Twelve Ways to Make Code Suck Less

Venkat Subramaniam

venkats@agiledeveloper.com @venkat_s

Why should we care about code quality?

We can’t be agile if our code sucks

Code is how we tell our colleagues how we feel about them.

"Lowering quality lengthens development time."—First Law of Programming.

http://c2.com/cgi/wiki?FirstLawOfProgramming

What’s Quality Code?

The quality of code is inversely proportional to the effort it takes to understand it.

http://blog.agiledeveloper.com/2010/05/thoughts-through-tweets_15.html

Twelve ways We can Help

12. Schedule Time to Lower Technical Debt

What’s Technical Debt?

Ward Cunningham

http://c2.com/cgi/wiki?WardExplainsDebtMetaphor

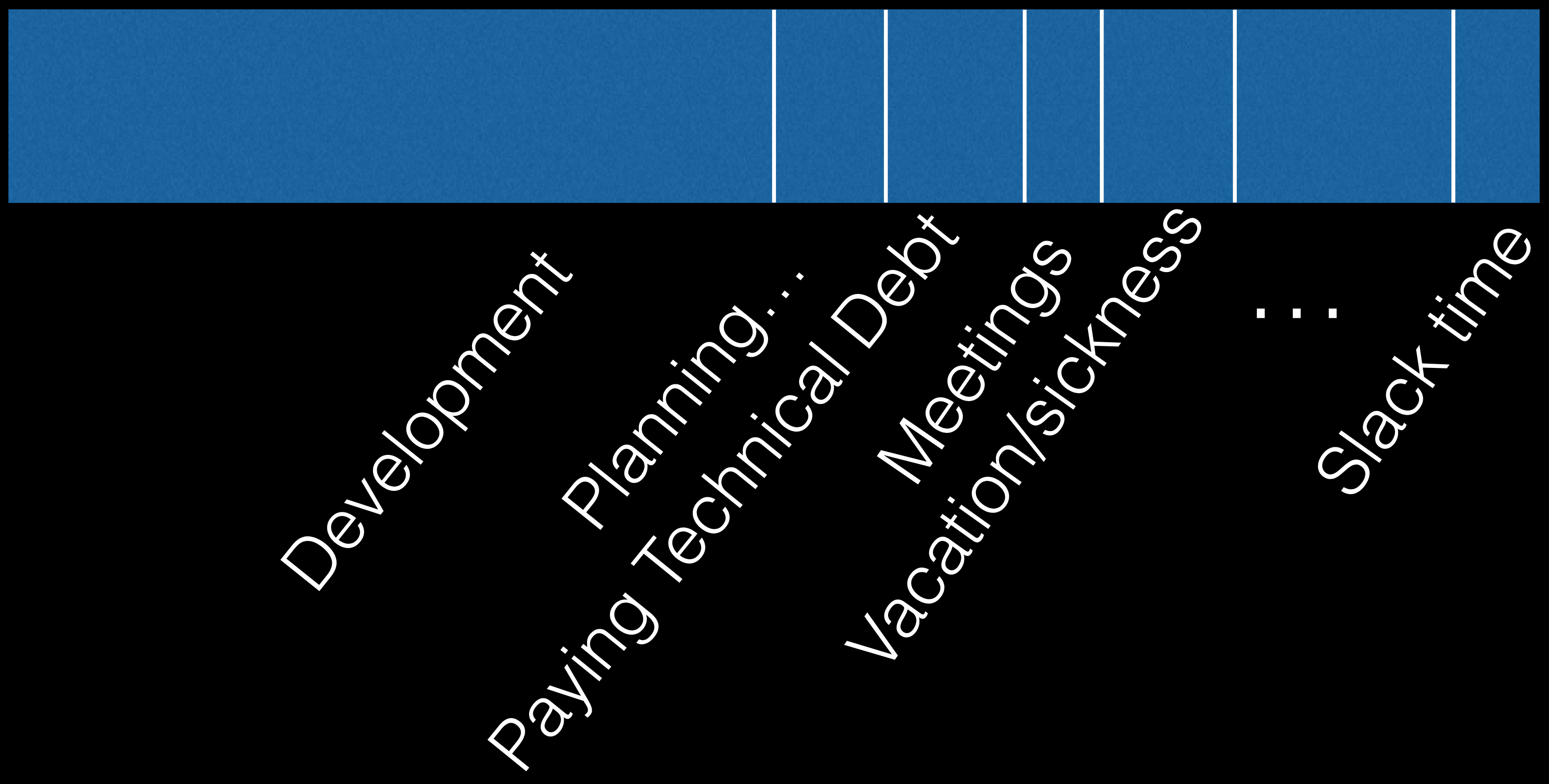
Example of Technical Debts

Low quality is not technical debt

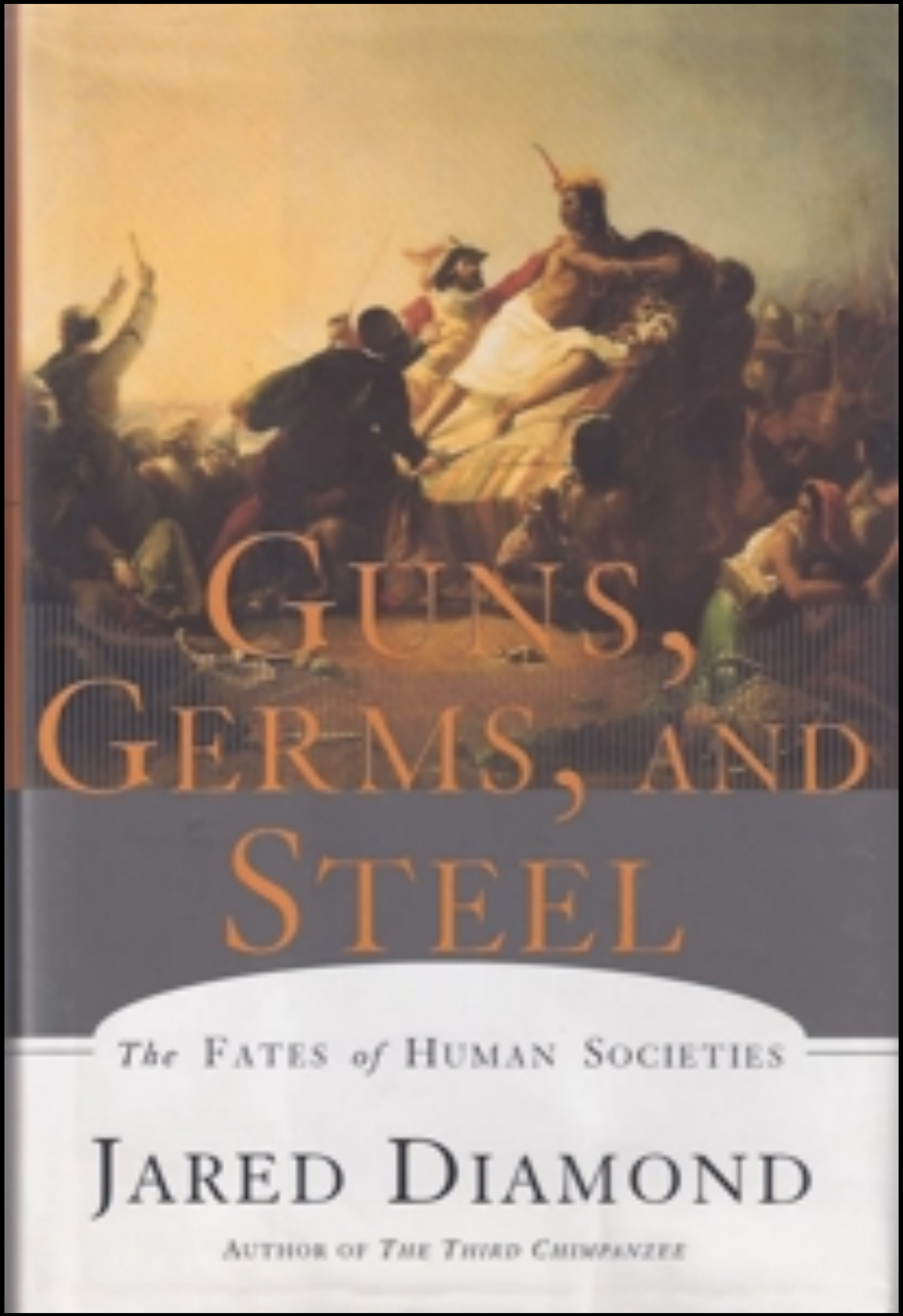
What Causes Technical Debt?

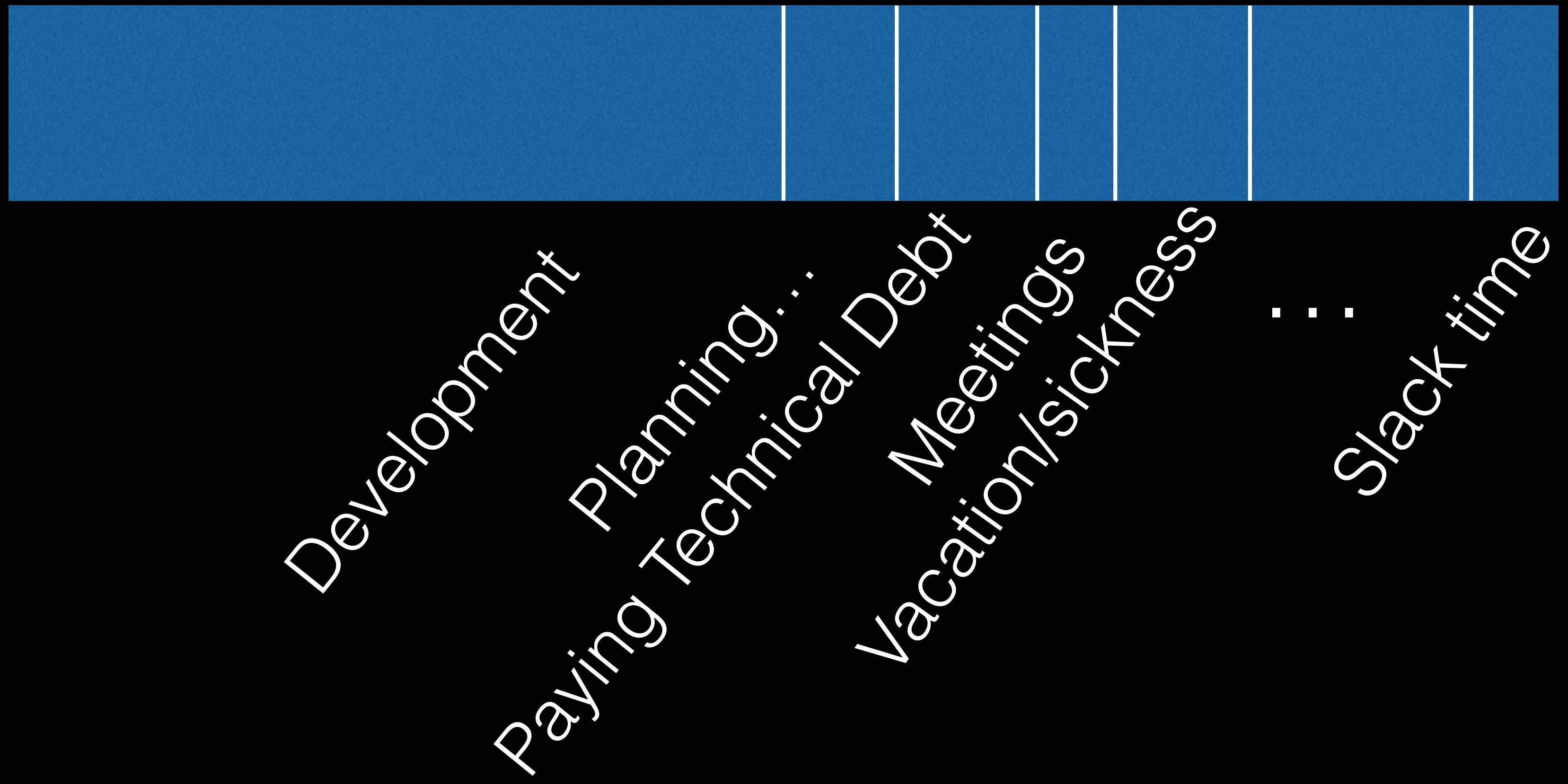
Not “All or Nothing”

Schedule Time


Development
Planning…
Paying Technical Debt
Meetings
Vacation/sickness
…
Slack time

Linda Rising


GUNS, GERMS, AND STEEL
The FATES of HUMAN SOCIETIES
JARED DIAMOND
AUTHOR OF THE THIRD CHIMPANZEE


Development
Planning…
Paying Technical Debt
Meetings
Vacation/sickness
…
Slack time

11. Favor High Cohesion

Narrow, focused,
does only one thing well

Why?

Think about frequency of change

high cohesion ==

low cyclomatic complexity

10. Favor Loose Coupling

Tight coupling make code

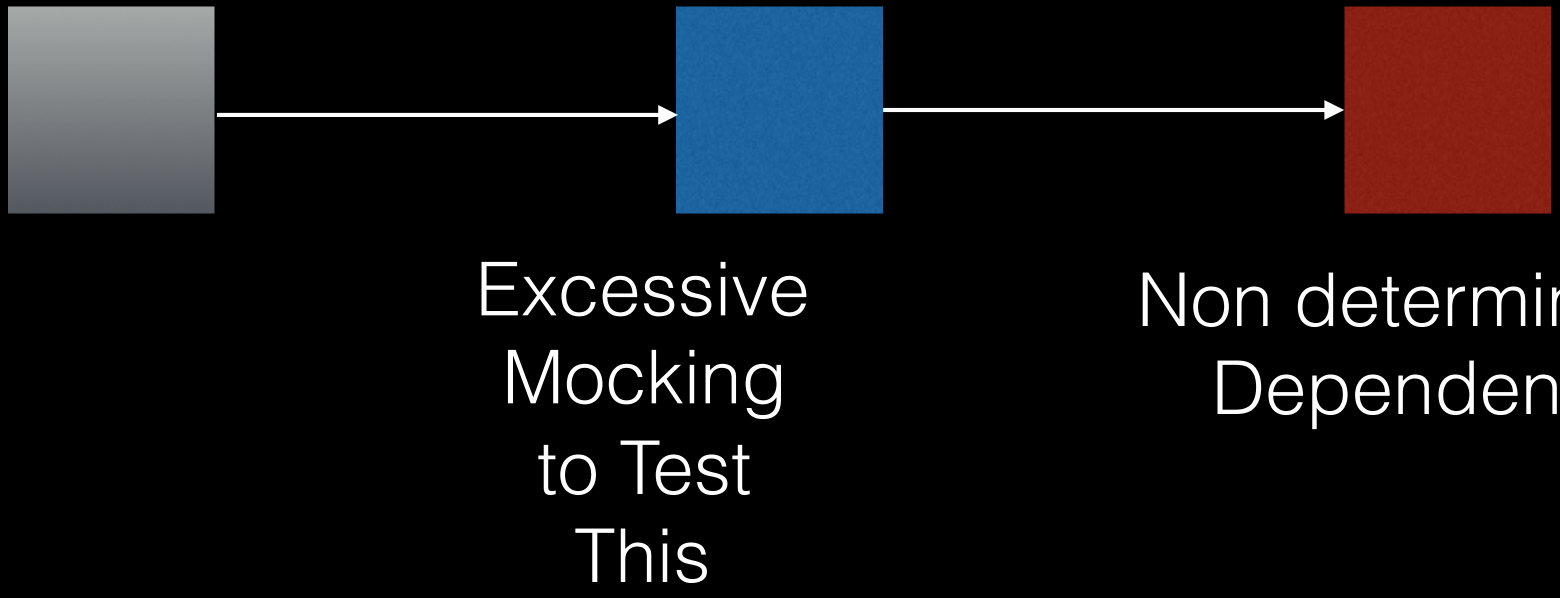
hard to extend

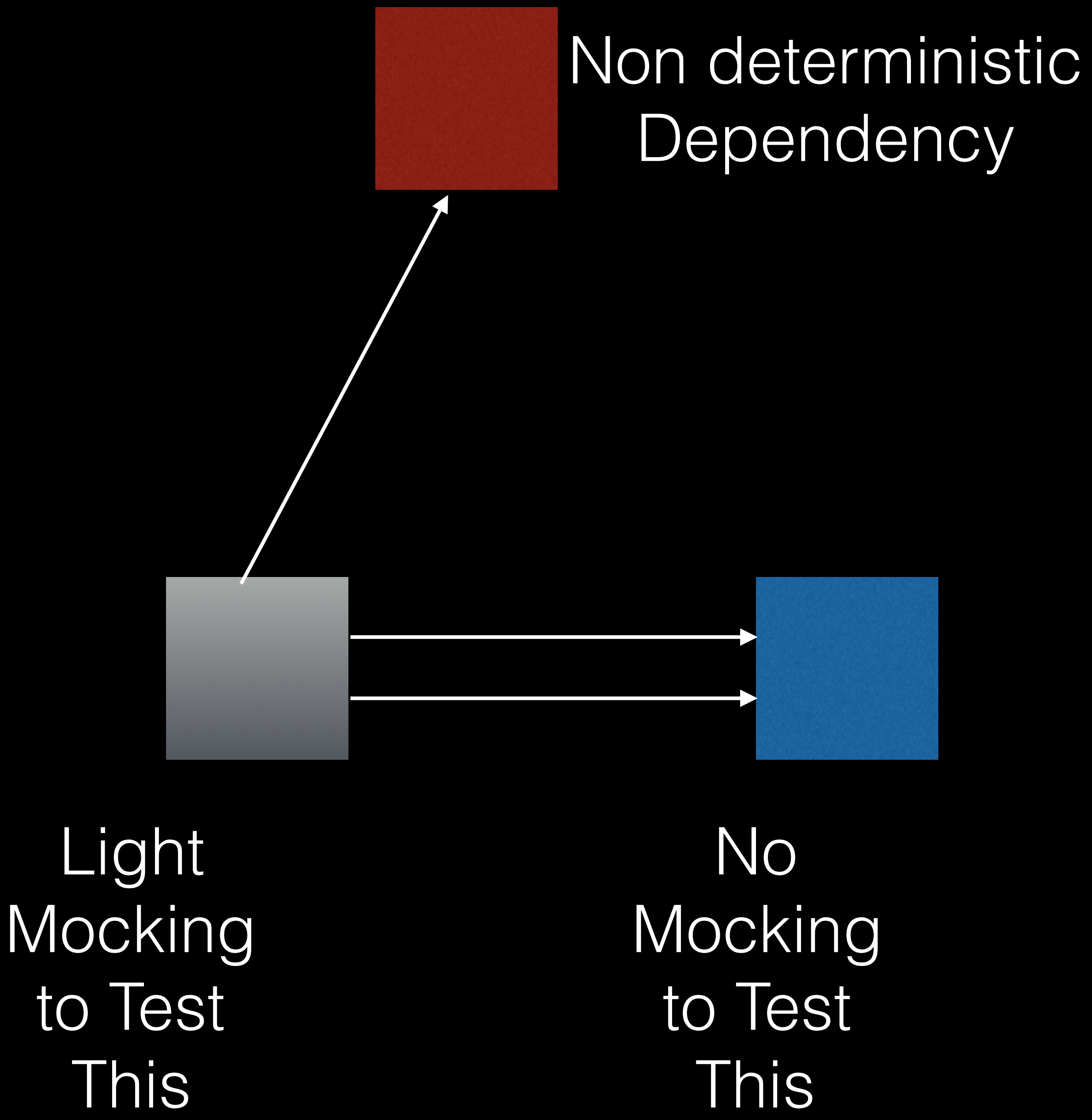
hard to test

Lower coupling

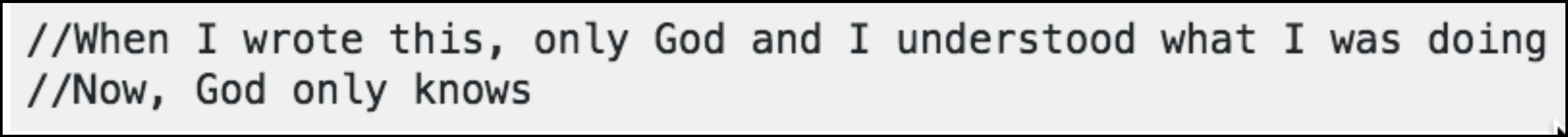
Loose vs. tight coupling

Eliminate where possible


Excessive Mocking to Test This
Non deterministic Dependency


Non deterministic Dependency
Light Mocking to Test This
No Mocking to Test This

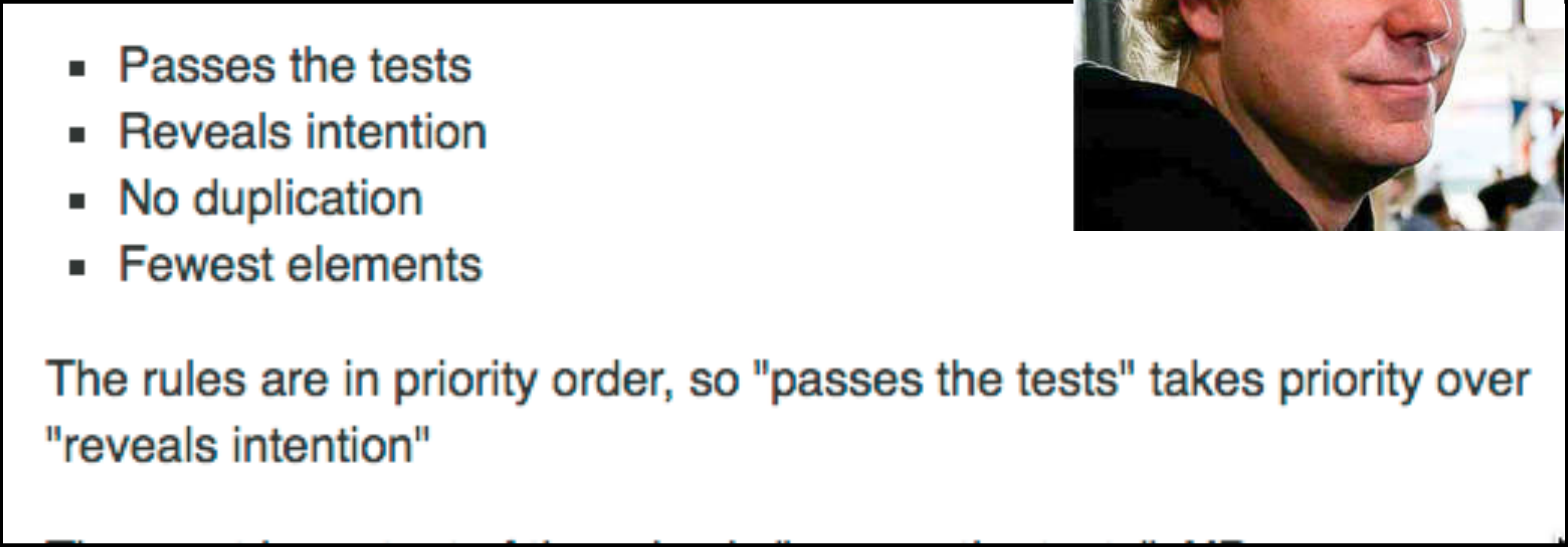
9. Program with Intention



```
//When I wrote this, only God and I understood what I was doing
//Now, God only knows
```

http://stackoverflow.com/questions/184618/what-is-the-best-comment-in-source-code-you-have-ever-encountered

Beck's Rule for Simple Design



- Passes the tests
- Reveals intention
- No duplication
- Fewest elements

The rules are in priority order, so "passes the tests" takes priority over "reveals intention"

http://martinfowler.com/bliki/BeckDesignRules.html

Programming Deliberately

- When you write test before writing code…

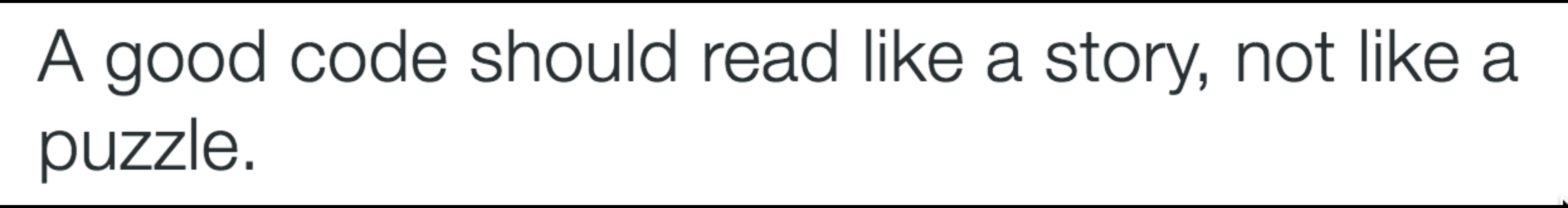
8. Avoid Primitive Obsession

Imperative code is packed with accidental complexity.

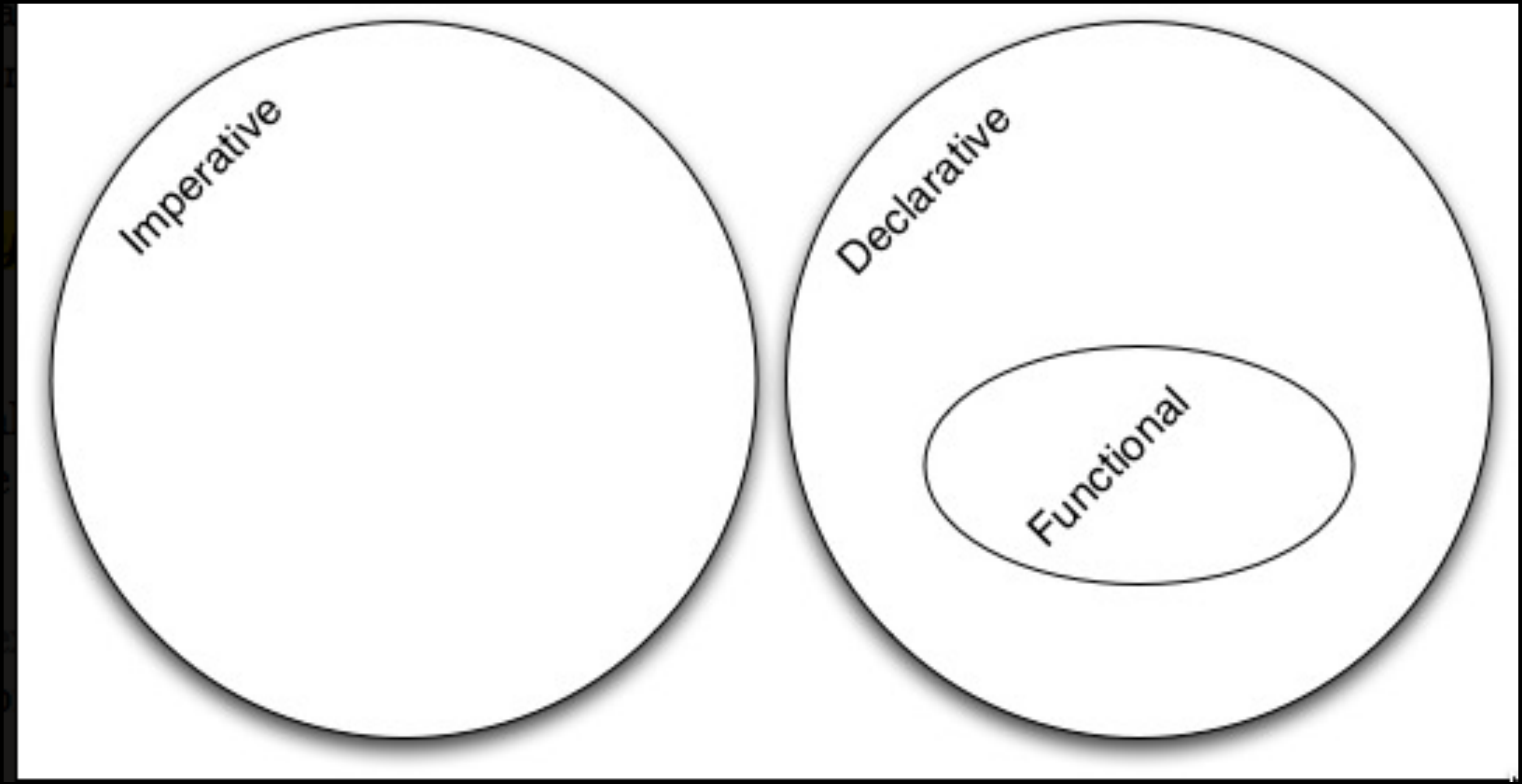
```
//Find the total of sqrt of first k primes starting with n
public static double compute(int n, int k) {
  int index = n;
  int count = 0;
  double total = 0;

  while(count < k) {
    if(isPrime(index)) {
      total += Math.sqrt(index);
      count++;
    }
    index++;
  }

  return total;
}
```

It’s code like this that prematurely turns programmers into managers

```
//Find the total of sqrt of first k primes starting with n
public static double compute(int n, int k) {
  return Stream.iterate(n, e -> e + 1)
               .filter(Sample::isPrime)
               .mapToDouble(Math::sqrt)
               .limit(k)
               .sum();
}
```



A good code should read like a story, not like a puzzle.



Functional Style == Declarative Style + Higher Order Functions

http://blog.agiledeveloper.com/2015/08/the-functional-style.html

7. Prefer Clear Code over Clever Code

```
Programs must be written for people to read,
and only incidentally for machines to execute.
                            -Abelson and Sussman
```

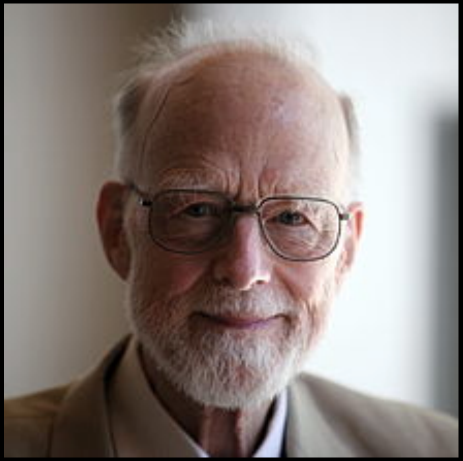
10% of the time, we write ugly code for performance reasons, the other 90% of the time, we write ugly code to be consistent.

http://blog.agiledeveloper.com/2010/05/thoughts-through-tweets_15.html

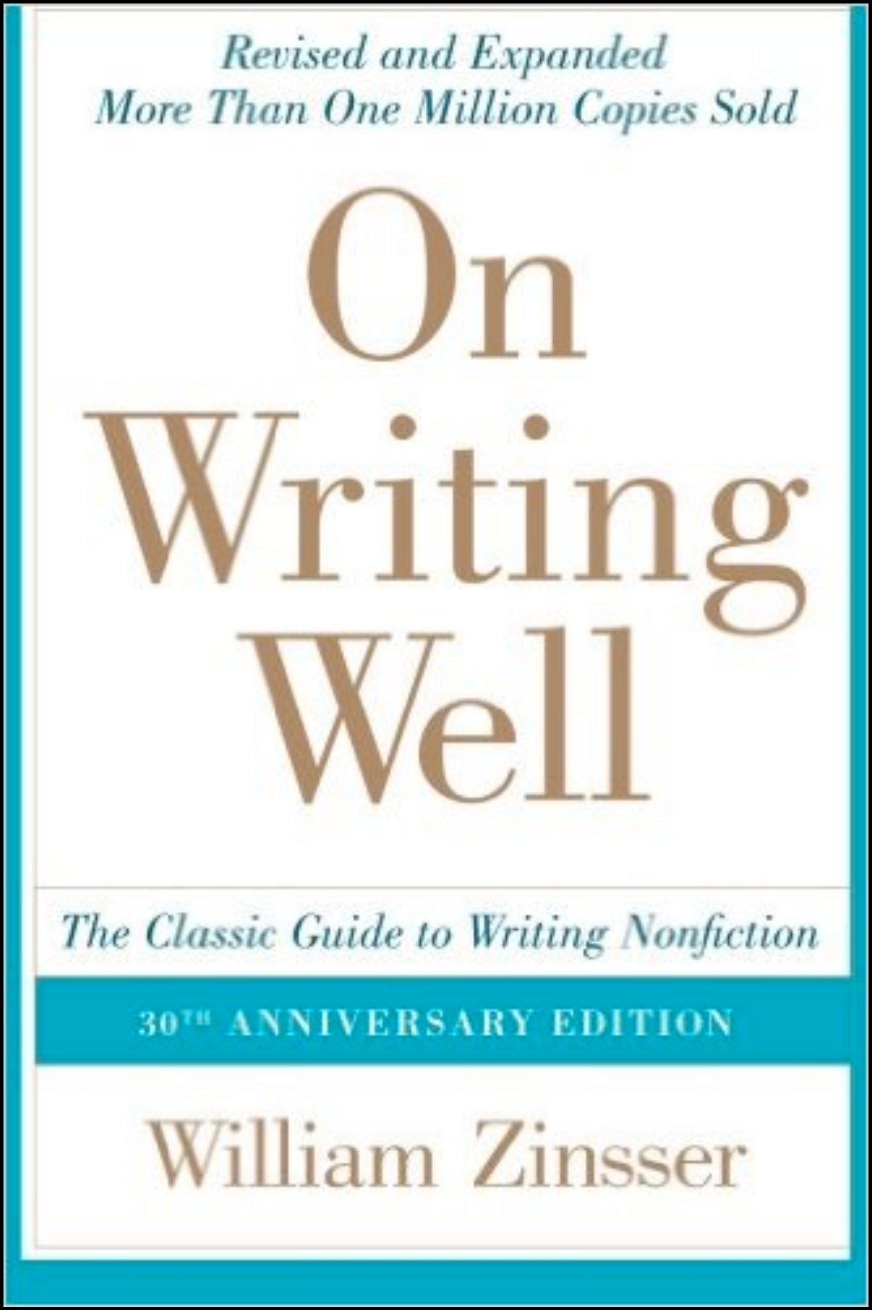
Those who sacrifice quality to get performance may end up getting neither.

http://blog.agiledeveloper.com/2010/05/thoughts-through-tweets_15.html

“There are two ways of constructing a software design. One way is to make it so simple that there are obviously no deficiencies and the other is to make it so complicated that there are no obvious deficiencies”— Tony Hoare



6. Apply Zinsser's Principle on Writing


Revised and Expanded
More Than One Million Copies Sold

On Writing Well

The Classic Guide to Writing Nonfiction

30TH ANNIVERSARY EDITION

William Zinsser

Simplicity

Clarity

Brevity

Humanity

5. Comment Why, not What

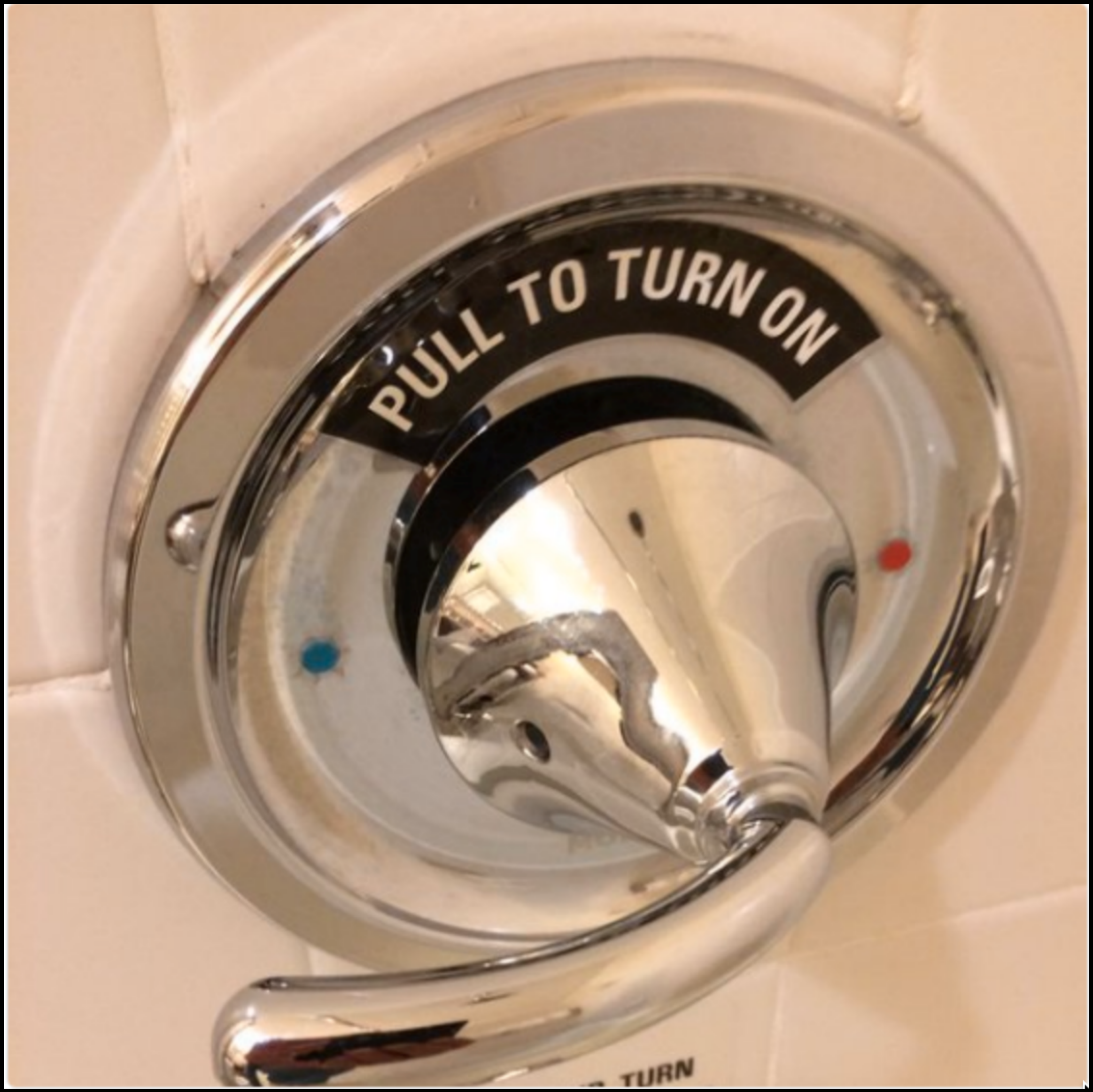
Don’t comment to cover up bad code

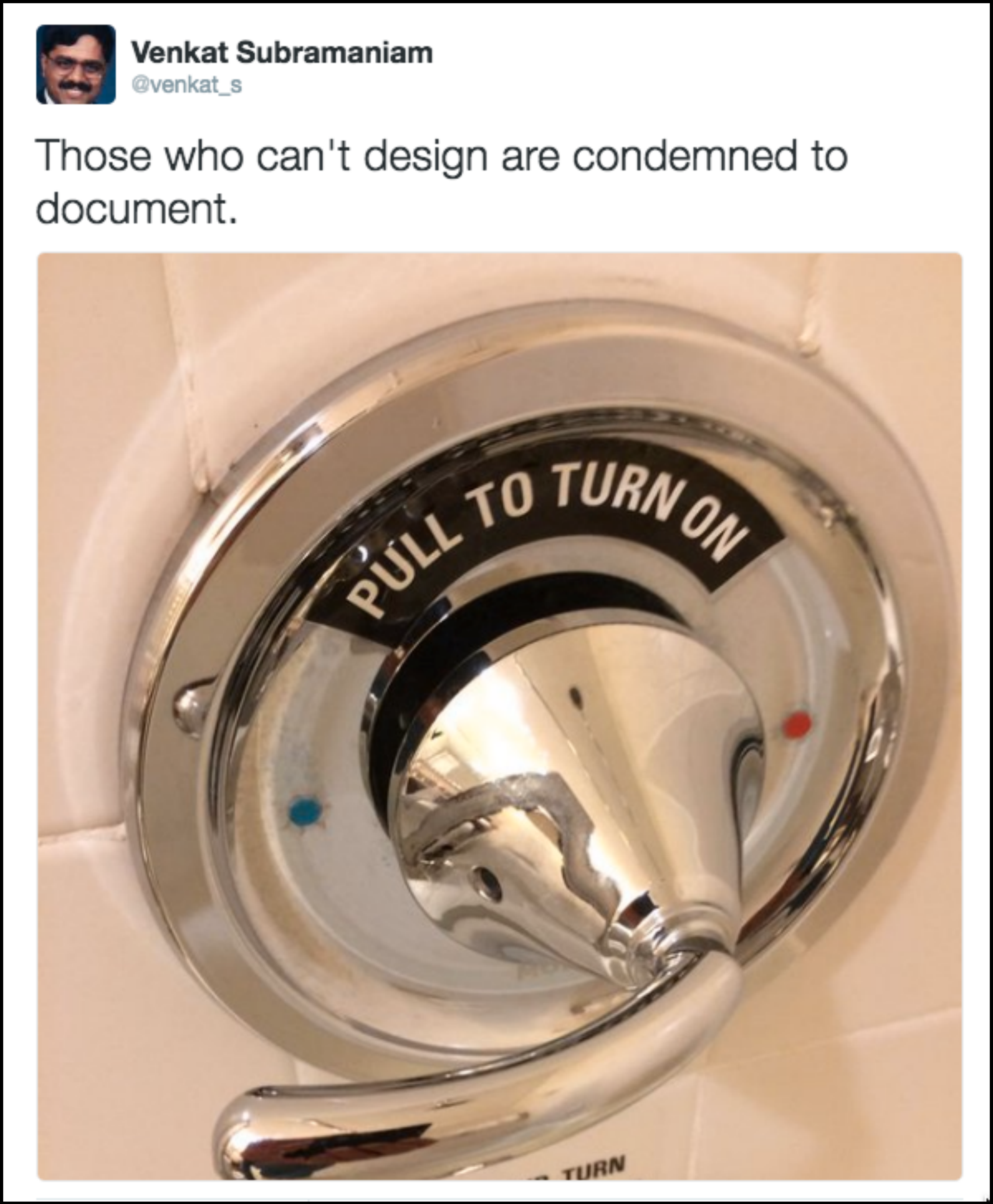
Write Expressive Self-Documenting Code

A good code is like a good joke

Writing comments is like explaining a joke

http://blog.agiledeveloper.com/2006/01/comments-on-comments.html


PULL TO TURN ON
TURN



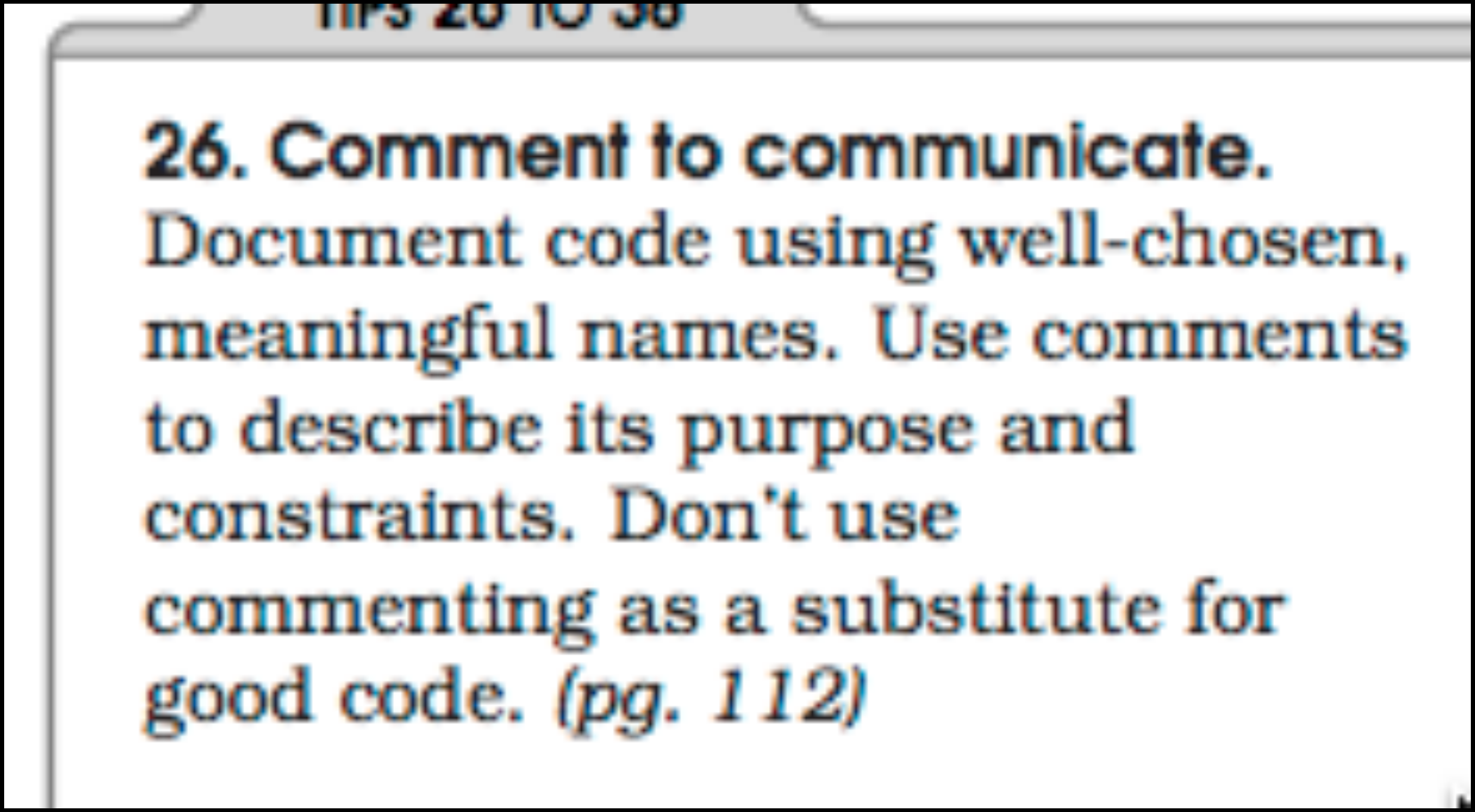
Venkat Subramaniam
@venkat_s

Those who can't design are condemned to document.

```
order(3)                3 cups?…
```

```
order(3) // large coffee
```

```
order(CoffeeSize.LARGE)
```



TIPS 26 TO 38

26. Comment to communicate.
Document code using well-chosen, meaningful names. Use comments to describe its purpose and constraints. Don't use commenting as a substitute for good code. *(pg. 112)*

https://media.pragprog.com/titles/pad/PAD-pulloutcard.pdf

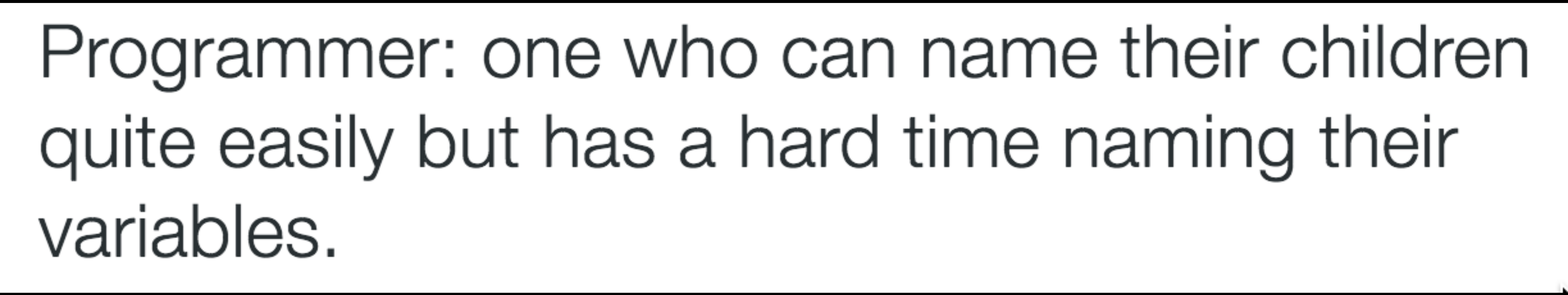
4. Avoid Long Methods—Apply SLAP

Perils of Long Methods

How long is long?

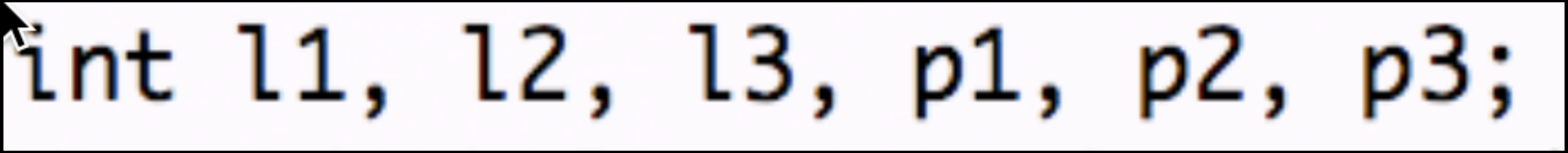
Turns out long is not about length of code, but levels of abstraction

3. Give Good Meaningful Names

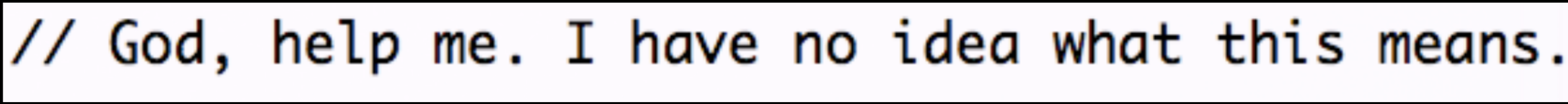


Programmer: one who can name their children quite easily but has a hard time naming their variables.

```
int l1, l2, l3, p1, p2, p3;
```

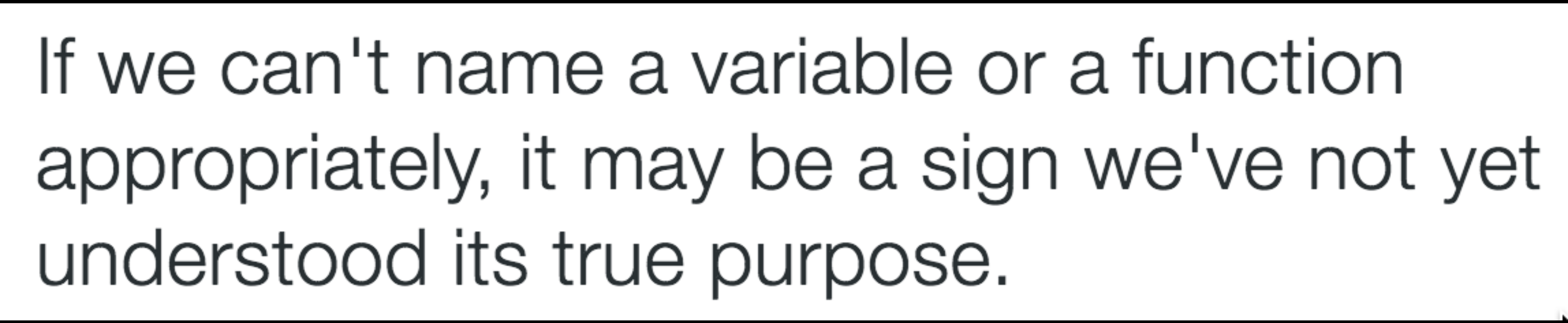


```
int l1, l2, l3, p1, p2, p3;
```



```
// God, help me. I have no idea what this means.
```

Variable Names Represent Abstractions



If we can't name a variable or a function appropriately, it may be a sign we've not yet understood its true purpose.

2. Do Tactical Code Reviews

Peer reviews catch 60% of defects.

Perspective-based reviews catch 35% more defects than nondirected reviews.

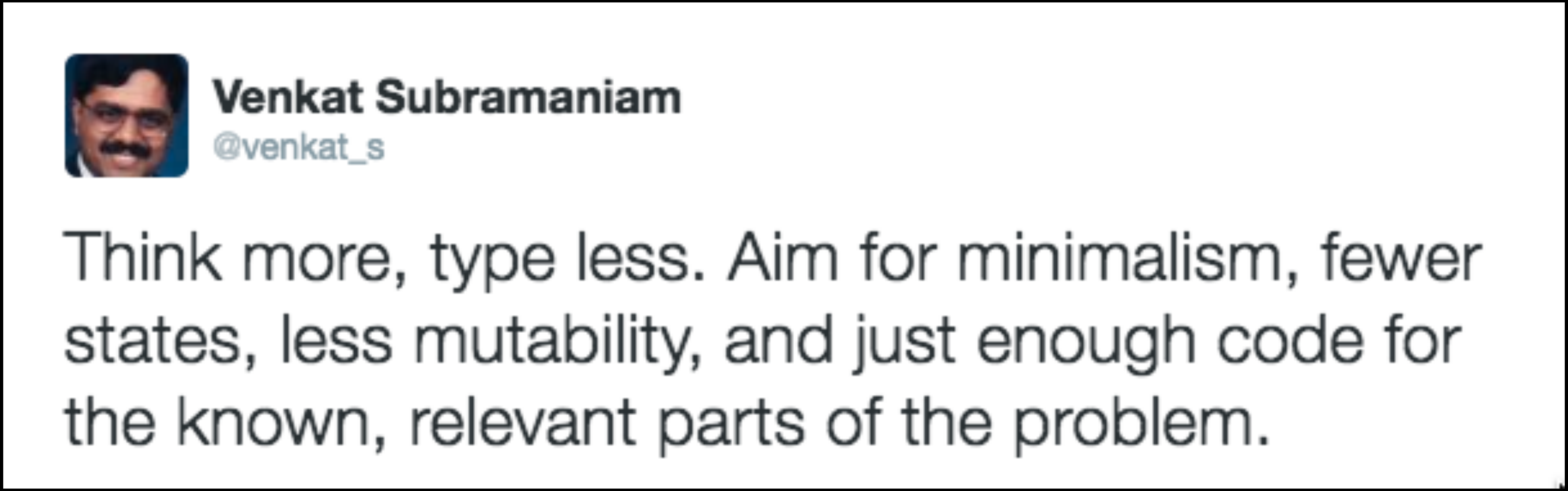
Peer reviews complement testing.

Technical and social activity.

https://www.cs.umd.edu/projects/SoftEng/ESEG/papers/82.78.pdf

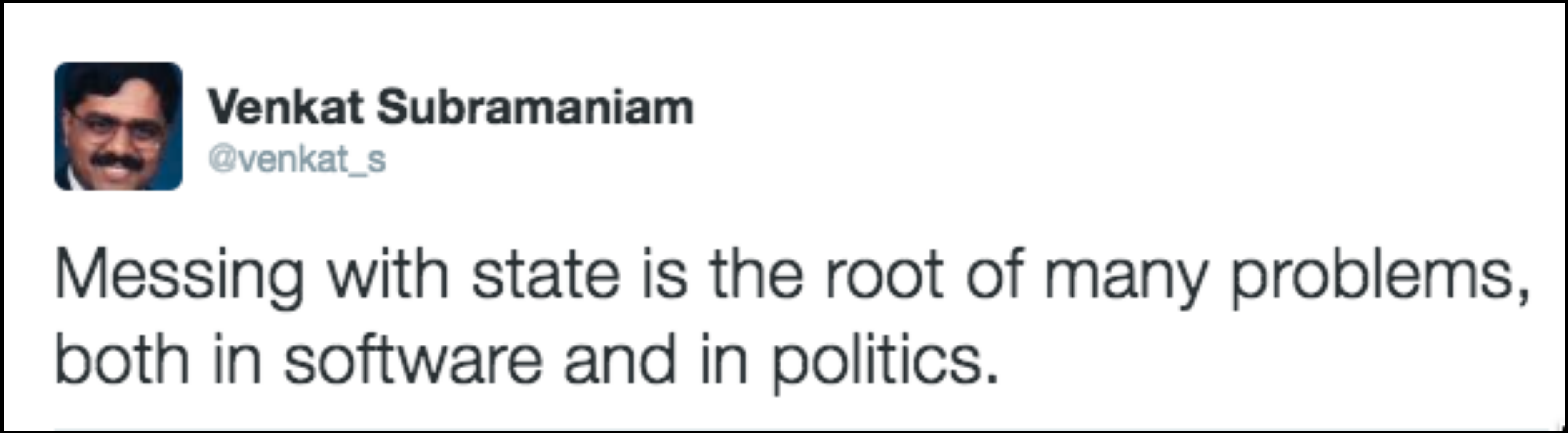
Facilitating Tactical Code Reviews

1. Reduce State & State Mutation



Venkat Subramaniam
@venkat_s

Think more, type less. Aim for minimalism, fewer states, less mutability, and just enough code for the known, relevant parts of the problem.



Venkat Subramaniam
@venkat_s

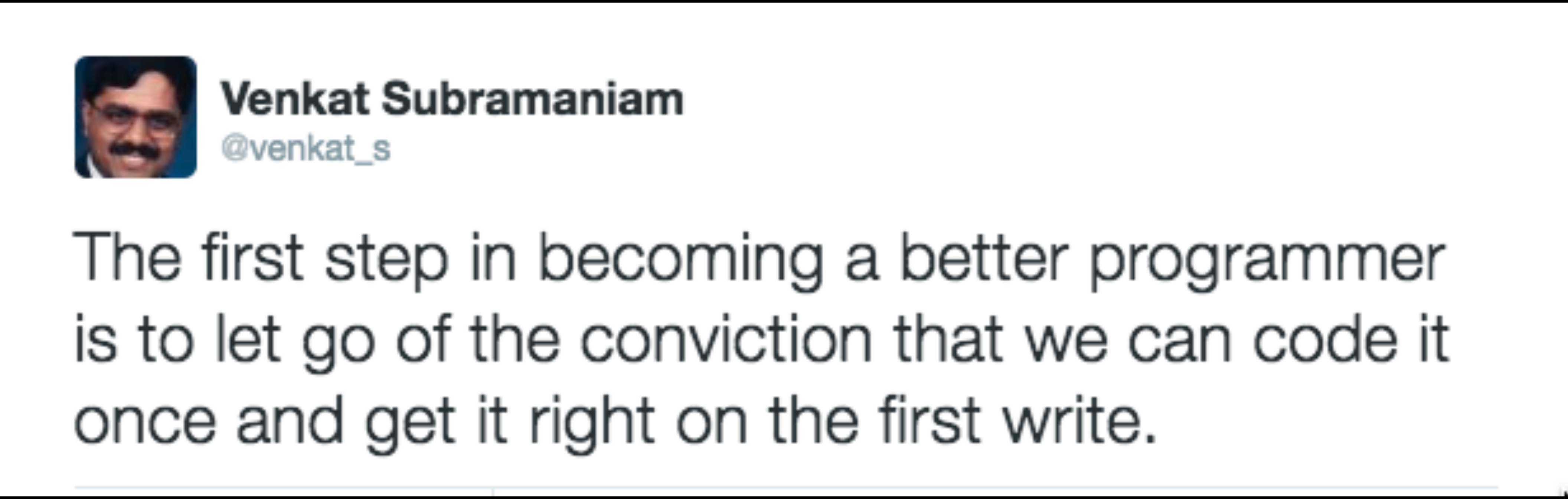
Messing with state is the root of many problems, both in software and in politics.

Venkat Subramaniam
@venkat_s

Mutability needs company, it often hangs around with bugs.

Twelve Ways to Make Code Suck Less

12. Schedule Time to Lower Technical Debt
11. Favor High Cohesion
10. Favor Loose Coupling
9. Program with Intention
8. Avoid Primitive Obsession
7. Prefer Clear Code over Clever Code
6. Apply Zinsser's Principle on Writing
5. Comment Why, not What
4. Avoid Long Methods—Apply SLAP
3. Give Good Meaningful Names
2. Do Tactical Code Reviews
1. Reduce State & State Mutation



Venkat Subramaniam
@venkat_s

The first step in becoming a better programmer is to let go of the conviction that we can code it once and get it right on the first write.

http://agiledeveloper.com/downloads.html

Thank you

http://agiledeveloper.com/downloads.html

venkats@agiledeveloper.com
@venkat_s