

"Good Enough" IS
Good Enough!

http://www.aleax.it/pycon13_geige.pdf

Google™

©2013 Google -- aleax@google.com

Some Cultural Assumptions...:

- everybody should always be striving for perfection at all times!
- settling for a software release that's anywhere below "perfect!" is a most regrettable compromise.
- do you mostly agree with these...? OR...:
- keep-it-simple, just-good-enough
 - launch early, launch often!
 - iteratively improve, enhance, refactor...

"Worse is Better"

- Richard Gabriel, 1989, a Lisp conference
 - "New Jersey" approach, AKA "WiB"
 - VS
 - "MIT/Stanford" approach, AKA "The Right Thing"
- years of debate afterwards (plenty of it by RG, sometimes as "Nickieben Bourbaki")...
 - on BOTH sides of the issue!-)

Worse-is-better (e.g: Unix)

- simplicity
 - implementation (esp!) AND interface
 - most important consideration in design
- correctness
 - (slightly) better be simple than correct
- consistency
 - "not overly inconsistent"
- completeness
 - can be sacrificed to any of the top 3
 - MUST be, if simplicity's threatened

"The Right Thing" ("MIT")

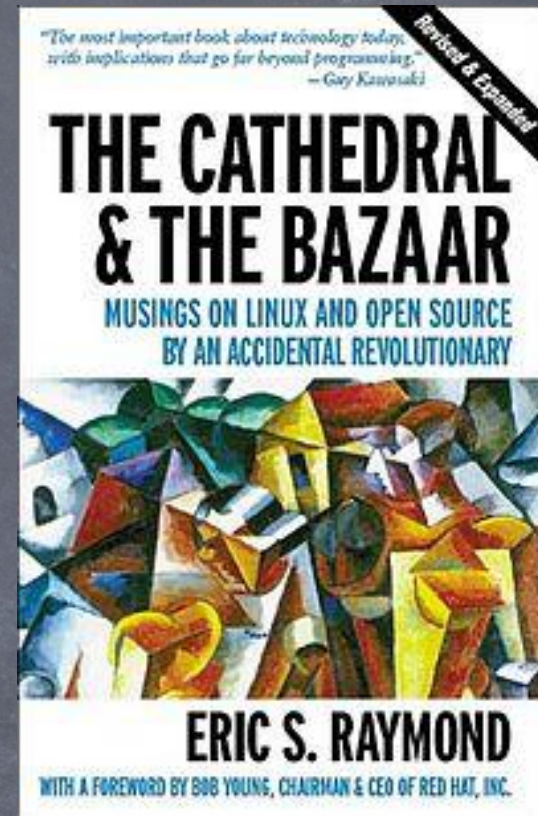
- simplicity
 - esp. interface
- correctness
 - absolute-must, top priority
- consistency
 - just as important as correctness
- completeness
 - roughly as important as simplicity

Quoting RG himself...:

- The right-thing philosophy is based on letting the experts do their expert thing all the way to the end before users get their hands on it.
- Worse-is-better takes advantage of the natural advantages of incremental development. Incremental improvement satisfies some human needs...

Cathedral, Bazaar...?

- Eric Raymond, 1997
- focus: two diverging models of software development
 - Cathedral: close to RG's "right-thing" MIT/Stanford
 - experts in charge
 - Bazaar: chaotic, launch-and-iterate NJ-like models -- crowd in charge
- The core Bazaar idea: "given enough eyeballs, all bugs are shallow"



BUGS?! I don't DO bugs!

- my very first program ever WAS bug-free
 - 1974: 3 freshmen HW design majors and a Fortran program to compute conditional probabilities of suit-division in bridge
 - we had to punch it into punched cards
 - we got one-&-only-one chance to run it...!
- it ran perfectly that first-and-only-time...!
- ...never ever happened again in my life.
- ...don't count on it, buddy...!-)

"Perfection" → BDUF

- If you want to only release "Perfection",
 - you clearly need "Big Design Up Front"
 - everything must proceed top-down,
 - perfect identification of requirements,
 - begets perfect architecture,
 - begets perfect design,
 - begets perfect implementations,
 - (it takes...) forever and ever, A-MEN!
 - alas! real life doesn't tend to co-operate...
 - stakeholders resent the "forever" part!-)

BDUF vs the real world

- requirements change all the time
 - you ain't ever gonna nail them perfectly!
 - architecture varies with design choices
 - design varies with implementation techs
 - implementation always has bugs
 - only discovered in real-world deployment
- >
- ITERATIVE development's the only way to go!
 - deploy SOMETHING, fix bugs, improve, ...
 - solve SOME user problems, win mindshare

"Perfect": verb, -adjective!

- perfecting your work is great
 - keep doing it -- based on real data!
- perfection is a process, NOT a state
 - you never "reach" it
 - goalposts keep shifting
 - no laurels to rest on!

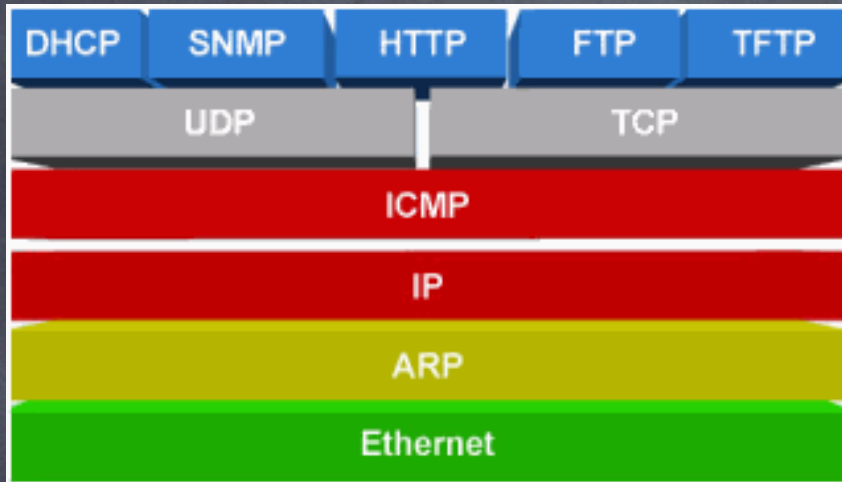
What **not** to skimp on

- light-weight, agile process and its steps
 - revision control, code reviews, testing...
 - proper release-engineering practices
- code style, clarity, elegance
- documentation



no cowboy coding!

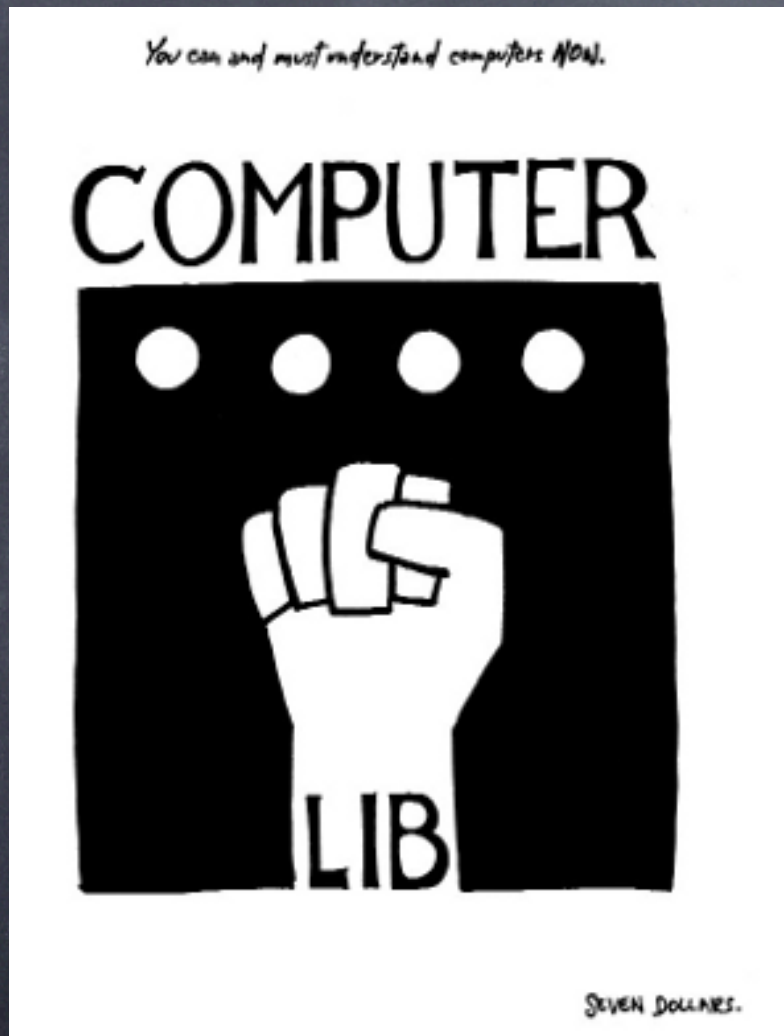
TCP/IP vs ISO/OSI



- rough consensus...
- ...and RUNNING CODE
(David Clark: MIT, but...
IETF front and center!)



Xanadu vs the WWW



Perfect, ideal
hypertext

HTML

CSS

PNG

GIF

JPEG

HTTP

URL

Hackish, incrementally
improved hypertext

Guess which one
conquered the world...?-))

Intr syscall: ITS vs Unix

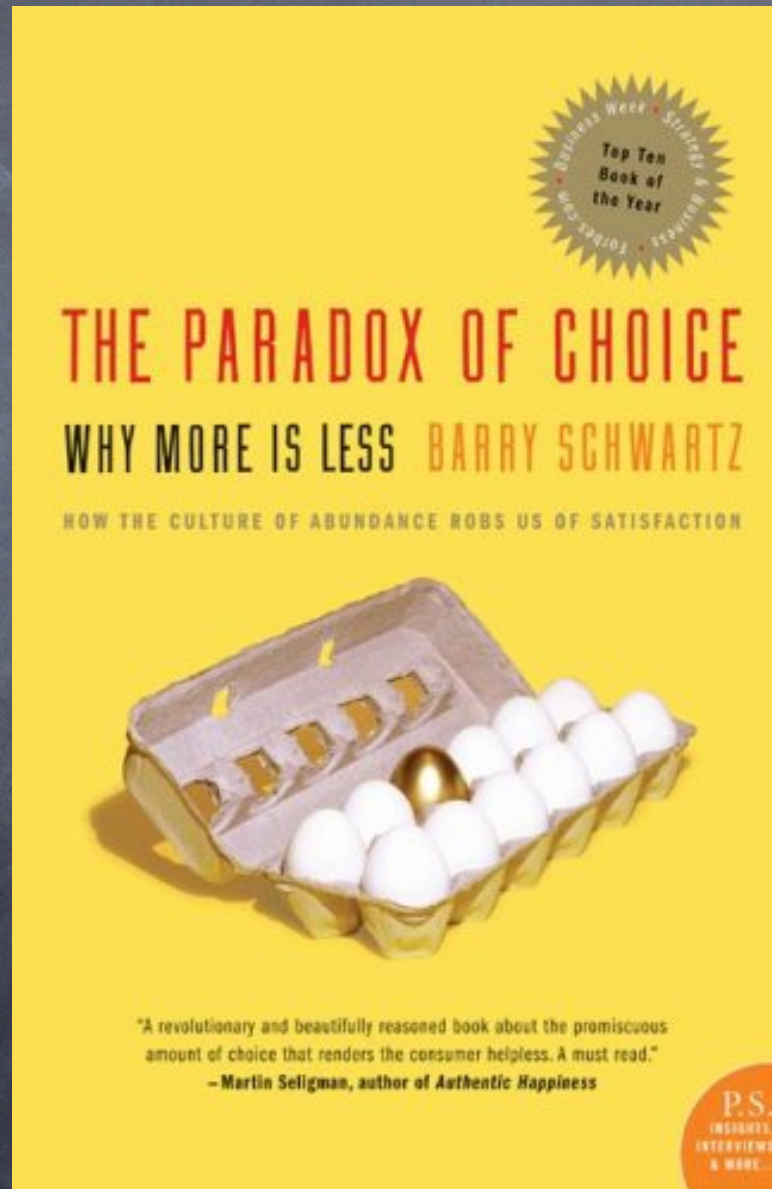
- MIT AI Lab's ITS:
 - every long-running syscall needs to be quasi-atomic AND interruptible...
 - so: every syscall must be able to...:
 - unwind state changes at ANY point
 - resume user-mode for intr. service
 - restart kernel-mode syscall again
- early Unix:
 - $\text{errno} \leftarrow \text{EINTR}$, return -1 -- that's it!-)

Satisficer vs Maximizer

Satisficer:

90% is just fine,
take it, move on!

80% may be OK
(20% of effort:
Pareto's Law)



Maximizer:
99.99% is NOT
100%,
so it's A FAIL!

Metaclass vs Decorator

```
class Meta(type):  
    def __new__(m, n, b, d):  
        cls = type.__new__(m, n, b, d)  
        cls.foo = 'bar'  
        return cls  
  
class X:  
    __metaclass__ = Meta
```

...VS...

```
def Deco(cls):  
    cls.foo = 'bar'  
    return cls  
  
@Deco  
class Y(object): pass
```


Good enough never is (or is it?)

- Eric Ries, <http://www.linkedin.com/today/post/article/20121008194203-2157554-good-enough-never-is-or-is-it>
- "Lean Startups" use the "middle way" to...:
- **minimum viable product**: that version of a new product which allows a team to collect the maximum amount of validated learning with the least effort
- 37signals' Hansson disagrees: "just build something awesome and ship it";-)

"Lowering expectations"?

- NO! our dreams **must** stay big! **BHAG!**
 - Rightly traced and well ordered: what of that? // Speak as they please, what does the mountain care?
- **however:** the best way TO those dreams remains "release early, release often"
 - **learn** from real users' interactions
- Ah, but a man's reach should exceed his grasp // Or what's a heaven for?
- Browning's Andrea del Sarto: **less is more!**

Q & A

http://www.aleax.it/pycon13_geige.pdf

