Forecasting the Future

Welcome to the June issue of CROSSTALK – and I hope you didn't miss the Systems and Software Technology Conference (SSTC) conference in Las Vegas about a month ago. It was great! First of all, the location was superb – conference facilities, hotel, location. Come on – Barry Manilow *and* the Star Trek Experience? Geek heaven. The weather was fantastic (although a bit warm during the day), and having the monorail to travel to/from the Strip was convenient.

The exhibits were also good (as usual). There seemed to be quite a few of the *process-oriented* vendors this year – a good thing, if you ask me. And, let's face it, the giveaways and gadgets were spectacular. The food was also great. Personally, I think that the speakers were better than ever this year. It was good seeing a lot of old friends, and making new ones. The only drawback was that with all of the distractions, far too many of us stayed up late, and made the 2008 SSTC conference a conference to remember. What more could you ask for?

Except that I am writing this column in March, and the SSTC conference is still a month in the future. However, I am totally convinced that almost everything I wrote above *will* be true, and that after the conference ends, I *will* be able to argue that I was *very* successful in predicting the future. If only software were so easy.

The theme of this issue is Software Quality. I have a very unique definition of quality. In my mind, quality was defined by Simon and Garfunkel back in 1970 on their "Bridge Over Troubled Water" album (arguably the best piece of music *ever* released). There was a song entitled "Keep the Customer Satisfied." Awesome lyrics. And that is the key – keeping the customer satisfied.

So, how do I achieve this dubious thing called quality which is hard to measure directly? Well, I am sure that this issue contains great articles about quality (since I'm writing in the future, I don't even know the article lineup yet!).

In my mind, quality needs to include customer satisfaction. Is the software going to be used in a passenger aircraft? Well then, as a potential customer, I am pretty darn hard to satisfy. The other day, I was flying "across the little pond" (returning from London across the Atlantic) and the in-seat entertainment system I was using crashed – I actually got a *core dump* error message and saw Linux rebooting. The lady sitting next to me had it happen to her, and her comment was, "I sure hope that the software that runs the aircraft works better." Well, having worked with several aircraft software developers, I can assure you that it *does* run much better. Is the software going to be used to simply rip a few of my old CDs to MP3s so that I can load them onto my latest gadget? Then I am willing to have it occasionally fail. I have a feeling that the latest SuperX MP3 Ripper program I downloaded free off the Web cost a *lot* less per line of code to develop than the software that will power the Joint Strike Fighter.

Which brings me back to forecasting the future. Forecasting the future is not an exact science. A friend of mine who is a meteorologist says that an 85 percent to 90 percent success rate in intermediate range predictions (one to three days out) is great. Two weeks out? More general predictions (i.e., *warming trend*) give the forecaster some leeway. One day out? A high of 73 with afternoon showers, ending by 9 p.m. tonight.

The secret to quality is the same: if you *really* think you can set a schedule (such as 28 lines of code per programmer per day) that will allow your developers to achieve a quality target ("No more than two errors found per 500 function points during integration testing") that is also reasonably accurate one year out, well ... how did you enjoy SSTC 2009? Even with *lots* of historic data for similar projects, each development effort is different. Weather forecasters have access to about a hundred years of hurricane data, but still cannot tell me one month out when and where (or even if) a hurricane is going to hit the United States.

Quality is fragile. It is hard to achieve and, once lost, it seems to be gone forever. You can't test quality back in – those who have tried know better. You have to plan aggressively for quality and you have to have a good process for it (I wasn't kidding earlier – the more process-oriented vendors at SSTC, the better it is for Department of Defense software in general).

Forecast the future as best you can. Revise your forecasts (and timelines) as you get closer to your goals. A good weather person has no problem saying, "Well, last week we said generally clear, but prepare for a heavy rain tomorrow." A good program manager might just have to say, "We had hoped to be in integration testing this week, but we need another month to complete inspections and peer reviews." Nobody *wants* to hear that their vacation and beach plans are going to be washed away, but it happens. Nobody *wants* to hear that we are having problems with code quality, but if you have a good forecast and update it as needed, maybe you won't.

Hope you enjoyed SSTC 2008 – and see you at SSTC 2009. Trust me – it was great, also!

—David A. Cook
The AEgis Technologies Group, Inc.
dcook@aegistg.com

Can You BACKTALK?

Here is your chance to make your point, even if it is a bit tongue-in-cheek, without your boss censoring your writing. In addition to accepting articles that relate to software engineering for publication in CROSSTALK, we also accept articles for the BACKTALK column. BACKTALK articles should provide a concise, clever, humorous, and insightful perspective on the software engineering profession or industry or a portion of it. Your BACKTALK article should be entertaining and clever or original in concept, design, or delivery. The length should not exceed 750 words.

For a complete author's packet detailing how to submit your BACKTALK article, visit our Web site at <www.stsc.hill.af.mil>.