The Clean Coder

Chapter 4: Coding

About the Author

Robert Cecil Martin

- Co-Author of the Agile Manifesto
- Author and Software Engineer
- Published Four Books on Software Engineering
- Significant Industry Experience
- Nicknamed “Uncle Bob”

Coding – Chapter Overview

Recommended rules and principles of coding

They do not pertain to the code itself

About behavior mood and attitude when writing code

“They are not intended to be absolute truths for anyone other than me.”

Key to mastering any discipline are: confidence and error-sense

Preparedness

- Coding is an intellectually challenging and exhausting activity
- There are many factors we as coders must juggle
  - Code must work!
  - Your code must solve the customer's problem
    - Balancing wants and needs
  - Code must fit well into the existing system
  - Code must maintain readability

Distracted Coding

3AM Code

- Dangers of tired coding
- Feeling “productive” and “dedicated”
- Wrong solutions
- Coding while distracted creates waste
- Dedication and professionalism are more about discipline than hours

Worry Code

We all have personal problems

At work, these are background worries

They do not mix well with coding

Solve them, then code

… or at least mitigate them (allocate some time to diminish them)

Flow State

Hyper productive, highly focused, tunnel-vision state of consciousness

Also called The Zone

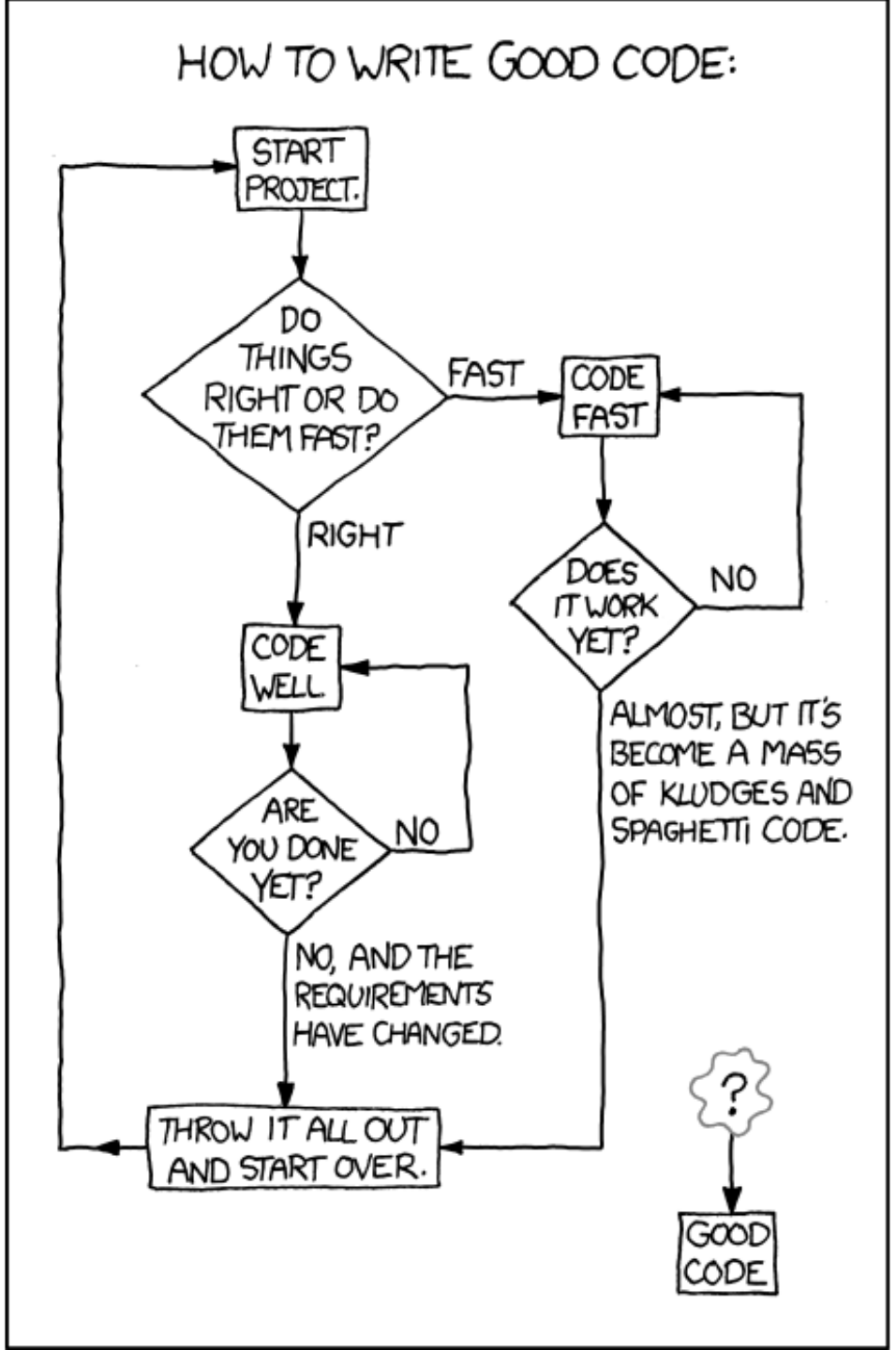
Programmers in “the zone” feel productive and infallible

Avoiding The Zone

“Mild meditative state in which rational faculties are diminished in favor of a sense of speed”

You lose much of the big picture in The Zone

You’ll likely reverse some decisions



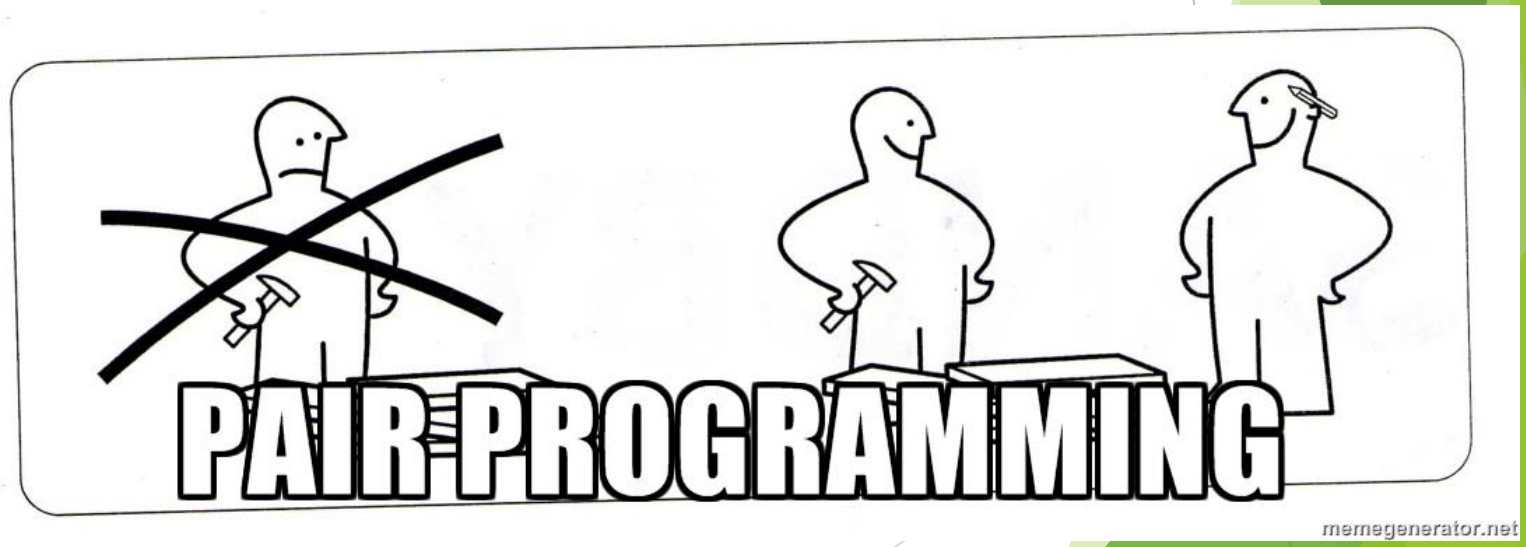
The Zone - continued

How to avoid

- Pair programming!
- Zone is non-communicative
- Pair Programming = Super communicative

When to be in the zone?

- When practicing



Music

Music is a mixed bag

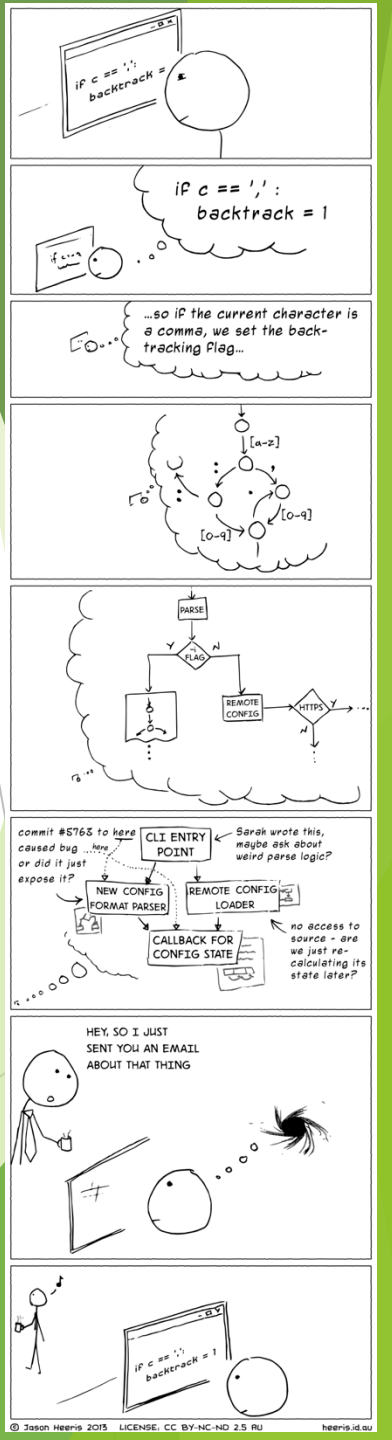
Good for some but maybe bad for others

- Understand if it helps or hinders you

Author is suspicious that music just helps people enter the zone…

Interruptions

- What do you do when interrupted while coding?
- Rude responses often come from the Zone
  - Resentment from being dragged out
- Interruptions of intense concentration
  - How to help mitigate the damage
    - Pair Programming (yep again)
    - Test Driven Development
    - Acceptance
- Professional attitude is a polite willingness to be helpful
  - Recall you might need to interrupt someone too

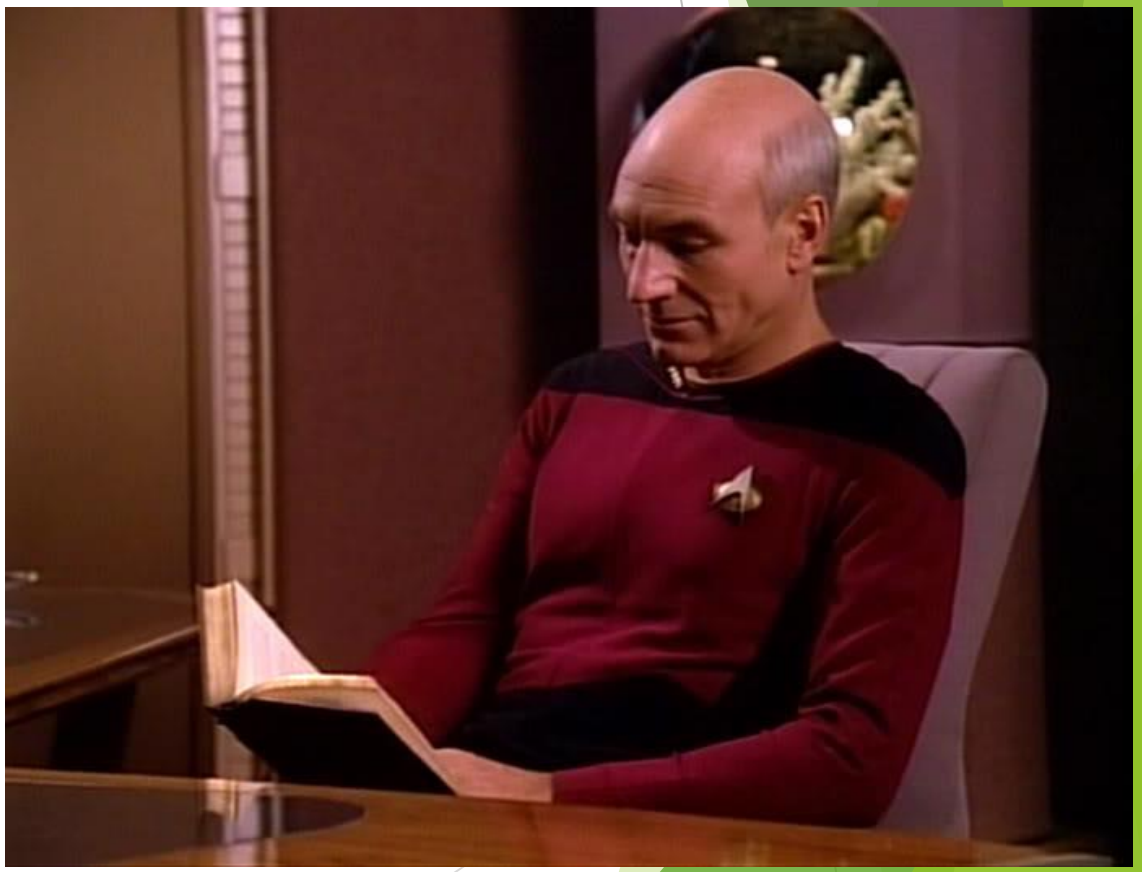


Writer's Block

- Procrastination
- Many things lead to writer's block
  - Fear
  - Anxiety
  - Depression
  - Lack of sleep
- How to resolve
  - PAIR PROGRAMMING (again!)
  - Physiological change from human interaction

Creative Input

- A vaccine against Writer's Block
- Creative output depends on creative input
- Read, Read, Read
  - Any and all topics
- Spend time away from problems
  - Actively DON'T think about them
- Feel stimulated to create when engaging with a creative work
  - Positive element of escapism



Debugging Time

Time Spent Debugging and Time Spent Coding

- In developers minds: Coding != Debugging
- In employers minds: They Cost the Same Thing

So we should work to mitigate time spent debugging

How: Test Driven Development!

Bugs and Professional Behavior

- Creating many bugs is unprofessional (consider doctors or lawyers fixing errors repeatedly!)

Pacing Yourself

- Software Development: A Marathon
  - Win by conserving resources and pacing yourself
  - Take care of yourself before, after and during “the race”
- When to Walk away
  - When you are stuck or tired, disengage for a while
    - “Give your creative subconscious a crack at the problem”
- The power of showers!
  - No distractions: gives your mind breathing room

Being Late

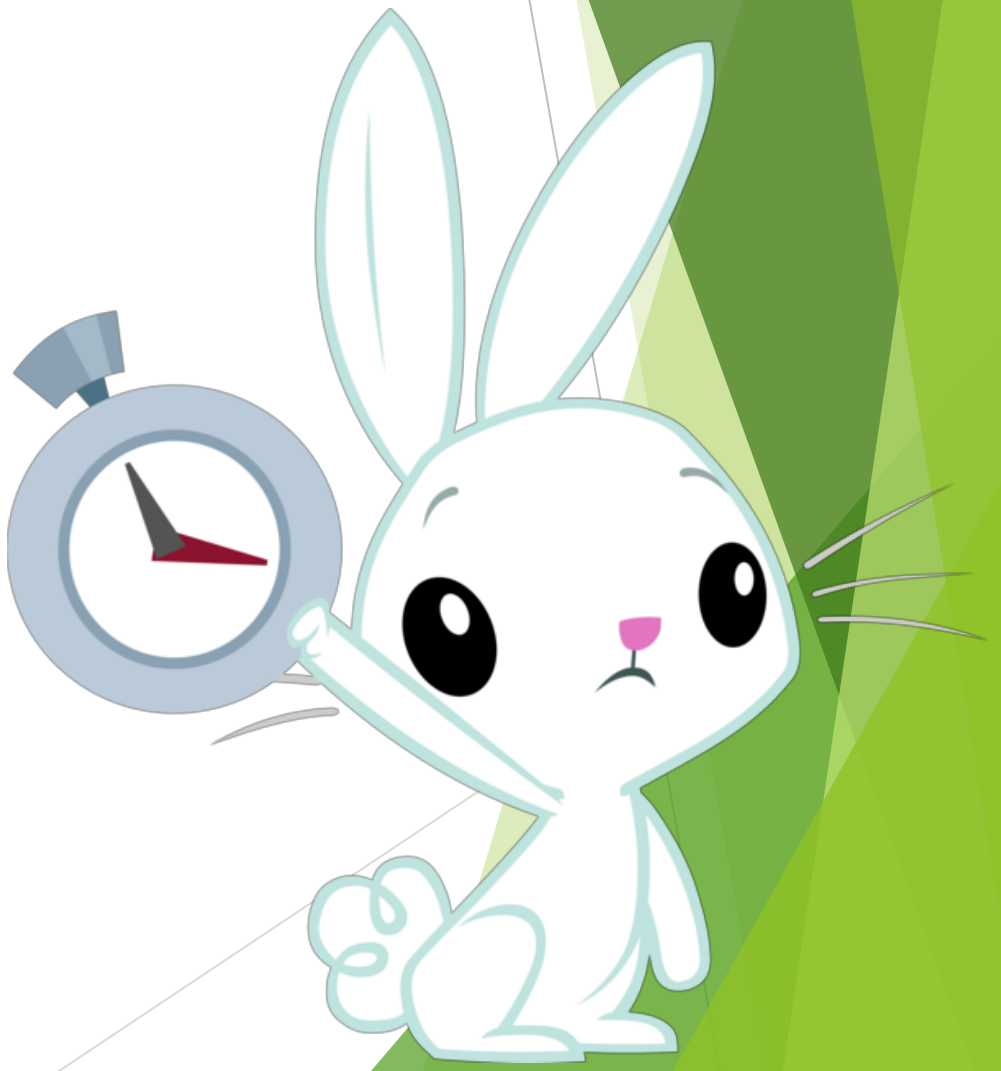
You WILL be late

But that’s OK

The important thing is *early detection* and *transparency*

Don’t:

- Tell everyone you will be on time until the very end and then let everyone down



Being Late – Do's

- Regularly measure progress against a goal
- Come up with three fact based dates:
  - Best Case, Nominal Case, Worst Case
- Be as honest as you can about these dates
  - Even to yourself
- Never incorporate hope into your estimates!

Hope

"Hope destroys schedules and ruins reputations"

What do you do if numbers show that you *might* miss a deadline?

Example: Trade Show in 10 days

- Your estimates (best, nominal, and worse cases): 8/12/20
- Do not **hope** to get it all done in ten days!
- Assume the correctness of your nominal estimate and plan accordingly
- Make sure everyone understands the situation
- Ensure there is a backup plan
- Do not inadvertently spread hope

Rushing

Oh but wait!

Your manager has a “great” idea

He/she says “Do what it takes to make the deadline!”

How do you respond?

- Say No
- Hold to your estimates
- Do not be tempted to rush
- Tell your boss you have considered the options

There is no effective way to rush!

Overtime

But wait! Your boss has more ideas!

“What if you work 2 more hours a day or this weekend!?”

Overtime CAN work but it’s risky

- 20% more hours != 20% more work done
- And, if it goes on for more than two or three weeks: It WILL fail

Overtime – How it works

Do not agree to work overtime unless

- You can personally afford it
- It is short term
- Your boss has a fall-back plan in case overtime effort fails

Number 3 is especially important

- If your boss cannot articulate to you what he/she is going to do if overtime fails, then do not agree to work overtime

Unprofessional Behavior

False Delivery

- Saying you are done when you are not
- Rationalizing new definitions of “done”
- Contagious practice
  - Programmers will collectively adopt and stretch creative definitions of “done”
  - Eventually int main() {} becomes “Almost done”

Define “Done”

- Have an objective, independent definition of done
- “When the code passes automated acceptance tests”

Help

Programming is hard

So help is very important

Helping others

- Professional responsibility to make yourself available to others
- A violation of professional ethics to sequester yourself in an office

Of course, you are allowed alone time

- But be polite and open about when you don't wish to be disturbed

Offer help!

When you help sit down and program with them

- Be fully engaged in helping!

Help [cont'd]

- Being helped
  - Be gracious about help offered
  - Accept help gratefully
  - Don't be territorial about your code
  - Honor bound to accept help
- Even if you are under pressure accept help
  - Give it 30 mins or so
- Learn how to ask for help
  - It's unprofessional to remain stuck when help is easily accessible

Help [cont'd]

Programmers tend to be self-absorbed introverts

They like to deal with multiple concepts and complexities, take care of minutia, and prove they are able to find great solutions

…

… while not dealing with the complexities of other people

[Of course, there are many exceptions]

Yet collaboration is critical to effective programming

So, need to rely on disciplines that drive to collaboration

Mentoring

Responsibility of Experienced Programmers

Effective mentoring teaches concepts much, much better than reading or online tutorials

Younger Programmers

Have a professional duty to seek out mentoring

Also, it's generally a good idea to get to know people in your office