Software Risk Engineering

By

Barry Blesser and Derek Pilkington

In the conceptual stage of a project, management is typically interested in crude estimates to enable go/no-go decisions relating to market windows, return-on-investment, and resource requirements. During this phase, management usually tolerates high variance estimates in order to minimize the initial engineering investment. The tolerance for variance reduces dramatically after management decides to proceed with a project; development estimates now influence business financial and marketing plans. Variance arises from uncertainty and risk, yet few developers and managers understand the principles of Software Risk Engineering, or even have a language to communicate in appropriate terms.

The discipline of risk engineering evolved primarily in large machine industries such as power plants and aircraft manufacturing. This discipline has a direct application to the high-tech software development process. In this discussion, we focus on the methods and consequences of an idealized factoring of the development process to optimize schedule and quality. All software development projects contain two conceptual domains: work and uncertainty.

Work Domain

The work domain is well known and understood by developers and management. A given collection of tasks will take a certain amount of person-hours to reach closure. Work-tasks are primarily those with little uncertainty about the ultimate success and resulting properties.

Uncertainty of Estimate

Work-task uncertainty arises in the estimating process, not in the work itself. Reducing this estimate variance requires a detailed architecture, planning and modularization before estimation. Typically, those providing an estimate must do so in the form of a single number. In fact, the real answer is a classic probability-density curve. For example, a single well-defined task has a variance in the following form; 5% chance of finishing in 3 days, 33% in 4 days, 50% in 5 days, etc. This is the language of probability. With 1,000 such tasks, a complete project will have very large variances, perhaps a factor of two or more.

The variance will likely have a Log Normal distribution, i.e. very long forward tails, as shown in figure 1. The chance of completion at any point along the horizontal time axis is the area under the curve to the left.

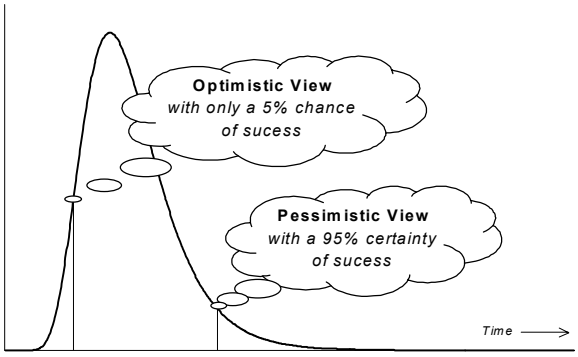

Figure 1

Such uncertainty is unacceptable to a manager who needs "firm" dates to plan a marketing launch and production. The developers confront the choice of how to convert a distribution into a single date or number. Should they report the target schedule optimistically (corresponding to a 5% chance of success), or should they report the target pessimistically (corresponding to a 95% chance of success?) Since these numbers may differ by a factor of two or more, the choice is often an unspoken psychological one. If the de-

velopers want to do the project, they report the schedule optimistically. For a distasteful project, they report pessimistically.

Figure 2 illustrates this choice using the same data as figure 1, only represented with the vertical axis showing the percent chance of completion against the horizontal time scale.

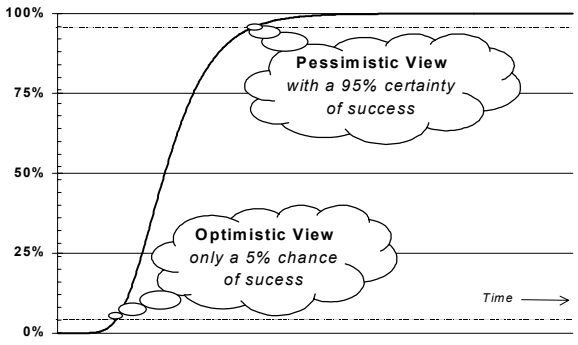

Figure 2

Consider possible outcomes:

1. The estimate does not meet business needs; management and developers negotiate. Developers succumb to pressure to reduce the estimate, picking a different point on the probability curve as the estimate, ignoring the reality of variance.
2. Management proceeds based on an optimistic estimate, the project "drags on." The business suffers from delayed introduction or quality/features compromises made to meet delivery expectations.
3. The estimate exceeds business needs, management scraps the plan because of the poor ROI, or because a window of opportunity would be missed.

None of these scenarios is attractive since they only have the illusion of informed decision making. A true negotiation should deal with how to reduce the variance, by increasing resources, redefining features, etc. Alternately, the negotiation should be an open discussion of the risks of accepting certain points on the curve based on the tolerance of the business and project to accept certain levels of risk.

Uncertainty of Solution

The previous discussion focused on "uncertainty of estimate" with an implicit assumption that the tasks themselves were straightforward, albeit of unknown effort. There is another type of risk related to the development itself. The structure of a development project cannot take into account unforeseen problems. Developers traditionally call these surprises. Complex projects often have many surprises, yet few managers formalize the management of these unknowns. Surprises can often dominate a schedule.

It is a very different skill to explore unknown difficulties compared to inventing a solution for known problems. For example, when trying to make an operating system perform a complex task, the developer must confront not knowing if the OS can even do such a task. An operating system is sufficiently complex that even its creators may not know if, or how, a given property can be made to function. In the end, the software solution may require only a few lines of code, but it may require one day or three months to understand what this code has to do. Worse yet, it could be unknown if a solution even exists. When no analogous working example of a similar solution exists, the developer continually confronts the fear of looking forever for a nonexistent solution. (This is one of many cases where the development process experiences non-convergence.) We refer to this as the "uncertainty of solution." The "uncertainty of estimation" and the "uncertainty of solution" are fundamentally different.

The developer's tendency towards a concrete psychology and management's need for hard determinism conspires to ignore unknowns. The most tractable version of this issue involves the debugging process. If the first 3 days of testing yields 15 bugs per day, then one can reasonably assert that there will be another 15 bugs on the 4th day of testing. The nature of the bugs is unknown, but statistical patterns predict their occurrence. This is a simple example of managing uncertainty. In a well-run project, a plot of bugs (per unit of testing) determines the convergence rate. By analogy, a given project will have a

certain surprise rate, even if one cannot make any statement about the details of the surprises. There are numerous other examples of these aspects of the development process.

Optimizing Variance or Mean

The combination of "uncertainty of estimation" and "uncertainty of surprise" provides a clear explanation of why complex projects are usually late. When managing a project and the underlying probability density curve, we must be aware of two very different attributes of that curve: variance and mean. It is possible to optimize one or the other. Reducing the variance produces more clarity, while reducing the mean has more economic utility. In some companies, the value system implies that a reduced variance (more reliable schedules) justifies extending the delivery time. Extensive early planning and experimentation will narrow the probability density curve but move it right. Conversely, other companies push to accelerate the schedule even if the uncertainty increases. This might correspond to doing very little planning but using dynamic management to handle the surprises. Figure 3 illustrates these two choices compared with a middle model (modest amount of planning and research conducted.)

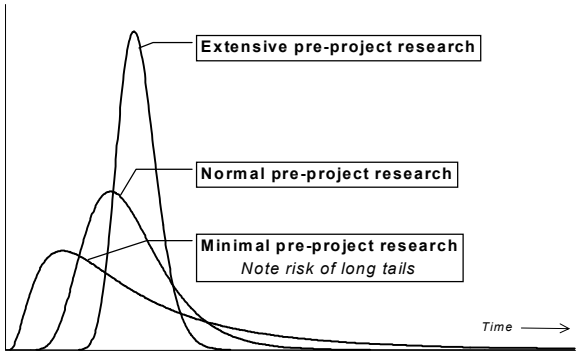

Figure 3

Again, for those who are more comfortable with a percentage chance of completion view we present figure 4. The 95% percent chances of completion for the three curves indicate the risk due to too little pre-project research / planning. In this example there exists a 2.4 to 1 time difference between the minimal and extensively planned 95% chance points. In other words, an extensively planned and researched project that has a 95% chance of completion in 5 man years could, if not researched sufficiently, take 12 man year to reach the same chance of completion.

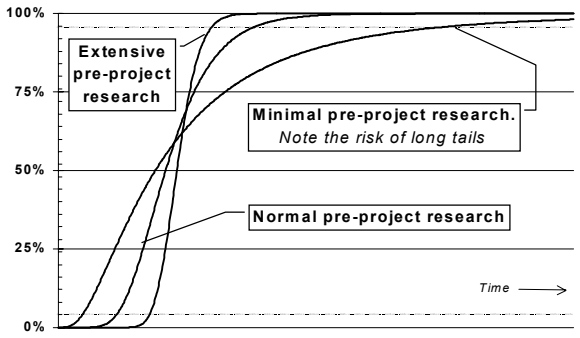

Figure 4

The specific business situation should be the determining factor in selecting an optimization. A business running many projects concurrently that is not dependent upon any single project for financial success can afford to take risks and lean towards a minimal pre-project phase. However, a business that is running a single and financially critical project is best served by a larger pre-project phase.

Milestone Risk

Traditional project milestones offer a way to avoid the pain of probability discussions. They give management a secure feeling in their ability to track progress; developers have a clear way to demonstrate success and competence. However, this does not solve the problem of "uncertainty of estimation" and "uncertainty of surprise." The creation of traditional milestones embodies and embeds a value trade-off system (optimization) of the uncertainties. Unfortunately, hiding the uncertainty does not allow anyone to see the optimization method. By default and often unaware, developers and management agree on a metric that raises the likelihood of all but the last milestone being met since this produces the optimum professional and emotional payoff; regretfully at the expense of de-optimizing the final economic result. Raising the probability of meeting the early milestones is straightforward; postpone

high-risk tasks until the end. In other words, delay the pain and only miss the last milestone of product release and delivery. Consider the converse, only the last milestone was successful, but all of the early ones were late.

The hidden approach of back-loading risk is the worst possible way to manage uncertainty. We can illustrate this in two trivial examples. Software developers may make an untested assumption about a key piece of design and then spend months writing the code. Hardware designers may make the assumption of 100% accuracy in all the manufacturers data sheets and then spend months developing a working board. Changing either of these assumptions will result in discarding a large amount of work. Delaying risk until late in a project adds the danger of having to discard large amounts of real work for little or no gain. Hence, it should be obvious that the optimum approach is to invest the right amount of effort in risk reduction before doing any "real" work. This is true even if it requires throwaway experiments to prove the viability of assumptions. Unfortunately, front-loading risk will show the least amount of progress in a traditional milestone sequence because there is no tangible "real" result. It is easier to show progress by doing the straightforward work.

A similar example involves the fact that a large percentage of a development project can take place in the tails (debugging and fine-tuning), after the system is nominally working but before achieving the required quality. It is often more economical to build in a self-diagnosing software architecture that dramatically reduces the debugging time. Again, this is an example of front-loading the issues that normally appear at the end of a project – during the last milestone.

Conclusion

A traditional theory is that variance gradually reduces as the project progresses, highest at the beginning and lowest at the end. This assumes that variability reduces as a function of the work completed. However, it does not take into account the amount of risk that remains in a project. Risk manifests itself as surprises. Developers and managers should always keep in mind the well-understood principle that the consequences of surprises depend very heavily on when discovered during the development process.

None of the techniques for risk optimization is particularly complex but they often go against the organizational biases of both developers and management. Managers and developers need education to appreciate the economic return of handling risk explicitly. This kind of language is, however, not a natural part of the professional life of either developers or managers. While the economic rewards (investment, productivity, and time-to-market) provide a dramatic competitive advantage, the logic of risk management does not significantly reduce the effort required to overcome human biases. In other words, the simplicity of these concepts does not imply that a change in behavior is easy.