Project Portfolios: Organizational Management of Risk

Assessing companies' differing attitudes toward identifying and calculating risk-taking endeavors.

Be truthful now, how did your 401(k) plan look at the end of the dot-com bust? All those tech stocks and their 25%+ annual gains kind of hit the brakes, didn't they? I own my part of the "irrational exuberance" that Federal Reserve Chairman Alan Greenspan accused us all of. And I suspect, like me, you paid a hefty price for your unbridled optimism. Don't get me wrong, I think optimism is good. But it must be realistic if you don't want to pay a high price.

Some companies involved in the business of software have a very curious attitude toward risk. On one hand, they may assert that they are risk tolerant, even bragging about it as if it was some measure of development vitality. Such companies may routinely embrace highly risky project scenarios. In these situations, I have heard cautionary voices discounted as nay-sayers, or even accused of not being team players. On the other hand, I have seen companies that willfully and enthusiastically embark on high-risk projects exhibit astonishment or even anger when such projects "fail"—casting about for some hapless scapegoat to take the blame for the decision and its failure. It is as if the concepts of risk and failure are somehow disconnected. This is a strange situation and it reminds me of some peoples' attitudes toward financial investments—if we merely think it will be so, it will be so. Well sometimes it might, but usually it won't.

All Projects Are Risky

Every software project involves some degree of risk. The reason is that, at the time we start a project, there are always key variables of the project that are simply not known. In fact, I contend the primary task of most software projects is to discover and resolve these unknowns rather than to build a system. Or perhaps more pointedly, the task of resolving the unknowns constitutes the bulk of the work and risk on the project. Management, whose primary function is to exert control, usually requests and requires that decisions are always guaranteed to be correct and that no surprises occur.

While we can and should do a lot more to avoid being surprised (sometimes over and over again) by the same problems we have experienced in the past, we cannot remove all variance between this project and previous projects. The primary purpose of any development is to do something we have not done before. If we

MICHAEL SCHRÖTER

When the payback at a given level of resource allocation exceeds the cost at a certain level of risk, it is a good plan for that project.

were able to remove all sources of variance, we would also remove the primary purpose of the project. Naturally, the number and type of these unresolved variables determines the level of risk. Projects that have most of their variables defined up front have little if any risk. Of course, the only way a project could have *all* variables predetermined would be if we had already built this exact system, in which case the compelling question would be: why are we building it again?

Give Me Your Best Estimate

"Give me your best estimate" is a common if poorly defined request made of estimators. It usually means "…something very soon." When presented with such a demand, I have occasionally replied with something like: "you can have the software next week, as long as you are willing to tolerate a 0.000001% chance of it working properly." While this is a semantically and statistically correct statement, I caution readers to use it with care. Delivered to an unsympathetic, stressed, or humorless boss, it might have career-modifying properties. That said, it has sometimes initiated a dialogue along the following lines:

- The person requesting the estimate has some nominal, though unstated, expectation of probability of success.
- That probability is not 0.000001%.
- The person would like it to be 100% guaranteed but (intellectually, if not emotionally) understands that, since the project is neither death nor taxes, it is probably not going to be quite so certain.
- Ergo, the expected probability lies somewhere between one chance in 100,000 and 99,999 chances in 100,000.

Getting someone to intentionally fix on, articulate, and commit to a specific success probability within this range can be a challenge. But why? Intellectually the case is clear. We know that too low a probability is, well, too low. And we know that, realistically, we usually cannot pay for, or wait for, a 100% guarantee even if it could be achieved. So why the difficulty in deciding just where to draw the line?

Some You Win…

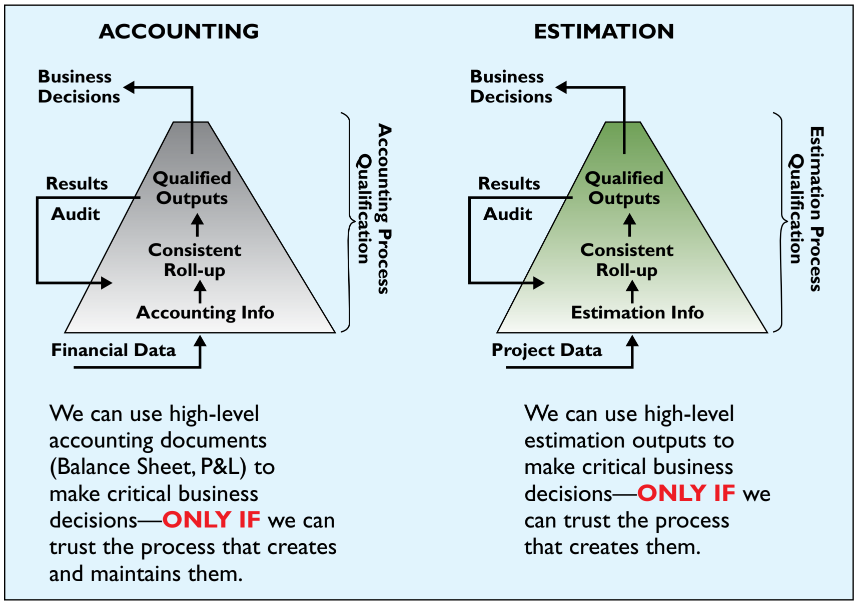
The reason for the difficulty is quite straightforward. Suppose we decide that 85% is a good probability of success for a given project. We then have to ask ourselves this question: *If we agree to and sign off on a probability of success of 85%, what else are we agreeing to?* If we accept that we require better than six chances in seven that we will provide to our customer the contracted functionality within the constraints of time, staff, and budget we have established, we must also accept that we can tolerate a *one chance in seven of failure* to provide this functionality under the same terms. This apparent open acceptance of the prospect of "failure" appears to be anathema to many organizations and many executives. The problem is that we want certainty of success. We are just not willing to pay for it.

In reality, there is a universe of possible project scenarios in which we allocate different levels of staff, budget different levels of cost, expect different schedules for delivery with different degrees of delivered functionality and embedded defects. For each of these scenarios, there is a finite probability of meeting a given set of development goals with a given set of development assets and resources. We can always adjust this probability. There is no trick to being uniformly successful in managing projects—simply allocate more resources, better people, and take longer.

Handicapping Horses

There is no trick to figuring out which horse is most likely to win a race at the racetrack: it is the favorite. It is always the favorite. Not that the favorite always wins, but it does so more often than other horses—which is, of course, why it is the favorite. But good handicappers do not just pick favorites. The essence of handicapping is not predicting the winner—it is maximizing return on investment. Good handicappers calculate and optimize the difference between the paid odds and what they perceive to be the actual odds of success. When the likely payout exceeds the perceived risk by a certain value, it is a good bet.

So it is for projects: when the payback at a given level of resource allocation exceeds the cost at a certain level of risk, it is a good plan for that project.



We can use high-level accounting documents (Balance Sheet, P&L) to make critical business decisions—**ONLY IF** we can trust the process that creates and maintains them.

We can use high-level estimation outputs to make critical business decisions—**ONLY IF** we can trust the process that creates them.

Equivalence of accounting and estimation.

No-Risk Projects

The problem with project planning in many organizations is they do not do an explicit risk calculation. Specifically, we do not set up structures that calculate and summarize risk in monetary terms so that decision makers can make intelligent decisions based on that risk. An analogy with accounting systems is appropriate (see the figure here). Companies are able to trust that their balance sheets can be used to make business decisions only to the extent their balance sheets are created by a valid, certified, and audited process. We run our project estimates and build our project plans, but we often don't know what level of risk we are taking when we commit to these plans. We don't know because the organization does not explicitly calculate and roll up the risk for all projects. We don't know because we don't ask what level of risk we are willing to pay for, and often we are not willing to accept the negative side of the risk as a conscious and intentional decision.

Sometimes risk is expressed in lists of "open issues" or in broad categorizations such as "high risk" and "low risk." Sometimes people are asked what level of confidence they have in their estimates. There is nothing wrong with these measures, but they are not easy to use to make critical decisions. Rarely is risk expressed in terms of money and what, say, the cost of an 85% probability of success would be versus an 80% probability.

Calculating Risk

Intrinsic risk is not easy to calculate on many projects because we are trying to measure what is not known. The known aspects and attributes of a new project are not hard to quantify, but that's not what takes the time, generates the work, and causes us problems—it is what we don't know that kills us. But the fact that risk calculation is difficult doesn't mean we shouldn't do it. The value of an explicit risk calculation in making more optimal decisions is very high. I have found a number of approaches very helpful when used judiciously, including:

• Identifying likely sources of Second Order Ignorance (what we don't know we don't know) [1] since this is the most intractable form of lack of knowledge to resolve. As a rule of thumb, anything with the adjective "new" in front of it is a pretty good place to start.
• Identifying likely variance ranges of typical unknowns. These include things like "three point sizing" (minimum, most likely, maximum) values for system size and scope metrics or the simpler min-max scoping. This can also be done by sampling the variance from a range of experts' opinions. I have found the Rand Corporation's Delphi approach to be very helpful in doing this.
• Causal analysis of historical estimate variance to identify where and why estimates were off and then extrapolating them to this situation.
• Statistically valid combination of risk factors. This is a topic that has occupied the attention of insurance companies for a very long time and is a science unto itself. Stochastic methods such as Monte Carlo can be helpful here.
• Applying standardized and defensible calibrations and procedures to convert input scope and asset factors into the appropriate resource prediction outputs necessary to make a good business decision.
• Factoring the variables into business scenarios, each incorporating risk assessments in the key limiting business areas.
• Establishing organizational structures and processes to roll up risk to the business-unit decision level.
• Refactoring risk into valid numerical business information [2]. The most effective metrics are usually ones beginning with a "$" sign. I have found this to be essential in getting risk management to actually function the way it needs to. Qualitative risk factors and lists of open points and issues are all well and good, but you can't beat a good monetary difference in cost or profit to focus peoples' attention and get them to make a decision.

Portfolios and Risk

In many respects, a collection of projects is like an investment portfolio for a company [3]. Each project is allocated and consumes assets. From each project we expect a reasonable return on our investment. But in the business of software, companies often do not know what level of risk they are taking when they sign off on a project plan because they have not set up the necessary processes to calculate and inform on the risk. And anyway, they haven't asked for it and sometimes won't accept it even if it were given. Most companies I deal with put intense pressure on the delivery schedule of *all* their projects, which has the effect of significantly increasing the level of risk (and cost and defects) across the board. This often comes without an equivalent increase in the level of return. In doing so, these companies may be inadvertently setting up a whole portfolio of high-tech junk bonds. As a business, we need to change this behavior, because it is not a good business practice. It is OK to invest in high-risk–high-return investments. It is not OK to invest in something that is high risk without the high return. It is also not OK to invest in something that is high risk and not know it is high risk, or to pretend or act like it is not high risk. The reason is that high-risk projects fail. Often. That's why they're considered high-risk projects.

And if your retirement income, your children's college fund, your career, or your company's future is linked to this investment, then prudence, discretion, and now Sarbanes-Oxley, dictate we know what level of risk we are taking, it is a reasonable one with respect to the likely return, and we don't take a big risk in *every* investment we make. C

References
1. Armour, P.G. *The Laws of the Software Process*. Auerbach Publishers, 2003, 9.
2. Ferguson, R.W. A project risk metric. *Crosstalk* (Apr. 2004), 12–15.
3. Henig, P.D. The efficient frontier. *CIO Insight* (June 2004), 28.

Phillip G. Armour (armour@corvus-intl.com) is a senior consultant at Corvus International Inc., Deer Park, IL and a research director in the Center for Software Development Innovation at Number Six Software, Arlington, VA.