



tested

clean

efficient

corre

efficient

correct

tested

clean

tested

efficient

corre

correct

clean

tested

code.reviews

correct

efficient

Janos Gyerik

correct

tested





clean

correc



what is it?

NOT code reviewed

commit 
 commit 
commit 

code reviewed

commit 
 commit 
commit 
review 
 commit 
commit 
review 
commit 
accept 



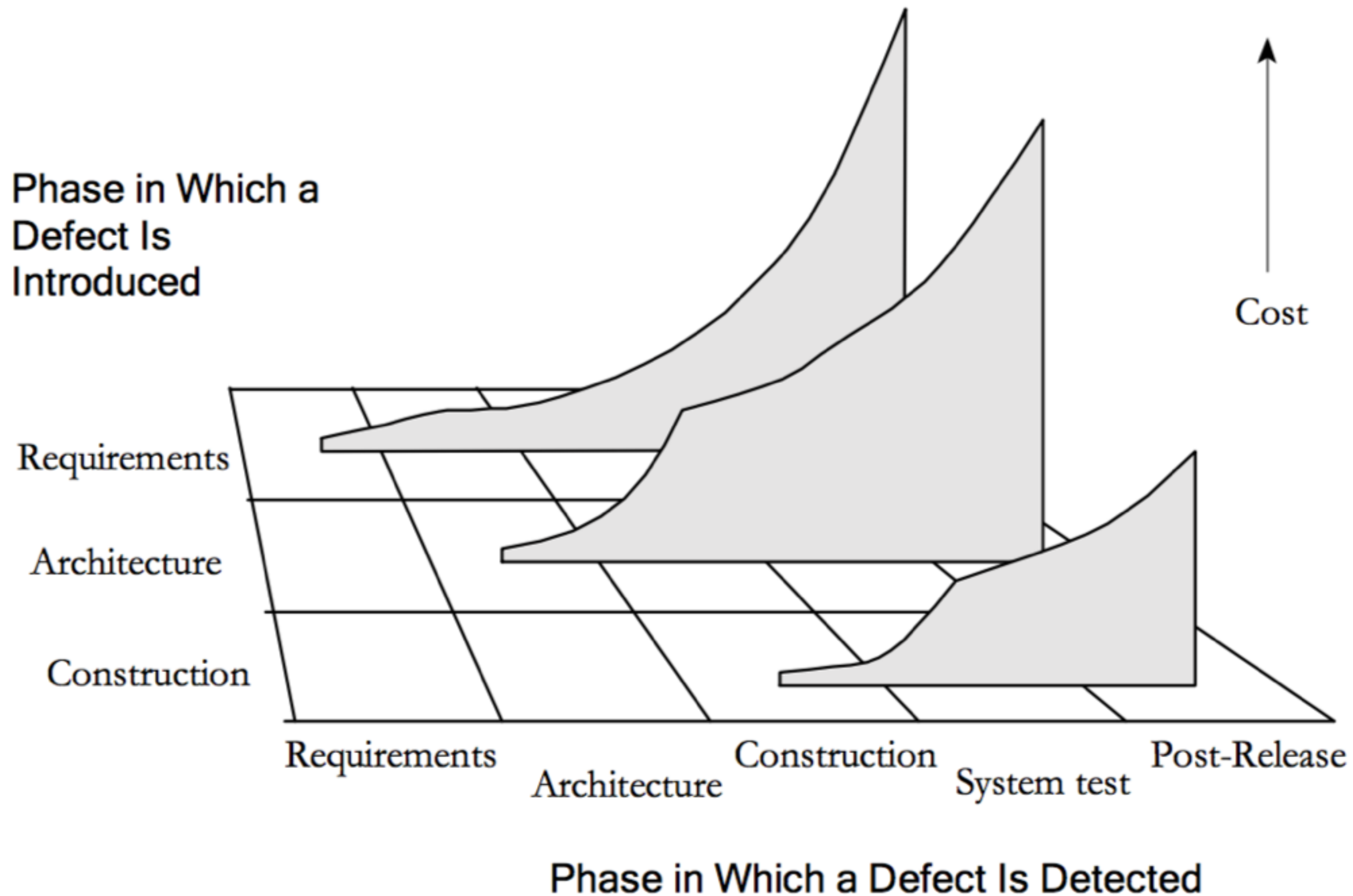
why do it?

catch early bugs
catch early bad patterns
catch early ugliness



save time





peer review 



peer pressure

anything that
gets reviewed
gets better

quality



time savings
cost savings



FACT

NOT code reviewed

buggy
messy
crappy
untested



code reviewed

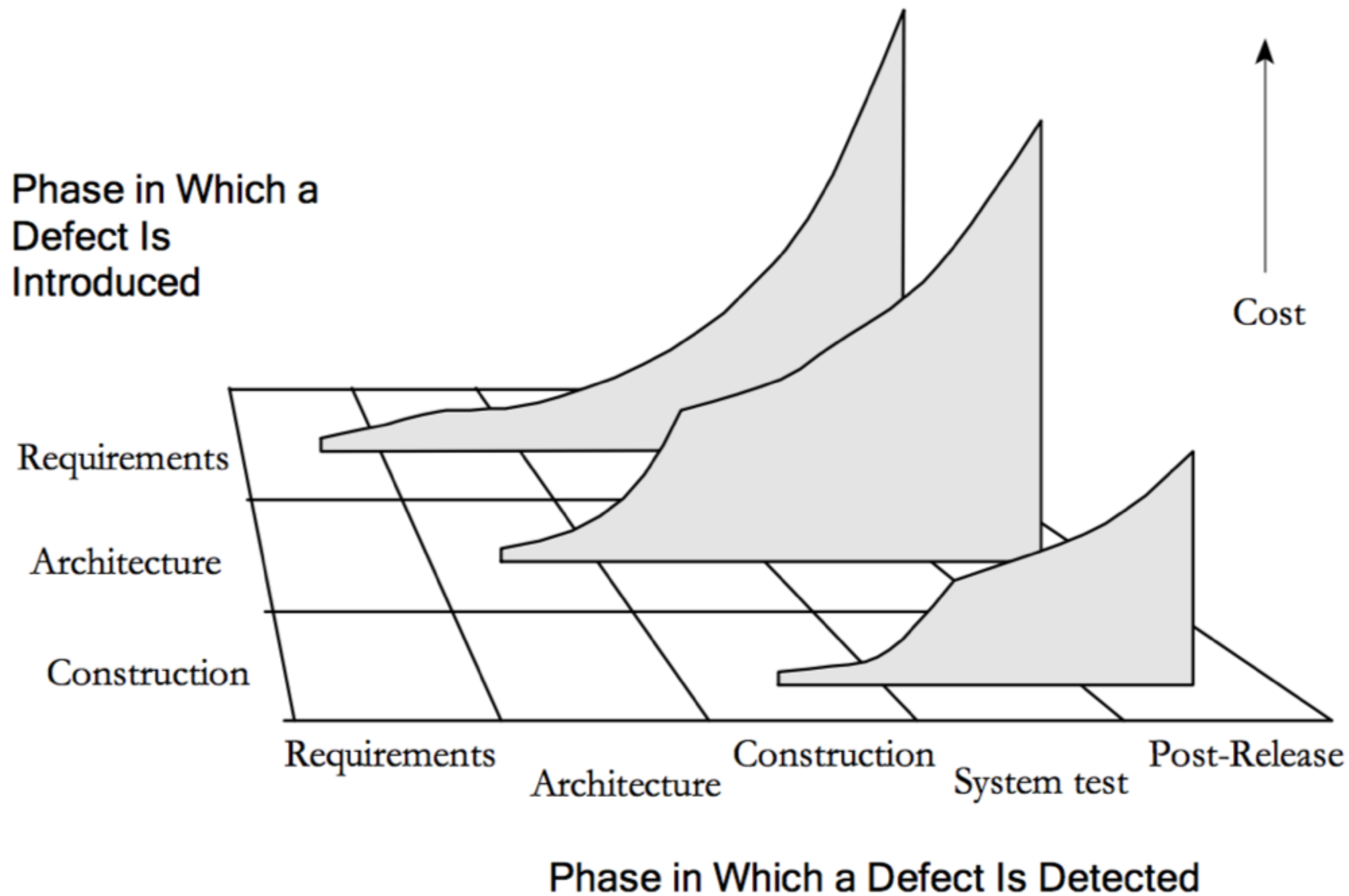
correct
readable
efficient
tested



information sharing 



NO MORE
bottleneck developers





why / when

NOT do it?

code reviews
are not practical when...

big bang
development



too hard to control

should be the exception,
not the norm!

code reviews
are not practical...

without
supporting
tools

tools help doing
code reviews
efficiently

don't waste your
time, get them
and use them!



what's
required?

*disciplined
commits*

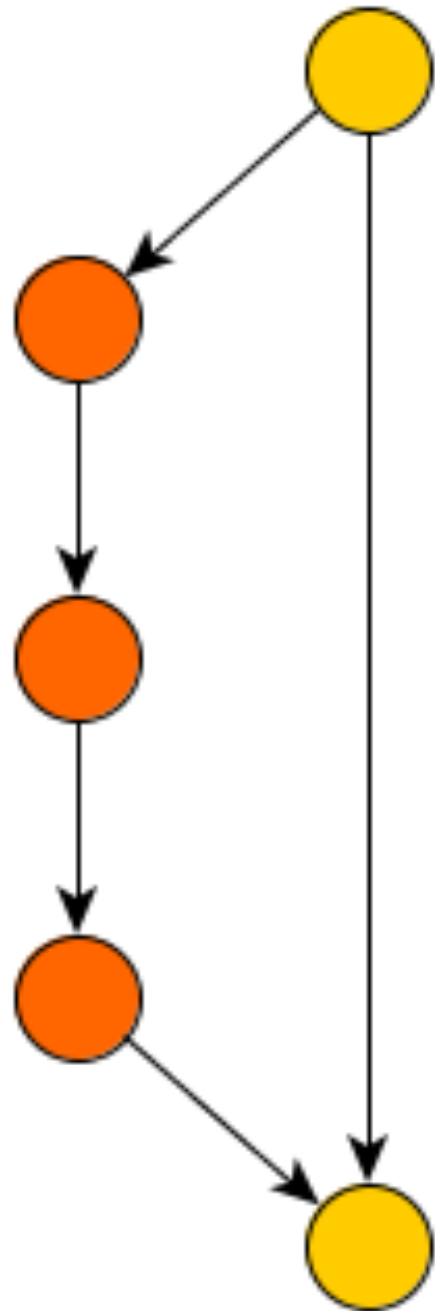


incremental changes

small and stable logical steps

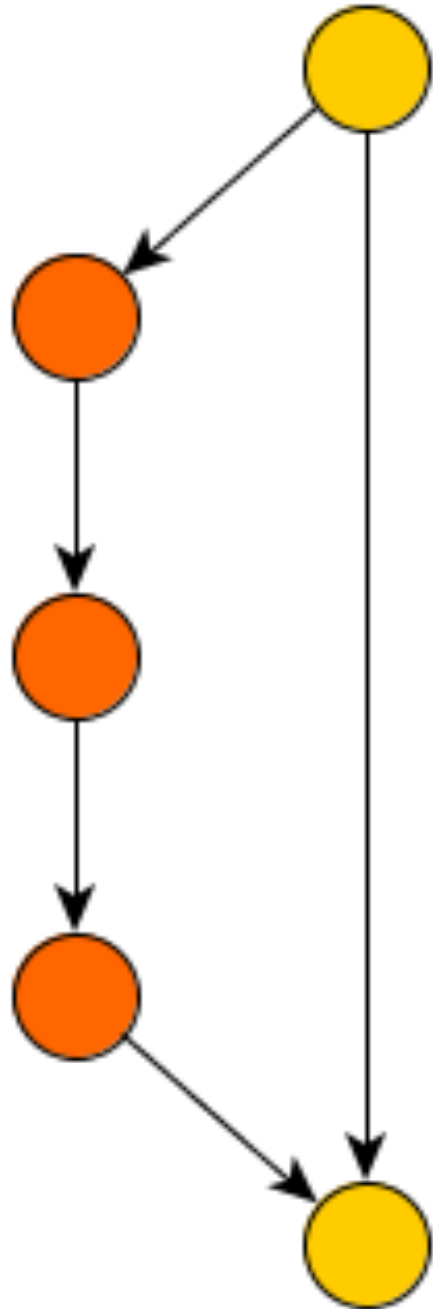
at all levels

feature



one feature
one purpose
one **branch**

feature



short-lived
max 3 days

(example good branch)

(example bad branch)



one commit
one logical
change



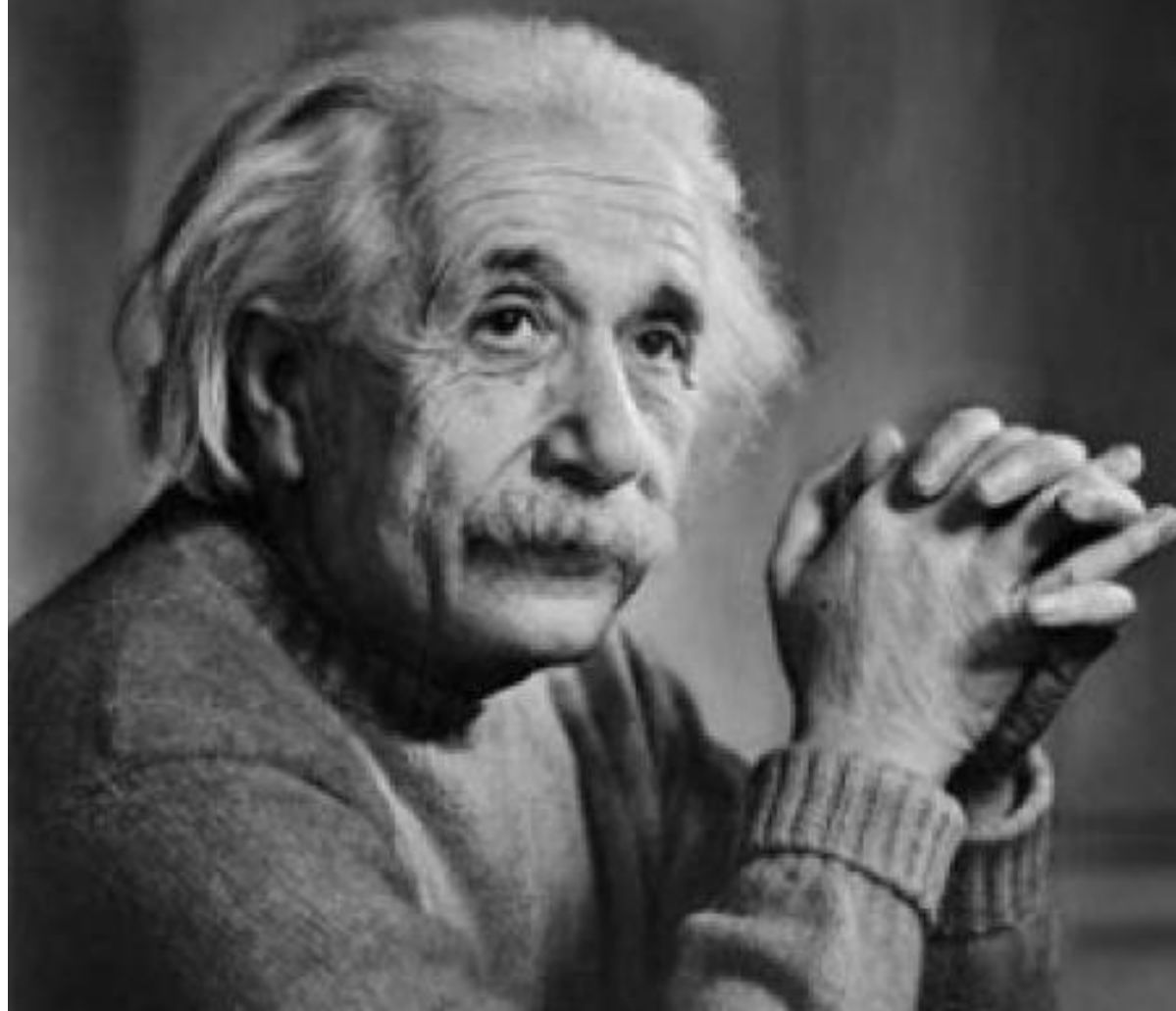
one good commit

==

stable build;
related changes;
no garbage; small;
good comment

If you can't explain it **simply**, you
don't understand it well enough.

– Albert Einstein



(example good commit)

example bad commit with many changes)

(example bad commit with garbage)



how to do it?


```
git fetch origin master
git checkout -b feature-x origin/main
# work work work
git commit
git commit
git push origin feature-x
# create merge request
```

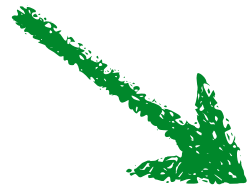
Create Merge Request

create; don't assign!

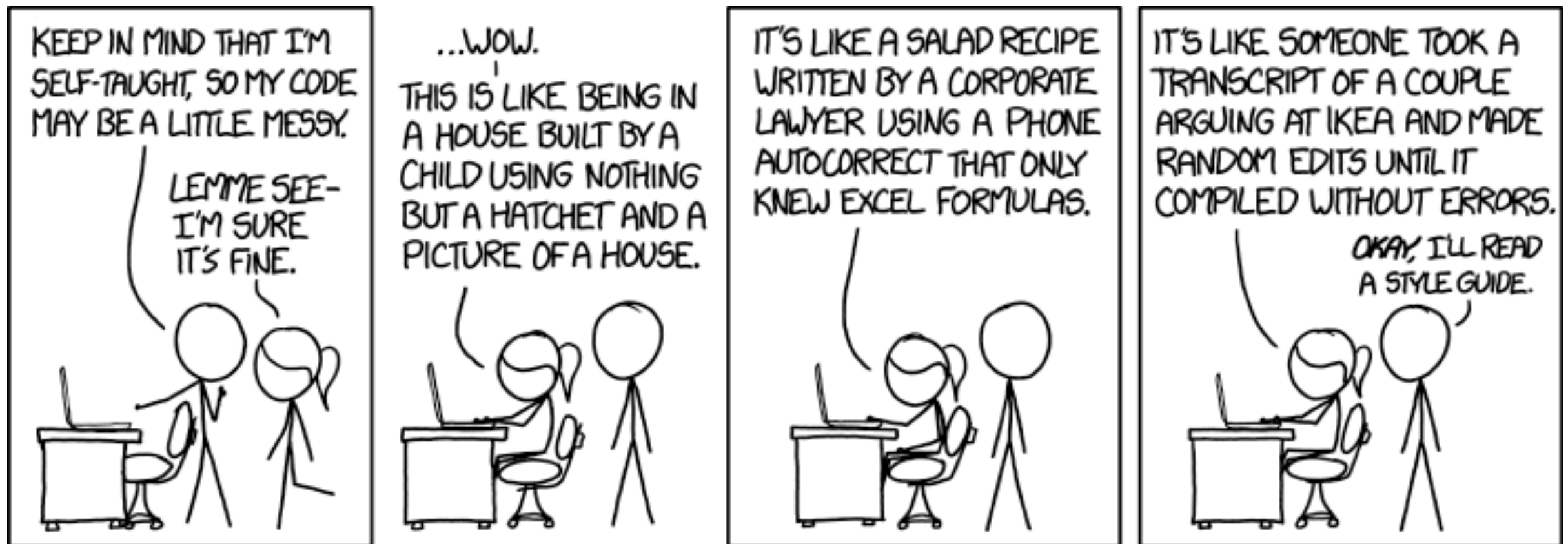
self-review: any WTFs?

ready? assign!

peer review



peer pressure



what to review?

NOT code reviewed

buggy
messy
crappy
untested



code reviewed

correct
readable
efficient
tested



readable

is it clear?

easy to read?

easy to understand?

**Code is read far more often
than written!**



FACT

correct

is the logic sane?

does it work?

bug suspects?

-> ask!

efficient

**any performance
concerns?**

-> ask!

tested

unit tests included?

unit test

opportunities?

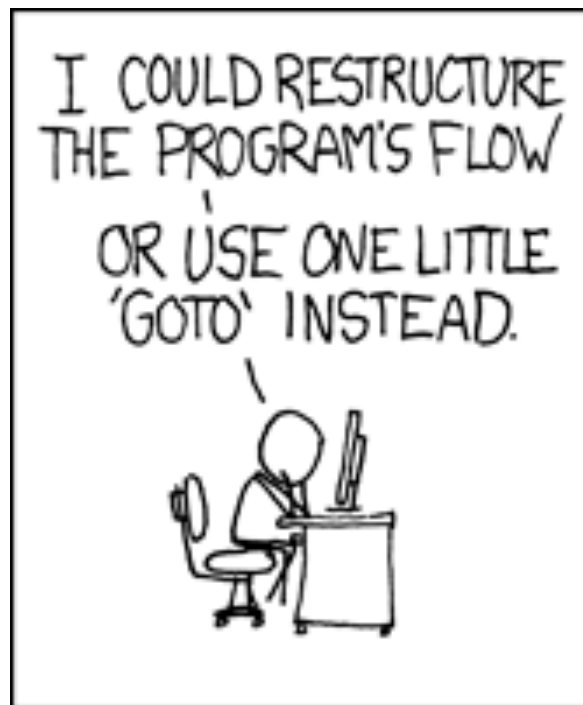
Code Complete


Effective Java

Sonar/Findbugs/...

codereview.stackexchange.com

how bad can it be?





how in-depth?

- not too much
- not too little
- just right
- DO IT FAST



attitude?

a code review is...

NOT about the developer
it is about the **code**

don't just say
something is "wrong"

suggest a better way

the focus is..

NOT on problems
it is on **solutions**

perfect code?

don't seek perfect

seek good enough

better is good enough

don't be a pain in the ass

be flexible

be constructive

The only valid measurement: WTFs/min

3

Good code



Clean Code

Bad code



07/03/2013

mistakes...

it's OK to make mistakes

it's NOT OK to not learn
from them

A motivational poster featuring a muscular man in a crowd. The man is shirtless, showing his well-defined muscles, and is looking upwards with his arms raised in a gesture of triumph or celebration. He is holding a large, glowing orange sphere in his right hand. The background is a dark, crowded arena with many people visible, suggesting a large-scale event or competition. The lighting is dramatic, with a strong blue glow on the right side and a warm orange glow on the left side. The text "DON'T BE AFRAID TO FAIL." is overlaid in the center in a bold, white, distressed font.

DON'T BE AFRAID TO FAIL.