results of this study are: (1) all the ADD models treat the architectural design as a decision making process; (2) all the models have consensus on capturing and documenting rationale, constraints, and alternatives of decisions; (3) various ADD models tend to use different terms to express identical concept; (4) not all the ADD models are supported by tools, some of them use text templates for capturing and documenting ADDs; (5) ADD personalization is a desired feature that is currently missing in most of ADD tools.

Our ongoing and future work focuses on and is prioritized in following aspects: (1) the application and usage comparison of these ADD tools in practice and industrial environment (user satisfaction, cost and benefit, etc.); (2) the general guidelines on how to use ADD models and related tools in architecting process; and (3) the proposition of the general ADD framework that covers selected key features of existing ADD models.

5. References

[1] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice, 2nd edition*, Addison-Wesley Pearson Education, 2003.

[2] P. Kruchten, P. Lago and H. van Vliet, Building up and Reasoning about Architectural Knowledge. *Proceedings of the 2nd International Conference on the Quality of Software Architectures (QoSA)*, pp. 39-47, 2006.

[3] J.S. van der Ven, A. Jansen, J. Nijhuis, and J. Bosch, Design decisions: The Bridge between Rationale and Architecture, In *Rationale Management in Software Engineering*, A.H. Dutoit, et al., eds, Springer, pp. 329-346, 2006.

[4] A. Jansen and J. Bosch, Software Architecture as a Set of Architectural Design Decisions, *Proceedings of 5th Working IEEE/IFIP Conference on Software Architecture (WICSA)*, pp. 109-120, 2005.

[5] R.C. de Boer, R. Farenhorst, P. Lago, H. van Vliet, V. Clerc, and A. Jansen, Architectural Knowledge: Getting to the Core. *Proceedings of the 3rd International Conference on the Quality of Software Architectures (QoSA)*, pp. 197-214, 2007.

[6] A. Jansen, J.S. van der Ven, P. Avgeriou, and D.K. Hammer, Tool Support for Architectural Decisions. *Proceedings of the 6th IEEE/IFIP Working Conference on Software Architecture (WICSA)*, pp. 44-53, 2007.

[7] R. Farenhorst, P. Lago, and H. van Vliet, Effective Tool Support for Architectural Knowledge Sharing, *Proceedings of 5th European Conference on Software Architecture (ECSA)*, pp. 123-138, 2007.

[8] P. Liang, A. Jansen, and P. Avgeriou, Refinement to Griffin Core Model and Model Mapping for Architectural Knowledge Sharing, *Technical Report RUG-SEARCH-07-L01*, University of Groningen, 2007. http://www.cs.rug.nl/~liangp/download/liang2007rgc.pdf

[9] M.A. Babar and I. Gorton, A Tool for Managing Software Architecture Knowledge, *Proceedings of the 2nd Workshop on SHAring and Reusing architectural Knowledge Architecture, Rationale, and Design Intent (SHARK/ADI)*, pp. 11-17, 2007.

[10] M.A. Babar, T. Dingsøyr, P. Lago, and H. van Vliet, *Software Architecture Knowledge Management: Theory and Practice*, Springer, 2009.

[11] M.A. Babar, R.C. de Boer, T. Dingsøyr, and R. Farenhorst, Architectural Knowledge Management Strategies: Approaches in Research and Industry, *Proceedings of the 2nd Workshop on SHAring and Reusing architectural Knowledge Architecture, Rationale, and Design Intent (SHARK/ADI)*, pp. 2-8, 2007.

[12] M. Shahin, P. Liang, and M.R. Khayyambashi, A Survey of Architectural Design Decision Models and Tools, *Technical Report SBU-RUG-2009-SL-01*, Sheikh Bahaei University & University of Groningen, 2009. http://www.cs.rug.nl/~liangp/download/shahin2009sad.pdf