

The Architect

Economic Modeling of Software Architectures

Rick Kazman, Jai Asundi, Mark Klein

The Cost-Benefit Analysis Method (CBAM) picks up where the Architecture Tradeoff Analysis Method (ATAM) leaves off: adding cost as an attribute to be considered among the tradeoffs when a software system is being planned.

Introduction

At the Software Engineering Institute, we have been doing analyses of software and system architectures, using the Software Architecture Analysis Method (SAAM) and the Architecture Tradeoff Analysis Method (ATAM), for more than five years. [For more on these subjects, see http://www.sei.cmu.edu/ata/ata_init.html.] When we do these analyses, we are primarily investigating how well the architecture has been designed with respect to its quality attributes (QAs): modifiability, performance, availability, usability, and so forth. In the ATAM, we additionally focus on analyzing architectural tradeoffs, the points where a decision might have consequences for several QA concerns simultaneously.

But the biggest tradeoffs in large, complex systems always have to do with economics: How should an organization invest its resources in a manner that will maximize its gains and minimize its risks? This question has received little attention in the software engineering literature, and where it has been addressed the attention has primarily focused on costs. Even in those cases, the costs were primarily the costs of building the system in the first place, and not its long-term costs through cycles of maintenance and upgrade. Just as important as costs are the *benefits* that an architectural decision may or may not bring to an organization. Given that resources for building and maintaining a system are finite, there must be some rational process for choosing among architectural options, both during initial design and subsequent periods of upgrade. These options will have different costs; will implement different features, each of which brings some benefit to the organization; and will have some inherent risk or uncertainty. Thus we need economic models of software that take into account costs, benefits, and risks.

The CBAM

For this reason, we have been developing a method for economic modeling of software and systems, centered on an analysis of their architectures. We call this method the Cost

Benefit Analysis Method (CBAM). The CBAM builds on the ATAM to model the costs and benefits of architectural design decisions and to provide a means of optimizing such decisions. A simple way to think about the objectives of this method is that we are adding money to the ATAM as an additional attribute to be traded off. We are showing how to make decisions in terms of benefits per dollars, as well as in terms of quality-attribute responses.

The CBAM begins where an ATAM leaves off and depends on the artifacts that the ATAM produces as output, as depicted in Figure 1.

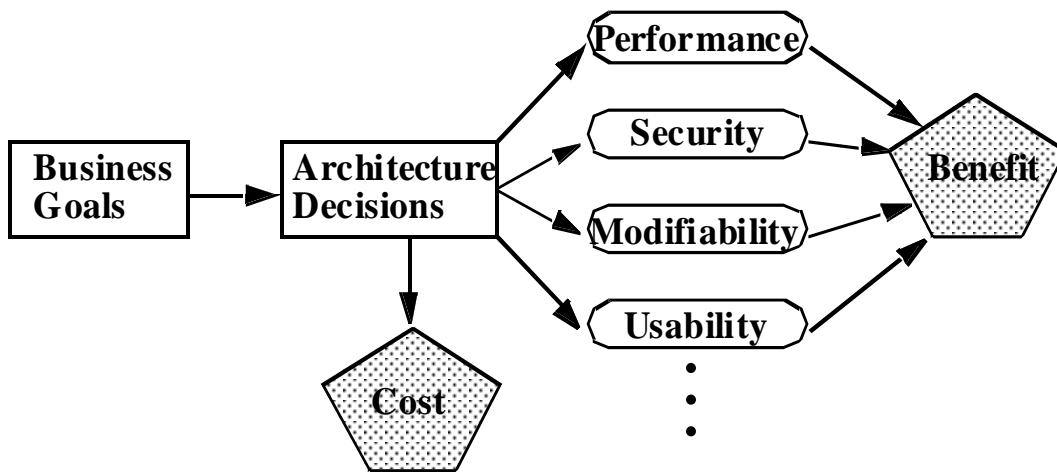


Figure 1: The Context for the CBAM

The ATAM uncovers the architectural decisions that are made (or are being considered) for the system and links these decisions to business goals and QA response measures via a set of elicited scenarios. The CBAM builds on this foundation, as shown by the shaded pentagons in Figure 1, by enabling engineers to determine the costs and benefits associated with these decisions. Given this information, the stakeholders could then decide, for example, whether to use redundant hardware, checkpointing, or some other method to address concerns about the system’s reliability. Or, the stakeholders could choose to invest their finite resources in some other QA—perhaps believing that higher performance will have a better benefit/cost ratio.

A system always has a limited budget for creation or upgrade, and so every architectural choice, in some sense, competes with every other one for inclusion. The CBAM does not make decisions for the stakeholders; it simply helps them elicit and document costs, benefits, and uncertainty and gives them a rational decision-making process. This process is typically performed in two stages. The first stage is for triage, and the elicited cost and

benefit judgments are only estimates. The second stage operates on a much smaller set of architectural decisions (also called architectural strategies), which are examined in greater detail.

There is uncertainty involved with the design of any large, complex system with many stakeholders. The uncertainty comes from three relationships:

- the uncertainty of understanding how architectural decisions relate to QA responses. That is to say, even if we are diligent in designing and analyzing our architecture, there is some uncertainty in knowing how well it will perform, adapt to change, or be secure, and there is uncertainty in understanding the environment in which the architecture will operate (e.g., knowing the distribution of service requests arriving at the system).
- the uncertainty of understanding how architectural decisions relate to cost. Cost modeling is not precise, and the best models only provide a range of cost values.
- the uncertainty of understanding how QA responses relate to benefits. Even with perfect knowledge of an architecture's responses to its stimuli and the distribution of these stimuli, it is still unclear in most cases how much benefit the organization will actually accrue from such a system.

As with the financial markets, different investments will appeal more or less to different stakeholders depending on those investments' inherent uncertainty. One function of the CBAM, then, is to elicit and record this uncertainty, because it will affect the decision-making process.

Using the CBAM

The CBAM consists of six steps, each of which can be executed in the first (triage) and second (detailed examination) phases.

1. choose scenarios and architectural strategies
2. assess QA benefits
3. quantify the architectural strategies' benefits
4. quantify the architectural strategies' costs and schedule implications
5. calculate desirability
6. make decisions

In the first step, scenarios of concern to the system's stakeholders are chosen for scrutiny, and architectural strategies are designed that address these scenarios. For example, if there were a scenario that called for increased availability, then an architectural strategy might be proposed that added some redundancy and a failover capability to the system.

In the second and third steps, we elicit benefit information from the relevant stakeholders: QA benefits from managers (who, presumably, best understand the business implications of changing how the system operates and performs); and architectural strategy benefits from the architects (who, presumably, best understand the degree to which a strategy will, in fact, achieve a desired level of a quality attribute).

In the fourth step, we elicit cost and schedule information from the stakeholders. We have no special technique for this elicitation; we assume that some method of estimating costs and schedule already exists within the organization. Based on these elicited values, in step 5 we can calculate a desirability metric (a ratio of benefit divided by cost) for each architectural strategy. Furthermore, we can calculate the inherent uncertainty in each of these values, which aids in the final step, making decisions.

Given these six steps, we can use the elicited values as a basis for a rational decision-making process—one that includes not only the technical measures of an architectural strategy (which is what the ATAM produces) but also *business* measures that determine whether a particular change to the system will provide a sufficiently high return on investment.

For more information on the CBAM, including a case study of how it was applied to NASA's ECS project, see: R. Kazman, J. Asundi, M. Klein, "Quantifying the Costs and Benefits of Architectural Decisions", *Proceedings of the 23rd International Conference on Software Engineering (ICSE 23)*, (Toronto, Canada), May 2001, 297-306.¹

¹ This paper may be retrieved on-line at:
<http://ieeexplore.ieee.org/iel5/7340/19875/00919103.pdf?isNumber=19875>

About the Authors

Rick Kazman is a Senior Member of the Technical Staff at the SEI. His primary research interests are software architecture, design and analysis tools, software visualization, and software engineering economics. He is the author of over 50 papers, and co-author of several books, including "Software Architecture in Practice", and "Evaluating Software Architectures: Methods and Case Studies".

Jai Asundi is a Visiting Scientist at the SEI with the Product Line Practice Program. His interests are in the area of economics driven software engineering, decision analysis for software systems, open-source software systems and outsourced software development. He has a Ph.D. in Engineering and Public Policy from Carnegie Mellon University.

Mark Klein is a Senior Member of the Technical Staff at the SEI. He has over 20 years of experience in research on software engineering, dependable real-time systems and numerical methods. Klein's most recent work focuses on the analysis of software architectures, architecture tradeoff analysis, attribute-driven architectural design and scheduling theory. He is co-leader of the SEI's work in the area of attribute-based design primitives. Klein has co-authored many papers and is co-author of two books including "Practitioner's Guide for Real-Time Analysis", and of "Evaluating Software Architectures: Methods and Case Studies".