Towards a framework for managing architectural design decisions

Manoj Bhat
Technische Universität München
Boltzmannstr. 3
Garching, Germany 85748
manoj.mahabaleshwar@tum.de

Klym Shumaiev
Technische Universität München
Boltzmannstr. 3
Garching, Germany 85748
klym.shumaiev@tum.de

Florian Matthes
Technische Universität München
Boltzmannstr. 3
Garching, Germany 85748
matthes@tum.de

ABSTRACT

Software architecture is considered as a set of architectural design decisions. The recent trends, both in research and industry, call for improved tool support for software architects and developers to manage architectural design decisions and its associated concepts. As part of our ongoing work, we propose a framework for managing architectural design decisions in large software-intensive projects. Each component within this framework addresses specific use cases including (a) extraction and classification of design decisions from issue management systems, (b) annotation of architectural elements, (c) recommendation of alternative decision options, (d) reasoning about decisions' rationale, and (e) recommendation of experts for addressing design decisions. These components are planned to be iteratively realized and evaluated using the design science research approach. We believe that the realization of such a framework will allow an architectural knowledge management systems to integrate with the design, development, and maintenance phases to support stakeholders not only to document design decisions but also to learn from decisions made in the past projects.

CCS CONCEPTS

• **Software and its engineering → Software architectures**; **Designing software**; • **Information systems → Decision support systems**;

KEYWORDS

Architectural design decisions, Software architecture, Framework

ACM Reference format:
Manoj Bhat, Klym Shumaiev, and Florian Matthes. 2017. Towards a framework for managing architectural design decisions. In *Proceedings of European Conference on Software Architecture, Canterbury, UK, September 2017 (ECSA'17),* 4 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Over the past decade, there has been a paradigm shift in the representation of software architecture. Instead of focusing on the result of the design process, that is, software design as an artifact, software architecture is considered as a set of architectural design decisions (ADDs) [6, 18] which leads to the creation of a software design. This perspective can be considered as one of the key advancements in software architecture research because design decisions represent the rationale that motivates the selection of architectural elements within the software design [8]. The representation of design decision as a first-class entity in the software architecture meta-model and realizing the fact that software design is a complex decision-making process has engaged researchers in borrowing and incorporating ideas related to *how people make decisions* from philosophy, cognitive science, and sociology. There has been a significant effort in understanding the decision-making process in software architecture (*How design decisions are made?*) which includes addressing research questions such as *What is a design decision?*, *Why was a design decision made?*, *When is a design decision taken?*, and *Who makes a design decision?* [9, 25]. Furthermore, there have been noteworthy tool-support contributions for systematically capturing design decisions to enable use cases such as reuse, reasoning, and recommendations related to design decisions [2, 7]. However, research addressing this topic is still in the early phase of supporting software architects and developers during the design decision-making process [9, 22, 26].

In this research endeavor, we aim to address some of the aforementioned challenges related to ADDs by providing tool support that could be integrated within software architects' and developers' working environment to (a) extract and reuse knowledge related to similar design decisions made in past projects, (b) recommend alternative architectural solutions that could be considered while making design decisions, (c) highlight and reason about the rationale behind design decisions, and (d) recommend experts to address specific design decisions. In this paper, we present how these use cases can be addressed by realizing the independent components within the proposed framework.

Since a major focus of our work lies in building components that learn from design decisions made in past projects for providing automation support, we emphasize on medium- and realization-level decisions (cf. ADDs are made at different abstraction levels [11, 23]). These design decisions are less likely to be influenced by projects' business and political aspects and are typically captured in disparate agile project management systems. Furthermore, we make the following assumptions based on the past empirical studies:

(1) Software architects (SAs) do not take a rationalistic approach but favor a naturalistic approach to decision making [22, 25].
(2) SAs make architectural decisions based on their expertise and past experiences [22, 24].

(3) ADDs can be associated with quantifiable attributes such as risk, cost, and time [12].
(4) The current state of the project and past project artifacts are observable given a predefined domain model [3].

The contribution of our work will aid software architects and developers (trainees, junior and newly recruited architects) in the industry, first, to get accustomed to the assigned long-running software projects by exploring and understanding design decisions that were already made to address specific quality concerns, second, to learn from design decisions that were made by other architects in similar past projects, and finally, to encourage the culture of explicitly documenting or even labeling design decisions by highlighting its benefits with regards to traceability and impact analysis during the development and maintenance of software systems.

2 RESEARCH METHODOLOGY AND OBJECTIVES

The long-term objective of this work is to develop a framework that tracks the current state of a project, extracts information from past project artifacts, and recommends stakeholders on how to address specific architectural concerns. Each component within the framework that addresses specific research problems follows the three Design Science Research Cycles proposed by Hevner [10]. As part of the *relevance cycle*, the requirements for each component is derived from interactions with our industry partner's software architects in a large (approximately 20-30 architects) Architecture Definition and Management group as well as from the use cases and future research directions prescribed in literature (for e.g., [7] and [15]). Furthermore, we conduct the evaluation of each of the components by integrating them within an architecture knowledge management (AKM) system named Architecture Management Enabler for Leading Industrial softwarE (AMELIE)[1]. Research associates and students studying computer science in our research department implement each component within the framework (*design cycle*). During the design cycle, we ground our design artifacts on well-established meta-models (for e.g., [1], [27], and [14]) and successfully implemented and tested technological concepts (for e.g., [19]) and share the lessons learned with the software architecture research community as part of the *rigor cycle*. The high-level overview of the framework is shown in Figure 1. We briefly describe the components within this framework in the following subsections.

2.1 AKM system

Understandably, the core of the framework is an AKM system that allows architects to manage the different categories (context, design, general, and reasoning) of AK in software-intensive projects (cf. [21] and [3]). In particular, we prescribe the use of a meta-model based AKM system that provides the flexibility to configure the domain models at runtime to meet the dynamic needs of diverse projects in different organizations in order to capture the static and dynamic aspects of AK. To this end, we have proposed a meta-model based AKM system named AMELIE that not only allows architects to capture AK but also provides reasoning capabilities using a model-based expression language. The detailed description along with the need for a meta-model, the configurable domain models and the support for domain-specific rules has been elaborated in [3]. Furthermore, in this work, we have demonstrated how to link the peripheral concepts related to ADDs including requirements, stakeholders, and architectural elements using a domain model. The extensible architecture of this system allows us to realize the components including decision classifier, document annotator, and recommendation engine as external reusable services.

[1]AMELIE is currently being iteratively developed and evaluated by our industry partner's business units.

2.2 SyncPipes

Empirical studies indicate that even though design decisions are not explicitly documented, they are implicitly captured in different systems including project management, issue management, source code version management, and meeting recording systems [16, 23]. The AKM system under consideration must be able to monitor and integrate with systems that are used by architects and developers to manage their day-to-day activities and hence seamlessly integrate within the working environment of stakeholders. To address this challenge, we developed a software platform – SyncPipes[2] – to extract and synchronize information from disparate systems that maintain project related information into the knowledge base (KB) of the AKM system. This research project was carried out as part of a Bachelor's Thesis and the research questions, the approach, and the evaluation are elaborated in [13]. SyncPipes is analogous to a "bot" with sensors and actuators wherein the sensors monitor the current state of the project and actuators keep the information within the KB synchronized. The current version of SyncPipes supports the integration of projects' artifacts captured in Excel, MS Project, Enterprise Architect, and JIRA with the AKM system.

2.3 Decision classifier

Design decisions are often not explicitly documented. However, there are sources (for e.g. issues in issue management systems) that implicitly maintain design decisions. These issues are extracted using the SyncPipes component, which are then automatically labeled as design decisions and are also classified into different decision categories including structural, behavioral, and non-existence decisions. In our recent work [5], we successfully confirmed the feasibility of automatically detecting design decisions from issues captured in issue management systems. We demonstrated that supervised machine learning algorithms, in particular, the Support Vector Machine (SVM) classifier can detect design decisions with a high accuracy (F-score) of 91.29%[3].

This component addresses the use case of *identifying design decisions made by architects and developers in the past as well as in the current software-intensive projects*. This component not only allows us to capture design decisions in our AKM system to avoid knowledge vaporization but also supports the use cases such as building expertise profile and identifying the relationships between design decisions discussed in the subsequent subsections.

[2]https://wwwmatthes.in.tum.de/pages/2gh0u9d1afap/SyncPipes
[3]It should be noted that the observation is based on a dataset comprising of 1,571 manually labeled issues from two large open source projects.

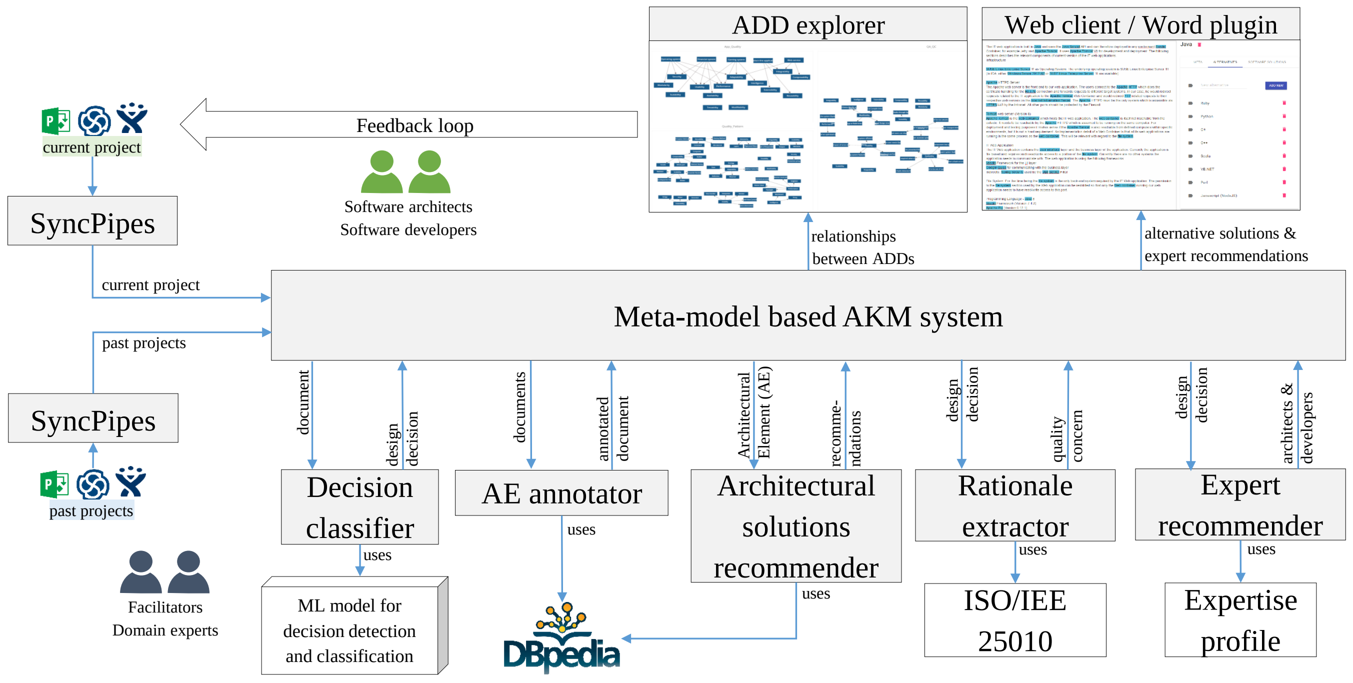

Figure 1: A framework for managing architectural design decisions

2.4 Architectural elements annotator

Within this component, we address the challenge of "*How to automatically identify and annotate architectural elements (AE) in textual documents?*". The AE annotator component consumes textual description (design decisions or software architecture documents) and highlights AEs within the text. AEs include architectural styles, methods, design patterns, and software and hardware systems. To automatically identify AEs within the given text, we propose a novel approach to use a publicly available cross-domain ontology (DBpedia ontology[4]). In our previous work [4], we illustrated the feasibility of identifying AEs using software architecture documents from four large industrial projects with an accuracy (F-score) of 84%. This annotator apart from identifying AEs, also enriches them with meta-information such as overview, programming language, and license retrieved from the DBpedia ontology.

2.5 Architectural solutions recommender

The architectural solutions recommender component also follows an ontology-based approach that uses the knowledge in the DBpedia ontology to *recommend alternative architectural solutions and software solutions for realizing an ADD.* This component executes SPARQL queries to generate recommendations and presents the results to architects using a web client[5] as well as a Microsoft Word plug-in. It should be noted that we do not contradict the fact that "the idea that we start out with **all** the alternatives and then choose among them is wholly **unrealistic** [20]". However, we believe that by presenting even a subset of alternative solutions to architects while they are documenting software architecture will encourage architects to reason about their design decisions. This approach is elaborately described in [4].

2.6 Rationale extractor

The focus of the rationale extractor component is to address the aspect of "*Why was a design decision made?*". This component consumes the design decisions identified by the Decision classifier component and then automatically maps the design decisions to the quality concerns (ISO/IEC 25010 standard) of software systems using natural language processing (NLP) techniques. The realization and the evaluation of this component are currently being carried out by a student as part of his Master's Thesis.

2.7 Expert recommender

Through this component, we aim to address the following research question: "*Who should be involved in making a design decision?*". Several empirical studies indicate that software architects do not always take rationalistic approach but favor a naturalistic approach to decision making (cf. Section 1 assumption 1). In this context, the personal experience of architects influences the design decision-making process (cf. Section 1 assumption 2). Based on the aforementioned assumptions, we build an expertise profile using the output of previously discussed components namely decision classifier and AE annotator component. We build this approach by reusing the concepts and terminology (experience atoms and expertise profile) proposed by Mockus and Herbsleb [17]. An expertise profile can be imagined as a matrix where rows correspond to architects and columns capture AEs. The cells within this matrix contain integer values that represent the architectural experience of an architect,

[4]http://wiki.dbpedia.org/services-resources/ontology
[5]https://amelietor-9f8c3.firebaseapp.com/

which is computed by analyzing the design decisions that were addressed by the respective architect. Such an expertise profile allows us to quantitatively measure the experience of architects. Therefore, for a new design decision identified by the decision classifier component, first, the AEs within the text are extracted by the AEs annotator component to generate a concept vector. Second, the concept vector is compared against the expertise profile to identify those architects who have addressed similar design decisions in the past. This component is currently being integrated with the AKM system to evaluate it with our industry partner.

2.8 ADD explorer

The focus of this component is to extract and visualize the relationships between design decisions that are automatically classified by the decision classifier component. To represent the relationships between design decisions, we plan to use the formal model of design decisions proposed by Zimmermann et al. [27] within our meta-model based AKM system. These relationships for instance include *decomposesInto*, *isCompatibleWith*, and *isInCompatibleWith* relations. Instantiating this model using data from past projects will allow us to explore the peripheral aspects of design decisions including traceability and complexity (based on assumption 3 in Section 1) for addressing similar design decisions.

3 CONCLUSION AND OUTLOOK

In this paper, we have presented our ongoing research and our vision of a framework for supporting architects during the decision-making process. Even though we have realized and partially evaluated some of the components including a meta-model based AKM system, Decision classier, AE annotator, and Architectural solution recommender within our framework, we are still in the initial iterative phase of design science cycle for the Rationale extractor, Expert recommender, and ADD explorer components.

One of the main factors that differentiate the proposed framework from the existing AKM tools [7] is that the existing AKM tools follow a top-down approach, that is, they expect domain experts to manually capture data within AKM tools to support use cases such as traceability and reasoning. Whereas, the proposed framework follows a bottom-up approach, wherein, the AKM system learns from design decisions made in the past by extracting information from tools (for e.g., MS Project, Enterprise Architect, and JIRA) and uses the knowledge captured in cross-domain ontologies without explicitly depending on the need to populate the knowledge base to generate recommendations. These recommendations are presented within tools (for e.g., Microsoft Word) that architects use for managing their day-to-day activities. We believe that integrating the AKM services into the software development and agile project management systems is critical for the improved adoption of AKM systems and also for encouraging stakeholders to reason and document their design decisions. Such a system realized based on the proposed framework will not only provide automation and tool support for AKM but will also allow us to quantitatively address research questions including "*Who should make design decisions?*", "*Why and what design decisions were made in the past?*" and "*What is the complexity of realizing a design decision?*".

REFERENCES

[1] 2011. ISO/IEC/IEEE Systems and software engineering – Architecture description. *ISO/IEC/IEEE 42010:2011(E)* (Dec 2011), 1–46. https://doi.org/10.1109/IEEESTD.2011.6129467
[2] Muhammad Ali Babar, Torgeir Dingsøyr, Patricia Lago, and Hans van Vliet. 2009. *Software architecture knowledge management.* Springer.
[3] Manoj Bhat, Klym Shumaiev, Andreas Biesdorf, Uwe Hohenstein, Michael Hassel, and Florian Matthes. 2016. Meta-model based framework for architectural knowledge management. In *Proc. of the 10th European Conf. on Software Architecture Workshops.* ACM, 12.
[4] Manoj Bhat, Klym Shumaiev, Andreas Biesdorf, Uwe Hohenstein, Michael Hassel, and Florian Matthes. 2017. An ontology-based approach for software architecture recommendations. In *23rd Americas Conf. on Information Systems, AMCIS 2017, Boston, MA, USA, August 10-12, 2017.* http://aisel.aisnet.org/amcis2017/SemanticsIS/Presentations/7
[5] Manoj Bhat, Klym Shumaiev, Andreas Biesdorf, Uwe Hohenstein, and Florian Matthes. 2017. Automatic extraction of design decisions from issue management systems: a machine learning based approach. In *Proc. of the 11th European Conf. on Software Architecture Workshops.* ACM.
[6] Jan Bosch. 2004. Software architecture: The next step. In *European Workshop on Software Architecture.* Springer, 194–199.
[7] Rafael Capilla, Anton Jansen, Antony Tang, Paris Avgeriou, and Muhammad Ali Babar. 2016. 10 years of software architecture knowledge management: Practice and future. *Journal of Systems and Software* 116 (2016), 191–205.
[8] Juan C Dueñas and Rafael Capilla. 2005. The decision view of software architecture. In *European Workshop on Software Architecture.* Springer, 222–230.
[9] Stephen T Hassard, Ann Blandford, and Anna L Cox. 2009. Analogies in design decision-making. In *Proc. of the 23rd British HCI group annual conf. on people and computers: celebrating people and technology.* British Computer Society, 140–148.
[10] Alan R Hevner. 2007. A three cycle view of design science research. *Scandinavian journal of information systems* 19, 2 (2007), 4.
[11] Anton Jansen. 2008. *Architectural design decisions.* Ph.D. Dissertation.
[12] Rick Kazman, Mark Klein, and Paul Clements. 2000. *ATAM: Method for architecture evaluation.* Technical Report. DTIC Document.
[13] Frido Koch. 2016. *REST-based data integration services for software engineering domain.* Master's thesis. Technische Universität München.
[14] Philippe Kruchten. 2004. An ontology of architectural design decisions in software intensive systems. In *2nd Groningen workshop on software variability.* Groningen, The Netherlands, 54–61.
[15] Peng Liang and Paris Avgeriou. 2009. Tools and technologies for architecture knowledge management. In *Software Architecture Knowledge Management.* Springer, 91–111.
[16] Cornelia Miesbauer and Rainer Weinreich. 2013. Classification of design decisions–an expert survey in practice. In *European Conf. on Software Architecture.* Springer, 130–145.
[17] Audris Mockus and James D Herbsleb. 2002. Expertise browser: a quantitative approach to identifying expertise. In *Proc. of the 24th ICSE.* ACM, 503–512.
[18] Dewayne E Perry and Alexander L Wolf. 1992. Foundations for the study of software architecture. *ACM SIGSOFT Software Engineering Notes* 17, 4 (1992), 40–52.
[19] Thomas Reschenhofer, Manoj Bhat, Adrian Hernandez-Mendez, and Florian Matthes. 2016. Lessons learned in aligning data and model evolution in collaborative information systems. In *Proc. of the 38th ICSE Companion.* ACM, 132–141.
[20] Herbert A Simon. 1998. What we know about learning. *Journal of Engineering Education* 87, 4 (1998), 343.
[21] Antony Tang, Paris Avgeriou, Anton Jansen, Rafael Capilla, and Muhammad Ali Babar. 2010. A comparative study of architecture knowledge management tools. *Journal of Systems and Software* 83, 3 (2010), 352–370.
[22] Antony Tang, Maryam Razavian, Barbara Paech, and Tom-Michael Hesse. 2017. Human aspects in software architecture decision making: a literature review. In *Software Architecture (ICSA), 2017 IEEE Int. Conf. on.* IEEE, 107–116.
[23] Jan Salvador van der Ven and Jan Bosch. 2013. Making the right decision: Supporting architects with design decision data. In *European Conf. on Software Architecture.* Springer, 176–183.
[24] Hans van Vliet and Antony Tang. 2016. Decision making in software architecture. *Journal of Systems and Software* 117 (2016), 638–644.
[25] Carmen Zannier, Mike Chiasson, and Frank Maurer. 2007. A model of design decision making based on empirical results of interviews with software designers. *Information and Software Technology* 49, 6 (2007), 637–653.
[26] Carmen Zannier and Frank Maurer. 2007. Social factors relevant to capturing design decisions. In *Proc. of the 2nd Workshop on SHAring and Reusing architectural Knowledge Architecture, Rationale, and Design Intent.* IEEE Computer Society, 1.
[27] Olaf Zimmermann, Jana Koehler, Frank Leymann, Ronny Polley, and Nelly Schuster. 2009. Managing architectural decision models with dependency relations, integrity constraints, and production rules. *Journal of Systems and Software* 82, 8 (2009), 1249–1267.