Technical Management Of Software Development

Experiences, Tips and Ideas

Alex Martelli

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

Is there a silver bullet?

- ...one single way to slay all monsters...?

Google

The talk is NOT about...

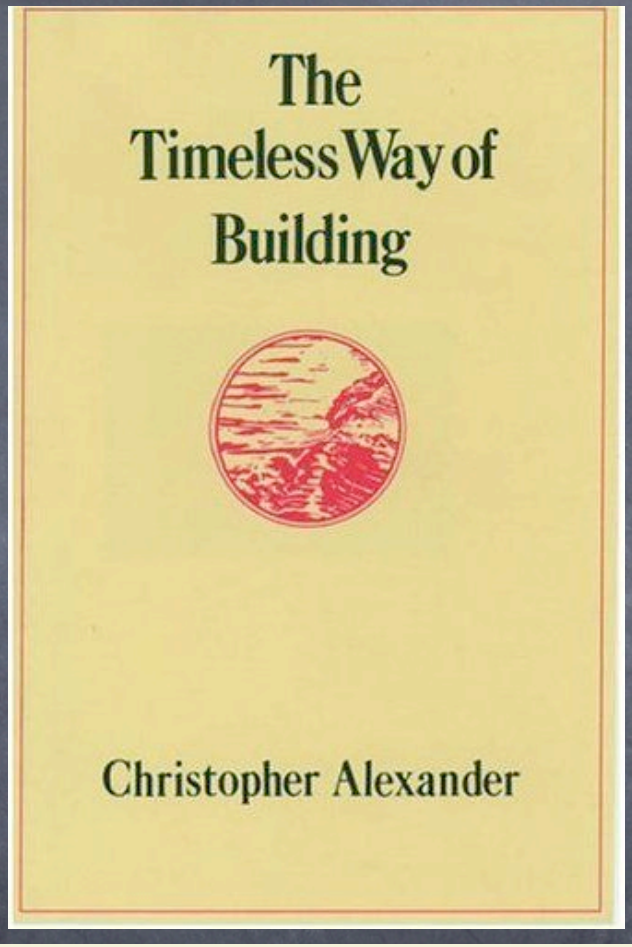
- (not at all about) strategic/executive management
  - business plans, finance, strategic vision...
- (mostly not about) project management
  - plans, schedules, budgets, iteration
- (hardly about) technologies/methods/tools
  - languages, OSs, frameworks, ...
  - however, I assume some level of Agility:
    - neither rigid Waterfall,
    - nor utter Chaos !-)

Google

Agile vs Waterfall vs Chaos

- Waterfall: plan, plan, plan, plan, plan, DO!
- Chaos: do, do, do, oops, undo, do, redo, ...
- Agile: plan, do, adjust; plan, do, adjust; plan, do, adjust; ...
  - successful software projects must be like that to some extent
  - Agile methods try to extract the aspects that work and apply them with discipline
  - and match with coding/testing techniques which best suit the reality at hand

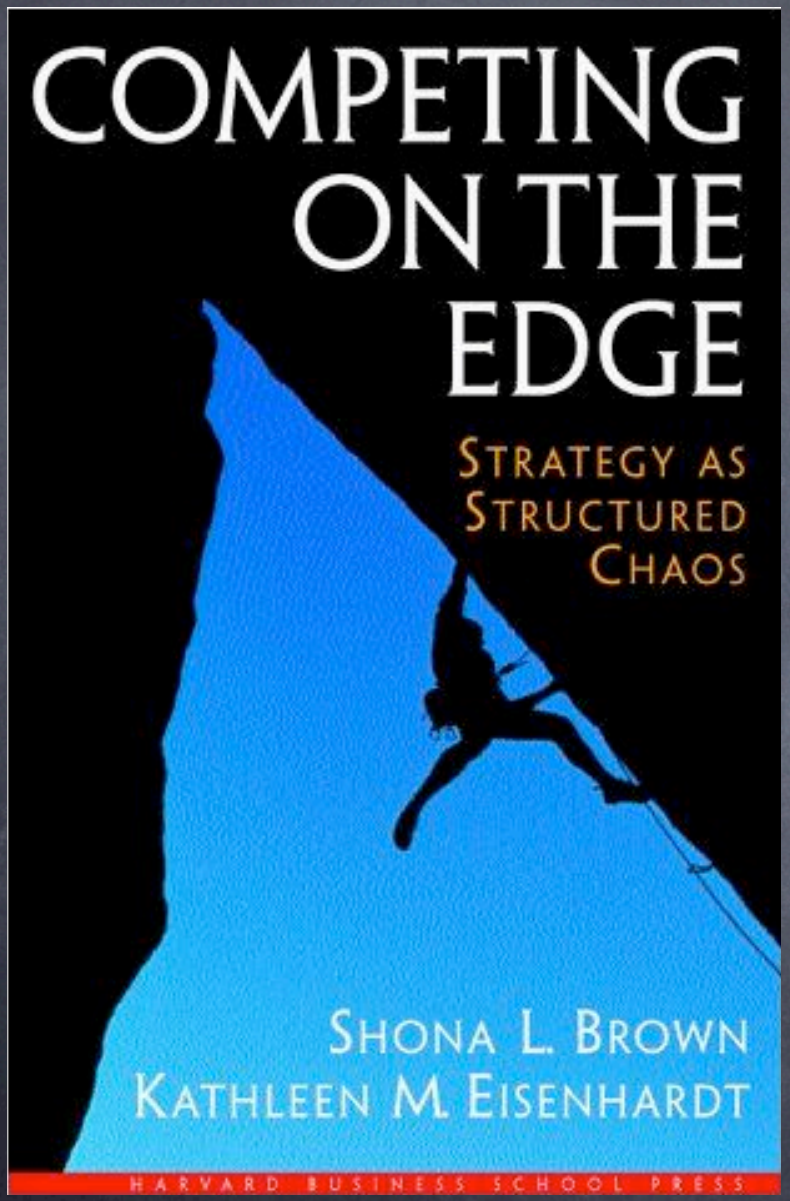
(Unlikely?) Agile Inspiration...



...a process which brings order out of nothing but ourselves; it cannot be attained, but it will happen of its own accord, if we let it.

Google

(btw): Strategic Agility



…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.**

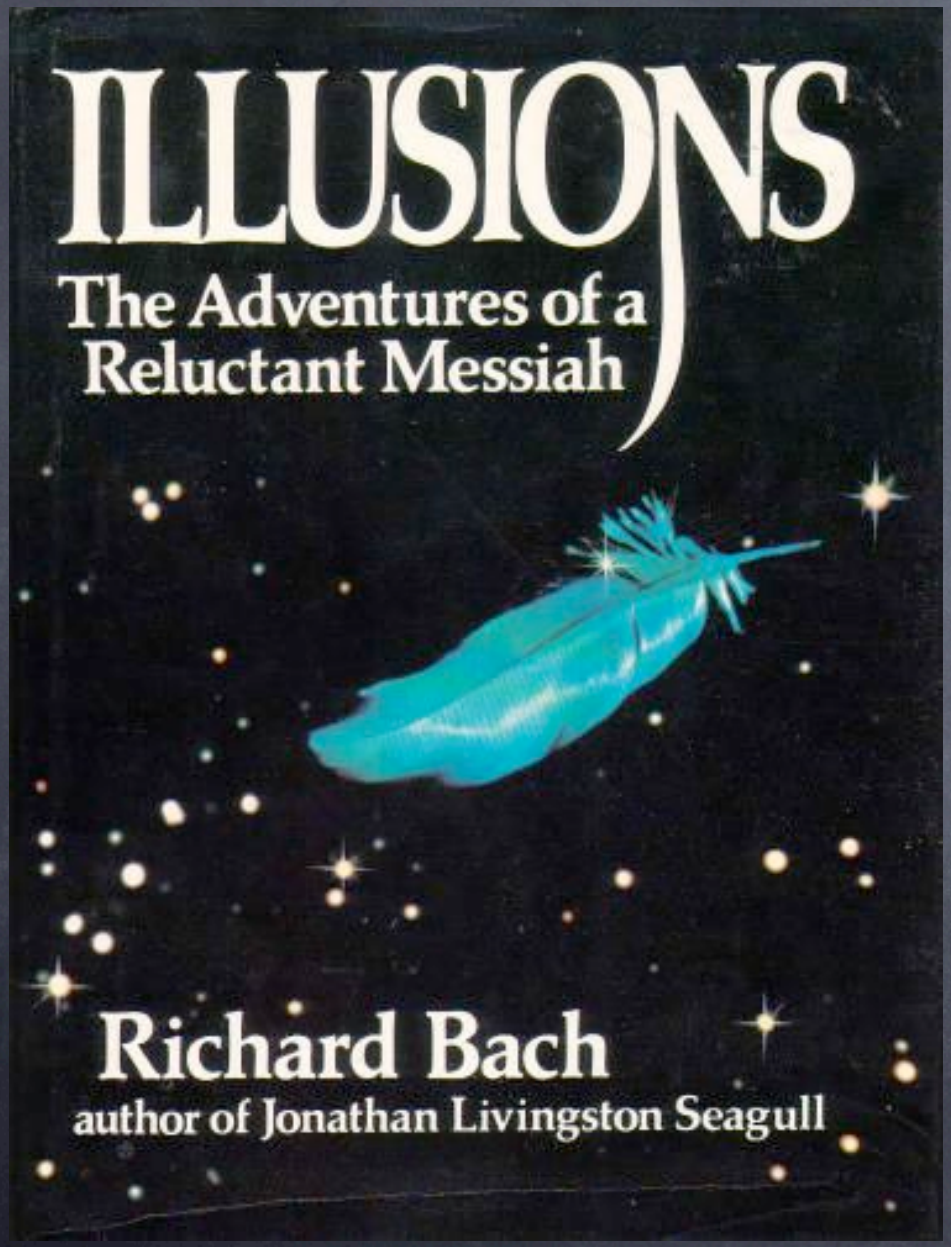
Google

This talk IS about...

- communicating my experiences (at Google and before) and related reflections and tips
  - not anecdotes, but the gist of them
  - about ONE successful way to do technical management of software development
  - other ways might also work, of course!
- disputing some very popular theses
  - while recommending some books that defend those theses!-)
- exchange of experiences, tips, reflections (both during the talk itself & in Q&A)

Google

...but, the best reason...:



You teach best what you most need to learn.

Google

The Stool of Development

needs three good, sturdy legs:

great strategic
right intention
leadership

effective
management

excellent developers

What makes strategic leaders great?

- deep vision + sharp action on business models, technologies, partnerships, staffing, agility: strategy + execution
- mutual trust, interaction and respect with managers and developers (and operations, and sales, and marketing, and ...)
- courage, integrity, humility, realism, pride, optimism, prudence

1. they do their (difficult) job well

2. they let me do mine!

Google

What makes developers excellent?

- great at design, coding, testing, debugging; mastery of algorithms, languages, tools, technologies, frameworks, the codebase: i.e., technical excellence
- mutual trust, interaction and respect with managers and each others (and UI, QA, ...)
- courage, integrity, humility, realism, pride, optimism, prudence

1. they do their (difficult) job well

2. they enable me to do mine!

Google

How do you get those?!

Every manager dreams of having great strategic leaders and excellent developers, but, how do you get them?

(1) luck (e.g.: you happen to work at Google!-)

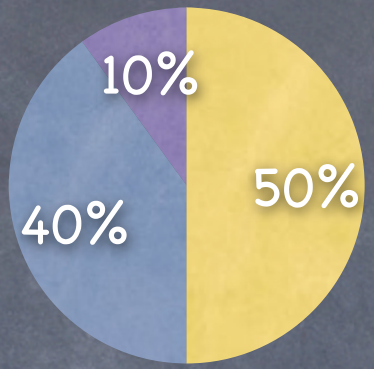
(2) choice (e.g.: come to work at Google!-)

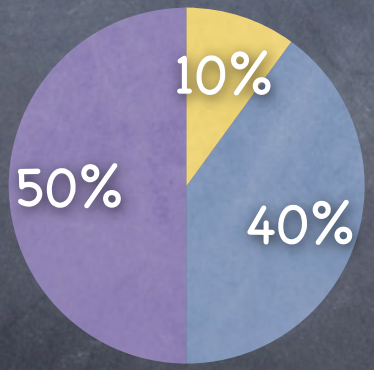
(3) grow them (not mutually exclusive w/1-2!)

- trust, in particular, always needs nurturing (more anon...)
- you can always teach a little
- you can always learn a lot

Two scary monsters...

Developing the wrong software





Percentages of blame

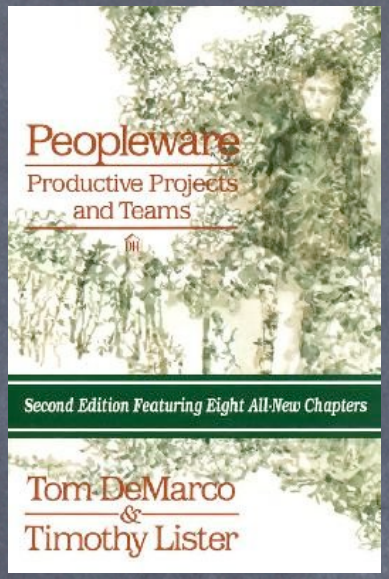
- Leaders
- Developers
- Manager

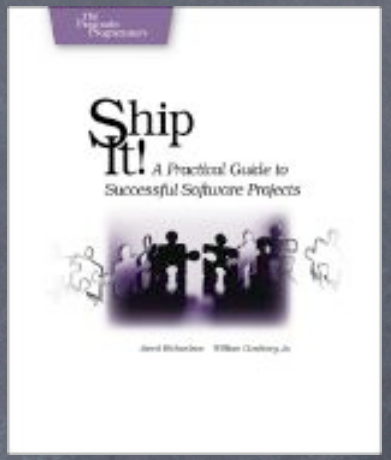
Developing the software wrong

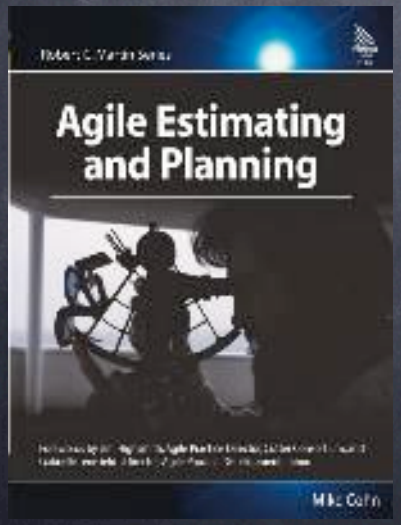
Google

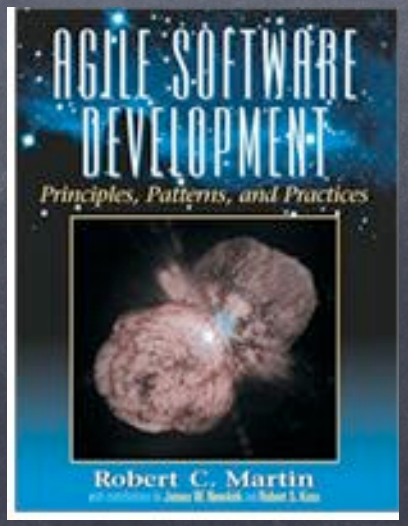
One way to learn...

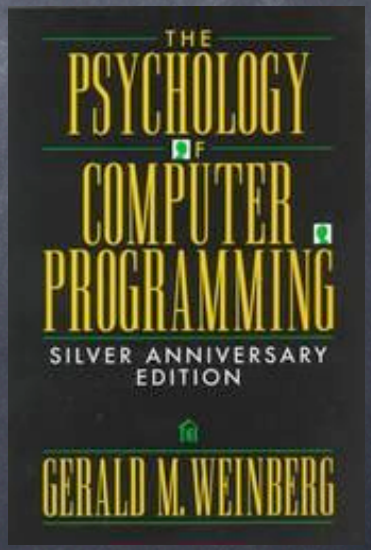
...is via good books -- and, there are many!







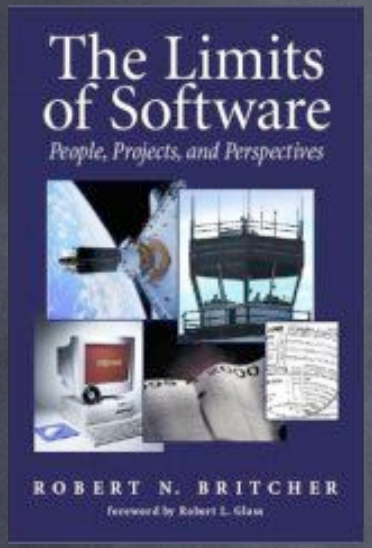


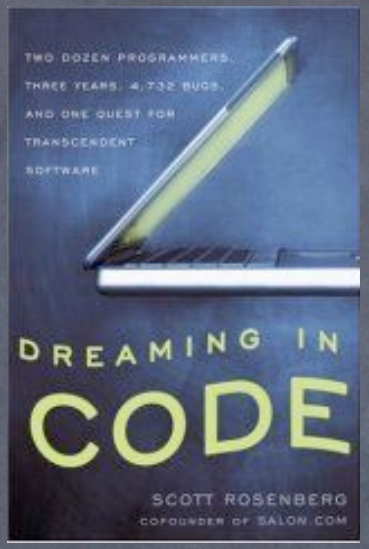


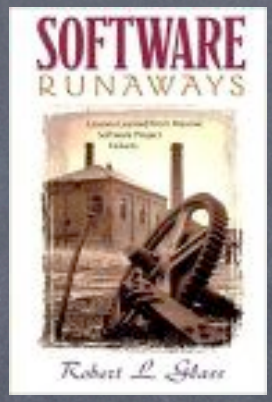
Google

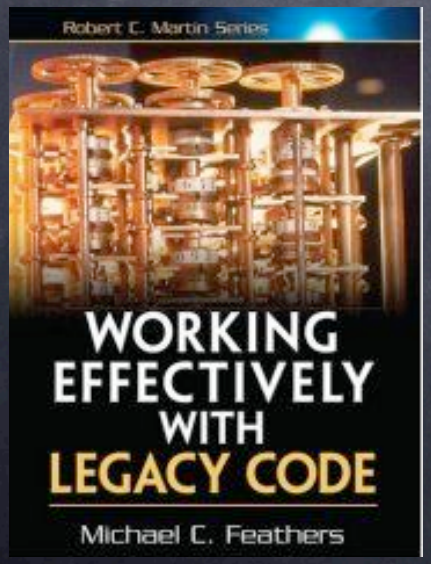
Ars longa, vita brevis...

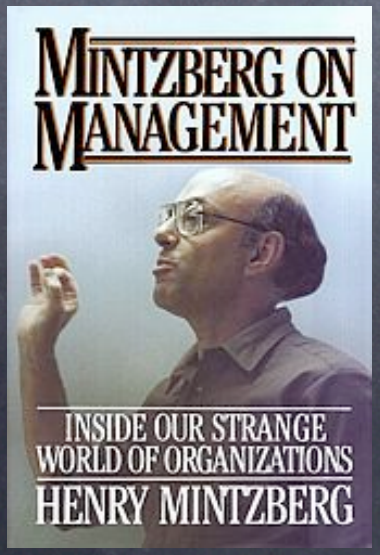
more books than one can make time for!

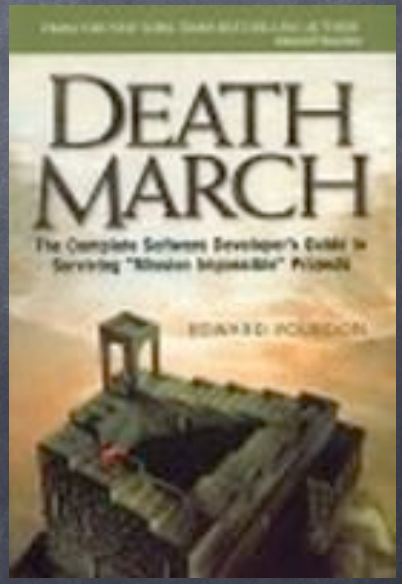












Google

Learning from books

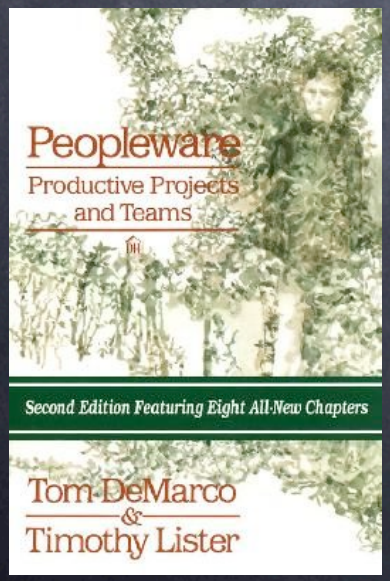
as we all know, requires the right approach:

- one always reads books critically
  - reading many divergent ones helps!-)
- one always sifts them through experience
  - try new tips & ideas out incrementally
    - "Big Bang" isn't just risky (and it is!),
    - it also clouds perception & analysis
  - use mostly your experience, but also -- use my experience, hers, his, ...
- NB: learning from tech talks, workshops, courses, needs much the same approach!-)

Google

One widespread opinion

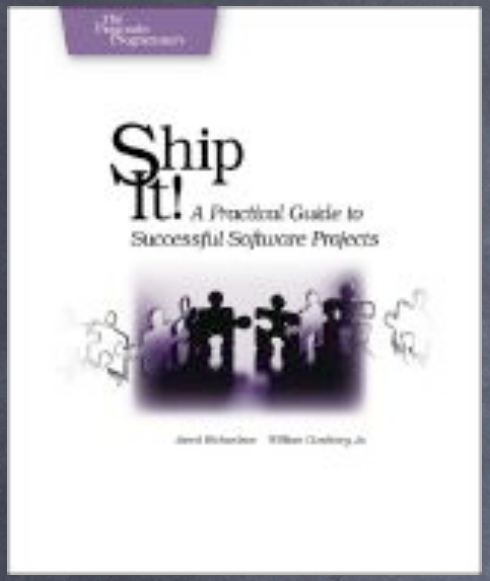
"Once you have four or more people in your group, you can't perform technical work and still be a great manager." (Wk 6)



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

Google

And a counterpoint to it



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

At Google, we distinguish pure "tech leads" (engineers who manage a project) from "uber tech leads" and "tech leads/managers" ("highly technical managers" aka HTM), but both can meet this definition.

Google

One way to be an HTM

- 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...

Google

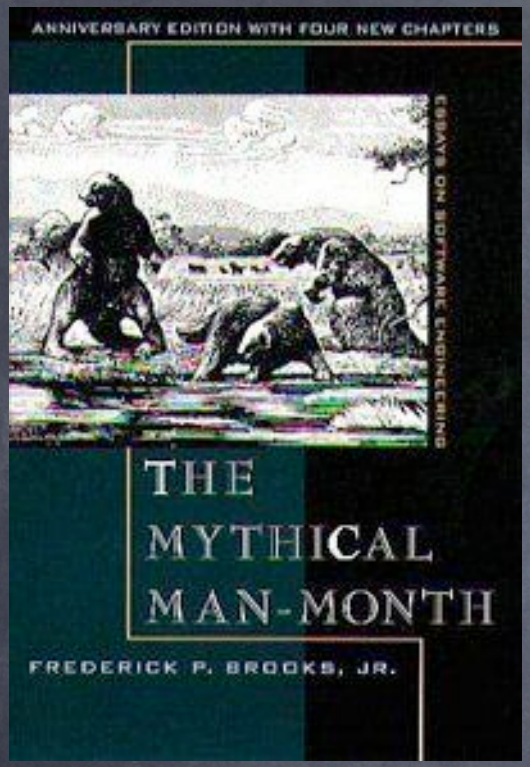
Wait, but, what about...

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

1. what about Brooks' Law?
2. where does one find the time!?
3. shouldn't a manager always delegate?
4. must be neglecting "real" mgmt work!
5. it's just a waste of technical talent
6. too many interrupts, you'd never get into "the flow state"!
7. ...supply your own objections...

Google

1. 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

The HTM is always essentially up-to-speed & always communicating, so: no extra overhead ☛ no Brooks' Law

Google

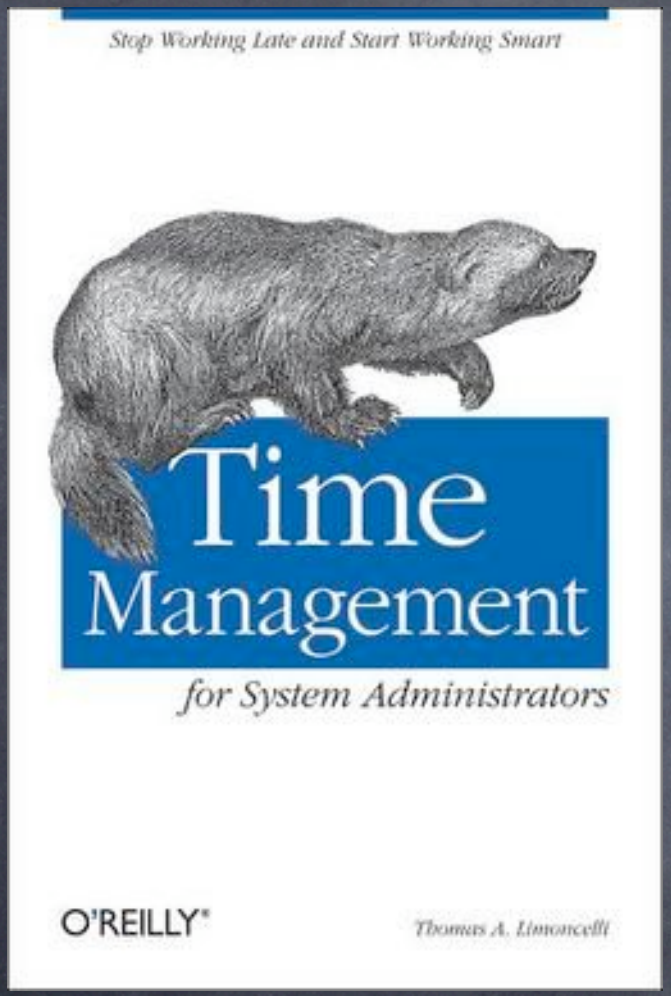
2. Where to find the time?

- 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, lunches...!-)

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

Time Management for ...



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

I can offer a few extra, HTM-specific tips...

Google

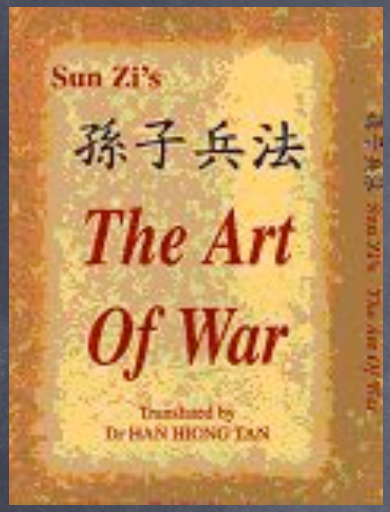
Time Mgmt 101 for HTMs

- 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

Time Mgmt 102 for HTMs

- 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!

Military Metaphors



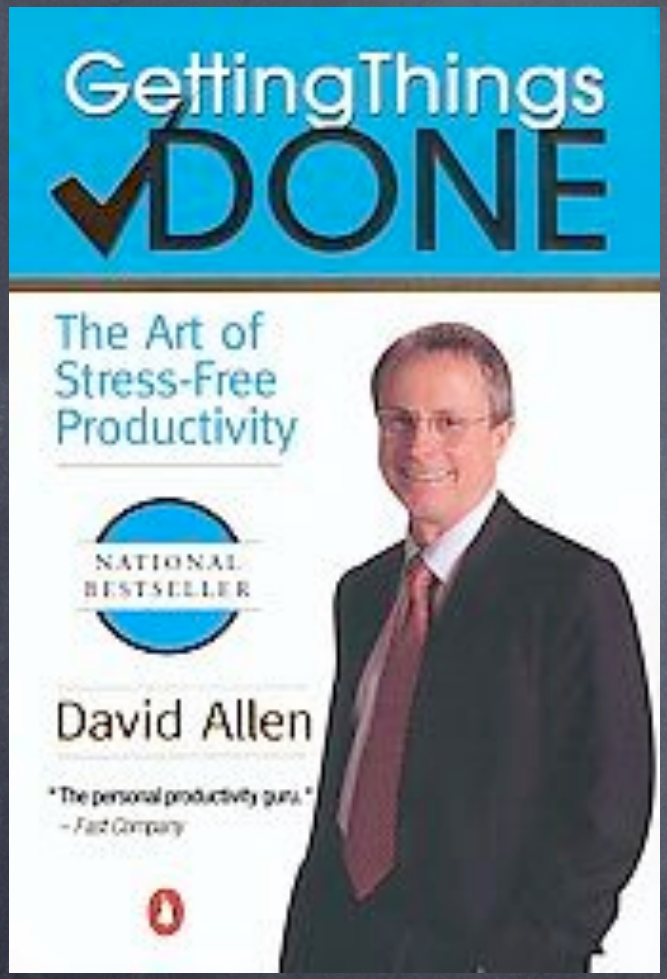
"While heeding my profitable advice, avail yourself also of any helpful circumstances, over and beyond ordinary rules." (v. 16)

NB: a few generals lost battles by never committing their reserves!-)

No plan survives contact with the enemy (von Moltke)

Planning is everything, plans are nothing (von Moltke)

Extremely popular:



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

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

Google

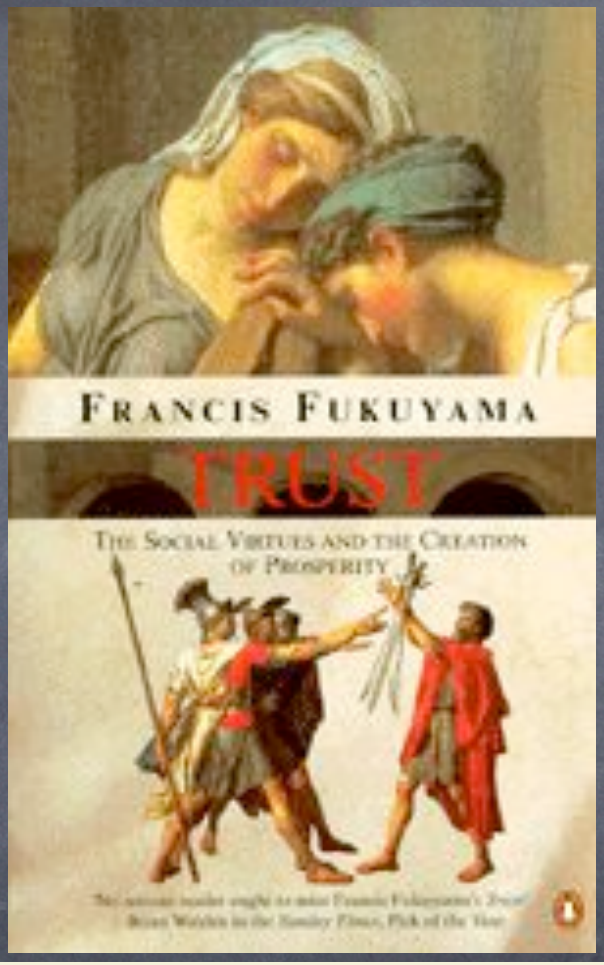
3. Shouldn't a mgr delegate?

- sure, but, delegate what?
  - delegating doesn't remove responsibility
  - always stay up to speed on all projects!
  - you must trust your developers to do what's right -- but, fulfill 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!

Trust...

Google

Trust...



The satisfaction we derive from being connected to others in the workplace grows out of a fundamental human desire for recognition.

Google

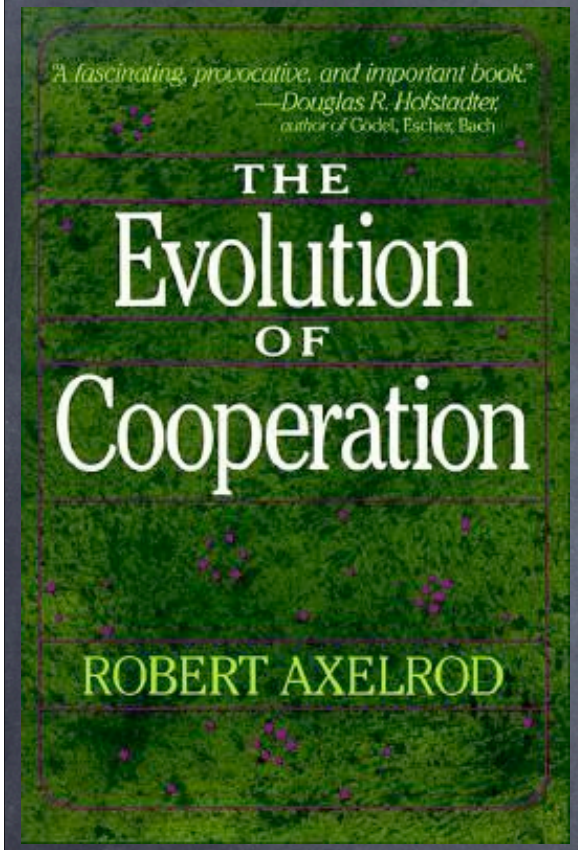
Trust...

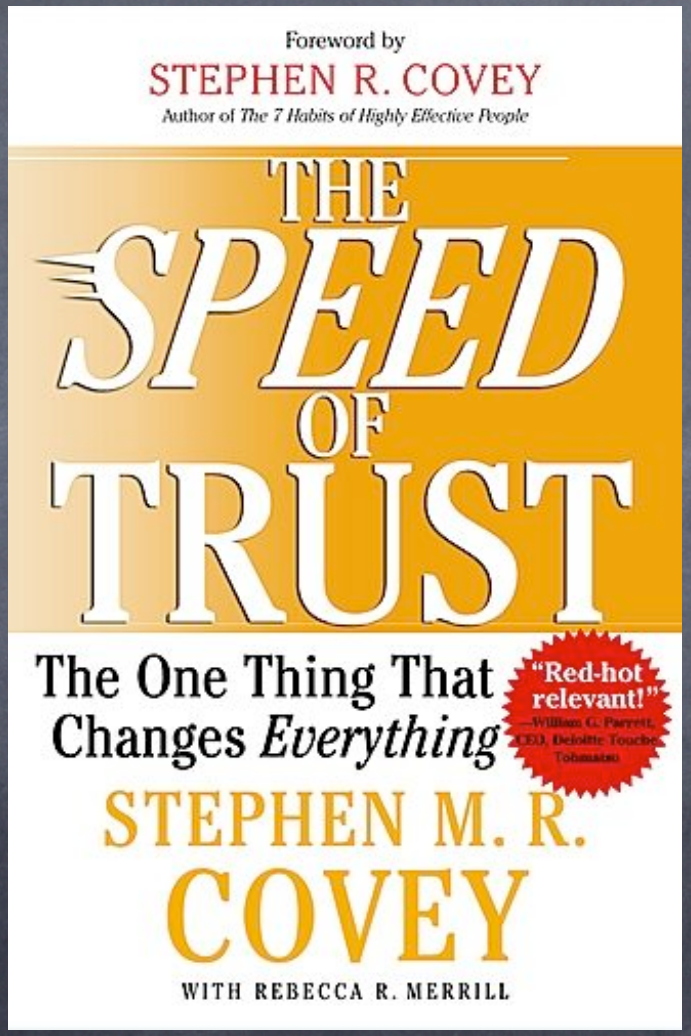
- is mutual, and built up over time
- you must earn & deserve developers' trust
  - technical ability & "technical currency"
  - true, not faked, interest in them as individuals, within and outside work
  - always give recognition and credit!
- they must earn & deserve yours
  - tech skills, integrity,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

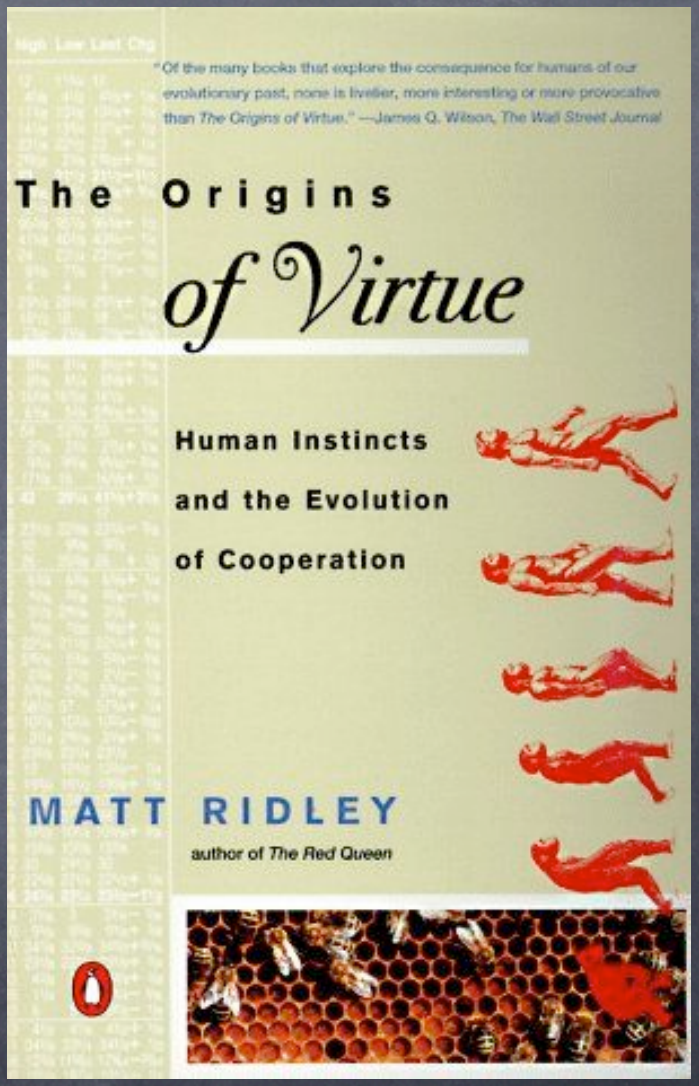
Trust begins at home

- to be worthy of others' trust, you must first be worthy of your own!
  - This above all: to thine own self be true,
    And it must follow, as the night the day,
    Thou canst not then be false to any man
  - don't tell yourself little white lies...!
  - "above all, don't fool yourself, don't say it was a dream, your ears deceived you: don't degrade yourself with empty hopes like these" (C. Cavavy, "The god forsakes Anthony")

More about Trust...







Google

4. 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!-)

No mere cogs in the wheel

- learn all you can about your developers' specific, individual strengths & weaknesses
  - particularly TLs, if any, but not just them
- play to their strengths
- shield their weaknesses, but also...:
  - help them to outgrow weaknesses
    - coaching, lessons, books, pairing, ...
    - run a marathon, not a 100-m dash!
- making unreasonable demands can burn them out (watch out for burnout!!!), but...
  - making only fully reasonable demands provides no challenge (stretch goals)

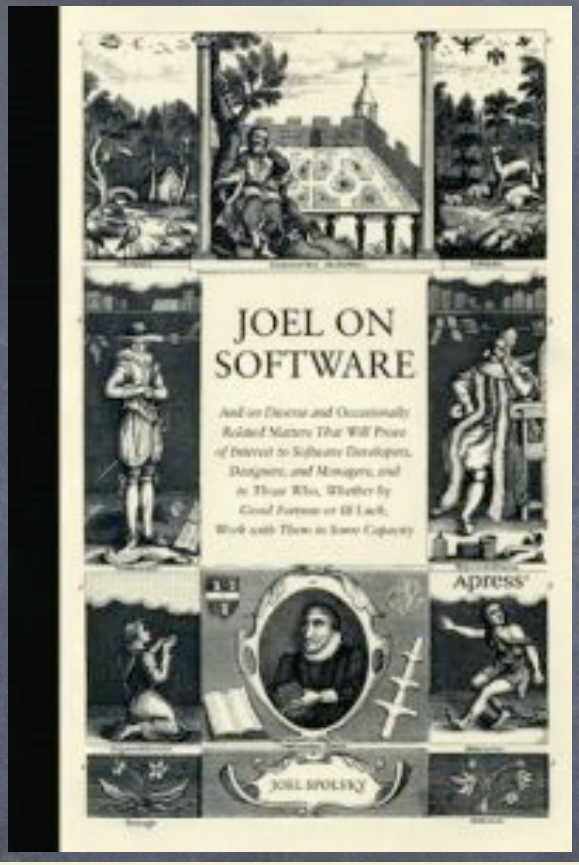
5. Waste of tech talent?

- it's not wasting, but leveraging it!
- in some places management is only for those who have nothing more to contribute technically... but not in successful shops!-)
- "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!-)

6. Interrupts and Flow

- read Limoncelli re managing interrupts
- you still have to deal with many of them
  - keep yourself out of critical-paths (or provide guaranteed alternate routing;-)
  - learn to tell what can wait 1 hour
  - learn to "push something on the stack", provide immediate attention to s/thing else, pop the stack and go right back
- in the end, it's possible that one just isn't designed for multitasking -- management always requires a lot of it, though!

For many diverse tips



you have to make a schedule... no programmer wants to... most are only doing it because their boss made them do it, halfheartedly, and nobody actually believes the schedule...

Google

Scheduling and Planning

- heed Joel's advice: appropriate technology is a spreadsheet (or whiteboard+stickies, or index cards...), not complex PERT/GANTT charts
  - pick very fine-grained tasks (to combat developers', and your own, optimism!-)
  - schedule vacations, holidays, training, sick days (proactively fight to avoid burnout!)
- heed Cohn's advice: estimate size, derive duration (&, size in arbitrary units * velocity)

Appropriate methods

- Agile: it's not only a good idea, it's a cool and trendy buzzword!-)
  - explore the whole space, together with your developers
  - but, strive for consistency among your projects (or, you'll go crazy!-)
  - retrospect & meta-refactor mercilessly
- controversy alert;-)...: some technologies are somewhat "more agile" than others (Ruby > Python > Java/C# > C++) -- choose wisely!

Appropriate tools

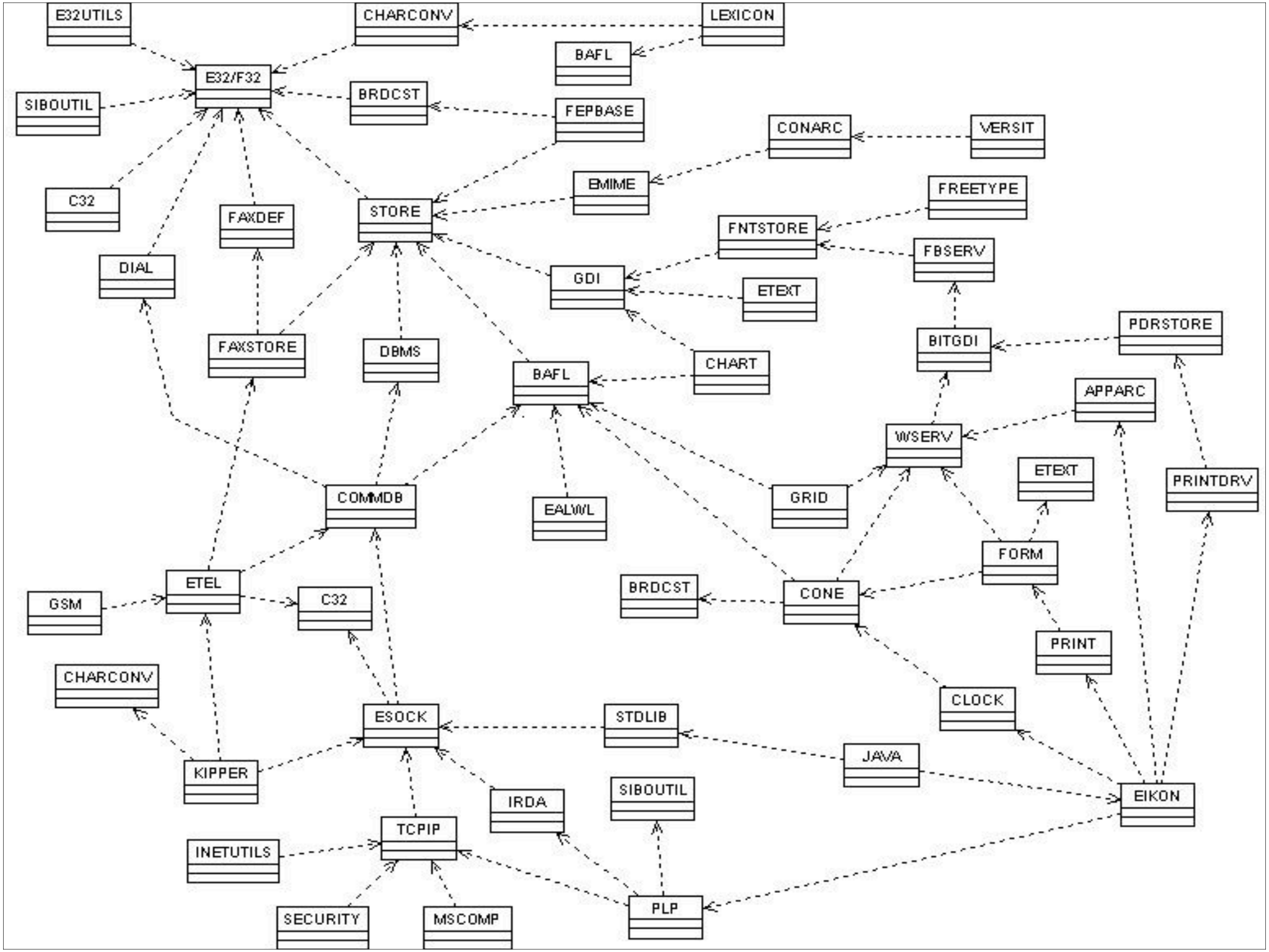
- what needs to be uniform for teamwork?
  - code style, naming, whitespace, idioms
  - OS, libraries, test framework, source-code control, issue tracking system
  - but NOT tools (editor, debugger, IDE...)
- the most important tool: a good source-code control system
  - & accompanying scripts
  - continuous build, automatic tests, ...
- 2nd most important: issue-tracking system that intearates well with the SCCS

Google

Is there a silver bullet?

- no, but many needles may keep 'em at bay!

Google


E32UTILS
CHARCONV
LEXICON
BAFL
E32/F32
SIBOUTIL
BRDCST
FEPBASE
CONARC
VERSIT
C32
BMIME
FREETYPE
FAXDEF
STORE
FNTSTORE
FBSERV
DIAL
GDI
ETEXT
FAXSTORE
DBMS
BAFL
CHART
BITGDI
PDRSTORE
APPARC
WSERV
ETEXT
PRINTDRV
COMMDB
EALWL
GRID
FORM
ETEL
BRDCST
CONE
GSM
C32
PRINT
CHARCONV
CLOCK
ESOCK
STDLIB
JAVA
KIPPER
IRDA
SIBOUTIL
EIKON
TCPIP
INETUTILS
SECURITY
MSCOMP
PLP

Q?

A!

Google

"The Timeless Way of Building", C. Alexander
"Competing on the Edge", S. Brown, K. Eisenhardt
"Illusions", R. Bach
"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 Panning", 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
"Joel on Software", J. Spolski

Google