

Robert N. Charette

How to Create a Successful Failure



Many people enjoy working on software projects that wing their way towards failure. After all, troubled projects offer so much more in the way of excitement and advantages than those that boringly plod their way to success. For instance, projects in turmoil provide numerous ways to look good. You can make both employer and customer deliriously happy, become a hero, and gain a promotion merely for getting something, anything, delivered on time—while on a smoothly running project, it's expected! Even if you don't make the delivery, you can still appear heroic and get promoted. But if you screw things up, who knows? Then there is the increased job security. Projects in turmoil usually go on far past their original deadlines, as customers are loath to cancel and lose their investment. Your company benefits too, as it gets paid much more than if the project was merely successful.

And if the project crashes, you never have to explain to others, especially prospective employers, your work experience. Just say those magic words, "The project got canceled," and immediate sympathy is offered for all the silly customers, stupid managers, and incompetent co-workers you had to put up with. The bigger the disaster, the greater the sympathy, and the less you have to explain your role in it.

Keep in mind the old adage, "There is opportunity in creating risk," and heed the following commandments if you wish to stay on the failure fast track. First, always offer more than you can deliver. Any software project, regardless of kind, creed, national origin, or capability maturity level can be made to fail if there exists insufficient time or money. Ensure you disguise any shortfall by emphasizing the wonderful cost-benefit or time-to-market ratio such a project possesses.

Never plan. Tell your customers and colleagues your gut instincts are more accurate than any formal plan. Equate understanding with action. Dive into your work to show how real progress is made. If they doubt you, regale them with stories of software project-planning disasters. Refer extensively to Peter Neumann's book (*Computer-Related Risks*, ACM Press, Addison-Wesley, 1995), but never let them read it—they might interpret it differently than you. If they still insist on having a plan, create one so intricately detailed that it papers the walls of five conference rooms, and insist it be kept daily up to date.

Always exhibit supreme confidence. Never show any doubt in

your abilities or judgment. Ensure you deny the existence of all potential sources of trouble. Keep difficulties to a minimum by declaring you want to see only solutions, not problems. Demote anyone who brings you a solution. Label all problems that can't be suppressed as "challenges." Claim all problems and risks are being dealt with, or will be in the near future. Vigorously reject any further negative news as being a personal affront to your integrity.

Unceasingly take the most expedient solutions. Make snap judgments and take shortcuts. Select solutions that violate project assumptions, although ones attacking symptoms rather than root causes work too. Try to provide a solution to a different problem than the one discussed, explaining you are applying a bit of lateral thinking. Never, ever think completely through a problem or find out if it has already been solved.

Be simple-minded. View all opinions, no matter how outrageous, as fact. Encourage rumors. Sloganeer where possible, but avoid emulating Forrest Gump—he succeeded. Work on only the easy things. Put first things last. Make suggestions for improvement, especially by using untried technology. Agree to all customer-initiated changes. Always hint such changes are almost free, since software is so flexible. Postpone decisions until more information is available, then make a snap decision. Remember, delay is your friend.

Create decision whiplash. Spawn situations where several decisions are subtly, but highly, interdependent. Then change your mind early and often, particularly the afternoons before holidays and vacations begin. When questioned as to why, just quote Emerson that a foolish consistency is the hobgoblin of little minds. Finally, leave the project but get rehired as an expert consultant. Then you can give the same advice but at twice the price, and what's more, be really listened to.

Over time, with more experience, you will be able to add your own unique commandments to the list. Just concentrate on ensuring that the root causes of risk (i.e., the lack of information, control, and time) all exist concurrently. And don't worry if the project really does fail and gets canceled. Somewhere, someone is already starting afresh. **□**

Robert N. Charette is the president of the ITABHI Corporation. He is a noted authority on relating risks to practice. He can be reached at 75000.1726@compuserve.com, or 703-425-0564.