AMERICAN PROGRAMMER®
FEBRUARY 1997 Vol. 10 No. 2

DEATH MARCH PROJECTS

AMERICAN PROGRAMMER

CONTENTS

FEBRUARY 1997 VOL. 10 NO. 2

2
STAYING SANE ON A DEATH MARCH
Paul Neuhardt

6
RELIABLE WORK ON A DEATH MARCH SCHEDULE
Dave Gorton and others at Bellcore, as told to Don Oxley and Bill Curtis

12
SURVIVING AND SUCCEEDING IN A DEATH MARCH PROJECT
Sharon Marsh Roberts

16
EUTHANASIA FOR DEATH MARCH PROJECTS
John Boddie

22
NINE WAYS TO TURN YOUR PROJECT INTO A DEATH MARCH AND GET THAT PROMOTION
David Kleist

27
YES! GIVE ME A DEATH MARCH!
Rick Zahniser

EDITORIAL BOARD

About a year ago, I began to hear the term "death march" often enough to begin paying close attention. In the software field, I define a death march project as one whose schedule and/or budget are 50–100 percent more ambitious than can reasonably be expected. Alternatively, a death march project could be defined as one whose risk of failure is greater than 50 percent. While management may attempt to compensate for the risk by introducing new "silver bullet" methodologies and tools, the most common strategy for coping with such projects is to ask the project team to cancel their vacations and weekends and work substantial periods of overtime until the project succeeds or collapses.

I vividly recall a death march project that I suffered through for two years in the mid-'60s. I was young and unmarried, with almost no social life, so I survived without too many after-effects. But several careers and personal lives were damaged by the project, including a member of my team who suffered a nervous breakdown a few months before the entire project was canceled. Since then, I've participated in a few more death march projects, managed a few, and witnessed several more as a consultant. Some of them succeed, but most of them fail, and even the projects that are deemed a corporate success tend to cause a high level of divorces, ulcers, and other signs of severe personal distress. It's a phenomenon that seems to be on the rise, too, with the ever-increasing pressure of globalization, intense competition, and organizations living on "Internet time."

In our first article, Paul Neuhardt explores the *personal* consequences of a death march project. Neuhardt notes that it's common to experience feelings of "helplessness, anger, guilt, fear, and depression" on such projects and argues that "it is vital that the people involved in a death march manage to separate the outcome of the project from their own self-image." Suggestions such as encouraging team members to take vacation time, in order to maintain their health and sanity, really do need to be emphasized.

By implication, death march projects are expected to fail, but our second article, by Dave Gorton, Don Oxley, and Bill Curtis, demonstrates that this need not be a foregone conclusion. In a death march project, there's a temptation to abandon all formal methods and procedures and rely instead on intense and frantic forms of hacking. In contrast, Gorton et al. describe a 30-person death march project at Bellcore that succeeded because it stuck to well-understood practices, clear responsibilities, and

specific responsibilities for process coordination. While it was a very short, intense project — only nine hours to make an unexpected modification to the nationwide "800" toll-free phone network — it nevertheless involved some 3,000 lines of code and 2,000 test cases. Even more impressive: it all worked!

Sharon Marsh Roberts returns to "peopleware" and management issues in her article. She notes the importance of a positive attitude, observing that "the first two conditions for success in a death march are vision and realism. The team must commit to delivering and must believe in the value of the end product. If the project is strictly impossible, who can succeed? . . . If the goal is superfluous, who cares?" On the other hand, she notes that "not all negative events have solutions, and dealing with too many of that kind puts a big strain on the team — to the point of collapse."

So while Gorton and his colleagues show that death march projects can succeed, Roberts points out that such projects can fail. Our next author, John Boddie, goes further and argues that in some cases, the best thing is to recognize the failure in advance and kill the death march project before things get any worse. Boddie advises that "you need to arrange the quick death of your project in such a manner that people on the project team are shielded from the fallout. . . . Since the manager is highly visible at the point of the project's demise, the risks associated with being the instrument of the project's termination are enough to make many managers draw back from what needs to be done."

Another common impression of death march projects is that their fate is predetermined: they're either destined to be a "Project Titanic," or they're destined to succeed because of the skill and determination of the project team and project leader. In any case, it rarely occurs to a software developer that a death march project might have been "manufactured" by a project manager or senior-level sponsor. As consultant David Kleist puts it, "Goals handily achieved make boring stories. Legendary figures are rarely people who easily do what they promise." Tongue in cheek, Kleist shows how ordinary endeavors turn into death march projects through guile and misinformation.

Finally, Rick Zahniser summarizes the differences between a death march project (whose results are usually negative) and the "breakthrough" projects he has described in previous issues of *American Programmer*. He suggests that one major cause of a typical (disastrous) death march project is the politics associated with what he calls a "typical power structure," and he provides some interesting guidelines for developers who find themselves stuck on a death march project within such a power structure.

During the summer of 1996, I wrote a book on death march projects. The first draft was downloaded from my Web site by several thousand people, and the hard-copy version will be published by Prentice Hall this spring. In this issue of *American Programmer*, I wanted to see what others had to say about the phenomenon. The wide range of experiences of the authors in this issue confirmed that my experiences were by no means unique. My hope is that the insights they offer will help prevent a few death march projects or enable some well-intentioned but unprepared software developers to survive a death march project with their wits and their sense of self-worth intact.

In the meantime, we'll move on to more technical matters in the next issue of *American Programmer*, which will focus on the Unified Modeling Language developed by Rational Software and a group of other software firms. Shortly after you receive that issue, we'll convene for the ***American Programmer Summit '97***. There is still some space available for the event. Visit the Cutter Web site to register or get more information. As always, we look forward to your comments and feedback; please feel free to contact us by phone, fax, or on the Internet.

Ed Yourdon ★
Internet: ed@yourdon.com
WWW: http://www.yourdon.com
http://www.cutter.com

American Programmer® (ISSN 1048-5600) is published 12 times a year by Cutter Information Corp., 37 Broadway, Suite 1, Arlington, MA 02174-5552 (617/648-8702 or, within North America, 800/964-8702; Fax 617/648-1950 or, within North America, 800/888-1816). American Programmer covers the software scene, with particular emphasis on those events that will impact the careers and jobs of programmers, systems analysts, and data processing managers around the world. Editor: Ed Yourdon. Publisher: Karen Fine Coburn. Managing Editor: Karen Kunkel Pasley. Production Editor: Rosanne DePasquale. Client Services Manager: Kara Lovering. List Manager: Doreen Evans. Reprint Manager: Carolyn Licata. Subscription rates are US$435 a year in North America, US$535 elsewhere, payable to Cutter Information Corp. Reprints, bulk purchases, past issues, site licenses, multiple subscription rates available on request.

Staying Sane on a Death March

by Paul Neuhardt

When I first heard Ed use the phrase "death march" in relation to software projects, I wanted to cheer. I had finally found the perfect description of what I had gone through on a few projects. The image of POWs being led through the jungle at the point of a gun, their physical and mental health deteriorating a tiny bit more with each lead-footed step, their faces locked into blank stares of resignation, seemed the perfect expression for what I and other team members had felt during those projects. The act of designing and creating systems, once a joy to us, had become something that we had to endure in order to keep the creditors away from our front doors. What happened?

Simply put, we were unprepared for the psychological impact of the death march project, and I'm not talking just about the stress of long hours and impossible demands from customers and/or managers. I also include the feelings of helplessness, anger, guilt, fear, and depression that often accompany these projects. As a participant on these projects, I have had to deal with all of these emotions in myself, and as a manager, I have had to deal with them in both myself and the teams I managed. I'll share with you some of the things I've learned about making death marches a bit more survivable for you and for those whom you lead down the jungle path.

There are some issues that are personal and apply to everyone involved on a death march, whether they are managers or not. Others are more important for a manager to deal with, since the manager can make a real difference in how the other members of the team handle their personal issues. I'll discuss some of the more personal issues first, then tackle some of the management issues.

PERSONAL ACTIONS

If You Don't Like to Walk, Don't Join the Marching Band

This was a favorite saying of one of my old high school band directors, and it applies to the death march as well. If you can tell that a project is going to be a death march and you know that you don't want to be a part of one, then for heaven's sake do not volunteer. Admittedly, you don't always have a choice other than take the project or find another job, but for the sake of argument, let's assume that you are given a choice of projects to work on. How do you know that a project you are being offered is going to be a death march? Well, with apologies to the comedian Jeff Foxworthy, you might be going on a death march if:

★ . . . the first part of the project specified is the delivery date.

★ . . . you are recruited for the project by a senior manager who can't remember your name because he only sees you once a year at the company Christmas party.

★ . . . bribes such as trips, dinners at fancy restaurants, bigger offices, or extra stock options are used to recruit team members. This goes double if the bribe is paid just for joining the team and not for actually delivering the final product.

★ . . . the pitch focuses on what a great programmer/manager/whatever you are and not on what a great project this is going to be.

★ . . . there is neither a firm budget for the project nor a process in place for developing one.

★ . . . there has never been a project at this place that wasn't a death march!

Separate the Person and the Project

Projects become death marches for a wide variety of reasons, few of which are directly the fault of the people working on them. Still, if the project you are working on is declared to be less than successful (and a death march is considered a flawed project by definition), one normal reaction is to personalize that sense of failure. It is vital that the people involved in a death march manage to separate the outcome of the project from their own self-image. Make sure you perform your job to the best of your abilities, then take pride in that. If the project succeeds, great. If it doesn't, you still can feel good about your personal accomplishments without making the project's failure your failure.

Get a Hobby or Join a Club

The problem with my last suggestion is that people need to feel that they are doing something worthwhile with their lives, and separating themselves emotionally from their work robs them of a common outlet for that need. But as long as you are on a death march, you are most likely going to have to get that sense of belonging and accomplishment somewhere else. My recommendation is to get a hobby and/or join some sort of social or charitable club. Various people I know have joined softball teams, taken quilting lessons, joined a fencing club, volunteered with a local adult literacy campaign, or joined groups such as the Lions Club or the Exchange Club. All of these people find it easier to deal with boredom, disappointment, and stress on the job because they have organized outlets other than work that provide them not only with a sense of satisfaction, but also with the feeling that they are part of something that is both good and bigger than they are.

Use Your Vacation Time

How many burnout cases do you know that take a perverse pride in how much unused vacation time they have built up? If you've been in this business long, you've probably run into several. These people tend to believe that the number of hours they work is the ultimate measure of success and that vacations are only for wimps.

When stress and tension are high at work, it is important to be able to step away from the office for a while and focus on something pleasing and relaxing, something that you are doing for yourself and not for your employer. While I recommend actually going away for vacation, that isn't always feasible. Fortunately, it can be just as effective simply to take a week to play golf, putter around the house, read some

of those novels you've been meaning to get to, or ride your bicycle around in the woods for a few days. The goal is not so much to go away as it is to spend time focusing your thoughts and actions on yourself and your loved ones and not on your job.

MANAGEMENT ACTIONS

Let People Gripe

On a death march, tempers flare, patience evaporates, and morale declines. It is a good and healthy thing to let people blow off a little steam by airing their frustrations in a public or semi-public forum. Hold meetings where the other team members have the floor and get to voice their concerns about how things are going. One good suggestion I came across recently was to use chat software to conduct these meetings if people feel they need to hide behind the anonymity of their computers. No matter how it is done, a good manager will find a way to let the team communicate not just the specific details of what they are doing, but also how the job affects them personally.

A few ground rules are helpful: no name calling or personal attacks are allowed, no individual person gets more than five straight minutes of time on the soapbox, and the sessions will have preset beginning and ending times. This lets people vent their frustrations without being consumed by the passion of the moment and never getting any real work done.

Be Ready to Be Your Team's Emotional Parent

I've heard so many managers say things like, "I'm not your parent." Well, like it or not, the manager on a death march project becomes a parent figure for the team. The team members look to you to set the tone for the team and show what behaviors are and are not acceptable. If you panic, they'll panic. If you yell a lot, they will, too. If you can maintain a sense of order and calm reason, it will help them to do the same. It is important that you put on a convincing show of calm, even if you are screaming on the inside.

Also, if you show interest in the personal lives of your team, they will feel better about coming to work no matter how frustrating the job may be. Make sure you know the names of everyone's significant other and children. Take the time to talk to everyone once in a while about how their personal lives are going. Furthermore, be prepared to have these conversations at a time of the employee's choosing and not yours. If people feel that they can drop by your office and spend three minutes telling you how they can't get their kids to do their homework or how their husband won't ever do the grocery shopping, they will feel much more able to come to you with work-related problems. They will know you really care about them and not just their output.

Declare the Occasional Day Off Together

One of the best peopleware-for-death-marches ideas I have ever been a part of is the retreat. Only this isn't one of those managerial retreats where people go sit in the woods and learn how to trust others or bond with other managers. This is a full-blown tactical retreat from work. Pick a place away from work such as a golf course, a boat, a park with volleyball and softball fields, or some other place where everyone on the team can get together and interact without the pressures of the job. This should be a regular work day so that people really feel as though they are getting a break from work. Also, don't make anyone declare a vacation day for this retreat.

Another good idea is to ban discussion of any work-related topic for the first two hours of the day. And while this is an exercise in team building and bonding, don't advertise it as such and don't make any specific attempts to encourage bonding. After a paid day off where they get to play and talk about subjects like life, love, and the pursuit of happiness, teams will bond on their own.

Encourage Vacations

We've already talked about how important it is for people to make use of vacation time on a death march. As a manager, you are responsible for setting an example by taking your own vacations as well as directly encouraging others to take theirs. The pressure to make up lost time on a death march project can be tremendous, and people may end up feeling that their use of vacation time will be held against them — that others will think they are not "team players." It is

important for you as a manager to dispel this feeling in your team.

Keep Everyone Informed

One of the most common complaints I hear from staff members on death march projects centers around a lack of communication from management. One of the biggest complaints I hear from death march managers is that people spend too much time gossiping about why things are over budget/overdue/underpowered and not enough time doing the work for which they were hired.

In my view, these two issues are tied together. If you keep your teams informed with as many facts as you can about the project's direction and schedule, they won't spend as much time trying to guess what is going on. If you are behind schedule, say so and make sure you tell people what caused the delay. If you don't have information, make sure you tell people that, too (i.e., tell them what you know and what you don't know). If requirements change, give people the rationale behind the change and then they won't spend a lot of time around the coffee pot kvetching about those lousy users who can't make up their feeble minds about what they want.

If someone brings rumors and gossip to you, don't try to quell the spread of the rumors. Rather, spend some time with that person comparing the rumors against the facts as you know them. After a few iterations of this process, the other team members will begin to do this as well, and you will find that the flow of wild rumors will slow considerably.

Expect a Higher-Than-Normal Turnover Rate

Not everybody is cut out to be part of a death march. I suppose one could argue that nobody is, but that is a discussion for another time. Even people who think that they can handle the stress of a death march at the outset may find that they just can't cope.

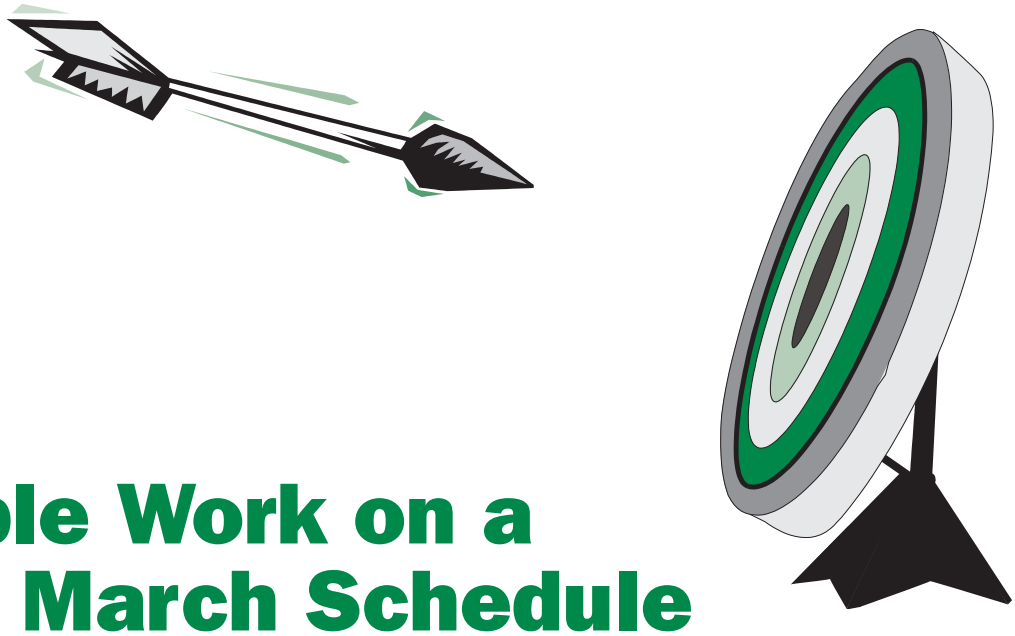
Managers on death marches must expect that the turnover rate of staff on these jobs is going to be higher than normal. When this starts to happen, the worst thing a manager can do is pull everyone together and tell them there is no reason to panic or look for other employment. Telling people not to panic is the best way to make them panic, which in this case means job hunting. Your best response to people leaving the team is to smile, wish them well, throw them a big party, and proceed quietly with life. This will set an example to the team to treat turnover as a normal event in life and not as another source of stress.

CONCLUSION

The term death march has a sinister ring to it and with good reason: the death march project can be a demoralizing experience for everyone involved. Individuals can help themselves by making sure they evaluate themselves by their individual contributions and not by the status of the project. They can also derive a great deal of benefit by looking for a sense of personal fulfillment that does not depend in any way on work. The manager on a death march project is responsible for attending to the mental health of the team as much as to the management of schedules and budgets.

Paul Neuhardt is a systems development manager for BJ's Wholesale Club in Natick, Massachusetts. He has been managing software development projects for the health care, database software, and retail industries for most of the last 13 years.

Mr. Neuhardt can be reached at BJ's Wholesale Club, One Mercer Road, P.O. Box 9601, Natick, MA 01760-9601 (CompuServe: 71673,454; Internet: pneuhardt@acm.org). ★



Reliable Work on a Death March Schedule

by Dave Gorton and others at Bellcore, as told to Don Oxley and Bill Curtis

A project with a death march schedule need not always devolve into a software crisis. A trained and experienced staff using well-established processes knows how to maintain control even when assigned a seemingly impossible schedule. This was the case at Bellcore[1] in June 1995.

THE START OF THE SMS/800 NINE-HOUR PROJECT

The fax from the Federal Communications Commission (FCC) arrived at 5:30 P.M. eastern time on Tuesday, June 13th. It was directed to Database Service Management, Inc. (DSMI), the Bellcore subsidiary responsible for managing the nation's supply of toll-free 800 numbers:

By order of this letter, we direct the Database Service Management, Inc. to limit to two hundred (200) per week

[1]Bellcore was established on January 1, 1984, to provide engineering, administrative, and other services to the telecommunications companies of Ameritech, Bell Atlantic, BellSouth, NYNEX, Pacific Telesis, SBC Communications, and US WEST. Since its founding, Bellcore has grown into a global leader in commercial development of communications software.

Through Bellcore-developed software, U.S. telecommunications companies handle more than 150 million service orders annually and manage almost 200 billion calls without a single service interruption. Bellcore software handles every 800/888 call in the United States, and over 500 companies in 55 countries have incorporated Bellcore software into their networking and telecommunications systems.

the amount of 800 numbers a Responsible Organization (RespOrg) may assign collectively to either "working" or "reserved" status. Compliance with this order is to begin at 12:01 A.M., eastern time, Wednesday, June 14th, 1995, and to continue until further notice from the Commission.

The memo went on to direct that the 800 number system be shut down between 5 P.M. and midnight on June 13th in order to make the necessary changes, and that, effective the next day, DSMI would be required to submit a daily report of 800 number consumption.

BACKGROUND: ADDRESSING THE DEPLETION OF 800 NUMBERS

The FCC order was prompted by concern about rapid depletion of the nation's supply of toll-free 800 numbers. Approximately 7.7 million number combinations are possible incorporating the 1-800 prefix. Of these, almost 6.5 million had been assigned by June 1995. An additional 500,000 numbers had been reserved, disconnected, or were not available for other reasons.

The Carrier Liaison Committee, an industry committee, oversees the 800 number process. In 1994, the industry projected that, given the industry's assignment rate of 30,000 new 800 numbers each week, the existing supply of toll-free numbers would run out, or "exhaust," in the fall of 1996. The industry arranged to deploy a new toll-free area code, 888, in April 1996 in response to this depletion.

The responsibility for developing, managing, and supporting the underlying software for both the 800 and 888 systems rests with the Intelligent Network Solutions Center of Bellcore's Software Systems Group, based in Piscataway, New Jersey. In early 1995, Bellcore was working on development of the new 888 toll-free designation; however, the 888 system would not be on line until April 1996.

In the spring of 1995, assignment rates for 800 service increased dramatically, and the industry realized that the supply of 800 numbers could run out before the new 888 numbers were available. During the week of June 6, 1995, approximately 113,000 new 800 numbers were assigned. This left only 600,000 remaining numbers, which, at a rate of 100,000 per week, would run out in July 1995.

Since the first 800 number appeared on the scene some 20 years ago, the market for 800 numbers has grown to $10 billion. Ninety percent of U.S. citizens report using 800 numbers, and, according to market research, more than a third report using them more than 60 times a year. Depletion of the 800 number supply represented a significant concern for the industry.

The industry asked the Federal Communications Commission for help in returning to its original usage rates, in order to conserve the existing numbers until the new 888 numbers could be deployed. In response, the Commission imposed a cap of 28,000 numbers assigned per week, in keeping with the industry's original projections. According to the memo:

> Because of the significant risk of this outcome — 800 exhaustion before introduction of portable 888 numbers — the Commission must take extraordinary, transitional measures to conserve the remaining numbers and to ensure that the rate of assignment slows to the rate projected by the industry in setting its schedule for introducing the 888 code.

IMPLEMENTING THE ORDER

Responsibility for implementing the FCC order fell to the Intelligent Network Solutions Center. Despite the late arrival of the FCC memo on June 13th, most of the team that had been working on the code that supports the SMS/800 system was still at work. The challenge to the team was to identify the changes required to implement the FCC order and execute them within the FCC's tight time frames without causing additional disruption in the 800 number assignment system.

Accomplishing the FCC order overnight would be a significant task. The SMS/800 system is built on more than a million lines of code. The central function of assigning and managing numbers is handled by components spread throughout the system. Implementing the FCC order required strategic changes throughout the various segments of the code as well as field changes to the SMS/800 database.

The first step was to notify the 800 number data processing center and the help desk so that they could respond appropriately to requests from the Responsible Organizations (i.e., companies that provide 1-800 service, such as AT&T, Sprint, and MCI). A conference call was held to

develop an overall action and communications plan, and a memo was distributed to the 140 Responsible Organizations by 6:30 P.M., outlining the situation and the response.

Meanwhile, the team began meeting to specify the requirements. The first decision was that the team would follow Bellcore's established development process, despite the tight time frames. The sequencing of activities would be compressed by conducting certain steps concurrently, and concerns about formal documentation would be deferred until the next day. But the fundamental elements of the process would remain intact.

The Software Development Process at Bellcore

Bellcore currently has almost 100 million lines of code in its inventory, and that code changes at a rate of 10–15 percent each year. Bellcore is also constantly adding to that inventory with new products and services. As a result, Bellcore has developed a strong culture emphasizing that a well-structured process is the starting point for effective software development and deployment.

This was not always the case. In 1994, Bellcore launched a major effort to improve its software development processes. The initiative, dubbed the QMO (Quality Method of Operation), describes Bellcore Software Systems' software development methodology from project initiation through support.

Prior to adopting the QMO, Bellcore's software development processes differed widely based on the product unit. Some units, such as the SMS/800 group, had well-defined processes, and as such served as early models for development of the QMO. Other product units were less disciplined, and, as a result, Bellcore often found itself missing deadlines, facing quality and customer satisfaction problems, and unable to respond rapidly to crises.

In June 1995, the QMO was officially applied to 16 key projects. With this step, Bellcore strategically began what was perceived as a transformation of the Bellcore Software Systems organization from a research environment to a commercial supplier of telecommunications software. The 16 projects that adopted the QMO were awarded ISO 9001 certification in September 1995.

In 1996, the QMO was extended to all of Bellcore, and in December 1996, Bellcore announced that the entire Software Services Center had received ISO 9001 certification. In the same month, Bellcore Software Systems was assessed to be at the "Defined" level (Level 3) of the Software Engineering Institute's Capability Maturity Model.

Many benefits have resulted from Bellcore's process and quality transformation since 1993. The most important and dramatic improvement is a 40 percent reduction in the density of defects in Bellcore Software Systems' delivered products. This quality improvement has resulted in substantial increases in customer satisfaction. Another benefit is that developers can move from project to project and become productive much more quickly because of the common use of the QMO process.

Applying the Process to the FCC Order

Back on the project, the SMS/800 team drew on a blackboard the changes that would need to be made and, early in the design stage, began partitioning the problem to stage the process. At the outset, a group meeting focused on change control, identifying the different pieces of the underlying code that would need to be revised, as well as new database fields and software component requirements. The team specified testing procedures and defined the workflow to ensure a smooth pipeline of new and revised code.

A key requirement was development of the allocation control component, since every Responsible Organization's activity would have to be tracked and daily reports generated. The allocation formula was complicated by the FCC's decision to allocate 800 numbers based on the market share of the 800 service providers. For example, those service providers who had been very active in the 800 number business would get a larger share of the weekly allotment than those providers who had done very little business in this area. The revised system would need to accommodate this formula.

Four teams were established and an overall time line was developed:

- ★ The *requirements team* was responsible for developing a clear understanding of the FCC's requirements. The SMS/800 team knew in this case that the FCC would not provide detailed product

requirements beyond the specifications in the order, so team members began an intense effort to identify key questions and answers.

What did the high-level functional requirements provided in the order really mean? Every change of number status would have to be counted. As soon as a Responsible Organization exceeded its allocation, the system would have to block further reservations from that Responsible Organization for the remainder of the week. Error messages notifying the Responsible Organization that it had exceeded allocation would need to be developed. Procedures for resetting the counter were needed. A revised report generation process would be required. These and other requirements were specified in group sessions around the blackboard.

★ The *development team* began identifying the modules that had to be changed as soon as the requirements were clearly understood. Where to place the allocation counter was a key issue, as was assuring that the allocation counter was effectively integrated with the multiple routines associated with different ways to receive a number assignment.

The team also decided that it was important to accomplish the changes without having to undertake a total database conversion. The SMS/800 database contains a massive amount of data, and a full conversion process would have been impossible given the time frames. Therefore, the development team worked within spare space in the existing database to add fields and redefine the record layout.

★ The *test team* was responsible for product test. The development team was doing code inspection; developers were testing each other's code in real time (concurrently with development), and code was returned to the original developer for changes. When the code met exit criteria from the development stage, it passed to the test team, which worked with independent testers at another company that manages the data processing center where the system is located. The test team conducted integration and regression testing, with the key question being, "Did the number assignment system still work properly after the changes?" — or, in the words of one team member, "Did we break anything?" When the code passed the exit criteria, help desk personnel from the company that provides these contract services did field acceptance testing in the production system before the system was brought back up.

★ The *deployment team* took responsibility after all testing was complete to get the final production system up and running.

In all, approximately 12 Bellcore staff members populated the various teams through the night. With the additional support from the company that manages the data center and the company that provides help desk personnel, approximately 30 people were involved in the project.

Implementing the system required changes to hundreds of lines of code spread over numerous modules throughout the system. Since the required changes impacted functionality in a large portion of this million-line system, the major challenge was not to affect the other parts. The team applied Bellcore's standard development process, tailored to meet the accelerated time requirements, but still intact in all of its essential elements. All changed modules were inspected (over 3,000 lines of code) and more than 2,000 test cases were executed, including full regression testing of all affected components. To the credit of the team, very few in-process faults were identified and, after a 6 P.M. shutdown, the system was brought back up at 3 A.M. on June 14th.

The entire process of implementing the FCC order was accomplished in nine hours. To date, no operational errors have been attributed to these system changes.

KEY ELEMENTS OF SUCCESS

At the core of this success was Bellcore's strong emphasis on process. With the process in place, there was no question about what had to be done — no question about how to move forward. This was a single cohesive team whose members worked together on a day-to-day basis and, in many cases, had worked together for several years.

According to Executive Director David Gorton, the development manager at the time, "If there had not been an existing,

well-understood process, we could not have gotten everyone together and effectively directed their efforts in a well-coordinated project. This project would have been much more difficult. The process supported an overall organizational effort that allowed this to work."

Certain elements of the process stand out as key success factors in this project:

★ **Clear responsibilities**. Well-defined roles had been established through consistent application of the process. There was no time wasted on questions of who did what or how things were to be handed off. The same procedures applied to a major project were applied in this case — just in a shorter period of time.

★ **Well-understood practices**. The existing process culture empowered the team to focus on understanding the requirements and creatively developing solutions, not on the generation of paper. Put more simply, the process did not constrain creativity, it enabled creativity. The existence of standard testing practices and exit criteria left no doubt what testing process would be used to ensure quality, nor was there any need to focus resources on identifying criteria for release. Ongoing testing procedures were well understood and accepted, and, just as in a long-term project, all code would go to an independent test team. There the code would be subjected to integration testing, functional testing, and regression testing; evaluated against well-defined exit criteria; and signed off and approved before shipping.

★ **Specific responsibility for process coordination**. With 20 to 30 people from three companies working on the problem for several days (after the initial nine-hour project, subsequent FCC orders and refinements resulted in additional changes), clear responsibility for coordination and release approval was essential. An important step was assignment of a coordinator who negotiated responsibilities and timing among the different groups involved in the project. The developers and testers could do their work while the coordinator handled logistics and served as the release manager, assuring all exit criteria were met.

★ **Automated product test**. Regression test cases were automated. This enabled teams to run through thousands of test cases quickly and efficiently to ensure that these emergency changes to add new functions did not inadvertently impact other features.

CONCLUSION

In June 1995, DSMI and Bellcore were required by the Federal Communications Commission to implement changes to the national 800 dialing system before the next business day. Using Bellcore's QMO development process, the SMS/800 project was able to implement hundreds of lines of code, retrofit them into the 800 system, and bring the system back up to full operational status within nine hours. To date, no defects have been found in the code implemented during this nine-hour project.

The process and team that were tested under stress in the wee hours of the morning on June 14th can now claim another major success: with over a million lines of code, the 888 system has been successfully deployed and launched. Implementing 888 support required changes to more than 20 percent of the system. The release that supported that launch completed its production life with only two field faults, and the transition to the new system has been seamless.

Bellcore's QMO proved itself to be a sufficiently strong and robust process to make this crash project a success. Those who were involved in implementing the FCC order for control of the SMS/800 system acknowledge that the process, which effectively supports individual and team achievement, led to the team's success.

David Gorton is currently executive director of intelligent network platform development at Bellcore. He has over 20 years of experience in telecommunications software development. In previous assignments, Mr. Gorton managed the development of software for service management systems, service control points, and intelligent peripherals for IN and AIN services. He holds a B.S. in mathematics and an M.S. in computer science from Purdue University.

Mr. Gorton can be reached at Bellcore, RRC 4A-373, 444 Hoes

Lane, Piscataway, NJ 08854-4182 (908/699-2441).

Don Oxley is co-founder and president of TeraQuest Metrics in Austin, Texas. He has 23 years of experience in managing and implementing large software systems. He leads the effort to define and develop TeraQuest's proprietary system for capturing and analyzing software process measures to provide continuous measurement for client companies. He trains and facilitates executives in software process improvement (SPI).

At Microelectronics and Computer Technology Corporation during 1991 to 1993, he was responsible for successfully restructuring the Software Technology Program and was project manager for the planning phase of First Cities, a project to establish a national trial of interactive multimedia in the home. During his career at Texas Instruments from 1978 to 1991, he held a variety of research, development, and management positions involving the creation of leading-edge computer and software systems. He holds 10 patents and earned an M.S. in computer science from the University of Illinois.

Mr. Oxley can be reached at TeraQuest Metrics, P.O. Box 200490, Austin, TX 78720-0490 (512/219-9152; fax 512/219-0587; e-mail: oxley@teraquest.com).

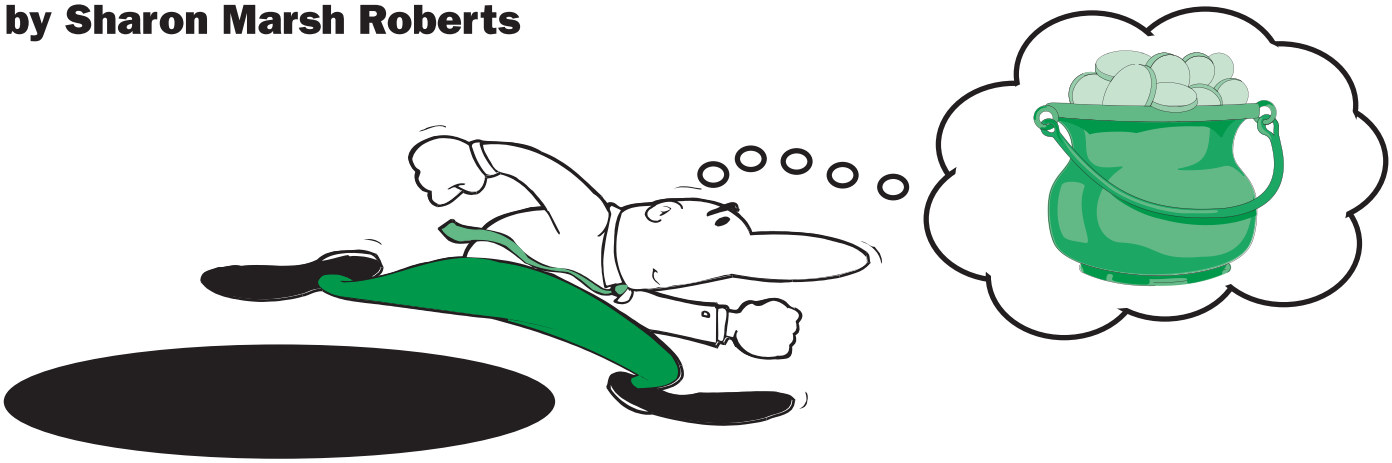
Bill Curtis is co-founder and chief scientist of TeraQuest Metrics, Inc., of Austin, Texas, a company specializing in process assessment and improvement programs, software measurement, risk analysis, and software management education. He is also a visiting scientist with the SEI, where he leads an effort to build a companion to the CMM, the People CMM, for developing and managing software talent. He directed the SEI's Software Process Program from 1991 to 1992, when it produced the CMM, DoD Core Software Metric Definitions, and Process Asset Library.

Prior to joining the SEI, Dr. Curtis directed research on advanced user interface, software design, and project coordination technology in the Human Interface Laboratory and Software Technology Program at MCC. Previously, Dr. Curtis was manager of software trends analysis at ITT's Programming Technology Center and manager of software management research in General Electric's Space Division. He has taught statistics and researched organizational effectiveness at the University of Washington.

Dr. Curtis can be reached at TeraQuest Metrics, P.O. Box 200490, Austin, TX 78720-0490 (512/219-0286; fax 512/219-0587; Internet: curtis@acm.org). ★

Surviving and Succeeding in a Death March Project

by Sharon Marsh Roberts



Accentuate the positive, eliminate the negative, latch onto the superlative . . .

Everyone loves a winner, whether the game is one of skill or chance. The death march project is an arena for risk-taking and rewards. It's the one chance in ten of saving users from their own management, ineptitude, or other ills. Given the overall poor statistics on systems projects, the death march project offers a chance for unexpected and acclaimed success.

How can a project manager increase the odds of success? Maintaining the well-being of the team is vital. A death march project has more than its share of strains. At best, the project is behind schedule for most of its span. The demands on the team are intense, as the users continue to hope (and sometimes push) for miraculous catch-ups. The manager must somehow be an "Energizer" and keep the team going.

The first two conditions for success in a death march are vision and realism. The team must commit to delivering and must believe in the value of the end product. If the project is strictly impossible, who can succeed? No circumstances (other than cancellation of the project) will change the outcome. If the goal is superfluous, who cares? The target project for a successful death march is one with marginal, not impossible, odds.

ACCENTUATE THE POSITIVE

The death march team members generally know that they face long odds. There may be early exits by some of those who find the odds too long. Encouragement will still the cries of those who want to exit. It will motivate those who intend to attempt heroic acts.

Everyone who willingly participates in a death march has sacrificed time and energy in the hope that there will be heroes. The first step in reaching the

death march goals is to acknowledge this sacrifice. Reward and treasure each intermediate checkpoint. Thank team members and those who work with the team. It's too easy to forget the small steps that lead to the final one.

The second step is to send out the word on the heroic successes of the team. Let's define the audience for the hero of the death march. It's the user community, management, and anybody else in the organization whom you can get to see the good things that are happening:

★ Get a user in management to tell other managers how pleasing the newest screens are.

★ Get a clerk to test out how easy the system is to use and to tell his or her manager.

Everything positive that happens during a death march should be cause for celebration. The celebration has dual goals: acknowledgment of effort and publicizing of results.

★ Did the data load from the legacy system run? Call and congratulate the legacy system representative. Thank the manager of the legacy system, the manager of the project, and anyone else.

★ Did the first input screen put all the data in the right places and output the right audit reports? Publish an image of the screen along with the summary project report to management. Let the users try out the screen and record their positive responses.

★

It may seem unrealistic to celebrate a small success when the project is six weeks (or six months) behind schedule. Few projects that start "behind schedule" ever end on time. So start building management and user expectations of increases in functionality from now until completion. Point out the status at project start (usually none). Talk about screens and functions to be delivered in the first month, the second, and later. Ask users and management to ignore the formal schedule and focus on your informal one.

Small successes build into a series, and the team now sees itself as "on a roll." The gamble pays off if the team sees its way to the project's completion.

A simple way to show progress is to put up on the wall a chart of realistic milestones. Suppose there are two input screens required in month one, with the third deliverable in month three. While it's obvious that the project is months behind schedule, observers have noted progress in getting screen two early. Users no longer expect the third input screen for another month or two.

A warm acknowledgment of efforts reassures the team members. It gives them a haven from the inevitable and frequent criticisms of outsiders.

ELIMINATE THE NEGATIVE

Every death march project has many kinds of negatives. Each one must be handled in its own way.

Negative Contributions

The output of some project staff members is negative. Sometimes it's attitude, sometimes it's capabilities. It's important to build a team by selection. Two possibilities are changes in roles and removal from the project.

Changes in roles address a problem of capabilities. A typical problem of capabilities occurs when a team member has been given something that turns out to be too complex for his or her level of understanding of the technology or the overall design.

Let's assume that in this project, adequate resources exist to switch staff. There may be a variety of other roles for a less expert programmer. Programming a less complex module, testing others' work, or working under a part-time mentor might boost productivity.

Project managers would do well to plan additional resources for death march projects. There is almost always a mismatch between the skills that are needed and the times those persons and skills are available. When a project staffing plan is too tight, switches will not be possible.

Teams respond well to appropriate reassignment of their members. Such actions reassure the members that the project manager is looking for productivity, not scapegoats. In one case, a novice programmer was moved from design to testing (where he had adequate support). The project team was pleased with his obvious success and the project continued.

But sometimes the only way to "eliminate the negative" is to

replace staff. Changes in staffing must be clean and sure. Team members do not respond well to threats and open-ended probation. However sudden, a change that is announced in a nonblaming tone, with an acknowledgment of regret, will stabilize the team.

The argument against change is often that a second chance is needed. But in the tight deadlines of a death march, there are demands that are immediate and unrelenting. The two approaches seem to be (1) keeping staff on board with a warning, yet escalating any future problems and publicly chastising them for failure or (2) immediately swapping resources, removing the person who doesn't really show good odds for achieving sufficient growth within the time frame of the death march. Making an immediate change is often perceived as harsh, but it's fairer to let staff know exactly how much room there is for growth and training. People are rarely fooled by "second chances" that are too short to allow growth. When a programmer is allowed to fail publicly, the whole team suffers. The real challenge is to see well in advance how much margin exists for growth.

Many managers argue that it's impossible to remove a programmer whose involvement with a death march is longer than a few weeks. But on each death march I have observed, the least competent programmer was causing bad deliverables, interpersonal conflicts, and aggravated users. When that person was removed, there *was* a temporary lag in delivering new screens or reports. However, fewer errors were reported than for prior deliverables, and other members were relieved that they no longer had to cover for that person's failings. Once the new programmer had been on board for a week or so, the delivery of reports started to match the current schedule.

Negative Events

How do successful death march teams react when something goes wrong? They accept it, solve the problems, and move forward.

Not all negative events have solutions, and dealing with too many of that kind puts a big strain on the team — to the point of collapse. For example, a team building a billing and accounts receivable system carefully accumulated a list of 106 problems and issues. The project was abandoned when neither the team nor management could deal with the state of it.

A strong team mulls over the causes of negative events and goes after the ones with a reasonable chance of solution. But the manager has to move the team members past the temptation to keep reanalyzing the unchanging and unchangeable, or they'll burn out.

Negative Feelings

A team on a death march spends a lot of time together. Feelings between members tend to be amplified. Annoyances can be frequent. Calm acceptance and a wry sense of humor can stabilize teams in hard times.

When the team laughs, it drives away the stresses of external forces. Something about which the team has laughed no longer has the power to bow it down.

Is it crazy to laugh at the system crash that wipes out a day's work? Maybe, but it's better than being slowed in recreating that day's work. It happened, it's over, let's move forward.

Of course, taking this advice isn't easy. It's much more instinctive to vent frustrations. Targeting one's frustrations and succeeding at the project are very different goals. The reason for laughing is that it puts the team in a psychological position to move forward.

LATCH ONTO THE SUPERLATIVE

An extreme challenge requires supreme skills. Nurture the most talented and well-focused team members. Few computer professionals choose their work based entirely on money. It's important to provide an environment that programmers seek. The best team member will want to embrace an organization or team that gives him or her the spotlight and an opportunity to succeed.

Cultivate relationships with the most helpful internal contacts in the corporation. Many outsiders will test communications by giving small amounts of negative feedback. Unheard feedback results in lost resources. Allies are built with gentle feedback. Thanking a contact with a lunch with the team can have the side effect of building good working relationships.

Cherish your most effective managers. Give all of them extreme appreciation. Attend to their questions as soon as possible. Give them reason to believe that their attentiveness is reciprocated. A manager who goes to bat for a team is worthy of the time required to meet any needs. The time it takes to run an extra query or report is much less than the time required to cultivate a broken relationship.

Excellent programmers do more than simply code a module a few hours faster than a novice. The differences in performance often reach a factor of 10 or more. Build a list of special experts, both inside and outside your organization. When you find a wizard in database design or in performance tuning, add that person to your list of contacts. Then call these people from time to time and nurture the relationships — you might need them when a march gets too close to death.

Be sincere and reasonable. A word of thanks from time to time is expected. An extreme expression of gratitude is sometimes appropriate. But it must be sane. Over-praise of trivialities will ring false to almost any reasonable team member. One manager had a specific tone that he used for "attaboy" comments. He never used that tone in normal speech; thus, the tone itself betrayed his lack of respect for the team.

Encourage team members to reward each other. The team leader or manager is by no means the only influence on the team member. His or her peers will provide their own responses. People bond by going to sporting events, lunch, or dinner or doing outside tasks such as car shopping. Help these team-building activities to happen.

A successful project will engender camaraderie as well as growth. Time spent in a successful death march can be ironically rewarding to a team and its members. Though frustrations, tensions, lack of sleep, and other ills abound, the prospect of eventual success makes room for laughter, calm, and unity. Not everyone will experience these positives equally, but one can offer them in a spirit of hope.

THE FINAL WORD: POSITIVE

The success of a death march is the triumph of the positive over the negative. So:

1. Create a positive environment
2. Eliminate obstacles
3. Expect and reward achievement

Of course, this advice is good for any project. In a death march, doing these things will give team members a reason to seek out your kind of management and that kind of work. Sooner rather than later, the team will want to reach new goals, take on new challenges, and again reign over schedule and need.

Sharon Marsh Roberts does planning, design, and management of systems projects for large corporations. She has been conducting financial and systems projects for over 10 years, first as a corporate manager, then for five years with a major consulting firm, and now as president of Roberts Financial Systems Inc. Her projects have included the design and implementation of several systems to warehouse financial data and analyze and report on it to nontechnical users, as well as systems for executive compensation and managed health care.

Ms. Roberts is national vice president of the Independent Computer Consultants Association (ICCA), where her goal is to seek new ways for consultants to work together to meet the needs of corporations. She received an M.B.A. degree from New York University and an Advanced Professional Certificate in Finance from the same institution. Her bachelor's degree is from Bucknell University. She is a CPA and a member of the AICPA and NYSSCPA.

Ms. Roberts can be reached at Roberts Financial Systems Inc., 10 N. Wood Avenue, Linden, NJ 07036; (908/217-1396; fax 908/862-0995; e-mail: 72133.337@compuserve.com). ★

Euthanasia for Death March Projects

by John Boddie

You have two choices. Either this project dies quickly, or you will die painfully. Any questions? I thought you'd have a few. After all, you didn't get into this business to fail, and you're not about to start now. Others may call your project a death march, but you aren't going to let it beat you. You'll look for ways to change the scope, extend the budget and schedule, and try to rally the staff for "one big, last push" to finish and deliver a system that may be somewhat less than originally envisioned but that will still be a pretty good product.

Well, if you really, really believe that this project is truly important — that it is genuinely going to improve the lives of some of your fellow human beings — then take your best shot and good fortune be with you. A few death marches *are* worth the pain. But there are very few of these, and the chance that yours is one of them is small. If, more realistically, you are tending a weed that has sprung from a half-understood buzzword and has been nourished by a testosterone fog,[1] you are going to suffer and then you are going to fail, and you are going to inflict agony on others as well. And they will all curse your name. And you will deserve every bit of it. And all this will come to pass unless you can help the project to die quickly and painlessly.

The immediate issue you face is one of tactics. You need to arrange the quick death of your project in such a manner that people on the project team are shielded from the fallout. When large-scale projects are canceled as failures, it's not uncommon for some people to lose their jobs and for others to find that their chances for career advancement have been "put on hold." Since the manager is highly visible at the point of the project's demise, the risks associated with being the instrument of the project's termination are enough to make many managers draw back from what needs to be done.

Take heart. If adversity didn't exist, there couldn't be any heroes.

HIRED KILLERS MAKE GOOD MONEY

Most large organizations have at least one senior manager who functions as "Chief S.O.B. Without Portfolio." This person's job is to stop the flow of bad news. At the systems project level, the CSOBWP will stop the steady slogging through the false promises and missed deadlines that typify every death march. On a larger scale, boards of directors hire these CSOBWPs to rescue failing businesses.

People fear these individuals, but they also grant them grudging respect. CSOBWPs step in and do something that those who preceded them were unable to do. They stop the bleeding. If the project is shut down, the fact that they shut it down is not a mark of failure. It is regarded as something that needed to be done. If they gutted the project and produced something that is genuinely useful, and did it quickly, their efforts will be counted a success.

The key point is that they did not allow the death march to keep marching.

MAKING FAILURE BLAMELESS

If a project is killed shortly after it starts, it's a mistake. It's not a failure. The term "death sprint" is not part of the lexicon. Remember that you can declare a "start" as soon as you are given some new authority, even if you've been a member of the project team for months. Since you are no longer doing the same job you were doing before, you can act as if the project just began. Nothing you do quickly will be counted as a failure. It will only become a failure if you delay.

The importance of taking the initiative cannot be overemphasized. You need to take the action from inside the project in order to turn it into a mistake. If the action to kill the project comes from outside, all you can do is react, and the project will wind up a failure. It is much better to be seen as recognizing one's own mistakes than it is to have them pointed out by someone else. People and organizations don't get too upset if they make mistakes. In fact, it has become a mark of organizational maturity to admit that mistakes have been made and lessons were learned. Failure is something entirely different. Failure is punished. What you want to do is to turn the death march, which is already a de facto failure, into a mistake that can be quickly brushed aside.

Occasionally, managers who set out to kill a project get diverted by a desire to inflict punishment on those who turned it into a death march. These searches for vengeance inevitably backfire. The perpetrators have either been moved off the project because someone wanted to

[1] Large doses of estrogen also work, but they're less common.

protect them, or, if you corner them, they will calmly state that the project was all right when they left it — therefore you must have screwed it up. If you are considering revenge, you should also consider that you already have a surfeit of aggravation. Why increase it? For you and your team to emerge from this project intact, everybody has to be blameless.[2] The project must be dispatched using untraceable means. The only untraceable means I know of is *new facts.*

A LETHAL OVERDOSE OF KNOWLEDGE

Let's say you "suddenly discover" that the project is not going to be finished in 60 days but rather in 60 months — and only if everything goes right. You can bring this new fact to the attention of any executive who is connected with budgeting the project or who is relying on using the wonderful new system that has been promised, and promised, and promised again.

"Look," you can say. "We've only been able to complete 800 function points in two years, and we're supposed to have 4,000 completed in six months' time. Does this make sense?" Your audience doesn't need to know what a function point is to realize that real delivery is four years away at the current rate of progress. Even if progress could be doubled (which nobody believes), the project would still be two years away.

Simple extrapolation is always available. On a death march project, there will be plenty of real data to give weight to the question, "Who are we trying to fool?" If you started with 150 requirements and you have been getting 30 new ones a month, simple extrapolation can show that the original estimates are completely removed from the reality of the actual situation. You can use this new fact to call for a reassessment of the business case. Just be sure to make it blameless. The project didn't wind up in trouble because the original requirements effort was botched. It was just that the whole thing was started prematurely. Everybody wanted to get the benefits of the new system. The complexity simply was not obvious. It was an honest mistake.

New facts can be gathered from outside the project as well. Changes in technology or the market can significantly affect the payback associated with pouring more money and people into the project. "Nobody's going to dial us direct. Everybody's going to use the Net. Why are we developing special programs that we will have to ship to our customers, not to mention the update problem? Shouldn't we be reassessing our current direction?" Yes, we made a mistake on this project, but who could have anticipated this sort of change?

Don't use new facts in a negative way. Show that you are learning from them. You are supposed to learn from mistakes. Use new facts as the basis for questioning whether you would be doing what you are doing if you had known earlier what you know now. You will be surprised at the number of people who already have the same answer you do.

Of course, you can't just run around telling everyone about the new facts as if you were some sort of modern-day Chicken Little claiming the sky is about to fall. These are facts, and they must be presented in a credible and professional manner.

THE HYPODERMIC PROJECT PLAN

On a death march, nobody wants to look at the project plan. It's probably hard even to find one. Instead, there will be short lists of tasks with dates written beside them and no way to tell if the list you're looking at is the current one or the one that was superseded last week. Most people think that the project plan no longer serves any useful purpose. That's only true if you are trying to save the project.

If, on the other hand, you are trying to turn a long-running failure into an instant mistake, the project plan is a handy tool for serving up the new facts I was just describing. By finding a copy of the original project plan, the extrapolations of confusion and delay encountered thus far can be reflected in computer-produced multicolored Gantt charts and tables. Eureka! You now can show "the complete picture" as a new fact. Remember, this is all a revelation to you as well as to your audience.[3] Nobody had any idea that the

[2]The exception to this rule involves consultants and other outsiders, but we'll cover that shortly.

[3]This is likely to be true. It's rare that there is detailed knowledge about how bad things really are.

project had grown so large and would consume so many resources. You aren't happy to be the bearer of bad news, but the numbers are what they are, and they're based on the actual experience to date. There's not much you can do about that.

Of course, you can also be creative when you introduce the new facts into the project plan. You can and should document all of the interdependencies, both among the project tasks and with the activities of outside organizations. Make sure you include all the interdependencies that might exist in the future as well. If you overstate things slightly, don't worry. Nobody is going to analyze your updates to see if they all can be verified. As a point of interest, the presence of all these hitherto unappreciated interdependencies is, in itself, a new fact. It "proves" that the project is far more complex than was first suspected.

The other advantage of myriad interdependencies is that their presence makes the project inelastic. Armed with the "corrected" project plan, you can now demonstrate how you tried to find an approach that reduced the scope of the project in an attempt to develop a strategy for rapid completion, only to find that the manner in which activities depend on other activities means that the time and the cost simply don't shrink by much. Offer to let other people try it as well. Nobody will call your bluff.

Somebody will always raise a stink about finding the person who let the project get out of control. Now is the time to be statesmanlike. Point out that people have been working very hard and that the desire to make progress as rapidly as possible probably led to some reporting breakdowns that were never followed up. In addition, the effort of all the interdependencies was not fully appreciated. Besides, you will point out, the issue is not where the project has been, but where it is going to go. It is, after all, consuming $41,000 per week (you can prove this by referring to the project plan), and a decision needs to be made quickly whether to shift the current resources over to projects where the money and people can be used more effectively.

THE MORE IMPORTANT PROJECT PRESCRIPTION

If your project is a death march, there are certain to be other projects in the organization that are also in trouble. A few of these may be worth saving. Think of how rewarding it would be to be a team player and transfer people to help out on a project that everyone agrees is more important to the enterprise. Think how rewarding it would be to not be working on your current project anymore.

If you do the job right, the death march will be allowed to die quickly and quietly because something else was more important. It wasn't a failure, it was simply a question of business priorities. Who knows, you might even get a team luncheon as the company's way of saying how much the hard work was appreciated and how sorry they are that the project is being canceled.

Of course, you can't run around offering your team members to any and all takers. That would be sabotage, and you can't do that to your project. What you need to do is to find a worthy project that will serve as a lifeboat for the members of your project team. You don't want to put them in a position where they are scrambling around trying to find ways to stay employed. If you are lucky, you will find a project in need that is also a project that your project is dependent upon. This dependency does not have to exist strictly in the sense of a system that would supply data to your system if the two of them were ever completed. It can be a dependency in the business sense. Why should you and your team strain to build a sales campaign coordination system if the marketing analysis system that is needed to identify which campaigns should be undertaken is not available?

It is not sabotage to question publicly whether it would be better to pool resources to complete System A (which is not your system) and then redirect them to complete System B (which just happens to be your death march). By looking at things this way, it becomes possible to turn the current disaster into a mistake of timing. If your suggestions are followed, your project will be "put on hold" while resources are redirected, and, once this happens, it will never see the light of day again.

The importance of having an alternative to your death march can hardly be overstated. If no alternative exists and your project is stopped, it will look like a failure despite your best attempts to convert it into a mistake.

The only condition where this is not true is the one where the

fundamental mistake was in trusting "the experts."

SACRIFICE AS A MEANS TO THE END

We're not talking about self-sacrifice here. We're talking about sacrificial lambs — your consultants. Consultants can be relied upon to foul things up and to be "out of sync" with the true direction of the project. The magnitude of the damage they have caused is the sort of new fact that can drive a stake through the heart of even the most bloodthirsty death march. They don't even have to be working on the project when you move in to administer the coup de grâce. All that is necessary is that their fingerprints be all over some of the original decisions that can now be identified as mistakes.

Most consultants are all too willing to help you in this cause, even though they may be confused with regard to your motives. Assuming that consultants were engaged early in the project, it's likely that they recommended a new way of doing things. This new way might have been to use a new methodology, some new software tools, or some new management approaches. In other words, they will have sent your project along a route that is not appropriate for your organization.

If your project got into trouble without outside help, you may have to rent some. If you can bring some consultants in to perform an assessment of the project, you should be able to set up a win-win situation. If the consultants come back and recommend killing the project, you can be reasonably sure that their report will not assign blame. After all, they don't want to offend anyone, because they want to be invited back to work on your project's successor. If the consultants report that the project can be saved, there is a good chance that the time and cost projected for saving it will be unacceptable to your management. The great thing is that, in either case, the consultants will use your project plan as one of their primary sources of information. In effect, they are simply editing the obituary you have already prepared.

RIGOR MORTIS

With all you are doing to help the company and the people on the project team by dispatching the project, some people will resist your efforts. These poor souls are not your enemies; they are misguided individuals who need to be brought from the darkness into the light. Once they are in the light, you can get a clean shot and avoid wounding them.

You can count on those who believe the project is still viable to claim that:

1 **It's your negativity that's the problem**. The project has difficulties, but they are not insurmountable.

   Response: Call their bluff. Say that you will be only too happy to step aside and let them take full responsibility for the project. You've looked at the facts, and all you are doing is pointing out the obvious. Killing the project at this point is a misdemeanor. Ignoring the facts is a felony.

2 **The project is in much better shape than it appears**.

   Response: Show me. Have them show you how much of the functionality originally proposed for delivery by today is working reliably.

3 **There's so much money and effort sunk into this project that it would be stupid to stop at this point**.

   Response: Ask how much more money and time should be spent before canceling the project stops being stupid. Ask them to be specific — 2.5 times the original estimates, perhaps?

4 **Don't give me excuses, give me results!** Get your butts in gear and finish this thing!

   Response: Say no. Tell the individual who is creating this bluster that you have looked at the facts, and what he or she wants is simply not going to happen. Often, this sort of challenge moves quickly into an attack on your negative attitude; we've already covered that above.

In each case, arguments against stopping the project are ridiculous, and you should have no trouble getting people who would make these arguments to see that. If they stand up and voice them in public, they will be putting the rope around their own necks, not yours. They will have defined themselves as part of the problem. It is not necessary to be overly

subtle when encouraging them to change their point of view.

LAST RITES

It's OK to be satisfied when you have stopped a project that needed stopping, but don't let your satisfaction show. Lament the fact that the mistake occurred. When discussing the end of the project, use the term, "It was the right decision." It wasn't a "good" decision or a "timely" one, it was just the "right" one.

I think it's a good idea to write a "lessons learned" memo for your boss. In it, cover everything but the lessons that should have been learned. Recognize the hard work done by individual members of the project team. Talk generally about the need for better project tracking and allowances for additional analysis prior to estimating. Mention that there were "hundreds of things that worked against success in this project," but don't refer to it as "doomed from the start." Leave the impression that this was a mistake and that everyone, including you, is now older and wiser.

Of course, there really were some lessons learned, and you probably learned more of them than anyone else. This is good. You will probably have several more opportunities to use your newfound knowledge as your career progresses.

John Boddie is a consultant based in Landenburg, Pennsylvania. He is the author of Crunch Mode *(Prentice Hall, Yourdon Press, 1987). In his career, Mr. Boddie has managed everything from a state lottery system to a system that controls the octane level in the gasoline you buy. He can be reached at Unusual Software, P.O. Box 7898, Newark, DE 19714 (610/274-8023; e-mail: 73757.3311@compuserve.com).* ★

Nine Ways to Turn Your Project Into a Death March and Get That Promotion

by David Kleist

Had we lived, I should have had a tale to tell of the hardihood, endurance, and courage of my companions which would have stirred the heart of every Englishman. These rough notes and our dead bodies must tell the tale.

Robert Falcon Scott (1868–1912), British Antarctic explorer

Let's face it: goals handily achieved make boring stories. Legendary figures are rarely people who easily do what they promise. Rather, it is triumphs over adversity or struggles against great odds that are lauded. No one relates the tale of the manager who sends team members home to healthy marriages and happy home lives, or lets them leave early on Friday to get a good table at happy hour. Nobody wants to hear how easy it was. We want to hear about shouting matches and thrown objects, project managers putting fists through walls, wholesale turnover of project teams, divorces, and suicides. If the project was easily completed, then the task wasn't very difficult.

Now if you are an ambitious project manager or project sponsor looking to make a name for yourself, these are depressing observations. The installation of the new general ledger package rarely provides the same potential for heroic performance as the race to the moon or the polar expeditions. If you want to increase greatly your chances for promotion, exposure, or fame, you'll need other people to see your project as overcoming nearly insurmountable obstacles. Since it's hard to hide a staff of 100 people lounging around, it's easier to keep people busy than to have them seem busy.

Fortunately for most software projects, there is little difference in appearance between projects constrained by external events and those constrained artificially.

In addition, we are not rewarded and recognized for the job we do, but for the one that people think we do. Combining these two observations, let's look at an easy solution for obtaining rewards and recognition: turn your project into a death march.

In this article, we are not defining death march projects as "Bet the company" development projects: these are rare, and they usually come under too much scrutiny to fit our purposes. In this case, we use the term "death march" to denote the expansion of a project into a grueling, intensive, lengthy, expensive, and difficult process that gives the appearance of requiring strong leadership in order to meet the project goals. Fortunately, project sponsors and steering committees rarely ask questions like "Why are you doing these things?" or "How would this work better?", especially since they usually don't have enough detailed information about the project processes to make a qualified assessment.

Of course, it is not enough to want to do a death march: there are forces that will try to prevent the conversion of a project into a death march. To counteract these forces, I've listed some guidelines on how to turn your project into a death march and keep it going.

1. Be sure that the project strategy is different than the business strategy for the company or division.

The overall strategy refers to the three types of strategies described by Michael Porter [1]: cost leadership, differentiation, and focus. When the IT strategy is misaligned with the company strategy, this allows for all kinds of confused messages to be given to the project team. It also helps if you can get the user representatives to misinterpret business strategy, thereby compounding the number of mixed messages.

For example, if the firm is a low-cost producer, have the project adopt a specialized, we-meet-all-needs strategy (a type of differentiation). This sets up a fundamental conflict between the costs to the business and the benefits as defined by the MIS area. This encourages the project team to generate additional requirements for its own project, while the project sponsors will look for ways to control and minimize costs. The efforts to squeeze additional functionality into the project while maintaining cost levels will add additional pressure on project staff.

Conversely, a low-cost approach for a project in a firm that is innovative in its services or products will build a self-maintaining process for creeping featurism. The business side will continually look for ways to include functionality needed to support business goals, while the project team will attempt to eliminate or constrain requirements in order to minimize costs. With luck, the project team will need to include several iterations of project rework in order to include needed functionality that it eliminated in the previous go-round.

2. Recognize people for what they are: Portable Production Units (PPUs).

Aside from tool skills, people are pretty much the same. If a person has the tool experience and a pulse, there's no good reason why he or she can't contribute to the project immediately. Understand that people can easily be replaced or added to the project. When in doubt, add more PPUs.

This recognition will also help you maintain some distance from the inevitable recycling of PPUs on your project. With time, even the best-performing PPUs will burn out and need to be replaced. The process of replacing them will be much more efficient if you learn to expect this and plan for it appropriately.

3. Be sure to hold the project team accountable for your decisions.

PPUs need to be driven. We've all seen the figures for varying rates of productivity among programmers and designers. It's obvious that the one thing that varies among these groups is the leadership. This is an opportunity for you to display strong skills in pushing the team forward.

Remember, you're the project manager (or project owner). Don't tolerate inconsistent messages. Because most team members will not have your overall view of the project, their understanding of the project issues will be flawed or incomplete. Help them be aware of these shortcomings. Conflicting opinions or ill-formed messages only confuse team members and detract from the strength of decisions. If a team member will not go along with the overall focus and message of the management team, be sure to have him or her removed, eventually. Decisive leadership requires strong decisions.

4. Build a coalition of stakeholders that have different, yet compatible, goals for the project.

This works best with vendors or consulting firms, especially if you

can get them to bring some staff onto your site. Keep the waters murky by avoiding clear definition of roles and responsibilities: instead, rely on verbal agreements and the good faith of all parties involved.

When you are lucky enough to have vendor or consulting staff on site, the best method for bringing them into your project is to isolate them in their own subprojects. This will minimize the communication and interaction between your group and the vendor's group. Working with a consulting firm gives you an additional option in addition to the one just listed: you can intermingle overpriced, technically challenged, polished staff with your staff. This approach helps build a death march by sapping time away from other team members and diluting their effectiveness.

5. Keep project goals vague.

If no one knows where you are going, no one is going to know how long it will take to get there. Of course, most budgeting processes discourage the allocation of funds without a reasonable return on investment. There are several ways around this: one is to get a golf-game project; another is to find a buyer of silver bullets.

The golf-game project is created when a senior manager of the client company plays golf with a senior manager of the vendor company. By the 13th hole, they agree on the purchase of a system. What it is supposed to do and what the benefits are come later. (Golf is used only as an example: many other activities may be substituted.)

The buyer of silver bullets is an executive willing to fund a project to avoid falling behind the competition in the use of new technology. This can be a particularly attractive option, since trade journals are a great source of market research and justification. Particularly helpful is a journal's newly created Annual Awards for Technology X in its first year: the glowing reports serve to override other objections and can be particularly persuasive if the winner is a competitor.

Should these options not be available, an alternate method would be to increase the vagueness of project goals. Look for opportunities to pull in additional functionality for the project. Expand the scope of the project. If it is not clear how to do this without being obvious, add additional groups of users or stakeholders that were not included in the original definition of the project. This is especially helpful if their requirements conflict with the original requirements (try to meet both sets) or if they are not related to the original set of requirements. Avoid any real opportunities for synergy, although you'll need to use the term in your justification for including additional requirements. You also could go back and question all the assumptions that were made about the project, especially in the second or third year of the death march. This will allow you to pull in the additional functionality left out in the project's first pass.

6. Try to entrap people in the project.

One of the biggest problems with a death march is sustaining the resources needed. People will want to leave. You have several options at this point:

1. Pay good money for employees or temporary help. This will keep some project staff from leaving and helps fill the pipeline with prospective hires and contracting staff.

2. Conduct the project in an area where the job market is tight. This limits the options available to employees and contractors. It becomes much easier to demand the kind of effort a death march requires when staff members feel they have no other options.

3. Try to staff with as many new hires as possible. People are reluctant to change jobs twice in a short amount of time. This works especially well if you've taken a technology approach to guideline #5 (keep project goals vague) and lured new hires with it. Granted, you don't really have to use the technology, but that's not important.

There are also tactics for delaying turnover among the staff on the project. If you have used the technology option for #5, postponing training for new technologies can hold off the exodus. Frequent reorganizations are another method for fending off decisions to leave the project: it often takes some time for people to determine if a change is a positive one. Shifting roles can be another method, since you may be asking staff to master new skills and duties or brush up on old ones. The time spent mastering

the new skills will take the focus away from other project issues.

7. Make sure that the project status relates a sense of progress.
A major problem in driving a death march to its conclusion is ensuring continued survival of the project. The risk in letting others know that this is a death march project is that they may not wish to continue funding the project, especially if it is in its early stages. If people get the idea that the project is a Flying Dutchman that will never reach port, they'll sink it. You need to give the project funding group and the project members some sense of achievement.

There are several ways to do this. First, use time elapsed and resources expended to track the phases of the project. In other words, after spending 600 person-hours and three elapsed weeks, declare the user interface design to be complete. Move on to the next phase. This allows you to declare portions of the project complete and communicate progress.

A second option is to begin work on the final deliverables as soon as possible. Programmers should start work on code or maybe design early on. The more work you complete at the beginning, the closer it will seem that you are to the end of the project. This also makes it easier to bring on more staff, since your current staff is already at 100 percent utilization.

In either case, try not to relate the rate of progress to the overall project. The relation of the rate against project size may raise too many questions about the completion time for the project. For example, a completion rate of 50 modules a month may sound impressive until one understands that the original estimates may have guessed at a final size of 2,500 modules.

8. Negotiate funding and completion dates.
Refuse to let reality impinge on your ability to push the project to completion. The business needs of the funding group should drive the priorities and emphasis of the project. Business constraints should determine the project parameters, not the project itself. If there was not a clear business need for the project, there would be no project request. Funding and completion dates should be determined solely by the project owners and sponsors.

This works particularly well if you've built a coalition of funding companies that use a cooperative effort to build the systems. While it might appear that this coalition is similar to guideline #4 (different yet compatible goals), the difference is that the situation for #4 often has one sponsor funding most or all of the development. In this scenario, we have several companies that each share in the costs of development. The negotiations about each contributor's bill will strengthen the urge to negotiate down the overall project cost.

9. Avoid failure at all costs.
Of all the things that a project manager can inject into a project to turn it into a death march, the two most powerful are an obsessive need for completion and a near-hysterical fear of failure. These can be powerful influences over the project. Don't hesitate to communicate these drives through expressions of anger, frustration, blame, and intolerance. Your fears can be projected onto the team members to spur them to greater efforts. Don't hesitate to use these to guarantee the completion of the project.

Despite all your efforts, the project still may not achieve the original goals set for it at the beginning. Should this be true, all is not yet lost. If you've successfully conducted the project as a death march, enough time may have passed to allow you to change the goals of the project. By changing the goals, you can proclaim victory and shut down the project. If you do use this option, encourage or drive team members to leave the company to avoid lingering questions or future embarrassments. It may also be a good idea to take your promotion in a different department or area, if possible.

As R.F. Scott points out, you really need to be alive in order to reap the rewards and tell the tales. All of these prescriptive items will be of no avail if you do not survive your own project. The guidelines I have listed are meant to be inflicted on other people. Should you find yourself threatened by your own project, don't hesitate to jump into the first lifeboat. Going down with the ship is great in adventure stories, but not necessary for your career. Unlike the Navy, sinking your own ship does not prevent you from getting another one, and your next will likely be bigger than your last.

REFERENCE

1 Porter, Michael E., and Victor E. Millar. "How Information Gives You Competitive Advantage." *Harvard Business Review* (July–August 1985), pp. 149–160.

David Kleist is a principal with Karenius Systems, where he works with small- and medium-sized firms in technology planning, project planning, requirements gathering, and project management. Prior to joining Karenius, he worked for several consulting firms and Fortune *500 companies, primarily in financial services and banking. Mr. Kleist has received an M.B.A. in information and decision sciences from the University of Minnesota and a B.A. in mathematics from Carleton College.*

Mr. Kleist can be reached at Karenius Systems, 3722 W. 50th Street #104, Minneapolis, MN 55410 (612/606-5091; e-mail: 70730.1613@compuserve.com). ★

Yes! Give Me a Death March!

by Rick Zahniser

Showstopper! [2] describes the death march toward the first release of Windows NT. Many of those marchers wound up *rich*! I've been on four large and small death marches. None of them made me rich, but I wouldn't trade the experience for money. (Well, maybe a *lot* of money!) I have some warm recollections of the death marches I've been on, and I would do another one under the right circumstances. Let me explain.

EVERYONE SHOULD DO A DEATH MARCH

A death march creates great memories and war stories. Mine warm the cockles of my heart because I learned so much. In fact, you are naive and gullible until you've been tricked into signing up for at least one.

Your co-marchers will become some of your closest friends, at least while you're on the march and perhaps beyond. And if you reach even a few of the march's goals, you will be very proud.

Notice, however, that most of the rewards are intrinsic. That's all right; much of our motivation in systems work is internal.

WHAT'S WRONG WITH A DEATH MARCH?

There are, of course, some serious costs in death marches, and most of them don't show up on a balance sheet. Marchers suffer both mental and physical deprivation

and so do their spouses. Generally the rewards, both real and promised, are not proportional to the efforts put forth by the marchers.

Most death marches occur in a Typical Power Structure (or TPS; see sidebar), and a couple of grievous sins are committed here. In a TPS, it's OK for managers to make commitments to higher management that overcommit their people. After that, it's OK for these same managers to coerce workers into working long hours to meet these commitments. Finally, managers are not held directly responsible for the commitments they've made, but the workers are held to the fire throughout the march. This is a thoroughly dishonest environment; few workers walk proud here.

HOW ABOUT A BREAKTHROUGH PROJECT?

Instead of a death march, you might get the same rewards from a breakthrough project [3]. Team research tells us that most of us really want to do something exceptional with our work, and we're willing to overcommit ourselves and then bust our hump to meet those commitments. Here are the rules that create such a situation:

- ★ **Joint vision**. The project vision must be jointly developed by the team members.
- ★ **Voluntary commitment.** Marchers must volunteer and must have the option to refuse.
- ★ **Rewards** should be proportional to the risks. Punishment is inappropriate.
- ★ **Publish or perish**. Both successes and failures must be very public; ideally, they should be shared by the networked team with some sort of groupware.
- ★ **Problems as potential.** All problems and risks are shared and exploited as opportunities for breakthrough rather than threats.

AVOIDING A DEATH MARCH

Can you turn a death march into a breakthrough project? Perhaps not, if you're working in a TPS. Here's the way the above rules don't work in a TPS:

- ★ **Vision.** In a TPS, the tip-top manager develops the vision without involving the marchers who will actually make the vision a reality. They probably regard the vision as an hallucination.
- ★ **Commitment.** TPS middle managers build the project plan and then coerce their people to "buy in" to it.
- ★ **Rewards.** There's a lot of secrecy in the TPS.
- ★ **Problems.** The TPS submerges problems and lets them grow beneath the surface. "Don't make waves" is the key to success and promotion in a TPS.

This is so much a way of life that many managers and marchers simply see it as "the way things are." However, both managers and marchers may be able to change things.

Managers: Earn Trust and Then Exploit It

If you're a manager, take the following steps on your next project:

TPS = TYPICAL POWER STRUCTURE

The typical power structure is hierarchical. Peter Block [1] contrasts hierarchical organizations with entrepreneurial ones as follows:

- ★ **Employment contract.** In the hierarchical organization, this is patriarchal: top-down, high control. The entrepreneurial organization believes that authority is based within the individual.
- ★ **Self-interest.** In the hierarchy, success is defined as moving up the ladder, gaining more power and responsibility, and being rewarded financially for that. In the entrepreneurial organization, success is defined in terms of contribution, service to customers, and personal satisfaction.
- ★ **Tactics.** The hierarchical organization uses manipulative tactics. The entrepreneurial organization uses authentic tactics, which encourage us to act on our own values.
- ★ **Dependency.** In the hierarchy, we feel that our survival is in someone else's hands. In the entrepreneurial organization, we feel that we are autonomous, that we own our own actions.

1. Share problems. If necessary, educate marchers to understand deadlines and financial constraints.
2. Build implementation plans jointly.
3. Ask for commitment and publish what you get.
4. Publish progress and publicly reward good performance.

Marchers: Set Your Own Terms

Even in a TPS, you can campaign for a better way:

1. Ask for full disclosure.
2. Set your own conditions for accepting imposed commitments and put them in writing.
3. Ask, "What's in it for me?" (Do all of the bananas go to the trainers, or do the monkeys get some, too?)
4. What happens if we fail? (Do we get "attaboys" for trying, or will we be scourged and pilloried?)

If you can't get any of these concessions, perhaps you need to find someplace else to march.

SIGN HERE!

I'm looking to sign on for a "death march," because I believe that I can turn it into a breakthrough project. How about you? Where do we sign?

REFERENCES

1. Block, Peter. *The Empowered Manager*. San Francisco: Jossey-Bass, 1987.
2. Zachary, G. Pascal. *Showstopper!: The Breakneck Race to Create Windows NT and the Next Generation at Microsoft*. New York: Free Press, 1994.
3. Zahniser, Richard A. "Breakthrough Project Management." *American Programmer*, Vol. 7, no. 6 (June 1994), pp. 13–17.

*During the 1960s, Mr. Zahniser built complex systems for ITT and IBM. In the 1970s, he was a consultant, team leader, and manager for CIBAR, Inc., and System Development Corporation, specializing in real-time and data-based systems. During the 1980s, he wrote and taught professional seminars for CIBAR Systems Institute and Learning Group International (*Structured Design and Programming, CASE Hands-On*) and built a number of PC-based applications to apply new analysis, design, and prototyping techniques and tools.*

In 1990, Mr. Zahniser founded CASELab, Inc., which focuses on improving software team productivity. He is the architect of Design By Walking Around, an N-dimensional approach to system analysis and design, and JMPSSTART, a highly structured team approach to rapid software development. He coaches teams in breakthrough project management and execution and supports networked teams with Microsoft groupware products. You can discuss these and other ideas with him on CompuServe's CASE Forum, where he is a section leader.

Mr. Zahniser can be reached at CASELab, Inc., 10915 Raygor Road, Colorado Springs, CO 80908 (719/495-3483; e-mail: Rick_Z@compuserve.com or rick@caselab.com). ★