Paper AD11-2011

Software Time Estimation: Are We There Yet?

Jack Fuller, Experis Manpower Group, Portage, MI

Abstract

Whether a software project succeeds or fails is often predicated not on the abilities of its programmers or the elegance of its code but on the simple fact of whether it was delivered on time. This means that for many projects the most critical task for determining success or failure occurs not while writing actual code but instead at the very beginning of project when time estimates are being generated for the various tasks which make up that project. In order to become a better programmer, we have to first become a better estimator of how long that programming will take.

This paper will present practical techniques for estimating how long it takes to build software.

Introduction

I find that the most difficult thing I am asked to do in my daily job as a SAS® software consultant doesn't actually involve writing any lines of SAS code. Instead, it occurs when I am asked to give an estimate of how long a task will take to complete. Every estimate can be different:

- A task may consist of a single program, a set of programs or a user application
- A task may or may not be subject to project management principles
- An estimate may be used in isolation or it may be used in conjunction with other estimates
- A bid may be structured for time and materials or fixed costs

While time estimation in its entirety can be an overwhelming topic, my main concern lies with helping the individual programmer who is asked to come up with an estimate for a specific task: what are some of the basic concepts related to estimation; where does estimation error originate; and what are some simple techniques (with an emphasis on Project Evaluation and Review (PERT)) that can used to mitigate estimation error.

Basic Concepts

Targets, Commitments, Plans and Estimates

One of the first things to do when tasked to provide an estimate is to determine what is actually being requested. A project manager may ask for an estimate when what he or she really wants is a commitment to meet a target.

Target	A business objective
Commitment	A promise to deliver something within specified criteria
Plan	A scenario for fulfilling a commitment to a target
Estimate	The time and/or cost to complete a task

If you provide an estimate when you are really being asked to provide a commitment to hit target, your project manager may be tempted to reduce the estimate in order to hit the target. This is a bad idea since most people -- including programmers -- tend to be optimistic with their estimates. It is a better idea to treat the estimate as a fact and instead adjust something else: feature sets, target dates, deliverables, etc.

Steve McConnell (pp 4-7) provides some tips on how to use these distinctions between targets, commitments and estimates to better handle the interactions between managers and developers:

- Tip #1 - Distinguish between estimates, targets and commitments.
- Tip #2 - When you're asked to provide an estimate, determine whether you're supposed to be estimating or figuring out how to hit a target.
- Tip #3 - When you see a single point "estimate," ask whether the number is an estimate or whether it's really a target.

Estimates as Probability Distributions

One way to look at estimates is as probability distributions. A simplistic – and completely unrealistic -- example would involve a schedule where an estimate will be always exactly correct (see figure 1).

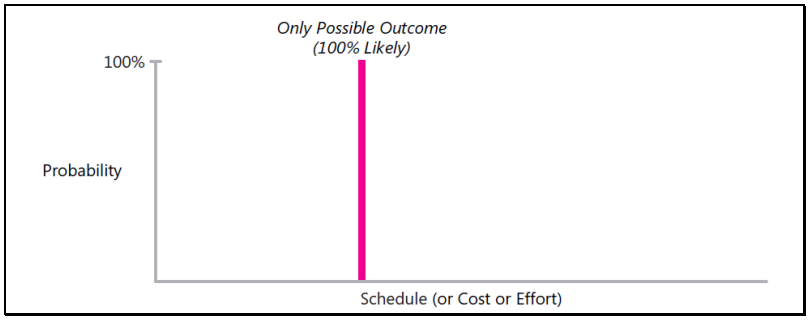

Figure 1
(McConnell p. 6)

Another way to look at estimates is to expect that estimates will follow a normal probability distribution (see figure 2). This is also incorrect.

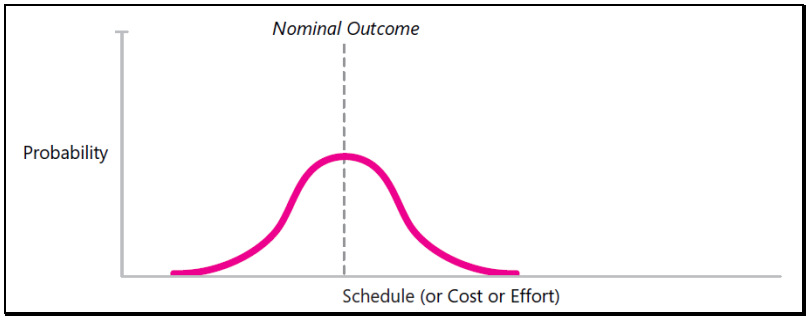

Figure 2
(McConnell p. 7)

While there is a limit to how quickly a task can be completed there is no limit as to how long it can take (see figure 3).

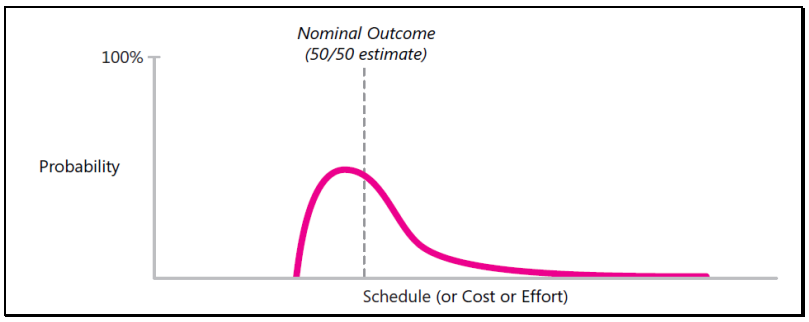

Figure 3
(McConnell p. 8)

Estimation's Purpose

The real purpose of estimates should not be to predict a project's outcome; instead, the real purpose should be to answer the question: should we go forward with this project? Experts have found that if the gap between the estimates and the targets is less than about 20 percent, a project manager can complete a project on time by adjusting project parameters like functionality, team size, timelines, etc. (McConnell p. 13). The main point should be to ensure that an estimate is useful, not that it is exactly accurate.

Estimation Error

Cone of Uncertainty

Estimating a task's duration is really, really hard. Try estimating the following:

- How long will it take to get to work in the morning?
- How long will it take to fly from Kalamazoo to Kansas City?
- How long will it take to put your kids to bed?

If we were to provide estimates throughout the duration of a task, we would see that our estimates improve over time as we knowledge increases about each task: your car has a flat:

- Your car has a flat
- It's snowing in Kalamazoo
- Grandma fed the kids caffeinated sodas

Software projects work the same way. As a project works to completion, our knowledge of the tasks and obstacles improves and we are better able to match our new estimates to reality.

The Cone of Uncertainty (see figure 4) demonstrates graphically how estimation variability decreases over the life of a project.

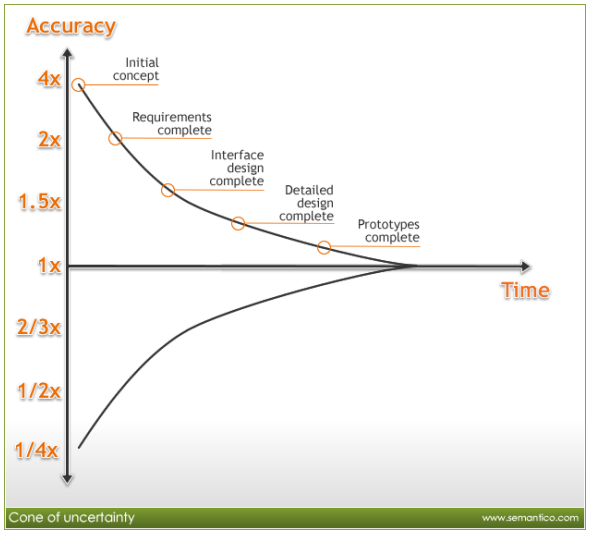

Figure 4 (Padley)

It is important to understand that the Cone of Uncertainty is actually a best case scenario. It takes a lot of hard work (detailed requirements gathering, change control, project management, etc) to force the cone to narrow. Without good project management practices, a project may never narrow to a cone; instead, it will exist as a nebulous cloud (see figure 5).

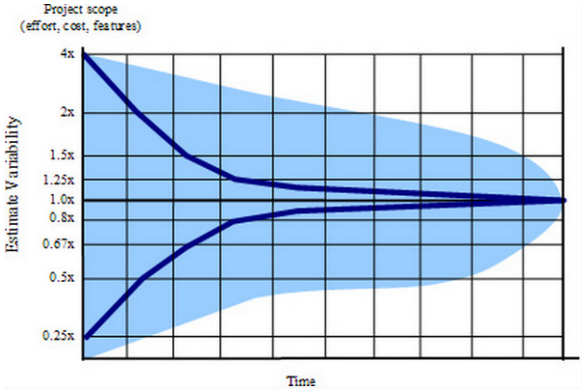

Figure 5
(McConnell, p. 38)

Among the tips Steve McConnell (pp 37-51) provides to deal with the Cone of Uncertainty are the following:

- Tip #11 – Consider the effect of the Cone of Uncertainty on the accuracy of your estimate. Your estimate cannot have more accuracy than is possible at your project's current position within the Cone.
- Tip #20 - Don't reduce developer estimates, they're probably too optimistic already.
- Tip #22 - Don't give off-the-cuff estimates. Even a 15-minute estimate will be more accurate.

Simple Estimation Techniques

Create a Work Breakdown Structure (WBS)

Create a Work Breakdown Structure where tasks are broken down into subtasks. The main reason for creating a WBS is that by the Law of Large Number and the Central Limit Theorem this will provide us with more accurate overall estimate. Essentially, overestimations and underestimation will tend to cancel each other out as our sample size of tasks increases.

- Law of Large Numbers
  - "… average of the results obtained from a large number of trials should be close to the expected value, and will tend to become closer as more trials are performed" (Wikipedia)
- Central Limit Theorem
  - "…the mean of a sufficiently large number of independent random variables, each with finite mean and variance, will be approximately normally distributed." (Wikipedia)

A second reason for creating a WBS is that it lends itself to the use of checklists and templates in order to prevent the omission of tasks. The following table lists examples of things that may need to be considered prior to providing task estimates:

Data Conversion	Project Management
Deployment	Security
Deployment Follow-up	Technical Documentation
Help System	Test Data Creation
Meetings	User Documentation
Performance Tuning	User Training

A third reason for creating a WBS is that it also lends itself to the idea of counting things and then multiplying by the average cost for each item. Don't be afraid to use large, coarse grained items early in the lifecycle of the project. This is one of the lessons of the Cone of Uncertainty. The following table lists examples of items that can be counted:

Calculations	Function Points
Change Requests	Reports
Classes	Requirements
Defects	Tables
Features	Use Cases
Files	Web Pages

Use Phase or Iterations

I have found it very helpful to break projects into smaller phases or iterations and then only provide estimations for those subsequent phases that make sense. For example, don't provide an estimate of the length of time it will take to code a report before you have completed requirements gathering. Not everyone sees this as obvious.

Breaking a project into pieces has the additional advantage of splitting a Cone of Uncertainty with a long timeline into a series of smaller Cones of Uncertainty with shorter timelines. And as we have already seen, we want to avoid providing estimates too far into the future. Examples of phases include the following:

- Database Design
- Documentation
- Prototyping
- Requirements
- Testing

Project Evaluation and Review Technique (PERT)

In order to use PERT estimation, create the following estimates for each task:

- Worst Case (W)
- Likely Case (L)
- Best Case (B)

These estimates are then plugged into one of the following formulas to give an Expected Value (EV):

- Optimistic PERT: Expected Value = (2*Best+3*Likely+Worst)/6
- Standard PERT: Expected Value = (Best+4*Likely+Worst)/6
- Pessimistic PERT: Expected Value =(Best+3*Likely+2*Worst)/6

These Expected Values can be shown graphically for a single task (see figure 6).

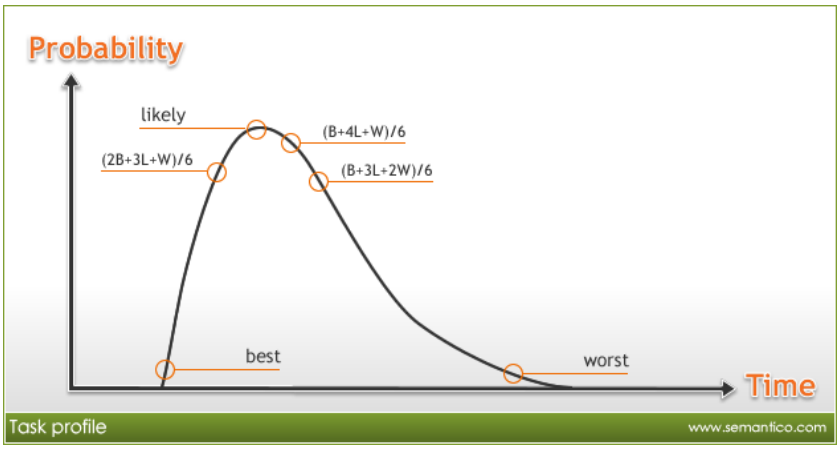

Figure 6 (Padley)

We can also view multiple task curves at the same time (see figure 7).

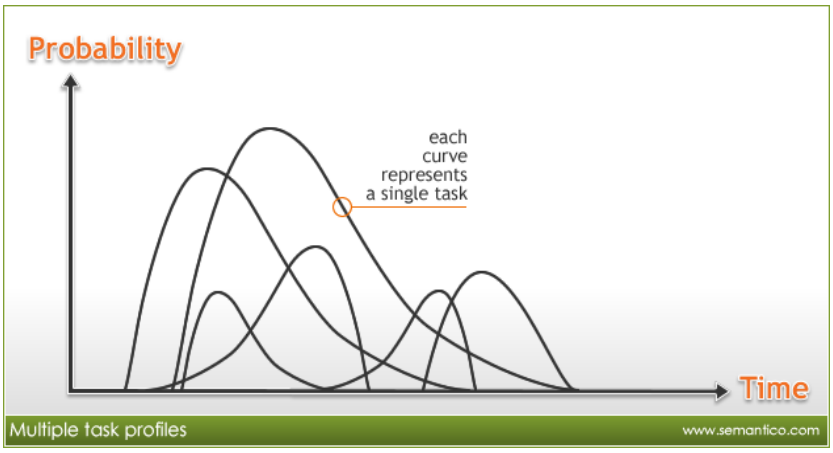

Figure 7 (Padley)

The tasks can also be combined to show the overall curve the entire project (see figure 8).

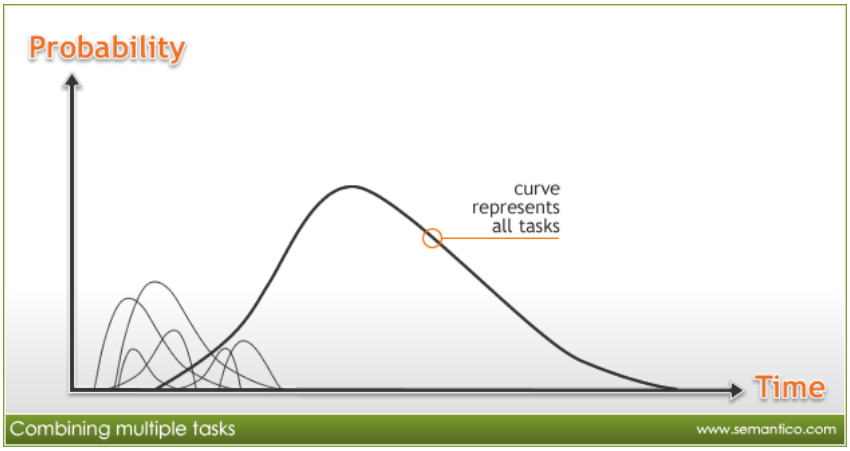

Figure 8 (Padley)

Finally, we can view the combined curve to show our profit or loss (see figure 9).

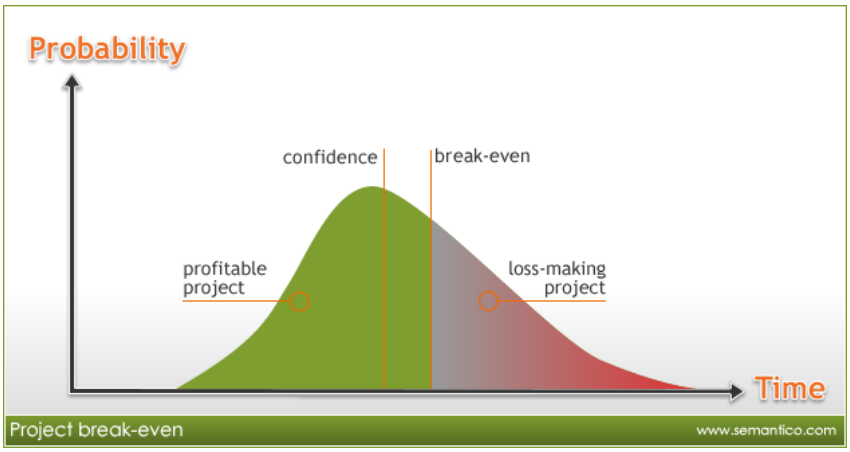

Figure 9 (Padley)

The use of three estimates when using PERT provides the benefits of

- overcoming our natural tendencies to be optimistic,
- forcing us to think about risks (especially while considering worst case estimates)
- and providing a framework for defending our estimates.

Conclusion

The main thing I have tried to do is to encourage you to think critically about the process of estimation. It has not only improved my estimates but has also greatly helped on those occasions when I have needed to defend those estimates. My job has become easier and more satisfying since I started investigating the process of estimation and adopting the techniques discussed in this paper

And read Steve McConnell's *Software Estimation: Demystifying the Black Art*. Your job just may get easier and more satisfying.

References

Padley, Richard. "Software Estimation and the Cone of Uncertainty." *The Discovery Blog: Semantico Looks at Online Publishing*. Semantico, 29 June 2009. Web. 7 Sept. 2011. <http://blogs.semantico.com/discovery-blog/2009/06/software-estimation/>

McConnell, Steve. *Software Estimation: Demystifying the Black Art*. Redmond, WA: Microsoft Press, 2009. Print.

Wikipedia. N.p., n.d. Web. 9 Sept. 2011. <www.wikipedia.org>.

Recommended Reading

McConnell, Steve. *Code Complete: A Practical Handbook of Software Construction*. Redmond, WA: Microsoft Press, 1993. Print.

McConnell, Steve. *Rapid Development: Taming Wild Software Schedules*. Redmond, WA: Microsoft Press, 1996. Print.

McConnell, Steve. *Software Estimation: Demystifying the Black Art*. Redmond, WA: Microsoft Press, 2009. Print.

McConnell, Steve. *Software Project Survival Guide:* . Redmond, WA: Microsoft Press, 1998. Print.

Contact Information

Your comments and questions are valued and encouraged. Contact the author at:

Jack Fuller
Experis Manpower Group
5220 Lovers Lane
Portage, MI 49002
Phone: 1.269.553.5126
Fax: 1.269.553.5100
E-mail: jack.fuller@experis.com