Supporting Architectural Decision Making for Systems-of-Systems Design under Uncertainty

Ioanna Lytra and Uwe Zdun
Faculty of Computer Science
University of Vienna, Austria
E-Mail: firstname.lastname@univie.ac.at

ABSTRACT

For the design and integration of complex systems-of-systems, various architectural decisions for recurring design problems need to be made. This requires that the software architects consider various design issues and alternatives, make trade-offs for competing requirements, and adapt the decisions to specific technologies and systems. Documentations of reusable architectural design decisions (ADDs), e.g., pattern-based decisions, provide rather informal guidelines for making recurring ADDs. These and other factors introduce many sources of uncertainty in the architectural decision making process. Existing approaches do not consider this inherent uncertainty of architectural decision making, which has been until now largely ad hoc and informal, without explicit, automated support. Apart from that, the design rationale for repeated ADDs often remains undocumented, leading to loss of architectural knowledge. To address these problems we propose to provide semi-automated support for decision making and documentation of reusable ADDs under uncertainty using a fuzzy logic expert system. We motivate our approach using a systems-of-systems example from the industry automation area in which our approach has been applied.

Keywords

Architectural Design Decisions, Design Pattern Selection, Architectural Knowledge, Systems-of-Systems, Fuzzy Logic

1. INTRODUCTION

During the design of software intensive systems that integrate or reuse existing software systems, leading to complex Systems-of-Systems (SoS), various recurring and non-recurring design problems, high-level as well as technology and system specific, need to be resolved. For the software architects, the process of selecting and implementing the right solutions contains multiple steps that require a certain amount of expertise and domain-specific knowledge. First of all, relevant alternative ADDs along with their forces and consequences need to be considered. Afterwards, the software architects need to understand how the selected ADDs will fit into the overall architecture. Finally, application-generic ADDs also have to be adapted to technology and system specific contexts.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Copyright 20XX ACM X-XXXXX-XX-X/XX/XX ...$15.00.

In this work, we target mainly recurring decisions rather than design issues that require creative thinking and problem solving. Let us consider, for instance, pattern-based ADDs which can be used as a foundation to document recurring ADDs [6, 15]. To find the appropriate software patterns during the decision making process alternative design patterns, pattern variants and implementations need to be considered. Also, various forces and consequences in the context of these alternative options and numerous competing requirements – often vague and imprecise – need to be balanced. Usually, pattern documentations are written in informal and narrative style and are subject to the reader's interpretation, as information concerning forces, consequences and technology mapping are imprecise and often scattered, not searchable, and not cross-referenced in the pattern texts.

In other words, the task of making rational ADDs contains many sources of uncertainty. Uncertainty is caused by imprecisely known or unknown information such as desired requirements and quality attributes of design solutions. For recurring design issues in large-scale software systems, the task of resolving the uncertainty of decision making is on the one hand complex and on the other hand tedious and time-consuming: It is complex because the human mind is weak in reasoning with large amounts of inter-related information that contain uncertainty [14]; tedious and time-consuming because the same decision making process has to be performed many times for similar design issues, sometimes by different architects. In our approach, we suggest to provide semi-automated support for architectural decision making under uncertainty using a fuzzy logic expert system. We motivate our approach with a SoS illustrative example and give some examples where decision making and documentation can be partly automated.

The remainder of the paper is structured as follows. In the following subsections we summarize the research challenges and present our proposal for addressing these challenges. In Section 2 we introduce a motivating example of SoS design based on an industrial case study and provide some examples of recurring ADDs. In Section 3 we present our fuzzy logic based approach for semi-automated decision support. We compare our approach to related work in Section 4. Finally, we conclude and discuss future work in Section 5.

1.1 Research Challenges

In the following, we summarize the *research questions* we address in our work.

RQ1: How to resolve the inherent uncertainty in architectural decision making? For a design problem at hand software architects need to consider competing and imprecise requirements, various design alternatives and technology implementations, informal documentations and domain expertise. All these factors introduce many sources of uncertainty. It is very challenging to make this uncertainty explicit during the decision making process.

RQ2: How to adapt application-generic ADDs to technology and system specific contexts? Software architects need to consider not only the general ADDs but also system and technology options. It is an open challenge to consider both application-generic and application-specific knowledge covering the whole design space at design time.

RQ3: How to provide (semi-)automated support for making and documenting reusable ADDs? For recurring design problems in a specific context making and documenting ADDs can be very tedious. Thus, an open challenge in designing complex software systems is to support the selection of the most appropriate solutions for a design situation at hand.

The proposed solutions to *RQ1* and *RQ2* will be used as basis for addressing *RQ3*.

1.2 Proposed Approach

Until now, decision making has been performed mainly ad hoc and informally without any automated support. A considerable amount of tools (e.g., ADDSS [4]) and methodologies (e.g., ATAM [3]) have been developed for assisting architectural decision making. However, the existing approaches do not consider the inherent uncertainty of reusable ADDs and their adaptation to technology or system specific contexts.

We propose to address the aforementioned research questions by introducing a fuzzy logic expert system for providing semi-automated decision support for recurring ADDs under uncertainty. Fuzzy logic helps us to deal with the imprecision and ambiguity of the decision making process. In particular, we integrate reusable ADDs and fuzzy logic by creating fuzzy models leveraging experts' knowledge (*RQ1*), and provide a fuzzy inference system to give automated guidance on reusable ADDs (*RQ3*). Along with the general fuzzy models we derive specialized fuzzy models for specific technologies and system contexts (*RQ2*). All fuzzy models are described using a domain-specific language (DSL) and get stored in a repository, thus providing reusable assets for architectural decision making.

Our overall approach is *semi-automated*, combining the human decision making by the architects with automated guidance for recurring decisions where the human decision maker requires help. The final decision is left to the software architect. After the decision is made the ADD rationale can be documented automatically.

2. MOTIVATING CASE

To illustrate the research questions we demonstrate an example of SoS design from the industry automation area which deals with service-based platform integration. Three heterogeneous platforms, a Warehouse Management System (WMS) for handling the storage of the goods into racks, a Yard Management System (YMS) for coordinating the trucks in a yard and a Remote Maintenance System (RMS) for monitoring incidents and communicating with operators in the warehouse, need to be integrated to allow an operator application to utilize the services provided by these platforms.

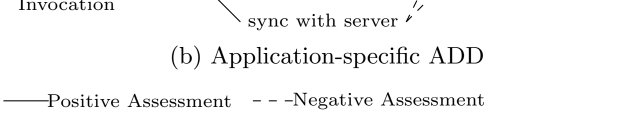
The operator system uses the services of the three platforms via a domain-specific virtual service platform (VSP) which performs service-based platform integration. The VSP must handle various integration aspects including interface adaptation between the platforms; integration of service-based and non-service-based solutions; routing, enriching, aggregation, splitting, etc. of messages and events; handling synchronization and concurrency issues, and so on. To design the details of such integration solutions and the systems on top of it, for all the integration aspects, high-level, application and technology-specific architectural decisions must be made.

In many cases, the software architects need to consider alternative design solutions, variants and implementations with different properties and quality attributes and balance imprecise requirements to come up with the best-fitting solutions. In this context, this task has to be performed multiple times and by different software architects with different experience and domain expertise. For instance, in our motivating case each of the three platforms (WMS, YMS and RMS) offers numerous services and for each single service its adaptation to the VSP (e.g., using adapter or proxy), its communication style (e.g., synchronous or asynchronous), and so on need to be decided separately, which leads to a big number of recurring ADDs. Also, general architectural decisions have to be grounded in specific technologies. Thus, both levels of decisions – application-generic and application-specific – need to be considered at the decision making process.

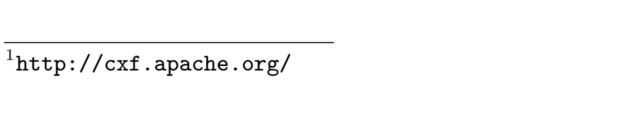
Example of Reusable ADDs.

For the needs of the design of the warehouse automation case study, we have consolidated in our previous work [8] a service-based platform integration pattern language and a reusable ADD model, covering high-level and low-level design issues. These reusable ADDs, which have to be repeated many times during the design process, constitute the basis for our fuzzy logic based approach. In particular, the existing knowledge and the expert experience about making these decisions repeatedly will be reflected in the fuzzy models using fuzzy rules.

Figure 1(a) depicts an excerpt of a pattern-based decision on asynchronous invocations, and Figure 1(b) shows an excerpt of a technology specific decision from the implementation of this decision using the Apache CXF[1] framework. Asynchronous invocations allow client applications to resume their work while waiting for response from a remote invocation [13], thus improving scalability and performance. In this example, three alternative design patterns (*fire and forget*, *sync with server* and *result callback*) are linked to the quality attributes *performance*, *reliability* and *(supports) acknowledgment* which are positive (solid lines) or negative (dotted lines) assessed during architectural decision making. Information about the forces and consequences of the aforementioned patterns, as well as the input requirements are usually imprecise. For instance, performance (see 1(a)) can be high, medium or low, but the understanding of these three linguistic values as well as the boundaries between them are usually vague. The implementation of the application-generic ADD using the Apache CXF framework requires that complementary quality attributes (*supported*, *transport neutral*) are also considered for the decision making.



(a) Application-generic ADD



(b) Application-specific ADD

——Positive Assessment – – –Negative Assessment

Figure 1: Reusable ADDs

[1]`http://cxf.apache.org/`

3. APPROACH OVERVIEW

Our proposed methodology aims to provide semi-automated support for specific recurring ADDs and resolve their inherent uncertainty. Rather than creating a new design from scratch, it automates the decision making for design problems that emerge repetitively in a specific context. Our purpose is to cover the whole design space for a design situation at hand consisting of generic, as well as technology-specific decisions. To address uncertainty we use Fuzzy Logic [14], which allows the numerical encoding of the vague linguistic values software engineers use to describe requirements as well as forces and consequences of reusable ADDs. Key concepts of Fuzzy Logic are fuzzy sets and their membership functions, which express *degrees of membership* spanned in the interval $[0, 1]$ for the elements of the fuzzy sets. These linguistic values can be interpreted using fuzzy sets which get mapped to overlapping membership functions (e.g., gaussian, trapetzoidal, etc.). For example, the property *performance* could be described as *high*, *medium* or *low* and these linguistic values can be mapped to overlapping membership functions, as shown in Figure 2.

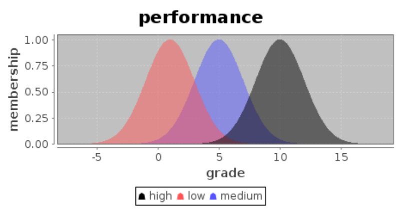


Figure 2: Gaussian membership functions for 3 linguistic values of property performance

Figure 3 presents an overview of our approach, namely the participating tools and roles. We distinguish between two stakeholder roles: *software architect (expert)* and *software architect (user)*. That is, different levels of experience are expected for architects who create the fuzzy logic models and users of our approach. The software architects (experts) use the *Fuzzy Decisions Models Editor* (a textual DSL editor) to capture architectural knowledge. A decision model contains alternative design solutions along with their properties and quality attributes and a set of expert IF–THEN fuzzy rules that guide the design decisions. From these decision models we derive specialized fuzzy decision models in which domain and technology specific knowledge can be included. Both kinds of decision models get stored in a *Fuzzy Decision Model Repository* for reuse by a *Fuzzy Inference System*. The software architects (users) use the *Requirements Editor* to give the desired requirements in crisp values using a grading system (e.g., 1–10) for fuzzy input variables like *performance* and *reliability* and binary values (i.e., 0, 1) for variables that accommodate only two values (e.g., Yes, No) like *supports acknowledgment*.

The *Fuzzy Inference System* returns the appropriate design alternatives and their ranking for the given requirements by evaluating and combining the fuzzy rules already defined in the fuzzy models. The list of the best-fitting design solutions is supposed to be used as a decision aid for the software architect who makes the final decision. After that, the input requirements and the inferred design solutions can be synthesized to produce *ADD Documentations*.

3.1 Fuzzy Application-generic and Application-specific Decision Models

Elements of architectural knowledge get mapped to fuzzy logic. In particular, design solution alternatives, variants and implementations get mapped to (binary-valued) fuzzy outputs. Forces and consequences of alternative ADDs, which constitute major decision criteria, are mapped to fuzzy inputs and membership functions. In the example of our DSL in Listing 1 two quality attributes (`reliability` and (`supports) acknowledgment`) are mapped to three gaussian (for the values `low`, `medium` and `high`) and two singleton (for the values `no` and `yes`) membership functions respectively.

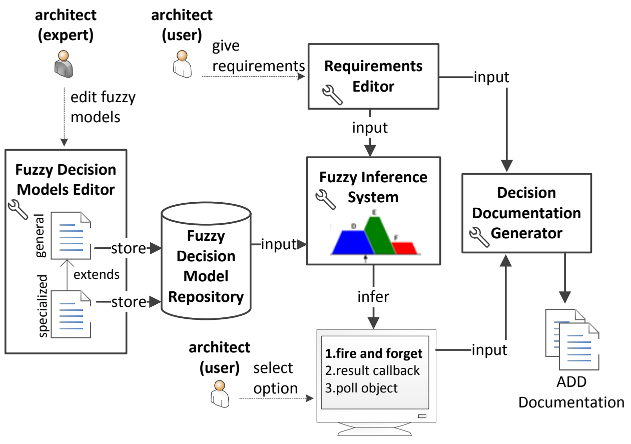


Figure 3: Fuzzy Logic Based Approach for Supporting Architectural Decision Making

```
attributes
   reliability gauss {low medium high}
   acknowledgment singleton {no yes}
end
```

Listing 1: Example of fuzzy outputs

The relationship between fuzzy outputs (i.e., design alternatives) and fuzzy inputs (i.e., their forces and consequences) can be expressed using IF–THEN fuzzy rules, as in Listing 2.

```
if performance is high and reliability is medium then
    result_callback
```

Listing 2: Example of fuzzy rule

To adapt general architectural knowledge to concrete system and technology contexts we derive specialized decision models by inheriting the general fuzzy models (i.e., fuzzy outputs and fuzzy rules). Architectural knowledge gets specialized by adding, prioritizing or deleting choices, forces, consequences and rules. Thus, the general architectural knowledge gets transferred to and combined with the technology-specific knowledge.

3.2 Derivation of ADDs and Documentation

A Mamdani-based fuzzy inference system [9] is used to infer best-fitting solutions for given requirements in crisp values (e.g., on a scale 1–10). In particular, the fuzzy inference system fuzzifies the input requirements, evaluates the fuzzy rules and aggregates the outputs, which produces an ordered list of possible design solutions with different weights. The outputs of the fuzzy inference system can be manipulated by changing the fuzzy models, namely the forces and consequences and their membership functions, the rules or the weight of the rules. So far, the fuzzy decision models need to be tuned manually; we plan, however, to semi-automate this process by suggesting and improving membership functions and rules.

Another important contribution of this approach is the automatic generation of architectural decision documentation. The input requirements, information of the fuzzy rules, the actual ADDs and the alternative ADDs get synthesized to produce design decision documentation.

4. RELATED WORK

Although there have been many works on making architectural decisions an explicit part of the architecture (e.g., [7]) their inherent uncertainty has not been studied systematically in the literature. In our work we define the sources that introduce uncertainty at the decision making process and discuss how to deal with this uncertainty.

Our approach is not the first one to systematize reusable application-generic and application-specific knowledge and provide decision making support. For instance, Zimmermann et. al [15] propose a reusable architectural decision model for SOA and a wiki-based decision support tool. Other research works document reusable pattern-based design decisions for web services [11] and service-based platform integration [8]. However, these approaches do not provide any automated support for architectural decision making and fail to cover the whole solution space and resolve the inherent uncertainty of ADDs. A number of approaches (e.g., [12, 15]) capture technology-specific knowledge in the fields of decision meta-models or decision templates. These approaches consider many levels of design knowledge during decision making, but usually the knowledge levels are only depicted as fields in decision meta-models and templates. So far, an automated support for specialization or generalization of design knowledge and a mapping between application-generic and application-specific knowledge, as proposed by our approach, has not been studied in the literature.

In many approaches, architectural decisions are the result of making trade-offs for the quality attribute requirements. For example, in the Architecture Tradeoff Analysis Method (ATAM) and Attribute-Driven-Design Method (ADD) [3] the analysis of architectural trade-offs is an important part of the architectural decision making process. Bachmann et. al [2] suggest a reasoning framework with quality attribute knowledge to help architects make trade-offs that impact individual quality attributes in an architecture. Both techniques focus on the quality attributes of software architectures as a whole and not on the quality attributes of single ADDs. Also, they do not deal with the inherent uncertainty of ADDs.

Aksit and Marcelloni [1] use fuzzy logic based techniques to defer the elimination of design alternatives, in order to enhance object-oriented methods. Moaven et al. [10] propose a multi-criteria decision support system for the selection of architectural styles using fuzzy Choquet integral. The last approach requires that quality attributes like flexibility and maintainability for each of the architectural styles (pipes & filters, layered, etc.) have been evaluated using a grading system through an online survey. None of these approaches has been extended to cover reusable ADDs. Esfahani and Malek [5] suggest a framework (GuideArch) for guiding ADDs under uncertainty. In their approach they fuzzify and prioritize properties of architectural design alternatives based on existing measurements and compare architectures based on fuzzy requirements. They assume that they already have all possible implementations and measurements of optimistic, pessimistic, and anticipated properties values in order to calculate the optimal architecture. In contrast, our approach provides semi-automated decision support at design time.

5. CONCLUSIONS AND FUTURE WORK

The introduced fuzzy logic based approach for design decision making addresses key challenges in the area of architectural decision support, especially for SoS design where same or similar ADDs have to be repeated many times. It is the first approach to provide automated guidance for recurring ADDs under uncertainty. It is our goal to support the software architects in recurring design making processes, so that they have more time left to spend on the challenging problems that require creative thinking. The whole design space, including application-generic as well as application-specific decisions is modeled once, however, the fuzzy logic decision models can be retrieved many times from the fuzzy model repository to derive appropriate solutions for desired requirements.

In the first stages of this research work we have created a DSL for specifying the fuzzy decision models and implemented a fuzzy inference system for deriving best-fitting design solutions. We have implemented this DSL for describing general and specialized design knowledge for an excerpt of the design space of the case study from Section 2 and plan to use our tools for making and documenting all recurring decisions in the context of this case study. We would like to evaluate the complexity and efficiency of our method with practitioners in various design situations that require reuse of decisions.

Our fuzzy logic based approach for decision making support launches new challenges for the future. We are interested in studying how to modify the fuzzy decision models and fuzzy inference system in order to integrate inter-decision dependencies, which are very common, for instance, in pattern-based decision making. Another open challenge is the automatic tuning of the expert system (i.e., the fuzzy rules and membership functions) by giving feedback from existing design decisions. Finally, we would also like to study the possibility of inferring generic knowledge from technology-specific knowledge.

6. ACKNOWLEDGMENTS

This work was partially supported by the EU FP7 project INDENICA (`http://www.indenica.eu`), grant no. 257483.

References

[1] Aksit, M., and Marcelloni, F. Deferring Elimination of Design Alternatives in Object-Oriented Methods. *Concurrency and Computation 13*, 14 (2001), 1247–1279.

[2] Bachmann, F., Bass, L., Klein, M., and Shelton, C. Designing software architectures to achieve quality attribute requirements. *Software, IEEE Proc. 152*, 4 (Aug. 2005), 153–165.

[3] Bass, L., Clements, P., and Kazman, R. *Software architecture in practice*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1998.

[4] Capilla, R., Nava, F., Pérez, S., and Dueñas, J. C. A web-based tool for managing architectural design decisions. *SIGSOFT Software Engineering Notes 31*, 5 (Sept. 2006).

[5] Esfahani, N., and Malek, S. Guided Exploration of the Architectural Solution Space in the Face of Uncertainty. Tech. Rep. GMU-CS-TR-2011-3, Department of Computer Science, George Mason University, 2011.

[6] Harrison, N., Avgeriou, P., and Zdun, U. Using Patterns to Capture Architectural Decisions. *IEEE Software 24*, 4 (2007), 38–45.

[7] Jansen, A., and Bosch, J. Software Architecture as a Set of Architectural Design Decisions. In *The 5th Working IEEE/IFIP Conf. on Software Architecture* (2005), IEEE Comp. Soc., pp. 109–120.

[8] Lytra, I., Sobernig, S., and Zdun, U. Architectural Decision Making for Service-Based Platform Integration: A Qualitative Multi-Method Study. In *Joint 10th Working IEEE/IFIP Conf. on Software Architecture & 6th European Conf. on Software Architecture* (2012), IEEE Comp. Soc., pp. 111–120.

[9] Mamdani, E. H., and Assilian, S. An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller. *Int'l Journal of Man-Machine Studies 51*, 2 (1999), 135–147.

[10] Moaven, S., Habibi, J., Ahmadi, H., and Kamandi, A. A Decision Support System for Software Architecture-Style Selection. In *Proc. of the 6th Int'l Conf. on Software Engineering Research, Management and Applications* (2008), IEEE Comp. Soc., pp. 213–220.

[11] Pautasso, C., Zimmermann, O., and Leymann, F. RESTful Web Services vs. "Big" Web Services: Making the Right Architectural Decision. In *Proc. of the 17th Int'l Conf. on World Wide Web* (2008), WWW '08, ACM, pp. 805–814.

[12] Tyree, J., and Akerman, A. Architecture Decisions: Demystifying Architecture. *IEEE Software 22*, 2 (2005), 19–27.

[13] Voelter, M., Kircher, M., and Zdun, U. *Remoting Patterns*. John Wiley & Sons, 2004.

[14] Zadeh, L. A. Fuzzy sets. *Information and Control 8*, 3 (1965), 338–353.

[15] Zimmermann, O., Zdun, U., Gschwind, T., and Leymann, F. Combining Pattern Languages and Reusable Architectural Decision Models into a Comprehensive and Comprehensible Design Method. In *7th Working IEEE/IFIP Conf. on Software Architecture* (2008), IEEE Comp. Soc., pp. 157–166.