1 Introduction

Happiness lies in being privileged to work hard for long hours in doing whatever you think is worth doing. One man may find happiness in supporting a wife and children. And another may find it in robbing banks. Still another may labor mightily for years in pursuing pure research with no discernible results.

Note the individual and subjective nature of each case. No two are alike and there is no reason to expect them to be. Each man or woman must find for himself or herself that occupation in which hard work and long hours make him or her happy. Contrariwise, if you are looking for shorter hours and longer vacations and early retirement, you are in the wrong job. Perhaps you need to take up bank robbing. Or geeking in a sideshow. Or even politics.

—Jubal Harshaw, in *To Sail Beyond the Sunset*
by Robert Heinlein (Ace Books, reprint edition, 1996)

What is a death march project? What makes IT organizations create such things? Why would anyone in his right mind agree to participate in such a project?

To many grizzled IT veterans, these are rhetorical questions. *Everything*, in their experience, is a death march project. Why do they happen? Because corporations are insane and, as consultant Richard Sargent commented to me, "Corporate insanity is doing the same thing again and again, and each time expecting different results."[1] And why do we participate in such projects? Because, as consultant Dave Kleist observed in an e-mail note, "Death march projects are rarely billed as such, and it takes a lot of work when being hired from the outside to discover if your hiring company is prone to creating death march projects."[2]

If you think the answers to these questions are obvious, feel free to jump to the next chapter. I'm sometimes think they *are* obvious, since most people never ask me what I mean by "death march." But if you're one of the people who has no idea what I'm talking about, or wonder if this is a book about military campaigns from World War II, it may be worth the effort to pause for a moment and contemplate what this is all about.

DEATH MARCH DEFINED

Quite simply, a death march project is one whose "project parameters" exceed the norm by at least 50 percent. In most projects, this means one or more of the following constraints have been imposed upon the project:

- The schedule has been compressed to less than half the amount estimated by a rational estimating process; thus, the project that would normally be expected to take 12 calendar months is now required to deliver its results in six months or less. Because of the competitive pressures of business competition in today's global marketplace, along with the concept of "Internet time" from the dot-com era of the computer industry, this is probably the most common form of death march project.
- The staff has been reduced to less than half the number that would normally be assigned to a project of this size and scope; thus, instead of being given a project team of 10 people, the project manager has been told that only five people are available. This may have come about as a result of someone's naive belief that a new programming language or application development environment will magically double the team's productivity—despite the fact that the team was given no training or practice with the new technology and probably wasn't even consulted about the decision to use the technology in the first place. Unfortunately, it's happening far more often, as this edition of *Death March* is being written in the spring of 2003, because of the ongoing economic recession and the associated cutbacks in IT budgets.
- The budget and associated resources have been cut in half. Again, this is often the result of downsizing and other cost-cutting measures, but it can also result from competitive bidding on a fixed-price contract, where the project manager in a consulting firm is informed by the marketing department that, "the good news is that we won the contract; the bad news is that we had to cut your budget in half in order to beat out the competitors." This kind of constraint often has an immediate impact on the number of project team personnel that can be hired, but the consequences are sometimes a little more subtle—e.g., it may lead to a decision to hire relatively inexpensive, inexperienced junior software developers, rather than higher-cost veterans. And it can lead to a pervasive atmosphere of penny-pinching that makes it impossible for the project manager to order pizza for the project team when they spend the entire weekend in the office working overtime.

- The functionality, features, performance requirements, or other technical aspects of the project are twice what they would be under normal circumstances. Thus, the project team may have been told that it needs to squeeze twice as many features into a fixed amount of RAM or disk space as its competitor; or it may have been told its system has to handle twice the volume of transactions that any comparable system has ever accomplished. The performance constraints may or may not lead to a death march project; after all, we can always take advantage of cheaper, faster hardware, and we can always search for a more clever algorithm or design approach to accomplish the improved performance. But doubling the functionality—i.e., the available features—usually means doubling the amount of work that has to be carried out; *that* does lead to a death march project.

The immediate consequence of these constraints, in most organizations, is to ask the project team to work twice as hard and/or twice as many hours per week as would be expected in a "normal" project. Thus, if the normal work-week is 40 hours, then a death march project team is often found working 14-hour days, six days a week. Naturally, the tension and pressure escalate in such environments, so that the death march team operates as if it is on a steady diet of Jolt cola.

Another way to characterize such projects is as follows:

> A death march project is one for which an unbiased, objective risk assessment (which includes an assessment of technical risks, personnel risks, legal risks, political risks, etc.) determines that the likelihood of failure is ≥ 50 percent.

Of course, even a project without the schedule, staff, budget, or functionality constraints described above could have a high risk of failure—e.g., because of hostile politics between the IT department and the user community. But most commonly, the reason for the high risk assessment is a combination of the constraints described above.

CATEGORIES OF DEATH MARCH PROJECTS

Not all death march projects are the same; not only do they involve different combinations of schedule, staff, budget, and functionality constraints, but they come in different sizes, shapes, and flavors.

In my experience, *size* is the most important characteristic that distinguishes one death march project from another. Consider four different ranges of projects:

- *Small-scale projects*—the team consists of three to six people who are working against nearly impossible odds to finish a project in three to six months.
- *Medium-size projects*—the team consists of 20–30 people, who are involved in a project expected to take one to two years.
- *Large-scale projects*—the project consists of 100–300 people, and the project schedule is three to five years.
- *Mind-boggling projects*—the project has an army of 1,000–2,000 or more (including, in many cases, consultants and subcontractors), and the project is expected to last seven to ten years.

For several reasons, the small-scale death march projects are the most common in the organizations that I visit around the world today; happily, they have the greatest chance of succeeding. A tightly knit group of three to six people is more likely to stick together through thick and thin, and this same group of highly motivated people is more likely to be willing and able to sacrifice their personal lives (as well as risk their health!) for three to six months if they know that the regimen of long nights, wasted weekends, and postponed vacations will come to an end in a matter of months.

The odds of successful completion drop noticeably with the medium-size projects and disappear almost completely with the large-scale projects. With larger numbers of people involved, it's more difficult to maintain a sense of cohesive team spirit; and the statistical odds of someone quitting, being run over by a beer truck, or succumbing to the various perils of modern society increase rapidly. What's crucial here is not just the number of people involved, but the time-scale: Working 80-hour weeks for six months may be tolerable, but doing it for two years is much more likely to cause problems.

As for the "mind-boggling" death march projects: One wonders why they exist at all. Perhaps the systems development efforts associated with the NASA project that landed a man on the moon in 1969 could be considered a successful example of a death march project; but the vast majority of such projects are doomed from the beginning. Fortunately, most senior managers have figured this out, and most large organizations (which are the only ones that could afford them in the first place!) have banned all such projects. Government organizations, alas, still embark upon them from time to time; along with potentially unlimited budgets with which to tackle truly mind-boggling projects, appeals to patriotic notions (e.g., "national security" in the post-9/11 era) may be sufficient to blind senior management to the futility of the task they've been given.

In addition to project size, it's also useful to characterize the "degree" of a death march project by such criteria as the number of user-organizations that are involved. Things are hard enough when the project team has to satisfy only one user or one

group of homogeneous users within a single department. Enterprise-wide projects are usually an order of magnitude more difficult, simply because of the politics and communication problems involved in cross-functional activities of any kind. As a result, the systems development projects associated with business re-engineering projects often degenerate into a death march; even though the development effort is modest in terms of hardware and software effort, the political battles can paralyze the entire organization and cause endless frustration for the project team.

Finally, we can distinguish between projects that are incredibly difficult and those that are fundamentally impossible. As John Boddie, author of *Crunch Mode*, points out,

> The combination of excellent technical staff, superb management, outstanding designers, and intelligent, committed customers is not enough to guarantee success for a crunch-mode project. There really are such things as impossible projects. New ones are started every day. Most impossible projects can be recognized as such early in the development cycle. There seem to be two major types: "poorly understood systems" and "very complex systems."[3]

This still leaves unanswered the questions of why a rational organization would embark upon such a project and why a rational project manager or technical person would agree to participate in such a project. We'll deal with those questions below.

WHY DO DEATH MARCH PROJECTS HAPPEN?

If you think about what goes on in your organization, it's not difficult to understand why death march projects occur. As Scott Adams, author of the incredibly popular "Dilbert" cartoons, points out,

> When I first started hearing these stories [about irrational corporate behavior] I was puzzled, but after careful analysis I have developed a sophisticated theory to explain the existence of this bizarre workplace behavior. People are idiots.
>
> Including me. Everyone is an idiot, not just the people with low SAT scores. The only difference among us is that we're idiots about different things at different times. No matter how smart you are, you spend much of your day being an idiot.[4]

Perhaps it's too depressing to imagine that you're an idiot and that you're surrounded by (and managed by!) idiots. Or perhaps you consider it an insult that someone would even make such a suggestion. In that case, Table 1.1 shows a more detailed list of reasons for the occurrence of death march projects:

Table 1.1 *Reasons for Death March Projects*

Politics, politics, politics
Naïve and/or devious promises made by marketing, senior executive, inexperienced project managers, etc.
Naive optimism of youth: "We can do it over the weekend"
The "startup" mentality of fledgling entrepreneurial companies
The "Marine Corps" mentality: *Real* programmers don't need sleep
Intense competition caused by globalization of markets
Intense competition caused by the appearance of new technologies
Intense pressure caused by unexpected government regulations
Unexpected and/or unplanned crises—e.g., your hardware/software vendor just went bankrupt, or your three best programmers just died of bubonic plague

While the items in Table 1.1 may seem obvious, they do need to be discussed—because they may indicate that your death march project is *so* crazy and irrational that it's not worth participating at all. Indeed, even without an explicit rationale of the sort shown in Table 1.1, you should seriously consider whether you want to spend the next several months (or years) attached to such a project; we'll discuss that separately later in this chapter.

Politics, politics, politics

Many software developers vow that they won't get involved in politics—partly because they've learned that they're not very good at playing political games, but also because they feel that everything about politics is repugnant. Alas, politics cannot be avoided: As soon as two or more people participate in some joint enterprise, politics are involved.

But when politics becomes the dominant "driving force" in a large, complex project, the project is likely to degenerate into a death march. Remember my definition of a death march project: It's one where the schedule, budget, staff, or resources are 50–100 percent less than what they should be. *Why are these constraints being placed on the project?* There are many possible explanations, as we'll see in the discussion below; but in many cases, the answer is simply, "Politics." It may be a power struggle between two ambitious vice presidents in your organization, or the project

may have been set up to fail as a form of revenge upon some manager who stepped on the wrong toes at the wrong time. The possibilities are endless.

There is little chance that you'll get the politicians to admit what's going on; however, if you're a technical staff member, it's not unreasonable to ask your project manager whether the entire death march project is a political sham. Even if you don't like politics, and even if you think you're a political neophyte, listen carefully to the answer your manager gives you. You're not stupid, and you're not *that* naive. If you have a sixth sense that there's some ugly politics dominating the entire project, chances are you're right; and if your immediate supervisor gives a naïve, ambiguous, or thoroughly unconvincing answer to your questions, you should draw your own conclusions. To put it another way: If your project sounds like something straight out of a "Dilbert" cartoon, chances are that it will be the kind of death march project in which no rational person would want to be involved.

What if your manager openly agrees with you? What if he or she says, "Yes, this whole project is nothing more than a bitter power struggle between Vice Presidents Smith and Jones…"? If that's the case, then why on earth is your manager participating in the project? As we'll see in the section about why people participate in death march projects, there may be many reasons; but your manager's reasons are not necessarily *your* reasons. The existence of ugly politics doesn't necessarily mean that you should abandon the project or quit your job right away, but it does mean that you should keep your own priorities, objectives, and sense of ethics separate from what's going on in the project—for it's quite likely that many of the decisions that take place (beginning with the schedule/budget/resource decisions that defined the project as a death march in the beginning!) are not being made with the best interests of the end-user or the enterprise in mind. If the project succeeds at all, it's likely to be an accident—or it may be because the intended victim (who may be your project manager, but may also be a manager several levels above your immediate manager) is a more clever politician than the opposition counted on.

Naive promises made by marketing, senior executives, naive project managers, and so on

Naiveté is often associated with inexperience; thus, it's common to see unrealistic commitments made by people who have no idea how much time or effort will be required to build the system they want. In the extreme case, this can lead to what my friend and colleague Tom DeMarco calls "hysterical optimism": Everyone in the organization desperately wants to believe that a complex project, which has never been attempted before but which has been realistically estimated to require three calendar years of effort, can somehow be finished in nine months.

The naiveté and optimism extends to the technical staff, too, as we'll see below. But for the moment, let's assume that it's your manager, or your marketing department, or the end-user who is responsible for the naively optimistic schedule or budget. The question is: How will they react when it eventually becomes clear that the initial commitments *were* optimistic? Will they extend the schedule, increase the budget, and calmly agree that things are tougher than they had imagined? Will they thank you for the heroic efforts you and your colleagues have made up to that point? If so, then it may turn out that the most important thing you need to do is avoid the classical waterfall life cycle, so that a realistic assessment of schedule, budget, and resources can be made after the first prototype version of the system is delivered.[5]

But in many death march projects, this kind of rational mid-course correction isn't possible. This can happen, of course, if a senior manager makes a naive promise to the customer, and then feels that the commitment has to be honored—no matter what. In the worst case, the person making the commitment knows full well what's going on: It's particularly apparent when the marketing manager confesses to the project manager over a beer after the celebrations accompanying a new contract from some gullible client, "Well, we wouldn't have gotten this contract if we told the client how long it will *really* take; after all, we knew that our competitors would be coming with some really aggressive proposals. And besides, you guys always pad your schedules and budgets anyway, don't you?"

The last comment is especially onerous if it comes from your boss, or from some manager two or three levels above you. Obviously, it suggests that the entire process of estimating schedules and budgets is a negotiating game, which we'll discuss in more detail in Chapter 3. But there is also likely to be some degree of naiveté, for the unspoken implication in your manager's complaint about "padding" the schedule and budget is that you *could* finish the death march project in time to meet the ridiculous deadline that has been imposed upon you. On the other hand, it could have something to do with the "Marine Corps" mindset that I'll discuss in that upcoming section. Similarly, the commitment to a ridiculous schedule and budget by the marketing department could turn out to be another form of politics, discussed earlier; that is, the marketing representative probably doesn't care whether the schedule and budget he proposed is ridiculous or not, because his primary objective is his sales commission, or meeting his quota, or pleasing his boss, etc.

Assume for the moment that the death march project has been created as a result of "pure" naiveté, absent of politics or other malicious influences. The question is: What should you do about it? As noted above, a key question is the probability that the decision-makers will revise their budgets and schedules when it becomes apparent that the original commitments can't be met. This is difficult to predict in advance, though it wouldn't hurt to check around and see what has happened to other death march projects in similar situations. (If this is the first such project that has ever occurred in your company, then you really are in uncharted territory!)

If you have the strong impression—either from your political instincts, or from the experiences from previous projects in your organizations—that management will hold fast to its original budget and schedule, no matter how much of a "denial of reality" is involved, then you need to make a much more fundamental decision about whether to proceed or not. Some of this involves the extent to which you can negotiate other aspects of your project—e.g., the technical staff that will be assigned to the project—which we'll discuss in Chapter 2.

Naive optimism of youth: "We can do it over the weekend"

Though management is a convenient scapegoat for many of the idiotic decisions associated with death march projects, the technical staff is not entirely blameless. Indeed, in many cases senior management will happily admit its naiveté and lack of experience with the process of estimating and scheduling complex projects. "How long do you think it will take?" a vice-president will ask the technical hot-shot, who may have been promoted to the rank of first-level supervisor just last week.

And if the technical hot-shot is ambitious and filled with youthful optimism (which usually borders on the teenage delusions of immortality, omnipotence, and omniscience) the answer is likely to be, "No problem! We can probably knock it out over the weekend!" A really good software engineer—well, "hacker" might be a more appropriate description here—is firmly convinced that he or she can develop *any* system in a weekend. Minor details such as documentation, error-handling, editing of user inputs, and testing are so boring that they don't count.

If you're the naively optimistic software engineer responsible for making the death march estimate, chances are that you don't even know what you're doing. You probably read the paragraph above, bristled at the apparent insult, and muttered, "Damn right! I really *can* build any system over the weekend!" God bless you; maybe you'll succeed. In any case, nothing that you hear from an old fart like me is likely to change your mind.

But if you are a battle-scarred veteran, and you can see that you're about to be roped into a death march project because some naive young technical manager has made a ridiculously optimistic commitment regarding the project's schedule, budget, and resources—what should you do? The best advice, I think, is "Run!" When such technical managers realize that they are in over their heads, they often collapse into truly irrational behavior or paralysis. In most cases, they haven't dealt with anything before that was so big and complex that it couldn't be overwhelmed by sheer cleverness or brute force (e.g., 48 hours of nonstop coding over the weekend). In any case,

they're certainly not in the mood to hear you say, "I told you so!" as their project falls behind schedule.

The "startup" mentality of fledgling entrepreneurial companies

I've not only watched this occur, I've participated in such projects and have been responsible for initiating them in several cases. Shortly after the first edition of this book was published, it appeared that any startup company with "dot-com" in its corporate name or product name could get more venture capital than it knew what to do with. But as became clear to the venture capitalists and hopeful investors, startup organizations are generally understaffed, underfinanced, undermanaged, and hysterically optimistic about their chances of success. They have to be: A cautious, conservative manager would never dream of starting a new company without tons of careful planning and a large bank account to deal with unforeseen contingencies.[6]

So, almost by definition, a large percentage of the projects associated with startup companies comprises death march projects. And a large percentage of these projects will fail; a large percentage of the companies will fail with them. *C'est la vie*—that's what high-tech capitalism is all about (not only in the United States, but all over the world). Having been raised in this culture all my life, I think it's perfectly normal; my attitude is also biased by the fact that I've been lucky enough to succeed in a few such ventures. Indeed, this scenario is often one of the *positive* reasons for embarking upon a death march project, as I'll discuss in more detail in that section below.

But not everyone is familiar with the culture and environment of a corporate startup. If you've spent the past 20 years of your career working with brain-dead COBOL zombies in a moribund government agency (or in, for that matter, most banks or insurance companies or telephone companies) and you've just taken a job with a startup firm because you were downsized, outsourced, or given an early retirement, then you probably have little or no idea what you're in for. Death march projects occur in big companies, too, but they're often staffed by cast extras from *Night of the Living Dead*. The environment is completely different in startup-company death march projects; it's like a rush of pure adrenaline.

At the same time, the startup companies often suffer from the kind of naive optimism discussed earlier. Many startup companies are founded by technical hotshots convinced that their new technology will make them richer than Bill Gates; other such companies are founded by marketing wizards who are certain they can sell Internet-enabled refrigerators to gullible Eskimos. Optimism is important in any startup venture, and the success of the corporate venture may depend on doing what nobody has ever been able to do before; but even an aggressive, optimistic startup company has to

obey basic laws of physics and mathematics. If you get involved in a startup-company death march project, check to see whether there is some kind of plan for success, or whether the whole venture is based on wishful dreaming.

The "Marine Corps" mentality: *Real* programmers don't need sleep

Startup companies are sometimes vulnerable to the "Marine Corps" syndrome, but I've seen it most often in the consulting organizations like EDS and the Big-X[7] accounting firms. It may reflect the personality of the corporate founder(s), and it may reflect the corporate culture in its earlier days—the corporate behavior at Microsoft, for example, has often been attributed to these factors. In essence, you'll be told by the appropriate manager, "*Every* project is like this, because that's how we do things around here. It works, we're successful, and we're damn proud of it. If you can't handle it, then you don't belong here."

Whether an attitude like this is civilized, humane, or right is a topic for a separate discussion. Indeed, the question of whether it's even successful is something to be discussed elsewhere; the important thing is to realize that it's *deliberate*, not accidental. If you're a martyr or a revolutionary, you might decide to attack the corporate culture, but the chances are that you won't succeed. It's quite possible that there will be some negative long-term consequences of the overall death march culture, for example, the best people may slowly drift away and the company may fail. But when it comes to *this* death march project, there's no point questioning why it has been set up with a nearly impossible schedule and budget. As the prototypical manager of such companies says, "If you can't handle it, then you don't belong here."

Sometimes there's an official rationale for such corporate behavior—for example, "We compete in a tough marketplace, and all of our competitors are just as smart as we are. The only way we succeed is to work twice as hard." And sometimes the death march projects are set up to weed out the younger (weaker) junior employees, so that only the survivors of the death march projects will reach the exalted status of "partner" or "vice president." Whatever the rationale, it's usually fairly consistent; there's not much point complaining about it for the sake of a single project.

That doesn't necessarily mean that you should accept an assignment on such a project; after all, just because every other project within the organization is a death march doesn't necessarily mean that yours will succeed, or that you will survive. It simply means that the decision to create such a project has an understandable origin.

Intense competition caused by globalization of markets

Organizations that might not have tolerated death march projects in the past are sometimes forced to do so in the 21st century, simply because of the increased level of competition associated with the global marketplace. The secondary factors here are the Internet and the Web, as well as governmental decisions to open previously protected markets or eliminate tariffs and quotas.

For some organizations, this is not a new phenomenon; the automobile and electronics industries, for example, have been facing stiff competition since the 1970s. But for other organizations, the appearance of European or Asian competitors in the North American marketplace can come as a rude shock. Once senior management has accepted the reality of serious competition, it may decide to embark upon a variety of radical moves, ranging from downsizing to outsourcing the entire IT organization to the other side of the world; it may also decide to compete head-on with a new product or service that requires a new, ambitious system to support it. *Voila!* A death march project has begun.

A relatively recent version of this globalization phenomenon is the outsourcing of software development projects to "offshore" organizations in India, China, Russia, or other countries. Having visited software organizations in several of these countries, I can testify that such organizations are typically *not* "sweat shops" that engage in the kind of death march projects that require their programmers to work 16 hours a day, seven days a week. Nevertheless, the presence of lower cost offshore programming resources may cause domestic software companies and IT departments to respond by making their higher priced programmers in the United States work longer hours. As one reader suggested to me in a recent email message:

"I see things getting worse. With the trend of outsourcing software development work overseas to dramatically cut labor costs, the remaining domestic software houses will suffer tremendous competitive pricing pressures. The only way to compete will be to get your product on the market first and cut costs. 'Deathmarch' may become standard procedure for many companies. An improving economy won't change these market realities."[8]

Intense competition caused by the appearance of new technologies

Competition from an expanded marketplace is often perceived as a defensive issue, but it can also be perceived as an aggressive, proactive opportunity—"If we build this new system, with double-byte characters, then we can offer our company's products

for sale in Japan." Similarly, the introduction of radically improved technology may cause a defensive response from a company that was reasonably happy with products built around an older technology; or it may lead to a proactive decision to utilize the new technology for competitive advantage. At the time this book was being written, technologies such as wireless computing and Web services were obvious examples of this phenomenon; but the amazing thing about our industry is that new examples appear every couple of years.

If the corporate response to the new-technology situation is essentially defensive in nature, then the death march project may be one that seeks to exploit the company's *old* technology far beyond its normal limits. Thus, if the organization has too much invested in the old technology (and the infrastructure surrounding it) to abandon it entirely, it may embark upon a rewrite of its old systems, with demands that the programmers find a way to make it ten times faster and sexier.

But many death march projects in this category are the ones that involve first-time usage of the new technologies. Think back to the first client-server projects, object-oriented projects, relational database projects, or Internet/Java projects in your organization; some of them may have been modest experiments to explore the potential benefits of the technology, but some of them were probably created as a competitive response to some other company's introduction of the same technology. And in the latter case, these projects can be huge, as well as being saddled with outrageously aggressive schedules and budgets.

But what really contributes to the death march nature of such projects—beyond the obvious characteristics of size, schedule, and budget—is the attempt to use bleeding-edge technology for an industrial-strength application. Even if the technology is basically usable, it often does not scale up well for large-scale usage; and nobody knows how to exploit its strengths and avoid its weaknesses; and the vendors don't know how to support it properly; and on and on...

While all of this may be perceived as an unpleasant experience by the older technical project team members (the ones who remember the "good old days" of FORTRAN II and assembly language), it's important to remember that the younger technicians and project managers *prefer* these new technologies, precisely because they *are* new. And these are the same folks whom I characterized above as naively optimistic about the schedule and budget constraints within which they're working. Is it any wonder that things degenerate into a death march project, with everyone working late nights and long weekends in order to coax the experimental new technology into some semblance of working order?

Intense pressure caused by unexpected government regulations

As noted above, one of the reasons for death march projects associated with globalization of markets is the decision by governmental authorities to reduce tariffs, eliminate import quotas, or other such decisions to "open" a previously closed market. But this is just one example of governmental influences that can lead to a death march project; deregulation of controlled industries, or privatization of government agencies are two other obvious examples. Indeed, many of the death march projects taking place today around the world are a direct result of a government decision to deregulate the telecommunications industry, the financial services industry, the airline industry, and so on.

However, there are also many instances of *increased* regulatory pressure from governmental authorities—especially in the areas of taxation, reporting of financial details to stock-market authorities, environmental regulations, and so forth. In any kind of democratic society, there's likely to be a great deal of advance notice about such regulations, because the legislative body argues and debates and fusses over details for months or years before the relevant legislation is enacted. But the details often aren't clear until the last moment, and the typical reaction from senior management is to ignore the whole thing until it become an unavoidable reality. And then, boom! Another death march project is created.

The particularly onerous thing about many of these government-mandated death march projects is the deadline: The new system *must* be operational by some arbitrary date such as January 1st, or fines of a million dollars a day will be imposed. There may be an opportunity to ask for an extension or a waiver, but in most cases, the deadline is absolute. And the consequences are usually as dire for the organization as those mentioned above: Layoffs, bankruptcy, or other calamities will occur if the new system isn't finished on time.

Notice that in projects like these, technology is usually not the issue; what characterizes the projects as death march in nature is the aggressive timetable. Of course, management sometimes complicates the situation by understaffing the project or hobbling it with an inadequate budget.

Unexpected and/or unplanned crises

Your two best programmers have just marched into your office to inform you that (a) they're getting married, (b) they're joining a missionary group building hospitals in the jungles of Africa, and (c) today is their last day on the job. Or your network services manager calls you to say that your vendor has just gone bankrupt and you'll have to reprogram everything in the next 30 days in order to use another vendor's network

protocol. Or your legal department calls to say that the company has been sued by ten zillion dollars because the company is not in compliance with Sub-paragraph 13(b) or Regulation Q of some arcane tax code that nobody even knew about.

Of course, you could argue that, in a well-managed company, the impending departure of your two best programmers would have been anticipated and planned for; and you wouldn't have been so silly as to be wholly dependent on one telecommunications vendor; and management would have had the foresight to check into the details of Regulation Q. Such crises, according to the purists, are the result of poor planning and poor management; an "unplanned crisis" is therefore an oxymoron.

Perhaps so; but as a practical matter, it's becoming increasingly difficult to anticipate and plan for all the crazy things that can happen in the business world today. For better or worse, we live in a world of chaos, and death march projects are a natural consequence of such chaos.[9] Indeed, even if we have a general idea that chaotic things *could* occur in the future, we may still have to respond to them in a death march fashion. Everyone in the vicinity of the San Andreas fault in California, for example, knows that a truly massive earthquake will occur sooner or later; but that won't prevent a rash of death march projects from starting up the day after the "big one" drops the western half of the state into the Pacific Ocean.

Indeed, even when we know *precisely* when a crisis will occur, it often leads to a death march project—because management's tendency is to avoid dealing with the situation until the last possible moment. How else could we explain the panic that crept into many IT organizations as the Y2K problem loomed ahead of them? We had known for a *long* time that January 1, 2000, was coming, and it was obviously a deadline that could not be postponed. We knew precisely what the nature of the problem was, and it didn't require newfangled technologies such as Java. So why did so many Y2K death march projects get launched in 1998 and 1999?

In any case, unforeseen crises can lead to all kinds of death march projects. In the worst case, they create projects for which the deadline is "yesterday, if not sooner"—because the crisis has already occurred and things will continue to worsen until a new system is installed to cope with the problem. In other cases, such as the unplanned departure of key project personnel, it can turn an otherwise rational project into a death march exercise because of the shortage of manpower and the loss of key intellectual resources.

For various reasons, these often turn out to be the worst kind of death march projects, *because nobody anticipated that they would turn out this way.* For the Marine Corps situation discussed above, there are no surprises: Everyone knows from the first day of the project that this one, like all previous projects, is going to require extraordinary effort. And for the startup companies, the death march project is anticipated with excitement; not only will it be exciting and challenging, but its success could make everyone rich.

WHY DO PEOPLE PARTICIPATE IN DEATH MARCH PROJECTS?

The theme of the discussion in the previous project is that organizations create and/or tolerate death march projects for a number of reasons. We may agree or disagree with those reasons, and we may sympathize with the ones caused by truly unexpected crises—but ultimately, as individuals, we have to accept them as a fact of life.

But that doesn't mean we have to participate in them. Most of this book presumes that you *will* participate in a death march project, though I will specifically suggest that you resign under certain circumstances. But the best time to do so, in most cases, is at the beginning. When told that you have been assigned to such a project (either as a leader or a technical staff member), you should consider saying, "No, thanks! I'll pass on this one." If that's not an acceptable response within your corporate culture, you almost always have the option of saying, "No, thanks! I quit!"

Obviously, some developers—and probably a larger number of managers—will argue that this is not a practical choice for them. I'll discuss this in more detail below, but for now it's sufficient to note that it's one of several possible "negative" reasons for participating in a death march project; it may not be fun, but perhaps it's not as bad as the alternatives. On the other hand, some developers (and some managers) *gladly* sign up for such projects; aside from the issue of naive optimism discussed above, why would any rational person volunteer to participate in a project that's likely to require 14-hour days, seven-day weeks, and a year or two of postponed vacations?

The most common reasons are summarized in Table 1.2; I'll discuss them in more detail below.

Table 1.2 *Reasons for Participating in Death March Projects*

The risks are high, but so are the rewards
The "Mt. Everest" syndrome
The "buzz" of working intensely with other committed people
The naiveté and optimism of youth
The alternative is unemployment
It's required in order to be considered for future advancement
The alternative is bankruptcy or some other calamity
It's an opportunity to escape the "normal" bureaucracy
Revenge

By the way, this is not intended as a complete list; Kevin Huigens[10] asked his project team to do a little brainstorming at one of their recent staff meetings, and they came up with the following list (Table 1.3):

Table 1.3 More Reasons for Participating in Death March Projects

Everybody wants to feel wanted	Perceived opportunity
Perceived money gain	Can't afford to lose job
Brought in from the outside to lead the project	Willing suspension of disbelief
Don't care whether project fails, get to work with cool technology	On-the-job-training on new technology
Eternal optimism	Challenge
Plain stupidity	Chance to prove yourself
To get the job done	It's the only project
Your friend is running the project	Your brother is running the project (It'd take more than friendship)
Your boss said so	You have no other life
Nothing better to do	Stock options
Existing pay vs. expectation of raise	Love is blind
Resumé-building	Ignorance
Camaraderie	Expectations for how long it will take are too low

Of course, all of this assumes that you know in advance that it *is* a death march project. As consultant Dave Kleist[11] observed to me, that's not always so easy when you're interviewing for a new job:

> ...it's rarely printed as part of the want ad. Not much sense in saying, "Are you interested in working incredible hours for no additional benefit beyond your hiring salary?"... Seriously, death march projects are rarely billed as such, and it takes a lot of work when being hired from the outside to discover if your hiring company is prone to creating death march projects.

And, as Steve Benting[12] pointed out to me, sometimes you get taken by surprise:

> *...it seems to be a well-thought-out project this time. You've got someone leading who has a real sponsor in management, the project plan appears to be solid, the people involved all appear to be good. Hell, you want to work on this thing. Then it collapses because your sponsor gets taken out in a political struggle, the project plan turns out to be built on assumptions that are incorrect, and one or two key people turn out to be flaky. You can learn to watch out for them, but sometimes you misjudge. And you don't want to believe that it's happening again.*

The risks are high, but so are the rewards

The startup-company scenario discussed in an earlier section is a good example of this situation. If you tell project team members that the success of their project will mean the company can go public, and that their stock options will make them instant millionaires, they'll happily work until they drop from exhaustion. They realize—at least in an intellectual way—that there are risks associated with the venture; but since many of them still believe that they're immortal and omnipotent, they don't pay much attention to the risks.

Indeed, considering the influences of Western culture (especially in the United States), it's not at all surprising to see young software developers voluntarily sign up for death march projects. We've been told in countless ways that the success of movie stars, rock singers, sports heroes, and Olympic athletes, as well as business executives and software entrepreneurs, depends largely on tireless energy, enormous commitment, long hours, and personal sacrifice. We never hear about the guile and duplicity, the shady deals and illegal activities, that are sometimes associated with success. And we rarely hear anything about luck and the importance of being in the right place at the right time; Bill Gates, for example, certainly exhibits all of the textbook characteristics of a successful business executive, but if a group of IBM executives hadn't shown up in Seattle in 1980 to look for a PC operating system, and if Gates hadn't been available when IBM was unable to meet with its originally intended OS contractor ... well, who knows where Microsoft would be today?

And one more thing: We don't hear enough about the real consequences of the "sacrifice" that a death march project usually requires—sacrifices, that is, in the area of personal health, mental health, and personal relationships. None of this is likely to matter very much to a 22-year-old technical person; and it often doesn't matter to the introverted, anti-social people who are attracted to the computer field. On the other hand, it's small wonder that you'll find fewer people in their mid-40s and 50s volunteering for death march projects; not only have they learned that most such projects

really *are* doomed to fail, but they've also learned (usually the hard way!) that it's not worth sacrificing their marriage and their relationship with their children.

Ultimately, this is a personal choice, based on personal values; I'm no position to tell anyone else what's right or wrong. I should emphasize, though, that I'm not as negative as one might think from the comments above. Though I'm much less naive than I was when I entered the computer field in the mid-1960s, I'm still attracted by entrepreneurial opportunities. Show me a sufficiently exciting risk/reward formula, and I'll sign up for yet another death march...

By the way, sometimes the rewards are psychological, not financial. As Sharon Marsh Roberts[13] observed to me,

> The "heros" are needed, wanted, desired. They are certain of their place in history, if only they can keep this project from outright sinking under its own weight.
>
> The same people take on EMT work and enjoy firefighting (literally). If you only win once in ten times, but everybody else lost all ten, wouldn't you be a hero, too?

And as Paul Neuhardt[14] put it,

> For me, it was ego, pure and simple. They told me that they just *knew* I could help prevent the project from becoming a death march. I was made the "technical project manager," given ego boosts on a regular basis, then hung out to dry along with the rest of the team. Left, right, left, right, left, *plop*!

The "Mt. Everest" syndrome

Why do people climb dangerous peaks like Mt. Everest, despite the pain and the risk? Because it's there. Why do people run a marathon and drive themselves to the point of physical collapse in triathlons? Because of the challenge. It's all the more exciting if the challenge is one that has never yet been successfully accomplished; of the six billion people on the planet, for example, only one can stand before us and say, "I was the first to walk on the moon." Some may think it's crazy, egotistical, and selfish to even try; but others are willing to brave the odds and deal with horrendous obstacles for the private thrill and the public glory of succeeding. As consultant Al Christians[15] remarked to me in a recent e-mail note,

> I am somehow prompted to reply "testosterone," which is about the same as "because it's there." There are plenty of jobs that raise the 'why?' question. Underground mining, cowboying, logging, smoke jumping, jet fighting, submarining, even high-rise window washing all have serious drawbacks far beyond what you describe for software

> projects, and yet all these have practitioners whose sense of self is linked to their profession.

And so it is with death march software projects. I had the chance to visit the original Macintosh project in the fall of 1983, a few months before the product was officially unveiled, and was humbled by the intensity of the team's commitment to its challenge. In addition to whatever other reasons the members might have had for working long hours and dealing with Steve Jobs' megalomaniacal ego, the team was utterly convinced (partly as a result of Jobs' charisma) that the Macintosh would revolutionize personal computing. They were lucky—they turned out to be right.

From this perspective, even the death march projects that fail can be *noble* failures. Countless projects in Silicon Valley have fallen into this category, often after burning tens of millions of dollars of venture capital; while most of the dot-com startups had no substance at all, some of them really did seem awe-inspiring and revolutionary until they went into bankruptcy. But even though they failed so badly that entire companies went bankrupt, and though they caused divorces, ulcers, and nervous breakdowns—even though they did all of this and more—the people who worked on those projects still speak of their experiences in hushed tones. "I worked on the Toys.com system," a grizzled veteran will tell her awe-struck apprentice. "Now *that* was a revolutionary piece of software!"

Though it may never reach the front pages of *Computerworld*, there are also numerous death march projects with lofty ambitions buried within large organizations—with application developers signing up gladly because the "corporate Mt. Everest" seems such a worthy challenge. Sometimes these projects fail because the marketplace or the corporate end-users don't want and don't need the glorious, revolutionary systems being developed; and sometimes they fail because the project team bit off more than it could chew and promised more than it could deliver.

There are two things to watch for if you find yourself being swept up in the hysteria of a Mt. Everest-style death march project. First, watch out for the projects that are predetermined failures. Suppose, for example, that someone told you that you could be on the first manned mission to Mars, and that you would even have the honor to be the first person to plant a foot on Martian soil. "Of course," your project manager goes on to say, "you won't have any oxygen tanks, because we won't have enough room on the space craft for all that extra weight. So it's a guaranteed fact that you're going to die—but think of the honor and the glory!" I'll discuss these projects in more detail in Chapter 3, under the heading of "kamikaze" projects; for now, the scenario speaks for itself.

The second thing to watch out for is that the challenge being described by your corporate management (or by the entrepreneurial founder of your software company) may not turn out to be such a big deal after all. This is a particularly insidious danger if the challenge is technical in nature, for example, "We'll be the first people on earth

to put an operating system with the functionality of Windows XP into 128K of ROM!" Granted, that would be an amazing technical accomplishment—but so what?

It's a good idea to ask the "So what?" question two or three times—in other words, *continue* asking the question in response to each successive answer you get from your corporate management. If the response to the Windows XP scenario posed above is, "Well, that means we could put *all* of Windows XP onto your wrist watch!", then ask, "So what?" again. In some cases, the answers will eventually become silly and you'll be jerked back into the real world. For example, suppose your boss answers the second "So what?" question above with the explanation, "Well, if we can also squeeze in a full voice-recognition system, that means you'll be able to write Visual C++ programs while you're walking down the street, by talking to your wrist-watch!"

No doubt there are a few dozen programmers who would say, "*Cool!*" and volunteer to spend the next three years of their lives on such a project. The fact that nobody in his right mind would ever use such a project is irrelevant to them; the technical challenge is a sufficient justification. Putting Windows XP, full voice recognition, and Visual C++ into 128K of ROM would give you supreme bragging rights at any convention of hackers and programmers; if that's what you live for, then by all means go ahead and sign up for the project.

It's also a good idea to explain the project in simplified non-technical terms to your spouse, or your "significant other," or your parents—or, even better, your children. *They* will ask the "So what?" question, without the burden of being tempted by the technical challenge. "You're going to give up your nights and your weekends and your vacations for the next two years in order to put Windows XP on a wrist-watch?" your spouse will ask incredulously. And your children will ask, "Yeah, but Mom/Dad, why would anyone *do* that?" If you can answer those questions without feeling utterly foolish, then you can sign up for the project with a clear conscience.

A worse form of Mt. Everest project is the one where the challenge matters *enormously* to corporate management, but not at all to anyone who stops and thinks about the situation for a second. "Why are we signing up for this death march project, boss?" the young programmer asks innocently. "Because," the boss thunders righteously, "it will increase our corporate earnings per share by a full 3.14159 cents!!" This means that if the programmer was lucky enough to have options on a hundred shares of the company's stock, and if every penny of increased earnings was paid out in dividends, the programmer would get a whopping $3.14; and if Wall Street traders got so excited by all of this that they boosted the price of the stock by a dollar, the programmer's net worth would increase by another hundred dollars. "And what else would I have to show for the thousands of hours of overtime you're asking me to sign up for, boss?" the young programmer asks. The boss is silent, for he knows that the honest answer is: *nothing*. The project is intrinsically boring, involves no interesting technology, and has a 75-percent chance of failing anyway.

But the very worst death march projects, in my opinion, are the ones where the boss deliberately manipulates the innocent project team into believing that a Mt. Everest-style challenge is involved, when the boss knows full well that it's not. Imagine the project team member who asks, "Why are we trying to build this batch, mainframe, COBOL airline reservation system in six months, boss?" The boss is likely to respond, "Because *nobody* in the entire airline industry has ever tried to do it in less than three years before!" I suppose that one could argue that there *is* a technical challenge involved in creating a batch-mode airline reservation system, but it's not the kind of technology that I would want on my resumé in the 21st century. In any case, what makes this scenario a death march project is not the technical challenge, but the ridiculous schedule imposed on the project. Why is the project manager doing it? Who knows—but it's not likely to be the sort of thing you'll want to brag about to your friends a year from now.

The naiveté and optimism of youth

Ours is a young industry, and many of the most exciting and challenging projects are being performed by, and led by, people in their twenties. It's not at all uncommon to see death march projects where the entire technical team is under 25, and where the majority are in the 21–23 age range. As such, they remind me of the fighter pilots and bombing crews recruited by the Air Force in the Gulf War: like Tom Cruise's character in *Top Gun*, they are young, idealistic, and absolutely convinced that they could do *anything*. As David Maxwell[16] put it:

> …projects are like a marriage. We tend to start off naively and full of hopes and slowly as reality sets in, we have to re-assess our expectancies within the relationship. There are many reasons apart from *logic* that attract people together into a marriage and it is the same case with projects. With a predominantly youthful workforce, it is likely that the "death march" project will occur again and again as a training ground for managers and developers alike. And, as I know from personal experience, I often repeat the same mistake many times before the penny drops.

Indeed, it's this supreme confidence that enables a death march team to succeed where traditional project teams have failed before. Part of the folklore of our industry is that the most successful products—ranging from Lotus 1-2-3 to the original Netscape Navigator Web browser—have been developed by a handful of people under conditions that no "rational" project team would have accepted. When these projects succeed, they often bring fortune and fame to the project team; and when they fail, they often provide some valuable lessons to everyone involved (though the corporate consequences may still be disastrous!).

It's important to note that the naiveté and optimism of youth are usually combined with enormous energy, single-minded focus, and freedom from such distractions as family relationships. Obviously, youth doesn't have a monopoly on any of this, but it's a lot more common to see a 22-year-old programmer willing and able to focus on the technical demands of a death march project for 100+ hours per week continuously for a year or two, than a 35-year-old programmer with a spouse and two children and a moderate passion for mountain climbing. The young programmer who signs up for a death march—as well as the relatively young project manager who optimistically promises success to the corporate chieftains—is implicitly saying, "Of *course* I'll succeed with this project; I'll overwhelm the obstacles with sheer energy!"

I won't make any value judgments about all of this, because it's pointless. As noted above, ours is an industry that attracts young people, and I don't think that will change in the next few years. I also think it's unlikely that young people will abandon their optimism, energy, and ability to focus single-mindedly on a problem. As for their naiveté ... well, it doesn't help much for battle-scarred veterans to accuse their younger colleagues of this disease.

The alternative is unemployment

Because we *do* have an industry populated by young, optimistic people, and because it's a vibrant industry that has been growing steadily (and sometimes rapidly!) for the past 30-40 years, I'm sometimes surprised to hear this explanation for participation on death march projects.

But even our industry has its downturns and recessions. As this edition of *Death March* was being written in the spring of 2003, the high-tech recession had been underway for roughly three years—with no obvious indication of when it would end. And for those too young to remember, there were also downturns in the early 1990s, the early 1980s, and the mid-1970s.

We're also an industry where rapid change renders some veterans obsolete. Indeed, there has been such enormous change during this decade that our profession—like so many other white-collar professions—has experienced significant downsizing, re-engineering, and outsourcing. Aggregate employment in the software industry may be rising steadily, but we sometimes forget that this means only that the C++ programming jobs are increasing more rapidly than the COBOL jobs are declining. In addition, the large IT shops that have expanded into bureaucracies of several thousand people have been particularly vulnerable to downsizing and outsourcing; senior management may not be ready to reduce the ranks of technical staff, but they're often eliminating the middle managers, administrators, and staff people.

All of this becomes a factor in death march projects. The reason your project team has only half as many people as it should have is that management has cut the entire software organization in half. And the reason that your project schedule is twice as demanding as it should be is that management is attempting re-engineering by edict: It has announced that the entire organization needs to be twice as productive as before, which translates into simple commands of "Work harder! Work faster!"[17]

This is not a book about re-engineering, and I don't want to comment on the re-engineering strategies employed by management. The significant issue here is that many technical staffers and many project managers feel an implied threat when projects are created in this kind of environment: If they don't agree to the death march project parameters, they'll be the ones to lose their job. For the 22-year-old, unmarried programmer, this shouldn't be a problem; for the 35-year-old project supervisor with a family and a mortgage, it can be a more serious problem. And for the 45-year-old programmer whose only skills are COBOL and CICS, it can be a serious problem indeed. Even though we do have a young industry, it's been around long enough that there are even some 55- and 60-year-old programmers who are grimly holding on until their pension is fully vested.

It's also common for middle-aged or older people to find that they're locked into a community, because their spouse has a job in the same town, or their children can't be pulled out of the local schools, or because the prospect of leaving behind aging parents and other family members is too painful. None of this seems a problem when the job market is growing, but anyone who lived in Poughkeepsie, New York, in the early 1990s knows exactly what I'm talking about. People living in Redmond, Washington, could conceivably find themselves faced with the same kind of rude shock five, 10, or 20 years from now.

I'm generally sympathetic to the middle-aged and older software professionals who find themselves in this position, though the re-engineering/downsizing phenomenon has been around long enough that I'm amazed to find technical people who ignore the possibility that it could happen to them. But this, too, is a subject for a different book; I've discussed it at length in *Decline and Fall of the American Programmer* and *Rise and Resurrection of the American Programmer*, and I'll confine my remarks here to the *reality* of such death march projects.

If your company has told you—either explicitly, or by innuendo—that your job will disappear unless you sign up for a project with a ridiculous schedule, budget, and resource allocation, what should you do? Obviously, this depends on your assessment of your financial, physical, emotional, and psychological situation; you also need to accurately assess the situation within your company. In some cases, the real threat is that your promotion, bonus, or salary increase will be withheld if you don't participate; I'll cover this separately below. But even if the threat is termination of employment, big

companies can't usually carry out their threat right away; you may have two or three months before your job disappears, and that may be enough time to find a job elsewhere.

What if the threat is more immediate and blunt? "Sign up for this death march project right now, or pack up your things and get out!" says your boss. It's inconceivable to me that a rational person would choose to work in such an environment, but let's assume the environment had been reasonably friendly until the latest re-engineering craze turned your boss into a raving lunatic. So here you are: Sign, quit, or be fired. What can you do?

If at all possible, my advice is: Quit now, because it's just going to get worse. You may have to live off your savings for a few months, and you may even have to take a pay cut while you gain experience in some newer technology; but chances are you'll be a happier person than if you knuckle under and continue on in a situation that has little or no upside potential. Sometimes you can accomplish this by volunteering for the death march project while simultaneously updating your resumé and starting the job search; however, this can create some ethical dilemmas if you feel that quitting in the middle of the death march project would leave your teammates stranded and helpless.

If you feel that you are truly stuck—because of imminent pension vesting, or because of unmarketable technical skills, or because personal commitments keep you locked into a one-employer town—then you might be tempted to take a more positive approach to the death march project. "By gosh, I'll show them that there's still some bark in this old dog," the middle-aged veteran will say. "I'll show management that I'm still just as good as those young whippersnappers, and we'll get this project done on time!" Your courage and positive outlook are admirable indeed, but just remember one thing: If your death march project succeeds, there will another one. Remember the theme at the beginning of this book: *Death march projects are not the exception, they have become the norm.*

It's required in order to be considered for future advancement

As noted above, there are times when the "invitation" to join a death march project carries with it a threat that future promotions and raises will be contingent upon (a) acceptance, and (b) success in the project. This is often associated with a re-engineering initiative—for example, "The people who lead the Megalith Bank into the 21st century will be the ones who have led us through this incredibly complex and challenging Total System 2020 re-engineering project!" If you find yourself in this situation, remember that politics is a key factor: The people who take credit for the success of the death march project may or may not be the people who participated in it. And the manager who proposes the death march project may be using the re-engineering

"crisis" solely as an opportunity to advance his or her career, with little or no concern for whether the project team members survive in the process.

If you've memorized every word of Machiavelli's *The Prince*, and if you enjoy playing political games, then such death march projects might sound like great fun. But most software professionals haven't seen *The Prince* since their college days (if ever) and, in addition to admitting their political naiveté, they'll also express disgust at the whole concept of politics and enormous disrespect for those who indulge in it. If that's the case, why would anyone sign up for the Megalith Bank's Total System 2020 project? The only plausible answer: because you sincerely believe that it's a one-time death march project, and because you really believe that it will help advance your long-term career within the Megalith Bank. And if you believe this, chances are pretty good that you also believe pigs can fly.

In the majority of cases I've observed, the threat of withholding promotions and raises is part of the "Marine Corps" culture discussed earlier in this chapter. Whether it's right or wrong doesn't matter at this point; what counts is that it's fairly consistent. If you receive such threats on your first death march project, you'll probably get them on your second, third, and fourth. You may have been too innocent or naive to contemplate the long-term implications of such a policy when you first joined the company, but sooner or later it will sink in. There are really only two options in this case: Accept it, or quit.

The alternative is bankruptcy or some other calamity

As mentioned earlier, some death march projects have been caused by the re-engineering, downsizing, and outsourcing decisions made by senior management, which in turn have often been caused by global competition, unexpected government regulations, and so forth. Whatever the cause, the results are the same: The employee signs up for the project because he sincerely believes that the alternative is bankruptcy or some other dire calamity. And the situation is often exacerbated by blunt statements from management that anyone unwilling to participate in the death march should resign forthwith, so that those who remain can concentrate on saving the company.

Again, the issue here is not whether the situation is right or wrong, or whether management should have taken earlier steps to avoid the crisis. The point is that once the crisis has arrived and management has initiated the death march project, you need to make a rational decision about whether or not to participate.

From the discussion in earlier sections of this chapter, you can anticipate my advice here: Step back and ask yourself whether this death march project is a one-time exception or the beginning of an ongoing pattern. Even if you win the battle, your company may have lost the war; indeed, your success with your death march project

may have the ironic consequence of delaying the final demise of the company just long enough to sustain a *second* death march project.

Again, this is a personal decision, and it may be colored by feelings of loyalty, sympathy, or a Hollywood-inspired desire to "win one for the Gipper"—a last hurrah to show the world that you and your company are not going to give up without a fight. And who knows: Maybe a tremendous success with your death march project will turn things around, as appeared to be the case when Borland delivered its Delphi product to the marketplace. None of us has a crystal ball when it comes to predicting the outcome of a death march project, nor can we accurately predict what the consequences of a death march success or failure will really be. Some companies die quickly, others die a long, lingering death; still others are acquired before the terminal rot sets in.

As you consult your own crystal ball, seek advice from as many people as possible, especially from those who have no vested interest in the outcome. You may find some honest, objective managers in your company who will candidly discuss the consequences of the death march failure/success; but you should also remember that the same managers have their own career and paycheck to worry about, and that their ego and political instinct may prevent them from sharing the really vital information you need to make an informed decision.

It's an opportunity to escape the "normal" bureaucracy

Technical staffers and project managers often complain that their corporate bureaucracy stifles productivity and introduces unnecessary delays into the software development process. But the larger the organization, the more entrenched the bureaucracy—especially in organizations where the methodology police enforce strict adherence to SEI-CMM or ISO-9000 processes. Similarly, the human resources department may have elaborate procedures that must be followed before new people can be hired, or before external contractors can be used on a project.

Death march projects often provide the opportunity to circumvent some, if not all, of the bureaucracy—and this is reason enough for frustrated software developers to sign up for such projects. In the extreme case, the effort takes on the characteristics of a "skunk works" project: The project team members move out of the corporate facility into a separate building, where they can carry out their work without the distractions of the normal bureaucracy. But even in a less extreme situation, a death march project can often get permission to use its own tools and programming languages, to try new technologies such as object-oriented programming, and to short-circuit much of the ponderous procedures and documentation that would otherwise be required.

Equally important, the death march project manager is often given far greater latitude when selecting team members than would normally be the case.

In the best case, all of these changes can transform a death march into a civilized experience—that is, the very procedures (and technology and people) that turned the project into a death march have been removed or replaced. And if the death march project is eminently successful, it can serve as a catalyst to make permanent changes to the technology, peopleware, and processes used in other development projects throughout the organization. Conversely, if the death march project fails, it might serve as an affirmation that the "standard" policies aren't that bad after all.

In any case, a situation like this is a perfectly plausible reason for working on a project that might otherwise seem uncivilized. In some organizations, certain software developers make a point of *always* signing up for such projects because it's the only way to avoid getting sucked into the bureaucracy.

Revenge

Revenge may not seem like a rational explanation for working on a death march project, but it's real nonetheless. The success of the death march project might be sufficient to wrest power away from an incompetent vice president, or it might serve to humiliate an obnoxious critic who continually tells you "it can't be done" within the schedule and budget constraints of the death march project. Revenge is a powerful emotion, and it is particularly evident in the senior management ranks of large organizations, where insults are remembered forever, and where crafty politicians will sometimes wait months or years to wreak revenge upon their enemies.

Revenge can be a very powerful personal motivator, but it's usually somewhat more difficult to imbue an entire project team with the emotion. And when it happens, it often creates a situation where the team loses track of the "official" objective of delivering a working system within a specified budget and schedule—after all, their first and highest priority is revenge.

If revenge is *your* motivation, then there's not much for me to say: This is another personal judgment call. But if you're signing up for a project in which it's the manager's revenge, or the team's revenge, fueling the project (and causing it to accept deadline and budget constraints normally unacceptable), then you should be very careful indeed. "The vice president is an idiot," your project manager might tell you, "and if we finish this project in six months, he'll be so humiliated in front of the Board of Directors that he'll have to resign!" Well, that's fine—maybe the VP really is an idiot, but do you really want to sacrifice your personal life for the next two years in order to bring about his demise? After all, the next VP is likely to be just as much an idiot as the last one.

On the other hand, if everyone perceives the Vice President to be the personification of Darth Vader, and if the project manager is seen to be a combination of Luke Skywalker and Yoda, then a death march project can be very invigorating indeed. In this case, the entire project is re-cast into a battle of Good versus Evil, and that's enough to make people accept incredible sacrifices without complaint.

SUMMARY

If the discussion in this chapter seems pessimistic and cynical, remember: It hasn't stopped death march projects from taking place. Companies both large and small are filled with politics and are staffed by managers and technical developers who suffer from hysterical optimism as well as the usual gamut of emotions such as fear, insecurity, arrogance, and naiveté. And the combination of re-engineering, downsizing, outsourcing, and global competition—together with the opportunities provided by new technologies such as object orientation, client-server, and the Internet—suggests to me that death march projects are likely to be a common occurrence for years to come.

And that's the primary point of this chapter. You may not agree with any of the rationales suggested here; you may not like any of the reasons for initiating such projects or joining such projects—but they're real nonetheless. The key point is to recognize and understand your own motivations at the beginning of a death march project, so that you can make a rational decision to join the team or look elsewhere for your next job. Since many of these projects are initiated during periods of great corporate stress and emotion, rational decisions are not as easy as you might think; it's all too easy to be swept away by the emotions of your fellow colleagues or your manager.

It's worth noting that some observers have a more optimistic outlook than what I've suggested; they believe that there will actually be *fewer* death march projects during the remainder of this decade. For example, when I asked in an online discussion forum during the spring of 2003 whether death march projects were likely to diminish, Erik Petersen responded by saying:

> Hopefully going down. Two reasons.
>
> The culture of the ever expanding IT budget has gone, and management wants projects to succeed (even if they still try to run them lean). From what I have seen, sub project deliverables are becoming more common and highlighting death march issues earlier, and projects are killed sooner.
>
> Better education should reduce the naiveté of newbies. Most graddies now learn about testing and quality as part of their studies. Some even learn project mgmnt. Agile methods are popular at universities as well, with test first development, etc. XP is becoming

> *more popular, complete with no overtime mantra. I think this will help to empower teams, both delivering better software, and standing up to unrealistic schedules and targets.*[18]

Whether there are more or fewer of them in the future, I should emphasize that I'm not necessarily opposed to death march projects; I agree with my colleague Rick Zahniser[19] that such projects can be an educational experience even if they fail:

> *I've told you before, I think everyone should be on at least one of these projects. However, there are some other things that you should do at least once:*
>
> *+ Spend a night in jail.*
>
> *+ Get commode-hugging drunk*
>
> *+ Raise a boy*
>
> *+ Raise a girl*
>
> *+ Start your own business*
>
> *+ Climb Mount Fuji*[20]

In any case, for the remainder of this book I'll assume that you *have* made a rational decision to join the death march project—though I'll remind you from time to time, in later chapters, that you always have the option of quitting during the project. But we'll assume that your primary objective at this point is to succeed, or at least survive, the death march project. In subsequent chapters, we'll see how that can be done.

NOTES

1. From: Richard Sargent, 72762,3342
To: Ed Yourdon, 71250,2322
Topic: Ed's new book project, Msg #159427, reply to #158778
Date: Mon, Jun 24, 1996, 2:22:20 AM
Ed,
A colleague of mine passed along this paraphrased quote. I think it applies here.
The definition of (corporate) insanity is doing the same thing again and again, and each time expecting different results.
I have no idea who originally framed the assertion, but it's gold!
Richard Sargent
5x5 Computing Solutions Inc.

2. From: Dave Kleist, 70730,1613
To: Ed Yourdon, 71250,2322
Topic: Ed's new book project, Msg #158064, reply to #158015
Date: Tue, Jun 4, 1996, 7:28:03 PM

Ed,

>> 1. Why would anyone in his right mind agree to work on a "death march" project (defined in the terms above)? <<

Because it's rarely printed as part of the want ad. Not much sense in saying, "Are you interested in working incredible hours for no additional benefit beyond your hiring salary? Does the idea of working endlessly on obsolete technology while 'waiting' for a slot to open up on that exciting GUI/DSS/warehouse/HTML subproject really entice you? Do you define three-tier architecture as an opportunity to hear what other project members will work on without your help?"

Seriously, death march projects are rarely billed as such, and it takes a lot of work when being hired from the outside to discover if your hiring company is prone to creating death march projects. In addition, death march projects only look that way. While they demand the hours, every hour is not productive. After a while, people find ways to do the things they being deprived of (pay bills, run errands). It just isn't billed that way. The environment still sucks, people hate it.

And, how accurate are those hours that are being billed? Where do they come from? Got any contractors? Ever heard of "nuisance hours" or "annoyance hours"? You know, where a contractor overbills because they can't stand some of the people they are working for. (Let me say right now that I've never done it and never will do it, but know people who have). The lead or manager does what the contractor thinks is stupid, and the contractor takes revenge (in their own quiet way). And, what about overhead? Are all hours to be marked to the project, including corporate and department meetings, training, etc.?

>> 2. If a colleague of yours was about to take on the task of managing a death-march project, what is the ONE THING you would advise him/her to do? <<

Try to craft an exquisite exit clause in the contract <VBG>. Seriously, one of the reasons for a runaway is the inability of someone to hear reality, usually upper management (either side, IT or business). Someone taking over a death march has got to find an angle for them to get some manuevering room (functionality, cost, time) in at least one aspect or they are doomed.

>> 3. Conversely: what is the ONE THING you would advise your colleague NOT to do, under any circumstances, when embarking upon such a project? <<

Acknowledge that it is going to be a death-march. Doesn't sound honest but admitting that it's going to be a killer can be demoralizing for two reasons: one, people don't like to hear that the next 6-12 months could be hell; two, management usually underestimates the negatives. Not much hope if you know right out of the gate that it's going to be ugly. I had friends who worked on one project that

```
had management openly admitting that there was going to be road-
kill on the project. Oddly enough, they had trouble recruiting
internal replacements once the turnover kicked in.
Seriously, admitting upfront that it's out of control is already
saying very little for one's management skills. If you ask, some-
times staff will volunteer ways to help keep it from becoming a
death march. In the death marches I've seen, the one thing that
I've seen common to them all is a lack of empowerment among the
staff.
- Dave
```

3. John Boddie, *Crunch Mode* (Englewood Cliffs, NJ: Yourdon Press/Prentice Hall, 1987), page 20.
4. Scott Adams, *The Dilbert Principle* (New York: HarperBusiness, 1996), page 2.
5. We'll discuss this idea in more detail in Chapter 4.
6. It should be emphasized that while the dot-com startups are probably the most recent, and perhaps the most extravagant, examples of this phenomenon, they are by no means the only ones. The software industry has spawned startup companies since the 1960s, if not earlier, and it will continue spawning them as long as smart people have intriguing ideas for exploiting new technology.
7. "X" used to be eight for many years, then it shrank to six. With the Enron/Worldcom scandals, the number seems to be shrinking steadily. Perhaps one day there will be only a "Big-One" accounting firm!
8. Email from "Ought-Six," June 17, 2003.
9. See my book, *Byte Wars: the Impact of September 11 on Information Technology*, for more discussion of this point.
10.
```
From: Kevin Huigens, 74762,2726
To: Ed Yourdon, 71250,2322
Topic: Ed's new book project
Msg #158577, reply to #158015
Date: Mon, Jun 10, 1996, 9:13:16 AM
Ed:
At our weekly staff meeting, my team and I had a brainstorming ses-
sion on your 3 questions. Here's our answers:
1. Why would anyone in his right mind agree to work on a "death
march" project (defined in the terms above)?
Everybody wants to feel wanted  Perceived opportunity
Perceived money gain Can't afford to lose job
Brought in from the outside to lead the project
Willing suspension of disbelief
Don't care whether project fails, get to work with cool technology
On-the-job-training on new technology  Eternal optimism
Challenge  Plain stupidity  Chance to prove yourself
To get the job done  It's the only project
Your friend is running the project  Your brother is running the
project (It'd take more than friendship)
```

Your boss said so You have no other life Nothing better to do
Stock options Existing pay vs. expectation of raise Love is blind
Resumé-building Ignorance Comeraderie
Expectations for how long it will take are too low
2. If a colleague of yours was about to take on the task of managing a death march project, what is the ONE THING you would advise him/her to do?
Leave me out Run! Keep your eyes open
Ask "What's the pay?" Get a lot of rest before you start the project
Make sure you can trust all of your co-workers
Realize the developers aren't your enemy, the managers are
Try to get management to understand the ramifications of the project
Communicate. Communicate. Communicate.
Keep the team small Hire new graduates Keep the team intact
Manage scope Review the design Focus is a substitute for time
Make sure testing plan is ready when it's time to test
Make sure you have a test plan Make sure everybody knows what to do
Documentation is critical Don't rush to code
Keep documentation updated and current
Everyone should have access to documentation
Have regular weekly progress meetings Have daily progress meetings
All code works before you leave at the end of the day
Keep plenty of good coffee on hand Make sure team is happy
Make sure team has everything they need
Use management by walking around
Make sure everyone understands what they're doing
3. Conversely: what is the ONE THING you would advise your colleague NOT to do, under any circumstances, when embarking upon such a project?
Don't plan a wedding Don't have unclear areas of responsibility
Don't allow design changes lightly Don't assume 1st version is final
Don't become irritated or angry Don't lose your cool
Don't let others lose their cool Don't forget to back stuff up
Don't expect everyone on the team to be dedicated
Don't get too personally involved in success or failure of the project
Don't rely too heavily on 1 member of the team
Don't allocate resources lightly
Don't assume team members understand the entire project
Don't overcommit Don't underestimate
Don't refrain from asking questions when you don't understand
Don't start the project
Don't start the project if you haven't got the money to finish
Don't commit to unreasonable dates

Don't be afraid to quit if you feel management is unreasonable
Don't be too hard on overworked, underpaid workers
Don't let meetings last > 1.5 hours
Don't be afraid to bend the rules
Don't forget to have a life Don't sweat the small stuff
Don't be afraid to let management know you need something
Don't be afraid to stand up to management
Don't forget to keep your resume updated
Don't accept as gospel info from so-called experts
Don't forget that management doesn't understand how to develop software
Don't forget that shortcuts just defer work, they don't eliminate it
Is that enough for you?
Kevin

11. From: Dave Kleist, 70730,1613
To: Ed Yourdon, 71250,2322
Topic: Ed's new book project
Msg #158064, reply to #158015
Date: Tue, Jun 4, 1996, 7:28:03 PM
Ed,
>> 1. Why would anyone in his right mind agree to work on a "death march" project (defined in the terms above)? <<
Because it's rarely printed as part of the want ad. Not much sense in saying, "Are you interested in working incredible hours for no additional benefit beyond your hiring salary? Does the idea of working endlessly on obsolete technology while 'waiting' for a slot to open up on that exciting GUI/DSS/warehouse/HTML subproject really entice you? Do you define three-tier architecture as an opportunity to hear what other project members will work on without your help?"
Seriously, death march projects are rarely billed as such, and it takes a lot of work when being hired from the outside to discover if your hiring company is prone to creating death march projects. In addition, death march projects only look that way. While they demand the hours, every hour is not productive. After a while, people find ways to do the things they being deprived of (pay bills, run errands). It just isn't billed that way. The environment still sucks, people hate it.
And, how accurate are those hours that are being billed? Where do they come from? Got any contractors? Ever heard of "nuisance hours" or "annoyance hours"? You know, where a contractor overbills because they can't stand some of the people they are working for. (Let me say right now that I've never done it and never will do it, but know people who have). The lead or manager does what the contractor thinks is stupid, and the contractor takes revenge (in their own quiet way). And, what about overhead? Are all hours to be

marked to the project, including corporate and department meetings, training, etc.?
>> 2. If a colleague of yours was about to take on the task of managing a death march project, what is the ONE THING you would advise him/her to do? <<
Try to craft an exquisite exit clause in the contract <VBG>. Seriously, one of the reasons for a runaway is the inability of someone to hear reality, usually upper management (either side, IT or business). Someone taking over a death march has got to find an angle for them to get some manuevering room (functionality, cost, time) in at least one aspect or they are doomed.
>> 3. Conversely: what is the ONE THING you would advise your colleague NOT to do, under any circumstances, when embarking upon such a project? <<
Acknowledge that it is going to be a death-march. Doesn't sound honest but admitting that it's going to be a killer can be demoralizing for two reasons: one, people don't like to hear that the next 6-12 months could be hell; two, management usually underestimates the negatives. Not much hope if you know right out of the gate that it's going to be ugly. I had friends who worked on one project that had management openly admitting that there was going to be roadkill on the project. Oddly enough, they had trouble recruiting internal replacements once the turnover kicked in.
Seriously, admitting upfront that it's out of control is already saying very little for one's management skills. If you ask, sometimes staff will volunteer ways to help keep it from becoming a death march. In the death marches I've seen, the one thing that I've seen common to them all is a lack of empowerment among the staff.
- Dave

12. From: Steve Benting, 72410,477
To: Ed Yourdon, 71250,2322
Topic: Ed's new book project
Msg #158362, reply to #158015
Date: Fri, Jun 7, 1996, 12:59:21 AM
Ed,
As long as you're asking...
>>1. Why would anyone in his right mind agree to work on a "death march" project (defined in the terms above)?<<
Because it seems to be a well-thought-out project this time. You've got someone leading who has a real sponsor in management, the project plan appears to be solid, the people involved all appear to be good. Hell, you *want* to work on this thing. Then it collapses because your sponsor gets taken out in a political struggle, the project plan turns out to be built on assumptions that are incorrect, and one or two key people turn out to be flaky. You can learn to watch out for them, but sometimes you misjudge. And you

don't want to believe that it's happening again. (I'm assuming some things here. I've only been involved on one large project, but it certainly went down hard. Delivery date was October, '94 and later moved to March '95. I was working on the contingency plan towards the end and left after most of the team in January '95. The new system still does not exist. The company is now in the process of purchasing someone else's system that doesn't have half of the functionality they originally required.)

>>2. If a colleague of yours was about to take on the task of managing a death-march project, what is the ONE THING you would advise him/her to do?<<

I would say to take care of his/her people as much as possible. Kick them all out of the office on Friday nights and try to make sure they're getting sleep. (Those months of 12-hour days six days per week can just burn out the developers, making them either quit or make too many mistakes.) No matter how badly the work needs to be done, you've got to take care of your people. Sometimes getting the most out of them requires sending them home. (If you know the project's in trouble when you start, you've got a long haul ahead during which you'll need good people.)

Also, make sure that you've got the best salary scale possible. It won't make all the difference, but it should be cheaper than attrition if it's enough to keep some people on.

>>3. Conversely: what is the ONE THING you would advise your colleague NOT to do, under any circumstances, when embarking upon such a project?<<

Don't let anyone put serious pressure on the employees besides you. Run interference to keep the developers free from others who are trying to ask them to run that 2-minute mile. (We had a developer working for us when I was the IS Manager -- and before the aforementioned project was started -- who was writing a new commissions system. The Sales VP came down to tell her that until she completed this system, her -- the sales manager's -- salespeople couldn't pay their mortgages. My VP quite rightly threw her out to let the developer work in peace.) That's not to say that you can't push those employees yourself, but you have to have some control over the stress levels in the organization if you're going to keep them going.

>> I'd like to solicit input, feedback, war stories, case studies, good jokes, etc.<<

This must be where I tell you about how, on that infamous project, the new President explained to me why he wouldn't sign off on requirements when asked to. (Needless to say, scope creep was a major factor in its death.) He was a down-home type who thrived on people taking his southern drawl as a sign that they were dealing with a country bumpkin. He had also just orchestrated the removal of our sponsor -- the previous President -- by killing the project.

His reason for the management group's refusal to sign off on requirements was that my VP was "going to hold our feet to the fire" with that document. In other words, he wouldn't agree to sign the document because he would have to live with it later! At this time, I knew I really needed to get out of there, and quickly...
Steve

13. From: S. Marsh Roberts [ICCA], 70007,4251
To: Ed Yourdon, 71250,2322
Topic: Ed's new book project
Msg #158111, reply to #158015
Date: Wed, Jun 5, 1996, 5:31:15 AM
Ed--
>> 1. Why would anyone in his right mind agree to work on a "death march" project (defined in the terms above)? It's understandable that an inexperienced software developer (or someone who hasn't had the pleasure of reading Scott Adams' "The Dilbert Principle") might be bamboozled by management's claim that the death march is an anomaly, and that the superhuman efforts are going to revolutionize the human race, defeat Communism, cure cancer, etc. But after you've heard this pitch two or three times, it sounds like a broken record. So why do we get sucked into this again and again?<<
The "heros" are needed, wanted, desired. They are certain of their place in history, if only they can keep this project from outright sinking under its own weight.
The same people take on EMT work and enjoy firefighting (literally). If you only win once in ten times, but everybody else lost all ten, wouldn't you be a hero, too?
>>2. If a colleague of yours was about to take on the task of managing a death-march project, what is the ONE THING you would advise him/her to do? (The "one thing" motif was suggested by Jack Palance in the wonderful movie "City Slicker", starring Billy Crystal)<<
I'd encourage him to keep his sense of humor. It may be gallows humor, but it's all that such a group has. <sigh>
>>3. Conversely: what is the ONE THING you would advise your colleague NOT to do, under any circumstances, when embarking upon such a project? <<
I would encourage him (and excuse me, it would be a him in 99/100 attempts) to not invest in options or get a large mortgage. You can only take high risk in one arena at a time, without risking a total wipeout of personal assets.
I once said that I would be willing to take a certain job whose incumbent tended over a seven year period to last no longer than a year. I figured that three months' salary would provide enough savings to recover from the inevitable.
Sharon

14. From: Paul Neuhardt, 71673,454
To: Ed Yourdon, 71250,2322

Topic: Ed's new book project
Msg #158349, reply to #158015
Date: Fri, Jun 7, 1996, 12:20:19 AM
Ed,
<< 1. Why would anyone in his right mind agree to work on a "death march" project (defined in the terms above)? >>
For me, it was ego, pure and simple. They told me that they just KNEW I could help prevent the project from becoming a death march. I was made the "technical project manager," given ego boosts on a regualr basis, then hung out to dry along with the rest of the team. Left, right, left, right, left, PLOP!
(The really embarassing thing is, I let these same people do it to me AGAIN just one year later. Once I began to feel myself falling into the step of the death march, I ran like hell for the door. Me, and about 60% of the rest of the staff. BTW, It's been four years now since I first got suckered in, and neither system has ever seen the light of day, nor will they.)
<< 2. If a colleague of yours was about to take on the task of managing a death march project, what is the ONE THING you would advise him/her to do? >>
To quote those mad englishmen in "Monty Python and The Holy Grail" I would say "RUN AWAAAAAYYYYYY!!!". It sounds like a flip answer, but it isn't really. Some of the most damaging effects of a death march project are psychological. Lower self esteem, depression, anxiety and sudden mood swings are all behaviors I have witnessed (and sometimes experienced) during these projects. I've seen at least one marriage break up in no small part because the partner involved in a death march let it consume her so totally that she bacame an entirely different person, one whom her husband (and most of the rest of us) had no desire to be around. I know another woman who, when a three year "death march" ended with the project being cancelled, said that it was the only experience in her life that even approached the heartbreak she felt when she miscarried during the sixth month of pregnancy. Now that's trauma. If you can get out, go.
<< 3. Conversely: what is the ONE THING you would advise your colleague NOT to do, under any circumstances, when embarking upon such a project? >>
If you can't beat 'em, this is one case where you do NOT want to join 'em. Do not let yourself become too emotionally attached to the outcome of this project. Like POWs on death marches, think about anything else but the march in order to survive. Try to go to work, grind out your day's brick for the wall, and go home. If you want stimulation and personal reward, read a book, join a social club, volunteer at the local animal shelter or buy a kiln and throw some clay pots. Do anything to keep your mind off of work as much as possible. The moment you get to attached to the project, the guards

with the rifles win and you, the lowly POW, lose.
Paul

15. From: Al Christians, 74031,316
To: Ed Yourdon, 71250,2322
Topic: Ed's new book project
Msg #158029, reply to #158015
Date: Tue, Jun 4, 1996, 12:04:05 PM
Ed
Sounds like you are going to have a lot of fun this summer.
1. Why would anyone in his right mind agree to work on a "death march" project?
Since you mentioned "City Slickers", the movie that used such regrettable sexual stereotypes, I am somehow prompted to reply "testosterone", which is about the same as "because it's there." There are plenty of jobs that raise the 'why?' question. Underground mining, cowboying, logging, smoke jumping, jet fighting, submarining, even high-rise window washing all have serious drawbacks far beyond what you describe for software projects, and yet all these have practitioners whose sense of self is linked to their profession.
But if you really think that reasons are needed, here are a few:
a. We think we learned so much in the last experience that it would be a waste to not find a project on which it could be applied.
b. We know that some of our colleagues are going to be suffering, and we don't mind doing our part to lessen their burden.
c. It's like a lottery ticket -- despite the odds, we can imagine the possibility of large rewards if we win big.
d. The high level of urgency that arises during these difficult projects redistributes power to those who know how to resolve the crises, i.e., us, and we like power.
> 2. If a colleague of yours was about to take on the task of managing a death march project, what is the ONE THING you would advise him/her to do?
Remember that the people who love him/her, love him/her for reasons that have nothing to do with the project.
> 3. Conversely: what is the ONE THING you would advise your colleague NOT to do, under any circumstances, when embarking upon such a project?
Since "this is the way it's been for a long time, and this is the way it's gonna continue to be," don't try to work at a pace that you can't sustain healthfully for a long time.
Al

16. From: David Maxwell, 100342,3620
To: Ed Yourdon, 71250,2322
Topic: Ed's new book project
Msg #158991, reply to #158015
Date: Mon, Jun 17, 1996, 4:53:16 AM

Ed

As I talked on another thread the other day, projects are like a marriage. We tend to start of naiively and full of hopes and slowly as reality sets in, we have to reassess our expectancies within the relationship. There are many reasons apart from *logic* that attract people together into a marriage and it is the same case with projects. With a predominantly youthful workforce, it is likely that the "death-march" project will occur again and again as a training ground for managers and developers alike. And, as I know from personal experience, I often repeat the same mistake many times before the penny drops.

Niezche, the German philosopher in the last century said that "society is governed by mediocrity". What he was presumably implying here is the central, conservative-stream will tend to dominate behaviour and control events. This central-stream is hell-bent on preservation from the extremes and will draw the blinds on anything that threatens their positions. What we are really asking for in IT is a radical re-shaping of the way projects are managed, with open vertical and horizontal communication.. and an openness to radicalism. This is very threatening for the central core of the typical task, role, club organisational culture. An Organisation with a cuture of existentialism has a much better chance of developing good projest on a regular basis but these Organisations are still a rarity.

An old girl-friend of mine who is in a leading position in one of the Major Business Schools regularly seeks advice from me as to how to overcome the deluge of internal politics and methods that are stifling their practices, certainly a case of not practicing what they preach! In addition, Computer Science departments the world over are paying scant regard to People and Management issues as the Lecturers themselves are, in general, inept outside of the technological framework.

So perhaps it is inevitable that, with an inappropriate education and cultural backdrop, we can expect "death march" projects to continue to be the norm... But looking at it from another perspective, these "death march" projects are the essential grist-for-the-mill for the few success stories that make the whole show worthwhile.

David

17. This scenario is *far* more common in North America than it is in Western Europe or the Pacific Rim countries that I've visited. While companies all around the world have engaged in re-engineering projects, it's less common, outside North America, to see the "radical" re-engineering projects that eliminate large numbers of employees. And for the same reasons—cultural traditions, social policies, government regulations—there are fewer death march projects in these countries. The work force, especially in Western Europe, is far more likely to be shielded from excessive overtime, and is far more likely to adamantly refuse to give up sick days, vacation days, holidays, per-

sonal days, and other forms of time off. Whether this is a good thing or a bad thing is outside the scope of this book.

18. Erik Petersen email, June 20, 2003.

19.
```
From: Rick Zahniser (SL), 70313,1325
To: Ed Yourdon, 71250,2322
Topic: Ed's new book project
Msg #158437, reply to #158015
Date: Fri, Jun 7, 1996, 10:57:25 PM
Ed,
>>why do they do it??<<
I think they do it because, as Al, suggests, they think they're
better than others who have tried. And, sometimes they really are!
(That doesn't eliminate the death march. In fact, it probably pro-
longs it.)
I've told you before, I think everyone should be on at least one of
these projects. However, there are some other things that you
should do at least once:
 + Spend a night in jail.
 + Get commode-hugging drunk
 + Raise a boy
 + Raise a girl
 + Start your own business
 + Climb Mount Fuji
(The Japanese have a saying:
"He who fails to climb Fuji-san is a fool. He who climbs Fuji-san
twice is an even greater fool.")
One thing to do:
Get a good manager, who is empowered to do the right things.
One thing not to do:
Kill yourself when the project goes south.
Sr. ric
```

20. Science fiction *aficionados* will recognize the similarity between Zahniser's advice and the wonderful aphorism from Robert Heinlein in *Time Enough for Love: the Lives of Lazarus Long* (Ace Books, re-issue edition, 1994): "A human being should be able to change a diaper, plan an invasion, butcher a hog, conn a ship, design a building, write a sonnet, balance accounts, build a wall, set a bone, comfort the dying, take orders, give orders, cooperate, act alone, solve equations, analyze a new problem, pitch manure, program a computer, cook a tasty meal, fight efficiently, die gallantly. Specialization is for insects."