Software Estimation's Cone of Uncertainty

Steve McConnell, Construx, Inc. from the Construx web site)

Introduction to the Cone of Uncertainty

Early in a project, specific details of the nature of the software to be built, details of specific requirements, details of the solution, project plan, staffing, and other project variables are unclear. The variability in these factors contributes variability to project estimates -- an accurate estimate of a variable phenomenon must include the variability in the phenomenon itself. As these sources of variability are further investigated and pinned down, the variability in the project diminishes, and so the variability in the project *estimates* can also diminish. This phenomenon is known as the "Cone of Uncertainty" which is illustrated in the following figure. As the figure suggests, significant narrowing of the Cone occur during the first 20-30% of the total calendar time for the project.

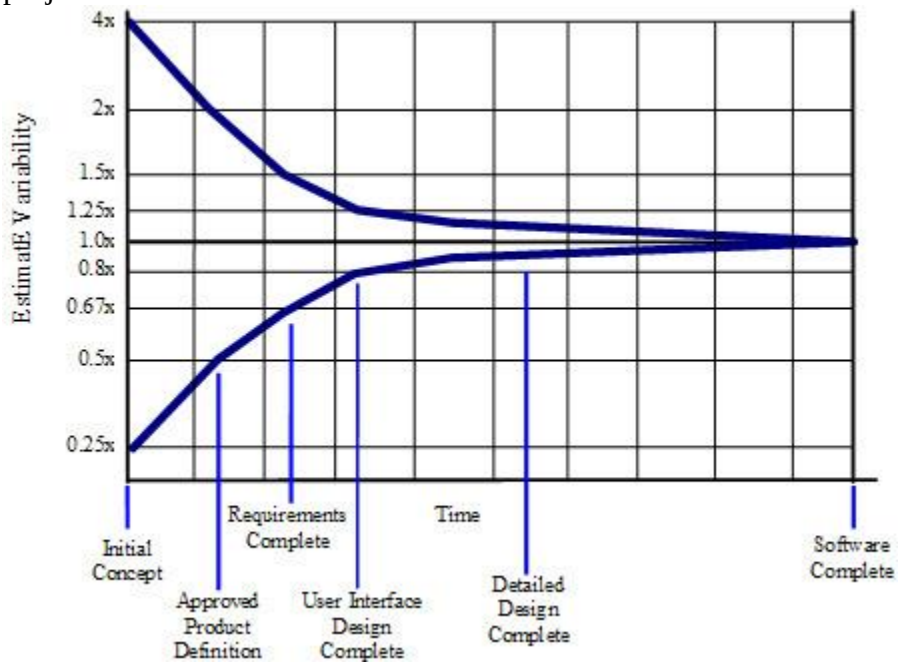

Figure 1 The Cone of Uncertainty

The horizontal axis contains common project milestones such as Initial Concept, Approved Product Definition, Requirements Complete, and so on. Because of its origins, this terminology sounds somewhat product oriented. "Product Definition" just refers to the agreed upon vision for the software, or "software concept," and applies equally to web services, internal business systems, and most other kinds of software projects.

The vertical axis contains the degree of error that has been found in estimates created by skilled estimators at various points in the project. The estimates could be for how much a particular feature set will cost and how much effort will be required to deliver that feature set, or it could be for how many features can be delivered for a particular amount of effort or schedule. This book uses the generic term "scope" to refer to project size in effort, cost, features, or some combination.

As you can see from Figure 1, estimates created very early in the project are subject to a high degree of error. Estimates created at Initial Concept time can be inaccurate by a factor of 4x on the high side or 4x on the low side (also expressed as 0.25x, which is just 1 divided by 4). The total range from high estimate to low estimate is 4x divided by 0.25x, or 16x.

Narrowing the Cone of Uncertainty

One question that managers and customers ask is, "If I give you another week to work on your estimate, can you refine it so that it contains less uncertainty?" That's a reasonable request, but unfortunately it's not possible

to deliver on that request. Research has found that the accuracy of the software estimate depends on the level of refinement of the software's definition. The more refined the definition, the more accurate the estimate. The reason the estimate contains variability is that the software project itself contains variability. The only way to reduce the variability in the estimate is to reduce the variability in the project itself.

An important—and difficult—concept is that the Cone of Uncertainty represents the best case accuracy it's possible to have in software estimates at different points in a project. The Cone represents the error in estimates created by skilled estimators. It's easily possible to do worse. It isn't possible to be more accurate; it's only possible to be more lucky.

Another way in which the Cone represents a best case estimate is that, if the project is not well controlled, or if the estimators aren't very skilled, estimates can fail to improve as shown by the Cone. Figure 2 shows what happens when the project isn't conducted in a way that reduces variability—the uncertainty isn't a Cone, but rather a Cloud that persists to the end of the project. The issue isn't really that the estimates don't converge; the issue is that the project itself doesn't converge, that is, it doesn't drive out enough variability to support more accurate estimates.

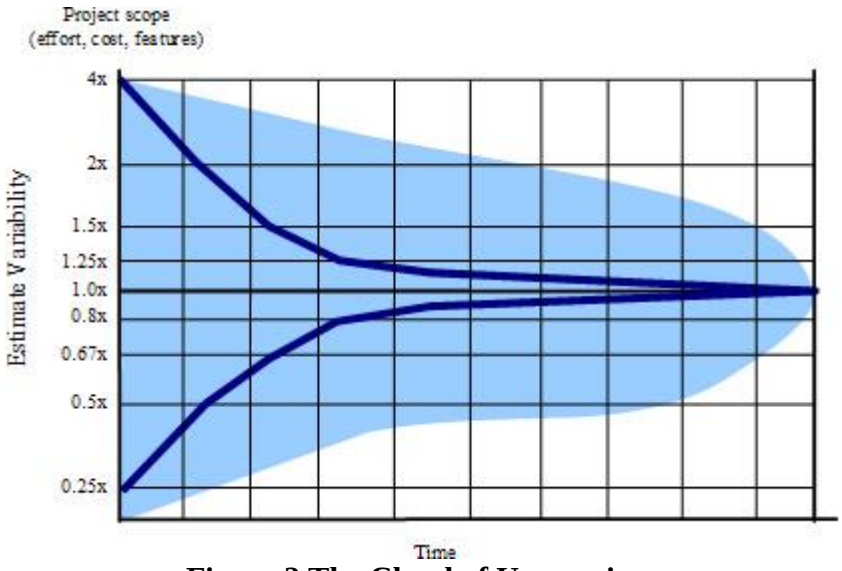

Figure 2 The Cloud of Uncertainty

The Cone narrows only as you make decisions that eliminate variability. As Figure 3 illustrates, defining the product vision (including committing to what you will not do), reduces variability. Defining requirements—again, including what you are not going to do—eliminates variability further. Designing the user interface helps to reduce the risk of variability arising from misunderstood requirements. Of course, if the product isn't really defined, or if the Product Definition gets redefined later, then the Cone will get wider, and estimation accuracy will be poorer.

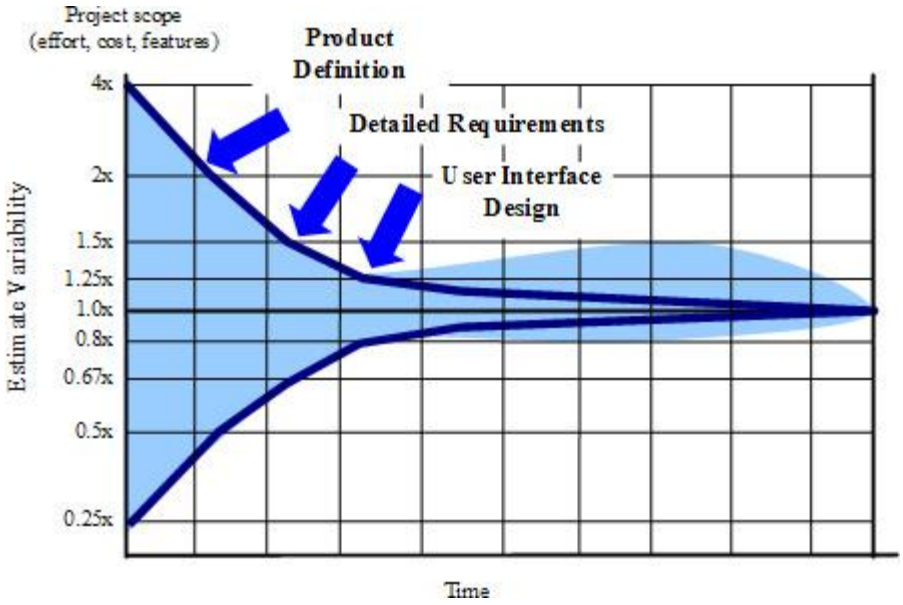

Figure 3 Forcing the Cone of Uncertainty to Narrow

Relationship Between the Cone of Uncertainty and Commitment
Software organizations inadvertantly sabotage their own projects by making commitments too early in the Cone of Uncertainty. If an organization commits at Initial Concept or Product Definition time, it has a factor of 2x to 4x error in its estimates. Commitments made too early in a project undermine predictability, increase risk, increase project inefficiencies, and impair the ability to manage a project to a successful conclusion.

Meaningful commitments are not possible in the early, wide part of the Cone. Effective organizations delay their commitments until they have done the work to force the Cone to narrow. Meaningful commitments in the early-middle of the project (about 30% of the way into the project) are possible and appropriate.

The Cone of Uncertainty and Iterative Development
The application of the Cone of Uncertainty to iterative projects is somewhat more involved than it is to sequential projects.

If you're working on a project that does a full development cycle each iteration—that is, requirements definition through release—then you'll go through a miniature Cone on each iteration. Before you do the requirements work for the iteration, you'll be at the "Approved Product Definition" part of the Cone, subject to 4x variability from high to low estimates. With short iterations (less than a month), you can move from "Approved Product Definition" to "Requirements Complete" and "User Interface Design Complete" in a few days, reducing your variability from 4x to 1.6x. If your schedule is fixed, the 1.6x variability will apply to the specific features you can deliver in the time available rather than to the effort or schedule.

What you give up with approaches that leave requirements undefined until the beginning of each iteration is long range predictability about the combination of cost, schedule, and features you'll deliver several iterations down the road. Your business might prioritize that flexibility highly, or it might prefer that your projects provide more predictability.

Many development teams settle on a middle ground in which a majority of requirements are defined at the front end of the project, but design, construction, test, and release are performed in short iterations. In other words, the project moves sequentially through the User Interface Design Complete milestone (about 30% of the calendar time into the project), and then shifts to a more iterative approach from that point forward. This drives down the variability arising from the Cone to about ±25%, which allows for good enough project control to hit a

target while still tapping into major benefits of iterative development. Project teams can leave some amount of planned time for as-yet-to-be-determined requirements at the end of the project. That introduces a little bit of variability related to the feature set, which in this case is positive variability because you'll exercise it only if you identify desirable features to implement. This middle ground supports long range predictability of cost and schedule as well as a moderate amount of requirements flexibility.