Chapter 1 : Death March - Edward Yourdon - Google Books

In project management, a death march is a project that the participants feel is destined to fail, or that requires a stretch of unsustainable overwork. The general feel of the project reflects that of an actual death march because project members are forced to continue the project by their superiors against their better judgment.

By Hugh Hancock I suspect that everyone reading this has worked truly insane hours at one time or another. And one thing that sprang to mind immediately was the ubiquitous death march. The trailer was bloody terrible, too. And right at the moment, I personally know at least three readers of this blog who are doing massive death marches on individual projects. That includes me, as I rush to finish the biggest project of my career. More on that in a day or so. This has been proved time and time again, notably in "Peopleware". Few people can profitably work more than 40 hours a week for any length of time. If you work more time than your comfortable maximum and keep doing it, your productivity will drop and keep dropping. Quite rapidly, you will become less productive than you would be if you worked 40 hours. This is probably a good reason not to buy shares in the computer games industry. A death march lets you steal time. During the release window of my first feature film, I cancelled all my exercise classes and ate, essentially, sugar. I figured I could pick things up after my life had calmed down. Rule 2 of Death Marches: As anyone who has suffered with a repetitive strain injury can tell you, sitting in a chair typing for 14 hours a day puts a real physical strain on your muscles, tendons, nervous system, hormonal balance, and circulation, just like any other challenging physical activity. Take ten minutes at the start of the day to stretch - here are some good ones to start with. Personal training sessions are great if you can afford them. You can spend it on stuff that will support or repair your muscles and skeleton: Please stop me falling apart. As previously mentioned, personal training sessions are also good. The Pilates lesson I took last week, for example, dealt with half a dozen niggling aches which past experience suggests would have by now been fighting for the title of "crippling". Famous people do it and it works! That smug-looking guy from the famous lifestyle blog does it for an hour a day before writing his blog posts in his minimalist office sitting in the lotus position! As you may have gathered, meditation is not exactly my favourite activity. Meditation essentially functions as a reboot for the brain. Concentrate on your breathing, do a body scan, repeat a mantra or visualise an image, whatever. In fact, I should go and do 5 minutes of meditation now. For your body, those hours still feel like hours. Hours of shallow breathing, insufficient blinking, repetitive micromovements, and static posture. Hours for muscles to get over-strained, spasm, be supported by other muscles which then also go into spasm, and so on. And then you get up to make a coffee, bend down injudiciously, and everything goes twang. And not in a fun way. Make sure there is no way you can possibly stop it from doing its thing: And then let it run, and prepare to hate it with a blazing fury. Being interrupted every half hour will feel hideous. The other half is what you do with your breaks. Of course, it helps to know what stretches to do. I shall be back on Tuesday and Wednesday, talking about the Future and Technology, and specifically the past, present, and future of virtual filmmaking and Machinima!

Chapter 2 : Characteristics of a Death March project - Stack Overflow

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 Â 50 percent.

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. 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. 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. 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. 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. 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. 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 functionalityâ€”i. Thus, if the normal work-week is 40 hours, then a death march project team is often found working 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. Of course, even a project without the schedule, staff, budget, or functionality constraints described above could have a high risk of failureâ€”e. But most commonly, the reason for the high risk assessment is a combination of the constraints described above.

Chapter 3 : Death March Projects by John Capron on Prezi

If you've never been part of a project death march, consider yourself lucky. The death march is generally defined as a project in which employees are asked to work insane hours on something that is doomed to fail.

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. Everyone is an idiot, not just the people with low SAT scores. No matter how smart you are, you spend much of your day being an idiot. Or perhaps you consider it an insult that someone would even make such a suggestion. In that case, Table 1. Naive optimism of youth: Indeed, even without an explicit rationale of the sort shown in Table 1. 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: Why are these constraints being placed on the project? The possibilities are endless. 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 Jonesâ€¦"? 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. 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. 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 honoredâ€”no matter what. What should you do about it? 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 impressionâ€"either from your political instincts, or from the experiences from previous projects in your organizationsâ€"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 projectâ€"e. 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! You probably read the paragraph above, bristled at the apparent insult, and muttered, "Damn right! I really can build any system over the weekend! In any case, nothing that you hear from an old fart like me is likely to change your mind. The best advice, I think, is "Run! 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. And a large percentage of these projects will fail; a large percentage of the companies will fail with them. But not everyone is familiar with the culture and environment of a corporate startup. 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: It may reflect the personality of the corporate founder s , and it may reflect the corporate culture in its earlier daysâ€”the corporate behavior at Microsoft, for example, has often been attributed to these factors. The only way we succeed is to work twice as hard. 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 s. 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. 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: 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. 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. 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. 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 projectsâ€”beyond the obvious characteristics of size, schedule, and budgetâ€”is the attempt to use bleeding-edge technology for an industrial-strength application. 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 authoritiesâ€”especially in the areas of taxation, reporting of financial details to stock-market authorities, environmental regulations, and so forth. 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: 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. 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. Such crises, according to the purists, are the result of poor planning and poor management; an "unplanned crisis" is therefore an oxymoron. For better or worse, we live in a world of chaos, and death march projects are a natural consequence of such chaos. 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, , was coming, and it was obviously a deadline that could not be postponed. So why did so many Y2K death march projects get launched in and ? 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"â€”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.

Chapter 4 : What is a Death March Project and Why Do They Happen? | Death March Defined | InformIT

In software development, a death march project generally refers to a project that has a fixed release date with fixed functionality and fixed resources - resulting in crazy demands from management.

Here is a man who built his career and made his fortune on software development methodologies. He wrote Manhattan-phonebook-sized tomes about them, taught countless seminars on them, built a consulting company around them, and helped get them institutionalized at some of the largest corporations in the world. Advertisements Structured design Ed Yourdon put himself on the map in the mids with a set of formalisms for software development collectively called structured methods. His work of that period is represented by the classic, oft-revised Structured Design: At that time, the first wave of large commercial systems had been designed and implemented. At best, ad-hoc management techniques were used. His work was one of the most successful attempts to provide some structure and sanity to large system development -- projects that involve tens or even hundreds of engineers. It included an elaborate graphical notation scheme for depicting elements of systems, which was sort of like a visual programming language. But it was not executable; it was simply a way of notating components of systems and their interrelationships so that developers and management could see the big picture. Using structured methods, you could describe systems from the top level down to a level at which relatively small pieces could then be programmed, in whatever language made sense, with confidence that they would all link together into a working whole. At first, they allowed programmers to draw the pictures on a screen and add documentation. The tools would make the drawings look nice, annotate them, sort, search, print, etc. They were highly expensive and required weeks of training, but the benefits were clear: CASE tools that actually generated code came out in the mid to late s. Smalltalk and CLU embodied what became known as object-oriented design principles, which by the late s were supplanting the older structured methodologies. So Yourdon teamed up with Peter Coad and adapted his principles to object-oriented design. Object-oriented CASE tools were developed that -- finally -- not only allowed programmers to draw pretty pictures but also generated code. In other words, CASE tools and structured methodologies were a nice idea that never quite achieved the mass acceptance that folks like Ed Yourdon hoped for. Two important factions actively eschewed structured methods: Yourdon became known as one of the most important founders of modern software engineering principles, but his ideas were more admired in theory than put into actual practice. This must have gotten Yourdon personally depressed during the late s and early s. Nontrivial software needs to be developed by teams of people using heavy-duty processes that give designers all of the fun and responsibility while reducing programmers to mechanical craftsmen. You can train people in developing countries like India and the Philippines to be mechanical craftsmen and pay them wages that are a tiny fraction of those earned by American software professionals. Global telecommunications make it possible for American companies to use such people as programmers on their development projects. Therefore, American programmers are overpaid versions of their overseas counterparts, and hence unnecessary. Therefore, programming in America is a dying profession. But others were never valid in the first place. Programming has never been allowed to become a profession like law, medicine, or architecture -- at least not in America. Industry applied the command-and-control management philosophy, which was in vogue in the s and 70s, to software development; structured and similar methodologies were natural outgrowths of such mentality. Only recently have a few enlightened software companies begun to figure out how to manage the so-called "cowboy" programmers who disdain bureaucracy but get most of the work done, and get it done quickly. But it did not become wrong. It was always wrong. Large corporations have been reducing the role of corporate management and giving more operating autonomy to smaller divisions, on the rationale that local empowerment must go hand-in-hand with local accountability and local performance. Analogously, Yourdon notices that the most productive software teams are those that are given the freedom to choose their own tools, methodologies, and working styles. A cynical version of this philosophy is "If you can do it the way you want, then you have fewer excuses for failing. He stresses notions like "personal best practices," "dynamic processes" i. All of this runs exactly opposite to his former command-and-control view of the world. Yourdon observes that, through highly-competitive free market forces, users have essentially been deciding how much quality they want. It may not be necessary to have a zero-defect word processor, even though no one doubts the importance of bug-free air traffic control systems and nuclear power plants. For most types of packaged PC software, users want OK quality now rather than bulletproofing later. This all makes sense, but it flies in the face of some of the admittedly extreme claims of certain methodology czars in previous decades: There was even an experimental technology called "transformational software development" that purported to do all of this automatically and thus produce zero-defect code. We all now know that this is bunk, because it is impossible to capture user requirements completely and accurately. Users rarely know, and can even more rarely articulate, what they want from software; and even if they can, the requirements are highly likely to change during the duration of the project. So now Yourdon sees quality as something that requires as much work as functionality. The most famous proponent of "good enough" software is, of course, Microsoft. They empower their top developers and add processes where they make sense and do not interfere. He makes room for dissenting opinions. But the majority of voices represented are those of his fellow methodologists and metrics gurus, rather than, say, the hotshot programmer whose incredible ideas and productivity put a startup company on the map, or even the development manager for a huge-selling shrink-wrapped application. Certain aspects of this stance are hard to dispute -- essentially, the notions that you should make informed choices about methodologies and processes rather than ignorant ones, and that any process is better than no process. But overall, he comes across like a guitar teacher who has despaired of teaching young Eddie Van Halen wannabes lots of scales, positions, and arpeggios when they just want to learn "Eruption" from the record well enough to play it for their girlfriends next weekend. He also summarizes certain aspects of software development that have helped change its mechanics over the years. First, teams of programmers can be more productive, and thus smaller, thanks to the latest generation of rapid development tools a la PowerBuilder and Delphi; Java technology, which makes software automatically platform-independent; and other innovations. This is especially important in light of another fairly recent discovery: People like Capers Jones and Meilir Page-Jones unrelated , who measure all sorts of things about software development projects, have published statistics that lead to this inescapable conclusion. For example, they have found that two things rise with the size of projects: In fact, Capers Jones found that almost three-quarters of the effort on the largest projects is non-productive. I can say from personal experience that this is a good rule. Reality dictates that such projects are virtually impossible to do successfully. You have to break them down into smaller chunks and thus live with architectural compromises. Marching at Internet speed Small and quick are the only projects that make sense anymore, but, to make matters worse, management often requires they become even smaller and quicker. Increased business competition, more volatile corporate politics, and other factors engender many projects that have utterly unreasonable deadlines and are resourced with half the money and people you need to get them done. Yourdon calls these projects death march projects. The only projects worth doing are those that promise breakthrough business gains, such as dramatic process change or competitive advantage. So instead the project is often sold on the basis of mundane, incremental business benefits. Incremental benefits mean small budgets and short timelines -- yet expectations are still on the breakthroughs. The results are projects with paltry budgets and timelines that must deliver miraculous results anyway. Large corporations treat software developers poorly. Despite much gum-flapping to the contrary, IT is generally not at the table when executives discuss business strategies. Corporate developers are considered back-office overhead expense, which often means that they are not provided with good working conditions, tools, training, etc. These firms offer excellent learning opportunities, great tools, solid career paths, and access to communities of talented people. But they require that you work "death march" plus hour weeks and renounce all personal life. The companies that need technology breakthroughs most desperately are almost always those that place the most obstacles in the way to getting them implemented. There is no dead hand of bureaucracy at a Silicon Valley start-up company. There sure is at, say, General Motors. Silicon Valley start-up companies also customarily move at a "death march" pace, because of the "Internet time" schedules under which they need to bring products to market. So what do you do when you are faced with a six-month timeline on a project that looks like it should take a year? You know you will be faced with long hours and unbearable pressure. What tools, what processes will help you get it done? The book is also full of practical tips for project managers who must run "death march" projects, on such subjects as maximizing productivity amid horrifyingly long hours, retaining and motivating good people, negotiating with management, navigating political waters, and so on. This book contains advice that is disappointingly mundane, considering the source. Here is the man who virtually invented software development methodologies, now telling us to avoid the corporate "Methodology Police" in order to get the project done on time. His advice is unquestionably good, but it amounts to this: Just use your common sense, stick with the tools you like best, and get the work done without unnecessary overhead. Who needs Ed Yourdon to tell them this? For them, this book would be like Cotton Mather telling his flock that a glass of wine with dinner every night is medically beneficial and makes you less uptight. These books are nevertheless worth reading, for a couple of reasons. First, Ed Yourdon is a very good writer. His style is reminiscent of the messages he is preaching: Second, he brings the perspective of one who has probably seen more different software development situations than anyone else on Earth. And, admittedly, he has always claimed that blind adherence to or inappropriate adoption of any software methodology, whether his or another, leads to disastrous results. He has always encouraged us to use common sense and put tools and processes in perspective. That perspective is always worthwhile, even if some of the baggage he carries seems old and worn.

Chapter 5 : Death March by Edward Yourdon

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.

Welcome to my Kickstarter! Its purpose is to raise money to commission gorgeous interior artwork for the first novel in my upcoming fantasy trilogy, as well as produce a hardcover version. The entire trilogy has already been written, Audible has secured the audiobook rights, and the ebook, trade paperback and audiobook are set to be released this August. Euphoria Online is a completed trilogy. I was all set to publish the books this spring when Audible expressed interest, offering me the chance to have the wonderful Vikas Adams narrate the audiobooks. This delay in the publishing presented me with an opportunity: I want to hearken back to those old Fighting Fantasy books I used to love and bring Euphoria Online to life with artwork depicting the world and monsters within its pages. To display character art of my heroes as they progress through the game, like my old favorite Diablo 3. This will take the book to the next level, and if the Kickstarter funds beyond my expectations I hope to commission artwork for the other two books and more. You can see more of his art here. The wyvern banner up top is another example of his work, created exclusively for Euphoria Online and depicting one of the monsters my hero goes up against in the first book. I must have been six years old when I picked up my first Fighting Fantasy book with its iconic green spine. Written by Steve Jackson and Ian Livingstone, those choose your own adventure books featured endless worlds filled with terrifying monsters â€" and I got to play the hero who defied them all. I bought this book in Not only did that interactivity thrill me, but the images contained within each book lit up my imagination. Now I write by day and game by night, and never thought that the twain would meet until I discovered something last year that blew my mind: LitRPG is a fantasy genre that features regular people that are thrust into virtual worlds and forced to become heroes. Stories about folks like you and me who either through magic or technology enter fantasy worlds like those of the Fighting Fantasy books and have to fight against impossible odds to win through. Gaming and books had finally come together for me, and I knew that I had to write something in that genre right away. As I said, the series has already been written and edited and is comprised of three books: In desperation, the best and the brightest unite under the aegis of the United Nations and connect every major computer system and cloud network in the world to create our last chance at salvation: How does the creation of this wondrous virtual world tie into the salvation of humanity? Mystified, people begin to play while endlessly speculating about its role in staving off global disaster. Enter our hero, Chris Meadows. When his brother is caught salvage diving off the sunken coast of Miami Beach, the government decides to make an example of him and seek capital punishment. Desperate, Chris leaps at an opportunity to play in Euphoria and take advantage of its most controversial aspect: He studied architecture in Athens, where he completed his postgraduate studies in space design. Then he realized it was time to pursue his real passion, painting. He spends his nights on roofs of houses in the Aegean, petting cats and gazing at clouds, hoping that one day it will be the other way around. Every now and then, when possessed by forgotten cloud deities, he paints. Vikas Adam audiobook narrator - Vikas is a classically trained actor with numerous credits in stage, film, commercials, and television, in addition to his over recorded audiobooks. He likes concept art especially character and environment design. Laura Hughes copy editor - Laura is a freelance editor based beneath the grey, pigeon-filled skies of northern England. In addition to writing fiction of her own, Laura is the founder of The Fantasy Hive, and has also been known to write articles for Fantasy Faction and Tor. The reward level breakdown is pretty simple: The audiobook will only be available once Audible releases it in August. Click here to view a high-resolution version that is much easier to read. This update here has a full run-down of how add-ons work and should help answer any questions. Click here for a larger version of the chart. I have a number of stretch goals planned for this Kickstarter. Others will be announced as funding levels are reached. He is the author of the epic fantasy series, Chronicles of the Black Gate, which allowed him to leave his marketing job and become a full time author on Jan,

Chapter 6 : Death March author Ed Yourdon admits he was wrong - SunWorld - July

In his book, Death March (2nd edition,), Edward Yourdon says Death March projects are becoming increasingly common. I hope that wasn't true circa , as my experience is that Death March projects were the norm throughout the s and most of the s.

They are the projects where he project schedule, budget, and staff are much less than what is necessary for completion of a daunting task. The planned objective is unrealistic. People are working 14 hours a day, six or seven days a week, and stress is taking its toll. Such projects have a seriously high risk of failure, yet the clients management is either blind to the situation or has no alternative. Why do these irrational projects happen, and what leads people to get involved in them? The personal goal of the project manager and team members often shrinks down to mere survival: I was involved in a number of death march projects in the s and I was also the sponsor who started some. Well, it is usually about the psychology and initially a lack of understanding of the true situation. So how do you notice if you are in a death march? There are many signs that may help you detect if your team is falling into this dynamic, but the main indicator is team motivation, the problem with motivation is that hard to tell when the motivation is going down until is very late. A few indicators could be: The team is starting to miss deadlines with apparently no reason. It could be an early sign that they are loosing motivation and concerns should be raised from the second deadline missed. Communication is not as fluent as is used to be. In a healthy team communication happens in an informal way almost continuously, if you detect that your team is starting to communicate less, probably is because motivation is going down. The team is been under high pressure for too long with no reward. This is one of the big motivation killers. Decisions have been imposed by client sponsor or management and the team does not agree with them. The term death march projects was coined by the American project professional Ed Yourdon in a theme issue of the American Programmer magazine in I got a copy from a project manager at Handelsbanken Markets in Stockholm in the autumn of and below you can find a scanned copy to download and read. Do so, it is worthwhile. American Programmer â€" â€" Death March Projects So, what are really the characteristics of a death march project, how do they happen and what can be done about them? In my experience, having been through a number of them, 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 three 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. 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. Unfortunately, it is happening often because of the ongoing economic recession and the associated cutbacks in 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. I have seen such an outcome in the procurement of a public sector project in Africa recently, and I dread the outcome for the client. Thus, if the normal work-week is 40 hours, then a death march project team is often found working 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 Red Bull. Of course, even a project without the schedule, staff, budget, or functionality constraints described above could have a high risk of failure, e. But most commonly, the reason for the high risk assessment is a combination of the mentioned constraints. Maintaining the well-being of the team is vital. A death march project has more than its share of strains. At best, the project is behind schedule for most of its span. The demands on the team are intense, as the project sponsors continue to hope and sometimes push for miraculous catch-ups. The first two conditions for success in a death march are vision and realism. The team must commit to delivering and must believe in the value of the end results. If the project is strictly impossible, who can succeed? No circumstances other than cancellation of the project will change the outcome. If the goal is superfluous, who cares? The target project for a successful death march is one with marginal, not impossible, odds. The success of a death march is the triumph of the positive over the negative. Create a positive environment 2. Cut the right corners 4. Find the one true measure of success 5. Expect and reward achievement Of course, this advice is good for any project. In a death march, doing these things will give team members a reason to seek out your kind of management and that kind of work. Sooner rather than later, the team will want to reach new goals, take on new challenges, and again reign over schedule and need.

Chapter 7 : Death march (project management) - Wikipedia

INTRODUCTION IT projects with greater than 50% chance of failure are commonly referred to as Death March projects. Although not openly discussed in the past, Death March (DM) projects are becoming more popular in discussions around the water cooler, on Internet blogs, and even in professional journals and periodicals.

Agile Product Development from Gang. Agile software development and Death March projects in the same sentence. In his book, Death March 2nd edition, , Edward Yourdon says Death March projects are becoming increasingly common. I believe one of the driving forces behind the move toward humane workplaces and lightweight methods was the prevalence of Death March projects. Such approaches began to gain traction in the early s, and became popularized after the Agile Manifesto was published in A book came out in to explain how to run Death March projects on purpose. Building Effective Systems on a Tight Schedule, explains how to drive a team of software developers to nervous breakdowns, cardiac problems, divorce, burn-out, and suicide as a way to deliver a project on a very tight timeline. In the s, I worked on many of these projects. I myself wasted many years when I could have been living a balanced life, and instead spent day and night pushing myself beyond all reasonable limits to deliver a piece of software whose value to humanity could never compensate for the human cost of producing it. Many thousands of people did the same. The good news is the Death March is no longer the norm in software development. It has a five-star rating on Amazon. The case study presented in the book is of a betting application for a casino. It helped casino owners extract money from gambling addicts in the same way as drug dealers extract money from drug addicts. I leave it as an exercise for the reader to judge whether the value to humanity justified the human cost exacted from the development team. Opinions differ, it seems. So, what happens on these Death March projects, anyway? In my experience, the following things happen: Management explains the importance of the project to the staff and asks for volunteers. People volunteer for their own reasons, but mainly because they are excited about the challenge. A team is organized from among the volunteers, representing all the skill sets and responsibilities necessary to deliver the solution. Administrative boundaries that normally divided functional silos, management hierarchies, and the like are suspended for the members of the team; everyone can communicate directly and in person. Rank, formal hierarchy, titles, and the like are temporarily suspended within the war room for the duration of the project. Stakeholders are present in the war room throughout the project and they provide the team with immediate clarification and feedback about needs and priorities. The scope of the work is so large compared with the available time that the team dispenses with conventional planning and estimation. They dive right in, focusing on the functionality directed by the key stakeholder. The team delivers as much of the most important functionality as they can within the allotted time. If any key functionality remains unfinished, they cover for it manually while continuing to tie up loose ends after the delivery date. When the project ends, the survivors crash for several days to recover from the exertion. The days immediately following the project are not normal work days. Some of the survivors decide to change jobs or change careers. Others take care of their new health problems or their divorces. Those who escape with most of their sanity intact swear that they will never again participate in a Death March project. They will not be available the next time management asks for volunteers. Now consider some of the key characteristics of Agile software development: Another difference is the recognition of the value of human beings. What may happen afterward is of no concern to the Death March manager. There are more similarities than differences between Agile and Death March projects. What does it mean? I think it means we can consider an Agile project as a Death March project that is stretched out over time. If a s-style project consisted of nine months of meetings followed by a three-month Death March, then an Agile project might consist of six months of slow Death March. We deliver in half the time, with a strong focus on customer-defined value, close alignment with stakeholder needs, effective use of practical methods and techniques, and close collaboration and teamwork. If people are doing only the things that directly help with delivery during the Death March, then why waste the first nine months in non-value-add activity? The team can continue to work on the next project in a sustainable way. Good luck with that. Download the free agile tools checklist from Gang. This guide will help you choose the right agile tools to position your team for success. Read More From DZone.

Chapter 8 : Death March, Second Edition [Book]

In Death March, Second Edition, Ed Yourdon sheds new light on the reasons why companies spawn Death Marches and provides you with guidance to identify and survive death march projects. Yourdon covers the entire project lifecycle, systematically addressing every key issue participants face: politics, people, process, project management, and tools.

Chapter 9 : Death March Projects

There is a famous book in our field written in the 's by Ed Yourdon called "Death March". In it he details the phenomenon in project management of death march software projects. He observed a trend in organizations who plan software projects to estimate so poorly that completion becomes overwhelming and unlikely.