The Inaccurate Conception

Some thoughts on the accuracy of estimates.

1 estimate \(\text{\textit{est-ta-m\textasciitilde vt}}\) [L \textit{aestimatus}, pp. of \textit{aestimare} to value, estimate]
   1 archaic a: ESTEEM b: APPRAISE
   2 a: to judge tentatively or approximately the value, worth, or significance
   b: to determine roughly the size, extent, or nature of c: to produce a statement of the approximate cost of
   Merriam-Webster’s Dictionary

The pursuit of an “accurate estimate” for a system development project is an article of faith among project planners and managers. Methods, tools, approaches, and formulae to devise the accurate estimate are the subject of articles, conference presentations, and books. The fact that the juxtaposition of the words “accurate” and “estimate” produces an oxymoron does not seem to deter people from the quest for what is clearly a contradiction in terms.

Are we looking for the wrong estimation criterion?

Whether Forecast?
When a weather forecast indicates a 40% chance of rain and it rains on you, was the forecast accurate? If it doesn’t rain on you, was the forecast inaccurate? Thought of in these terms, the concept of accuracy takes on a different meaning.

It seems that the fact it does or does not rain on you is not a particularly good measure of the accuracy of the rain estimate.

The Cone of Uncertainty
In his excellent book, \textit{Software Estimation: Demystifying the Black Art} [3], Steve McConnell disinterred the “Cone of Uncertainty” originally shown years ago by Barry Boehm [2].

The Cone of Uncertainty describes the typical range of attainable accuracy of a software project estimate at different times in the project life cycle. Figure 1 shows that in the early stages of the project, the range might be from four times the final actual result to one quarter of the actual. Not very accurate, is it? Both authors assert this is about the best you can expect to do at that point in time, though the actual range is highly situational. That is, we could do worse than 4X–0.25X, but we can’t realistically expect to do much better. But why is this true?

Certainty comes from knowledge of something. To use a mathematical analogy, we have knowledge and therefore certainty when we are aware of the presence of a variable; we also know its value. On the other hand, uncertainty comes from things we don’t know. There are two types of things we don’t know: we are aware of the presence of a variable but don’t know its value or are unaware of the presence of a variable (and of course don’t know its value, too). The presence of knowledge is Zero Order Ignorance (0OI), and the two types of uncertainty are First and Second Order Ignorance.
(1OI and 2OI) [1]. Except under very special circumstances, when we begin a project there are many things we don’t know. Some of these things we are aware of (1OI) and some we are not (2OI). It is this lack of knowledge, particularly the 2OI variety, that generates the uncertainty. Since software development is primarily a knowledge-acquisition activity, we spend most of our time discovering this knowledge. Simply put: over time our knowledge grows and our ignorance diminishes.

The ranges mapped out by the Cone of Uncertainty describe a probability function with respect to time (or budget, staffing, or other project attributes—see Figure 2). In early phases of the life cycle, the width of the Cone of Uncertainty at that point in time generates a rather flat probability distribution (see Figure 2). This means there is a wide range of answers that might generate a solution. If the function is extremely flat, the model will infer the project should take somewhere between a few weeks and a few decades. Since every project takes somewhere between a few weeks and a few decades this estimate might be accurate but is not useful in making a business decision.

**HOLD TO 10%**

I recently spoke to a senior executive at a software consultancy. He was emphatic that when his project managers create an estimate, he will hold them to within 10% of the value they come up with. The Cone of Uncertainty would seem to fundamentally contradict the feasibility of that approach. The reason why an estimate cannot usually beat the Cone is that information about the project is imprecise at that point in time; the data is unavailable, unpredictable, inexact, or just plain wrong. No mathematical equation can have an output more precise than its input. In fact, this executive’s hard-nosed approach to estimation will likely generate defensive behavior on the part of his project managers and estimators. Specifically, it may encourage them to try to overstate the problem, to “pad” the estimate in the hope of getting more resources. More resources of time, budget, and staff always means the project has a higher probability of achieving its goals within those resources. Management, however, is often allergic to estimate padding and
seeks to remove it. Removing the additional resources naturally has the effect of making the project riskier.

**MANAGEABLE RISK?**

The probability of success and its associated risk is a function of what is known and what is not known. Some risk may be manageable with a little effort, and we can usually attenuate risk if we are willing to expend a lot of effort. But some risk may not be manageable or predictable at all. The Cone of Uncertainty simply maps onto the knowledge/unknowledge content of the project over time.

It is interesting to note that the executive mentioned here does not really want an accurate estimate; what he wants is to have some assurance the project will be delivered within some constraint of budget, time, and staff. It might seem this is the same thing as an accurate estimate, but it’s not. Perhaps this is where we should look for “accuracy”?

**THE ACCURATE COMMITMENT**

I recently completed an estimate of a large project for a client. It directly supported the estimate done by the project’s program management office, but I calculated the probability of achieving the stated budget at 65%, which was just about where it should be. Since this probability was higher than expected, the project manager asked “Can I quote a lower price then?” The answer is, of course, yes—but at a higher level of risk. In fact, he could quote the price a lot lower, if he was also willing to take a very high risk of not actually achieving it. This is where accuracy is really measured.

We don’t need an accurate estimate; we need an accurate commitment.

The commitment is the point along the estimate probability distribution curve where we promise the customer and assign resources. This is what we need to hit, at least most of the time. It is not a technical estimation activity at all but is a risk/return-based business activity. It is founded on the information obtained from the estimate, but is not the estimate. Using Figure 3 as an example, if we needed an accurate commitment in the earliest (Initial Concept) phase based on how the diagram shows the project actually worked out, we would have had to commit at around a 75% probability. From the figure, committing to the “expected” result at Initial Concept would have led to a significant overrun beyond that commitment, and the project would have “failed.” We can consider the 50% (expected) result to represent the cost of the project and the 25% increment to the higher commitment level to represent the cost of the risk of the project.

The accuracy of a weather forecast is not whether it rains or not but whether it rains at the likelihood it was forecast to rain. Similarly, the accuracy of an estimate on a project is not whether the project achieves its goals but whether it correctly forecasts the probability of achieving its goals. In fact, I saw this recently, where a project had been committed at an estimated 20% of success. While
the project failed we could (and did) reasonably argue that the estimate was, in fact, accurate since it correctly forecast the failure; paradoxically, if the project had been successful we could consider the estimate to have been wrong.

The problem wasn’t the estimate but the commitment. If we commit to a low probability of success on a project we are gambling with our budget, staff, and customer relations and that is often not a good thing.

**Rein in the Commitment**

If there is a 40% chance of rain and I decide to walk outside in my expensive suit without an umbrella and it rains on me and ruins my suit, it is not the fault of the forecast; it is the fault of my decision.

If we calculate the level of risk we are taking on a software project and choose to roll the dice with the odds stacked against us, it’s not the fault of the dice. It is neither fair nor, well, accurate to blame a good estimate for a bad commitment.

**References**


**Phillip G. Armour** (armour@corvus-intl.com) is a senior consultant at Corvus International Inc., Deer Park, IL.

© 2008 ACM 0001-0782/08/0300 $5.00

DOI: 10.1145/1325555.1325558