

Utility Theory Can Be Useful

Vibha Sazawal

University of Washington

vibha@cs.washington.edu

Economics-based software engineering techniques often employ money as the single criterion when selecting among alternatives. Indeed, the use of money as one's measure of value could be considered the feature that makes a technique "economics-based" in the first place. Economics-based software engineering methodologies apply techniques from economic theory to software design and process decisions. These decisions are often framed in terms of choosing one *alternative* from a set. Each alternative is defined in terms of the following: known costs or gains, and uncertain future costs or gains. Future costs or gains, known as *consequences*, are affected by events that may or may not occur. This uncertainty about whether consequences may or may not occur is *risk*.

Using interest rates, some kind of market, or expert opinion, an estimate of the risk involved in choosing an alternative can be computed. Suppose that an expert assigns probabilities that the various consequences will occur given each alternative. These probabilities can be used to weight the alternatives on the basis of how likely each consequence is of occurring if that alternative is chosen. The result is a computation of the expected monetary return for each alternative. One should choose the alternative that offers the highest expected monetary return. Money is the sole deciding factor.

While money is important to almost all software endeavors, the use of money as the sole criterion when making decisions may not be appropriate in all situations. Sometimes decision-makers also have qualitative goals such as flexibility, portability, and maintainability. Some economic software engineering approaches concentrate on how to assign a quantitative monetary value to these qualitative goals [1]. These approaches work well in some situations, but there are two cases which are not addressed by money-centric economic approaches -- one, when monetary benefits of a qualitative goal are simply too difficult to estimate and two, when there really are no monetary benefits to a goal.

As an example of the first case, companies often incur costs in the hopes of attaining customer goodwill. The companies' overall goal is revenue generation, but the exact amount of revenue obtained from customer goodwill is difficult to quantify monetarily. There is not a direct market that prices customer goodwill for the company in the way that stock exchanges value stocks. The company can, however, measure goodwill on its own scale of what the company's decision-makers consider "good" to what they consider "poor". The company would like to use this scale of goodwill as a proxy for money, since the proxy is easier to measure.

An example of the second case is a software company that highly values low numbers of defects in its software, not due to the profit obtained from doing so, but due to the corporate culture and personal ethics of the software engineering management. In this case, the company would continue its policies even if it could make more money by releasing software with a higher number of defects. Of course, the company does not want to entirely bankrupt itself in its pursuit of defect-free software; thus, it genuinely has two objectives -- low numbers of defects and monetary return.

When decision-makers want to choose the best decision over multiple objectives, utility theory may be helpful. *Utility* is the evaluation of a possible consequence relative to the best consequence that could occur and the worst consequence that could occur [2]. A consequence is simply an outcome that occurs with a certain probability given what decision is made. The utility of the consequence is a measure of how "good" if a given consequence or outcome were to occur, relative to the best and worst consequences that could occur.

Utility theory is a complex and rich field and it is not possible to describe it in full here. Many books [2][3][4] describe utility theory as their sole subject matter and with good reason -- if done incorrectly, it is very possible to get entirely invalid results. When using utility theory to solve a decision making problem, one's goal is to choose the alternative that maximizes expected utility. This is done by associating a utility with each possible consequence and then determining the probability that each consequence occurs given what alternative you choose. These probabilities are then used to weight the consequences of each alternative in order to determine with action or alternative is best.

Unlike value, which is an objective measure, utility is a subjective measure. The utility of an item varies from decision maker to decision maker depending upon each individual's preferences. In contrast, the monetary value of an item does not change: a \$50 bill has a value of \$50 to everyone. However, it is not hard to imagine that a \$50 bill may be "worth more" to a penniless person than it is to a millionaire. This subjective notion of "worth" is measured by utility.

I will now present an example that applies utility theory to a software process decision. Two points are important when exploring this example:

- 1) I have invented the preferences of the decision maker (a hypothetical manager named Ms. Brown). In practice, these preferences would be extracted from a real person.
- 2) The utility analysis performed in this example is quite simplified. A detailed and accurate utility analysis would be far beyond the scope of this paper. This example is just supposed to illustrate what an application of utility theory would be like.

For example, say that Ms. Brown is leading a project to build a software system. She has four objectives for the system:

- 1) implement all functionality described in the specification of the system,
- 2) minimize the costs of building the system, and
- 3) build the system in such a way that it is easier to modify the system later if needed, and
- 4) finish the system by the project deadline.

Ms. Brown is presented with a number of approaches from her project team leaders as to how the project should be conducted. After talking to upper management, she decides that the budget and specification for the project are quite fixed. Thus, she only considers approaches that plan to complete the entire system and do so under budget.

However, the remaining two approaches vary in how they accommodate ease of change and how closely they meet the deadline for the project. (For the purposes of this example, ease of change will be incorporated into the system by having good documentation and employing a modular design.)

Let me call these two approaches the “traditional approach”, $a_{\text{traditional}}$ and the “experimental approach”, $a_{\text{experimental}}$.

The traditional approach involves using the techniques that Ms. Brown's teams have used in the past. Typically the outcome is a system that is less flexible than what Ms. Brown would like, but the system will be reasonably documented at least and sometimes is broken into two modules. In the traditional approach, the schedule often slips, but not by very much.

The "experimental" approach involves using a new technique never before used by Ms. Brown's company. This "new, novel" approach promises excellent results and ease of change. Ms. Brown believes that the approach will either work wonderfully, producing very modular code, or the approach will end up being disastrous, producing hard-to-modify code. Ms. Brown also believes that even using the "experimental" approach, there is probably no way to build a very modular system without finishing the project late.

Which project plan should Ms. Brown choose?

First, let us consider 4 consequences:

Consequence A	Finish on time and build a monolithic system, hard-to-change system without documentation.
Consequence B	Finish one week late and build a monolithic system with extensive documentation intended to help making changes easier.
Consequence C	Finish two weeks late, but build a system with two modules and extensive documentation. The two modules are very large.
Consequence D	Finish four weeks late, but build a system with many modules and extensive documentation. The many modules are well designed and there is not much coupling.

I will rewrite the outcomes as follows. Let x equal the number of days the project is late and let y equal the "ease of change" of the system.

Consequence A: $\{ x = 0, y = \text{"monolithic, no documentation"} \}$

Consequence B: $\{ x = 7, y = \text{"monolithic, documentation"} \}$

Consequence C: $\{ x = 14, y = \text{"2 modules, documentation"} \}$

Consequence D: $\{ x = 28, y = \text{"many modules, documentation"} \}$

The second task in selecting a project plan is to determine Ms. Brown's utility function. One begins this process by looking for properties of Ms. Brown's preference structure that make assessment of the utility function easier. This is performed by asking a series of structured questions to Ms. Brown. These questions include scenarios where she is given two consequences and must state which one she prefers, or she is given two gambles and must state which one she prefers. A gamble is denoted as $\langle a, p, b \rangle$, which means that consequence a will occur with probability p , and consequence b will occur with probability $1-p$ [2]. Ms. Brown may also be offered a gamble and a fixed consequence and must choose which is preferable, the gamble or the fixed consequence. The purpose of these questions is not only to find out just whether a consequence is preferred over another consequence, but "how much" the consequence is preferred.

During this question and answer period, it is discovered that for Ms. Brown, consequence attributes x and y are *mutually utility independent*. This means that the

preference structure for y given a fixed value for x did not depend on the value of x, and the preference structure for x given y did not depend on the value of y.

This means that we have a utility function of the form:

$$u(x,y) = k_x(u_x(x)) + k_y u(y) + k_{xy} u_x(x) u(y)$$

where $u_y(y)$ is the conditional utility function on y given a value of x, and $u_x(x)$ is the conditional utility function on x given a value of y [4]. The constants k are weighting or scaling factors that represent the relative importance of x and y.

I then assess the utility of each of the four consequences using the formula listed above. To do this, I assess utilities for the different levels of x and y that appear in the consequences. This is done via a question and answer process. The best possible consequences are given a utility of 1, while the worst possible consequences are given a value of 0. Ms. Browns' preferences appear in the table below:

$u(0 \text{ days late}) = 1$	$u(\text{many modules, doc}) = 1$
$u(7 \text{ days late}) = \frac{3}{4}$	$u(2 \text{ modules, doc}) = \frac{2}{3}$
$u(14 \text{ days late}) = \frac{1}{2}$	$u(\text{monolithic, doc}) = \frac{1}{3}$
$u(28 \text{ days late}) = 0$	$u(\text{monolithic, no doc}) = 0$

Thus, using the formula listed above, and scaling factors $k_x = 1/2$, $k_y = 3/4$, and $k_{xy} = -1/4$, I have the following utilities for the outcomes.

$$\begin{aligned} u(\text{Consequence A}) &= \frac{1}{2} \\ u(\text{Consequence B}) &= \frac{9}{16} \\ u(\text{Consequence C}) &= \frac{2}{3} \\ u(\text{Consequence D}) &= \frac{3}{4}. \end{aligned}$$

Now that I have this information, which approach should Ms. Brown employ? The third step is to determine which approaches will lead to which consequences. Ms. Brown believes that the traditional approach will result in either Outcome B or Outcome C happening, and the experimental approach will result in either Outcome A happening or Outcome D happening.

Thus, the expected utility of the traditional approach is :

$$u(a_{\text{traditional}}) = p \cdot u(\text{Outcome B}) + (1-p) \cdot u(\text{Outcome C}), \text{ where } p \text{ is the probability that Outcome B will occur.}$$

The expected utility of the experimental approach is:

$$u(a_{\text{experimental}}) = q \cdot u(\text{Outcome A}) + (1-q) \cdot u(\text{Outcome D}), \text{ where } q \text{ is the probability that Outcome A will occur.}$$

Ms. Brown estimates that the values of p and q are 0.5 and 0.75 respectively. This means that $u(a_{\text{traditional}}) = 59/96 \sim 0.61$ and $u(a_{\text{experimental}}) = 9/16 \sim 0.56$. The traditional approach has a higher expected utility.

An important question to ask is how utility theory should be employed in software design and process decision making. I have already asserted that utility theory can be

useful in cases where qualitative goals exist in addition to the maximization of money returned. However, can utility theory be used "right out of the box" as I have done here? Are there any properties of software engineering that make it unique with respect to the application of utility theory and deserve special attention? I propose that there are properties of software engineering that require special attention. These considerations are as follows: with software, the rate of change is immense and the technologies employed are often new. This implies that there is little time to make decisions and that when decisions are made, little information may be available to assist in the decisions. This lack of information greatly impedes the assessment of probabilities. Furthermore, circumstances change frequently due the high rate of change; this means that the probabilities of consequences and perhaps even the set of consequences themselves may change in time.

These special properties of software engineering suggest that it may be unwise to look for a single alternative or action that maximizes expected utility. Rather, it makes more sense to treat the decision problem as an exploration of the consequence space.¹ In our exploration, we see which alternatives are sensitive to changes in the probabilities assigned to relevant consequences or when additions or deletions to the set of consequences are made.

Suppose in my example above that p and q were unknown. Instead of computing expected utility values for $a_{\text{traditional}}$ and $a_{\text{experimental}}$, we would plot lines -- $u(a_{\text{traditional}}) = -5p/48 + 2/3$, and $u(a_{\text{experimental}}) = -p/4 + 3/4$. We can then study how different values of p and q change the answer we get. We can also explore removing outcomes and adding in new outcomes and seeing how the answer changes. Utility theory simply provides the structure with which to explore the decision-making space.

In conclusion, I believe that utility theory can be of use when solving software decision problems with multiple objectives. The approach is also useful when money is the sole criterion but other easier-to-measure criteria are used as proxies. Utility theory brings structure and method to what is often an ad hoc decision making process.

I would like to thank Professor A. Ravindran of Pennsylvania State University for his assistance in finding sources on utility theory. I also thank Ken Yasuhara of the University of Washington for his time and valuable input.

- [1] Sullivan, K. "Software Design: The Options Approach". 2nd International Workshop on Software Architecture, Proceedings of the Second Joint SIGSOFT/ACM Conference on Foundations of Software Engineering, October, 1996.
- [2] Keeney, R. and Raiffa, H. Decisions With Multiple Objectives. USA: John Wiley & Sons, Inc., 1976.
- [3] Lindley, D. Making Decisions, 2nd. ed. Great Britain: John Wiley & Sons Ltd., 1985.
- [4] Golub, A. Decision Analysis: An Integrated Approach. John Wiley & Sons, Inc., 1996.

¹ This approach is not a novel idea. However, well-meaning researchers trying to apply utility for the first time often make the mistake of looking for "an answer" as a result, rather than a greater understanding of the problem and the alternatives at hand.