Dealing with Uncertainty in Early Software Architecture

Naeem Esfahani
Computer Science Dept.
George Mason University
nesfaha2@gmu.edu

Kaveh Razavi
Computer Science Dept.
George Mason University
srazavi2@gmu.edu

Sam Malek
Computer Science Dept.
George Mason University
smalek@gmu.edu

1. MOTIVATION

A software system's *early architecture* is the set of principal decisions made at the outset of a software engineering project. Early architecture encompasses choices at application, system, and hardware level that could have an impact on the software system's properties.[1] A common practice is to carefully assess the system's early architecture for its ability to satisfy functional and non-functional requirements, as well as other stakeholder concerns, such as cost and time to delivery.

Early architectural decisions are crucial, as they determine the scope of capabilities and options that can be exercised later in the system's life cycle. Given the crucial impact of early architectural decisions on the system's properties, changing them in subsequent phases of the engineering process are often both difficult and costly. At the same time, making such decisions is a complex task mired with lots of uncertainty. Getting them "wrong" poses a risk to any software engineering project.

One of the major thrusts of the software engineering research has been to transform the process of making such decisions from an art form exercised successfully by a select few to a repeatable process guided through scientific reasoning and formal analysis. A few notable examples include ATAM [4], CBAM [10], and ArchDesigner [1]. Such efforts have not aimed to replace the engineer's experience and knowledge, but to rather augment it through provisioning of appropriate methods and tools.

While great strides have been made on this front, the existing architecture decision-making approaches do not provide a quantitative method of dealing with uncertainty [8]. In fact, there is no quantitative method of even comparing two architectures under uncertainty, let alone selecting the "right" architecture from the many possible candidates in such circumstances [1, 8].

2. NEW IDEA

In this paper, we provide an overview of GuideArch, a quantitative framework aimed at guiding the exploration of architectural solution space under uncertainty. It allows the architect to make informed decisions using imperfect information. This alleviates the architect from manually sifting through an often large solution space, and instead allows her to focus on the decisions that are critical to the system's success.

[1]While our definition of "early architecture" is rather broad, and could incorporate decisions dealing with hardware, system, and software, our focus is mainly on those impacting the software.

Unlike any existing approach [1, 4, 10], GuideArch explicitly represents the inherent uncertainty in the knowledge and incorporates that in the analysis. It enables an incremental method of making and refining architectural decisions throughout the engineering process. As the rough estimates in the early stages give way to precise estimates in the later stages, GuideArch allows the architect to refine the models and explore other suitable alternatives.

GuideArch employs *fuzzy mathematical* methods [14] to reason about uncertainty. We have devised a novel fuzzy operator that forms the foundation for quantitative comparison of architectural candidates under uncertainty. The fuzzy operator is then used to develop advanced analysis techniques, including optimization and ranking of architectures, and identification of critical design decisions.

Our research is motivated by the fact that precisely specifying the effect of architectural decisions on goals (e.g., functional or non-functional requirements) is difficult, particularly in early phases of engineering. This observation is also corroborated by other researchers and practitioners [1, 10]. As a result, the process of making early architectural choices is a risky proposition mired with uncertainty.

Previous approaches (e.g., [1, 10]) that support the process of making decisions and optimizing the system's architecture ignore these challenges, hampering their adoption in real-world risk-averse domains. We collectively refer to these as the *traditional approaches*. We illustrate their shortcoming using a problem in which the objective is to choose from a pool of 16 candidate architectures, such that *Cost* and *Battery Usage* are minimized.

In our research, early architecture is defined simply as a set of decisions. For instance, an architect of a mobile application may face several architectural decisions: the architectural style of the application, the technique for locating the device, the data persistence mechanism, and so on. For each decision she may have several alternatives, e.g., in the case of style, the architect may be able to choose from pipe-and-filter or client-server style. A candidate architecture results from selecting a viable alternative for each and every decision. While this is a rather simplified way of looking at the notion of architecture, we believe it is expressive enough to capture the essence of architectural decision-making problem. Conventionally one associates an architectural diagram as the kind of artifact an architect deals with, however, in effect, such a diagram is nothing more than a collection of decisions. Moreover, often at the early architecting phase, an architect does not have an architectural model of the system. It is the high-level decisions made early on that manifest themselves in the form of constructs comprising the architectural model in the later stages of software devel-

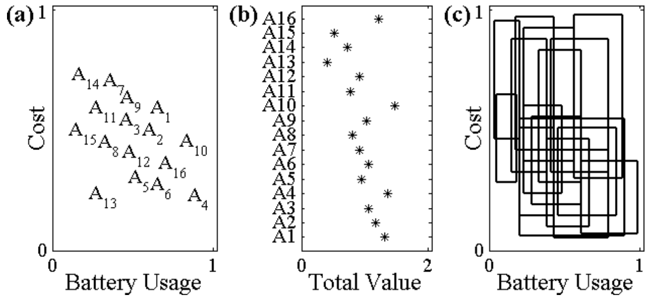


Figure 1: Assessing architectural candidates: (a) 16 candidate architecture in a cost vs. battery usage trade-off, (b) simple additive approach to resolve the trade-offs, and (c) cost vs. battery usage under uncertainty, where each rectangle represents the space of values that an architecture may take.

opment lifecycle.

The traditional approaches assume that the architect is able to precisely specify the impact of candidate architectures on properties of interest. If that was the case, then one could visualize the situation as in Figure 1a. Here, for the sake of clarity, the values for *Cost* and *Battery Usage* are normalized between 0 and 1. Assuming both properties have the same level of importance, to compare the 16 candidates, for each architecture we first sum up the values obtained in the two properties. Figure 1b achieves just that, as it shows the overall value for the candidate architectures. In this space, architectures can be compared with one another. For example, we can see that A_{13} is the best architecture, as it obtains the smallest total value.[2] It is also possible for several architectures to obtain the same value, in which case the architect would need to provide a prioritization scheme, such that more emphasis is placed on certain properties. However, for clarity we do not consider such cases here. While the aforementioned approach is theoretically sound, it is not useful in practice, as it does not incorporate the underlying uncertainty in the impact of architectural decisions on properties of interest.

The complexity of incorporating uncertainty in the analysis is shown in Figure 1c. Here, the architect's uncertainty is represented in terms of range of impact that an architectural candidate may have on the properties of interest. For example, the impact of a given architecture on *Battery Usage* is no longer a single number, but rather a range of values. As a result, each architectural candidate may obtain a value anywhere within the area occupied by the corresponding rectangle. Clearly, comparing two architectures with overlapping rectangles is difficult. It is not clear how the rectangles in Figure 1c can be transformed to a space where the trade-off analysis can be performed.

To gain a better appreciation for the complexity of this problem consider that the simple example used in Figure 1 consists of only 16 architectural candidates and 2 properties of interest, but a typical software system often consists of many more candidates and properties. Manually exploring such a large space is a big burden. Incorporating uncertainty into the analysis makes a problem that is already challenging so overwhelmingly complex that a manual assessment without the appropriate tools becomes impossible.

[2]For *Battery Usage* and *Cost*, lower values are preferred; hence, they are simply added together. For properties that higher values are preferred (e.g., reliability) we subtract the value of the property from the total value. Therefore, even in those cases, a smaller total value is preferred.

3. APPROACH

We accept uncertainty as a natural component of architecting a software system, particularly in the early phases of engineering. Our objective is not to eliminate uncertainty, but to provide quantitative techniques and tools for making informed decisions in such circumstances. This section first describes the scope of our research, followed by an intuitive description of our approach.

3.1 Definition and Scope of Uncertainty

Before delving into the approach, it is important to clarify what we mean by uncertainty. The scope of uncertainty dealt with in our paper has to do with not knowing the exact impact of architectural alternatives on properties of interest, i.e., not being able to precisely specify the impact as a crisp value. However, there are other sources of uncertainty in early architecting that are not tackled in our work. Consider for instance the uncertainty introduced by the following questions: Have all of the properties of concern been elicited? Have all of the decisions and alternatives been identified? While the ability to answer such questions is clearly crucial, they fall outside the scope of our research.

3.2 Representing Uncertainty in Alternatives

Instead of modeling the anticipated impact of an architectural alternative on the system's properties as a point estimate, we represent it as a *range* of values. Specifying the impact in terms of a range is aligned with the way humans in general conceptualize uncertainty and provides an intuitive method of modeling the architect's knowledge.

The range of impact may be estimated in a number of ways, including the data available from similar designs in other systems, architect's prior knowledge, prototype or early simulations of the system, manufacturer specification, scientific publication, etc. For instance, based on a combination of manufacturer specification and prior experience with smartphones, the architect may estimate that Location Finding using GPS has $10\mu J$ of anticipated battery usage, with $8\mu J$ and $14\mu J$ in optimistic and pessimistic situations, respectively.

While there are other elaborate methods of representing uncertainty, such as *probability distribution*, our experience suggests that such representations are not very useful in this setting. Probability distribution is useful for representing uncertainty due to error or noise in the behavior of a phenomenon (e.g., data collected from a sensor). But uncertainty in architecting is often due to lack of knowledge, and not variability. We note, however, that if such models of uncertainty are available, the range could be derived using the techniques described in [5].

The key contribution of GuideArch is the ability to provide quantitative analysis of the trade-offs given such loose specifications. We achieve this by representing the uncertain parameters as *fuzzy numbers*. A fuzzy number is founded on the concept of *fuzzy set* [13]. In a fuzzy set, the elements have a degree of membership. Degree of membership is a value between zero and one: a value of zero indicates the element is certainly not a member of the set, a value of one indicates it is certainly a member, and a value in between indicates the extent of certainty it is a member. Fuzzy math

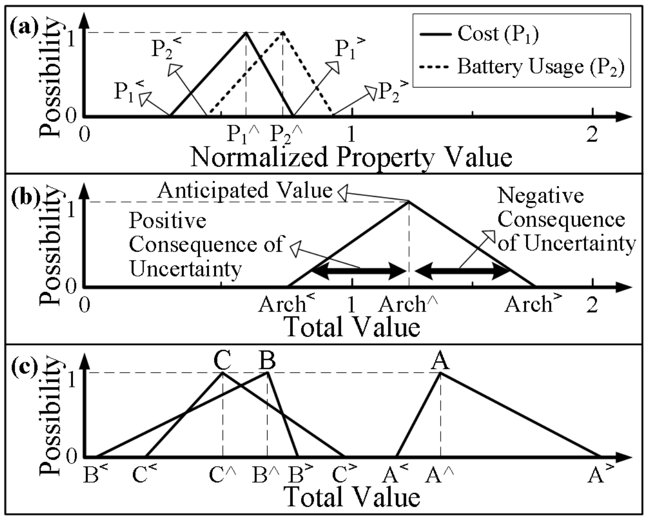


Figure 2: Uncertainty modeled as fuzzy values using possibility theory: (a) the fuzzy values for Cost and Battery usage, (b) their summation to determine the architecture's total value, and (c) the total value for three hypothetical architectures.

is grounded in *possibility theory* [13], which provides an alternative interpretation of uncertainty to that of probability theory. A common misconception is that fuzzy math is imprecise. On the contrary, fuzzy math, just like probability, provides a precise and sound method of dealing with uncertainty.

We assign the possibility of 1 to the *anticipated* value, and possibility of 0 to the *optimistic* and *pessimistic*, respectively. We use "∧", "<", ">" to represent anticipated, optimistic, and pessimistic, respectively. We let the possibility to decrease linearly from the anticipated to the optimistic and pessimistic points. Thus, the effect of each design alternative on each property is modeled as a *triangular fuzzy value* [14]. For instance, Figure 2a depicts the fuzzy values corresponding to the range of *Cost* (P_1) and *Battery Usage* (P_2) for an architectural candidate. Due to uncertainty, the actual value of the property may be anywhere in that range.

3.3 Calculating Uncertainty in a Candidate

Given the fuzzy impact of alternatives on properties, we can quantify the overall value of a given architecture. Similar to the approach used to transform Figure 1a to Figure 1b, we transform the candidate solutions in Figure 1c to a scalar space, such that they can be compared with one another. The total value for an architecture *Arch* can be calculated as *fuzzy summation* [9] of the impact of alternatives on the properties. When fuzzy numbers are summed up, the pessimistic, anticipated, and optimistic values are added independently of each other to arrive at a new fuzzy value. For instance, adding fuzzy values for *Cost* and *Battery Usage* in Figure 2a results in the fuzzy value shown in Figure 2b, which represents the total value of the corresponding architecture *Arch*.[3] Since an architecture with a lower value is preferred, we call the situation in which the actual value is between anticipated and pessimistic the negative consequence of uncertainty (risk), and the situation in which the actual value is between anticipated and optimistic the positive consequence of uncertainty (opportunity).

[3]The fuzzy summation used here is defined in [9], which has proven to produce another proper possibility distribution.

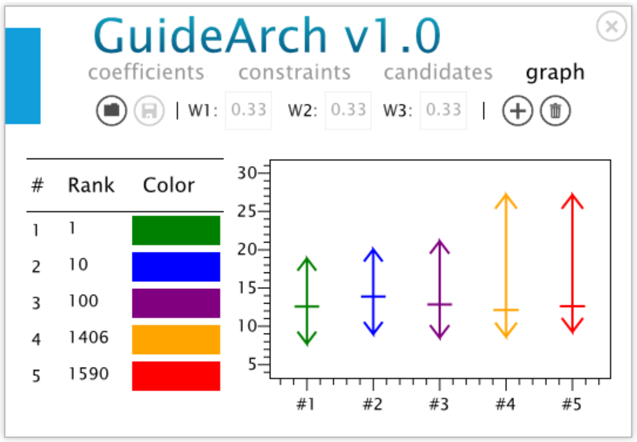
3.4 Comparing Candidate Architectures

Fuzzy summation allows us to transform the multi-dimensional problem into a single scalar value, but since the scalar value itself is fuzzy, comparing the architectural solutions remains a challenge. When comparing two fuzzy numbers, the one with the "better" range is superior. We say the fuzzy value of one architecture is better than another if it has a: (C1) smaller anticipated value, (C2) larger positive consequence of uncertainty, and (C3) smaller negative consequence of uncertainty [7].

Figure 2c shows the total value of the properties for three hypothetical architectures (A, B, and C), which are represented as fuzzy values. Using Figure 2c we describe two possible scenarios that may occur in comparing architectures this way. The first scenario occurs when a given architecture is inferior to others with respect to all three criteria. For instance, in Figure 2c, architecture A is inferior to architectures B and C with respect to all three criteria. The second scenario occurs when there are trade-offs. For instance, architectures B and C present a trade-off, as architecture B is superior to architecture C with respect to C2 and C3, and inferior with respect to C1. Such trade-offs can be resolved by giving weights to each criteria (i.e., W1, W2, and W3).

Figure 3 shows a snapshot of the graphing facility in GuideArch tool, which is used for analysis of architecture space. Each double sided arrow (i.e., $\updownarrow$) depicts the fuzzy total value (recall Figure 2b) for a candidate architecture. For a given candidate *Arch*, the lower side of the arrow (i.e., $\downarrow$) depicts $Arch^<$, while the upper side of the arrow (i.e., $\uparrow$) and the middle of the arrow (i.e., $-$) depict $Arch^>$ and $Arch^\wedge$, respectively.

Architects can use this facility to depict the total fuzzy values of the candidate architectures side-by-side for further analysis. This achieves the goal of Figure 2c with much less clutter. For instance, by showing candidates *#1* and *#4* (which are ranked 1st and 1406th respectively) side-by-side, we can observe that candidate *#4* has a lower anticipated value compared to candidate *#1*. However, since candidate



#	Rank	Color
1	1	🟩
2	10	🟦
3	100	🟪
4	1406	🟧
5	1590	🟥

Figure 3: A snapshot of GuideArch's graphing facility, which is used to compare the candidate architectures side-by-side. By analyzing candidates, the architect will gain a better understanding of the architectural space.

#1 has a better range of uncertainty compared to candidate *#4*, it has a better rank.

Note that traditional approaches, that do not consider uncertainty in their analysis, would only consider the anticipated value and select candidate *#4* over candidate *#1* as a better architecture. By looking at these differences, the architect gains a better understanding of the architectural trade-offs under uncertainty. GuideArch facilitates this process by allowing the architect to select each arrow and look at the detailed information about the corresponding architecture (not shown here due to space limitations).

Interested reader can access GuideArch at `http://mason.gmu.edu/~nesfaha2/Projects/GuideArch/About.htm`.

4. RELATED WORK

Making architectural decisions is a problem that has been studied from both design-time and run-time perspectives. The uncertainty issues in the latter have been mainly researched in the area of autonomic computing [6]. Here we discuss only those targeted at design-time, since that has been the focus of our work.

ArchDesigner [1] is an approach to find an optimal architecture that meets conflicting stakeholders' quality goals. CBAM [10] is a quantitative approach for economic modeling of software engineering decisions, which builds upon ATAM [4]. CBAM provides the cost and benefit of different architectural candidates. ArcheOpterix [2] is a tool for optimizing an embedded system's architecture. It uses evolutionary algorithms for multi-objective optimization of such systems. While many of these approaches acknowledge the challenges posed by uncertainty, none addresses it explicitly and via a mathematical framework.

Palladio [3] uses information about components comprising the architecture to derive analytical models and simulate the system's performance. Random variables are used to specify uncertainty in service demands and iterations. Meedeniya et al. [11] estimate the reliability of a given software architecture by combining reliability of its elements (expressed as probability distributions) using Monte Carlo simulation. These approaches are complementary to GuideArch, as they could be used to specify the range of performance and reliability, respectively. Unlike GuideArch, however, neither considers ranges of possible behaviors for different architectures.

Noppen et al. [12] use *design tree* to navigate in the design space, which may include imperfect information. However, they assume that the number of alternatives for given design decisions of each step is very small. GuideArch, on the other hand, focuses on a single design step with large number of alternatives. In that sense, our work is orthogonal and complementary to their work.

5. CONCLUSION

In any software project, early architectural decisions represent some of the most crucial decisions engineers ever make. Yet there is a lack of techniques and tools for helping the engineers make those decisions. We presented GuideArch, a novel framework that guides the engineers in making the best choices possible under uncertainty.

One area of future work is to extend the current model from a *unified* stakeholder perspective to a *multiple* stakeholder perspective. We currently assume all stakeholders have agreed on the impact of alternatives on properties and their priorities. However, this may not always be the case, presenting GuideArch with yet another source of uncertainty. Future work also includes extending GuideArch to deal with the other types of uncertainty discussed in Section 3.1. Moreover, GuideArch currently represents uncertainty as a *triangular fuzzy value*, which is not only the most widely used fuzzification approach, but also universally applicable. However, in our future work, we also plan to experiment with alternative representations (e.g., *trapezoidal* and *Gaussian*) that seem to offer some unique trade-offs.

6. REFERENCES

[1] Al-Naeem, T., Gorton, I., Babar, M. A., Rabhi, F., and Benatallah, B. A quality-driven systematic approach for architecting distributed software applications. In *Int'l Conf. on Software Engineering* (St. Louis, Missouri, May 2005), pp. 244–253.

[2] Aleti, A., Bjornander, S., Grunske, L., and Meedeniya, I. ArcheOpterix: an extendable tool for architecture optimization of AADL models. In *ICSE Workshop on Model-Based Methodologies for Pervasive and Embedded Software* (Vancouver - Canada, May 2009), pp. 61–71.

[3] Becker, S., Koziolek, H., and Reussner, R. The palladio component model for model-driven performance prediction. *J. Syst. Softw. 82*, 1 (Jan. 2009), 3–22.

[4] Clements, P., Kazman, R., and Klein, M. *Evaluating Software Architectures: Methods and Case Studies.* Addison-Wesley Professional, Nov. 2001.

[5] Dubois, D., Prade, H., and Sandri, S. On possibility/probability transformations. In *IFSA Conference* (Seoul, Korea, July 1993).

[6] Esfahani, N., Kouroshfar, E., and Malek, S. Taming uncertainty in Self-Adaptive software. In *Int'l Symp. on the Foundations of Software Engineering* (Szeged, Hungary, Sept. 2011), pp. 234–244.

[7] Facchinetti, G., and Ghiselli Ricci, R. A characterization of a general class of ranking functions on triangular fuzzy numbers. *Fuzzy Sets and Systems 146*, 2 (Sept. 2004), 297–312.

[8] Garlan, D. Software engineering in an uncertain world. In *FSE/SDP Wrkshp. on the Future of Software Engineering Research* (Santa Fe, New Mexico, Nov. 2010), pp. 125–128.

[9] Kaufmann, A., and Gupta, M. M. *Fuzzy mathematical models in engineering and management science.* North-Holland, 1988.

[10] Kazman, R., Asundi, J., and Klein, M. Quantifying the costs and benefits of architectural decisions. In *Int'l Conf on Software Engineering* (Toronto, Canada, May 2001), pp. 297–306.

[11] Meedeniya, I., Moser, I., Aleti, A., and Grunske, L. Architecture-based reliability evaluation under uncertainty. In *Int'l Conf on the Quality of Software Architectures* (Boulder, CO, June 2011), pp. 85–94.

[12] Noppen, J., van den Broek, P., and Aksit, M. Software development with imperfect information. *Soft Comput. 12*, 1 (Aug. 2007), 3–28.

[13] Zadeh, L. A. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets Syst. 100* (June 1999), 9–34.

[14] Zimmermann, H. *Fuzzy Set Theory and its Applications (4th Edition)*, 4th ed. Springer, Oct. 2001.