Poster: Making Well-Informed Software Design Decisions

Arman Shahbazian, Youn Kyu Lee, Yuriy Brun, and Nenad Medvidovic

University of Southern California
Los Angeles, CA 90089-0781, USA
{armansha, younkyul, neno}@usc.edu

University of Massachusetts Amherst
Amherst, MA 01003-9264, USA
brun@cs.umass.edu

ABSTRACT

Design decisions software architects make directly impact system quality. Real-world systems involve a large number of such decisions, and each decision is typically influenced by others and involves trade-offs in system properties. This paper poses the problem of making complex, interacting design decision relatively early in the project’s lifecycle and outlines a search-based and simulation-based approach for helping architects make these decisions and understand their effects.

CCS CONCEPTS
- Software and its engineering → Designing software

KEYWORDS
Model-driven engineering; software architecture

ACM Reference format:
https://doi.org/10.1145/3106237.3106277

1 MAKING ARCHITECTURAL DESIGN DECISIONS

Fairly early in a software system’s life cycle, software architects make a set of critical design decisions that form the system’s architecture. Some of these decisions are made fresh, while others are borrowed from previous versions of the system or other existing similar systems. Some decisions are influenced by established architectural styles and patterns, some by selected implementation frameworks and libraries, and some by the architects’ expertise and prior experiences. To name just a few aspects of systems affected, architectural design decisions address system structure, system behavior, component interaction, system deployment, system evolution, and non-functional properties [28]. It has been long accepted that the number grows quickly with the complexity of the system [5]. As an example, designing Apache Hadoop required well over one hundred design decisions [26]. And the large number of decisions is only part of quantifying the difficulty of system design, as many decisions involve intertwined factors and force trade-offs in system properties that must be considered [4, 24].

A software architecture includes many variation points that can take on one of a set of possible alternatives, e.g., employing either an encrypted or plain-text data storage, or using either a relational database, a document database, or a key-value store. A design decision is the selection of one of these alternatives. Ideally, when making a decision, architects carefully assess each alternative and how it satisfies or affects each of the system’s requirements; however, this is frequently not done in practice [6]. The Healthcare.gov portal is a recent example of ineffective design-decision impact assessment, leading to serious technical problems at launch [17] and a development cost of US$1.5B, despite original estimates of ~US$100M [16]. For example, the portal’s downtimes of up to 60% were caused by flawed architectural and deployment design decisions. The system was deployed using a single-node NoSQL database that also stored federal government employee information, rather than using a distributed database configuration. This decision alone caused half of the system outages [30].

As long as the design decisions are even partially independent, the space of possible systems resulting from all the selections of concrete alternatives grows exponentially in the number of decisions. Manually comparing these potential systems is infeasible for most systems, and the community has recognized the need for tools to support architects in evaluating these decisions [12].

2 STATE-OF-THE-ART DECISION SUPPORT

To make effective design decisions, architects need to understand the effects of the decisions on the final system. Thus, it is helpful to be able to objectively assess these potential final systems before building the systems and even before making all the decisions [24]. The existing approaches to assess such systems rely on static or dynamic analysis of system models. Static analysis techniques tend to require architects to develop complex mathematical models, which imposes steep learning curves and significant modeling effort and limits on the resulting system’s scalability [3, 11, 21]. Depending on the mathematical models they rely on, these techniques are confined to specific kinds of software system models, or are heavily dependent on error-prone and sometimes inaccessible expert inputs [10].

Dynamic analysis techniques — architectural model simulations [2, 9, 15, 18] — come with shortcomings of their own (e.g., false negatives, longer execution times), but are more capable of capturing
the randomness reflective of reality [14] and are more amenable to constructing models that are tailored to the task at hand. Despite notable efforts [7, 20, 29, 31], simulations of software architectural models have not been as widely employed as traditional static analyses [1] because creating simutable system designs is difficult [9], running simulations on complex models is time consuming and requires explicitly addressing scalability issues [23], trade-offs in system properties caused by design decisions complicate quantitative assessment [22], and analysis of system behavior may rely on massive datasets [8, 25].

3 SIMULATION-BASED SEARCH

One possible way to address the shortcomings of prior approaches is to use a search-based strategy together with architectural-model-driven discrete-event simulation to evaluate the potential systems corresponding to the model’s design decisions. Such an approach can help architects make design decisions by providing concrete simulation-based evidence on the effects each decision (and their combination) can have on the final system and its specific properties and requirements.

The challenges of such an approach include (1) enabling architects to effectively specify simutable system-design models that precisely capture the design decisions, their alternatives, and their interactions, as well as system properties of interest, and (2) scalably executing the potentially many concrete instantiations of the models with each design decision confined to a specific alternative to evaluate the decisions’ impact on system properties.

While simulating all of the potential systems that result from concrete instantiations of the decisions is feasible for real-world systems, heuristic-based search has been highly successful in relatively efficiently approximating optimal solutions [13]. Even exploring a subset of the overall search space is likely to be helpful and enable architects to make better informed decisions. Existing optimization techniques have been successfully applied to similar problems, e.g., computing the effects of possible deployment architecture on system quality of service [19]. Further, modern cloud computing enables executing thousands of system simulations in parallel, and recent advances data processing and analysis [27] can help create specialized techniques to increase the efficiency of the required analyses.

ACKNOWLEDGMENTS

This work is supported by the National Science Foundation under grants no. CCF-1564162 and CCF-1618231, and by Huawei Technologies Co., Ltd.

REFERENCES