Was the Project Really Late? (An Introduction to Schedule Risk Analysis.)

Paper to be given at the 2012 Construction CPM Conference.

> **"The best laid plans o' mice an' men gang aft a-gley. "** (Robert Burns, 1759 to 1796)

I don't know exactly what that means either, but Burns was drawing attention to the inadequacy of plans. We all hear about how most projects are late, but late compared to what? Compared to the project plan, that's what. But suppose the plan is flawed? Suppose our expectations about project completion were not reasonable?

The planned project completion date is just a forecast. When the weather forecast says it is going to rain and it doesn't, we blame the forecast for being wrong. We don't say the weather was wrong. So, why, when a project fails to be completed on time do we blame the execution of the project rather than the plan?

Today I am going to suggest that the reason so many projects are "late" is that the original project plan was not realistic. This is not a new idea. In "Brookes' Law Repealed", Steve McConnell says:

> **"Late chaotic projects are likely to be much later than the project manager thinks -- project completion isn't three weeks away, it's six months away. ……., but that's not a result of Brooks' Law. It's a result of underestimating the project in the first place."** (Steve McConnell, "Brooks' Law Repealed," IEEE Software, vol. 16, no. 6, Nov/Dec, 1999. Brooks' Law states that putting more resources onto a project when it is running late will make it take even longer.)

And I am going to discuss one particular reason why so many plans are unrealistic. It is called merge bias. But before we get into what merge bias is we need to discuss uncertainty, and two more quotes:

> **"In these matters the only certainty is that nothing is certain."** (Pliny the Elder, 23 to 79 AD)

> **"Prediction is very difficult, especially if it's about the future."** (Nils Bohr, 1885 to 1962)

Project plans are of course about the future, and everything about the future has some uncertainty associated with it. When we say that a task will take 5 days we know that this will probably not turn out to be precisely true. We may have a small amount of uncertainty or a large amount, but we almost certainly have some uncertainty!

As a result of our uncertainty about the task durations, our estimate of project completion is just that, an estimate. It will be valid only to the extent that our estimates of task durations turn out to be true. Since there are generally many tasks, it is almost certain that many of them will take less time than estimated while others will take more time than estimated. So, in reality the project finish is likely not to be as projected by a deterministic CPM or PERT but to be somewhere in a range of dates that we might suppose to be spread symmetrically around that estimate.

But it is actually worse than that. If the above were true, projects would be late only about half the time, and half the time they would be early. Maybe the deterministic estimate is not in the center of the likely range. Maybe it is not even within the range. **In other words, maybe the estimate is biased.**

We are about to see that this is in fact the case for most project networks, and again the reason is merge bias.

Merge bias occurs whenever two or more paths converge in a network and the uncertainty about their durations is such that any of them might turn out to be critical. Consider a task which has just two predecessors which run in parallel. Suppose that each could take anything from 1 to 6 days with equal probability. And suppose further that they will not take fractions of a day. This may sound rather contrived, but I am doing it so we can simulate them with a pair of dice and not get involved in a lot of calculus.

The expected duration of the individual tasks is of course 3.5. But what is the expected time for them **both** to be complete? Let's consider the dice to find out. The following table shows all 36 possible outcomes of throwing two dice.

	1	2	3	4	5	6
1	1	2	3	4	5	6
2	2	2	3	4	5	6
3	3	3	3	4	5	6
4	4	4	4	4	5	6
5	5	5	5	5	5	6
6	6	6	6	6	6	6

Figure 1.

Of the 36 possibilities only one has 1 as the maximum while no less than 11 have 6 as the maximum. So already with just two parallel paths we can see that the time it takes for both of the tasks to be complete is likely to be longer than the deterministic estimate of 3.5 days. If we do the math we find that the expected value is actually almost 4.5 days.

We cannot do this easily for more dimensions, which correspond to more parallel tasks. But I have a spreadsheet in which we can easily calculate the odds for any number of parallel tasks. (This spreadsheet will be made available for download from www.barbecana.com after the conference.)

With 3 parallel tasks, the expected value is almost 5 days, and with 5 tasks it is about 5.4 days. And with 10 it is 5.8 days. Clearly it will converge on 6 days. Another way to look at this is to restate Murphy's law, less pessimistically than Murphy:

If there are *many* things which could go wrong then *at least one of them* will.

Project networks are complex things, tasks interacting with each other in various combinations of ways. Sometimes they are in series, so that their durations merely need to be added, other times they are in parallel so that (as in the above example) we are interested only in the longest path. In general, they are

too complex for us to determine the distribution of the end date analytically as we did with the spreadsheet.

So, we resort to a technique called Monte Carlo simulation. It is a bit like simulating a real life event by throwing dice except that it is done on a computer. Computers can generate what are called pseudo-random numbers, and with these we can generate what look like samples from any distribution we like.

At this point, to introduce the idea of simulation, I will show a simulation of the above dice-throwing example with a program I wrote for the purpose. It demonstrates that over the long haul, say 10,000 trials, the results of simulation are virtually indistinguishable from the theoretical result, and also that it is very fast. (This program will be made available for download from www.barbecana.com after the conference. Figure 2 on the following page illustrates what it looks like.)

But simulation can be applied to the most complex situations, where a theoretical analysis would be intractable. In the context of a project network, Monte Carlo simulation:

1. takes a sample from the user-defined distribution for each task duration;
2. does a time analysis based on these sampled durations;
3. stores the results of this time analysis (maybe in a summary form like a histogram); and
4. repeats steps 1 through 3 several hundred or thousand times.

At the end of this process it produces histograms representing the probability distributions of any calculated result of interest, which generally includes:

1. early and late start and finish dates for all tasks;
2. free and total floats for all tasks; and
3. costs for all tasks.

For the remainder of this presentation, we will be using Full Monte ™, Barbecana's schedule risk analysis add-in for Microsoft Project. (This too can be downloaded from www.barbecana.com for a free 30-day trial.)

I have prepared a network with 6 tasks in parallel, all with an expected duration of 3.5 weeks and a range of 1 to 6 weeks. Right now, only one of the parallel tasks is hooked up to the successor task B. MSP says the project will finish on February 29th.

I do a simulation and guess what, we get the same answer. This is because with just one of the parallel tasks hooked up to the successor, the others are never critical. There is no merge bias.

I will now hook up the others, one at a time, doing a simulation each time. The deterministic MSP estimate does not change, but the Full Monte estimate does! With just two hooked up, the finish date has moved on 6 days.

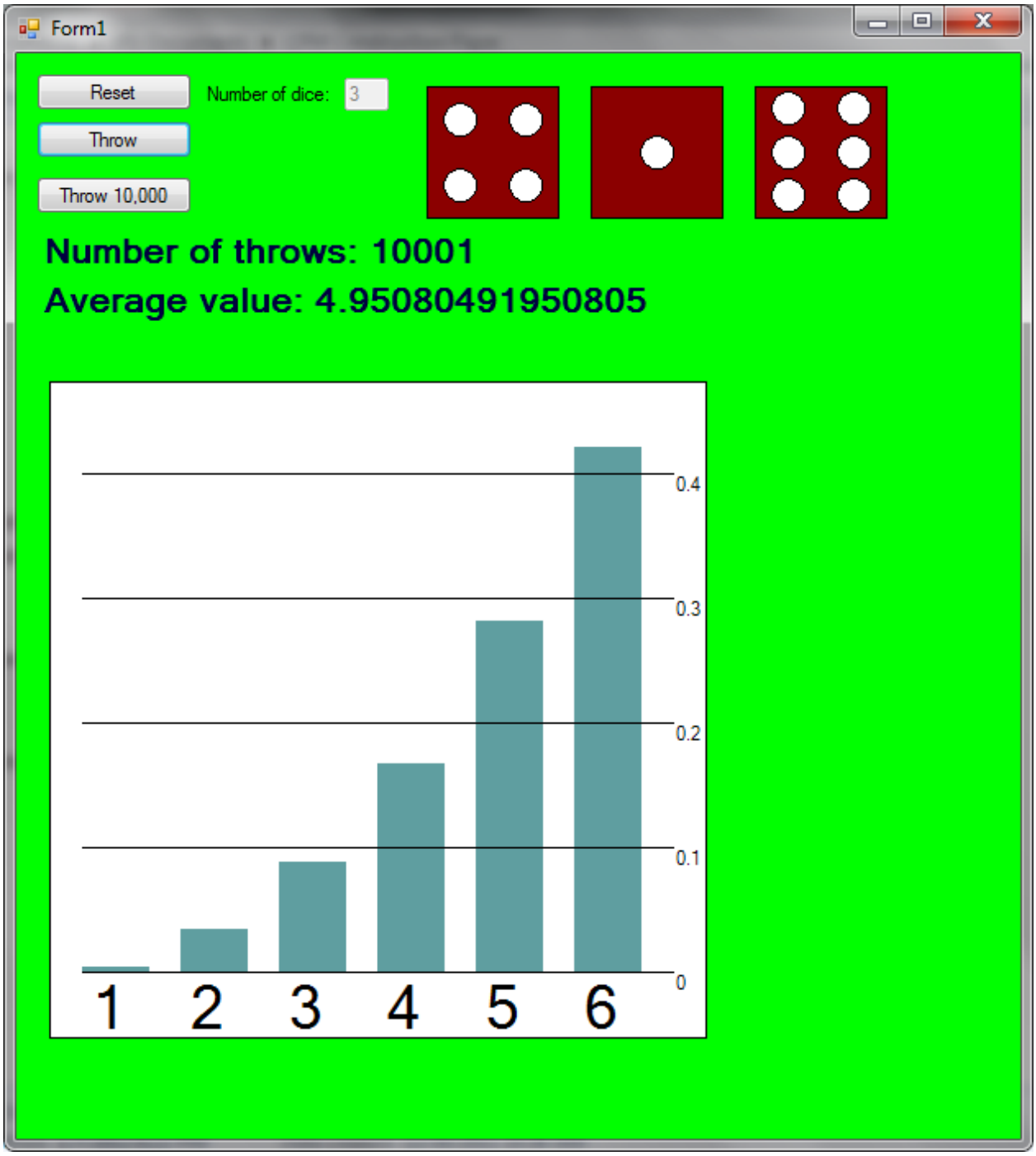

Form1
Reset
Number of dice:
3
Throw
Throw 10,000
Number of throws: 10001
Average value: 4.95080491950805
0.4
0.3
0.2
0.1
0
1
2
3
4
5
6

Figure 2.

With all 6 tasks hooked up, Figure 3 shows the histogram of project finish date.

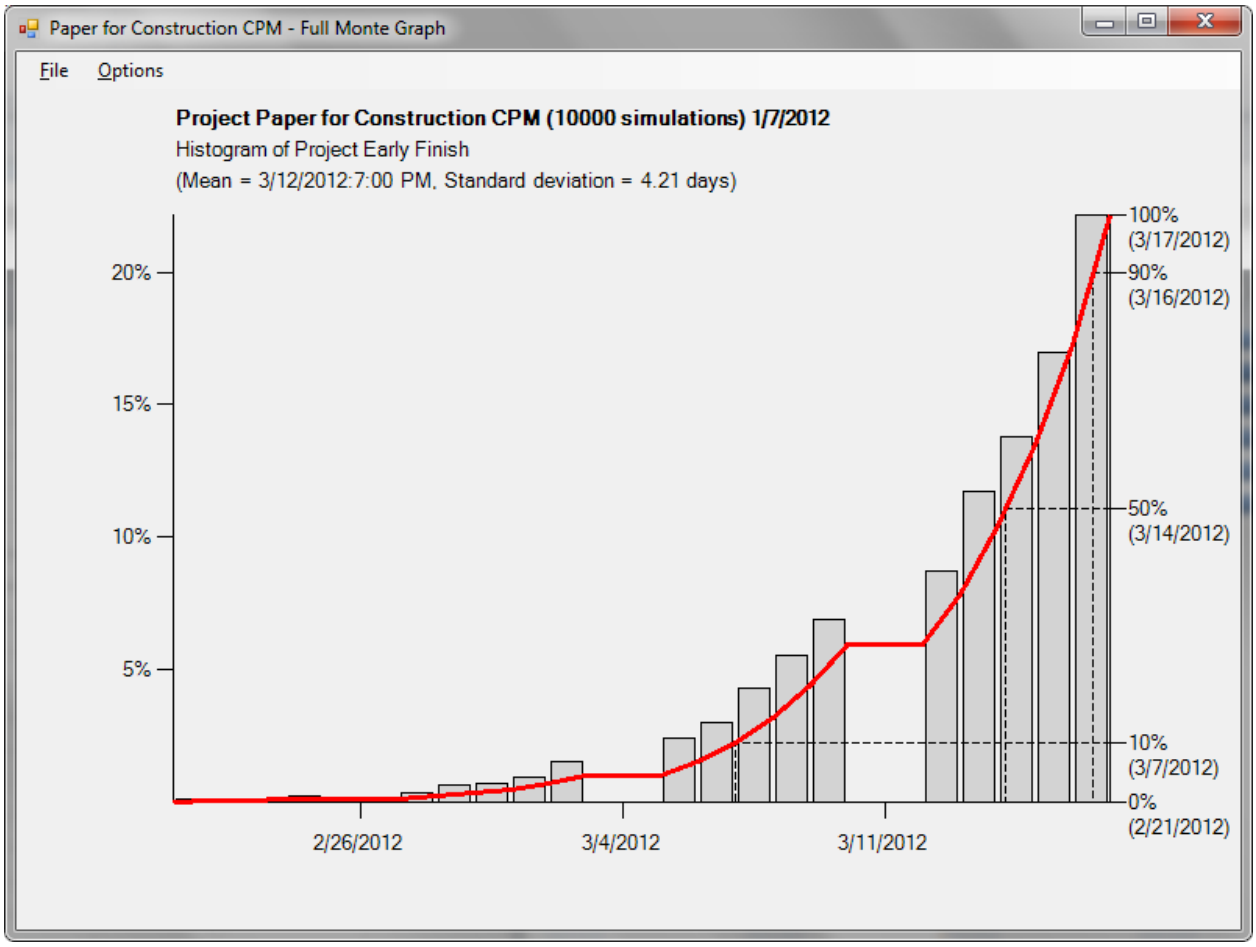


Figure 3.

The gaps are due to weekends, by the way. The mean finish date is March 12th, 12 days later than the deterministic projection; that's over 2 weeks in a 10-week project. The red S-curve shows the cumulative probability and the chance of finishing on or before the deterministic estimate is about 2%. **This is merge bias.**

Yet if we look at the end distribution of the end dates of individual tasks, shown in figure 4, we see as we would expect that these are uniform.

We might also look at the slack on one of the tasks, as shown in figure 5. About 20% of the time the slack is less than half a day. It is zero of course 1/6th of the time, which we can see by double clicking on any row to get the criticality index. (Not shown here.)

Now, the uniform distribution is the best one to illustrate merge bias, which is less pronounced if the original distributions are normal for example. In our previous example, the effect would be only about 7 days instead of 12. But our example is also very small, and merge bias compounds over multiple merges in a typical project.

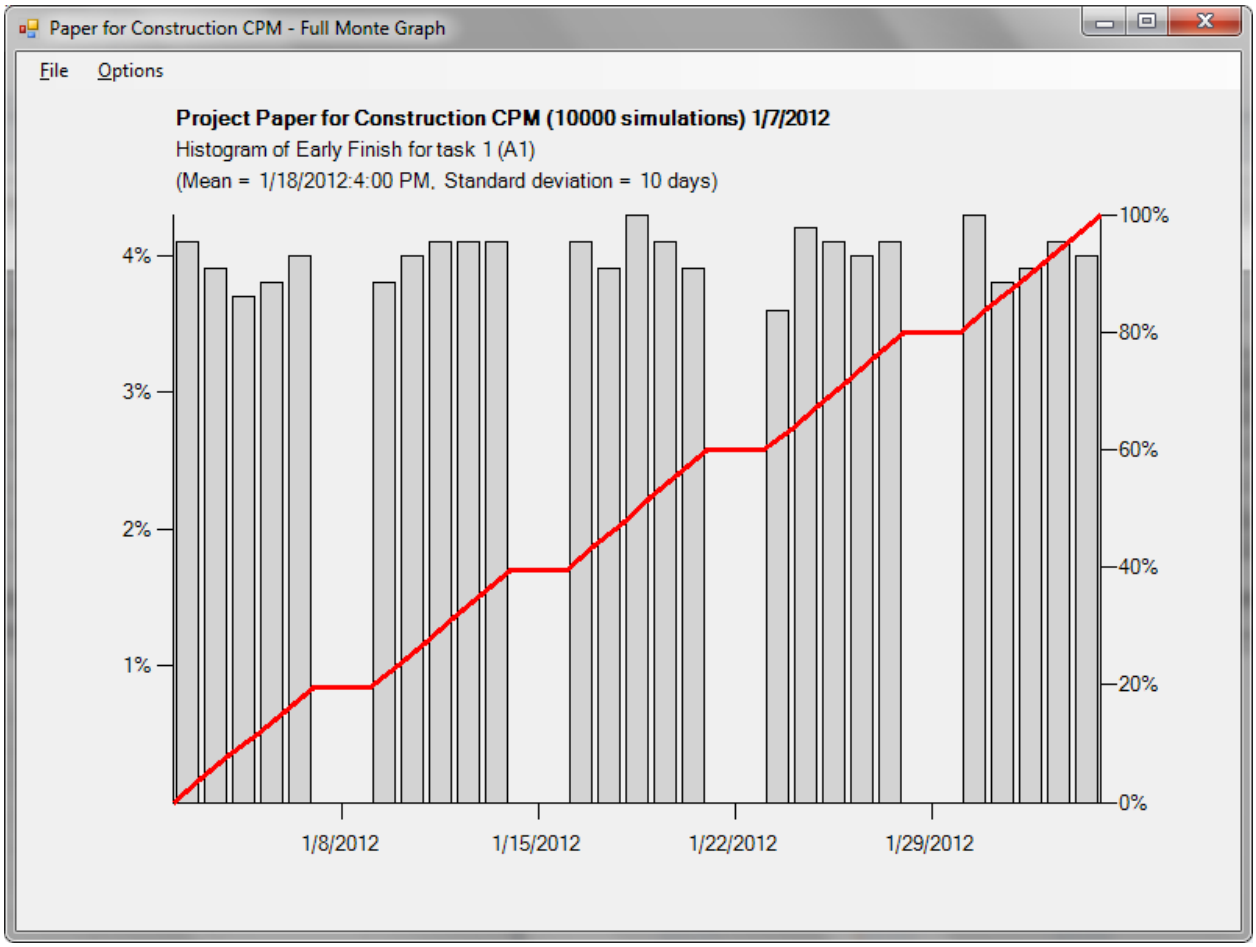


Figure 4.

Real project networks are of course more complicated than our small example, but they almost always have points at which paths merge, and each one of these points can introduce bias. This bias is always in the same direction – to make the deterministic estimate optimistic – and there are no countervailing features which can have the opposite effect. As a result, deterministic estimates of project finish dates are always too optimistic. This is especially true as the project progresses, because merge bias builds up over time. So, early milestones may be met while later ones become progressively worse**.**

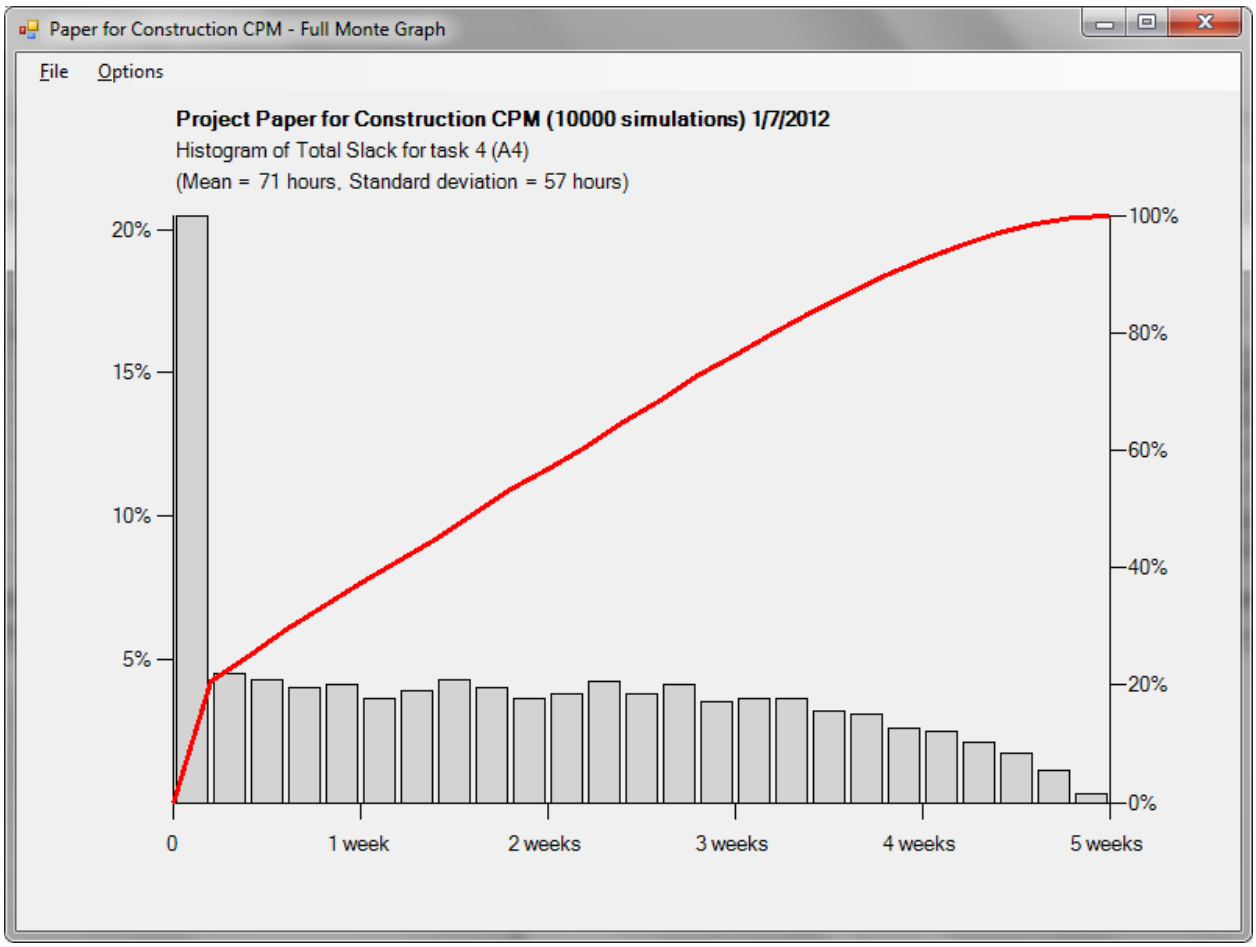
I will conclude with a couple of observations about the effect of uncertainty on resource requirements.

1. First of all, if the parallel predecessors all use the same resource they may have to be done in series anyway, in which case the merge bias goes away, but this is not generally the case.

2. The second point results from the fact that total resource usage is not affected by merge bias, so the deterministic value of the resource usage is an unbiased estimate. However, merge bias can have the effect of spreading this total out over a longer time period, so the peak resource usage may be lower than expected. The result is that more resources may be applied to the project than it actually needs. So one can have the apparently paradoxical situation in which the

project is running late but resources are standing idle. I have not observed this in practice, but I would be very interested to hear from anyone who has.

My final conclusion is that project managers should stop beating themselves up for being "late" and educate their stakeholders about merge bias, and about the value of Monte Carlo simulation in setting more realistic expectations.



.Figure 5.