More Ugly Truths about Consulting Assignments

I enjoyed reading Robert Glass's column "How Not to Prepare for a Consulting Assignment, and Other Ugly Consultancy Truths" ("Practical Programmer," Dec. 1998, p. 11). From my perspective as an experienced software developer, Glass hit the nail on the head—as far as impossible schedules being a common problem and management not wanting to compromise schedule or features. And as much as developers may complain about schedules and blame upper management, a lot of the responsibility needs to be shared by developers and project managers for not having the guts to say, "No, you didn't hear me the first time. We cannot deliver a working product by that date."

Master schedules are often developed based on input from overly optimistic developers and project managers, and little or no attention is given to addressing risks. A related contributing factor is that off-the-cuff estimates are often cast in concrete. Part of my responsibility as a developer is to "push down" on suspicious-looking estimates and schedule dates and find out what went into them.

Kudos for making the format and contents of *Communications* more relevant to the practitioner over the past few years.

Ron Pyke
Bellevue, WA

Although I have the utmost respect for Glass's long tenure as a wise person, I am a bit amused by his epiphany about software schedules. Someone of Glass's prominence and time in the profession surely knows that imposed schedules have always been with us. It was yesterday's problem, as well as the "most common characteristic of software projects today," as Glass put it. If Glass consulted with someone other than a software company he would find the situation is much worse among companies for whom software is not their main business. After all, in addition to the inadequate testing Glass cites, these folks have to cope with user training, custom documentation, extra customization, and hand-holding.

Packages can make that situation worse. Imagine the poor soul working within a rigid schedule, trying to bring in a software package (not *develop* a package) to help a company's business practices. Or a hospital IT director struggling with a schedule he or she is told is a matter of life and death. And what about the financial aid people at an educational institution trying to slam a package into a situation in which it does not scale.

In my experience, it is rare to find a chief executive with any understanding of software or its schedules. (Hardware is difficult enough.) We can occasionally "metaphorize" the situation by talking about buildings, contract amendments, or contingency budgets, but it is seldom easy. Quality is a technical responsibility, but the general management view is that the schedule is also a technical matter.

By the way, when people "tell upper management the truth" they get defenestrated, as Glass apparently may not know.

Albert L. LeDuc
Miami, FL

I really enjoyed the December 1998 "Practical Programmer" column. I think Glass succinctly described the problem: "For every manager that says 'no' to achieving the impossible, there are 10 standing by who are willing to say 'yes'." The really difficult part

PAUL WATSON

about this is the pain and suffering inflicted on the subordinates of the yes-managers.

Thanks for a great column.

Dave Kleist
Minnetonka, MN

Wow, does Glass's column sound familiar.

I worked for a legal software company in which management could not restrain itself from requesting changes to the software. These change requests came at a rate of several hundred a week. The programming staff (of two) was constantly buried in change-request forms. We noticed that a large number of change requests were duplicates, and that we'd sometimes make dozens of changes on a particular "screen," only to rewrite it the following week because management wanted something completely different.

One evening, after everybody else had gone home, we took the newest few inches off the stack and snuck the rest of them out to the dumpster. Then we took the rest to a conference room and sorted them by what part of the system they affected, throwing most away. Management was delighted with the progress, apparently measuring the height of the "to do" stack as their measure of productivity. We never told how we did it.

Management finally decided that we needed to get the program released. We spent a week looking through the stack (at that moment several inches thick) with management and deciding which changes were "do's" and which were "waits." Then we outlined a schedule to completion. The second day into the schedule, management canceled the planned release and the schedule and submitted several hundred more change requests. This continued for several years. Surprisingly, we got the system installed in several beta sites where the beta users were delighted with the system.

We were about to send it to the beta testers as the version neared completion. Management went to a convention to demonstrate the software. Response was overwhelming. We seemed to be on the verge of success. But …

Management, which had resisted windows (it was a DOS networking product) returned from the convention realizing that it was not a Windows product. Depression, accusations, and recriminations followed, and a few months later, the programming staff was fired and an expensive consultant hired to write a Windows product. As I understand, three years later, they still haven't delivered a product, and the buyers are looking elsewhere.

Tom Ruby
Industry, IL

As to Glass's statement, "Schedule pressure is the most serious problem facing software projects today" … Amen!

I was on several of these death marches at my last job. No one could actually believe that things turned out as well as they did. Here's some advice: Never be the one person on these projects whose code tends to work—you will be on *every* death-march project the organization has to offer.

Almost no progress can be made on improving quality in such an environment. As an advocate for combining formal methods and modern testing techniques, this is particularly disturbing. Some organizations will set up all the infrastructure to do a better job, only to be circumvented on every project. I have been told by many young developers that employees don't want to hear anything except how quickly they can code C++.

It seems the marketing folks of the leading software companies have done a good job convincing customers that software should be riddled with bugs. As a consequence, everyone now believes the only important thing is how quick you can get the product out. Many have come to the conclusion that schedule-driven projects are the norm. The only thing that will reverse this thinking is for some company to loose market dominance due solely to a rival product's superior quality.

Alwyn Goodloe
Vienna, VA

User's Bill of Rights

Clare-Marie Karat's "Guaranteeing Rights for the User" ("Viewpoint," Dec. 1998, p. 29) has a good analysis and a worthwhile goal, but is presented to the wrong audience. Her User's Bill of Rights, viewed from a slight angle, looks suspiciously like a list of 10 major problems with today's PC software. And she's right—about the problems.

But I think she's wrong about the solution. She calls on the computer industry to design products and systems for their intended users. I'm surprised that a social psychologist would appear to assume the design of today's software products involves a technological process at all; it's a business/marketing process. She's "disheartened by the misinformation … on prod-

ucts these days." So am I, but I don't see it coming from the technology industry. Put simply, it is a direct consequence of mass marketing, which moved into the software business when software became a consumer product.

It's highly unlikely that many *Communications* readers are involved in marketing decisions; most of us live in the world of producing things (and have at least some regard for facts). The problems listed in Karat's User's Bill of Rights do need to be addressed, but if she really wants to see that happen, she needs to make her presentation to the software marketing executives. They don't listen to us technoids.

Daniel L. Covill
San Diego, CA

The Internet came along and spoiled the User's Bill of Rights.

What if a user complains about a safeguard whose purpose is to discourage users from violating laws regarding harassment, export control (encryption), or copyright? The users' only right is politeness when told they're wrong.

The customer and the user may be different people with different objectives. Internet Web browsers and Web sites are paid for by vendors trying to market to the user. The paying customer has the right to expect cooperation from the computer industry in marketing to the user—even if the user finds this undesirable.

Some of the laws now governing the Internet apply not to the user's experience but to what server, company, and political jurisdiction handles the transactions under the hood. While it may be unfortunate that users are being asked to understand the inner workings of certain Internet applications, this issue cannot be addressed by the computer industry. Users should write their Congressional representatives.

I am concerned a 10-point User's Bill of Rights may create the false expectation that the issues are simpler than they really are.

Erik DeBenedictis
Redwood City, CA

I enjoyed reading about a fictional User's Bill of Rights. It was only at the end that I realized Karat might be serious, claiming "the user is always right."

It's been many years since I used an IBM software product (Fortran II compiler, JCL for various mainframes). But I never found those (admittedly older) products any more user-respectful than the typical Microsoft product of today. Has IBM really changed into a user-respecting firm? Speaking of Microsoft, my daughter (not a computer expert) began her weekly phone call by asking: "What do you think about people who take an axe to their computer terminal?" Turned out she was expressing her own experience with the widespread user dissatisfaction with Word and Excel—their fragility, their inscrutability, their intrusiveness, and their incorrigibility.

If one could ever elicit a candid history of most or all of those string-and-baling-wire packages from their original developers, we would discover they were created as described in another piece in the same issue as Karat's "Viewpoint." Glass's "How not to Prepare for a Consultancy Assignment, and Other Ugly Truths" stated: "In retrospect, I should have known that my client's problem was excessive schedule pressure [when management demands the impossible]. I think that's probably the most common characteristic of software projects today [as yesterday and probably tomorrow], across industry lines and domain lines."

Don Walter
Los Angeles, CA

I agree 100% that the industry needs to do better and having a neatly articulated ideal to shoot for will help.

In today's world of shrink-wrapped, mass-marketed software products, we can no longer approach the world the way we did 15 years ago when sophisticated users reported problems on a daily basis and had fixes delivered to keep their half-million-dollar investment running.

Times have changed, and so have expectations.

Gary R. Smith
Novi, MI

I enjoyed Karat's "Viewpoint." I too am a social psychologist concerned about similar issues—not just "make it simple, stupid," which the tech world is waking up to, but how interfaces can be designed to match user preferences (learning/cognitive styles).

Thanks for the fresh air I found in Karat's User's Bill of Rights.

Linda A. Jackson
East Lansing, MI

Response

The User's Bill of Rights is not to be taken literally. It is a vision statement—a perspective from which we, as an industry, can build better solutions. That

is the key, seeing it as a set of goals. Yes, there are caveats and dependencies. Yes, users make lots of mistakes. It seems the technology industry would benefit, though, from keeping in mind the perspective of the customers who will use the solutions we create. And really, the integration problems we have in the industry are serious; we have a ways to go before software and hardware components from different companies really plug and play together. Customers face real challenges using computer technology. People with computer skills and experience who invest generous amounts of time to understand and troubleshoot the solutions they purchase still have significant problems with them.

We can do better as an industry, and we can cooperate to achieve better ease of use for our users. I ask that we as an industry work toward a better balance in developing products and solution components for our customers and end users. In too many situations across the industry, I have seen the technology drive the products and solutions created, rather than the products and solutions being driven by a real understanding of the intended users, their goals, and context of use. Furthermore, as an industry, we need to provide accurate information with products so consumers can make informed choices.

Based on the overwhelming and forceful response from users to the *BusinessWeek* article on the User's Bill of Rights, consumers are very frustrated with the state of technology products on the market today. Seems best that we in the industry address these issues ourselves now—all of us, the human-computer interaction specialists, programmers, graphic designers, information developers, product managers, architects, marketing managers, sales and customer support staff, and senior executives. It's going to take a collaborative effort by members of the industry to achieve the ease-of-use goals for computer users. There's an opportunity for a huge win-win for the industry, our customers, and the communities in which we live. Let's do it.

Clare-Marie Karat
Hawthorne, NY

Please address all Forum correspondence to the Editor, *Communications*, 1515 Broadway, New York, NY 10036; email: crawfordd@acm.org.