Guidelines and Metrics for Configurable and Sustainable Architectural Knowledge Modelling

Carlos Carrillo
ETSIST-DTE
Universidad Politécnica de Madrid
Madrid, Spain
carlos.carrillo@upm.es

Rafael Capilla
Faculty of Computer Science
Rey Juan Carlos University
Madrid, Spain
rafael.capilla@urjc.es

Olaf Zimmermann
University of Applied Sciences of Eastern Switzerland
Rapperswil, Switzerland
ozimmerm@hsr.ch

Uwe Zdun
Faculty of Computer Science
University of Vienna
Vienna, Austria
uwe.zdun@univie.ac.at

ABSTRACT

Architectural Knowledge Management (AKM) has been an active research area in the last decade; the importance of making the right architectural decisions – and making these at the right time – has been recognized by the contemporary software engineering practices. Several AKM meta-models, templates and tools have been proposed and applied in practice to capture architectural design decisions and minimize architectural drift during software evolution. However, most of these AKM models, and the architectural decisions captured with them, lack contextual awareness, flexibility and maintainability over time. In this position paper, we outline an extended AKM meta-model and a set of guidelines with the goal to (i) allow AKM tool engineers to construct more configurable and therefore flexible AKM tools, (ii) allow knowledge engineers and method coaches to create more sustainable and therefore maintainable decision logs (AK model instances). We approach these two goals by way of mapping the extended AKM meta-model concepts to quality attributes for architectural knowledge as well as supporting AK metrics.

Categories and Subject Descriptors

D.2.11 [**Software Architectures**]

General Terms

Management, Documentation, Design.

Keywords

Software architecture, architecture erosion, architecture knowledge, technical sustainability, technical debt.

1. INTRODUCTION

Architectural drift is a major concern in avoiding architecture erosion during software evolution, i.e., the problem that changes are applied in a system, but not reflected in the design [1]. Architectural Knowledge (AK) often vaporizes if architecturally significant design decisions are not recorded; hence, it is important to capture such design decision rationale along with the actual designs. Since 2004, the problem of AK vaporization has been widely recognized and addressed by the software architecture community [2] [3].

ECSAW '15, September 07 - 11, 2015, Dubrovnik/Cavtat, Croatia
© 2015 ACM. ISBN 978-1-4503-3393-1/15/09…$15.00
DOI: http://dx.doi.org/10.1145/2797433.2797498

Preserving AK and capturing architectural decisions explicitly brings about new problems – this AK has to be maintained and evolved over time along with the architectures and their implementations.

There has been significant AKM research in the form of meta-models, methods, templates and tools [4] to incorporate this knowledge into already existing views on software architecture [5]. However, the majority of these approaches suffer from an inadequate level of rigidity, e.g., if the knowledge is captured in certain predefined static templates [6]. The AK captured in such inflexible templates is sometimes difficult to maintain and evolve. Moreover, as good design decisions endure over time, it becomes relevant to estimate the longevity of good design decisions and identify which of them remain more stable over time as a way to achieve architecture sustainability. Our main contributions of this paper is a configurable meta-model to achieve AK sustainability and a set of criteria that suggest ways to estimate the technical sustainability of AK.

The remainder of this paper is structured as follows. In Section 2 we describe related work. Section 3 sketches the building blocks for a flexible and configurable AK meta-model, while in Section 4 we outline a set of guidelines (criteria) to achieve architecture sustainability via AKM and suggest a set of mappings for each sustainability criterion to quality attributes (QAs) and AKM metrics. Finally, in Section 5 we draw our conclusions and identify future work.

2. RELATED WORK

In the past eleven years, a significant body of research has been produced in the research area of AKM. Many of these efforts are summarized in [4]. All these initial attempts use rigid templates for capturing the relevant AK and define a big number of dependencies between design decisions and other software artifacts which may complicate maintenance tasks. Only the Architecture Design Decision Management (ADDM) tool [7] seems not tied to any particular meta-model, as it offers a customization mechanism that can be adapted for different users and personalization of AK. Since 2011, new research efforts have brought about new models and tools that extend the capabilities of the first generation of AK tools. Among these efforts we can highlight the ADDMM model described in [8] as a meta-model with focus on evolution of design decisions and bidirectional traceability between decisions and other software artifacts which the authors use to evaluate the impact in the evolution of design decisions. The work described in [9] provides fine-grained trace links between design decisions, constraints, requirements and other elements of software systems. The authors introduce "impact-relations" to specify the impact of design decisions over other software artifacts. Other AK tools and models like ADvISE [10] and SAW [11] offer support for reusable decisions and collaborative aspects respectively. A set of tags in the SAW tool allow for configurable models to visualize better the knowledge captured. Finally, the ADMentor approach [12] suggests a flexible, template-based decision backlog that can be configured for knowledge capturing and sharing based on predefined, but extensible meta-information. This last approach is one of the few AK models that suggest flexible and configurable ways for AK. Consequently, there is still a challenge to produce more flexible and configurable AK models able to yield and manage sustainable AK, address more explicitly the longevity of the design decisions, and provide criteria to maintain this AK in a sustainable fashion in the long term. We address these issues in the following sections.

3. A FLEXIBLE AND CONFIGURABLE AK META-MODEL

In this position paper we suggest a new flexible and configurable AK model to address the rigidity of previous approaches. Our work is based on our background and previous experiences as creators and users of AK tools (i.e. the ADDSS and ADvISE AKM tools) and based on a previous work [13] where we build an AK meta-model with the following extended capabilities: (i) fine grained links between design decisions and other software artifacts, (ii) decision history and evolution, and (iii) explicit support for runtime decisions.

3.1 Drivers for sustainable AK

The notion of sustainability of AK and technical sustainability for software architecture described in our previous work [14] was the inspiration to produce a highly configurable and adaptable meta-model that: (i) promotes the sustainability of the AK captured, (ii) helps to improve the longevity of good decisions and corresponding architectures, and (iii) provides extensible and flexible capabilities for a new generation of AK tools avoiding rigid templates for capturing AK. These three drivers are the main reasons for the new AK meta-model that we will sketch in the following subsection.

3.2 A sustainable AK model

Based on our analysis of previous AK models we synthesize in this approach our view for a sustainable and configurable new AK model. The architecture part based on the ISO/IEC/IEEE 42010 standard [3] is illustrated in Figure 1 and comprises a reduced but enough elements derived from the standard and supporting the design artifacts, stakeholders, architectural viewpoints, and requirements that are commonly uses in any design process. Let us highlight the novel parts of our approach:

A)- Architecture and decision models: Figure 1 describes the architecture and decision models of our proposed meta-model. The decision model is represented by the `DD Core` package, as a minimalistic approach for capturing design decisions (based on [14]), as we only capture a reduced set of AK items (`Design`

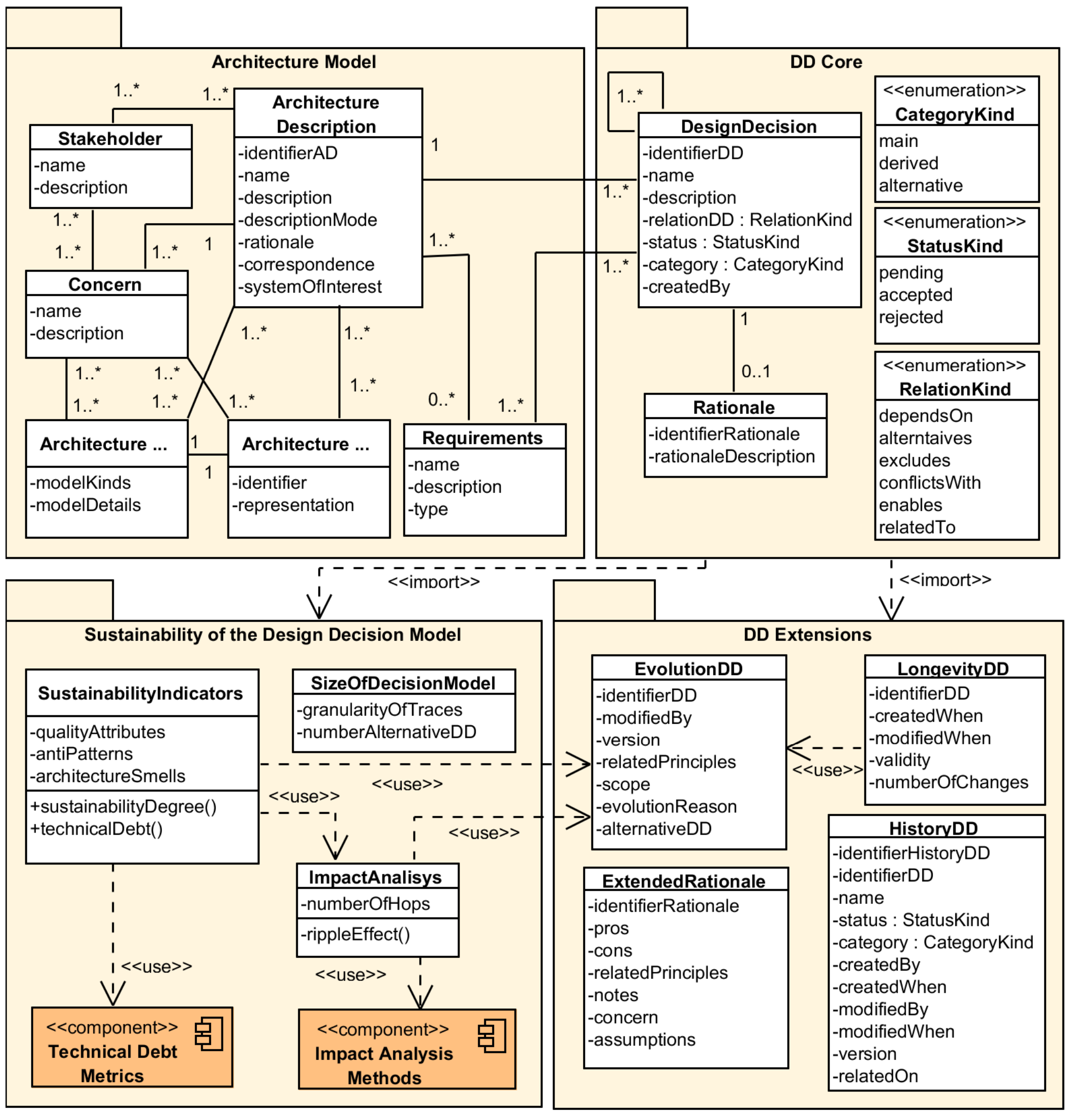

Figure 1. A configurable meta-model for architectural knowledge management.

Decision and Rationale classes). In addition, the rationaleDescription attribute describes the underpinning reasons of the design decisions while the other three classes are enumeration lists containing the allowed values for the attributes relationDD, status and category of the design decisions.

B)- Extensions to design decisions model: In our configurable AK meta-model we define now extensions to the classes supporting the design rationale that in previous models are all fixed. Hence, we enable AK tool designers to capture a minimal set of design rationales and provide extensions that can be captured or not. This makes the meta-model highly configurable and flexible and hence enables us to achieve more sustainable AK. Figure 1 depicts the DD Extensions package which encompasses a set of optional classes that any AKM tool can implement in a non-mandatory way. We give to AKM tool builders the freedom to support the optional classes defined in this part of the model, but we do not prescribe the attributes of these classes that must be supported by such AKM tools.

We apply the same criterion for the ExtendedRationale class as the number of AK items may vary from one project to another. Finally, in order to measure better the longevity of decisions and the stability of the architecture, we define the LongevityDD class and a validity attribute to set a date that indicates when a decision is valid or must be revised, while the numberOfChanges attribute indicates the number of times a decision can change.

C)- Sustainability of the decision model: In the figure we describe the last part of our configurable AK meta-model which pertains to those classes aimed to measure and provide indicators about the sustainability of the decision network and to estimate better the impact analysis of changes.

This package defines a configurable class named SizeOfDecisionModel where we configure the size of the decision space. The granularityOfTraces attribute specifies if the trace links from decisions to other software artifacts are established for subsystems, packages or classes, while fine-grained links (e.g., a decision that motivates the creation of an attributed) are not considered. In addition, the size of the decision model is limited by the numberAlternativeDD attribute. The rest of the classes address the sustainability of the AK and architecture as we provide explicit ways to measure the ImpactAnalysis class using an external <<component> that may contain any existing method commonly used to estimate the impact of changes or the ripple effect of a decision. The numberOfHops attribute is used to configure the number of related decisions that will be analyzed during the estimation of the ripple effect. Finally, the SustainabilityIndicators class defines attributes and methods to estimate the quality of the decisions based on anti-patterns or architecture smells identified that can be measured using technical debt metrics. We use the proposed configurable meta-model to define in next section a set of criteria to achieve AK sustainability.

4. CRITERIA FOR AK SUSTAINABILITY

In order to estimate the sustainability of the AK and consequently the architecture, we need to decide which QAs to measure and which metrics can be used. Hence, we define first a set of criteria to achieve this sustainability for the different parts of the AK meta-model and we suggest some metrics.

4.1 Sustainability criteria

From our experience with Service-oriented Architecture (SOA) decisions for IT services we derived a number of principles and quality attributes used in industrial AKM projects [15]. Nevertheless, evaluating the sustainability of the AK require a renewed set of guidelines able to guide the designer on which QAs and metrics can be used to achieve AK sustainability. If we consider the decisions network as a set of interrelated nodes (i.e., the decisions) and edges (i.e. the trace links among decisions), we suggest in the work the following seven criteria able to estimate the sustainability of AK models.

Criterion 1. Granularity of the design decisions: This criterion limits the granularity of the decisions to be captured at the level of classes, and avoids finer-grained decisions as a way to reduce the size of the decision model and make it more manageable. For example, if a design decision involves the creation of a UML class in the logical component architecture, such decision and their corresponding trace links will be captured. However, those decisions involved in the creation of a UML attribute or method for a particular class will not be recorded. In cases where other cross-cutting concerns may affect lower-level elements, the designer can configure the granularity to finer-grain levels and tailor this item to different projects or specific needs. We do not prescribe or set a specific granularity level, but rather we offer it as a configurable AK capturing option (within the context of method *tailoring* [12]).

Criterion 2. Size of the decision model: With this criterion we limit the number of design choices. We use the attribute numberAltenativeDD belonging to the class SizeOfDecisionModel described in Figure 1. At this stage of this research we cannot set a specific limit but from our experience using AK tools, we have observed a range of [1:7/10] alternative decisions.

Criterion 3. Number of attributes captured: Making the decision space more manageable implies capturing less amount of information. Figure 1 provides mechanisms to capture a minimal set of attributes in the DD Core package that can be extended if needed using the classes and attributes of the DD Extension package defined in Figure 1.

Criterion 4. Granularity of the trace links: Similarly to Criterion 1, we delimit the number of trace links between different software artifacts, from decisions to requirements and to classes, and we avoid fine-grained trace links such as those between decisions and attributes or methods.

Criterion 5. Number of decisions impacted: When a decision changes there is an impact on the related decisions. With this criterion we reduce the necessity to evaluate decisions that are less relevant for a decision that changes. We define such restriction using the attribute number_of_Hops in the entity Impact Analysis. A similar argument like in criterion 1 can be said for the limit of related decisions a ripple effect algorithm must explore. In many cases this limit is set by the designer but we need to carry out some experimentation before suggesting a number of decisions that will be worthy to explore when a decision changes.

Criterion 6. Number of times a decision changes: We use the attribute numberOfChanges defined in the LongevityDD class to measure how many times a decision can change and also the interval of changes of each decision using the attributes createdWhen and modifiedWhen.

Thereby we estimate how often a decision is modified and analyze better the longevity of decisions.

Criterion 7. Validity of decisions: One first attempt to estimate the lifetime of decisions can be found in [16]. In this previous approach, the `validity` attribute defined in the `LongevityDD` class is used to set the date until when a decision is valid and must be reviewed. Hence, obsolete decisions can be removed from other analyses. Sometimes, long-living systems suffer from technology obsolesce, and decisions that were valid at a particular time in the past are no longer valid after a long period and become obsolete. Therefore, we use this attribute to allow designers to set a specific date where decisions should be revisited; if a decision is considered obsolete it can be removed and possibly replaced by a new decision.

4.2 Relating quality and sustainability

Finally, for each criterion we suggest in this section a list of potential quality attributes that can be address and metrics that can be used to estimate each criterion, such as Table 1 shows. As work in progress, we only select some quality attributes that be believe meet each criterion.

Table 1. Criteria and quality factors to estimate sustainability of AK models

Criterion (C)	Quality Attributes (QAs)	Metrics
C1.Granularity of the design decisions	Complexity: The granularity of the design decisions, viewed as a graph of nodes, is reduced as we can limit the number of nodes in the network. Cost of the effort capturing the decisions	NodeCount [17] Cost metrics are not defined (N/D) yet
C2.Size of the decision model	Complexity, Cost of the effort capturing the decisions	Number of Children [18] Cost is N/D
C3.Number of attributes captured	Cost of the effort capturing a number of variable attributes	Cost is N/D
C4.Granularity of the trace links	Complexity and Cost to maintain the trace links	NodeCount EdgeCount [17]
C5.Number of decisions impacted	Changeability: We can reduce the amount of effort to change the decisions impacted by a change limiting the number of decisions analyzed.	Change Impact Analysis [19]
C6.Number of times a decision changes	Changeability Stability: If a decision changes less number of times it affects to the stability of the architecture, as good decisions endure over time.	Decision Volatility [19]
C7.Validity of the decisions	Changeability, Stability, Timeliness	Decision Volatility [19]

The rationale for this initial selection was based on: (i) the impact of the QA for each criterion and the items measured, and (ii) representative metrics available to evaluate such criteria, except for cost. With regard to the completeness of the table, more quality attributes can be added, but this will be an outcome of evaluating more metrics for each criterion. As we can consider technical sustainability as a combination of *maintainability* and *evolvability*, we didn't add these attributes in the seven criteria because we assume they can be computed as a combination of other QAs defined for each criterion. This is why these two QAs are factored out of the table. However, in this position paper we do not provide such formulas to compute both QAs yet as we leave this for future work.

5. CONCLUSIONS

In the ongoing research presented in this paper, we emphasize the role of configurable Architectural Knowledge (AK) meta-models to promote capturing and using AK in a more flexible and sustainable way than in previous works. As a first attempt towards this goal, we proposed seven AK sustainability criteria, related quality attributes and supporting metrics. We still need to investigate whether additional metrics will be required; other quality attributes may also have to be analyzed to provide better estimations about these sustainability indicators. Hence, future work is twofold: (i) define and evaluate metrics tailored for AK and as well as estimations of the cost of capturing this AK, and (ii) build a configurable AK tool supporting this approach. We plan reengineering one or more of the already existing AKM tools in order to add the features presented in this paper. We expect to validate the adequacy and practicality of our approach capturing relevant design decisions in both agile and non-agile industry projects against the proposed criteria.

6. REFERENCES

[1] J. van Gurp, J. Bosch, S. Brinkkemper: Design Erosion in Evolving Software Products, 2006.

[2] J. Bosch, Software Architecture: The Next Step, Proceedings of the 1st European Workshop on Software Architecture (EWSA 2004), Springer-Verlag, LNCS 3047, 194-199, 2004.

[3] ISO/IEC/IEEE 42010:2011 Systems and Software Engineering — Architectural Description, First edition, IEEE, 2011.

[4] A. Tang, P. Avgeriou, A. Jansen, R. Capilla, M. Ali Babar, A Comparative Study of Architecture Knowledge Management Tools, Journal of Systems and Software 83(3), 352-370, 2010.

[5] P. Kruchten, R, Capilla, J.C. Dueñas, The Decision's View Role in Software Architecture Practice. IEEE Software, 26(2), 36-42, 2009.

[6] J. Tyree, A. Akerman, Architecture Decisions: Demystifying Architecture. IEEE Software 22(2), 19-27, 2005.

[7] L. Chen, M. A. Babar, H. Liang, Model-centered customizable architectural design decisions management, 21st Australian Software Engineering Conference (ASWEC), 23-32, 2010.

[8] I. Malavolta, H. Muccini, V. Smrithi Rekha, Supporting Architectural Design Decisions Evolution through Model Driven Engineering. Software Engineering for Resilient Systems (SERENE), 63-77, 2011.

[9] S. Gerdes, S. Lehnert, M. Riebisch, Combining Architectural Design Decisions and Legacy System Evolution. 8th European Conference on Software Architecture (ECSA), 50-57, 2014.

[10] I. Lytra, H. Tran, U. Zdun, Supporting Consistency between Architectural Design Decisions and Component Models through Reusable Architectural Knowledge Transformations. 7th European Conference on Software Architecture (ECSA), 224-239, 2013.

[11] M. Nowak, C. Pautasso, Team Situational Awareness and Architectural Decision Making with the Software Architecture Warehouse. 7th European Conference on Software Architecture (ECSA), 146-161, 2013.

[12] O. Zimmermann, L. Wegmann, H. Koziolek, T. Goldschmidt, Architectural Decision Guidance across Projects. 11th Working IEEE/IFP Conference on Software Architecture (WICSA), 2015.

[13] R. Capilla, O. Zimmermann, U. Zdun, P. Avgeriou, J.M. Küster, An Enhanced Architectural Knowledge Meta-model Linking Architectural Design Decisions to other Artifacts in the Software Engineering Lifecycle. 5th European Conference on Software Architecture (ECSA), 303-318, 2011.

[14] U. Zdun, R. Capilla, H. Tran, O. Zimmermann, Sustainable Architectural Design Decisions. IEEE Software 30(6), 46-53, 2013.

[15] O. Zimmermann, C. Miksovic, J.M. Küster, Reference architecture, metamodel, and modeling principles for architectural knowledge management in information technology services. Journal of Systems and Software 85(9): 2014-2033 (2012.

[16] R. Capilla, F. Nava, A. Tang, Attributes for Characterizing the Evolution of Architectural Design Decisions, 3rd International IEEE Workshop on Software Evolvability, IEEE CS, 15-22, 2007.

[17] T. Mens, M.Lanza, A Graph-Based Metamodel for Object-Oriented Software Metrics. Electronic Notes in Theoritical Computer Science 72(2), 2002.

[18] S. Sarkar, G.M. Rama, A.C. Kak, Api-Based and Information-Theoretic Metrics for Measuring the Quality of Software Modularization. IEEE TSE, 33, 14-32, 2007.

[19] K. Sethi, Y. Cai, S. Wong, A. Garcia, C. Sant'Anna, From Retrospect to Prospect: Assessing Modularity and Stability from Software Architecture Joint Working IEEE/IFIP Conference on Software Architecture, 269-272, 2009.