Navigating The Technical Interview Maze

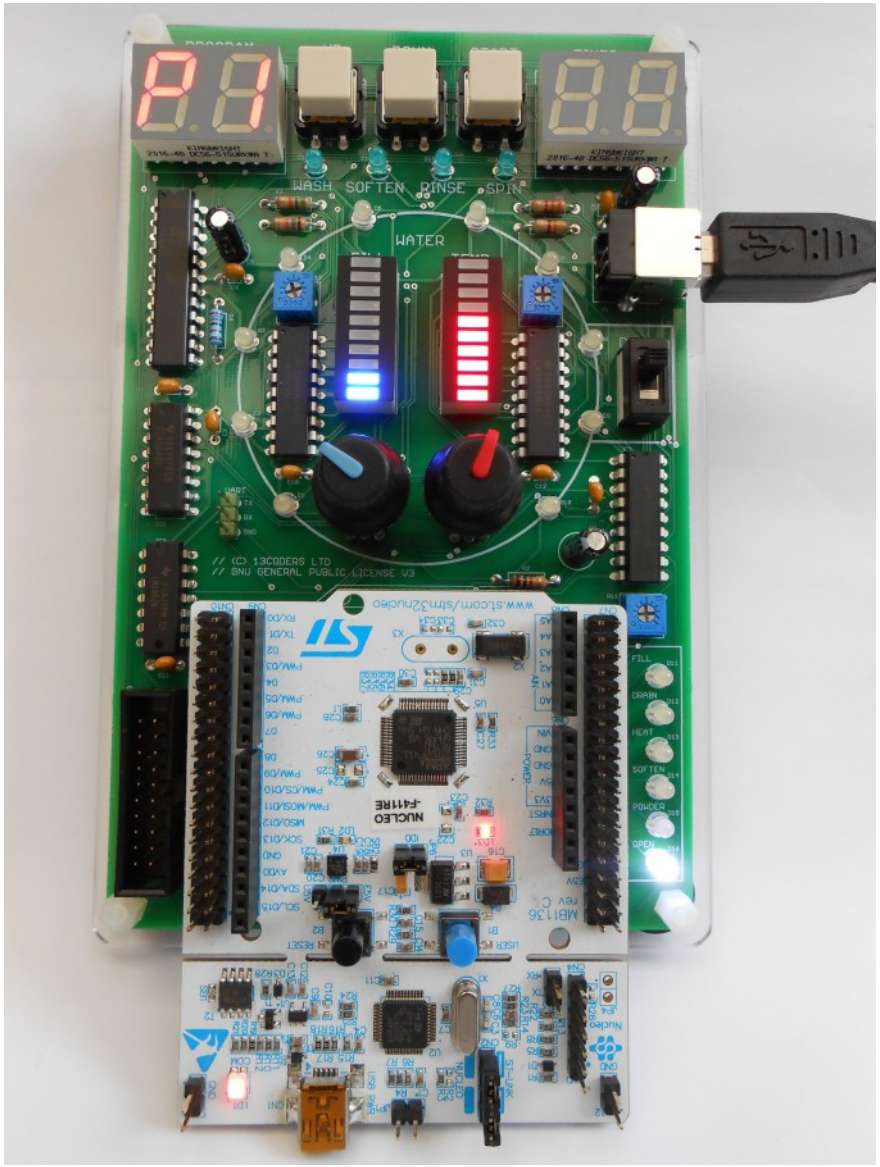
How the whole technical hiring thing works - companies, careers, technical interview formats

Mike Ritchie
@13coders

The obligatory mini-biography

- 30+ years working in software development 😱
- Lots of different industry sectors and technologies
- Companies ranging in size from tiny to hugeous
- Done some non-hands-on-dev things too:
  - engineering management
  - building teams
  - coaching graduate entry software developers
- I do a mix of development and training now
- Oh, and I build bizarre hardware too…

The type of stuff I work on



I mostly work on small embedded devices, and build custom hardware to help embedded engineers do test-driven development and continuous integration.

This design is the "Washing Machine", a physical code kata for learning and practicing embedded TDD.

News Kata About Pricing Students Privacy

Hey there

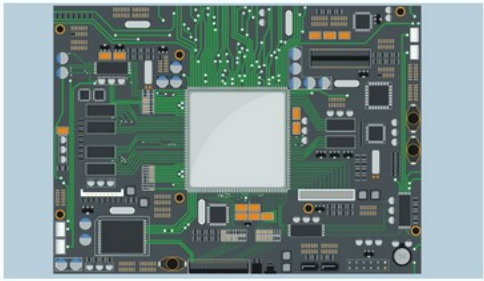
If your organisation is working on embedded systems development, we can help you. We can boost the TDD and design skills in your development team, give a fresh perspective on a project, or help you build something new.

☎ +44 7808 480387 / ✉ info@13coders.com

Learn more about us »

13coders

Development »



We develop embedded real-time systems for the ARM Cortex-M family of microcontrollers. We follow a disciplined, test-driven approach that leverages the best tooling from the proprietary and open source domains.

Continuous Delivery is the default mode for us: every commit is a safe, stable, secure and shippable product, subjected to battery of automated tests, dynamic and static analysis, fuzz testing, and performance measures...learn more »

Training Workshops »

We offer a 2-day training workshop for embedded systems developers covering TDD and XP practices on a unique hardware platform, designed specifically for embedded systems "deliberate practice".

We're happy to customise the content, or to add additional days to focus on the specifics your team needs - design and TDD for RTOS systems, CI/CD builds, or behaviour-driven development. Just tell us what you need ...learn more »

Consultancy & Review »

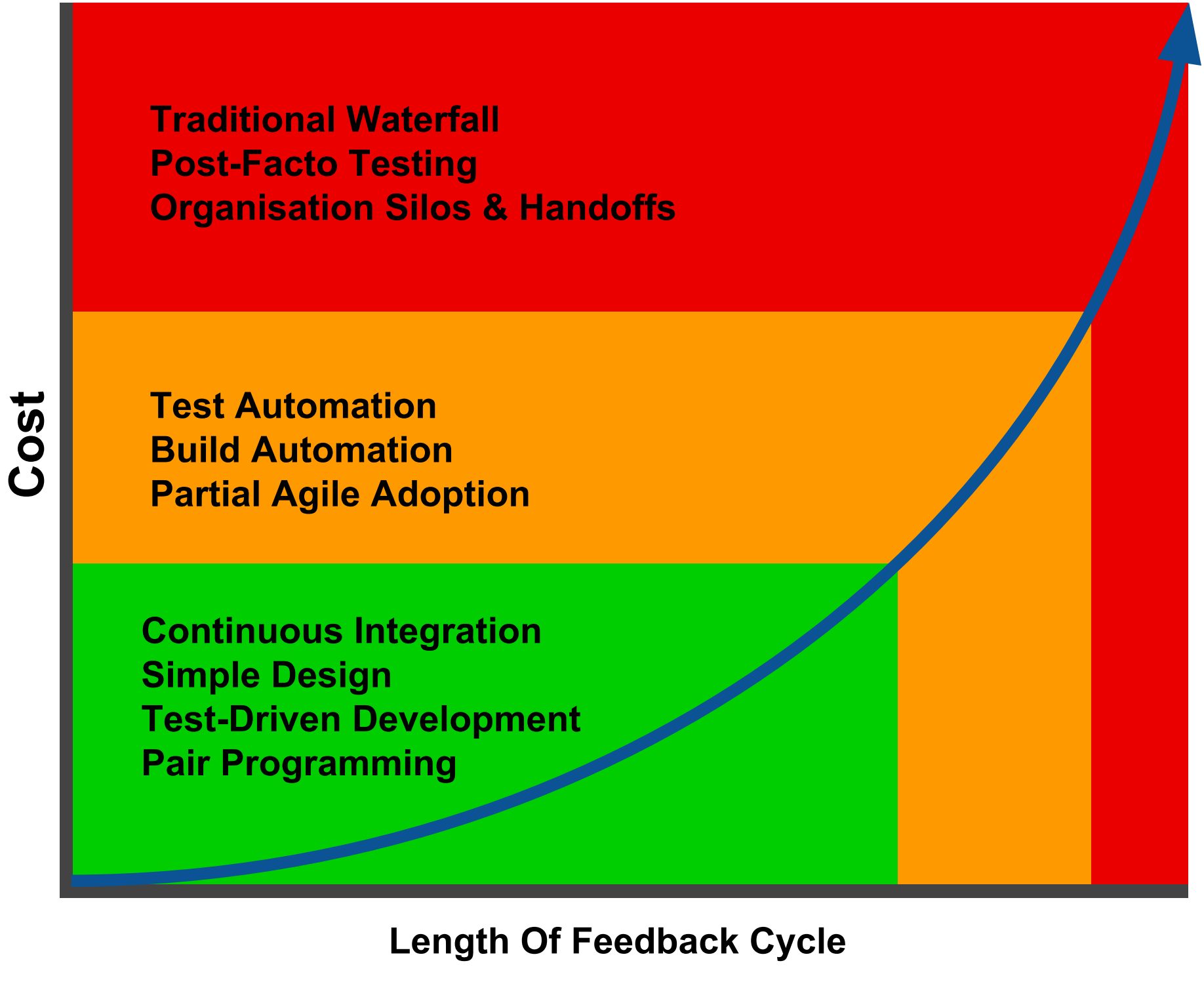
Sometimes you need a fresh pair of eyes on your methods, code and team setup. Whether this is to diagnose a problem, or to make your development effort more effective, we'll give insight and practical recommendations.

We can do targeted code reviews for specific issues, or broader reviews across your codebase, backed up with measurements and visualisation....learn more »

https://www.13coders.com/

What's in store?

- Me saying this stuff I'm saying right now
- Whiteboard exercise (groups, 45m + retro)
- Hands-on coding challenge (pairs, 45m + retro)
- Technical Q&A exercise (pairs, 45m + retro)
- And any questions, any time
- Treat this as an "ask me anything" format


Traditional Waterfall
Post-Facto Testing
Organisation Silos & Handoffs
Test Automation
Build Automation
Partial Agile Adoption
Continuous Integration
Simple Design
Test-Driven Development
Pair Programming
Cost
Length Of Feedback Cycle

All too often, programmers are divided into average programmers and rockstar or ninja programmers. Where there's a rockstar, there's a trashed codebase with broken classes and spaced-out control flow. Where there's a ninja, there's mysterious bugs and build problems that occur in the middle of the night.

Where there's an average, there's a distribution. In the long term, what matters is less **where on the distribution someone is** than where they are **headed.**

If you want to divide programmers into two groups, there are programmers who get better and programmers who don't. **You care about the first group.**

Kevlin Henney, in preface to "Becoming a Better Programmer" by Pete Goodliffe.

守破離

Shu or "obey"

Learn traditional wisdom, learning fundamentals, techniques and heuristics.

Ha or "digress"

Breaking with tradition.

Ri or "separate"

Transcendence, no techniques, all moves are natural.

The twin keys to your early progress: (a) your innate drive to learn and improve, (b) your coachability by experienced engineers.

Source: https://en.wikipedia.org/wiki/Shuhari

Your value-add and breadth

Business Product TDD Analysis Security

UX Testing

Your core in-depth skills

Design

Language

Frameworks

Tools

Automation

Platforms

Performance

Low-level

Debugging

...be a T-shaped techie

Questions?

“The hiring funnel”

CV / application screen

↓

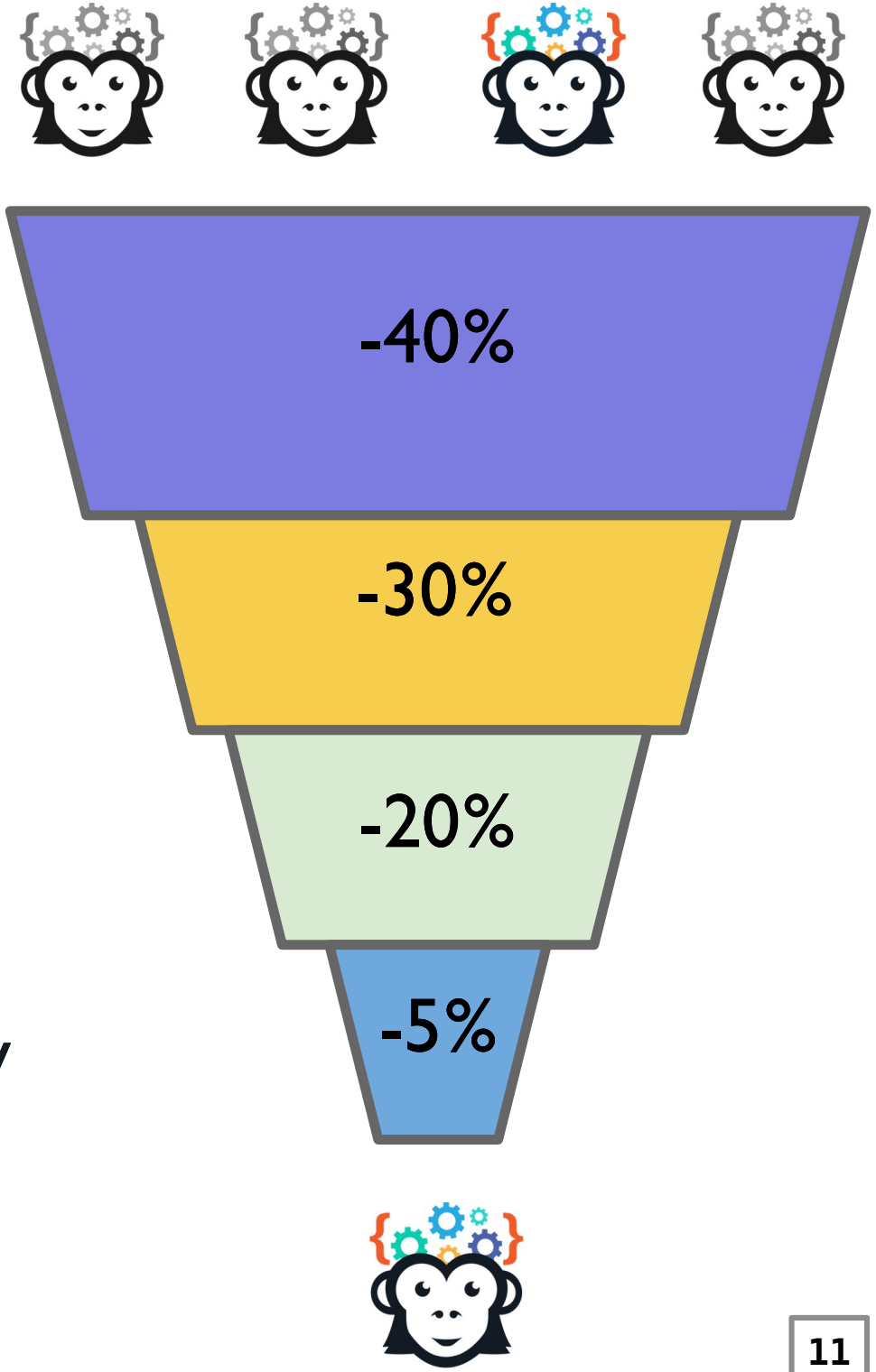
Phone screen or tech test

↓

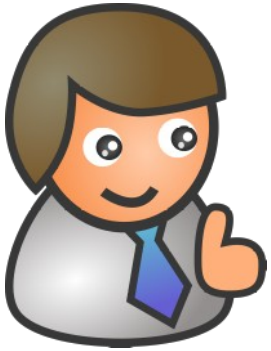
Face-to-face tech interview

↓

Competency-based interview



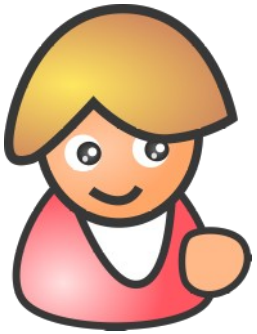
The cast of characters



Reginald the recruiter.

You, the amazing candidate.

Henry the hiring manager.



Theresa the tech interviewer.

What makes interviewers happy

Words & actions match

You can do what you say you can do.

Clear, authentic narrative

Your story is credible, consistent, and real.

You're in the driving seat

You're in charge of your own development.

Positivity

You're upbeat and engaged in the process.

...and unhappy

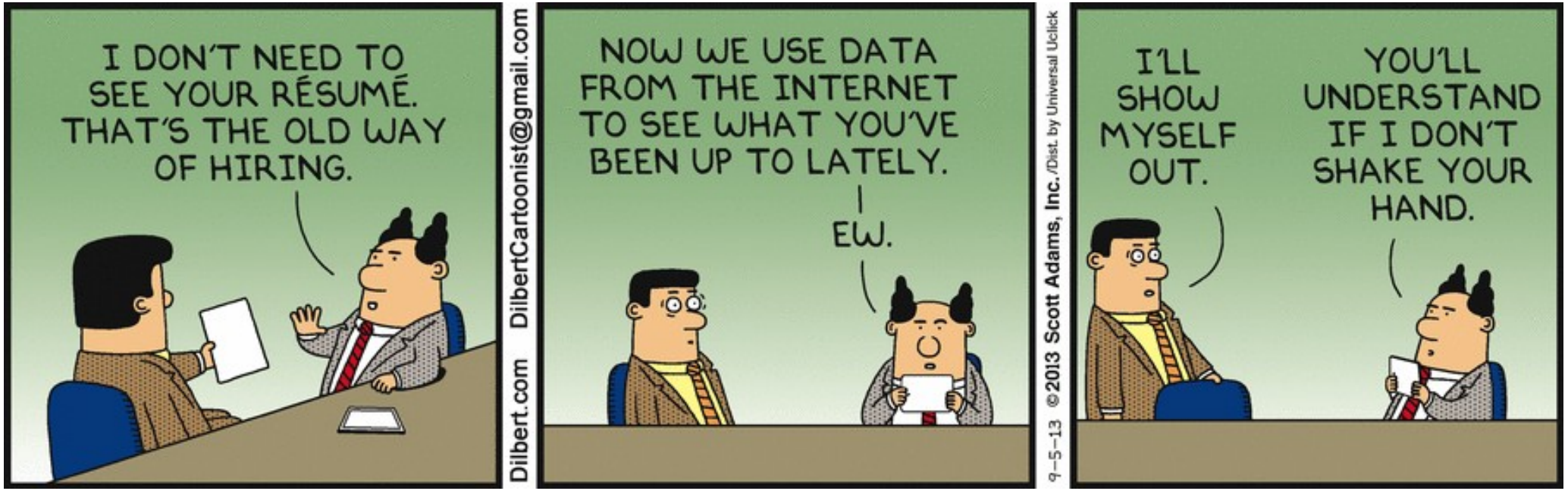
Disinterest and disengagement

Kills 99.9% of all job applications.

Stone.

Dead.

CVs, applications, cover letters



Some advice on CVs

- Assert your developer-ness up front
- Tailor your experience for your audience
- Weave a narrative thread if possible
- Avoid huge lists of frameworks and tools
- Pick your strongest skills and showcase those
- Remember the basics:
  - A couple of pages, reverse chronological order
  - Aim for error-free : get it reviewed

Search GitHub

Pull requests Issues Marketplace Explore

Mike Ritchie

klutz

I work **@13coders**, mainly on C++ for embedded. I also design and build hardware, and sometimes it works for a little while.

@13coders

Edinburgh, United Kingdom

http://www.13coders.com/

Organizations

Overview Repositories 22 Stars 167 Followers 15 Following 4

Pinned repositories

Customize your pinned repositories

13coders/**cookiecutter-kata-gtest**

A cookiecutter template for creating a simple C++ TDD code kata project using with Google Test / Google Mock

C++ 2 1

13coders/**embedded-eagle-libraries**

Reusable libraries for EagleCad

13coders/**embedded-systems-kata**

The "Washing Machine" kata for embedded systems development

Eagle 8

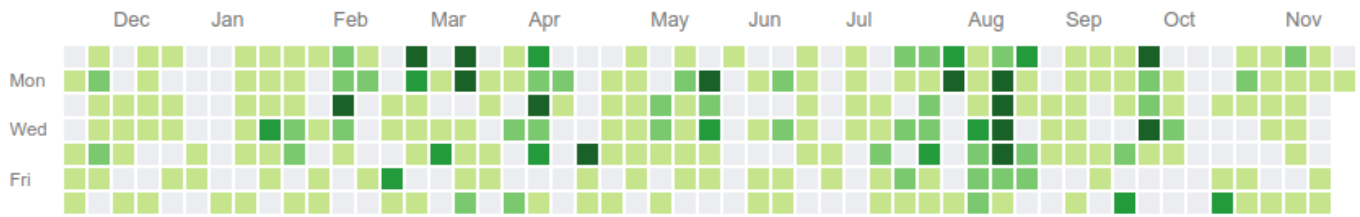
13coders/**cookiecutter-kata-catch**

A cookiecutter template for deliberate TDD practice in C++ with Catch! and Trompeloeil

C++ 1

1,311 contributions in the last year

Contribution settings



Learn how we count contributions.

Less More

Search GitHub

Pull requests Issues Marketplace Explore

Mike Ritchie
klutz

klutz (noun) : a clumsy, awkward or foolish person. Someone who falls down a lot.

I work **@13coders**, mainly on C++ for embedded. I also design and build hardware, and sometimes it works for a little while.

@13coders
Edinburgh, United Kingdom
http://www.13coders.com/

Organizations

Overview Repositories 22 Stars 167 Followers 15 Following 4

Pinned repositories

Customize your pinned repositories

13coders/**cookiecutter-kata-gtest**

development

Eagle 8

13coders/**embedded-eagle-libraries**

sable libraries for EagleCad

3coders/**cookiecutter-kata-catch**

okiecutter template for deliberate TDD practice in C++ with Catch! and Trompeloeil

C++ 1

1,311 contributions in the last year

Contribution settings

Dec Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov

Mon Wed Fri

Learn how we count contributions.

Less More

Search GitHub

Pull requests Issues Marketplace Explore

Mike Ritchie
klutz

I work **@13coders**, mainly on C++ for embedded. I also design and build hardware, and sometimes it works for a little while.

@13coders
Edinburgh, United Kingdom
http://www.13coders.com/

Organizations

Overview Repositories 22 Stars 167 Followers 15 Following 4

Pinned repositories

Customize your pinned repositories

13co
A cookie
kata proj
C++

ed-eagle-libraries
gleCad

13co
The "Washing Machine" kata for embedded systems development
Eagle 8

utter-kata-catch
A cookiecutter template for deliberate TDD practice in C++ with Catch! and Trompeloeil
C++ 1

Try to populate this as much as you can – it gives your profile a more "looked after" feel.

1,311 contributions in the last year

Contribution settings

Dec Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov

Mon
Wed
Fri

Learn how we count contributions.

Less More

Search GitHub

Pull requests Issues Marketplace Explore

Mike Ritchie

klutz

I work **@13coders**, mainly on C++ for embedded. I also design and build hardware, and sometimes it works for a little while.

@13coders

Edinburgh, United Kingdom

http://www.13coders.com/

Organizations

Overview Repositories 22 Stars 167 Followers 15 Following 4

Pinned repositories

Customize your pinned repositories

13coders/**cookiecutter-kata-gtest**

A cookiecutter template for creating a simple C++ TDD code kata project using with Google Test / Google Mock

C++ 2 1

13coders/**embedded-eagle-libraries**

Reusable libraries for EagleCad

13coders/**embedded-systems-kata**

The "Washing Machine" kata for embedded systems development

Eagle 8

13coders/**cookiecutter-kata-catch**

A cookiecutter template for deliberate TDD practice in C++ with Catch! and Trompeloeil

C++ 1

1,311 contr

ribution settings

Mon

Wed

Fri

Nov

Learn how we count contributions.

Less More

Customise the repositories that you want to showcase by pinning them to your profile front page.

9 commits | 1 branch | 0 releases | 1 contributor | GPL-3.0

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

klutz Correspond | Latest commit 1fc0cda on 24 Aug

{{ cookiecutter.k	3 months ago
.gitignore	3 months ago
LICENSE	3 months ago
README.md	3 months ago
cookiecutter.jsor	3 months ago

Always have a **README.MD** : make your work easy to navigate and understand. The easier it is to understand, the more time a tech interviewer will spend looking at it.

README.md

Template for TDD Code Katas in C++

Hey there. This is a cookiecutter template for a simple TDD code kata using Catch! and Trompeloeil. It's intended to give you a repeatable way of very quickly getting started for a "deliberate practice" session with C++.

Features of this template

This generates a project for doing a test-driven code kata in C++.

- Includes Catch! and Trompeloeil libraries
- Generates a header, "production" source file and an empty test
- Generates a CMake build which will work on most platforms
- Has some convenience targets for generating etags and running unit tests

Questions?

Telephone screens

Phone screens : knowledge-based

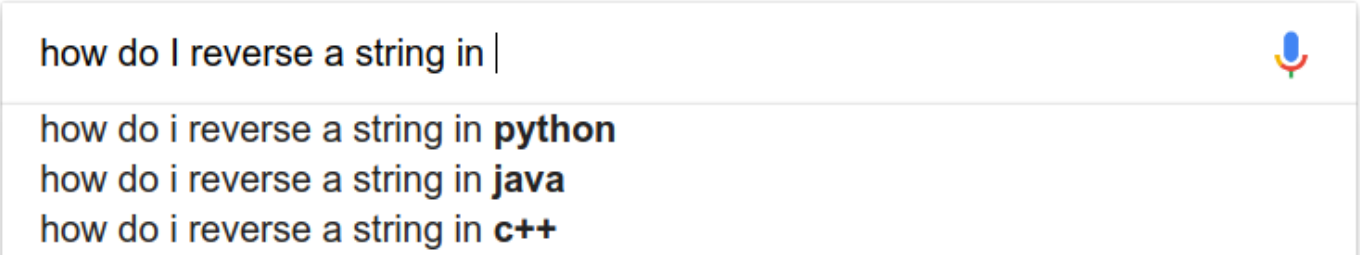
- Short, fairly specific questions:
  - Language, core API
  - CS concepts (algos, data structures)
  - Tools and 3rd party frameworks
  - Virtually anything relating to your skills...

Phone screens : experience-based

- Usually based on a reading of your CV:
  - "Tell me what you did on <project>..."
  - "It says here you used <framework>...how did you find that ?"
  - "Tell me how you switched to <tool>..."
- Sometimes these questions are used as a launch-off point into more technical questions

Phone screen approach

- Be friendly...remember it's a filter
- Keep your answers concise
- Ask for clarification if you need it
- Say when you don't know
- Don't do this →




I ARE PROGRAMMER
I MAKE COMPUTER BEEP BOOP
BEEP BEEP BOOP

Tech interview formats to expect

- Technical Q&A 😐
- Pair programming / code katas 😍
- Whiteboard design exercise 😐
- Algorithm and data structure problems 🤔
- “Fix this code” bug-hunts 😟
- Online code tests 🤔
- Homework projects 😀

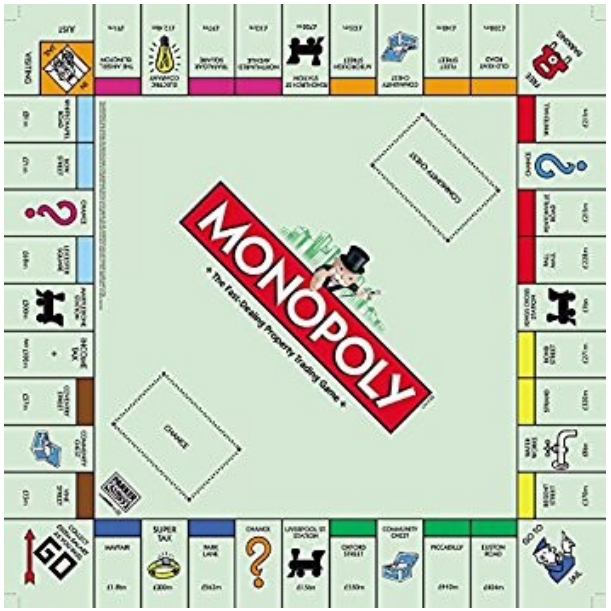
Coping strategies for nerves

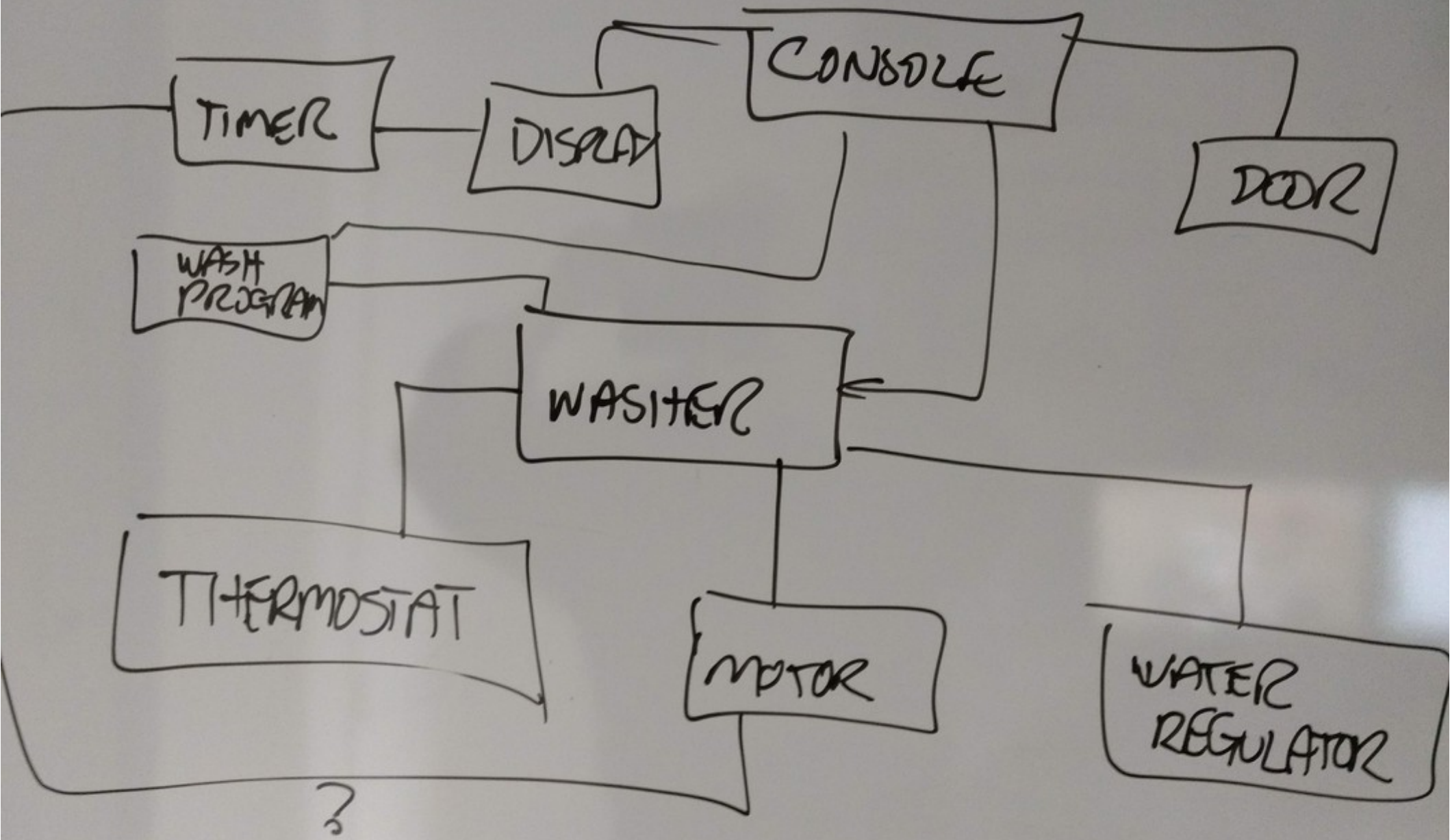
- Focus on the problem, not the interview
- Think out loud
- Sketch out ideas
- Make a start, but don't rush
- Discuss it with the interviewer
- Completion and perfection are not expected

Questions?

Whiteboard!

- You're given a problem, and you sketch out a solution
- Boxes and lines
  - ...but don't get hung up on UML notation
- Done right, it's all about the exploration
- Our challenge:
  - The "monopoly" game
  - Design a solution
  - Work in small groups
  - Rotate the pen!




CONSOLE
TIMER
DISPLAY
DOOR
WASH PROGRAM
WASHER
THERMOSTAT
MOTOR
WATER REGULATOR
?

Some whiteboarding tips

- Think "classes, responsibilities, collaborators"
  - What it does more than what it has
- Avoid fixating on implementation
- Easy wins - the physical parts of the game
- Things that happen as well as things that exist
- Broad sketch first, but then focus on something
- Take turns at the board, pass the pen around

Retrospective

Data structures and algorithms

- Greatly loved by Google amazon
  - copied by many others (rightly or wrongly)
- They're a fair approach for some roles
- Sometimes the wrong emphasis for others
- Might be done on a board, or in code
- Increasingly set as an online challenge

CODING HORROR
programming and human factors

26 Feb 2007

Why Can't Programmers.. Program?

I was incredulous when I read this observation from Reginald Braithwaite:

> Like me, the author is having trouble with the fact that 199 out of 200 applicants for every programming job can't write code at all. I repeat: *they can't write any code whatsoever*.

Online tests

codility

- Challenges set by the employer
- Tend to require "algorithmic thinking"
- Can be attempted in any language
- Basic, web-based IDE
- Timed countdown for maximum freak-out factor

Sign up as an individual and practice for free

Detected time complexity: O(N) or O(N * log(N))

expand all	Example tests	
▶ example1 first example test		✔ OK
▶ example2 second example test		✔ OK
▶ example3 third example test		✔ OK
expand all	Correctness tests	
▶ extreme_single a single element		✔ OK
▶ simple simple test		✔ OK
▶ extreme_min_max_value minimal and maximal values		✔ OK
▶ positive_only shuffled sequence of 0...100 and then 102...200		✔ OK
▶ negative_only shuffled sequence -100 ... -1		✔ OK
expand all	Performance tests	
▶ medium chaotic sequences length=10005 (with minus)		✔ OK
▶ large_1 chaotic + sequence 1, 2, ..., 40000 (without minus)		✔ OK
▶ large_2 shuffled sequence 1, 2, ..., 100000 (without minus)		✔ OK
▶ large_3 chaotic + many -1, 1, 2, 3 (with minus)		✔ OK

Efficiency is assessed as well as correctness.

Focus on getting it right first, then improving performance:

"If it's wrong, it doesn't matter how fast it is"

The system will run edge cases as well as increasingly large inputs to measure efficiency as the input size increases (time complexity).

Try thinking about the edge cases and adding custom tests for these.

Codility Online Coding Test

Write a function:

```
class Solution { public int solution(int[] A); }
```

that, given an array A of N integers, returns the smallest positive integer (greater than 0) that does not occur in A.

For example:

Given A = [1, 3, 6, 4, 1, 2], the function should return 5.
Given A = [1, 2, 3], the function should return 4.
Given A = [−1, −3], the function should return 1.

Google "codility demo test"

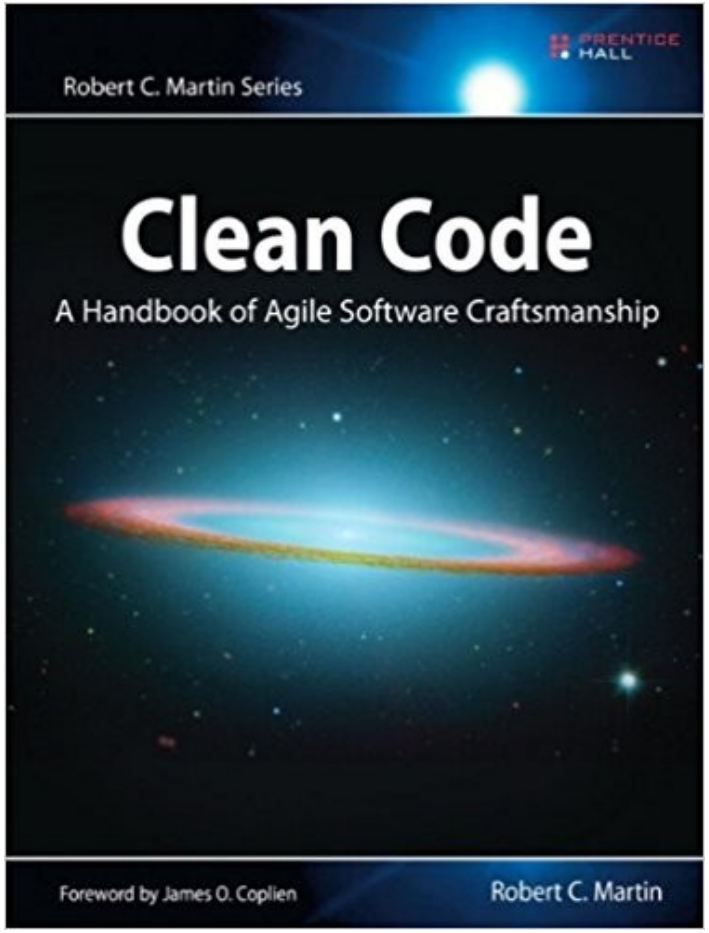
Retrospective

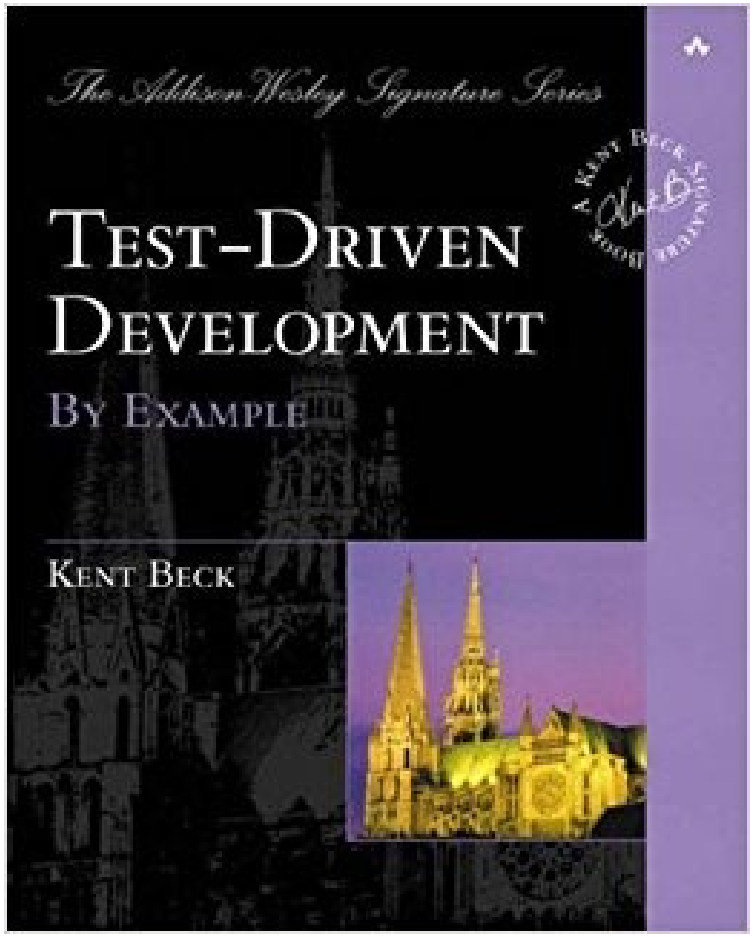
Technical Q&A

- Interviewers will often be working to a script...
  ...your challenge is to turn it into a conversation
- Can be on
  - language
  - libs
  - tools…anything
- You'll find some recurring themes, though
- Sometimes (not always) a "last resort" format
- Please Google for this exercise !

Retrospective

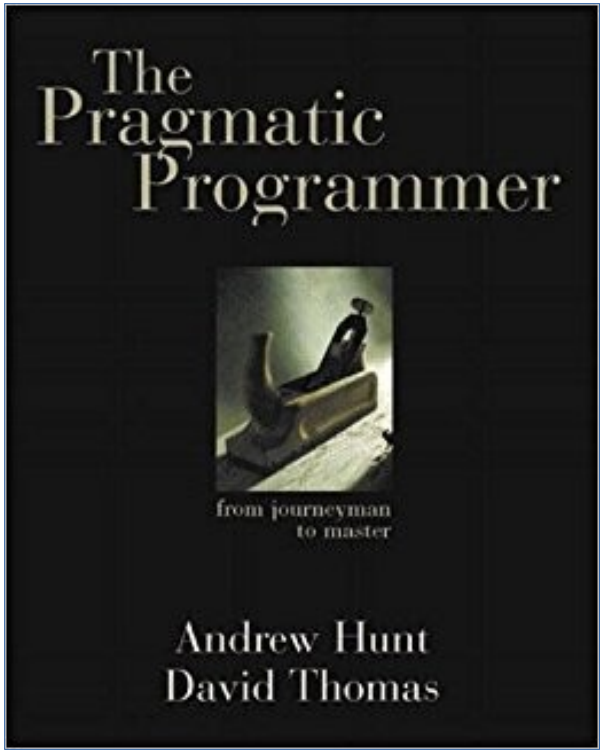
Books on clean code and TDD

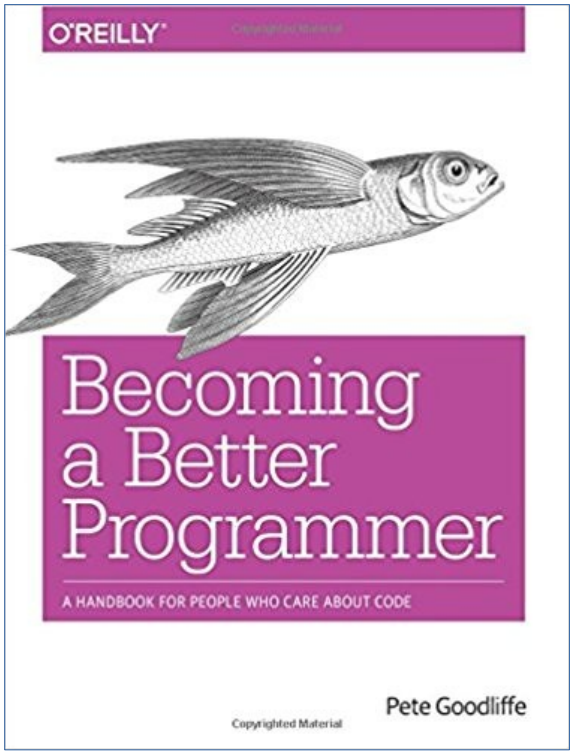


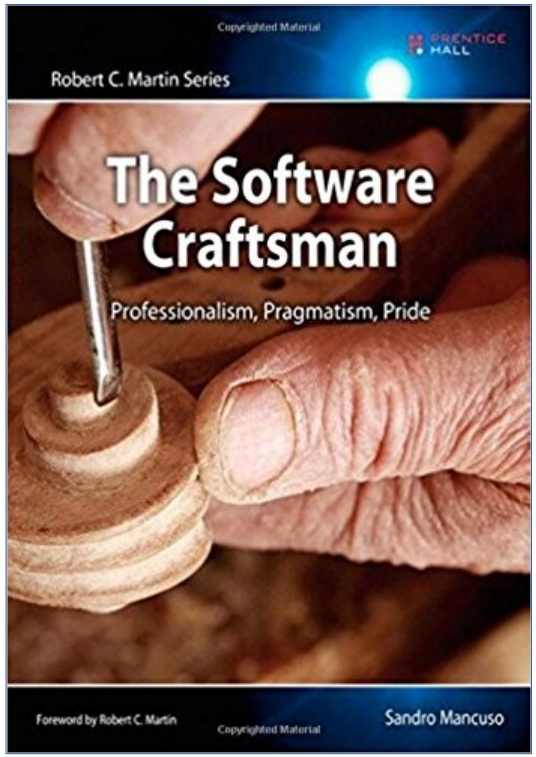


Both great books, in different ways.

Programming wisdom distilled

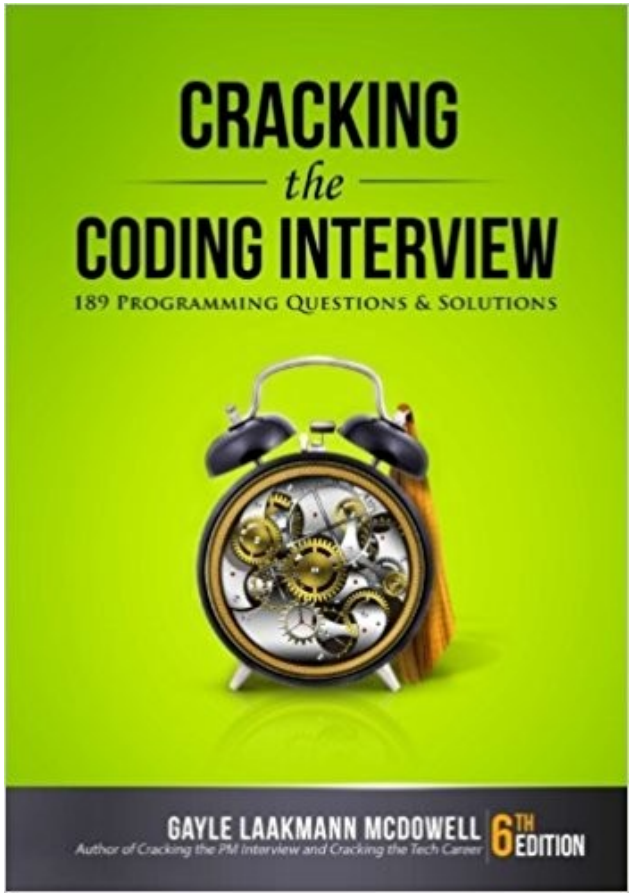


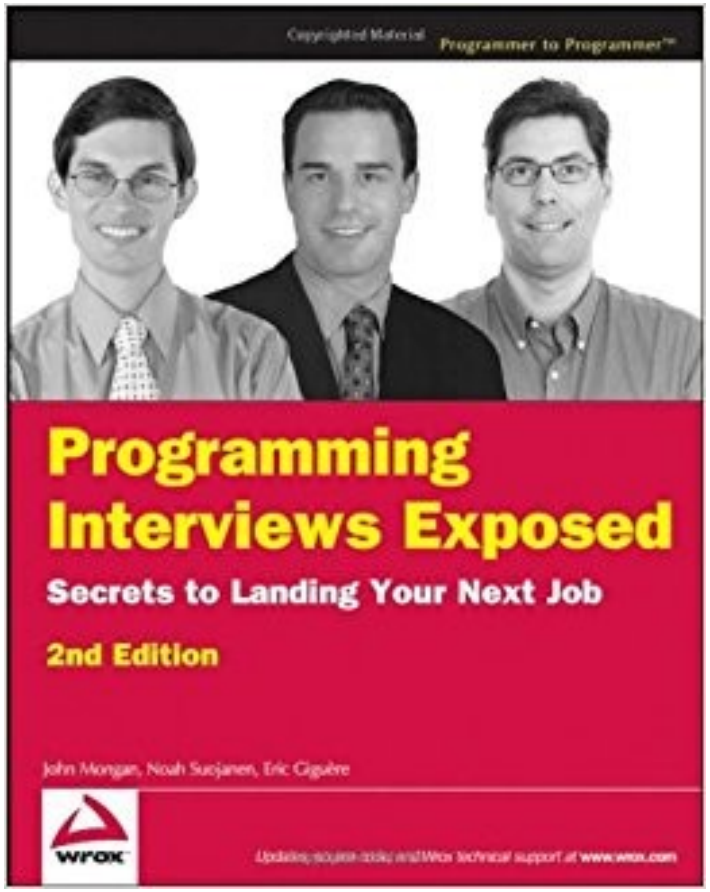




Books <u>about</u> programming, rather than "how to program". All excellent.

Books on getting hired



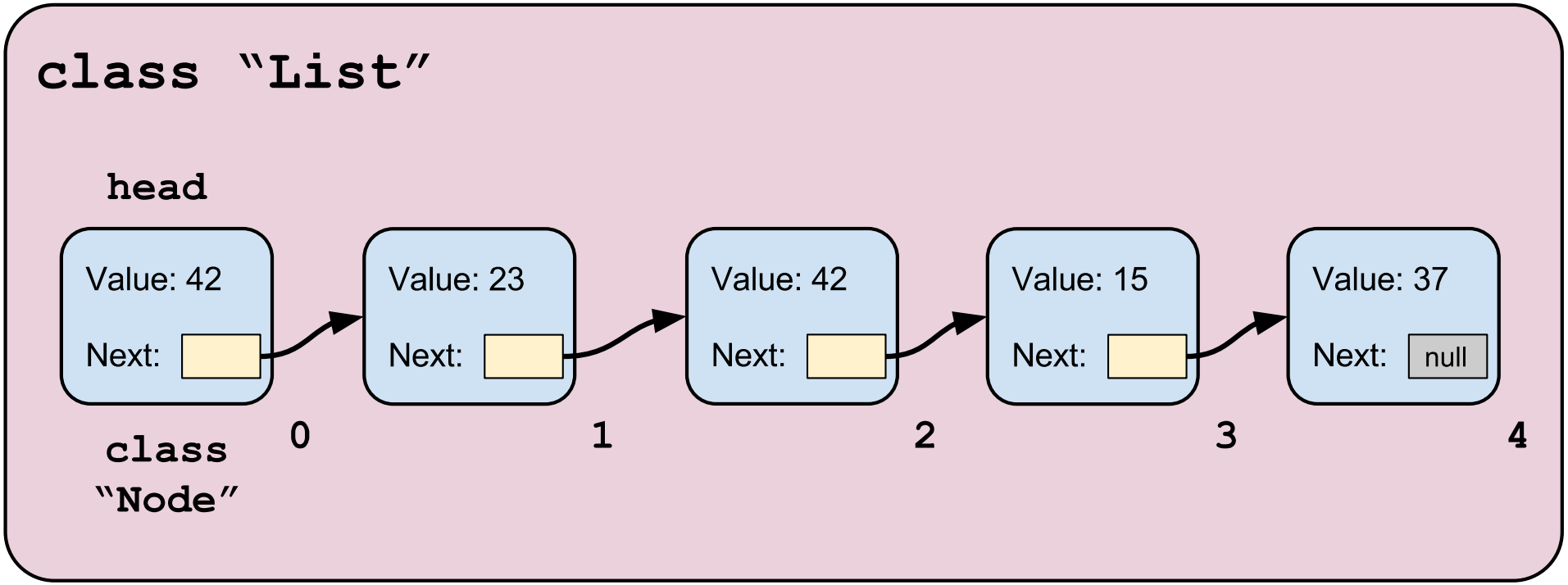


Gayle Laakmann McDowell's book is the go-to source for programming interview prep.

That's it !

Thanks for listening, stay in touch.

mike@13coders.com
https://www.linkedin.com/in/13coders/
@13coders




```
int getSize()
void insertFront(int value)
int getHead()
int countOf(int value)
void removeAt(int position)
void insertAfter(int position, int value)
```

Hands-on coding : preparation

- Make sure your laptop is configured, and tools working
  - The right editor/IDE
  - Pick one, stick to it, build familiarity
  - Also Git integration
- Practice setting up a really simple project structure
  - One /source folder, one /tests
  - Your test framework of choice
- Create a new project, add a test, build, run, commit

Hands-on coding : execution

- Git is your friend. Use git:
  - initialise a repo as soon as you have a skeleton
  - commit regularly at safe points
  - back out if you take a wrong turn
- Baby steps : get comfortable with TDD
- Write a test list:
  - reduces nerves
  - helps you focus
- Sketch, but don't over-sketch