Quantifying the Costs and Benefits of Architectural Decisions

Rick Kazman[1], Jai Asundi[1,2], Mark Klein[1]
[1]*Software Engineering Institute*
[2]*Dept. of Engineering and Public Policy*
Carnegie Mellon University
Pittsburgh, PA 15213
kazman@sei.cmu.edu, asundi@andrew.cmu.edu, mk@sei.cmu.edu

Abstract

The benefits of a software system are assessable only relative to the business goals the system has been developed to serve. In turn, these benefits result from interactions between the system's functionality and its quality attributes (such as performance, reliability and security). Its quality attributes are, in most cases, dictated by its architectural design decisions. Therefore, we argue in this paper that the software architecture is the crucial artifact to study in making design tradeoffs and in performing cost-benefit analyses. A substantial part of such an analysis is in determining the level of uncertainty with which we estimate both costs and benefits. In this paper we offer an architecture-centric approach to the economic modeling of software design decision making called CBAM (Cost Benefit Analysis Method), in which costs and benefits are traded off with system quality attributes. We present the CBAM, the early results from applying this method in a large-scale case study, and discuss the application of more sophisticated economic models to software decision making.

1. Motivation

At the Software Engineering Institute we have been doing analyses of software and system architectures, using the Software Architecture Analysis Method (SAAM) [9] and the Architecture Tradeoff Analysis Method (ATAM) [8], for more than five years. 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 places 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 risk? This question has received little attention in the software engineering literature in the past, and where it has been addressed it has primarily been focused on costs [1], and even then these costs are 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, during both initial design and its 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.

For this reason we are developing a method for doing economic modeling of software and systems, centered around an analysis of its architecture. We call this method the CBAM (Cost Benefit Analysis Method). The CBAM builds upon the ATAM to model the costs and the 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 dollars 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.

Related to this work is the Next Generation Process Model (NGPM) [2]. This model helps the designers and maintainers of a software system to converge on the system's next-level objectives, constraints, and alternatives. The NGPM uses the Theory W, which involves identifying the system's stakeholders and their respective win conditions. Using a negotiation process they determine a mutually satisfactory set of objectives, constraints and alternatives.

In this paper we will describe the CBAM and discuss its application to NASA's Earth Observing System Data Information System (EOSDIS) Core System (ECS) project. The ECS is distributed data information system consisting of about 1.1 million lines of code responsible for acquiring, managing, and distributing very large volumes of climate-related data throughout the world.

Finally, we will show how some techniques from traditional economic theory are applicable and, in fact, necessary for modeling software investment decisions.

0-7695-1050-7/01 $10.00 © 2001 IEEE

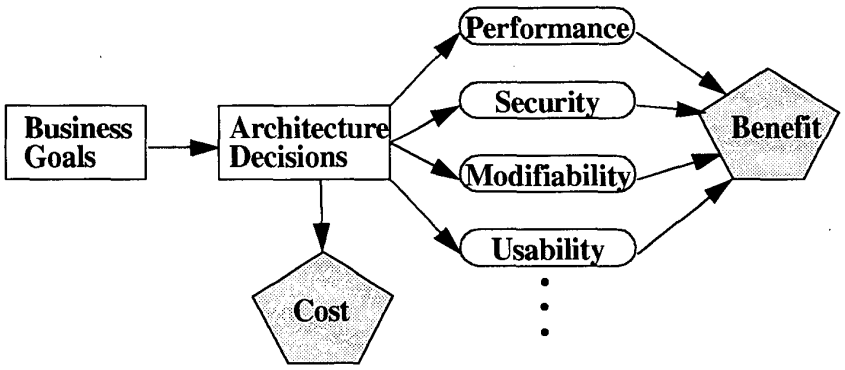

Figure 1. Context for the CBAM

2. Context for the work

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

2.1. ATAM outputs

When an ATAM is completed, we expect to have a set of artifacts documented as follows [7]:

- a description of the *business goals* that are crucial to the success of the system
- a set of *architectural views* that document that existing or proposed architecture
- a *utility tree* which represents a decomposition of the stakeholders' goals, for the architecture. The utility tree starts with high-level statements of QAs and decomposes these into specific instances of performance, availability, etc. requirements and realizes these as scenarios
- a set of *risks* that have been identified
- a set of *sensitivity points* (architectural decisions that affect some QA measure of concern)
- a set of *tradeoff points* (architectural decisions that affect more than one QA measure, some positively and some negatively)

The ATAM determines a set of key potentially problematic architectural decisions, based upon the stakeholders' business goals. These architectural decisions result in some specific QA *responses*: a particular level of performance, security, usability, modifiability, and so forth. But those architectural decisions also have an associated cost. For example, if an architectural decision is made to use redundant hardware to increase reliability, then this has one cost consequence. If checkpointing to a disk file is used instead, then this architectural decision will have a different cost. Furthermore, both of these architectural decisions will result in a measurable level of reliability (perhaps measured as mean time to failure or steady-state availability). These QA *responses* will have some value to the developing organization. Perhaps the organization believes that its stakeholders will pay extra for a highly reliable system (a telephone switch, for example) or that the organization will get sued if the system is not highly available (a medical monitoring device for example).

The ATAM uncovers the architectural decisions made and links them to business goals and QA response measures. The CBAM builds on this foundation by filling in the shaded pentagons in Figure 1, determining the costs and benefits associated with these decisions.[1] Given this information, the stakeholders can then decide whether to use redundant hardware, checkpointing, or some other architectural decision addressed at increasing the system's reliability. Or the stakeholders can 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 is, in some sense, competing with every other one for inclusion.

The CBAM does not make decisions for the stakeholders; it simply aids them in the elicitation and documentation of costs, benefits, and uncertainty and gives them a rational decision-making process. It also aids them in doing triage, as we shall explain.

2.2. Triage

When dealing with a major system design or upgrade, the number of improvements desired by the stakeholders and the space of possible architectural options for meeting those improvements is potentially huge. In our case study, as we shall show, the architects were considering 67 different scenarios, most of which required distinct sets of code

1. The CBAM does not include a new way of determining costs (although we think that an architecture-aware cost estimation method is a desirable goal). It assumes that *some* method of cost estimation already exists within the organization.

and, in some cases, architectural modifications.

To wade through such a large space of possible changes meant that we need to take a two phase approach. In the first phase of the CBAM we do triage, determining costs and benefits only very roughly. In the second phase we do far more detailed examination of a selected subset of the architectural approaches that appear to be promising.

2.3. Dealing with uncertainty

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

- the uncertainty of understanding how architectural decisions map onto 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 or adapt to change or be secure and there is often uncertainty in the understanding of 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 map onto cost. Cost modeling is not precise, and the best models only give a range of cost values (e.g. [1] and [5]).
- the uncertainty of understanding how QA responses map onto benefit. Even if one had 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 their inherent uncertainty. Part of the CBAM, then, is to elicit and record this uncertainty because it will affect the decision-making process. Eliciting and validating uncertainty is the difficult part [11]; our specific techniques for accomplishing this will be discussed in the next section.

3. The steps of the CBAM

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

1. Choosing Scenarios and Architectural Strategies
2. Assessing QA Benefits
3. Quantifying the Architectural Strategies' Benefits
4. Quantifying the Architectural Strategies' Costs and Schedule Implications
5. Calculate Desirability
6. Make Decisions

Each of the following subsections will describe these steps in detail.

3.1. Choosing scenarios and architectural strategies

One of the outputs of the ATAM is a prioritized set of scenarios that describe how the system should respond, react, be modified, etc. These scenarios are accompanied by specific stimuli and response goals. In this step we select the set of high importance scenarios for which improvement is desired, their QA response goals, and their currently associated architectural decisions.

For each high importance scenario for which improvement is desired we need to describe a set of possible architectural strategies (AS): changes to the existing architectural design. At the end of this step, then, we have a set of desired improvements to the system, and for each improvement we have an enumeration of the affected portions of the existing architecture and a description of one or more ASs for realizing the improvement.

3.2. Assessing quality attribute benefits

To aid in decision making, the CBAM needs to determine both costs and benefits. Determining costs is a well-established component of software engineering and, as stated above, is outside the scope of the CBAM. Determining benefits is less well-established. As a means of determining the benefit of an individual AS, a benefit evaluation function needs to be created. Benefit should be correlated with the degree to which architectural strategies support QA goals, which in turn relate back to business goals. These goals are both outputs of the ATAM.

In the CBAM we must use this information to determine the relative value or importance of each QA in the architecture. To do this we have each of the stakeholders assign a *quality attribute score* (*QAScore*) to each QA system goal (performance, interoperability, modifiability, availability, security, etc.). We let the customer determine which stakeholders should be in a decision-making capacity. From our experience in conducting ATAMs we know that this is typically a small team consisting of the project manager, the architect, and a few customer/user representatives. We ask the stakeholders to choose these scores so that they total 100. For example:

- Performance: 25
- Security: 20
- Modifiability: 20
- Availability: 5
- Interoperability: 15
- Integrability: 15

We also have each stakeholder articulate the aspect of the quality attribute that lead to their score. For example, while security might have a score of 20, it is the *data confidentiality* aspect of security that is the primary determinant of this score.

3.3. Quantifying the architectural strategies' benefits

We then use these scores to evaluate each of the individual architectural strategies. Very rarely does an AS only affect a single QA. ASs will have effects on multiple QAs, some positive and some negative, and to varying degrees. To capture this, we ask the stakeholders to rank each AS in terms of its *contribution* (*Cont*) to each QA on a scale of –1

to +1. A +1 means that this AS has a substantial positive effect on the QA (for example, an AS under consideration might have a substantial positive effect on performance) and a -1 means the opposite. Based upon this information each AS_i can now be assigned a computed benefit score from –100 to +100 "Benys" using the following formula:

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

For example, given the QAscores listed above, we can calculate benefit scores for two hypothetical ASs as follows (note that we only consider the QAs for which there is a non-zero contribution):

AS1: Performance(1.0), Security (-.05), Availability (-0.6), Modifiability(-0.4)

Benefit(AS1)= (1*25) + (-0.5*20) + (-0.6*5) + (-.04*20)
= 4

AS2: Performance(-0.4), Interoperability(1.0), Integrability(0.8), Modifiability(1.0)

Benefit(AS2)= (-0.4*25) + (1*15) + (0.8*15) + (1*20)
= 37

This score allows us to rank the benefit of every architectural change that has been contemplated. But clearly this evaluation is fraught with the uncertainty that we described above. How do we capture this uncertainty?

Our approach to capturing uncertainty is to use stakeholder judgement variations as a measure of uncertainty. Consider the stakeholder judgements listed in Table 1:

Quality Attribute	Stake-holder 1	Stake-holder 2	Stake-holder 3
Performance	0.9	1.0	0.8
Security	-0.5	-0.6	-0.4
Modifiability	**-0.1**	**-0.8**	**-0.3**
Availability	-0.6	-0.6	-0.4

Table 1. Capturing stakeholder uncertainty

These numbers are typical of what is elicited from stakeholders. While they are frequently in accordance there are times when they show great differences of opinion, as in the stakeholders' scores for the Modifiability implications of an AS in Table 1. We can use Kendall's concordance coefficient [12] (similar to Spearman's correlation coefficient but it is a non-parametric measure) for the group as a whole as a measure of the uncertainty of the group. The more highly correlated the group, the higher the concordance coefficient and hence the lower the uncertainty. For example, considering all of their QA judgements, the stakeholders in Table 1 above have a Kendall's coefficient (W) of 0.6905, an indicator of a low level of concordance (F-stat=4.461, p=0.08).

We can perform the same test with respect to each stakeholder's *QAScores* and with respect to their computed *Benefit* score for each AS.

Of course, some of the variation can be removed by ensuring that the stakeholders mean the same thing when they say, for example, "performance" and by reminding them of what was stated earlier about the business goals of the system and how those relate to QA goals. But some of the uncertainty is endemic. People evaluate the chance of success differently, or judge the benefits differently. These differences in judgement are a true characterization of the group's uncertainty and this needs to be captured as an aid to making business decisions.

Now consider that we have two sources of uncertainty here: the variation in stakeholders judgements in *QAScores* and *Cont*ribution scores. Rather than analyzing these separately, we must combine them into a single *Benefit* score and examine the uncertainty associated with variation in *Benefit*. This is because, while we could compute a mean and standard deviation for each *QAScore* and $Cont_{ij}$ value or a concordance coefficient for each set of *QAScore* and $Cont_{ij}$ values, it is not clear what it would mean to multiply the uncertainty values together when we produce the *Benefit* score. We choose instead to compute Benefit scores for each AS for each stakeholder and then look at the concordance coefficient of these values.

3.4. Quantifying the architectural strategies' costs and schedule implications

The previous step calculated a measure of benefit for each architectural strategy under consideration. Next we must calculate the expected cost of implementing each architectural strategy ASi that results in the expected benefit. This cost estimation might be extremely crude in the triage step—perhaps nothing more than a (High, Medium, Low) judgement. Since detailed cost estimation is relatively time consuming, this effort should only be spent on the subset of ASs that appear to be promising.

In addition to eliciting the costs and benefits of the ASs under consideration, prudent planning dictates that we estimate the schedule implications of each ASi in terms of elapsed time, shared use of critical resources, and dependencies among implementation efforts. Perhaps an AS is otherwise desirable, but does not fit in with the organization's time-to-market goals. During this step we will note any contention for shared resources among these estimates (hardware, software, or personnel), for these will also affect the feasibility of an AS.

3.5. Calculate desirability

We are now in a position to calculate a *desirability metric* by which the various ASs that have passed the triage stage can be compared. We can use this metric to rank these ASs. The metric is as follows:

Desirability(ASi) = Benefit(ASi)/Cost(ASi)

where *Benefit* and *Cost* are taken as mean values and *Desirability* is in the units Benys/$.

Now the ASi may be ranked according to this metric. Using this metric, the schedule constraints, and a characterization of the uncertainty, we are now in a position to make decisions.

3.6. Make decisions

At this point we understand the costs, benefits, and schedule implications of each proposed AS. In addition to looking at the *Desirability* metric, we can plot the ASs according to their benefits and costs, as shown in Figure 2. ASs which are chosen for development are shown in bold. Surrounding each AS is the uncertainty region, where the vertical span indicates uncertainty with respect to benefits and the horizontal span indicates uncertainty with respect to cost. We choose to depict these regions as ellipses, although if the number of samples is small (and it typically is, since one would normally have only a few stakeholders making these judgements) then it is feasible to simply represent every data point.

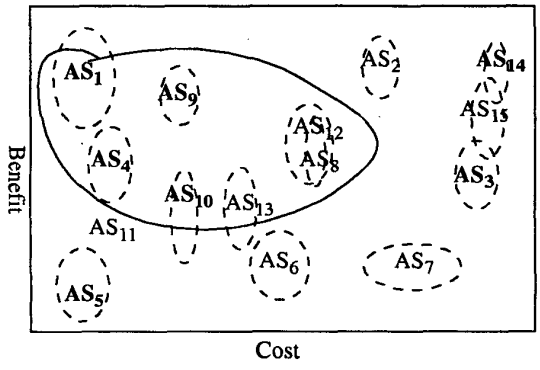

Figure 2. Plotting the costs and benefits of architectural strategies

Frequently it is the case that certain changes are dictated by external forces—keeping up with a competitor, being forced to port to newer hardware, meeting government regulations, complying with standards, etc. Thus some of the ASs may be non-negotiable, as indicated by AS_{14} and AS_3, in Figure 2. Of course, if these ASs are non-negotiable, it is reasonable to assume that they bring a high benefit to the organization. In any case, these ASs simply *will* be chosen and so there is no point in doing further decision analysis on them.

Next, the set of high benefit, low cost (i.e. high *Desirability*) ASs will be examined, as indicated by the region circled by a solid hand-drawn line in Figure 2. Some of these may be excluded because of time-to-market or resource conflicts. Additionally, some ASs that are outside this region may be included, because of dependencies (i.e. AS_4 might depend upon AS_5, which would not otherwise have been included; in such cases the desirability of the total set of architectural strategies needs to be considered). While examining this region, it is important to consider the uncertainty field surrounding each AS. For example, consider AS_8 and AS_{12}. While AS_{12} has a higher mean Benefit score than AS_8, it also has a greater span of uncertainty in both its costs and its benefits. In fact, its uncertainty region almost completely encompasses that of AS_8. Thus when choosing between the two, one might prefer the more certain AS_8 to the riskier AS_{12}. This is why we must consider and be able to quantify uncertainty when assessing costs and benefits. Without this assessment, AS_{12} would have been the "obvious" choice on the basis of its mean scores.

3.7. Analyzing the data

Initially, when analyzing the data elicited from the stakeholders, we calculated the means and standard deviations of the data: the *QAScores*, the *Contribution* values, and the derived *Benefit* and *Desirability* scores. However, we quickly abandoned this line of analysis for two reasons: 1) because we had no way of verifying the underlying assumption of a uniform distribution of judgements; and 2) given the small numbers of stakeholders, we can just look at all the data, and so knowing the standard deviation is irrelevant. Knowing the distribution of the data points is more interesting, and this can be seen by simply plotting *all* of the elicited stakeholder judgements.

One type of statistical analysis that we have found important to do is looking at the degree of *concordance* in stakeholder judgements. It is important to not only measure the degree of uncertainty for economic reasons (so that the stakeholders can make decisions based upon complete evidence of costs, benefits, and risks), but also so that we can, over time, build up a set of measurable expectations for the performance of the stakeholders in making judgements. For example, one use of the Kendall concordance coefficient to tell stakeholders that they need to go back to the drawing board and talk to each other more, or do more prototyping, or investigate the underlying technologies (because their judgements are not highly correlated). But we can also use it to tell the stakeholder group where they stand with respect to other stakeholder groups, historically speaking, in terms of their level of concordance. We do not have a basis for making these judgements yet, but as we perform more CBAM exercises we will quickly be able to develop a set of "norms" against which we can compare stakeholder performance.

3.8. Interim summary

At the end of the CBAM exercise, we have guided the stakeholders to determine a set of architectural strategies that address their highest priority scenarios. These chosen strategies furthermore represent the optimal set of architectural investments. They are optimal based upon considerations of: benefit, cost, schedule, within the constraints of the elicited uncertainty of these judgements and the willingness of the stakeholders to withstand the risk implied by uncertainty.

4. An example: the ECS

We have applied the CBAM to NASA's ECS project, a very large-scale information system. The goal of the system is to collect climate-related data using a variety of sensors on several satellites and make the data widely available world-wide 24 hours per day in various forms in support of inter-disciplinary earth science.

The system is geographically distributed among four major sites, but the same version of the software runs on each of the sites. Data volume is large and in recent history doubling each year. The system gathers over 100 gigabytes of data per day and manages hundreds of terabytes of data in total.

The system contains approximately 1.1 million lines of custom code in about 12,000 modules and employs about 50 COTS products. The ECS is in its maintenance phase but is in the process of planning major upgrades to its capabilities and so we first applied the ATAM to scrutinize both the existing architecture and the architectural decisions being contemplated. When the ATAM was complete, we had an understanding of the business case, a high-level understanding of the architecture, and had generated utility tree and sets of scenarios, risks, sensitivity points, and tradeoffs.

4.1. Choosing scenarios and architectural strategies

This was the starting point for the CBAM. The architects were faced with 67 scenarios for their consideration—an overwhelming and infeasible number. Each of these scenarios would bring value to NASA and the overwhelming majority were addressed by distinct ASs. In fact, there were 58 distinct ASs put forth by the architects for consideration. Except for one AS which was deemed to be crucial and non-negotiable, most of them needed to be analyzed with respect to their costs and benefits, to determine where NASA should invest in the ECS.

4.2. Assessing quality attribute benefits

The stakeholders embarked upon an exercise wherein they assigned a *QAScore* to each of eight qualities. The seven stakeholders each scored the QAs independently. These scores were then averaged. The results were as shown in Table 2, where each column represents the set of judgements of a single stakeholder (with the mean shown, for information purposes, at the right).

Quality	Stakeholder							Avg
	1	2	3	4	5	6	7	
Operability	18	20	15	20	30	20	15	19.7
Maintainability	17	15	25	20	30	25	15	21.0
Scalability	15	10	5	10	5	0	20	9.3
Performance	15	15	20	10	10	0	10	11.4
Extensibility	8	10	15	5	5	15	10	9.7
Reliability	7	15	10	20	15	20	15	14.6
Security	5	5	5	5	0	0	5	3.6
Usability	15	10	5	10	5	20	10	10.7

Table 2. Elicited *QAScores*

The names of the QAs that the stakeholders chose, and precisely what they meant by them are not important for the purposes of this exposition. What *is* important is that they agreed on these meanings and furthermore explicitly traced these meanings back to their business goals when determining their relative importance.

When we computed *W*, Kendall's Coefficient of Concordance on these rankings (considering the *QAScores* as rankings, rather than as values) the result was:

W = 0.8383

which shows a high degree of concordance among the 7 stakeholders (F-stat=31.0989). This shows significant agreement among the stakeholders at the .0001 level. This doesn't mean that there was no uncertainty, but it is simply showing that the rank orderings of the *QAScores* were significantly correlated among the stakeholders.

4.3. Quantifying the architectural strategies' benefits

The stakeholders then ranked the contribution (*Cont*) of each AS and we calculated a mean *Benefit* score for each AS. We also recorded the individual stakeholders rankings, as these determine the uncertainty in the *Benefit* score.

The mean *Benefit* scores elicited from the stakeholders ranged from a (surprising) low of -17.5 to a high of 70.8, with a mean of 30.6.

For example, here are the scores reported by stakeholder 1 for AS0140 "Replace DCE with TCP/IP":

QAScores = {Maintainability(0.8), Operability(0.9), Reliability(0.8), Scalability(0.6), Extensibility(1.0)}

Benefit = 0.8*17 + 0.9*18 + 0.8*7 + 0.6 *15 + 1*8
= 52.4

Table 3 gives the 7 stakeholders' mean *Benefit* scores for the 6 highest ranked of the 57 ASs that were under consideration for the ECS. In addition to producing these mean scores we compute *W* for each of these, as a means of gaug-

ing overall concordance.

Architectural Strategy	*Benefit* Score
AS0100	69.6
AS0140	70.2
AS0150	67.5
AS0160	61.6
AS1088	70.4
AS1120	52.2

Table 3. The top 6 mean *benefit* scores

4.4. Quantifying the architectural strategies' costs and schedule implications

Next, the costs and schedule implications of each AS were assessed. Given that this was a triage effort, the costs were simply estimated on a 1-100 scale. What was important here was the relative range of costs and not their absolute values, since this was input to the triage decision. Also, the schedule implications were estimated for each AS.

Finally, a table of dependencies among the ASs was created, which lists 44 dependencies among the 57 ASs. These artifacts are not individually complex, but are important in making the final decisions of which ASs to implement

4.5. Calculate desirability

The *Benefit* and *Cost* scores resulted in the ability to calculate *Desirability* metrics for each AS. When thinking about desirability we are theoretically interested in the sign and the magnitude of the ratio (the magnitude can range from 0 to 100). Obviously we prefer positive values to negative ones and large numbers to small ones. In practice, however, we only concern ourselves with ASs that have a positive *Desirability* value.

In the ECS's case the maximum estimated mean *Desirability* value was 10.83, achieved by AS1000 (not shown here) and AS1010:

> XMDT automatically saves files associated with failed portions of a request.

While the lowest mean *Desirability* value considered (excluding negative values) was 0.75, achieved by AS1030:

> DGC replaces the use of AutoDSP with a custom dispatching mechanism.

The 6 highest mean *Desirability* scores determined by the stakeholders are given in Table 4. Note how these is little overlap between these ASs and the six with the highest Benefit scores—only a single AS in common: AS0140! This is an extremely important point and illustrates the impact of considering costs and benefits in making architectural tradeoff decisions. In our previous ATAM exercises we concentrated solely on technical issues; essentially we were informally characterizing benefits, but purely in terms of quality attributes. Once we explicitly consider and elicit both costs and benefits, however, the set of highly desirable ASs changes *dramatically.*

Architectural Strategy	*Desirability* Score
AS0030	5.94
AS0140	8.75
AS1000	10.83
AS1010	10.83
AS1086	5.00
AS1090	9.17

Table 4. The top 6 mean *desirability* scores

4.6. Make decisions

At this point the ECS team was in the position of being able to make decisions regarding which ASs to choose for

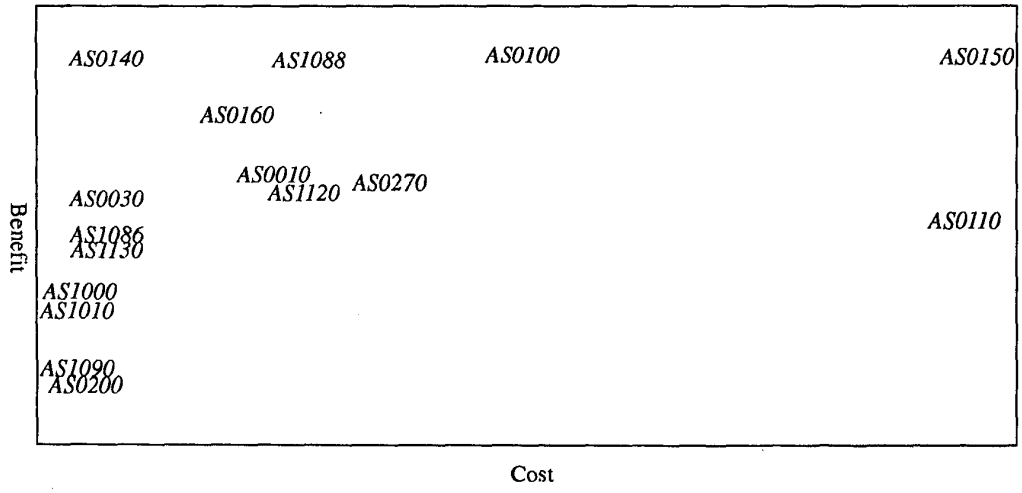

Figure 3. Plotting the Costs/Benefits of the ECS's Architectural Strategies

further investigation. Of the 57 ASs, 16 were chosen for further investigation, which principally means a more detailed cost estimate and a detailed examination of schedule implications. It is estimated that only 6 or 8 of these will be finally chosen for implementation, due to budgetary constraints. The 16 that were chosen for further study are shown in Figure 3.

4.7. Summary of the ECS case study

The precise description of which 6 or 8 ASs are finally chosen for implementation is irrelevant to the point of the CBAM and the message of this paper. The point of the CBAM is that it has given the stakeholders a principled, repeatable method for making architectural choices, and understanding the consequences of these choices in terms of costs, benefits, schedules, and risks (in the form of uncertainty). This method has been successful in that it guided the ECS stakeholders to consider many ASs that they would have otherwise overlooked, for two reasons:

1. Some ASs did not have extremely high *Benefit* scores (such as AS0030, AS0200, AS1086, AS1000, AS1010, AS1090, and AS1130), but are very inexpensive to realize and hence have some of the highest *Desirability* scores.
2. Some ASs have high *Benefit* scores but do not appear to be individually desirable because they are extremely costly—such as AS110 and AS150—but these ASs should be considered because many other ASs are dependent on them (there are 9 and 8 dependencies for AS110 and AS150 respectively).

The CBAM causes the stakeholders' attention to be focussed on areas of the Cost/Benefit graph (such as the lower left-hand corner) which they might otherwise have neglected, and gives them a principled set of steps for making system investment decisions.

5. Challenges for the CBAM

The current form of the CBAM is an application of a probabilistic Cost-Benefit analysis framework combined with a decision analysis framework to make software and system architecture decisions [10]. The techniques of decision analysis are used to define the design space and eventually to maximize the sum total expected utility (*Desirability*) of all proposed architectural changes.

There are many challenges of applying the Cost-Benefit analysis framework to the design of real software systems. We discuss the most important issues below.

5.1. Coping with the design space

An inherent problem that we face with software architecture design is the size of the solution space over which we need to perform the optimization exercise. We determine the architectural strategies available to the designers by guiding and facilitating the stakeholders, who hold a wealth of experience and domain knowledge. The CBAM aids in minimizing the size of the design space and guiding the stakeholders to good (but not necessarily optimal) solutions.

The CBAM process of choosing design strategies to be implemented is satisficing rather than optimizing since we can never be sure that the entire solution space has been analyzed. The only way of ensuring that *most* alternatives are considered is by involving a wide range of stakeholders in the process and in constantly having process checks, brainstorming, and external scrutiny (if possible).

5.2. Sensitivity to uncertainty in cost, benefit values

While software engineering research has investigated economic issues for many years, it has primarily concerned itself with modeling the *cost* of software systems. In spite of the many cost models that have been developed, the accuracy of the estimates is limited. The inherent uncertainty in cost modeling must be taken into account when making system investment decisions. Another problem is that, as mentioned earlier, cost models that are "architecturally aware" are not yet available.

Another challenge that we face is that gauging the efficacy of the various design options is also fraught with uncertainty. Quantifying benefit is a daunting challenge and the present version of the CBAM is (to our knowledge) a first step in that direction. To ensure that the stakeholders' decisions are robust we must determine the sensitivity of their decisions based upon the range, variance, and concordance of their elicited judgements.

5.3. Short time span

It is quite typical, when people think about their goals for a system, that they skew their concerns (and hence the benefits that they are seeking) toward short-term gains rather than considering the complete product life-cycle implications. Pressing, immediate issues of operation tend to take precedence over far-sighted strategic business goals. This issue can be addressed to some extent in a method such as the CBAM. This is done during the elicitation process by asking the stakeholders to explicitly consider the long-range benefits with regards to a change; that is, to place a value on them.

In this manner, the rationale for decisions are made explicit and are tied back to the stated business goals. The resultant present values are obtained by appropriately discounting the expected future benefits[2]. We believe that this

2. If stakeholders use dollar values to quantify benefits, then the Net present value is $\Sigma(B_i \times (1+r)^{-i})$ where r is the rate of return and i the year for which B_i is the benefit.

helps to counterbalance the tendency of the stakeholders to focus on short-term needs.

6. Plausible improvements to the existing method

Investing in a software project is similar to investment under uncertainty in any field. The economics and finance literature discusses a number sophisticated techniques to address this issue [6] and these are beginning to be adopted by software engineers. For example, the use of real-options theory and portfolio theory for software design has been suggested by Sullivan *et al* [13] and Butler et al [3]. Drawing sufficient parallels between financial markets and software products to make these theories useful is at times challenging, but the theories are nonetheless compelling.

More research needs to be carried out to both operationalize and validate the use of 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 this experimental application 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. One can't seriously contemplate investing in an object or a few functions, but one can (and should) think of investing in major system capabilities and qualities.

Using the CBAM one chooses a set of changes that have a high *Desirability* metric. Due to the nature of risk averseness of an organization, all these changes must have a relatively high benefit and a relatively low cost profile. It is thus possible that architectural changes that have high benefit, high cost, and high uncertainty are eliminated and likewise for low benefit, low cost changes. These area s of the Desirability space should be systematically explored in the CBAM process.

6.1. Using a portfolio approach

The idea behind the portfolio approach to investment is to reduce the overall uncertainty by implementing either uncorrelated or negatively correlated strategies [4]. For example, we could have a particular architectural strategy, AS1, that has high benefit with relatively high cost. Another architectural change, AS7, could have low benefit and low cost. AS1 is considered as a high risk option while AS7 is a low risk option. Further analysis of the dependencies shows that AS1 and AS7 are competing technologies that are mutually exclusive and negatively correlated as far as their uncertainties are concerned. In simpler terms, we find that if AS1 is successful, AS7 will not be and vice versa.

This way of thinking leads us to the following generalization: that there will be a number of other changes ASi and ASj which could be combined together to reduce the *overall* risk of the system even though a large number of them are doomed to failure (in fact, this is necessarily so!). The architectural changes thus chosen could be considered to be a portfolio that attempts to ensure a certain level of success, while simultaneously reducing the overall risk. It does so, however, at a higher cost.

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

6.2. The real-options approach

The essence of the real-option formulation is that there may be value in postponing an investment decision. Considering the high uncertainty involved in software technologies and projects, time to gather more information is always welcome. In the situation where we are considering many architectural strategies, we could apply the real-option formulation based upon the dependency structure of the strategies. For example, let AS2 and AS3 be two architectural strategies, where AS2 is low-cost, low-benefit and AS3 is high-cost, high-benefit. Analysis of AS2 and AS3 shows that, to implement AS3, AS2 must first be implemented (i.e., there is a hierarchical dependency). Now consider the case where the low desirability of AS3 is due to the high level of uncertainty associated with it, and this uncertainty is expected to reduce drastically over time. The reduction in uncertainty could be due to the information gained through the implementation of AS2 (e.g. AS2 could be viewed as a prototype for AS3, perhaps using a new technology in a non-critical portion of the system).

We could thus frame the problem as follows: the investment decision to be made is the implementation of an architectural change AS3. In light of the uncertainty involved, AS3 does not seem to be a prudent investment at the present time. However, we could buy the option of implementing AS3 at a later date by implementing AS2 instead (the cost of the option is thus the cost of implementing AS2). By implementing AS2, we could learn more about AS3 and hence the net value associated with AS3 changes over time until we decide to implement it (the "strike" date of the option) or to completely forgo the implementation.

We thus see that the techniques and reasoning used in stock market investment decisions could be applied to software design decisions. The CBAM, through its idea of quantification of benefit, its quantification of uncertainty, and its identification of dependencies, is an important step in this direction.

7. Conclusions

This paper has presented the CBAM. The CBAM is:

- a method that causes the stakeholders to quantify the benefits as well as the costs, dependencies, and schedule implications of architectural decisions
- a method that explicitly captures the uncertainty surrounding costs and benefits
- a scalable method: it can be inexpensively employed for

triage or it can be used exhaustively once the search space of ASs has been narrowed.

When developing or evolving any engineered artifact, including software-intensive systems, one is always faced with the problem of determining where to spend a finite set of resources. The problem is to maximize the benefits resulting from developing or evolving the system given a limited of pot of dollars to spend. Frequently this involves selecting the right set of product features. Determining the "right" feature set is clearly *necessary* for a system's eventual success, but it is not always *sufficient*: the quality attributes associated with the features are key. A feature with poor performance or poor security may be worse than not having the feature at all. Moreover, it is quality attributes that determine the overall architecture of a large and complex system. A significant amount of effort is typically dedicated to getting these attributes right. Consequently, we have developed and prototyped a decision-making method that explicitly accounts for the costs and benefits associated with engineering qualities into the software architecture of a system. In the CBAM, costs and benefits are qualities that are traded off, in addition to the technical quality attributes.

We have presented the steps of the CBAM in this paper and have illustrated how we applied the method to NASA's ECS, a large scientific information system. Our initial experience and the feedback that we have received from the ECS project indicates that the method holds significant promise for facilitating the decision making process. It has aided the stakeholders in determining how to allocate limited resources to their software evolution effort and they have participated enthusiastically. Moreover, the method dovetails nicely with our existing method for evaluating software architectures, the Architecture Tradeoff Analysis Method.

Encouraged by our first experience we have also examined the CBAM for places where our approach might be strengthened. This introspection has led to three areas of future work: examining how to better elicit and understand the method's sensitivity to uncertainty; exploring how portfolio theory can be use to exploit dependencies in a "portfolio of architectural options"; and how to use the theory of real-options to invest a little bit today in a sub-optimal decision with the hope of delaying a risky decision until it becomes less risky.

While it is still in its early stages of development and refinement, we are optimistic about the prospects for the CBAM. We see it becoming a practical and theoretically sound method that can be used for making architecturally informed investment decisions based on an analysis of costs, benefits, and uncertainty.

8. Acknowledgments

We would like to gratefully acknowledge the support of Linda Northrop and the staff of the SEI's ATA initiative, and Mike Moore and the staff of NASA's ECS project.

9. References

[1] Boehm, B. *Software Engineering Economics*, Prentice Hall, 1981.

[2] Boehm, B., Bose, P., Horowitz, E., Lee, M., "Software Requirements As Negotiated Win Conditions," *Proceedings of ICRE94*, IEEE Computer Society Press, Colorado Springs Colorado, April 1994.

[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] Capers Jones, T., "Estimating Software Costs", McGraw-Hill, 1999.

[6] Dixit, A., Pindyck, R. S., *Investment Under Uncertainty*, Princeton University Press, New Jersey, 1994.

[7] Kazman, R., Klein, M., Clements, P., "ATAM: A Method for Architecture Evaluation", CMU/SEI-2000-TR-004, Software Engineering Institute, Carnegie Mellon University, 2000.

[8] Kazman, R., Barbacci, M., Klein, M., Carriere, S. J., Woods, S. G. "Experience with Performing Architecture Tradeoff Analysis", *Proceedings of the 21st International Conference on Software Engineering*, Los Angeles, CA, May 1999, 54-63.

[9] Kazman, R., Abowd, G., Bass, L., Webb, M., "SAAM: A Method for Analyzing the Properties of Software Architectures," *Proceedings of the 16th International Conference on Software Engineering*, Sorrento, Italy, May 1994, 81-90.

[10]Mishan E. J., *Economics for Social Decisions: Elements of Cost-Benefit Analysis*, Praeger, New York, 1973.

[11]Morgan, G., Henrion, M, *Uncertainty: A Guide to dealing with Uncertainty in Quantitative Risk and Policy Analysis*, Cambridge University Press, 1990.

[12]Siegel, S., *Nonparametric Methods for the Behavioral Sciences*, New York: McGraw-Hill, 1956.

[13]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.

[14]Winterfeldt, D. and Edwards, E., *Decision Analysis and Behavioral Research*, Cambridge University Press, 1986.