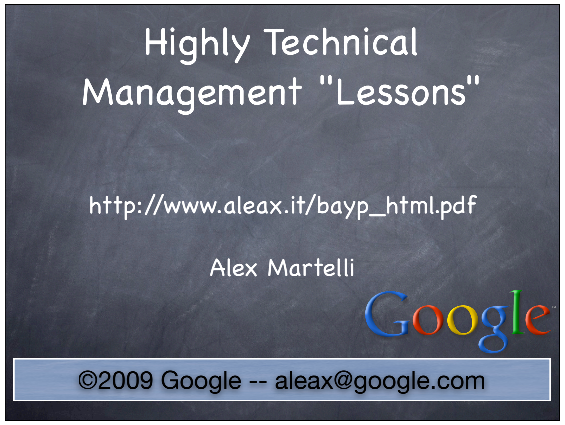
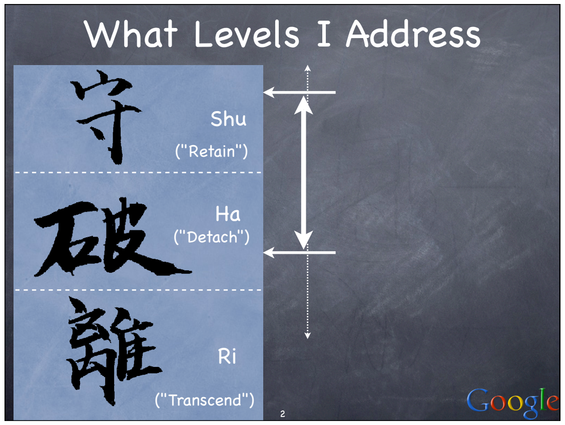


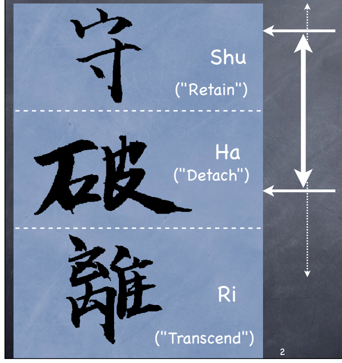
Highly Technical Management "Lessons"

http://www.aleax.it/bayp_html.pdf

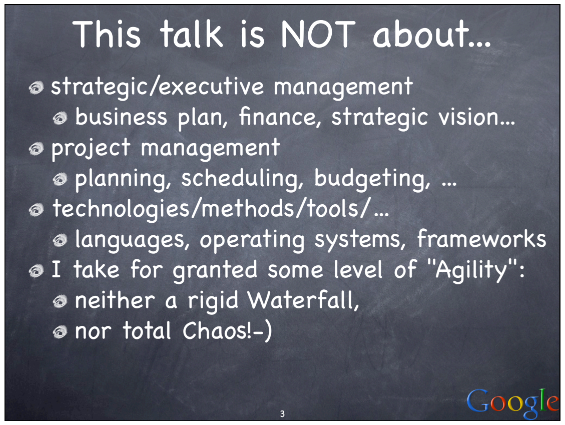
Alex Martelli
Google



What Levels I Address



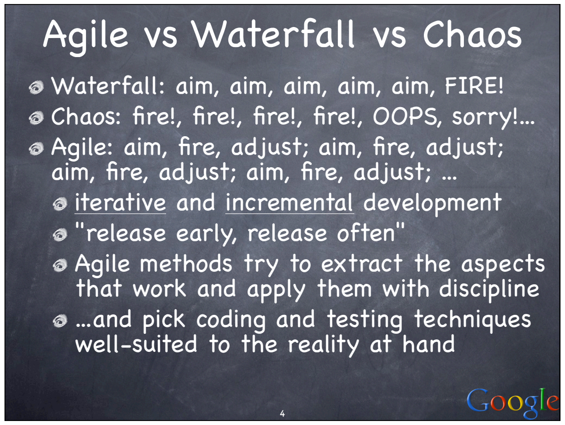
Shu: where you're "learning by rote", ONE technique, action by action
Ha: where you're comparing/contrasting techniques, picking the right one case by case
Ri: where you leave "book learning" behind, create your own techniques



This talk is NOT about...

- strategic/executive management
  - business plan, finance, strategic vision...
- project management
  - planning, scheduling, budgeting, ...
- technologies/methods/tools/...
  - languages, operating systems, frameworks
- I take for granted some level of "Agility":
  - neither a rigid Waterfall,
  - nor total Chaos!-)

A crucial tip: always clarify your NON-goals up front!-)

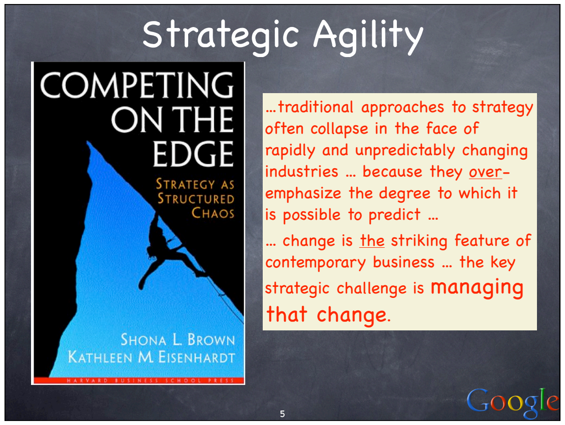


Agile vs Waterfall vs Chaos

- Waterfall: aim, aim, aim, aim, aim, FIRE!
- Chaos: fire!, fire!, fire!, fire!, OOPS, sorry!...
- Agile: aim, fire, adjust; aim, fire, adjust; aim, fire, adjust; aim, fire, adjust; ...
  - iterative and incremental development
  - "release early, release often"
  - Agile methods try to extract the aspects that work and apply them with discipline
  - ...and pick coding and testing techniques well-suited to the reality at hand

4

yes, there _are_ degrees of "agility" (but no: Chaos isn't one such degree -- it's a "falling of the edge":-).



Strategic Agility

COMPETING ON THE EDGE

STRATEGY AS STRUCTURED CHAOS

SHONA L BROWN
KATHLEEN M EISENHARDT

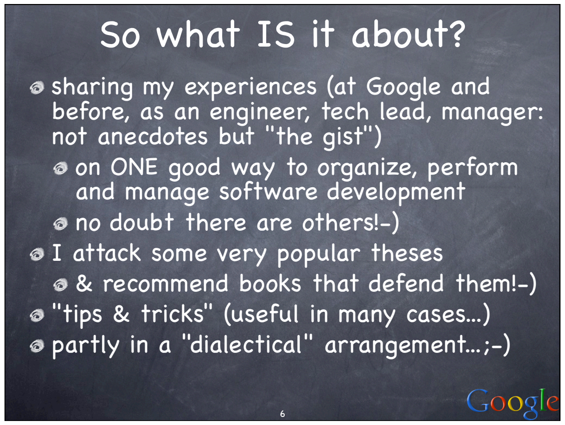
HARVARD BUSINESS SCHOOL PRESS

...traditional approaches to strategy often collapse in the face of rapidly and unpredictably changing industries ... because they over-emphasize the degree to which it is possible to predict ...

... change is the striking feature of contemporary business ... the key strategic challenge is managing that change.

5

Shona was a partner at McKinsey (leader of Global Strategy Practice) when she wrote this book -- soon after she joined Google, where she's now Senior Vice President for Business Operations. Kathleen is Professor of Strategy and Organization at Stanford.

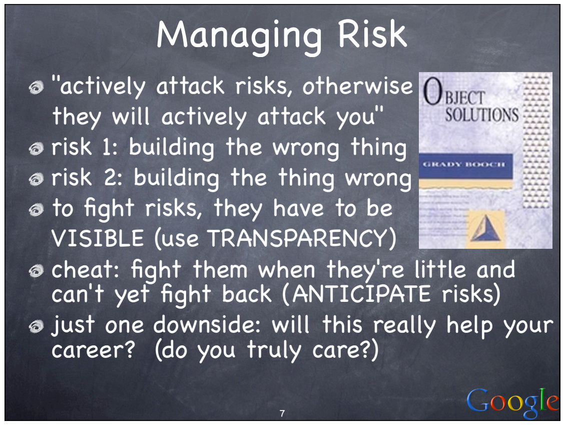


So what IS it about?

- sharing my experiences (at Google and before, as an engineer, tech lead, manager: not anecdotes but "the gist")
  - on ONE good way to organize, perform and manage software development
  - no doubt there are others!-)
- I attack some very popular theses
  - & recommend books that defend them!-)
- "tips & tricks" (useful in many cases...)
- partly in a "dialectical" arrangement...;-)

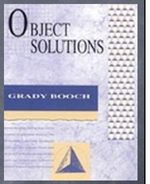
6

Secondary but helpful tips: clarify your goals (once the NON-goals are sharply set;-).

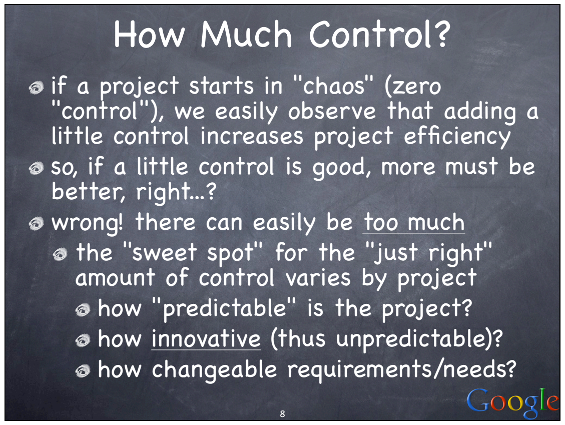


Managing Risk

- "actively attack risks, otherwise they will actively attack you"
- risk 1: building the wrong thing
- risk 2: building the thing wrong
- to fight risks, they have to be VISIBLE (use TRANSPARENCY)
- cheat: fight them when they're little and can't yet fight back (ANTICIPATE risks)
- just one downside: will this really help your career? (do you truly care?)



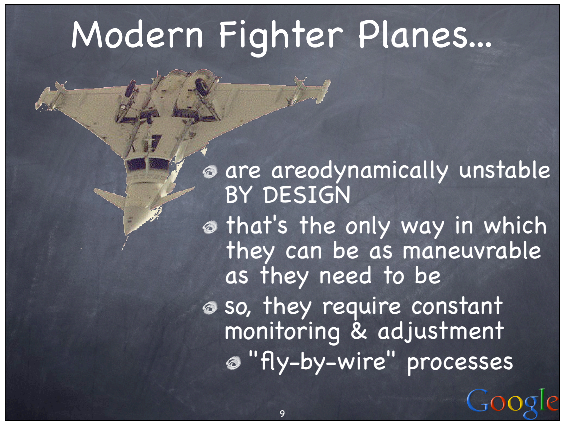
We often reward "heroic effort" -- and not the foresight and planning that make the heroism unnecessary and offer much better ROI. Bosses and customers used to be shielded from problems and risks may freak if you use transparency to display all risks and problems with your projects, even though this puts them in control and maximizes chances of success and quality. Do you care about being rewarded and praised, or are you out for Self-Actualization and Transcendence? (See Manslow Pyramid later)



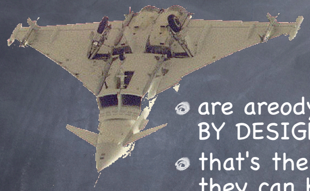
How Much Control?

- if a project starts in "chaos" (zero "control"), we easily observe that adding a little control increases project efficiency
- so, if a little control is good, more must be better, right...?
- wrong! there can easily be too much
  - the "sweet spot" for the "just right" amount of control varies by project
    - how "predictable" is the project?
    - how innovative (thus unpredictable)?
    - how changeable requirements/needs?

This truth is "fractal" -- applies (in different but similar ways) at all levels, from projects to huge firms.

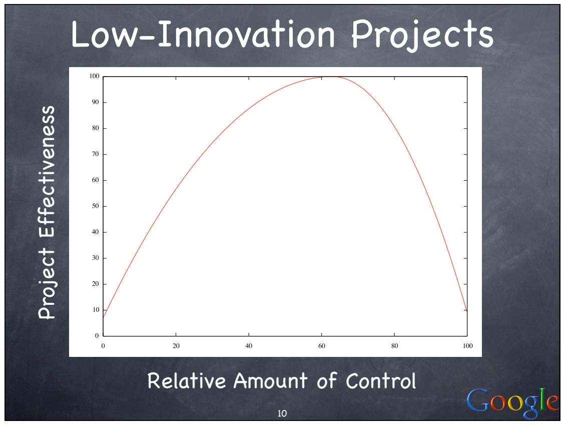


Modern Fighter Planes...



- are areodynamically unstable BY DESIGN
- that's the only way in which they can be as maneuvrable as they need to be
- so, they require constant monitoring & adjustment
  - "fly-by-wire" processes

AKA, "the price of [maneuverability and fast response to change] is constant vigilance".

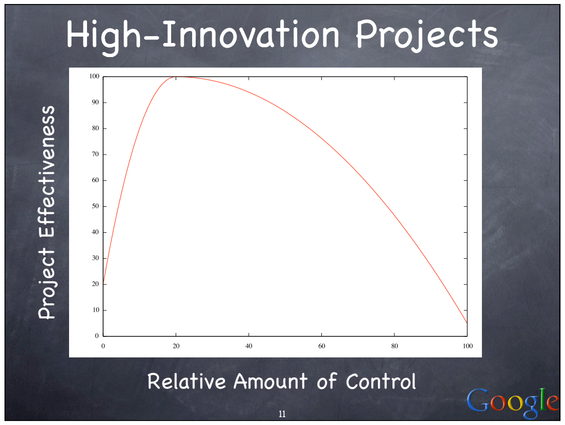


Low-Innovation Projects

Google

10

Again, not just "projects" -- conceptually applies to any endeavor and to whole organizations.



High-Innovation Projects

Google

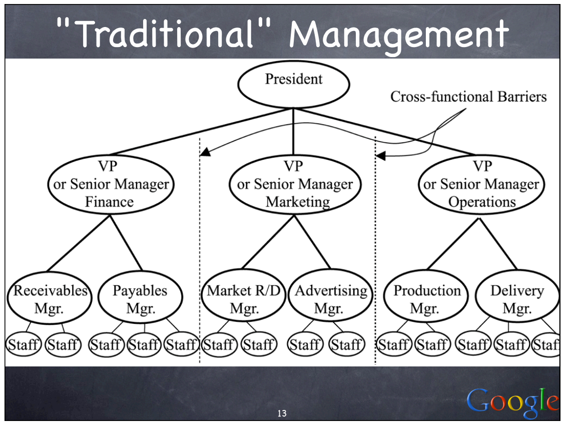
11

Who IS a Manager?

Google

12

I actually picked only managers I respect and appreciate for this slide (well, not the PHB, I guess, but he does give me MANY laughs:-). The lower-left corner knight is Swedish King Gustavus Adolphus. The football guys on the right are Italy's Word Champion team from 2006 -- the one holding the cup is Cannavaro, the playing captain; in the top left is Lippi, the non-playing manager/trainer of the winning team, holding the same cup later. Iacocca, Mayer, Jobs, Cox, Hopper, hopefully need no intro:-)



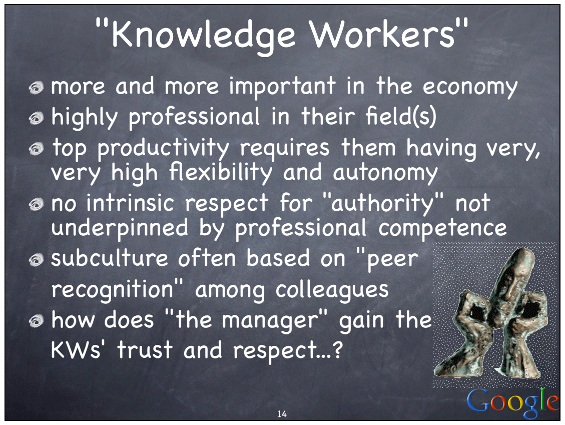
"Traditional" Management

President
Cross-functional Barriers
VP or Senior Manager Finance
VP or Senior Manager Marketing
VP or Senior Manager Operations
Receivables Mgr.
Payables Mgr.
Market R/D Mgr.
Advertising Mgr.
Production Mgr.
Delivery Mgr.
Staff

13

Google

'nuff said...



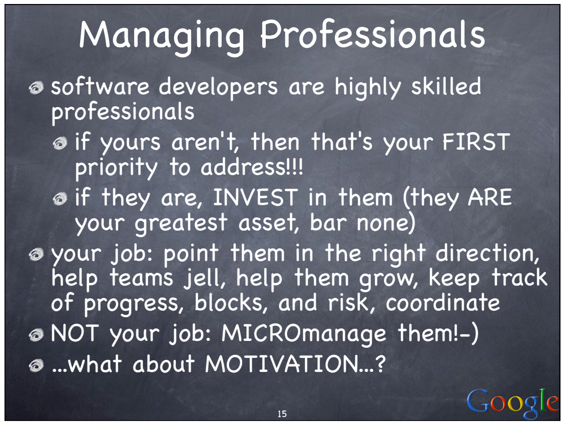
"Knowledge Workers"

- more and more important in the economy
- highly professional in their field(s)
- top productivity requires them having very, very high flexibility and autonomy
- no intrinsic respect for "authority" not underpinned by professional competence
- subculture often based on "peer recognition" among colleagues
- how does "the manager" gain the KWs' trust and respect...?

14

Google

It's a new term, but the concept is very old (though it accounts for a growing fraction of the economy as time goes by). Given key workers who live in a subculture where technical competence is the main determinant of status, how does the manager gain their trust and respect? (This is known in the trade as "foreshadowing":-).



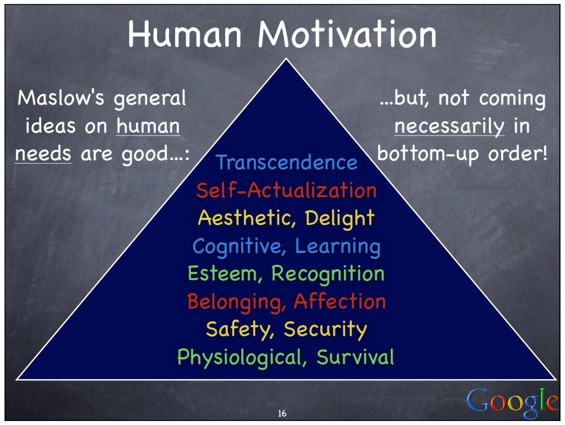
Managing Professionals

- software developers are highly skilled professionals
  - if yours aren't, then that's your FIRST priority to address!!!
  - if they are, INVEST in them (they ARE your greatest asset, bar none)
- your job: point them in the right direction, help teams jell, help them grow, keep track of progress, blocks, and risk, coordinate
- NOT your job: MICROmanage them!-)
- ...what about MOTIVATION...?

15

Google

Hiring well is VERY difficult (I've heard good things about Joel's latest book, "Smart and Get Things Done", which is right on the subject, but haven't read it myself yet at the time of writing). Removing from the team/organization people that are wrong for it is, or SHOULD be, even harder (you ARE messing with another person's life -- if that doesn't make you shudder, I'm worried), but nevertheless crucial for the success of the project, team, firm, etc.



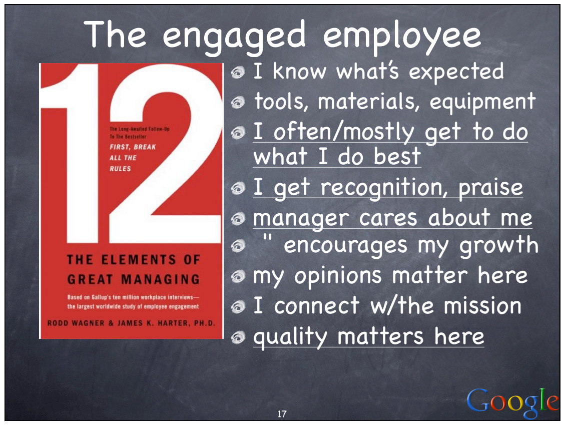
Human Motivation

Maswlow's general ideas on human needs are good...:

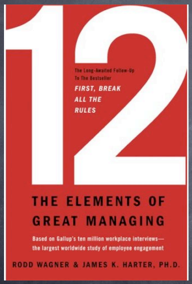
...but, not coming necessarily in bottom-up order!

- Transcendence
- Self-Actualization
- Aesthetic, Delight
- Cognitive, Learning
- Esteem, Recognition
- Belonging, Affection
- Safety, Security
- Physiological, Survival

Most places quote the simpler, older 5-level version, but I'm very keen on this one (from Maslow's later works). "Transcendence"'s main mainfestation is "helping others *for the sake* of helping others" (not for reciprocity, belonging, to get praise, etc), and I concur it IS the highest peak of human achievement, beyond even self-actualization. As for order, it's obviously NOT linear -- e.g., soldiers often risk their lives (survival and safety levels) for the sake of their "unit" (belonging, recognition), etc.

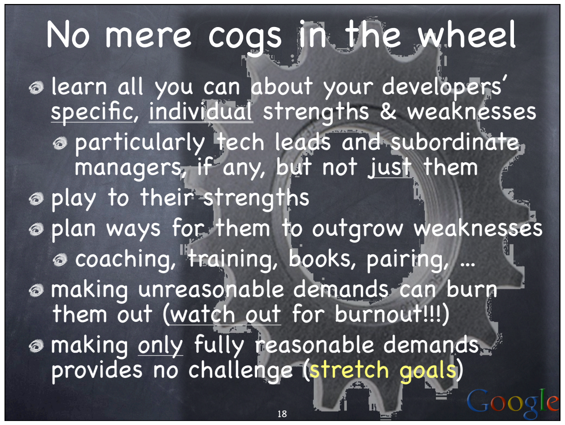


The engaged employee


12
The Long-Awaited Follow Up To The Bestseller
FIRST, BREAK ALL THE RULES
THE ELEMENTS OF GREAT MANAGING
Based on Gallup's ten million workplace interviews—the largest worldwide study of employee engagement
RODD WAGNER & JAMES K. HARTER, PH.D.

- I know what's expected
- tools, materials, equipment
- I often/mostly get to do what I do best
- I get recognition, praise
- manager cares about me
- " encourages my growth
- my opinions matter here
- I connect w/the mission
- quality matters here

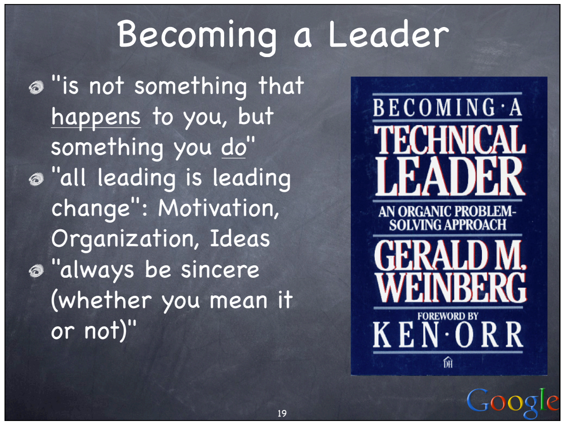
The exact wording of the 12 principles is claimed as copyright by the book's authors, so I've paraphrased the ideas behind them (ideas can't be copyrighted, thanks be!) -- actually, only the nine out of 12 that I think matter most in this context (the authors correctly point out how difficult it is to fit considerations of _compensation_ in there!). Exercise: match each element to the relevant points in Maslow's Pyramid.



No mere cogs in the wheel

- learn all you can about your developers' specific, individual strengths & weaknesses
  - particularly tech leads and subordinate managers, if any, but not just them
- play to their strengths
- plan ways for them to outgrow weaknesses
  - coaching, training, books, pairing, ...
- making unreasonable demands can burn them out (watch out for burnout!!!)
- making only fully reasonable demands provides no challenge (stretch goals)

Managing professionals as if they were "interchangeable parts" is the single worst source of management mistakes. Professionals (good ones, especially) thrive on challenge -- but you have to balance that (stretch goals) with avoiding their burning out.

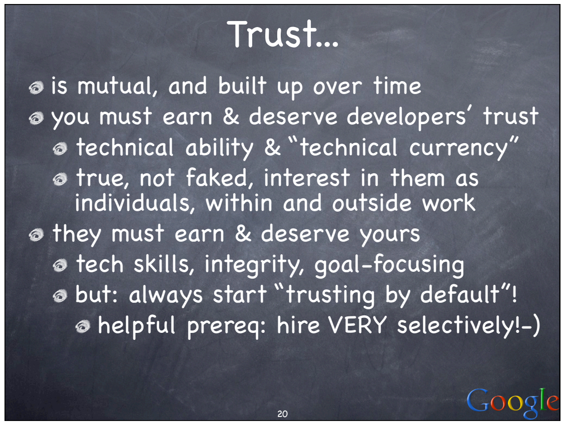


Becoming a Leader

- "is not something that happens to you, but something you do"
- "all leading is leading change": Motivation, Organization, Ideas
- "always be sincere (whether you mean it or not)"

BECOMING A TECHNICAL LEADER
AN ORGANIC PROBLEM-SOLVING APPROACH
GERALD M. WEINBERG
FOREWORD BY KEN ORR

The third point is obviously a funny quip -- but there's depth behind it, just like there is behind every funny anecdote and homily in this excellent book. A book that can easily be appreciated at any level of experience but will give you more and more in proportion to what you have to start with, and what you put into reading, studying, and applying it in your life.

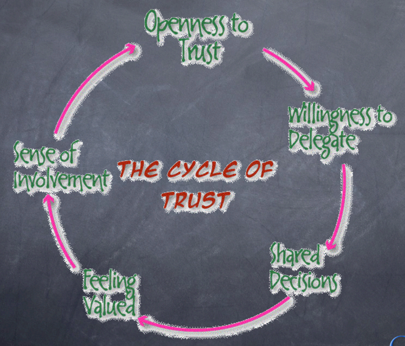


Trust...

- is mutual, and built up over time
- you must earn & deserve developers' trust
  - technical ability & "technical currency"
  - true, not faked, interest in them as individuals, within and outside work
- they must earn & deserve yours
  - tech skills, integrity, goal-focusing
  - but: always start "trusting by default"!
    - helpful prereq: hire VERY selectively!-)

Pointing out the most important issues in the two-way exchange of trust between manager and developers -- and the (I hope:-) non-obvious connection back to hiring. You don't hire just for technical excellence -- character and personality are ALSO key.

Trust is a Virtuous Circle

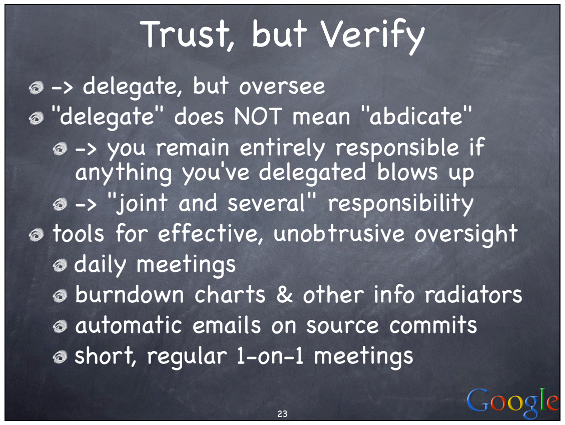


Trust can build on itself, gradually and progressively (of course, a breach of trust can similarly snowball into very justified MIStrust; perhaps the most common example in our context is the manager that tries to grab credit for himself, or shunt blame to developers, rather than properly taking his responsibilities and giving credit to his people).

More about Trust...

THE Evolution OF Cooperation
ROBERT AXELROD
STEPHEN R. COVEY
THE SPEED OF TRUST
The One Thing That Changes Everything
STEPHEN M. R. COVEY
The Origins of Virtue
Human Instincts and the Evolution of Cooperation
MATT RIDLEY
Google

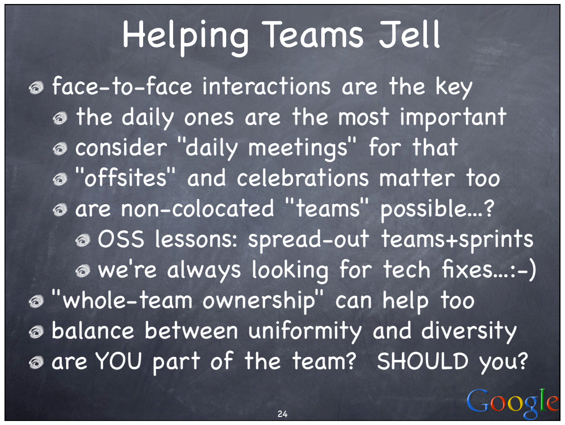
Axelrod works on simple but interesting mathematical models and simulations of "prisoner dilemma" (and has further published more work on the subject). Covey's is a typical, high-quality "self-help-style" book (like his dad's:-). Ridley starts from a biology (particularly, genetics) perspective, but in this book tries to tie in threads from economics (including game theory), sociology, anthropology, ... You shouldn't read JUST about management theory and software development issues, after all...!-)



Trust, but Verify

- -> delegate, but oversee
- "delegate" does NOT mean "abdicate"
  - -> you remain entirely responsible if anything you've delegated blows up
  - -> "joint and several" responsibility
- tools for effective, unobtrusive oversight
  - daily meetings
  - burndown charts & other info radiators
  - automatic emails on source commits
  - short, regular 1-on-1 meetings

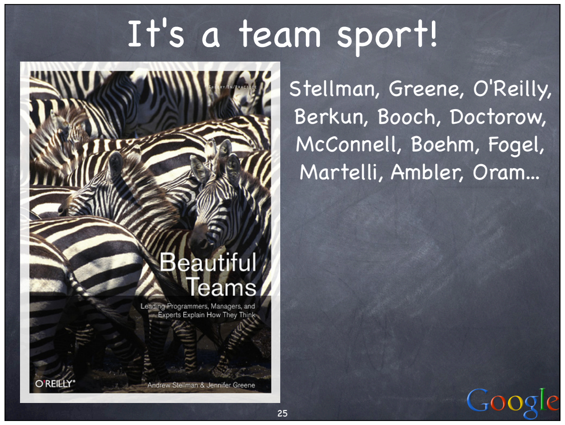
Yeah, the title's a quote out of context:-). But the key is: you keep responsibility for all you delegate; the best mechanisms to let you keep an eye on things are just the same that help *project transparency* all around, plus 1-on-1 meetings (which you should have anyway) to find out about things that escape the "transparency" mechanisms (mostly "people issues"). A formal mentor relationship *outside* the reporting chain is also extremely useful (doesn't help YOU, but helps the FIRM...).



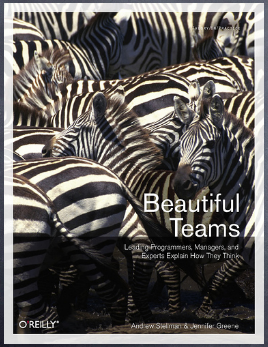
Helping Teams Jell

- face-to-face interactions are the key
  - the daily ones are the most important
  - consider "daily meetings" for that
  - "offsites" and celebrations matter too
  - are non-colocated "teams" possible...?
    - OSS lessons: spread-out teams+sprints
    - we're always looking for tech fixes...:-)
- "whole-team ownership" can help too
- balance between uniformity and diversity
- are YOU part of the team? SHOULD you?

Teams are what really matters most, and many aspects can help a team "jell" and become MUCH more productive -- chiefly the right kind of face to face interactions, but issues of "ownership" and uniformity vs diversity also play key roles. A crucial question (that I'm still leaving open -- more foreshadowing!-) is whether the manager CAN, and SHOULD, be _part_ of the team s/he manages.



It's a team sport!

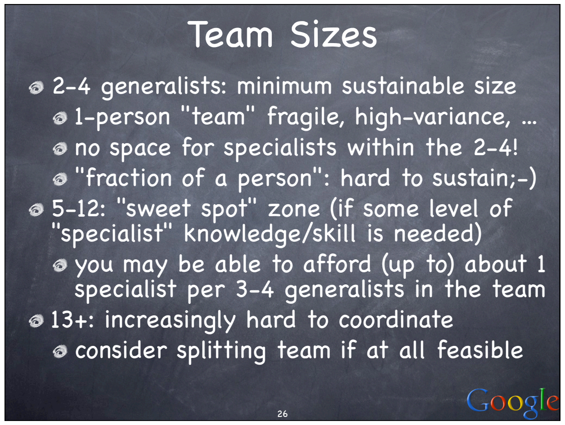

Beautiful Teams
Leading Programmers, Managers, and Experts Explain How They Think
O'REILLY
Andrew Stellman & Jennifer Greene

Stellman, Greene, O'Reilly, Berkun, Booch, Doctorow, McConnell, Boehm, Fogel, Martelli, Ambler, Oram...

Google

25

My contribution is the least of the book's chapters, but being in such exhalted company made my heart spin;-). BTW, what would normally be the many authors' royalties go to charity.



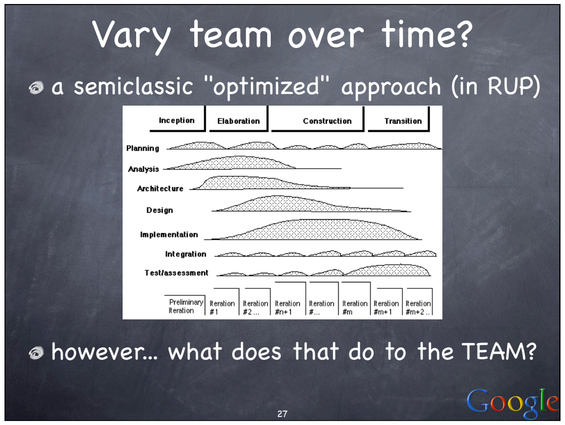
Team Sizes

- 2-4 generalists: minimum sustainable size
  - 1-person "team" fragile, high-variance, ...
  - no space for specialists within the 2-4!
  - "fraction of a person": hard to sustain;-)
- 5-12: "sweet spot" zone (if some level of "specialist" knowledge/skill is needed)
  - you may be able to afford (up to) about 1 specialist per 3-4 generalists in the team
- 13+: increasingly hard to coordinate
  - consider splitting team if at all feasible

Google

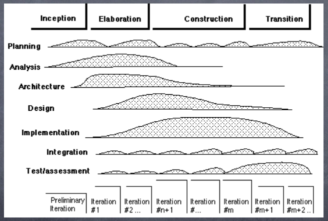
26

I've seen alleged "teams" of 40 or more people -- but "alleged" is key here;-). More often these days you see single people assigned as the whole "team" for small subprojects, but that's far from optimal either. If I could have my pick, I'd have 5 generalists when I can do w/o specialists, or about 2 specialists and 6 generalists for a big ambitious project when the specialists' advanced contributions are crucial to the goals (and ideally use techniques, such as mentoring and pair programming, to "grow" the specialists AND the generalists at the same time;-).



Vary team over time?

- a semiclassic "optimized" approach (in RUP)

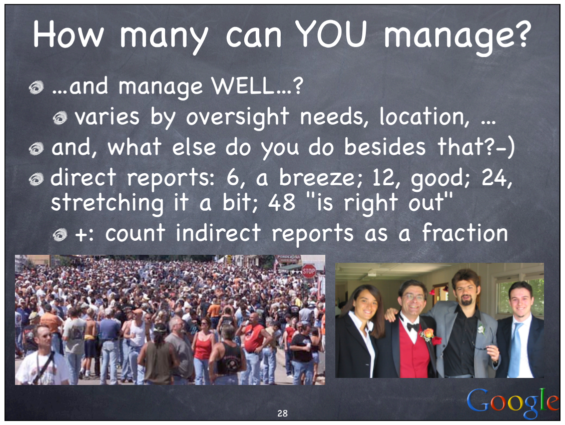


- however... what does that do to the TEAM?

Google

27

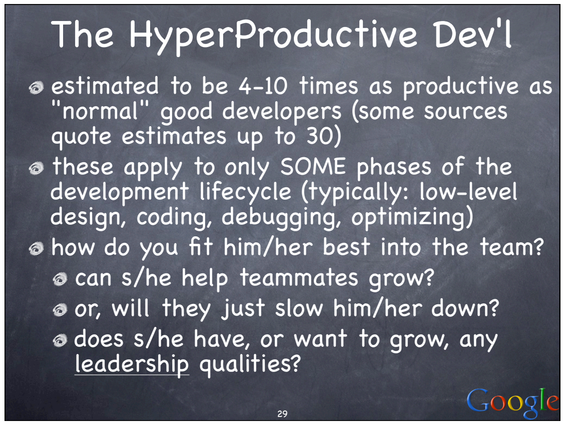
Systematically changing staffing levels and composition over a projects' macro-terms lifecycle seems an obvious approach... but you SHOULD consider what it WILL to do team's cohesion and "jelling". I'm NOT advocating "stationary" teams -- projects SHOULD have a beginning and an end, and so (on a longer timescale) generally should team's lifecycles -- but DO consider the people (and their ineffable dynamics as a jelled team!), NOT just the single task!!!



How many can YOU manage?

- ...and manage WELL...?
  - varies by oversight needs, location, ...
- and, what else do you do besides that?-)
- direct reports: 6, a breeze; 12, good; 24, stretching it a bit; 48 "is right out"
  - +: count indirect reports as a fraction

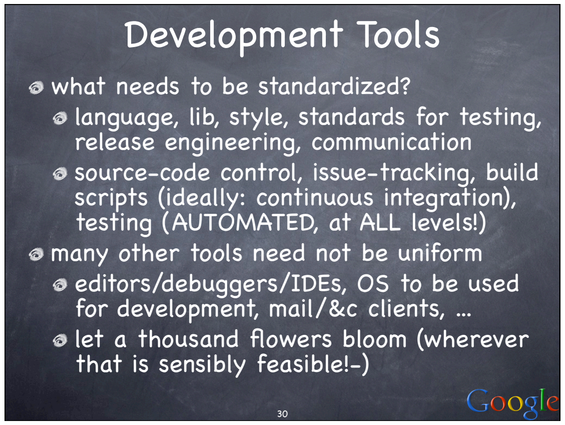
In one team or several, there IS a limit to the number of people you can manage effectively, depending on how much hand-holding (or shielding, oversight, etc) your people need, whether you have other things to do besides people-managing, etc. If you're a second- or higher-level manager, don't forget to count SOME fraction of your indirect reports -- they WILL at least occasionally need your direct attention and involvement (you DON'T want to ossify the organization in order to avoid that, believe me!)



The HyperProductive Dev'l

- estimated to be 4-10 times as productive as "normal" good developers (some sources quote estimates up to 30)
- these apply to only SOME phases of the development lifecycle (typically: low-level design, coding, debugging, optimizing)
- how do you fit him/her best into the team?
  - can s/he help teammates grow?
  - or, will they just slow him/her down?
  - does s/he have, or want to grow, any leadership qualities?

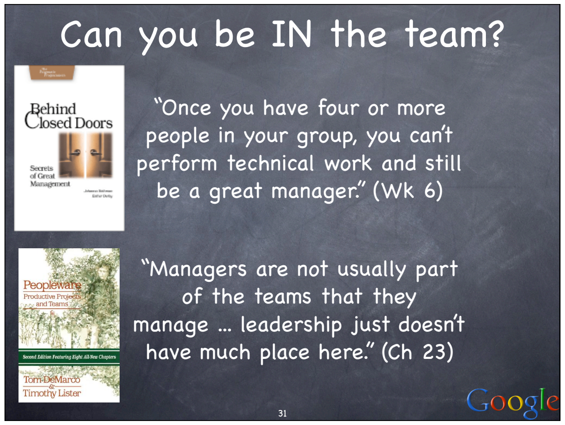
A nice problem to have, in a sense;-). But, problem (or, "opportunity":-) it sure is. How do you let the HPD fully express his or her incredible productivity while NOT damaging the team and other members thereof, and indeed HELPING them grow...?



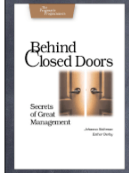
Development Tools

- what needs to be standardized?
  - language, lib, style, standards for testing, release engineering, communication
  - source-code control, issue-tracking, build scripts (ideally: continuous integration), testing (AUTOMATED, at ALL levels!)
- many other tools need not be uniform
  - editors/debuggers/IDEs, OS to be used for development, mail/&c clients, ...
  - let a thousand flowers bloom (wherever that is sensibly feasible!-)

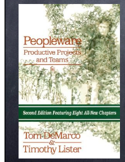
People want to "mark their space" -- let each developer happily use their favorite editor, debugger, IDE, etc. But you must stand firm on other issues: the key SHARED tools are versioning system, issue-tracking system, continuous integration and automated testing system (the 3 had better be well harmonized and integrated, too:-). Also, there needs to be consensus on programming language (and style, libraries, frameworks, ...), and standards for testing levels, release/deployment activities, docs and other forms of communication -- all indispensable issues to afford "whole team ownership", an absolutely crucial practice.



Can you be IN the team?



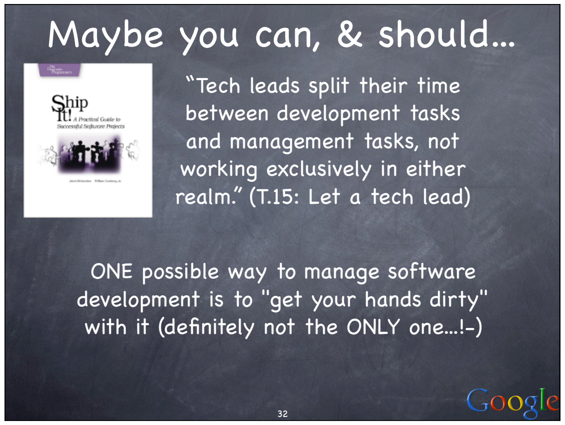
"Once you have four or more people in your group, you can't perform technical work and still be a great manager." (Wk 6)



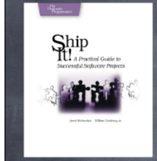
"Managers are not usually part of the teams that they manage ... leadership just doesn't have much place here." (Ch 23)

31

Two excellent books (which I both heartily recommend) heartily endorsing the traditional idea that managers can never do technical work, be part of their teams, exercise "leadership"...



Maybe you can, & should...

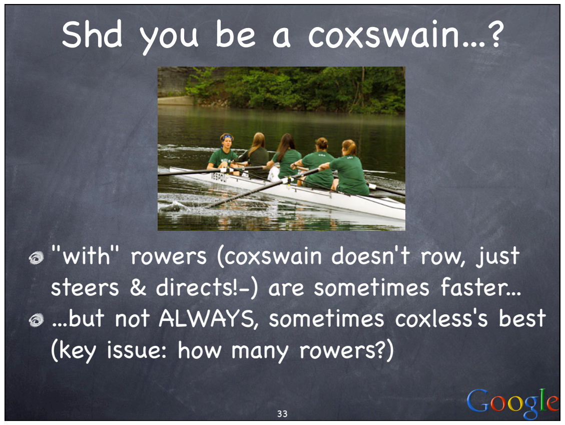


"Tech leads split their time between development tasks and management tasks, not working exclusively in either realm." (T.15: Let a tech lead)

ONE possible way to manage software development is to "get your hands dirty" with it (definitely not the ONLY one...!-)

32

...and another excellent book arguing otherwise. So maybe, while not the ONLY way, ONE way to manage knowledge workers IS to be (part-time) one of them...?

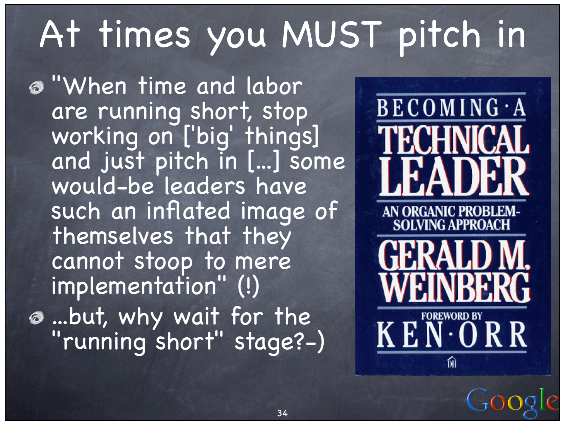


Shd you be a coxswain...?

- "with" rowers (coxswain doesn't row, just steers & directs!-) are sometimes faster...
- ...but not ALWAYS, sometimes coxless's best (key issue: how many rowers?)

33

Rowing lets us argue both ways -- for large-enough teams of rowers, a non-rowing coxswain appears to be a plus; but for smaller boats, the "without" times appear better;-)



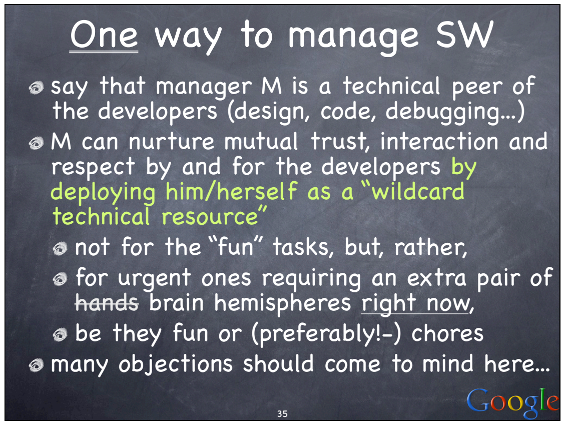
At times you MUST pitch in

- "When time and labor are running short, stop working on ['big' things] and just pitch in [...] some would-be leaders have such an inflated image of themselves that they cannot stoop to mere implementation" (!)
- ...but, why wait for the "running short" stage?-)

BECOMING·A TECHNICAL LEADER
AN ORGANIC PROBLEM-SOLVING APPROACH
GERALD M. WEINBERG
FOREWORD BY KEN·ORR

34

Weinberg is talking about generating ideas vs executing on them -- but I think this generalizes well... except that you shouldn't wait for emergencies where "time and labor are running short" before you DO pitch in -- that's very much the wrong approach.

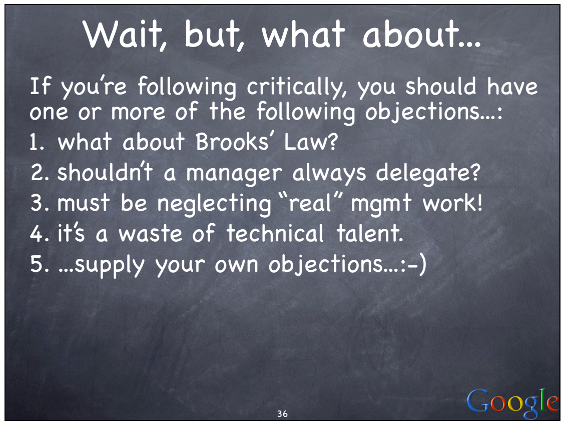


One way to manage SW

- say that manager M is a technical peer of the developers (design, code, debugging...)
- M can nurture mutual trust, interaction and respect by and for the developers by deploying him/herself as a "wildcard technical resource"
  - not for the "fun" tasks, but, rather,
  - for urgent ones requiring an extra pair of ~~hands~~ brain hemispheres right now,
  - be they fun or (preferably!-) chores
- many objections should come to mind here...

35

So here's my core thesis, which I've been pushing for years -- one approach to let a technical manager "get their hands dirty" in SOME of the team's professional work.



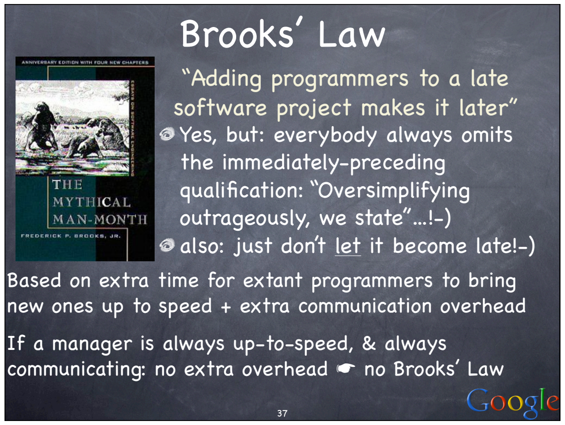
Wait, but, what about...

If you're following critically, you should have one or more of the following objections...:

1. what about Brooks' Law?
2. shouldn't a manager always delegate?
3. must be neglecting "real" mgmt work!
4. it's a waste of technical talent.
5. ...supply your own objections...:-)

36

Of course, many objections usually get raised here (I'm selecting a subset compared to what I do when I teach about this very specific idea exclusively, rather than about management in general as I do here:-).



Brooks' Law

"Adding programmers to a late software project makes it later"

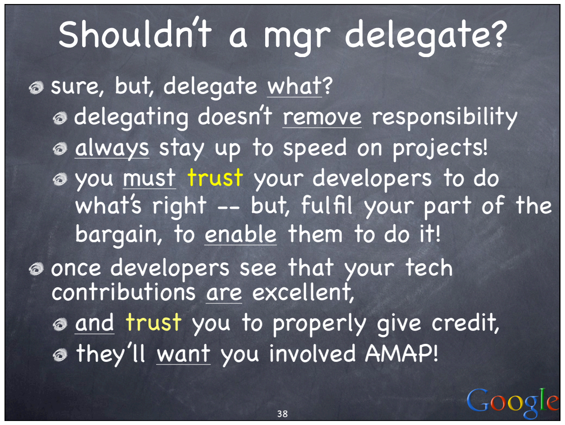
- Yes, but: everybody always omits the immediately-preceding qualification: "Oversimplifying outrageously, we state"...!-)
- also: just don't let it become late!-)

Based on extra time for extant programmers to bring new ones up to speed + extra communication overhead

If a manager is always up-to-speed, & always communicating: no extra overhead ☛ no Brooks' Law

37

I LOVE Brooks -- but his Law is often misquoted, misused, and misapplied.

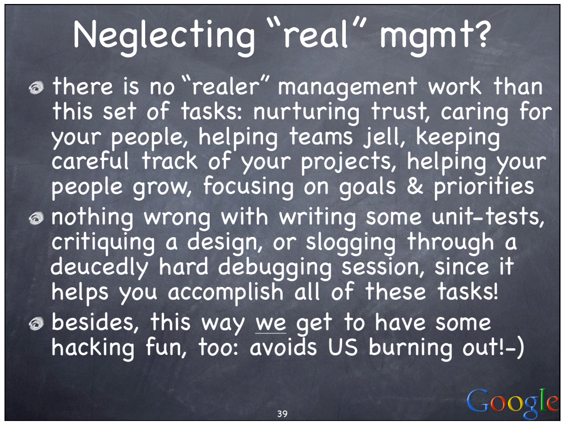


Shouldn't a mgr delegate?

- sure, but, delegate what?
  - delegating doesn't remove responsibility
  - always stay up to speed on projects!
  - you must trust your developers to do what's right -- but, fulfil your part of the bargain, to enable them to do it!
- once developers see that your tech contributions are excellent,
  - and trust you to properly give credit,
  - they'll want you involved AMAP!

38

Delegation is not abdication!

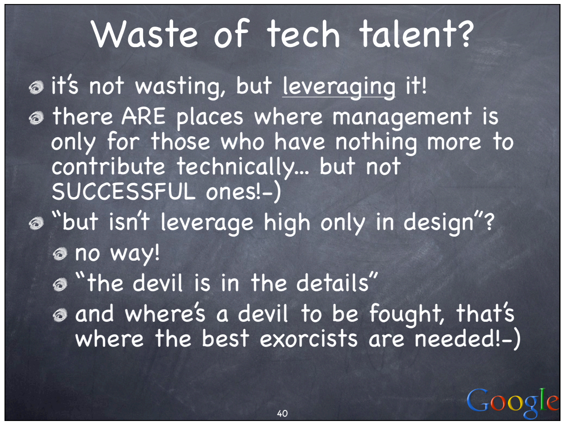


Neglecting "real" mgmt?

- there is no "realer" management work than this set of tasks: nurturing trust, caring for your people, helping teams jell, keeping careful track of your projects, helping your people grow, focusing on goals & priorities
- nothing wrong with writing some unit-tests, critiquing a design, or slogging through a deucedly hard debugging session, since it helps you accomplish all of these tasks!
- besides, this way we get to have some hacking fun, too: avoids US burning out!-)

39

"Real management" is anything and everything real (successful) managers do, that helps their projects!-) Balance is needed (as it always is "out here on the edge":-) -- don't hog the fun tasks, don't override your tech leads and make their jobs irrelevant, don't take credit for technical accomplishments, etc, etc -- but part of the balance is getting the challenge and fun to deal directly with hard technical tasks, at least once in a while!-)

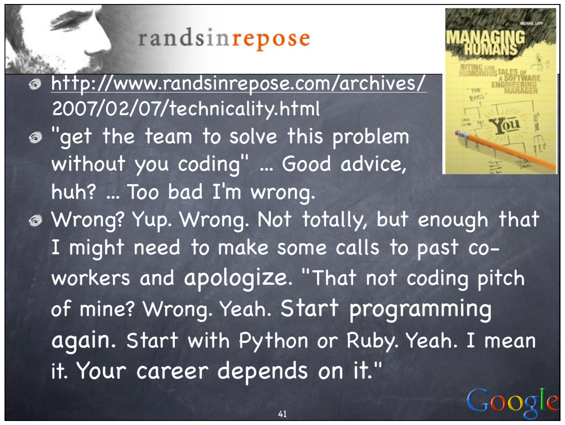


Waste of tech talent?

- it's not wasting, but leveraging it!
- there ARE places where management is only for those who have nothing more to contribute technically... but not SUCCESSFUL ones!-)
- "but isn't leverage high only in design"?
  - no way!
  - "the devil is in the details"
  - and where's a devil to be fought, that's where the best exorcists are needed!-)

40

That's kind of the flip objection to the last one -- that great techies should never move on to management to avoid "wasting" their tech talent (or, in a more attenuated form: that they should strictly focus on "upstream" activities, not coding, debugging, docs, deployment, testing, ...). Successful organizations in the hi-tech field MUST have good ways to balance leadership and technical contributions (dual ladders AND ways for people to "straddle" both ladders, hop back and forth between them, etc:-).

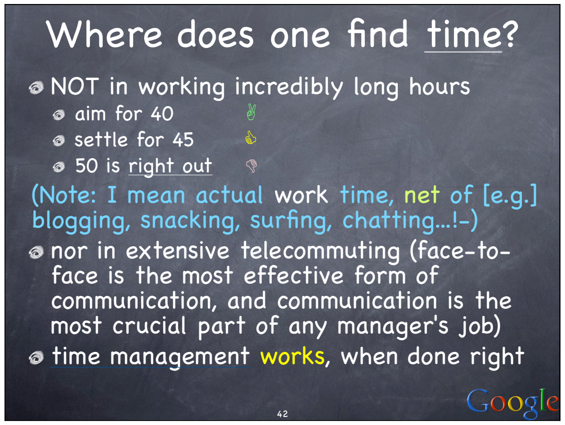


randsinrepose

- http://www.randsinrepose.com/archives/2007/02/07/technicality.html
- "get the team to solve this problem without you coding" ... Good advice, huh? ... Too bad I'm wrong.
- Wrong? Yup. Wrong. Not totally, but enough that I might need to make some calls to past co-workers and apologize. "That not coding pitch of mine? Wrong. Yeah. Start programming again. Start with Python or Ruby. Yeah. I mean it. Your career depends on it."

41

I was particularly happy, a good while after I'd started my solo crusade to allow tech mgrs SOME hands-on tech contributions, to see one of my favorite bloggers on technical management issues pick up on exactly the same theme. Lopp's book is good, too (if you can stand books taken/inspired from blogs -- I'm OK w/them). And his ability to state he was wrong, and apologize, comes VERY close to PROVING he's a great manager (and all-around human being), not just a great writer about it!-)



Where does one find time?

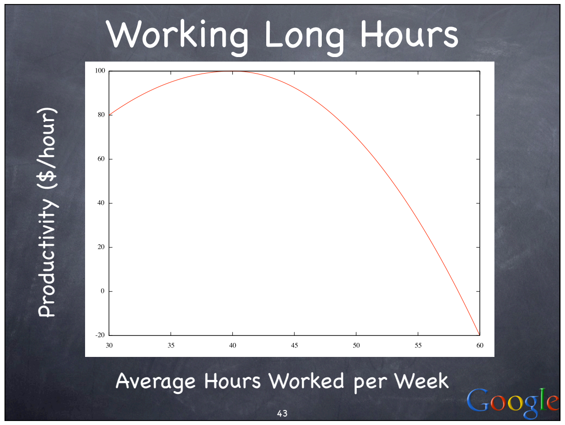
- NOT in working incredibly long hours
  - aim for 40
  - settle for 45
  - 50 is right out

(Note: I mean actual work time, net of [e.g.] blogging, snacking, surfing, chatting...!-)

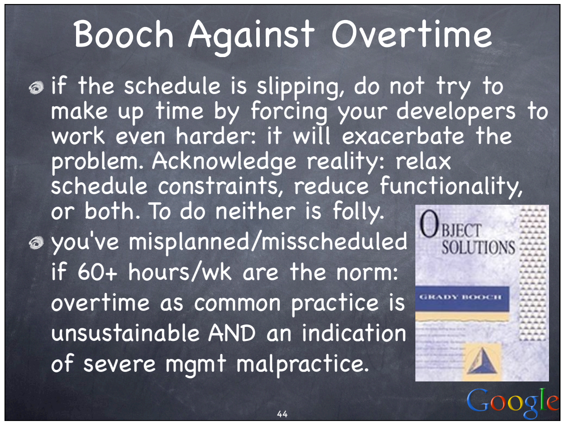
- nor in extensive telecommuting (face-to-face is the most effective form of communication, and communication is the most crucial part of any manager's job)
- time management works, when done right

42

There's a warped sense of pride in "working" 60-hours weeks -- but it's a horrid idea (even if you're used to it, you're unlikely to be producing at the top of your potential). Telecommuting's a very sweet dream... but face-to-face still IS best! So, you need a set of techniques, tricks and tips known classically as "time management" (or more modernly as "getting things done").



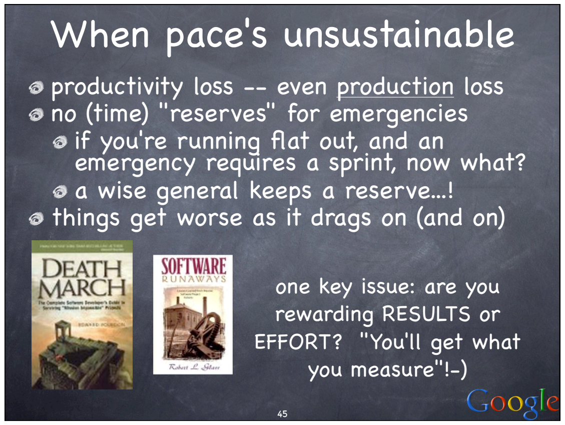
Of course, this is merely indicative (e.g., I know that at my best I'm worth more than $100/hr!-), but I hope it leaves the right impression -- there is SOME number of work hours per week where your productivity is maximal, and if you way overdo it your productivity will go negative -- by working in a far-too-tired state you'll do DAMAGE (negative value). Actually, it's been measured that the productivity per-hour of French workers (lowest hr/year in G8) is higher than in the US (highest), though the total production of _France_ as a whole suffers (Sarkozy promises to make them work more hours/year -- and yet he WON!-)



Booch Against Overtime

- if the schedule is slipping, do not try to make up time by forcing your developers to work even harder: it will exacerbate the problem. Acknowledge reality: relax schedule constraints, reduce functionality, or both. To do neither is folly.
- you've misplanned/misscheduled if 60+ hours/wk are the norm: overtime as common practice is unsustainable AND an indication of severe mgmt malpractice.

OCCASIONAL weeks at 60, 70, even 80 hours of actual work are inevitable -- emergencies, urgencies, and so on. But *keep them occasional*, for yourself as much as for your reports. Remember: it's a marathon, not a 100-meters dash!



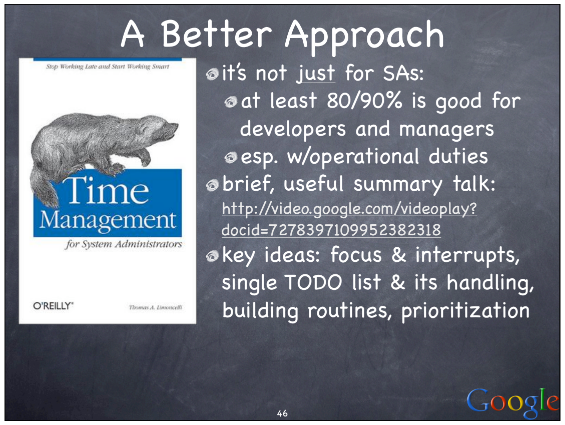
When pace's unsustainable

- productivity loss -- even production loss
- no (time) "reserves" for emergencies
  - if you're running flat out, and an emergency requires a sprint, now what?
  - a wise general keeps a reserve...!
- things get worse as it drags on (and on)

one key issue: are you rewarding RESULTS or EFFORT? "You'll get what you measure"!-)

DEATH MARCH
SOFTWARE RUNAWAYS
Google
45

The "key issue" about performance measurement and rewards is truly key. Again, it's too easy to reward "heroic efforts" rather than good strong planning which saves the needs for such efforts and obtains better RESULTS.



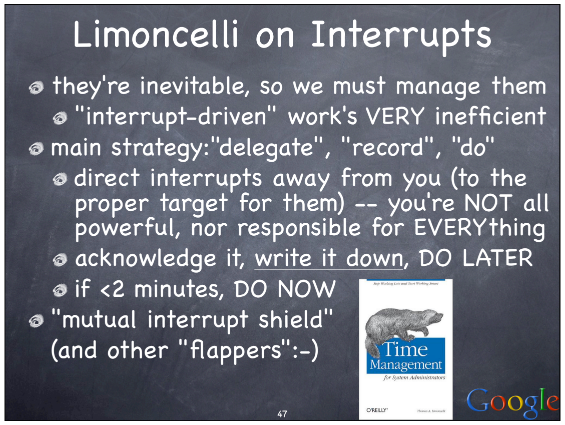
A Better Approach

- it's not *just* for SAs:
  - at least 80/90% is good for developers and managers
  - esp. w/operational duties
- brief, useful summary talk: http://video.google.com/videoplay?docid=7278397109952382318
- key ideas: focus & interrupts, single TODO list & its handling, building routines, prioritization

Stop Working Late and Start Working Smart
Time Management
for System Administrators
O'REILLY
Thomas A. Limoncelli
Google

46

Tom's first-person experience is as a (great!) system administrator, so, quite rightly, that's what he writes about -- but his observations and tips, for the most part, do generalize to all of us.



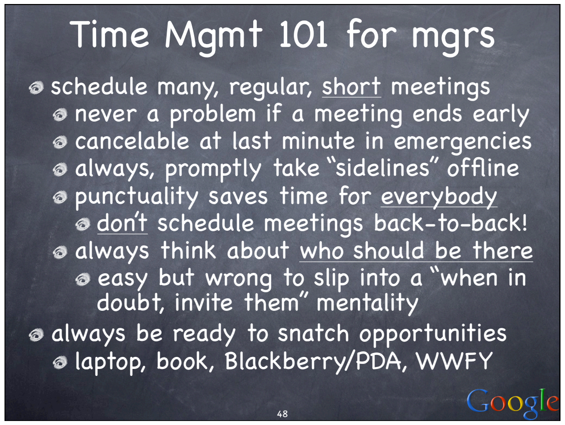
Limoncelli on Interrupts

- they're inevitable, so we must manage them
  - "interrupt-driven" work's VERY inefficient
- main strategy:"delegate", "record", "do"
  - direct interrupts away from you (to the proper target for them) -- you're NOT all powerful, nor responsible for EVERYthing
  - acknowledge it, write it down, DO LATER
  - if <2 minutes, DO NOW
- "mutual interrupt shield" (and other "flappers":-)

Time
Management
O'REILLY
Google

47

"Deflecting" and otherwise managing interrupts is a technique I single out because it's quite as important for managers as it is for sysadmins. The "mutual interrupt shield" (unless you're high up enough to warrant an administrative assistant, and, even then...) is a particularly useful, although non-obvious, technique for managers.

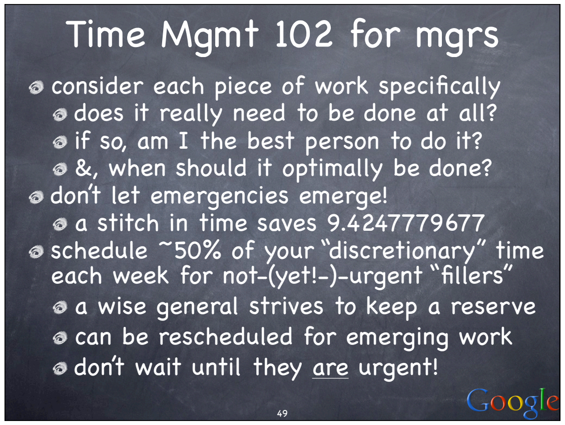


Time Mgmt 101 for mgrs

- schedule many, regular, *short* meetings
  - never a problem if a meeting ends early
  - cancelable at last minute in emergencies
  - always, promptly take "sidelines" offline
  - punctuality saves time for *everybody*
    - *don't* schedule meetings back-to-back!
  - always think about *who should be there*
    - easy but wrong to slip into a "when in doubt, invite them" mentality
- always be ready to snatch opportunities
  - laptop, book, Blackberry/PDA, WWFY

48

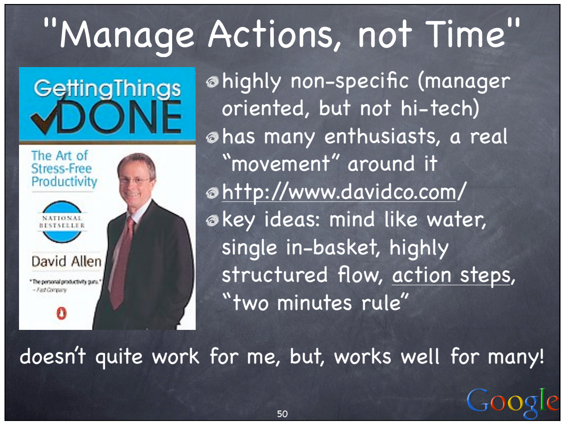
Here are a couple of elementary but crucial tips specifically for managers (the "snatch opportunities" one applies more widely;-).


Time Mgmt 102 for mgrs

- consider each piece of work specifically
  - does it really need to be done at all?
  - if so, am I the best person to do it?
  - &, when should it optimally be done?
- don't let emergencies emerge!
  - a stitch in time saves 9.4247779677
- schedule ~50% of your "discretionary" time each week for not-(yet!-)-urgent "fillers"
  - a wise general strives to keep a reserve
  - can be rescheduled for emerging work
  - don't wait until they <u>are</u> urgent!

49

These are a bit more advanced, but no less crucial. Care to guess what that weird number IS, btw?-)


"Manage Actions, not Time"

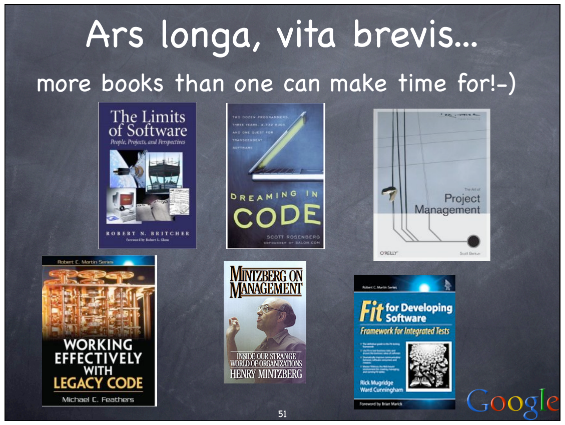
Getting Things DONE
The Art of Stress-Free Productivity
National Bestseller
David Allen

- highly non-specific (manager oriented, but not hi-tech)
- has many enthusiasts, a real "movement" around it
- http://www.davidco.com/
- key ideas: mind like water, single in-basket, highly structured flow, action steps, "two minutes rule"

doesn't quite work for me, but, works well for many!

50

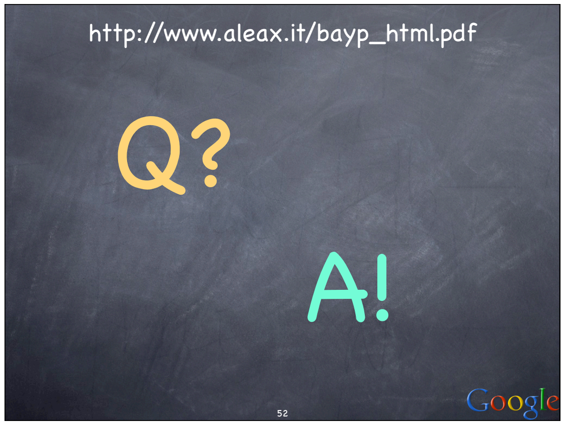
I'm not really into the GTD movement, myself; but I've seen it work great for many, so I feel I must at least mention it!


Ars longa, vita brevis...

more books than one can make time for!-)

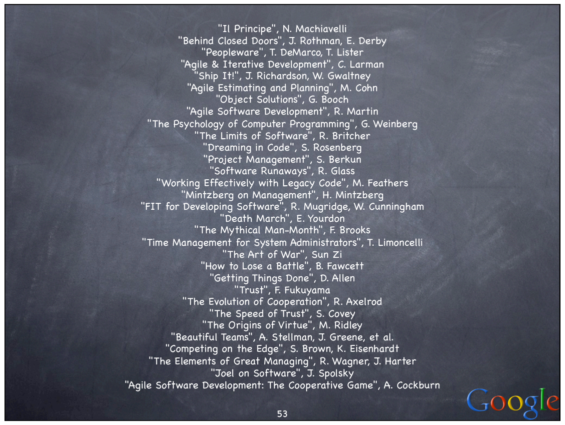
51

Mintzberg (many books) is at the same time a Solon of management teaching, and a trenchant critic of current pratice in the field. Berkun has many good tips and some interesting deep reflections on project management. FIT (and Fitnesse) are great ways to develop test-driven SW (at acceptance-test level, roughly). Feathers is great if you need to deal with untested, badly documented legacy SW (don't we all!). Britcher is pessimistic but worth reading (and critiquing). "Dreaming in Code" is the best case study yet on "why do bad things happen to good projects" -- Rosenberg has no prejudged idea, gives


http://www.aleax.it/bayp_html.pdf

52

Also, remember I'm always around on the mailing list, and passionate about these issues;-).


"Il Principe", N. Machiavelli
"Behind Closed Doors", J. Rothman, E. Derby
"Peopleware", T. DeMarco, T. Lister
"Agile & Iterative Development", C. Larman
"Ship It!", J. Richardson, W. Gwaltney
"Agile Estimating and Planning", M. Cohn
"Object Solutions", G. Booch
"Agile Software Development", R. Martin
"The Psychology of Computer Programming", G. Weinberg
"The Limits of Software", R. Britcher
"Dreaming in Code", S. Rosenberg
"Project Management", S. Berkun
"Software Runaways", R. Glass
"Working Effectively with Legacy Code", M. Feathers
"Mintzberg on Management", H. Mintzberg
"FIT for Developing Software", R. Mugridge, W. Cunningham
"Death March", E. Yourdon
"The Mythical Man-Month", F. Brooks
"Time Management for System Administrators", T. Limoncelli
"The Art of War", Sun Zi
"How to Lose a Battle", B. Fawcett
"Getting Things Done", D. Allen
"Trust", F. Fukuyama
"The Evolution of Cooperation", R. Axelrod
"The Speed of Trust", S. Covey
"The Origins of Virtue", M. Ridley
"Beautiful Teams", A. Stellman, J. Greene, et al.
"Competing on the Edge", S. Brown, K. Eisenhardt
"The Elements of Great Managing", R. Wagner, J. Harter
"Joel on Software", J. Spolsky
"Agile Software Development: The Cooperative Game", A. Cockburn

53