



All Late Projects Are the Same

Tom DeMarco



BY THE TIME this article appears, I will have started my 50th year working in information technology. At the beginning of my career, we all thought that the key advances of the next 50 years would be in rocketry. But now we know otherwise: it is computers and software that have changed the world in ways that none of us ever considered.

What's the Matter with You Software People?

I began my career as a circuit designer at Bell Laboratories. Early in the 1960s, I was switched, along with many of my peers, from hardware to software simply because the hardware on our project was finished long before the software. This seemed surprising at the time: How could it be that software was harder than hardware? It took me a while to figure it out, but everything we were doing had as its unstated goal to move the hard stuff out of the hardware and into the software. Before too long, all the complexity was in the software. I tried in vain to explain this to various project managers, who complained, "The hardware guys never give me any trouble—what's the matter with you software people?"

"What's the matter with you software people?" was a major theme of most of the rest of the 20th century. In spite of our astonishing and transformational successes, we obsessed over our failures—in fact, the literature of the period is full of failure stories. You never would have guessed from all the glum retrospectives that the very software people who were being treated as village idiots were in the process of enabling the global economy, connecting people and companies across the

world and far above it, and remaking the nature of virtually every business on Earth.

By the 1990s, a significant part of my practice was litigation support, which was a natural consequence of raising my rates to the level that only legal departments could afford. Of course, litigation is all about failure, so perhaps I was seeing more than my share of it. Surprisingly, the failures began to look pretty much alike. Company A contracts to build a software system for Company B and is late to finish, or it goes on beyond its contracted delivery date and the work is cancelled. B sues A or vice versa, one of them hires me, and we all obsess over failure for a while and then settle. In the end, A and B are poorer, the lawyers and I are slightly richer, and nothing has changed.

In poring over nearly a billion dollars worth of software litigation, I came across no failures due to poor quality, slow response time, or unworkable human interface; all the failures were about lateness. Although the question "What's the matter with you software people?" sounds complicated, the answer was surprisingly simple: we're occasionally late.

Get Ready for Astounding Insight

About this time, I began telling anyone who would listen that all late projects are the same. I think I was right about this, although my explanation at the time was flawed. I thought all late projects were the same in

continued on p. 103

continued from p. 104

that they were really estimation failures, not performance failures. This was cute but not very accurate at a deep level because so many projects don't really do any estimating at all. Rather, they propose a goal and then get someone to espouse it as an estimate. Delivery by January of next year? Sure, why not? This sounds dumb, but so many fine software products have been built after such a start that I'm tired of railing about the necessity of early accurate estimation.

I still believe that all late projects are the same, but for an entirely different reason. When I tell you the reason, you'll think I'm stating such an obvious idea that it barely qualifies as an idea at all. But bear with me...

All projects that finish late have this one thing in common: they started late.

Is that deep or what? A project that took two years to finish and needed to be done by 31 December 2010 should have been started on 1 January 2009. It wasn't—it started in early 2010, so it finished late and might have been judged a failure. If it seems pointless of me to suggest that the project should have started earlier, consider the reasons why it didn't. I can think of three:

1. Nobody had the guts to kick off the project until the competition proved it doable and desirable; by then, the project was in catch-up mode and had to be finished lickety-split.
2. If the project were started long enough before its due date to finish on time, all involved would have had to face up to the fact from the beginning that it was going to cost a lot more than anyone was willing to pay.
3. No one knew that the project needed to be done until the window of opportunity was already closing.



NEW DEPARTMENT

Sounding Board is a new feature in *IEEE Software* that will highlight short opinion pieces from valuable members of our community: new ideas, interesting challenges, controversial views, different viewpoints. If you've heard of something or someone that we should invite to be featured here, contact me at kruchten@ieee.org. These pieces will also be available on the magazine's website (www.computer.org/software) for your comments, rebuttals, or disagreements.


—Philippe Kruchten, *Sounding Board* editor

The “window of opportunity” concept explains why Google had to be the very first to build a search engine, otherwise its competitors would have gobbled up all the business. Wait a minute—Google didn't build the first search engine, you say? It was 15 years late coming to the party? I suspect the window of opportunity argument is nearly always a sham, and reason three on my list is really a disguised instance of reason one or two.

Reason one—blindsided by the competition—is a legitimate business failure. Interestingly, it's not software developer failure that's in question here, but that of some marketing arm that got one-upped by superior marketers in another company. Making a lot of noise about those software folks who failed to build the catch-up product fast enough is just a way to deflect attention from what really happened and who is responsible.

This leaves us with projects that started late because they didn't offer enough value to justify their true cost. This is garden variety failure, in my opinion: it happens all the time. If a project offered a value of 10 times its estimated cost, no one would care if the actual cost to get it done were double the estimate. On the other hand, if expected value were only 10 percent greater than expected cost, lateness would be a disaster. Yes it would be a disaster, but instead of obsessing over

“What's the matter with those software folks who didn't deliver on the schedule we gave them?” we need to ask instead, “Why did we ever kick off a project with such marginal expected value?”

The louder the complaints about project lateness, the more likely it is that the project set out to deliver marginal value and was therefore kicked off under the false premise that it could be completed on the cheap. What's really wrong with us software folks is that we're continually beating ourselves up for something that's somebody else's fault. 

TOM DEMARCO is a principal of The Atlantic Systems Guild and the author of numerous books about system building and the people who do it. Contact him at tdemarco@systemsguild.com.

**IEEE
Software**

NEXT ISSUE:

**Algorithms
and Today's
Practitioner**