

A Foundation for the Economic Analysis of Software Architectures

EDSER-3 Position Paper

Jai Asundi^{1,2}, Rick Kazman^{2,3}

¹Dept. of Engineering and Public Policy

²Software Engineering Institute
Carnegie Mellon University
asundi@andrew.cmu.edu

³Dept of Information Technology Management

University of Hawaii
Honolulu, HI
kazman@sei.cmu.edu

Abstract

This position paper is an attempt to show how real-options theory and portfolio theory—can be applied to making software investment decisions in practice. We have previously developed a method, called CBAM, for making software investment decisions based upon architectural analysis, and have applied this method to a large-scale system development—NASA’s ECS project. Building upon this experience, we reflect upon the relationship between the theory of economic models and what can be done in practice.

1. Introduction

Despite the criticality of software architectures to a system’s success, software architects do not have well developed structures to reason about the costs and benefits of the various design options available to them. The Architecture Tradeoff Analysis Method (ATAM) [6] provides a framework for software architects to reason about the technical tradeoffs faced while designing and maintaining a software system. But these technical tradeoffs have largely ignored economic considerations.

We have been developing a method for doing economic analyses of software and systems, centered around their architectures. We call this method the CBAM (Cost Benefit Analysis Method) [7]. The CBAM builds upon the ATAM adding techniques that help the analyst to better understand the benefit and cost implications of the architectural design decisions being made. Given the fact that there are large uncertainties—both technical as well as economic—during the architecture design stage of a software system, the architects will be facing risks with regards to the system’s ability to meet its business goals. One of the advantages of our approach lies in the fact that it forces the stakeholders to quantify these risks, by explicitly quantifying the benefits of architectural decisions, as well their costs, quality attribute responses, and the uncertainty surrounding these estimates.

The financial literature contains several interesting techniques such as real-options theory and portfolio theory to help decision makers manage uncertainty and risk in their investments. Given that software projects are also invest-

ments[2], then it seems reasonable that the (relatively mature) techniques outlined in the financial literature could be borrowed and applied to software projects. Sullivan [10], Butler [3] and a few others have already described how these techniques could be used in the context of software projects and how one might think about the design problem in that framework. However, we have not yet seen any example of a substantial software project actually using these techniques to help in their decision making process. We attribute this to the fact that obtaining economic data in software projects is much harder than in financial markets from where these techniques have been borrowed. In this paper we show how we can apply these techniques in practice.

2. Overview of the CBAM

The CBAM begins where an ATAM leaves off and depends upon the artifacts that the ATAM produces as output. The problem at hand is to choose from large set of desired changes to the system, each of which we call an Architectural Strategy (AS)

In the ATAM process we simply generate a list of the quality attributes of importance to the project. In the CBAM we go further, by having the stakeholders individually rate each of the attributes such that the sum of their scores is 100. The attribute ratings are denoted by $QAScore_j$, where j is the quality attribute being rated. Hence, by design, $\Sigma(QAScore_j) = 100$ and for all j , $QAScore_j > 0$. We then elicit from the stakeholders an estimate of the impact resulting from the application of each AS, in terms of how it affects each quality attribute of the system. We call these contribution scores ($Cont_{ij}$). Based upon the $QAScore$ and the $Cont$ score we can calculate a rough benefit score as follows:

$$Benefit(AS_i) = \sum_j (Cont_{ij} \times QAScore_j)$$

A desirability metric is also calculated as follows:

$$Desirability(AS_i) = Benefit(AS_i)/Cost(AS_i)$$

Figure 1 plots the ranges of benefit values against the cost values for the respective ASs. Looking at the plot we realize that ideally we would like to choose those strategies

that lie on the top left corner: strategies with a high benefit and low cost. The ovals denote the uncertainty in the elicited values of benefit and cost. We shall revisit this plot later in the paper.

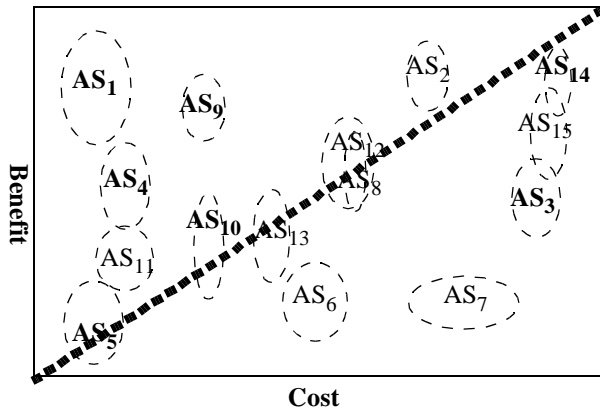


Figure 1. Plotting the benefits and costs of architectural strategies

3. Software Projects as Investments Under Uncertainty

The development of a software product can be viewed as an investment today with an expectation of benefits or rewards in the future. The investment in software projects, just like many other investments, have a characteristic that they are largely *irreversible*, have considerable *uncertainty* and there is some amount of leeway as to the *timing* of the investment.

3.1. Advanced economic methods

The economics and finance literature discusses a number sophisticated techniques [5] that have caught the interest of many in the software engineering community. For example, the use of real-options theory (Sullivan [10]) and portfolio theory (Butler [3]) for software design have been suggested. Benaroch [1] in their study of information systems use the real-option formulation to analyze a decision problem

More research needs to be carried out to both operationalize and validate the use of these economic theories in software investment decisions. This, however, requires large-scale case studies, which are rare to come by. We were extremely fortunate to have NASA's ECS project willing to participate in the experimental application of these techniques within the framework of the CBAM.

We believe that addressing the software investment problem at the architectural level of abstraction will make the application of these sophisticated techniques easier. The reason for this is obvious: architecture is an appropriately

high level of abstraction at which to think about strategic *investment* decisions. A level at which one could reason about the impact of decisions to the business goals of the system via the quality attributes.

3.2. The real-options approach

Traditional investment decisions for engineering projects have been based on a simple rule that a project is undertaken only if the *Net-Present Value* (NPV) is greater than zero.:

$$NPV = \sum_{t=0}^T \frac{(\text{Benefit}_t - \text{Cost}_t)}{(1+r)^t} > 0$$

This simply means that the benefit derived from the project is greater than the cost incurred. However, the implicit assumption of this formulation is that the investment is reversible and that the decision to be made is now-or-never¹. We know that this is not the case: costs in software development are irreversible (one can't, in general, go to the market and sell off the implemented module) and project managers usually have some leeway with respect to the timing of project's implementation. It is possible that the manager could *postpone* the decision of implementation until he receives additional information about the benefit he could derive from any particular AS. By waiting an additional amount of time he could obtain a higher return on investment at the cost of the lost convenience yield (the amount of benefit gained per year due to implementing the system immediately).

The insight gained from the real-option formulation is that we want to postpone a decision when the NPV is less than the value of the option to delay. We can estimate these values based upon the information that we elicit in the CBAM process because we are already estimating the costs and expected benefits as part of the elicitation process.

3.3. An idealized view of the real-options approach

In real-options literature [5], the benefit or value is assumed to follow a Wiener process which is easiest represented by stochastic equations (also known as *brownian motion with drift*):

$$dB = B\alpha_1 dt + B\sigma_1 dz$$

$$dC = C\alpha_2 dt + C\sigma_2 dz$$

where B is the benefit, C is the cost, α is the drift parameter (i.e. the amount by which the value steadily increases) and σ is the variance parameter. Similarly if the cost of investment also varies stochastically over time (which we would argue is the case with software projects) then it is

1. In the case of pure uncertainty, then expected values of Benefit and Cost are used to calculate NPV

shown that the decision rule is defined by:

$$\frac{B}{C} \geq p^* = \frac{\beta_1}{(\beta_1 - 1)} \delta_B > 1$$

where δ_B is the convenience yield on the asset and β_1 is a function of variance in benefit, cost, the correlation between the two and also the yields on assets and costs². This equation suggests that unless the value of the asset (software project in our case) is p^* times the cost of investment, the project manager should not go ahead with implementation of the project. The fact that $p^* > 1$ shows that there is an additional ‘safety-factor’ involved in making decisions under uncertainty and this is a powerful result. In terms of the example that we have been pursuing here, p^* is shown as the dashed line in Figure 1. ASs significantly above that line should be immediately implemented while those below should be postponed.

From our experience with the ECS project in eliciting benefit and cost information we realized that gathering data that directly fits into models described in financial literature is difficult. We have some information, like the Benefit and Cost estimates. However, to calculate β_1 and the convenience yield, δ_B , will prove to be challenging. Software experts typically have little idea about the true meaning of “discount rate” for the software project or what, if any, is the “dividend” payable by a software project and what “correlation” one can attribute to various ASs.

3.4. Using a portfolio approach

The idea behind the portfolio approach to investment is to reduce the overall risk (uncertainty) in returns by investing in assets that do not have a strong positive correlation [4]. Consider two assets A and B, with returns r_A and r_B that have standard deviation σ_A and σ_B respectively. If we invested some fraction x_A of our resources on A and $(1-x_A)$ on B, then the overall return, R_{AB} , we would obtain from our ‘portfolio’ would be:

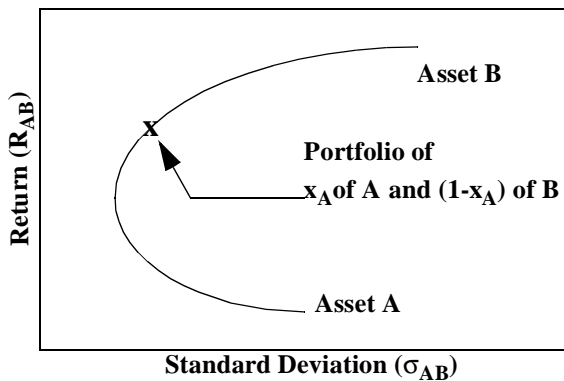


Figure 2. A portfolio of two assets

$$R_{AB} = x_A * r_A + (1-x_A) * r_B,$$

and the variance, and hence risk, would be given by:

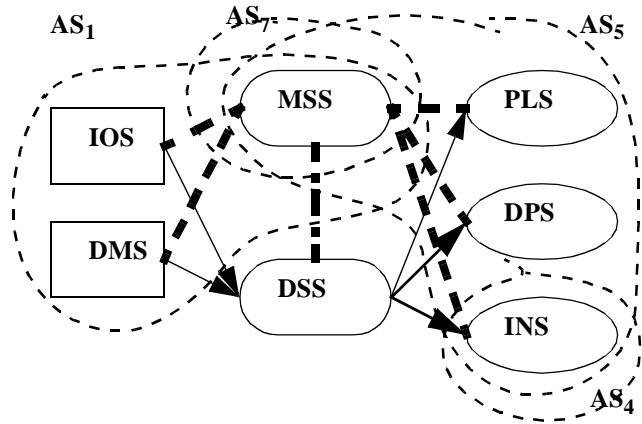


Figure 3. Dependency/Correlation between ASs

$$\sigma_{AB}^2 = x_A^2 \sigma_A^2 + (1-x_A)^2 \sigma_B^2 + 2x_A (1-x_A) \rho_{AB} \sigma_A \sigma_B$$

where ρ_{AB} is the correlation in returns between asset A and asset B. Figure 2 shows the curve for all values of x_A on a plot of returns versus standard deviation. From the plot, B is a high risk asset, but yields a higher return whereas A is a low risk but lower return asset. Thus we see that we can choose a particular x_A for which the resultant portfolio has a lower variance or risk than for either of the assets independently. The return is a weighted average of the two investments. If one chooses a risk level then the return can be calculated. This is the basic principle behind investing in a portfolio of assets. Portfolio theory is suitable for cases where one has to follow a risk mitigating or minimizing strategy while providing a reasonable return.

It is well understood that in many software projects, managers wish to minimize the risk of choosing a particular technology if it is uncertain. Concerns of inter-operability, vendor survivability, and market acceptance are foremost in their minds. Portfolio theory provides an ideal framework for managers to manage the uncertainty and hence the risk of their decisions.

Now how do we apply this theory to architectural design? Considering that our problem is one of choosing a small set of ASs for implementation, we can restate this problem in the portfolio theory terms as follows: we have a particular architectural strategy, AS_1 , that uses a new technology. If this technology gains widespread market acceptance then it has a potentially high benefit but with relatively large uncertainty and moderate cost. Another architectural strategy, AS_7 , is our customary way of implementing the feature with moderate benefit and cost, but low uncertainty. AS_1 is thus a high risk option while AS_7 is a low risk option.

So to operationalize portfolio theory in the context of the CBAM we need three types of information: returns, variances, and correlations among ASs. We have already captured information on returns and variances in the CBAM in the form of benefit information and its associated uncertainty. Hence to use portfolio theory, we only need to introduce the concept of a “dependency structure” between the various ASs under consideration as a proxy for correlation information. Figure 3 depicts how the various ASs poten-

2. For a detailed analysis refer to [5] pp. 207-11

tially affect the components of the system architecture. We can see that AS₇ and AS₁ overlap—they affect some of the same components. But since they are implementing competing technologies, then they are considered to be negatively correlated (i.e. the success of one virtually guarantees the failure of the other). If we consider AS₁ and AS₅, we see that they also overlap in the sense that they both affect the MSS component. So how do we use this information to determine the correlation between the ASs?

Depending on what each of the ASs are trying to accomplish and which portions of the components they affect we can elicit a proxy measure for the correlation. For strategies that have no obvious dependency we can assume a zero correlation. ASs that are subsets of others can be considered as having a correlation of 1. As part of the CBAM we elicit correlations between ASs. We can use correlation information to populate a matrix as shown in Table 1.

Arch Strategy	AS1	AS4	AS5	AS7
AS1	1.0	0.0	0.4	-1.0
AS4		1.0	1.0	0.0
AS5			1.0	-0.3
AS7				1.0

Table 1: Correlation between ASs

The optimization problem thus becomes that of minimizing risk, given the constraints of total investment (budget) and the returns and uncertainties of the various ASs. We can thus choose a portfolio of strategies based upon the benefit scores we elicited in Section 3 and the correlation coefficients that we have elicited here. The resultant set of ASs may not be the best in terms of return/benefit, but it attempts to minimize the project’s overall risk.

When combining the various strategies to form a portfolio, one needs to look at their structural overlap, their technical overlap, the technical feasibility of implementing all of them at the same time, as well as the implied additional schedule and budget considerations.

3.5. A hybrid technique (options+portfolio)

The application of the pure form of options or portfolio theory to making a design decision is possible in some cases. However, in the case of ECS—and, we believe, most systems—we need to apply them in a hybrid form. There are some subsystems where a risk mitigation strategy is required, while in other subsystems, a maximization strategy is appropriate.

In the situation where a project is considering many architectural strategies, we can apply the real-options formulation based upon the dependency structure of the strategies. In our example, AS₄ and AS₅ are two architectural strategies that overlap completely. In fact, AS₄ is a subset of AS₅ (hence the correlation of 1 in Table 1). Analysis of AS₄ and AS₅ shows that to implement AS₅, AS₄ must first be

implemented (i.e., there is a hierarchical dependency). The problem can thus be framed in the real-option framework by considering that the price of the option to postpone AS₅ is the cost of implementing AS₄. Since AS₄, when implemented, will provide more information about AS₅ it will either reduce the uncertainty regarding the benefits that can be obtained or provide the waiting period for any exogenous factors to take effect while keeping the option to implement AS₅ open.

Hence a strategy that the designers could adopt while making a choice of ASs, is to choose a set of ASs that tries to minimize the risk using the portfolio framework and then deciding on which ASs to *postpone* based on the uncertainty and correlation of the ASs. New ASs could be added to the chosen set (depending on resource availability) to provide the *real-option* of postponing some uncertain AS. These additional ASs that provide the option would necessarily have to be either subsets of the original strategy or strongly correlated to it.

3.6. Assumptions of these technique

There are a number of assumptions that are made to make the analysis of real-options and portfolios easier. These assumption may hold in the financial world, but when considering software projects, they may seem tenuous.

One of the basic assumptions is that there is no arbitrage in the financial markets. Now within software projects, we cannot think of trading individual software components/modules! The non-trading of assets makes it difficult for one to claim that the resultant strategy is “optimal”. However, we can argue that economic strategies provide us with guidance and help us build better but not necessarily optimal systems.

Another assumption made is that the various parameters like discount rate, drift rates and convenience yields are exogenous to the system. In our case, the context dependence of the problem results in varying values of these parameters and hence leads to endogeneity.

4. Conclusions and Future Work

The current form of the CBAM is a structured elicitation process within a probabilistic Cost-Benefit analysis framework combined with a decision analysis framework [8]. This blend of techniques aids us in making software and system architecture decisions. The techniques of decision analysis are used to define the design space and economic techniques are used to maximize the sum total expected utility (*Desirability*) of all proposed architectural changes.

We have explained how advanced economic techniques—real-options theory and portfolio theory—can be practically applied to software engineering projects. An area of future work is in the continued application of these techniques to software projects and obtaining more data for

its validation. Solving a problem in theory and in practice are very different. We need more experience in applying these techniques in practice to guide us in optimizing them.

One important area we see is for firms that develop software products to increase their institutional capability with regards to economic modeling. Understanding the relationship between their business goals, quality attributes and the utility they derive from them will be a first step. Over time, if a firm were to develop models (even if they were heuristic) relating a quality attribute response to utility, or benefit, then the elicitation exercise would be much easier and less subjective.

Similarly we can imagine cost models to be built from empirical data so that the economic tradeoffs involved are easily identified. Cost models that are “architecturally aware”, i.e. those that consider architectural components as units and consider the architectural design itself as affecting the overall system cost, would be useful in the context of the CBAM.

Have we shown that economic techniques are appropriate and advantageous for software decision making? Yes. Have we proven that they are optimal? No. Such a proof is extremely difficult to come by. However, we have shown that such techniques can be practically applied given information that we can and do currently elicit from real-world software projects. We believe that the application of economic techniques is inherently better than the ad-hoc approaches that projects (even quite sophisticated projects) employ today. We cannot prove the optimality of our techniques—these techniques are optimal in theory, but the application of them introduces some uncertainty. Because we use expert judgements our techniques may not be optimal but they are at least satisficing.

Furthermore our experience with the CBAM (as well as earlier experience with the ATAM) tells us that just giving people the appropriate tools to frame and structure their decision making process is already an enormous benefit.

5. Acknowledgments

We would like to thank Linda Northrop and the staff of

the SEI’s ATA initiative and Mike Moore and the staff of NASA’s ECS project for their support. We also wish to thank Benoit Morel of EPP, CMU for helping us grapple with the finer points of Real-options and Portfolio Theory.

6. References

- [1] Benaroch, M. and Kaufman, R.J., “A Case for Using Real Options Pricing Analysis to Evaluate Information Technology Project Investments”, *Information Systems Research*, Vol. 10, No. 1, March 1999, pp70-86.
- [2] Boehm, B. *Software Engineering Economics*, Prentice Hall, 1981.
- [3] Butler, S., Chalasani, S., Jha, S., Raz, O. and Shaw, M. “The Potential of Portfolio Analysis in Guiding Software Decisions”, *First Workshop on Economics-Driven Software Engineering Research*, <http://www.cs.virginia.edu/~sullivan/EDSER1/>.
- [4] Brealey, R., Myers, S., *Principles of Corporate Finance*, McGraw-Hill, New York, 1981.
- [5] Dixit, A., Pindyck, R. S., *Investment Under Uncertainty*, Princeton University Press, New Jersey, 1994.
- [6] Kazman, R., Klein, M., Clements, P., “ATAM: A Method for Architecture Evaluation”, CMU/SEI-2000-TR-004, Software Engineering Institute, Carnegie Mellon University, 2000.
- [7] Kazman, R., Asundi, J., Klein, M., “Quantifying the Costs and Benefits of Architectural Decisions”, *Proceedings of the 23rd International Conference on Software Engineering*, Toronto, Canada, May 2001, to appear.
- [8] Mishan E. J., *Economics for Social Decisions: Elements of Cost-Benefit Analysis*, Praeger, New York, 1973.
- [9] Morgan, G., Henrion, M., *Uncertainty: A Guide to dealing with Uncertainty in Quantitative Risk and Policy Analysis*, Cambridge University Press, 1990.
- [10] Sullivan, K.J., Chalasani, P., S. Jha, S., Sazawal, V., “Software Design as an Investment Activity: A Real Options Perspective,” in *Real Options and Business Strategy: Applications to Decision Making*, (L. Trigeorgis, ed.), Risk Books, 1999.