BLOG @CACM

The *Communications* Web site, <http://cacm.acm.org>, features more than a dozen bloggers in the BLOG@CACM community. In each issue of *Communications*, we'll publish selected posts or excerpts.

Follow us on Twitter at <http://twitter.com/blogCACM>

DOI:10.1145/3371385

<http://cacm.acm.org/blogs/blog-cacm>

In Search of the Shortest Possible Schedule

Bertrand Meyer considers how to speed up software engineering.

Bertrand Meyer
The Shortest Possible Schedule Theorem: Yes, You Can Throw Money at Software Deadlines
<http://bit.ly/36tYxcX>

October 27, 2019

Some folk wisdom going around in software engineering, often repeated for decades, is just wrong. It can be particularly damaging when it affects key aspects of software development and is contradicted by solid scientific evidence. The present discussion covers a question that meets both of these conditions: whether it makes sense to add staff to a project to shorten its delivery time.

My aim is to popularize a result that is well known in the software engineering literature, going back to the early work of Barry Boehm,¹ and explained with great clarity by Steve McConnell in his 2006 book on software cost estimation *Shortest Possible Schedule*.² While an empirical rather than a logical result, I believe it deserves to be called a theorem (McConnell stays shy of using the term) because it is as close as we have in the area of software engineering management to a universal property, confirmed

by numerous experimental studies.

This article contributes no new concept since McConnell's Chapter 20 says all there is to say about the topic; my aim is simply to make the Shortest Possible Schedule Theorem better known, in particular to practitioners.

The myth about shortening project times begins with an observation clearly correct. Everyone understands if our project has been evaluated through accepted cost-estimation techniques, to require three developers over a year, we cannot hire 36 people to complete it in a month. Productivity does not always scale up.

Yet neither does common sense. Too often the conclusion from the preceding trivial observation takes the form of an old saw, "Brooks' Law": adding people to a late project delays it further. The explanation is that newcomers cost more through communication overhead than they bring through actual contributions. While a few other sayings of Brooks' *Mythical Man-Month* have stood the test of time, this one has always struck me as describing, rather than any actual law, a definition of bad management. Of course if you keep haplessly throwing people at deadlines, you are going to add

communication problems and make things worse. But if you are a competent manager, expanding team size is one of the tools at your disposal to improve the state of a project, and it would be foolish to deprive yourself of it. A definitive refutation of the supposed law, also by McConnell, was published 20 years ago.³

For all the criticism it deserves, Brooks' pronouncement was limited in its scope: it addressed adding staff to a project already late. It is even more wrong to apply it to the more general issue of cost-estimating and staffing software projects at any stage of their progress. Forty-year-old platitudes have even less weight here. As McConnell's book shows, cost estimation is no longer a black art. It is not an exact science either, but techniques exist to produce solid estimates.

The Shortest Possible Schedule theorem is one of the most interesting results. It is much more interesting than Brooks's purported law, because it is backed by empirical studies (rather than asking us to believe a pithy pronouncement), and instead of a general negative view, it provides a positive result complemented by a limitation of that result, and both are expressed quantitatively.

Figure 1 gives the general idea of the SPS theorem. Figure 2 provides a more precise view.

The "nominal project" is the result of a cost and schedule estimation yielding the optimum point. The figure and the theorem give project managers both a reason to rejoice and a reason to despair:

► Rejoice: by putting in more money, that is, more people (in software engi-

neering, project costs are people costs),^a you can bring code to fruition faster.

► Despair: there is a firm limit to the time you can gain: 25%. It seems a universal constant of software engineering.

The “despair” typically gets the most attention at first, since it sets an absolute value on how much money can buy (so to speak) in software: try as hard as you like, you will never get below 75% of the nominal (optimal) value. The “impossible zone” in Figure 1 expresses the fundamental limitation. This negative result is the reasoned, precise modern replacement for the older folk “law.”

The positive part is just as important. A 75%-empty glass is also 25% full. It may be disappointing for a project manager to realize no amount of extra manpower will make it possible to guarantee more than a 25% reduction in time. But it is just as important to know such a reduction, not at all insignificant, is reachable given the right funding, people, tools, and management skills. The last point is critical: money by itself does not suffice, you need management; Brooks’ Law, as noted, is mostly an observation of the effects of bad management.

Figure 1 only carries the essential idea, and is not meant to provide precise numerical values. Figure 2, the original figure from McConnell’s book, plots effort against time rather than the reverse but, more importantly, it shows several curves, each corresponding to a published empirical study or cost model surveyed by the book.

On the left of the nominal point, the curves show how, according to each study, increased cost leads to decreased time. They differ on the details: how much the project needs to spend, and which maximal reduction it can achieve. They all agree on the basic Shortest Possible Schedule result: spending can decrease time, and the maximal reduction will not exceed 25%.

The figure also provides an answer, although a disappointing one, to another question. So far this discussion has assumed time was the critical resource and we were prepared to spend more to get a product out sooner. But sometimes it is the other way around: the critical resource is cost or, concretely, the number of devel-

Figure 1. General view of the Shortest Possible Schedule theorem.

Figure 2. Original illustration of the Shortest Possible Schedule (reproduced with the author’s permission)

Source: Adapted and extended from Software Sizing and Estimating: MK II (Symons 1991), Software Cost Estimation with Cocomo II (Boehm et al 2000), “Estimating Web Development Costs: There Are Differences” (Reifer 2002), and Practical Project Estimation, 2nd Edition (ISBSG 2005).

opers. Assume nominal analysis tells us the project will take four developers a year and, correspondingly, cost 600K (choose your currency). We have a budget of 400K. Can we spend less by hiring fewer developers, accepting it will take longer?

On that side, right of the nominal point in Figure 2, McConnell’s survey of surveys shows no consensus. Some studies and models do lead to decreased costs, others suggest that with the increase in time, the cost will increase, too.

(Here is my interpretation, based on my experience rather than on any systematic study: you can achieve the original goal with a somewhat smaller team over a longer period, but the effect on the final cost can vary. If the new time is $t' = t + T$ and the new team size $s' = s - S$, t and s being the nominal values, the cost difference is proportional to $Ts - t'S$. It can be positive as well as negative, de-

pending on the values of the original t and s and the precise effect of reduced team size on project duration.)

The firm result, however, is the left part of the figure. The Shortest Possible Schedule theorem confirms what good project managers know: you can, within limits, shorten delivery times by bringing all hands on deck. The precise version deserves to be widely known. ■

References

1. Boehm, B.W. *Software Engineering Economics*, Prentice Hall, 1981.
2. McConnell, S. *Software Estimation Demystifying the Black Art*, Microsoft Press, 2006.
3. McConnell, S.: Brooks’ Law Repealed, in *IEEE Software*, vol. 16, no. 6, pp. 6–8, November–December 1999.

Bertrand Meyer is a professor of software engineering (emeritus) at ETH Zurich (Switzerland), chief technology officer of Eiffel Software (Goleta, CA, USA), professor at Politecnico di Milano (Italy), and head of the software engineering lab at Innopolis University (Russia).

^a This is the accepted view, even though one might wish the industry paid more attention to investment in tools in addition to people.