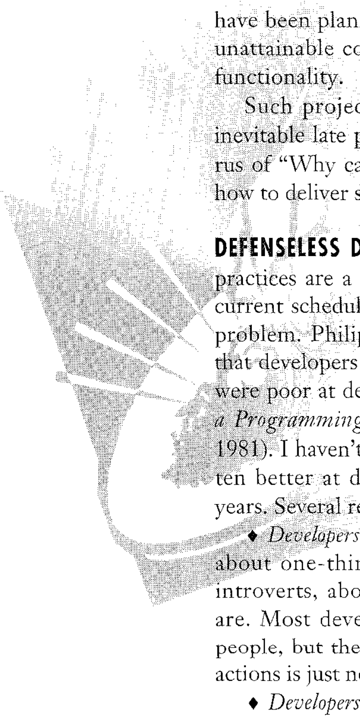
How to Defend an Unpopular Schedule

Prospecting for programmer's gold



ESTIMATING SOFTWARE-PRODUCT SIZE and software-project time and resource needs is hard. But once you have made an estimate, you still must convince your customer or boss to accept it. If the estimated schedule is too long, customers and bosses will pressure you to shorten it — not because your analysis is flawed but simply because they want the software sooner. All too often they succeed and, as a result, many of us find ourselves working on projects that have been planned from the outset to achieve an unattainable combination of cost, schedule, and functionality.

Such projects are programmed to fail. The inevitable late project is greeted with a loud chorus of "Why can't these software guys figure out how to deliver software on time?"

DEFENSELESS DEVELOPERS. Current estimation practices are a problem, but I am convinced that current scheduling practices are the more serious problem. Philip Metzger observed 15 years ago that developers were fairly good at estimating but were poor at defending their estimates (*Managing a Programming Project, 2nd Ed.*, Prentice-Hall, 1981). I haven't seen any evidence that we've gotten better at defending our estimates in recent years. Several reasons for this come to mind.

♦ *Developers tend to be introverts.* Although only about one-third of the general population are introverts, about *three-fourths* of all developers are. Most developers get along fine with other people, but the realm of challenging social interactions is just not their strong suit.

♦ *Developers are at a disadvantage when negotiating schedules with management or marketing.* Gerald Weinberg notes that marketers are often 10 years older than their developer counterparts and negotiate for a living. In other words, they tend to be seasoned, professional negotiators (*Quality Software Management, Vol. 3, Congruent Action*, Dorset House, 1994). The deck is stacked against developers during schedule negotiations.

♦ *Developers tend to be temperamentally opposed to the use of negotiating tricks.* Such tricks offend their sense of technical accuracy and fair play. Developers don't want to offer excessively high initial estimates even when they know that customers, marketers, or bosses will start with excessively low bargaining positions.

To help developers become better negotiators, the rest of the column describes how to negotiate schedules effectively.

PRINCIPLED NEGOTIATION. Start improving your negotiating skills with the *principled negotiation strategy* described in *Getting to Yes* (Roger Fisher and William Ury, Penguin Books, 1981). This method has several appealing characteristics. It doesn't rely on negotiating tricks, but does explain how to respond to such tricks when others use them. It's based on the idea of creating win-win alternatives: You don't try to beat the person you're negotiating with, you try to cooperate so that both of you can win. It's an open strategy. You don't have to fear that the person you're negotiating with has read the same negotiating book and knows the same tricks. The method works best when all involved parties know about it and use it.

Why can't these software guys figure out how to deliver software on time?

The principled-negotiation strategy consists of four parts, which relate to software-schedule negotiations as follows.

Separate the people from the problem. All negotiations involve people first, interests and positions second. When the negotiators' personalities are at odds — as, for example, developers' and marketers' personalities often are —negotiations can get hung up on personality differences.

Begin by understanding the other side's position. I've seen situations in which nontechnical

Continued on page 118

Editor:
Steve McConnell
Phantom Lake Engineering
PO Box 6922
Bellevue, WA 98008
smcconn@aol.com

Continued from page 120

managers had good business reasons for wanting specific deadlines. In one case, a manager felt pressure from had good business reasons for wanting specific deadlines. In one case, a manager felt pressure from the marketing department to produce a software product in six months. I told him that the best I could do was 15 months. He said, "I'm not giving you a choice. Our customers are expecting the software in six months." I said, "I'm sorry. I wish I could develop the software in six months, but 15 months is the best I can do."

Focusing on interests rather than positions opens up a world of possibilities.

He just froze and stared at me for two or three minutes. Why? Was he using silence as a negotiating maneuver? Maybe. But I think it was because he felt trapped and powerless. He had promised a six-month development schedule, and now the person who was supposed to lead the development effort was telling him he wouldn't be able to keep that promise.

Most middle managers aren't being stupid or irrational when they insist on a deadline you know is impossible. They simply don't know enough about the technical work to realize that their deadline is impossible. But they do know all too well how much pressure they feel from their own bosses, customers, and other departments.

What can you do? Work to improve your relationship with your manager or customer. Be cooperative. Work to set realistic expectations. Position yourself as an advisor on schedule matters and avoid slipping into the role of adversary. Point out that by refusing to accept an impossible deadline you're really looking out for their best interests. Point to your organization's history of schedule overruns, and tell them you're unwilling to set up either of you for failure. These points are easiest to make after you've demonstrated a willingness to look for win-win solutions.

Focus on interests, not positions. Suppose you've decided to sell your car to buy a new boat. You've calculated that you must sell your car for $5,000 to buy the boat. A prospective buyer approaches you and offers $4,500. You say, "There's no way I can sell this car for less than $5,000." The buyer replies, "I'm sorry, $4,500 is my final offer."

When you negotiate this way, you focus on positions rather than interests. Positions are bargaining statements so narrow that, for one person to win, the other must lose.

Now suppose the car buyer says, "I truly can't go over $4,500, but I happen to know that you're in the market for a new boat. I'm the regional distributor for a boat company. I can get that boat you want for $1,000 less than you could buy it for anywhere else. Now what do you think of my offer?" Well, now the offer sounds pretty good because it will leave you with $500 more than you would have had if the buyer had just agreed to your price.

Focusing on interests rather than positions opens up a world of negotiating possibilities. Your boss might begin negotiations by saying he needs Giga-Blat 4.0 in six months; you might reply initially that you can't deliver it in less than nine months. But your boss's interest might be in keeping a promise made to the sales organization and your interest might be in working less than 60 hours a week for the next six months. Between the two of you, you should be able to redefine the product so that your boss can keep his promise and you can work a normal schedule.

If one or both parties remain inflexible, software negotiations can become one-dimensional tug-of-wars that focus only on schedules. Don't dig into a position. Make it clear that you're happy to consider a full range of alternatives.

Invent options for mutual gain. Instead of thinking of negotiation as a zero-sum game in which one person wins at the other's expense, think of it as an exercise in creative problem solving: The truly clever negotiator will find a way for both parties to win.

Your most powerful tool in schedule