Reference Chapter 23
Software Architecture in Practice
Third Edition
Len Bass
Paul Clements
Rick Kazman

BITS Pilani
Pilani|Dubai|Goa|Hyderabad

Module 9

Economic Evaluation through

Cost Benefit Analysis Method

Harvinder S Jabbal
SSZG653 Software Architectures

Chapter Outline

innovate achieve lead

- Decision-Making Context
- The Basis for the Economic Analyses
- Putting Theory into Practice: The CBAM
- Case Study: The NASA ECS Project
- Summary

innovate achieve lead

BITS Pilani

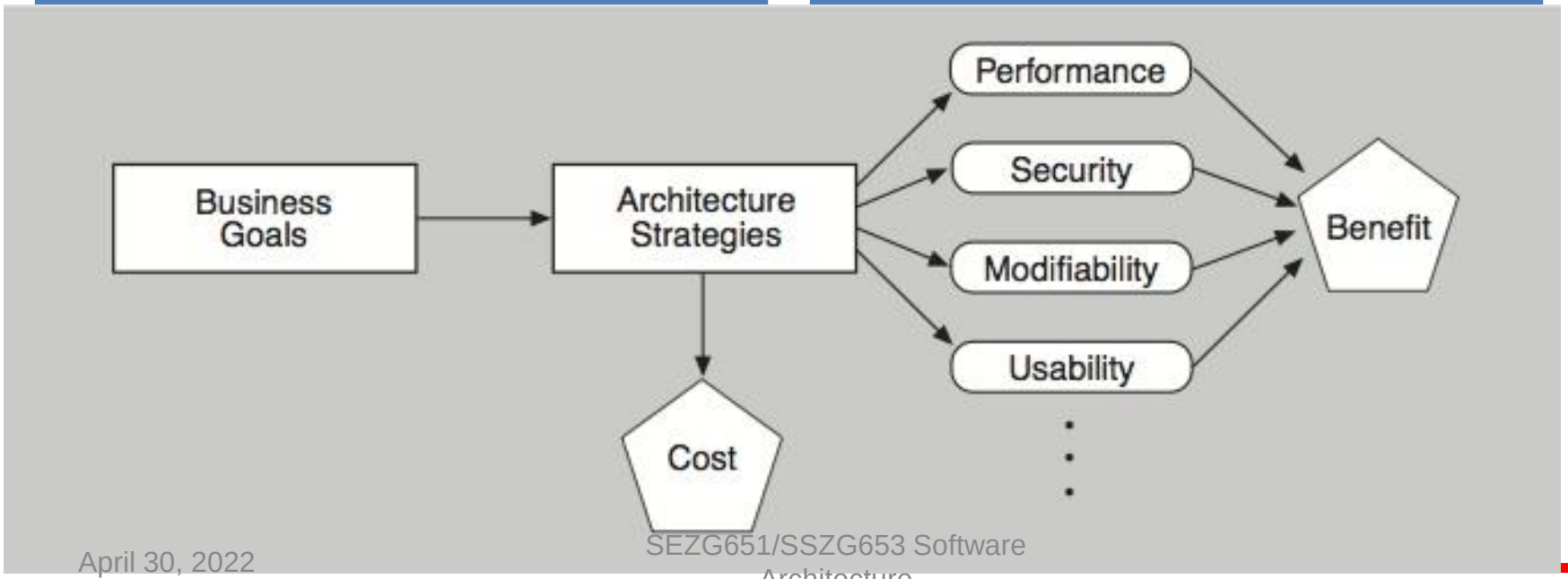
Pilani|Dubai|Goa|Hyderabad

Decision-Making Context

Decision-making Context

The quality attributes achieved by architecture decisions have economic implications in terms of the benefits (*utility*) that can be derived from those decisions.

This interplay between the costs and the benefits of architectural decisions guides (and torments) the architect.



Decision-making Context

Example:

- Using redundant hardware to achieve a desired level of availability has a cost
- Checkpointing to a disk file has a different cost.
- Both of these architectural decisions will result in measurable levels of availability that will have some value to the organization developing the system.

Knowing the costs and benefits associated with particular decisions enables reasoned selection from among competing alternatives.

Economic analysis does not make decisions for the stakeholders. It simply aids in the elicitation and documentation of *value for cost* (VFC).

- VFC: a function of the costs, benefits, and uncertainty of a "portfolio" of architectural investments.
- It gives the stakeholders a framework within which they can apply a rational decision-making process that suits their needs and their risk aversion.

Economic analysis isn't something to apply to every architectural decision, but rather to the most basic ones that put an overarching architectural strategy in place.

innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

Basis for Economic Analyses

Basis for Economic Analyses

innovate achieve lead

Begin with a collection of scenarios generated from requirements elicitation, architectural evaluation, or specifically for economic analysis.

Examine how these scenarios differ in the values of their projected responses.

Assign utility to those values.

- The utility is based on the importance of each scenario with respect to its anticipated response value.

Consider the architectural strategies that lead to the various projected responses.

- Each strategy has a cost, and each impacts *multiple* quality attributes.
- That is, an architectural strategy could be implemented to achieve some projected response, but while achieving that response, it also affects some other quality attributes.
- The utility of these "side effects" must be taken into account when considering a strategy's overall utility.

Combine this overall utility with the project cost of an architectural strategy to calculate a final VFC measure.

Utility-Response Curves

Our economic analysis uses quality attribute scenarios (from Chapter 4) as the way to concretely express and represent specific quality attributes.

When we vary the values of the responses, and ask what the utility is of each response we get a set of points that we can plot: a *utility-response curve*.

Each scenario's stimulus-response pair provides some utility (value) to the stakeholders, and the utility of different possible values for the response can be compared.

- To help us make major architectural decisions, we might wish to compare the value of high performance against the value of high modifiability against the value of high usability, and so forth. The concept of utility lets us do that.

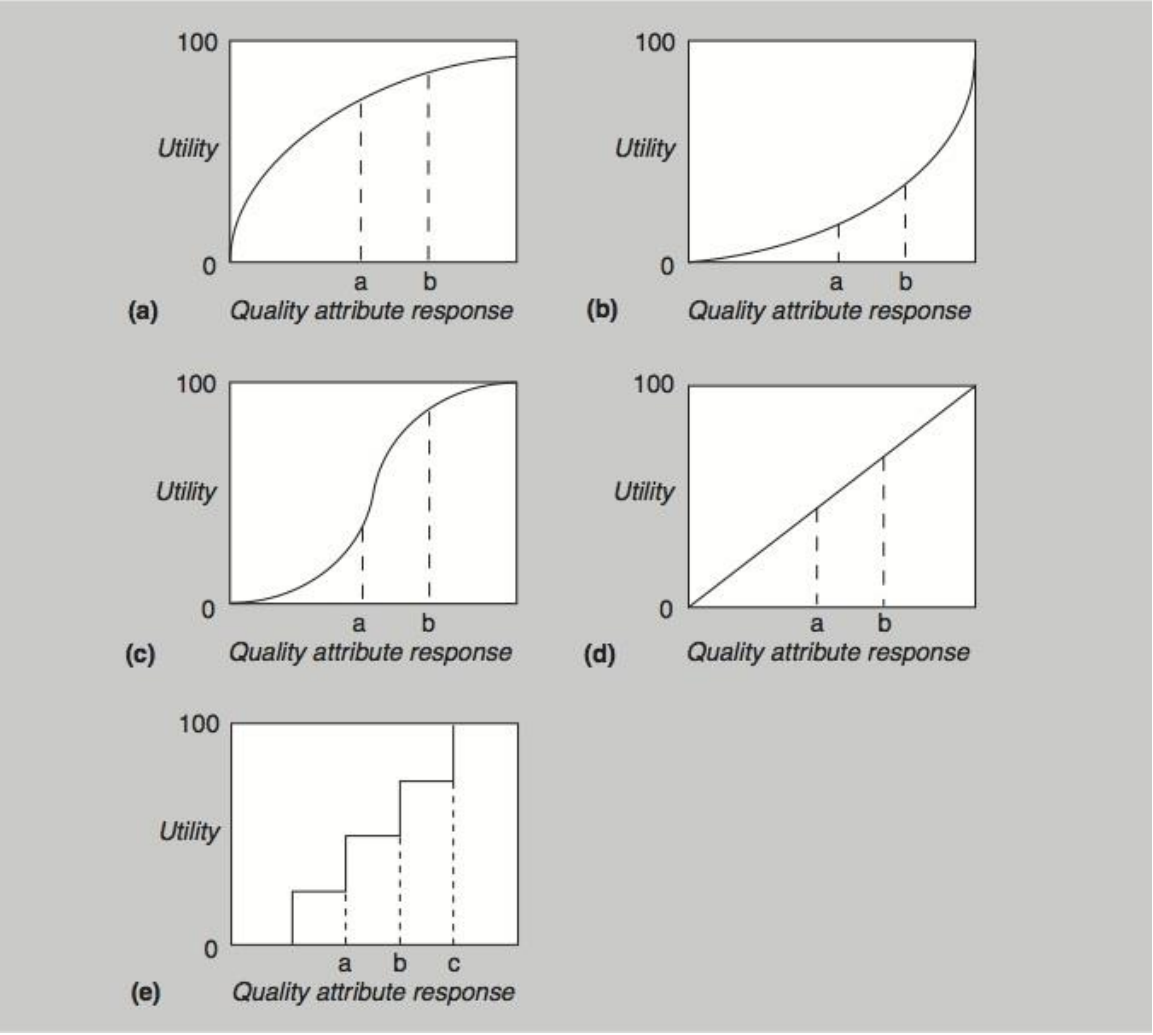
With prodding, stakeholders can express their needs using concrete response measures, such as "99.999 percent available." But how much would they value slightly less demanding quality attributes, such as "99.99 percent available"? Would that be almost as good?

If so, then the lower cost of achieving that lower value might make that the preferred option,

Capturing the utility of alternative responses of a scenario better enables the architect to make tradeoffs involving that quality attribute.

We can portray each relationship between a set of utility measures and a corresponding set of response measures as a graph—a utility-response curve.

Example Utility-Response Curves



Weighting the Scenarios

Different scenarios will have different importance to the stakeholders.

To make a choice of architectural strategies that is best suited to the stakeholders' desires, we must weight the scenarios.

- It does no good to spend a great deal of effort optimizing a scenario in which the stakeholders actually have very little interest.

Determining Benefit and Normalization

The overall benefit of an architectural strategy across various quality attribute scenarios is the sum of the utility associated with each scenario, weighted by the importance of the scenario.

For each architectural strategy i, its benefit B_i over *j* scenarios (each with weight W_j) is

- $B_i = \sum_j (b_{i,j} \times W_j)$

Each $b_{i,j}$ is calculated as the change in utility (over whatever architectural strategy is currently in place, or is in competition with the one being considered) brought about by the architectural strategy with respect to this scenario:

- $b_{i,j} = U_{expected} - U_{current}$

This is the utility of the expected value of the architectural strategy minus the utility of the *current* system relative to this scenario.

Calculating Value for Cost

The VFC for each architectural strategy is the ratio of the total benefit, B_i, to the cost, C_i, of implementing it:

- $VFC = B_i / C_i$

The cost C_i is estimated using a model appropriate for the system and the environment being developed, such as a cost model that estimates implementation cost by measuring an architecture's interaction complexity.

You can use this VFC score to rank-order the architectural strategies under consideration.

innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

Putting Theory into Practice: The CBAM

Practicalities of Utility Curve Determination

To build the utility-response curve, we first determine the quality attribute levels for the best-case and worst-case situations.

The best-case quality attribute level is that above which the stakeholders foresee no further utility.

- For example, a system response to the user of 0.1 second is perceived as instantaneous, so improving it further so that it responds in 0.03 second has no additional utility.

The worst-case quality attribute level is a minimum threshold above which a system must perform; otherwise it is of no use to the stakeholders.

These levels—best-case and worst-case—are assigned utility values of 100 and 0, respectively.

Practicalities of Weighting Determination

One method of weighting the scenarios is to prioritize them and use their priority ranking as the weight.

- For *N* scenarios, the highest priority one is given a weight of 1, the next highest is given a weight of (*N*–1)/*N*, and so on.
- This turns the problem of weighting the scenarios into one of assigning priorities.

The stakeholders can determine the priorities through a variety of voting schemes.

- One simple method is to have each stakeholder prioritize the scenarios (from 1 to *N*) and the total priority of the scenario is the sum of the priorities it receives from all of the stakeholders.
- Voting can be public or secret.

Other schemes are possible. Regardless of the scheme used, it must make sense to the stakeholders and it must suit their culture.

Practicalities of Cost Determination

There are very few cost models for various architectural strategies.

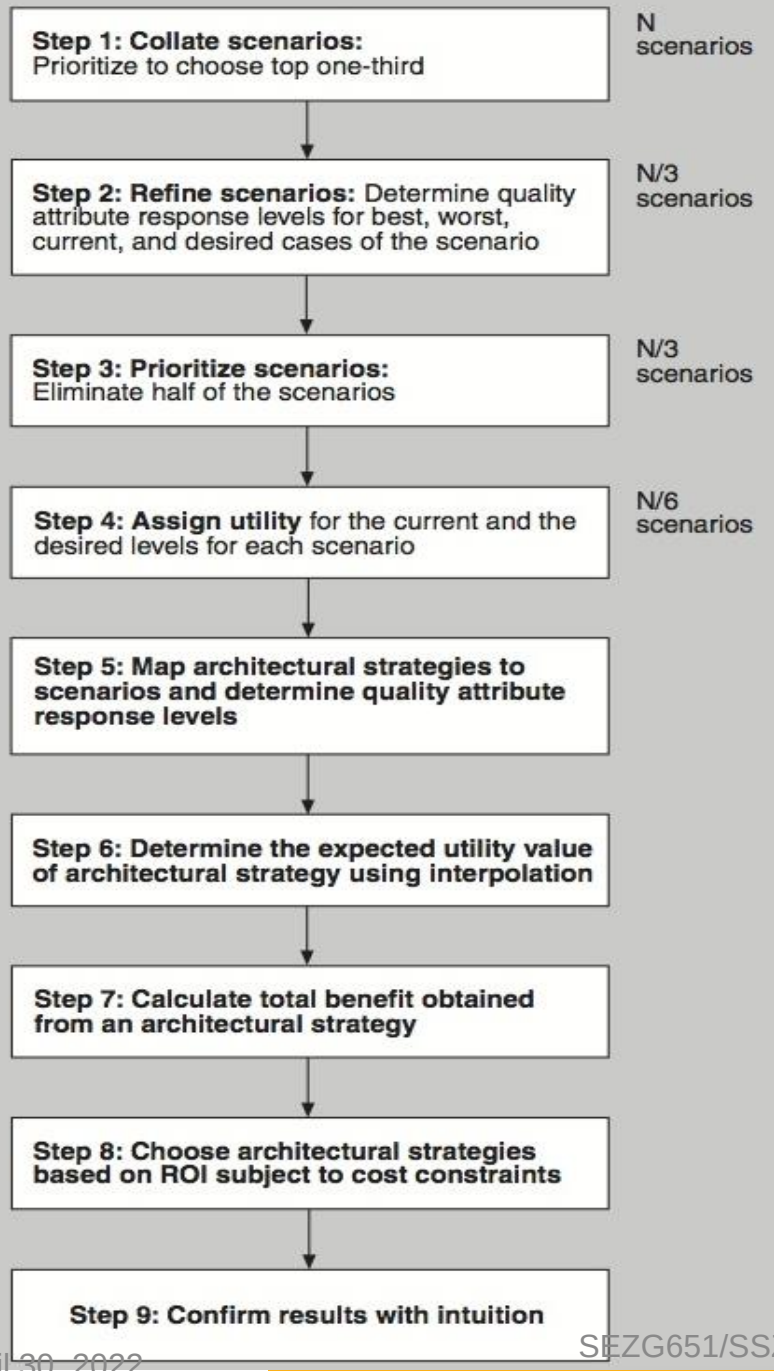
There are many software cost models, but they are based on overall system characteristics such as size or function points.	These are inadequate to answer the question of how much does it cost to, for example, use a publish-subscribe pattern in a particular portion of the architecture.	There are cost models that are based on complexity of modules (by function point analysis according to the requirements assigned to each module) and the complexity of module interaction, but these are not widely used in practice.	More widely used in practice are corporate cost models based on previous experience with the same or similar architectures, or the experience and intuition of senior architects.

Practicalities of Cost Determination

Architects often turn to cost estimation techniques.

An absolute number for cost isn't necessary to rank candidate architecture strategies.

- You can often say something like "Suppose strategy A costs $x. It looks like strategy B will cost $2x, and strategy C will cost $0.5x." That's enormously helpful.
- A second approach is to use very coarse estimates. Or if you lack confidence for that degree of certainty, you can say something like "Strategy A will cost a lot, strategy B shouldn't cost very much, and strategy C is probably somewhere in the middle."



innovate achieve lead

Cost Benefit Analysis Method (CBAM)

The CBAM places the principles just discussed into a set of steps: a method.

CBAM Stakeholders

The stakeholders in a CBAM exercise include people who can authoritatively speak to the utility of various quality attribute responses.

They probably include the same people who were the source of the quality attribute scenarios being used as input.

Step 1

innovate achieve lead

Collate scenarios.

- Give the stakeholders the chance to contribute new scenarios.
- Ask the stakeholders to prioritize the scenarios based on satisfying the business goals of the system.
- This can be an informal prioritization using a simple scheme such as “high, medium, low” to rank the scenarios.
- Choose the top one-third for further study.

Step 2

- Refine the scenarios chosen in step 1, focusing on their stimulus-response measures.
- Elicit the worst-case, current, desired, and best-case quality attribute response level for each scenario.
- For example, a refined performance scenario might tell us that worst-case performance for our system's response to user input is 12 seconds, the best case is 0.1 seconds, and our desired response is 0.5 seconds. Our current architecture provides a response of 1.5 seconds.

Step 3

innovate achieve lead

Prioritize scenarios.

- Prioritize the refined scenarios, based on stakeholder votes.
- Give 100 votes to each stakeholder and have them distribute the votes among the scenarios, where their voting is based on the desired response value for each scenario.
- Total the votes and choose the top 50 percent of the scenarios for further analysis.
- Assign a weight of 1.0 to the highest-rated scenario; assign the other scenarios a weight relative to the highest rated.
- This becomes the weighting used in the calculation of a strategy's overall benefit.
- Make a list of the quality attributes that concern the stakeholders.

Step 4

Assign utility.

- Determine the utility for each quality attribute response level (worst-case, current, desired, best-case) for the scenarios from step 3.
- You can conveniently capture these utility curves in a table (one row for each scenario, one column for each of the four quality attribute response levels).

Scenario	Worst Case	Current	Desired	Best Case
Scenario #17: Response to user input	12 seconds	1.5 seconds	0.5 seconds	0.1 seconds
	Utility 5	Utility 50	Utility 80	Utility 85

Step 5

Map architectural strategies to scenarios and determine their expected quality attribute response levels.

- For each architectural strategy under consideration, determine the expected quality attribute response levels that will result for each scenario.

Step 6

Determine the utility of the expected quality attribute response levels by interpolation.

- Using the elicited utility values (that form a utility curve), determine the utility of the expected quality attribute response level for the architectural strategy.
- Do this for each relevant quality attribute enumerated in step 3.
- For example, if we are considering a new architectural strategy that would result in a response time of 0.7 seconds, we would assign a utility proportionately between 50 (which it exceeds) and 80 (which it doesn't exceed).
- The formula for interpolation between two data points (x_a, y_a) and (x_b, y_b) is:

$$y = y_a + (y_b - y_a)\frac{(x - x_a)}{(x_b - x_a)}$$

- For us, the *x* values are the quality attribute response levels and the *y* values are the utility values. So, employing this formula, the utility value of a 0.7-second response time is 74 .

Step 7

Calculate the total benefit obtained from an architectural strategy.

- Subtract the utility value of the "current" level from the expected level and normalize it using the votes elicited in step 3.
- Sum the benefit due to a particular architectural strategy across all scenarios and across all relevant quality attributes.

Step 8

Choose architectural strategies based on VFC subject to cost and schedule constraints.

- Determine the cost and schedule implications of each architectural strategy.
- Calculate the VFC value for each as a ratio of benefit to cost. Rank-order the architectural strategies according to the VFC value and choose the top ones until the budget or schedule is exhausted.

Step 9

innovate achieve lead

Confirm results with intuition.

- For the chosen architectural strategies, consider whether these seem to align with the organization's business goals.
- If not, consider issues that may have been overlooked while doing this analysis.
- If there are significant issues, perform another iteration of these steps.

innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

CASE STUDY:
The NASA ECS Project

Case Study: The NASA ECS Project

The Earth Observing System is a constellation of NASA satellites that gathers data for the U.S. Global Change Research Program and other scientific communities worldwide.

The Earth Observing System Data Information System (EOSDIS) Core System (ECS) collects data from various satellite downlink stations for further processing.

ECS's mission is to process the data into higher-form information and make it available to scientists in searchable form. The goal is to provide both a common way to store (and hence process) data and a public mechanism to introduce new data formats and processing algorithms, thus making the information widely available.

Case Study: The NASA ECS Project

The ECS processes an input stream of hundreds of gigabytes of raw environment-related data per day. The computation of 250 standard “products” results in thousands of gigabytes of information that is archived at eight data centers in the United States. The system has important performance and availability and modifiability requirements.

The ECS project manager had a limited annual budget to maintain and enhance his current system. The manager used the CBAM to make a rational decision based on the economic criterion of return on investment.

Step 1: Collate Scenarios

IN PRIORITY ORDER

Note that they are not yet well formed and that some of them do not have defined responses. These issues are resolved in step 2, when the number of scenarios is reduced.

1. Reduce data distribution failures that result in hung distribution requests requiring manual intervention.
2. Reduce data distribution failures that result in lost distribution requests.
3. Reduce the number of orders that fail on the order submission process.
4. Reduce order failures that result in hung orders that require manual intervention.
5. Reduce order failures that result in lost orders.
6. There is no good method of tracking ECSGuest failed/canceled orders without much manual intervention (e.g., spreadsheets).
7. Users need more information on why their orders for data failed.
8. Because of limitations, there is a need to artificially limit the size and number of orders.
9. Small orders result in too many notifications to users.
10. The system should process a 50-GB user request in one day, and a 1-TB user request in one week.

Step 2: Refine Scenarios

The scenarios were refined, paying particular attention to precisely specifying their stimulus-response measures.

The worst-case, current-case, desired-case, and best-case response goals for each scenario were elicited and recorded in a table.

Step 2: Refine Scenarios

Scenario	Worst	Current	Desired	Best
1	10% hung	5% hung	1% hung	0% hung
2	> 5% lost	< 1% lost	0% lost	0% lost
3	10% fail	5% fail	1% fail	0% fail
4	10% hung	5% hung	1% hung	0% hung
5	10% lost	< 1% lost	0% lost	0% lost
6	50% need help	25% need help	0% need help	0% need help
7	10% get information	50% get information	100% get information	100% get information
8	50% limited	30% limited	0% limited	0% limited
9	1/granule	1/granule	1/100 granules	1/1,000 granules
10	< 50% meet goal	60% meet goal	80% meet goal	> 90% meet goal

Step 3: Prioritize Scenarios

In voting on the refined representation of the scenarios, the close-knit team deviated slightly from the method.

Rather than vote individually, they chose to discuss each scenario and arrived at a determination of its weight via consensus.

The votes allocated to the entire set of scenarios were constrained to 100.

Although the stakeholders were not required to make the votes multiples of 5, they felt that this was a reasonable resolution and that more precision was neither needed nor justified.

Step 3: Prioritize Scenarios

Scenario	Votes	Worst	Current	Desired	Best
1	10	10% hung	5% hung	1% hung	0% hung
2	15	> 5% lost	< 1% lost	0% lost	0% lost
3	15	10% fail	5% fail	1% fail	0% fail
4	10	10% hung	5% hung	1% hung	0% hung
5	15	10% lost	< 1% lost	0% lost	0% lost
6	10	50% need help	25% need help	0% need help	0% need help
7	5	10% get information	50% get information	100% get information	100% get information
8	5	50% limited	30% limited	0% limited	0% limited
9	10	1/granule	1/granule	1/100 granules	1/1,000 granules
10	5	< 50% meet goal	60% meet goal	80% meet goal	> 90% meet goal

Step 4: Assign Utility

In this step the utility for each scenario was determined by the stakeholders, again by consensus.

A utility score of 0 represented no utility; a score of 100 represented the most utility possible.

Step 4: Assign Utility

		Utility Scores			
Scenario	Votes	Worst	Current	Desired	Best
1	10	10	80	95	100
2	15	0	70	100	100
3	15	25	70	100	100
4	10	10	80	95	100
5	15	0	70	100	100
6	10	0	80	100	100
7	5	10	70	100	100
8	5	0	20	100	100
9	10	50	50	80	90
10	5	50	50	80	90

Step 5: Develop Architectural Strategies and Determine Their Expected Quality Attribute Response Levels

Based on the requirements implied by the preceding scenarios, a set of 10 architectural strategies was developed by the ECS architects.

For each architectural strategy/scenario pair, the response levels expected to be achieved with respect to that scenario are shown (along with the current response, for comparison purposes).

Step 5 Results (Scenarios 1-5)

Strategy	Name	Description	Scenarios Affected	Current Response	Expected Response
1	Order persistence on submission	Store an order as soon as it arrives in the system.	3	5% fail	2% Fail
			5	<1% lost	0% lost
			6	25% need help	0% need help
2	Order chunking	Allow operators to partition large orders into multiple small orders.	8	30% limited	15% limited
3	Order bundling	Combine multiple small orders into one large order.	9	1 per granule	1 per 100
			10	60% meet goal	55% meet goal
4	Order segmentation	Allow an operator to skip items that cannot be retrieved due to data quality or availability issues.	4	5% hung	2% hung
5	Order reassignment	Allow an operator to reassign the media type for items in an order.	1	5% hung	2% hung

Step 5 Results (Scenarios 5-10)

6	Order retry	Allow an operator to retry an order or items in an order that may have failed due to temporary system or data problems.	4	5% hung	3% hung
7	Forced order completion	Allow an operator to override an item's unavailability due to data quality constraints.	1	5% hung	3% hung
8	Failed order notification	Ensure that users are notified only when part of their order has truly failed and provide detailed status of each item; user notification occurs only if operator okays notification; the operator may edit notification.	6	25% need help	20% need help
			7	50% get information	90% get information
9	Granule-level order tracking	An operator and user can determine the status for each item in their order.	6	25% need help	10% need help
			7	50% get information	95% get information
10	Links to user information	An operator can quickly locate a user's contact information. Server will access SDSRV information to determine any data restrictions that might apply and will route orders/order segments to appropriate distribution capabilities, including DDIST, PDS, external subsetters and data processing tools, etc.	7	50% get information	60% get information

Step 6: Determine the Utility of the "Expected" Quality Attribute Response Levels by Interpolation

Once the expected response level of every architectural strategy has been characterized with respect to a set of scenarios, their utility can be calculated by consulting the utility scores for each scenario's current and desired responses for all of the affected attributes.

Using these scores, we may calculate, via interpolation, the utility of the expected quality attribute response levels for the architectural strategy/scenario pair.

Step 6 Results

Strategy	Name	Scenarios Affected	Current Utility	Expected Utility
1	Order persistence on submission	3	70	90
		5	70	100
		6	80	100
2	Order chunking	8	20	60
3	Order bundling	9	50	80
		10	70	65
4	Order segmentation	4	80	90
5	Order reassignment	1	80	92
6	Order retry	4	80	85
7	Forced order completion	1	80	87
8	Failed order notification	6	80	85
		7	70	90
9	Granule-level order tracking	6	80	90
		7	70	95
10	Links to user information	7	70	75

Step 7: Calculate the Total Benefit Obtained from an Architectural Strategy

Total benefit of each architectural strategy can now be calculated:

- $B_i = \sum_j (b_{i,j} \times W_j)$

This equation calculates total benefit as the sum of the benefit that accrues to each scenario, normalized by the scenario's relative weight.

Using this formula, the total benefit scores for each architectural strategy are now calculated.

Step 7

Strategy	Scenario Affected	Scenario Weight	Raw Architectural Strategy Benefit	Normalized Architectural Strategy Benefit	Total Architectural Strategy Benefit
1	3	15	20	300	
1	5	15	30	450	
1	6	10	20	200	950
2	8	5	40	200	200
3	9	10	30	300	
3	10	5	–5	–25	275
4	4	10	10	100	100
5	1	10	12	120	120
6	4	10	5	50	50
7	1	10	7	70	70
8	6	10	5	50	
8	7	5	20	100	150
9	6	10	10	100	
9	7	5	25	125	225
10	7	5	5	25	25

Step 8: Choose Architectural Strategies Based on VFC Subject to Cost Constraints

To complete the analysis, the team estimated cost for each architectural strategy.

The estimates were based on experience with the system, and a return on investment for each architectural strategy was calculated.

Using the VFC, we were able to rank each strategy.

The ranks roughly follow the ordering in which the strategies were proposed: strategy 1 has the highest rank; strategy 3 the second highest. Strategy 9 has the lowest rank; strategy 8, the second lowest.

This simply validates stakeholders' intuition about which architectural strategies were going to be of the greatest benefit. For the ECS these were the ones proposed first.

Step 8

Strategy	Cost	Total Strategy Benefit	Strategy VFC	Strategy Rank
1	1200	950	0.79	1
2	400	200	0.5	3
3	400	275	0.69	2
4	200	100	0.5	3
5	400	120	0.3	7
6	200	50	0.25	8
7	200	70	0.35	6
8	300	150	0.5	3
9	1000	225	0.22	10
10	100	25	0.25	8

Results of the CBAM Exercise

innovate achieve lead

- The most obvious results of the CBAM are the ordering of architectural strategies based on their predicted VFC.
- However, there are social and cultural benefits as well.
- The CBAM process provides structure to what is always largely unstructured discussions, where requirements and architectural strategies are freely mixed and where stimuli and response goals are not clearly articulated.
- The CBAM process forces the stakeholders to make their scenarios clear in advance, to assign utility levels of specific response goals, and to prioritize these scenarios based on the resulting determination of utility.
- Finally, it produces clarification of both scenarios and requirements, which by itself is a significant benefit.

Summary

Architecture-based economic analysis is grounded on understanding the utility-response curve of various scenarios and casting them into a form that makes them comparable.

Once they are in this common form—based on the common coin of utility—the VFC for each architecture improvement, with respect to each relevant scenario, can be calculated and compared.

Applying the theory in practice has a number of practical difficulties, which CBAM solves.

The application of economic techniques is inherently better than the ad hoc decision-making approaches that projects employ today.

Giving people the appropriate tools to frame and structure their discussions and decision making is an enormous benefit to the disciplined development of a complex software system.

Thank you……..

Credits

- **Chapter Reference from Text T1: 23**
- Slides have been adapted from Authors Slides Software Architecture in Practice – Third Ed.
  - Len Bass
  - Paul Clements
  - Rick Kazman