Death March projects

Ed Yourdon

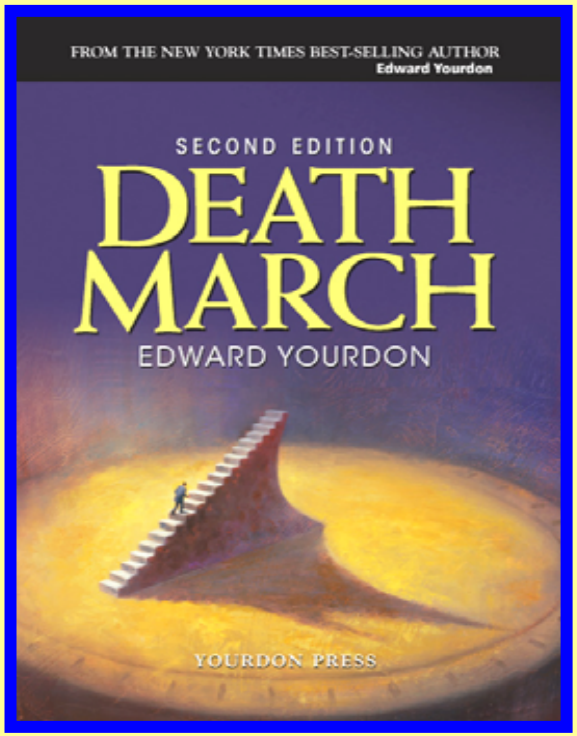
email: ed@yourdon.com

web: http://www.yourdon.com

blog: http://www.yourdon.com/blog

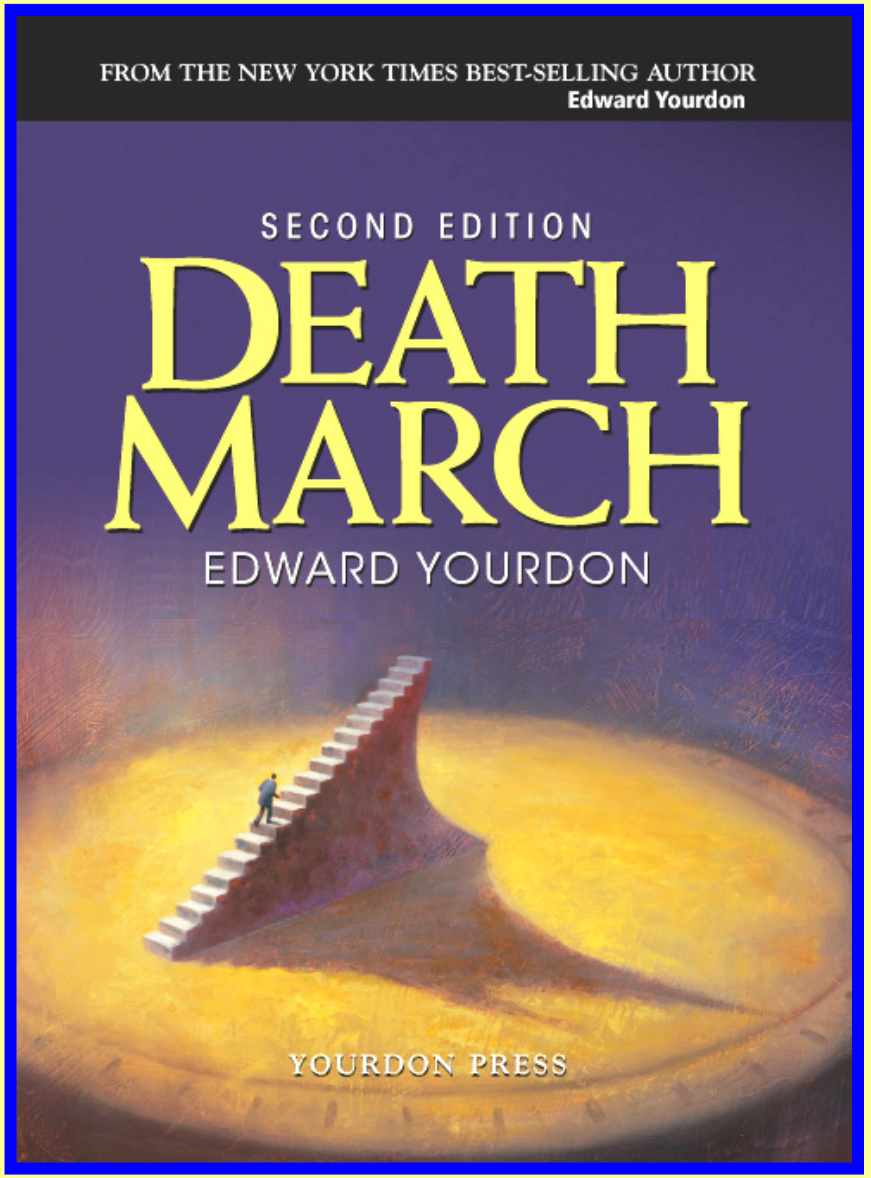
JaSST keynote presentation

January 30, 2007

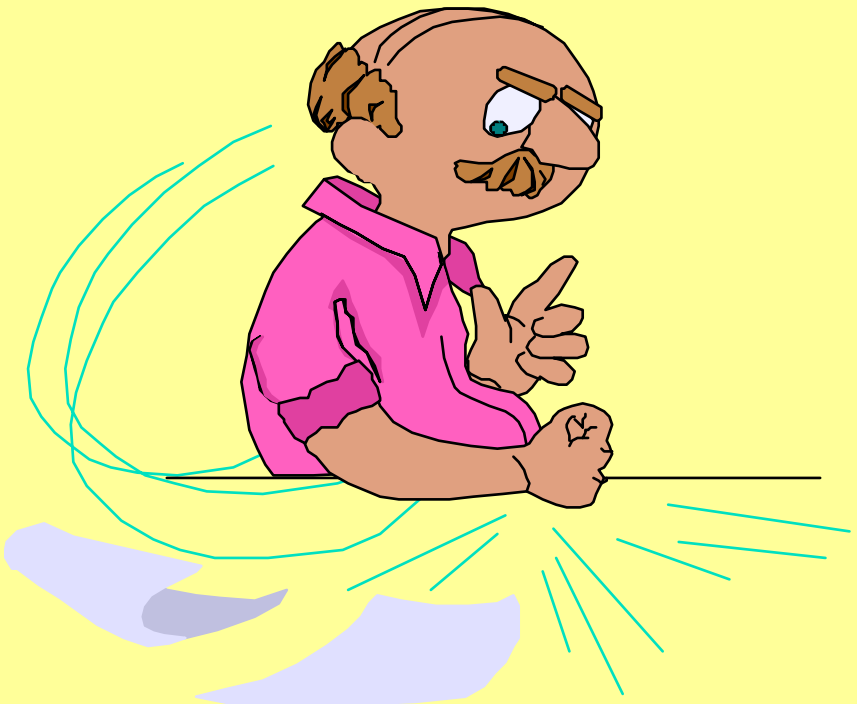


What is a “death-march” project?

- ***Origin:*** **"extreme" sports, ultra-marathon, triathlons, etc. Sometimes known as "death-march" projects**
- ***Almost always:*** **Significant schedule pressure -- project must be finished in *far* less than "nominal" time.**
- ***Often:*** **Staffing shortages -- project must be done with significantly fewer people than in a "normal" project**
- ***Sometimes:*** **budget limitations -- inadequate funding to pay for people, development tools, and other resources.**
- ***Inevitably:*** **greater risks (>50% chance of failure), more pressure, unpleasant politics**
- ***Almost always:*** ***heavy*** **overtime (more than just 10-20% extra effort), personal sacrifices, increased burnout, higher turnover, lower productivity, lower quality**
- ***Increasingly often:*** **significant corporate consequences (e.g., bankruptcy), lawsuits, *personal* legal liabilities**



Your next assignment



- “Give me an estimate for the XYZ system.
- I think it will take…
- 6 months
- 5 people
- ¥2,500,000
- I need the estimate by the end of the day.”

Your assessment

- ☆ I think it will take…
- ☆ 12 months
- ☆ 10 people
- ☆ ¥10,000,000
- ☆ …but I really need more time for a careful estimate!”

Question for discussion: what if this scenario occurs on <u>every</u> project?

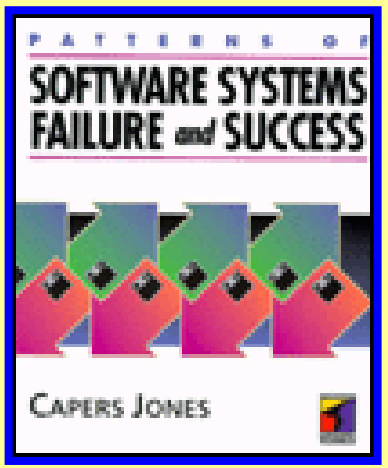
INTRODUCTION :
What's so different about death-march projects?

- ***Users and managers are becoming ever more demanding.*** **Many of today's projects are defined as "mission-critical"**
- ***Many "extreme" projects require BPR to succeed*** **-- just like the early days of client-server projects, during which we learned that 80% of BPR projects were failures.**
- ***Peopleware issues are often exacerbated*** **-- shortage of skills, decreasing company loyalty, inclusion of people with no formal IT background ("testing? what's that?"), employee burnout**
- ***Pace of business demands ever-faster implementation*****. Death-march project schedules tempt project teams to abandon all "process"**
- ***"External" extreme projects are often exposed to much greater risks than before*** **-- e.g., performance problems, reliability, security, privacy, threats of lawsuits, etc.**
- **New technologies are emerging ever faster... e.g., wireless/mobile technologies, Web 2.0 (and beyond), new development environments, etc.**
- **For additional details, see my Aug 2000 *Computerworld* article, "What's So Different About Managing E-projects?"**

Software Project Outcomes

	Early	On-Time	Delayed	Canceled	Sum
1 FP	14.68%	83.16%	1.92%	0.25%	100.00%
10FP	11.08%	81.25%	5.67%	2.00%	100.00%
100FP	6.06%	74.77%	11.83%	7.33%	100.00%
1000FP	1.24%	60.76%	17.67%	20.33%	100.00%
10,000FP	0.14%	28.03%	23.83%	48.00%	100.00%
100,000FP	0.00%	13.67%	21.33%	65.00%	100.00%
Average	*5.53%*	*56.94%*	*13.71%*	*23.82%*	*100.00%*

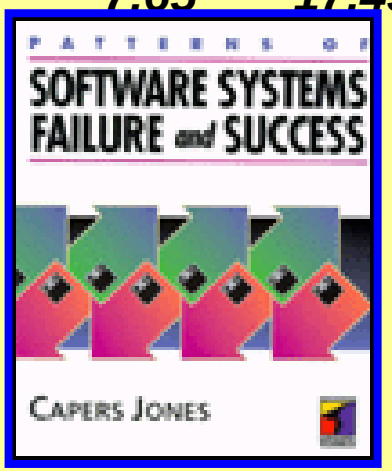
From Capers Jones, *Patterns of Software Systems Failure and Success* (International Thomson Computer Press, 1996)



Software Planned and Actual Schedules in Calendar Months

	Minimum	Actual Average	Maximum	Estimate	Planned Variance	Percent
1 FP	0.06	0.16	0.40	0.15	0.0	16.25%
10FP	0.35	1.07	2.35	1.00	0.07	6.54%
100FP	3.60	10.00	19.00	9.00	1.00	10.00%
1000FP	12.24	27.20	43.52	22.00	5.20	19.12%
10,000FP	24.90	49.80	84.66	36.00	13.80	27.71%
100,000FP	44.28	73.80	132.84	48.00	25.80	34.96%
Average	*14.24*	*27.01*	*47.13*	*19.36*	*7.65*	*17.43%*

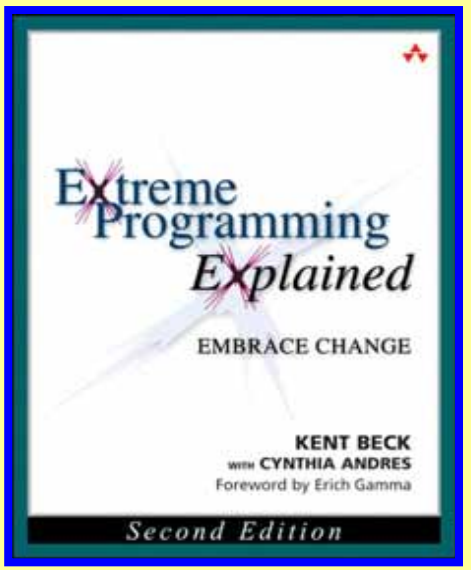
From Capers Jones, *Patterns of Software Systems Failure and Success* (International Thomson Computer Press, 1996)



Note that there is often *enormous* political pressure to produce "politically correct" estimates

1.2 the quality issue

- ☆ Average error rate: approx 1 defect/function point, *after delivery to user.*
- ☆ Variation in error rate: 100X across industry.
- ☆ One reaction to these gloomy statistics: developing new IT technologies, formal methods, and management approaches to enable "zero-defect" IT systems
- ☆ Another approach being hotly debated in U.S. today: "good enough" software, and "good enough" hardware. Examples: Windows 3.1, maybe Windows 2000 -- and the Web, which is always "slightly broken"
- ☆ This leads to related debates between various "process paradigms" -- e.g., the SEI approach versus the "thin method/fat skill" approach exemplified by Kent Beck's XP approach (see his *eXtreme Programming eXplained*)



1.2, cont'd: Typical IT Quality statistics

Defect origins	*Defect potentials*	*Removal efficiency*	*Delivered defects*
Requirements	**1.00**	**77%**	**0.23**
Design	**1.25**	**85%**	**0.19**
Coding	**1.75**	**95%**	**0.09**
Documentation	**0.60**	**80%**	**0.12**
Bad fixes	**0.40**	**70%**	**0.12**
TOTAL	**5.00**	**85%**	**0.75**

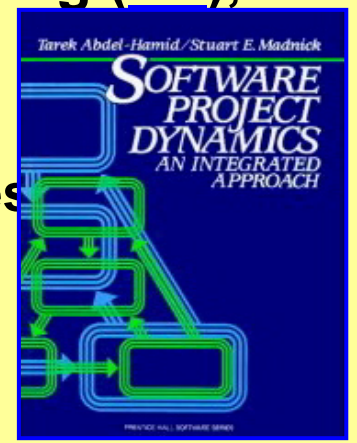
from "Revitalizing Software Project Management," by Capers Jones, *American Programmer*, (now *Cutter IT Journal*) June 1994

What has changed? (*much has not!*)

- ***Economy:*** **no more wild-and-crazy dot-com projects, with sex/drugs/ rock-n-roll, and lots of stock options (except for Web 2.0 startups!)**
- ***September 11:*** **which has launched many new death march projects in an era of sudden, unanticipated, extreme, and occasionally malevolent disruptions**
- ***Outsourcing:*** **I described this in a 1992 book titled *Decline and Fall of the American Programmer*. (ranked #772,793 on Amazon as of 29/01/2007)**
- ***Pervasive access to high-speed Internet:*** **so at least we can collaborate efficiently, and write code while stuck in traffic…**

Managing death-march projects

- ☆ *Not:* tyrannical behavior (which rarely works anyway, and won't be tolerated more than once if people can vote with their feet)
- ☆ *Not:* Charismatic, visionary leadership — though it may be sufficient to accomplish the task.
- ☆ *Compatible with, but not the same as:* extreme programming (XP), agile methods, rapid application development (RAD), etc.
- ☆ *An appreciation that time is the most precious resource*
  - ✓ Avoiding the squandering of time usually raises political issues
  - ✓ Requires appreciation of the *dynamics* of software processes
  - ✓ Must help project team members manage their time effectively
- ☆ *Usually a combination of:*
  - ✓ Savvy "political" behavior — *including the absolute necessity of breaking some rules!! (recent example: Hurricane Katrina)*
  - ✓ Skillful negotiation of deadlines, budget, resources
  - ✓ Appropriate "peopleware" strategies
  - ✓ Appropriate system development processes (SEI-CMM, XP, agile, etc)
  - ✓ Monitoring and controlling *real* progress on a day-by-day basis
  - ✓ Appropriate use of development tools & technologies



2. PROJECT POLITICS

- ☆ Key questions for extreme projects
- ☆ Identifying owners, customers, shareholders, and stakeholders in an extreme software project
- ☆ Stakeholder disagreement
- ☆ Determining the basic nature of the project
- ☆ Levels of commitment

2.1 Key Questions

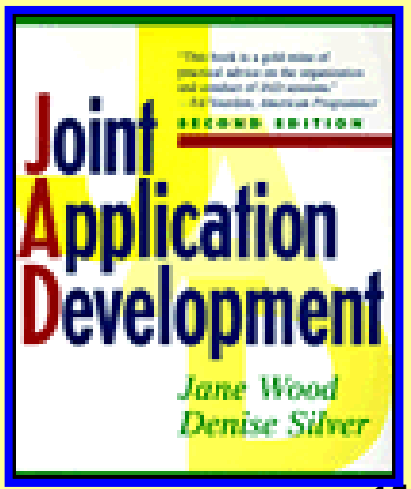
- ☆ **What are the realistic chances for success at various stages?**
  - ✓ On day one of the project -- is it doomed before you even begin? (see "How to Survive a Doomed Project," by Dan Tynan, *Infoworld*, Nov 21, 2005)
  - ✓ At various life-cycle milestones: analysis, "feature freeze," design, "code complete"
  - ✓ Halfway through the project schedule (is the smell of death in the air?)
  - ✓ A month (or a week, or a day) before the deadline
- ☆ ***When* should it become evident to an intelligent observer/participant that the XYZ project has become Project Titanic?**
- ☆ **Who is the person who should *know* whether there's a realistic chance of success, or whether there are serious problems?**
  - ✓ The project manager
  - ✓ The user/customer
  - ✓ Senior management (or will they claim they were innocently clueless? Remember Y2K, and the warnings from attorneys to Boards of Directors!)
  - ✓ External auditors, QA reviewers, consultants, innocent bystanders
- ☆ **What should a reasonably intelligent observer/participant *do* when it becomes obvious that the project is in trouble?**
  - ✓ Work 24 hours a day to "save" the project?
  - ✓ Escalate the problem to higher levels of management?
  - ✓ Gather together key stakeholders to find a consensus solution?

2.2 Identifying the players

- ☆Key point: your chance of success in a death-march project is often zero if you don't know who has the power to declare success or failure (see "Spelling Success," by your humble speaker, *Computerworld*, Feb 19, 2001)
- ☆Also crucial that everyone on the project understands this -- not just the project manager
- ☆Typical players:
  - ✓Owner: who pays for, and/or "accepts" the system?
  - ✓Customer: who will actually use the system?
  - ✓Shareholder: one of many "co-owners"
  - ✓Stakeholder: someone whose interests are affected by the success/failure of the project, and who can affect the outcome of the project — and who may thus become an ally or obstacle to project success. See "Project Clarity Through Stakeholder Analysis," by Larry Smith, *Crosstalk: The Journal of Defense Software Engineering*, Dec 2000
  - ✓The "loser user"
- ☆Note: these roles may not be obvious, and they may shift during the course of the project, if some political event (e.g., H-P's recent boardroom squabble) erupts unexpectedly.

2.3: Stakeholder disagreement

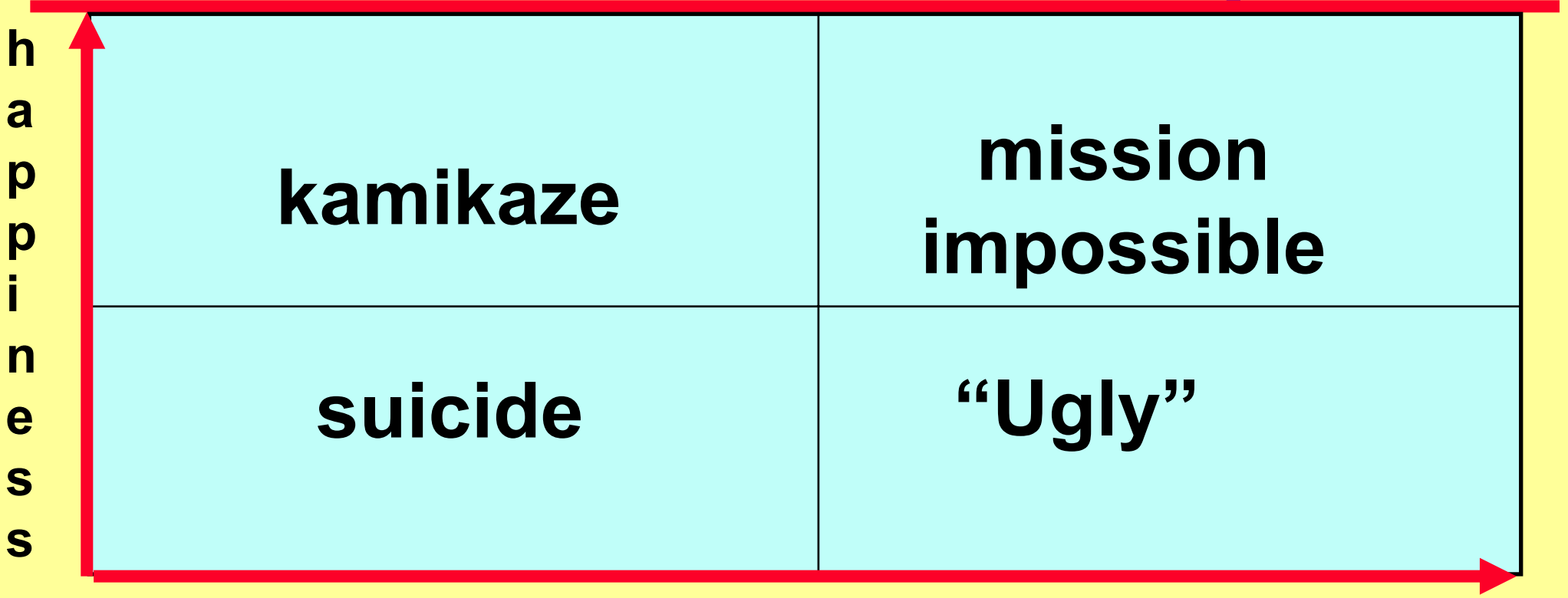
- ***An observation from Tom DeMarco:*** **incomplete, ambiguous specifications are often a sign of irreconcilable stakeholder disagreements, rather than incompetence on the part of systems analysts**
- **In addition, these disagreements invariably cause significant project delays, which imperils the deadline.**
- **Ask for a commitment from stakeholders that all issues requiring their approval or review will be resolved in *no more than 24 hours*. For a good example, see discussion of "extreme software engineering" in the 100-day projects carried out by Tom Love's ShouldersCorp.**
- **Another solution that project manager should try very hard to implement: JAD sessions. (see *Joint Application Development* for good discussion of techniques.)**



2.3, cont'd: more on 24-hour turnaround

- ***Remember: time is likely to be your most precious resource***
- **Key *political* strategy: document and publish day-for-day slippage when key decisions are delayed.**
- **Crucial to recognize the political nature of this behavior, and the fact that it's a response to a political phenomenon.**
- **If this behavior is declared politically unacceptable, then it's nevertheless crucial to identify it as a key risk factor for the overall project**
- **"The surest poison is time." Ralph Waldo Emerson**

2.4 Determining the Basic Nature of the Project



- ☆ Get the project team members to indicate where they think *they* fit into this grid.
- ☆ Remember: public assurances from executive sponsor may not reflect the "reality" of the situation
- ☆ Also remember that the situation can change dynamically, based on politics, external business conditions, government regulations, etc.

2.5 Levels of Commitment

- The parable of the chicken and the pig
- Remember that different members of the “key players” have different levels of commitment...
- ...some of which may be publicly stated, and some of which may not
- ...and all of this may change on a daily basis
- A reminder: one of the primary reasons we succeeded with Y2K is that (for the first time!) we actually had *substantial* commitment from the CEO and the Board of Directors; does the same level of commitment exist for typical extreme projects?

3. PROJECT NEGOTIATIONS

- ☆ Managing project definition at the beginning of the project
- ☆ Using project definition to manage requirements creep
- ☆ Estimating techniques
- ☆ Tools for assisting estimation process
- ☆ Tradeoffs between schedule, budget, staff, quality
- ☆ Tools for rational negotiation
- ☆ Documenting political issues and decisions
- ☆ What to do when rational communications are impossible

3.1 Managing Project Definition: What does "success" mean?

- Many projects succeed/fail on day 1, before any technical work is done.
- *Fundamental requirement:* identifying who has the right to declare "success" — owner, project manager, auditor, end-user, etc.
- Typical definitions of "success"
  - ✓ JFK, 1961: "We will put a man on the moon by the end of the decade, and bring him back alive."
  - ✓ Finishing on time — or at least "reasonably close" to on-time
  - ✓ Staying within budget — or at least not too far above the budget
  - ✓ Delivering required functionality, or at least a tolerable % of "must-have" features
  - ✓ Providing "good enough" level of quality, in terms of "show-stopper" bugs and ability to resume availability quickly enough after a crash
  - ✓ Providing adequate throughput/capacity/response-time
  - ✓ Getting the next round of VC funding, or launching the IPO
- Key indicators of a doomed project:
  - ✓ Nobody can articulate what "success" means for this project…
  - ✓ …or it's such a vague definition that it will be hard to demonstrate success
  - ✓ Key stakeholders cannot reach agreement on definition of success
- Combination of these constraints may prove impossible —so success often depends on agreement as to which constraints can be compromised or relaxed.
- *Biggest risk*: lack of realistic triage at beginning of project. (See my *Computerworld* article, "The Value of Triage")

3.1 More on "good enough"

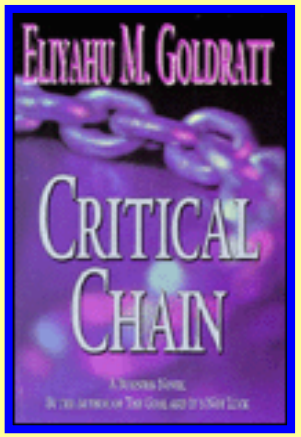
- **General concepts**
  - Familiar concepts of Pareto Principle (the "80-20 rule") are often ignored
  - ***Remember:*** **"Triage" is *not* the same as "prioritization"**
  - ***Early*** **triage is crucial**
  - Documentation of good-enough standards is crucial
- **Functionality**
  - Critical: without these functions, system can't be used at all
  - Important: without these functions, there will be substantial cost/problems
  - Desirable: without these functions, users will whine and complain
- **Quality**
  - Defects by severity
  - Mean Time To Repair (MTTR) by severity level
  - Credible evidence of "convergence" to an acceptable level of quality
- **Performance (response time, throughput, volume, capacity, etc.)**
  - Critical: failing this level means that daily/weekly workload can't be done
  - Important: failing this level means significant loss of productivity/capacity
  - Desirable: failing this level creates a noticeable, annoying impact

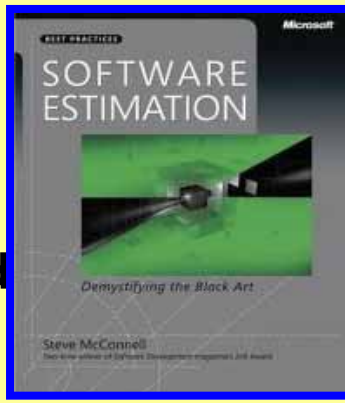
3.2 Using Project Definition to Manage Requirements Creep

- ☆ Typical behavior in projects: new requirements are added at the rate of 1% per month (see this website for more discussion)
- ☆ Requirements "creep" and requirements "churn" are a major element of project management risk -- particularly in death-march projects that often involve fundamental changes to business processes.
- ☆ But if you don't have a formal document describing the requirements, it's hard to identify creep or churn.
- ☆ Assuming that you do have such a document, you need to use it to negotiate schedule/budget/staff modifications if the requirements change or increase.
- ☆ Biggest risk of all: an ambiguous spec is usually a sign of unresolved conflict between diverse political camps in the user community.
  - ✓ Related risk: techies assume that it's their fault they can't understand ambiguous spec.
  - ✓ For more about this, see my *Computerworld* article, "How to Help Users Write Good Requirements Statements"

3.3 Estimating Techniques

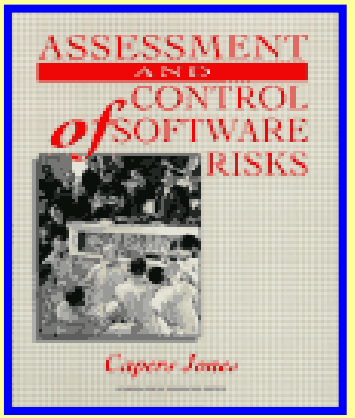
- ☆Fundamental truth: to estimate a project you need metrics from previous projects. Most software organizations have almost no metrics about previous death-march projects (because there are no survivors).
- ☆Thus, most of what's described as "estimating" is either "guessing" or "negotiating" (see "Metrics and the Seven Elements of Negotiation," by Michael Mah, *IT Metrics Strategies*)
- ☆Political reality: estimates are produced by people with little prior estimating experience, and who have a vested interest in believing their optimistic predictions
- ☆A radical suggestion: create a separate estimating group whose work is judged and rewarded by the accuracy of its estimates, not the political acceptability of estimates
- ☆For complex projects, get a commercial estimating tool (typically, less than 5% of IT projects have access to such tools!!)
- ☆For a new approach to estimating, see "critical chain" approach, which "pools" safety estimates across several tasks within a project. See Eliyahu M. Goldratt's *Critical Chain* (North River Press, 1997), and Steve McConnell's *Software Estimation: Demystifying the Black Art* (Microsoft Press, 2006)





3.3 cont'd: more estimating guidelines

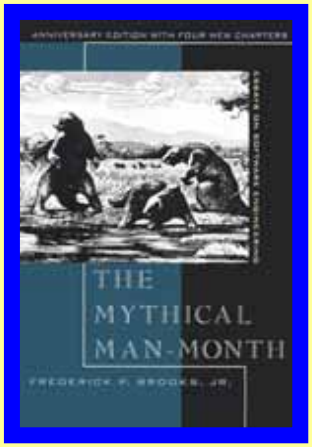
- ***Avoid single-person estimation*** **-- estimates should be undertaken by more than one person, and preferably in a team-based process.**
- ***Always complete a risk assessment prior to estimation*** **-- any estimate undertaken without a formal risk assessment is almost certain to be incorrect.**
- ***Be sure you have taken into account the "Top 10 Project Factors" (from Assessment and Control of Software Risks*****, by Capers Jones):**
  - ✓ **Team skills and experience**
  - ✓ **Level and quality of executive sponsorship**
  - ✓ **Level and quality of stakeholder "buy-in"**
  - ✓ **System or product size**
  - ✓ **System or product complexity**
  - ✓ **Required product quality**
  - ✓ **Development tools and environment**
  - ✓ **Level of innovation required**
  - ✓ **Product requirement stability**
  - ✓ **Schedule constraint (sometimes known as schedule compression)**



- ***Where possible, use relevant (unbiased, uninvolved) experts*** **-- who can give you a confirmation or a warning about the "reasonableness" of your estimates**
- ***Always carefully document estimating assumptions*** **-- especially because some of these assumptions will be based on political "assurances" that are never written down, and subject to change.**
- ***Review the work breakdown structures*** **-- to make sure you have not overlooked such project tasks as education, documentation, etc.**

3.5 Tradeoffs between schedule, budget, functionality, staff, quality

- ☆Biggest risk: tradeoffs are usually negotiated, under pressure, late in the project schedule — without accepting (or even acknowledging!) the non-linear tradeoffs involved...
- ☆...and without accepting the reality that much of the partially-finished work will be scrapped forever
- ☆To negotiate tradeoffs rationally, you need to have one of the commercial estimating tools mentioned earlier
- ☆For more details, see "Internet-Speed Deadline Management: Negotiating the Three-Headed Dragon," by Michael Mah, *IT Metrics Strategies*
- ☆Key point: it's not a linear tradeoff — see Fred Brooks, *The Mythical Man-Month* (20th anniversary edition, Addison-Wesley, 1995)
- ☆Relationship is a non-linear, third-order polynomial relationship — see Larry Putnam and Ware Myers, *Measures for Excellence: Reliable Software on Time, Within Budget* (Prentice-Hall, 1992)

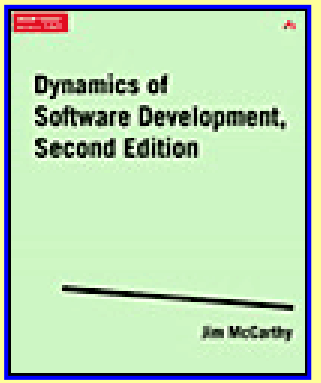


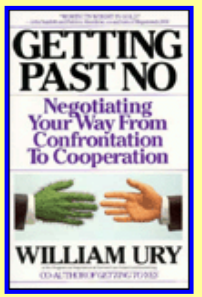
3.4 Negotiating Games

- ☆Doubling and add some...
- ☆Reverse doubling
- ☆Guess the Number I'm Thinking of...
- ☆Double Dummy Spit
- ☆The X-Plus Game
- ☆Spanish Inquisition
- ☆Low Bid
- ☆Gotcha — throwing good money after bad
- ☆Chinese Water Torture
- ☆Smoke and Mirrors/Blinding with Science
  - ✓thanks to Rob Thomsett, "Double Dummy Spit, and Other Estimating Games," American Programmer (now *Cutter IT Journal*), June 1996

3.6 Negotiating strategies

- ☆Don't get tricked into making an "instant estimate" — ask for time to think about (a week, a day, even an hour)
- ☆State the estimate in terms of confidence levels, or ± ranges, etc.
- ☆Jim McCarthy (formerly of Microsoft, author of *Dynamics of Software Development*): make the customer, or other members of the organization, share some of the uncertainty.
- ☆Project manager: "I don't know precisely when we'll finish — but I'm more likely to be able to figure it out than anyone else in the organization. I promise that as soon as I have a more precise estimate, I'll tell you right away."
- ☆Do some reading and research to become better at negotiating, e.g.:
  - ✓ *Bargaining for Advantage: Negotiating Strategies for Reasonable People*, by G. Richard Shell (reissue edition, Penguin Books, June 2000)
  - ✓ *Getting Past No: Negotiating Your Way from Confrontation to Cooperation*, by William Ury (Bantam Doubleday Dell, 1993)
  - ✓ *The Prince*, by Niccolo Machiavelli (Viking Penguin, 1999)



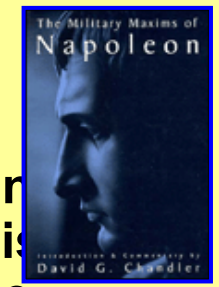


3.7 Documenting political items

- Document all of these key issues, as part of the permanent record -- otherwise, you're likely to find that people "forget" unpleasant realities that you told them about when the project began
- Or even worse, the people who made commitments to you about key political issues may have quit, died, retired, been fired, or gotten run over by a crazy Tokyo taxi driver.
- Key things to document (probably in a "project plan" document)
  - Background of the project (why is this project being carried out?)
  - Scope (what to leave in, what to leave out; use a context diagram for simplicity)
  - Technical assumptions
  - Technical dependencies
  - Constraints
  - Project approach (what kind of development process will be used)
  - Quality approach
  - Development plan (resource schedule, project schedule, release schedule)

3.6 What to do when rational negotiation breaks down

- ☆Remember Nancy Reagan's advice: "Just say no!"
  - ✓Would you commit to running a marathon in one hour? Even if you're Lance Armstrong (who finished the 2006 NYC Marathon in 2:59:35)?
- ☆Quit (the project or the company)
- ☆Appeal to a higher authority
- ☆Go see the movie *Gladiator*, and learn to say, like Russell Crowe, "We who are about to die salute you!"
- ☆*Decide which "rules" you need to break in order to achieve an "irrational" set of demands that have been imposed upon you.*
- ☆Redefine the project as a kamikaze, suicide, etc., and make sure entire project team knows it.
- ☆Key point: project leader has to believe in the possibility of achieving project goals -- and must be able to convince team members without "conning" them
- ☆"It follows that any commander in chief who undertakes to carry out a plan which he considers defective is at fault; he must put forth his reasons, insist on the plan being changed, and finally tender his resignation rather than be the instrument of his army's downfall." See Napoleon, *Military Maxims and Thoughts*; also *The Military Maxims of Napoleon*, by William E. Cairnes (De Capo Press, 1995



4. PEOPLEWARE ISSUES

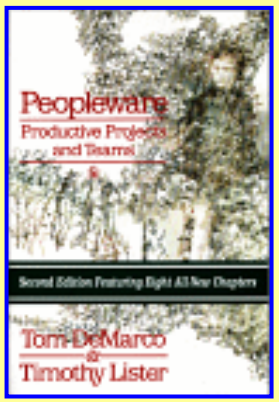
- ☆Hiring and staffing issues
- ☆Recruiting
- ☆Identifying loyalty and commitment issues
- ☆Team-building issues

4.1 Identifying Loyalty and Commitment

- ☆ Ideal case (from the manager's short-term perspective): loyalty to project comes before all else.
- ☆ Often associated with young, unmarried technonerds, with no life outside the office
- ☆ May also depend on charisma of project leader
- ☆ Also depends on length of project: total devotion is feasible for 3-6 month death-march project, but not ≥ 24 months
- ☆ Crucial that everyone knows what the loyalty issues are -- e.g., "If my spouse threatens to divorce me, then I'll have to quit this project."

A thought about motivation

- ***"Motivation" is often just a managerial euphemism for (unpaid) overtime***
- **"There is nothing more discouraging to any worker than the sense that his own motivation is inadequate and has to be 'supplemented' by that of the boss...**
- **You seldom need to take Draconian measures to keep your people working; most of them love their work."**
- **Tom DeMarco and Tim Lister, *Peopleware***



“Voluntary” overtime

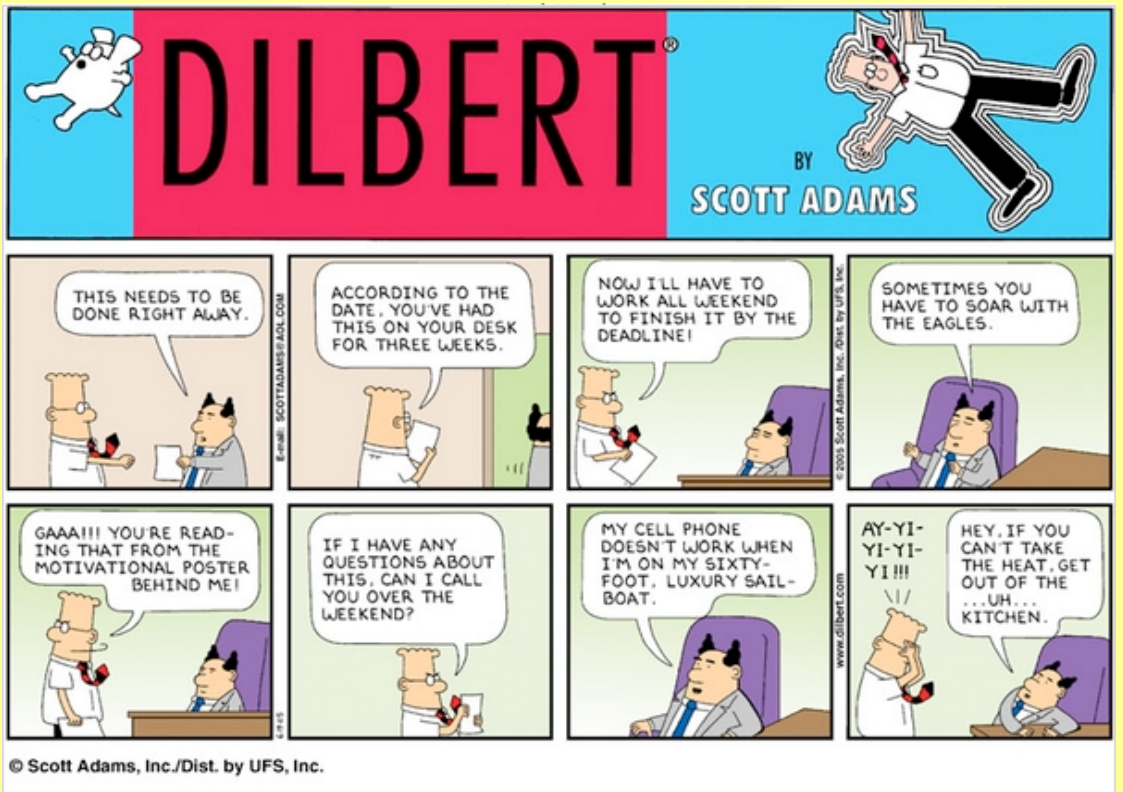
- An inevitable part of many software projects, especially high-intensity death-march projects
- Typically easier to accomplish if team is young, unmarried, antisocial, and passionate about the project. (See "adminspotting" for description of mindset; get adminspotting t-shirts!)
- Also easier to accomplish if overall duration of project is 3-6 months.
- Beware the insidious effects of long-term heavy overtime on projects of 18-24 months, etc.
- Worst mistake is not recording it, so that it appears to be “free” and “infinite”

Signs of an Effective Team

- ***Strong sense of identity:*** **Team names, common "in-jokes," common lunch area, etc.**
- ***Sense of eliteness:*** **Team members feel they are part of something unique.**
- ***Joint ownership:*** **Participants are pleased to have their names grouped together on a product.**
- ***Obvious enjoyment:*** **Do good work and have fun. (Seems obvious, but "having fun" is contrary to corporate culture in many organizations!)**

TEAMICIDE

- ☆Defensive management — not trusting the team
- ☆Bureaucracy — too much paperwork
- ☆Physical separation of team members
- ☆Fragmentation of people's time
- ☆Quality reduction of the product
- ☆Phony deadlines
- ☆Clique control — splitting up teams
- ☆(see "Teamicide revisited," by Tom DeMarco and Tim Lister)

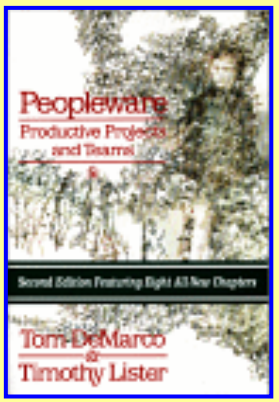


The impact of workplace on quality

- ☆ Workers who say their workplace is acceptably quiet are 1/3 more likely to deliver zero-defect work.
- ☆ 66% of zero-defect workers report noise-level is OK in their workplace
- ☆ 8% of one-or-more-defect workers report noise level is OK in their workplace

Best and Worst Performers in the Coding War Games

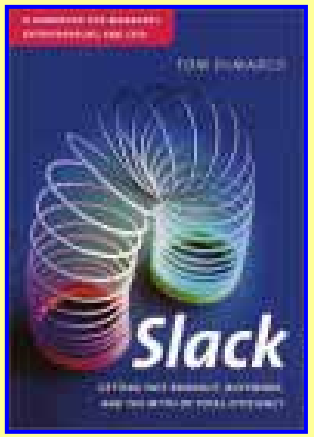
Environmental Factor	*1st Quartile*	*4th Quartile*
1. How much dedicated workspace do you have?	78 sq ft	46 sq ft
2. Is it acceptably quiet?	57% yes	29% yes
3. Is it acceptably private?	62% yes	19% yes
4. Can you silence your phone?	52% yes	10% yes
5. Can you divert your calls?	76% yes	19% yes
6. Do people often interrupt you needlessly?	38% yes	76% yes



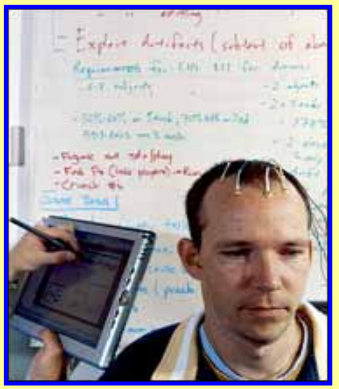
(from a 600-person coding "war game," discussed in Tom DeMarco and Tim Lister, *Peopleware*

Office Space and Interruptions

- Some software development work requires intense, uninterrupted periods of concentration.
- Typical environment for *all* of today's projects: 300 "interruptions" per day!
- Phone calls, etc require a period of "re-immersion" of 5-15 minutes. For a discussion of this, see Tom DeMarco's book, *Slack: Getting past burnout, busywork, and the myth of total efficiency*

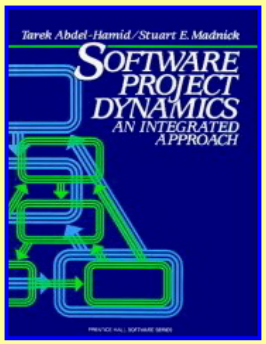


- New info: see "Meet the Life Hackers," from *New York Times*, Oct 16, 2005. Based on 1,000 hours of observations of two West Coast high-tech firms:
  - ✓ "Each employee spent only 11 minutes on any given project before being interrupted and whisked off to do something else. What's more, each 11-minute task was itself fragmented into even shorter three-minute tasks, like answering e-mail messages … And each time a worker is distracted from a task, it takes *25 minutes*, on average, to return to that task."



5. SOFTWARE PROCESSES

- "Heavy" vs. "Light/agile/XP" processes vs. Anarchy: *don't let death-march projects provide the excuse for utter anarchy*. (See *Building the Empire State* for an example of agile processes for building the Empire State Building.)
- "Best practice" and "worst practice" concepts: in a death-march project, it may be more important to avoid *disastrous* processes than to strive for *perfect* processes. (Consider a project "breathalyzer test")
- In a death-march project:



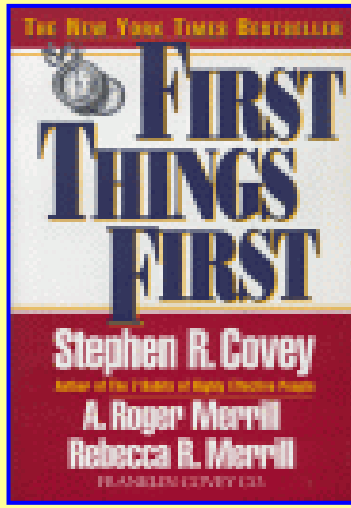
  - ✓ The *dynamics* of processes (delays between activities, feedback loops, etc.) are crucial…
  - ✓ For minimizing the "cycle time" of the overall process (i.e., the duration from beginning of project to the deadline)
  - ✓ So get a copy of iThink, and read Tarek Abdel-Hamid's classic text, *Software Project Dynamics*. Also, watch for the forthcoming publication of Raymond Madachy's *Software Process Dynamics*.
  - ✓ A trivial, but deeply important, observation from Barry Boehm: the most important reason projects *finish* late is that they *start* late.

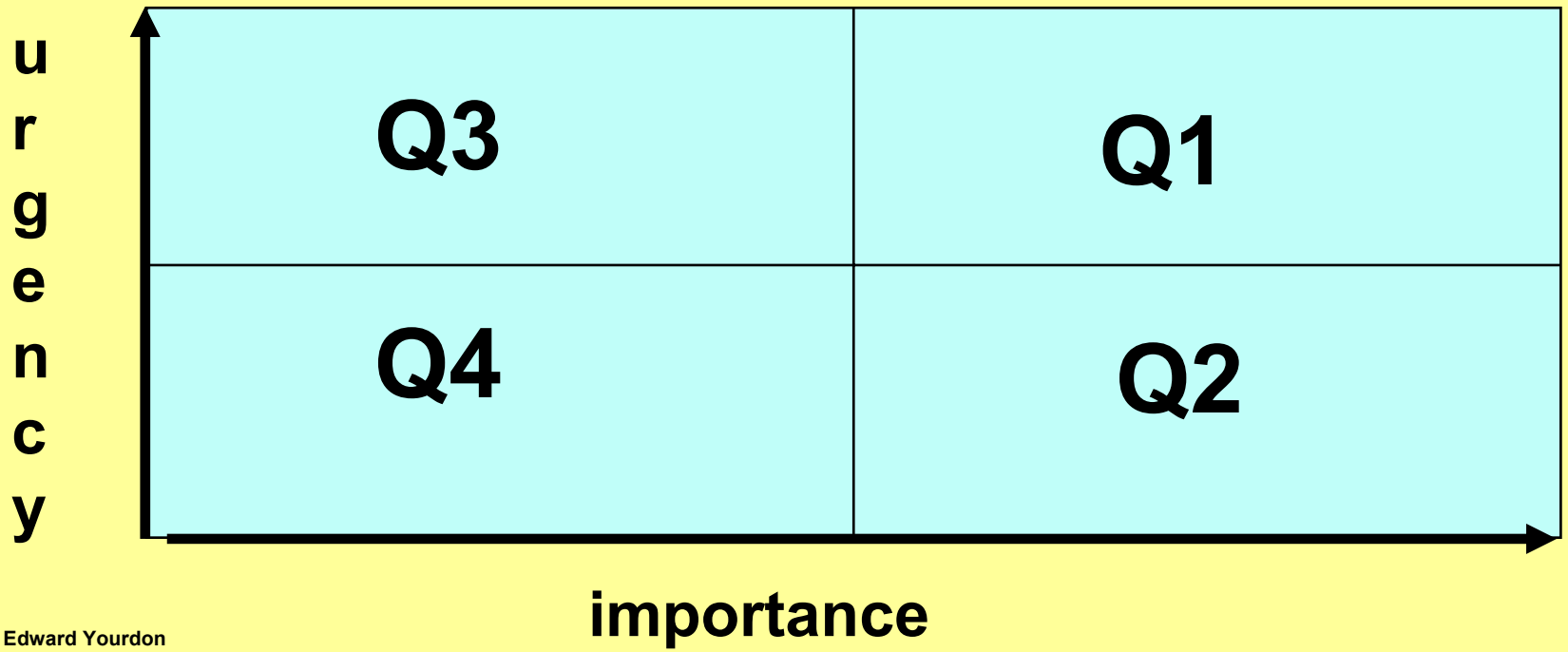
6. MONITORING AND CONTROLLING PROGRESS

- ☆Helping project team members monitor and control their own progress
- ☆Measuring, managing, and controlling progress
- ☆Maximizing project life cycle review

6.1, Helping Project Team monitor its own use of time

- ☆Peter Drucker: "Until we can manage time, we manage nothing else"
- ☆Best reference: *First Things First*, by Stephen R. Covey, et al (Fireside, 1996)
- ☆Key objective for management: help your project team distinguish between urgent and important.
- ☆Project manager should act as a buffer to keep Q3 items from distracting the project team.
- ☆For more details, see my Apr 23, 2001 *Computerworld* article, "*Finding Time to Think*."





6.2 General comments about managing and monitoring progress

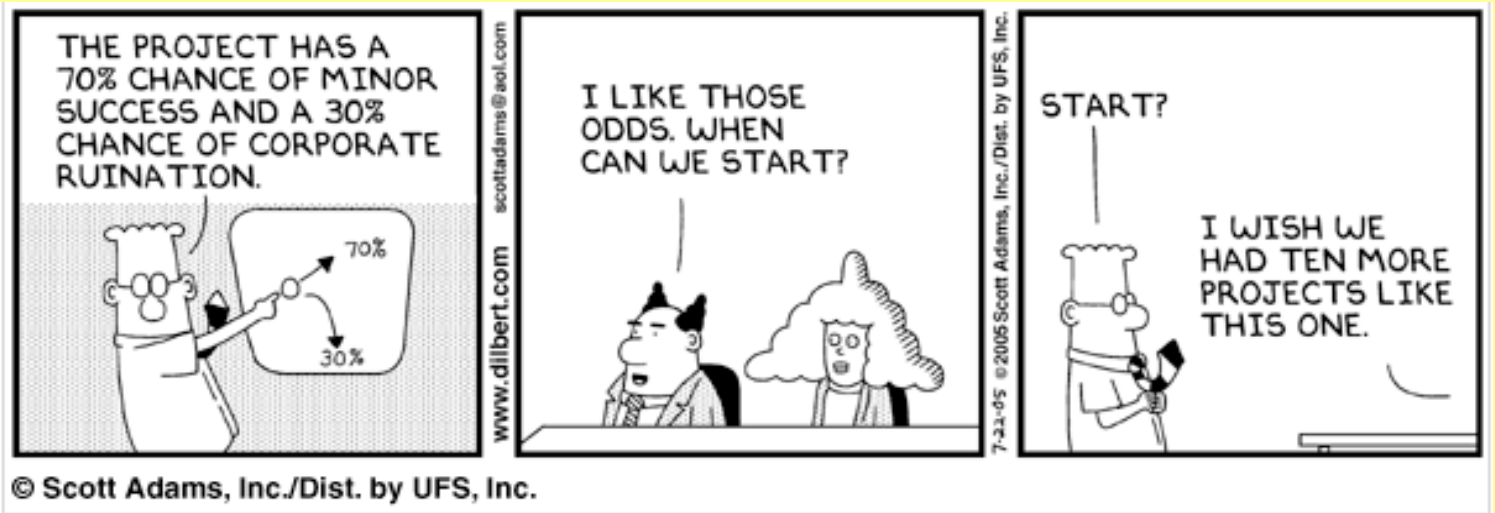
- ☆ Management approaches based on classical waterfall approach are almost certain to fail in large, complex projects
- ☆ Need some kind of "time-boxing" approach based on versions, features, deliverables, etc.
- ☆ Jim McCarthy: "Never let a programmer disappear into a dark room." (The antithesis of this phenomenon: "visible processes.")
- ☆ If team understands what features and dependencies are required for a milestone completion, they will exert their own pressure upon themselves, rather than having the manager beat them up.
- ☆ *If you miss one milestone deadline, it's crucial to succeed on the next one.*
- ☆ Mini post-mortems can be incredibly valuable (but end-of-project postmortems are usually a waste of time).

The “daily build”

- ☆ Popularized by Dave Cutler at Microsoft
- ☆ Jim McCarthy (former head of Microsoft’s Visual C++ project): “The daily build is the heartbeat of the project — it’s how you know you’re alive”
- ☆ Should be automated, and performed overnight — or even more often.
- ☆ Various “tricks” can be used to increase its effectiveness
  - ✓ Punishing people who “break” the daily build
  - ✓ Using red-flag/green-flag at office entrance

6.3 Risk Management

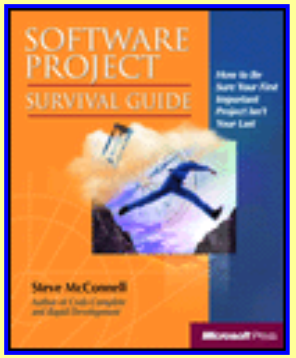
- ☆ Introduction and definition
- ☆ The context of risk management
- ☆ Critical cultural aspects
- ☆ Risk assessment vs. risk control
- ☆ Likely causes of risk in software projects



Likely Causes of Software Risks

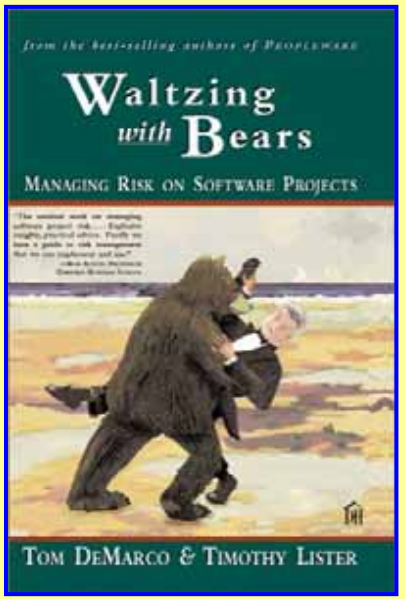
(According to Steve McConnell, author of *Software Project Survival Guide)*

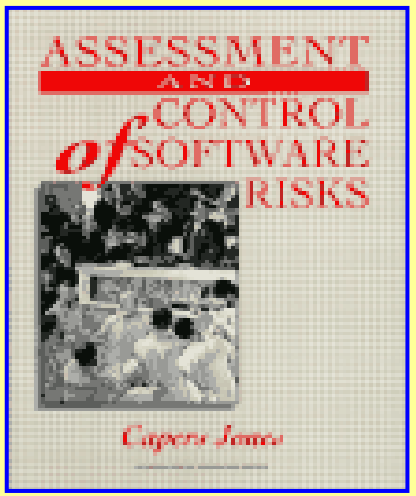
1. Poorly-defined or changing requirements
2. Poor project planning/management
3. Uncontrolled quality problems
4. Unrealistic expectations/inaccurate estimates
5. Naive adoption of new technology

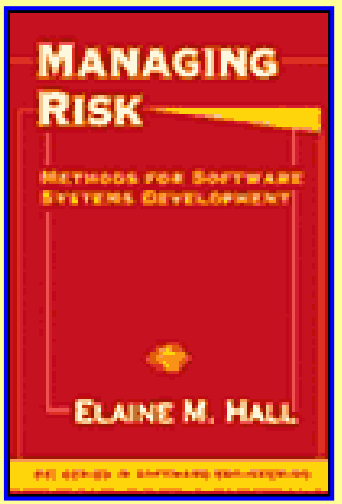


Risk Management books

- ☆See *Waltzing with Bears*, by Tom DeMarco and Tim Lister (Dorset House, 2003)
- ☆See *Assessment and Control of Software Risks*, by Capers Jones (Prentice Hall, 1994)
- ☆See also *Managing Risk: Methods for Software Systems Development*, by Elaine M. Hall (Addison-Wesley, 1998)







Cultural Issues

- As Tom DeMarco points out, an organization's "true" approach to risks, and risk management, are often illustrated by some of the "folklore" that is not written down, but is an important part of the culture:
- "He who identifies a risk is responsible for solving it."
- "We don't have risks in our organization, we just have opportunities."
- "Don't tell me about a problem unless you've got a solution to the problem."
- "We don't lie, but we tell the truth slowly."
  - A phrase attributed to President Clinton's press secretary, Mike McCurry
  - See this article for an explanation by subsequent presidential Press Secretary Joe Lockhart about *why* McCurry felt obliged to tell the truth slowly.
  - Hint: McCurry made his comment in the midst of the Monica Lewinsky scandal, when he had to choose his words carefully — perhaps because he felt that President Clinton wasn't telling *him* the truth.

Words to live by in a death-march project

"I wake up each morning determined to change the World ...

...and also to have one hell of a good time.

Sometimes that makes planning the day a little difficult."

E.B. White

found in the opening of the preface of

***Succeeding with Objects*,**

by Adele Goldberg and Kenneth S. Rubin

(Addison-Wesley, 1995)

Death March projects

Ed Yourdon

email: ed@yourdon.com

web: http://www.yourdon.com

blog: http://www.yourdon.com/blog

JaSST keynote presentation

January 30, 2007