WHY DOES SOFTWARE COST SO MUCH?

In the absence of meaningful standards, a new industry like software comes to depend instead of *folklore.* As the industry matures, a first order of business is to recognize and question the folklore. I propose to do this by considering that familiar question: "Why does software cost so much? How we ask that question and how we answer it reveal much about our folklore.

WHEN DID YOU STOP BEATING YOUR WIFE?

Author and consultant Jerry Weinberg [1] claims to have encountered the question "Why does software cost so much?" more than any other in his long career. The correct answer, he says, is "Compared to what?" There is a likable logic to that: Most of the things we do with software in the 1990s are barely conceivable without software, so there is no valid basis of comparison. Yet Jerry's answer, charming as it may be, won't do you much good. At best, it will just annoy your questioner. No answer is going to satisfy the questioner (typically your boss or user), because he or she *is not really interested in a answer.* People don't ask that question to have an answer.

"Why does software cost so much?" is not a question at all; it's an assertion. The assertion is that software is too pricey. The person who poses this rhetorical question may seem to be motivated by mere intellectual curiosity: "Gee, I've always wondered, just why software cost so much?" The real motivation, however, has nothing to do with curiosity. It has only to do with getting the brutal assertion on the table. It's a negotiating position. You are being put on notice that software costs unconscionably much to build, and no budget or schedule you're likely to request will be considered reasonable. Your boss or user may agree to your budget, but only under *extreme* duress. Since the amount budgeted is already terribly, terribly excessive, any slip or overrun is virtually a crime against nature.

In a recent interview [2], Cadre Technologies founder Lou Mazzucchelli observes that "software consumers are not satisfied with either the quantity or the quality of our output." Right on target. Software consumers in vast numbers are telling us that our efforts just don't begin to measure up to their expectations. Software, at least the kind that companies build for their own use, is much too expensive, takes too long to build, isn't robust enough, and isn't easy enough to use or good enough in any other way either.

Now I have a very grumpy question for those who complain that the software development community hasn't measured up to their expectations: Where in hell did those expectations come from?

WE INTERRUPT THIS SOBER TREATISE FOR A DIATRIBE

You and I and other like us built the software industry from scratch over the last thirty years. We started with thin air and made it into a $300 billion a year business. (See [3] to understand how I arrive at this figure.) In all of economic history, there has never been a more staggering accomplishment. Think of that. In the time it takes for you to read this article, the software industry will have generated something more than $12 million.

What has it taken to build this huge new industry so quickly? Hint: It wasn't just getting some programmers together and teaching them to sling code. It required the active participation of a marketplace. Somebody had to toady up huge quantities for money to buy all the software we built.

And they did. Not only did a lot of somebodies buy all we could produce at the cost we charged, they complained about not being able to buy even more, about the so-called software backlog.

All of this growth was not the result of poor quality and poor productivity. The only conceivable explanation for the phenomenal success of the software industry is that it has regularly delivered a standard of quality and productivity far beyond the real expectations of the market. But all the time that our buyers (our managers and our users) were lining up to cash in on the bargain, they were complaining. Their actions and their words gave two diametrically opposed messages.

Before I comment on why this might be, it's interesting to observe that this isn't a recent phenomenon. They didn't congratulate us for years and years and then only become upset at the downturn in the 1990's. No, they complained all the way from zero to $300 billion a year.

I myself am a bit peeved by this. (Perhaps you could tell.) I feel like we have accomplished wonders and been yelled at the whole time.

AN ANALOGY FROM 1904

Imagine how the Wright brothers would have felt had they had a similar reception. Imagine you're Orville, for example. It's December 7, 1904, at Kitty Hawk, North Carolina, 7:30 A.M., and you're clambering into Flyer One. "Let 'er rip," you say, or words to that effect. The engine coughs to life. You rev it up and there's movement. There is not just movement, there is speed, speed and bumps and wind and …by God, you're up! You're off the ground. This is it: heavier-than-air flight! You've done it. You've pulled off a miracle, and the world will never be the same. You're so elated you're barely even afraid. Like a practiced pilot, you bring the flyer down.

But just as you're coming to a stop, you notice a guy in a business suit, holding an attaché case and looking sourly at his watch. He approaches and says, "Orville, I'm really disappointed in this project. I had great expectations of you and you've let me down. Here it is, nearly 8 A.M., and I have to be in L.A. for a dinner meeting tonight. And you guys are nowhere! You haven't invented the jet engine yet or the stewardess or the airport or airline meals or those little drinks in miniature bottles. You've let me down completely! I'm going to miss my meeting and it's all your fault."

That guy in the suit is the software consumer whom Lou Mazzuchelli was talking about, the user or manager who is "not satisfied with either the quantity or the quality of our output."

NOW THE TRUTH ABOUT WHERE THOSE EXPECTATIONS COME FROM

No one ever expected the software industry to achieve what it has achieved. Not a single futurist predicted the extraordinary productivity and quality we have accomplished. Then how is it possible that the industry as a whole is performing beyond the wildest expectations, while all the individual projects are underperforming outrageously? It's not possible. Those projects aren't underperforming at all, and their consumers know it. Pay attention to what they do, not what they say, to get the real message. The real message is that software consumers are telling us our software is the best bargain they ever heard of.

So why all the bellyaching? People complain to us *because they know we work harder when they complain.* We have trained them to do this. When they complained in the past, we worked harder. We gave them more for their money (even more than the extraordinary bargain they would have gotten anyway) because they pretended to be discontented. Boy, are we dumb.

A HELPFUL HINT FROM THE ACADEMIC COMMUNITY

In a recent article [4], authors Professors Lederer and Prasad set down some guidelines for better software cost estimating and perhaps unintentionally provided some insight about managerial complains that software costs too much. Their guidelines were delivered from a survey on estimating techniques mailed to a sample of some four hundred professional software managers. What seemed more important to me than any of the article's conclusions was a tiny "factoid" tucked away in their commentary about the survey responses: The great majority of respondents reported that their software estimates were dismal ("only about one in four projects is completed at a cost reasonably close to the estimate"), but they weren't on the whole dissatisfied with the estimating process. Lederer and Prasad report that

> *forty-three percent of the subjects indicated that their current estimating was "very satisfactory" or "moderately satisfactory" (the two highest ratings).*

What's going on here? Software estimates bear little or no resemblance to reality, but the managers aren't dissatisfied? How can that be? Oh…I think I'm beginning to understand. Maybe the purpose of the estimating process is not to come up with a realistic answer, but to come up with an *unrealistic* answer. Maybe the estimating process is not supposed to guide them as to what to *pretend* to expect. This is the kind of estimating process that tells your boss, for example, to set September 1995 as the "right" expected delivery date for your new project. It provides a schedule that is neither ridiculous nor feasible. That is, after all, the object of the exercise. It is the very definition of the "right" schedule:

The right schedule is one that is *utterly* impossible, just not *obviously* impossible.

HOW DO MANAGERS KNOW?

It is a great tribute to qualify software management that managers know unerringly how to set this right schedule. This task isn't easy. In the abstract, at least, it is just as hard to predict a date that is just short of possible as to pick one that is safe and reasonable. Both require some prediction of what the true achievement capability of a team will be, and that is a difficult matter.

How do our managers learn to do this hard thing? Again, the answer is we've trained them. They watch our faces when they set schedules. If we look relieved, they know they haven't turned the screws enough. If we just giggle at them, they know they've gone too far.

SUCCESS IN ESTIMATING AND FAILURE IN BUSINESS

"Why does software cost so much?" is an assertion masquerading as a question. The assertion that software is too expensive is part of a cost-containment ploy. The cynical notion that a good schedule is one that no one has a prayer of achieving is another part of that ploy. The constant refrain that software developers just are not productive enough has only one purpose: to make software developers work harder. It is a goad, a goad that appears to work because software developers are sincere and professional and a little dopey.

The problem is our industry is over-goaded. The work of software development is largely incompressible. When you're under pressure, your first response is to cut out extraneous activities: chats and bull sessions. That may indeed be productive, but as the pressure continues, there is nothing more you can do. You can't work faster; you might stay later, but that has a long-term cost

in your personal life. A little overtime in the next week may help to achieve Friday's deadline, but overtime applied over months and months gives an illusion of progress. The apparent progress of overtime is wiped out by compensatory "under-time," burnout, disillusionment, waste, and employee turnover.

As the pressure increases, the only real option is to pay for speed by reducing quality. I sometimes think, rather bitterly, that reduced quality is a conscious goal of those who pressure projects. They are saying, "Loosen up, folks. Learn to rush the product out the door without worrying so much about quality." This comes, of course, at the very moment that companies are paying lip service to qualify as never before.

In the short term, paying for speed by reducing quality seems to make sense. It wins us a few battles, but never the war. In the long run, that compromise will take us just where it took the American automobile industry in the 1980s: to reduced prominence in the market and reduced capacity to compete with new global players.

THE MORAL

If you ask the wrong question, you'll never get the right answer. Instead of asking "Why does software cost so much?" we need to begin asking "What have we done to make it possible for today's software to cost so little?" The answer to that question will help us continue the extraordinary level of achievement that has always distinguished the software industry.

REFERENCES

[1] Gerald M. Weinberg, *Quality Software Management,* Vol. 1: *Systems Thinking* (New York: Dorset House Publishing, 1992).

[2] "Lou Mazzucchelli on Software Engineering," *Computer Design*, August 1991, pp. 25-27.

[3] John, E. Hopcroft and Dean B. Krafft, "Toward Better Computer Science," *IEEE Spectrum*, Vol. 24, No. 12 (December 1987), pp. 58-60.

[4] A.L. Lederer and J. Prasad, "Nine Management Guidelines for Better Cost Estimating," Communications of the ACM, Vol. 35, No. 2 (February 1992), pp. 51-59.