Software Testing

Who am I?

Nikolas Vourlakis

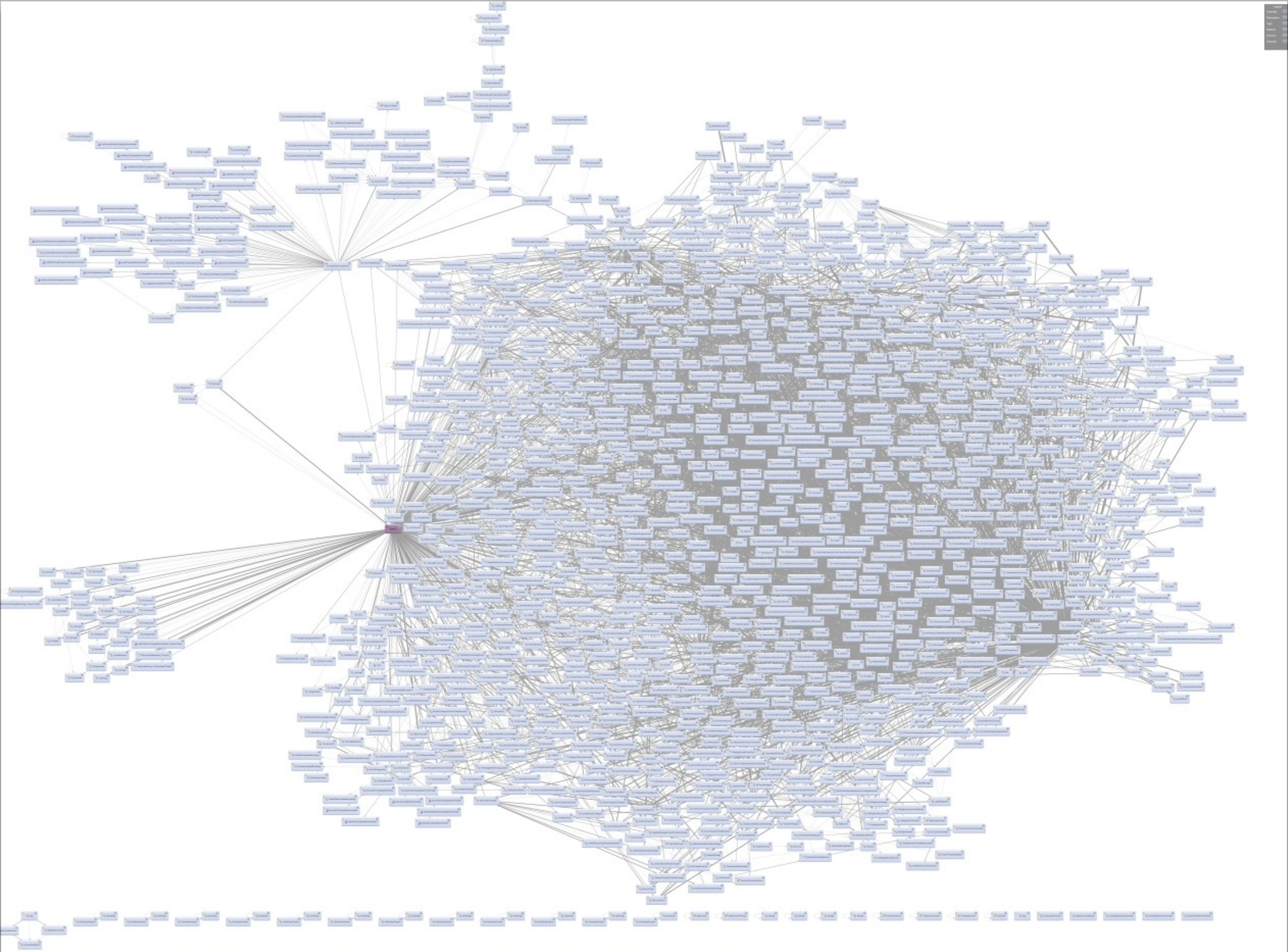
- Started with Ruby on Rails
- Now a backend developer @ imgZine on node.js
- And I like clean code

Software development is hard

Every line of code you write has ethical implications

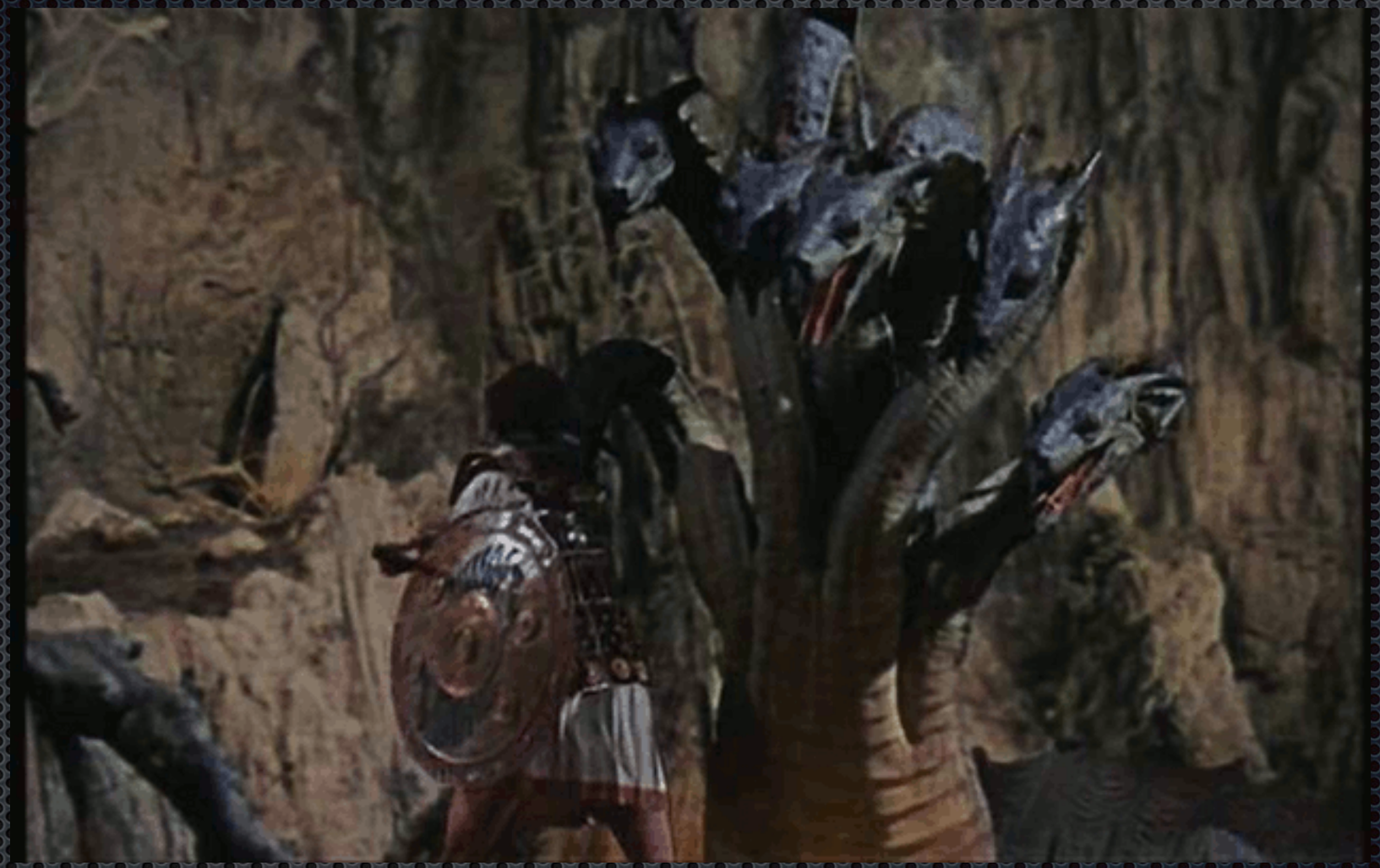
–Grady Booch

Just inherited a code base



May I refactor a bit?

Are you sure about this?



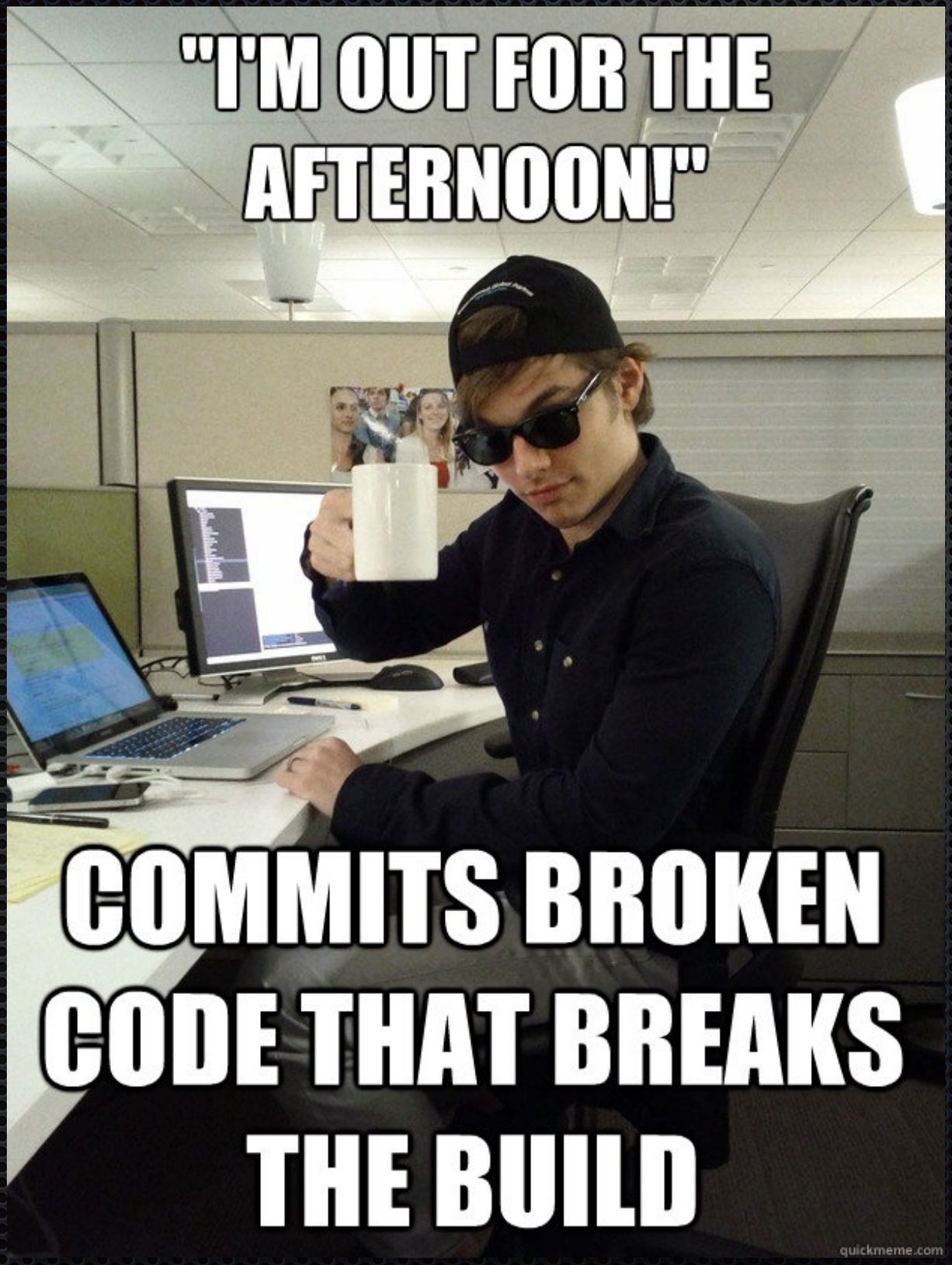

I'M GOING TO BE BLAMED FOR EVERY BUG
I GUARANTEE IT
imgflip.com

New project, fresh start

- Amazing speed at first
- Feature requests come and go
- Deadlines are looming
- Stress adds up

«Deadline is in 5 days. We are falling behind. If things can be solved with a quick hack, do it!»

–Anonymous Project Manager


"I'M OUT FOR THE AFTERNOON!"
COMMITS BROKEN CODE THAT BREAKS THE BUILD
quickmeme.com

«It’s just a bug in the calculation of VAT when we charge the credit card. I’ll make a quick fix. No need to test.»

–Anonymous Developer

Where are we?

- We have lost control of our code
- Fear of change
- No flexibility
- No agility

Testing Overview

Testing Categories

Tests can be separated into 4 distinct categories:

Manual Tests / User Acceptance Tests

Functional / UI / E2E Tests

Integration Tests

Unit Tests

Manual Tests / UATs

- Tests mirror user stories
- Passing tests mean fulfillment of customer requirements
- Slooooooooow: counted in days/weeks/months

UI / E2E Tests

- Test the flow of an application from start to finish
- Simulate real user scenarios
- Closest automated tests to production
- More often than not, they are flaky
- Very hard both to develop and maintain
- Very slow

Integration Tests

- Combine tested units
- Access resources (network / database)
- closer to production env
- Failures are hard to diagnose
- Harder to maintain
- Complete features might still not work

Main Target: Find bugs that UT can’t

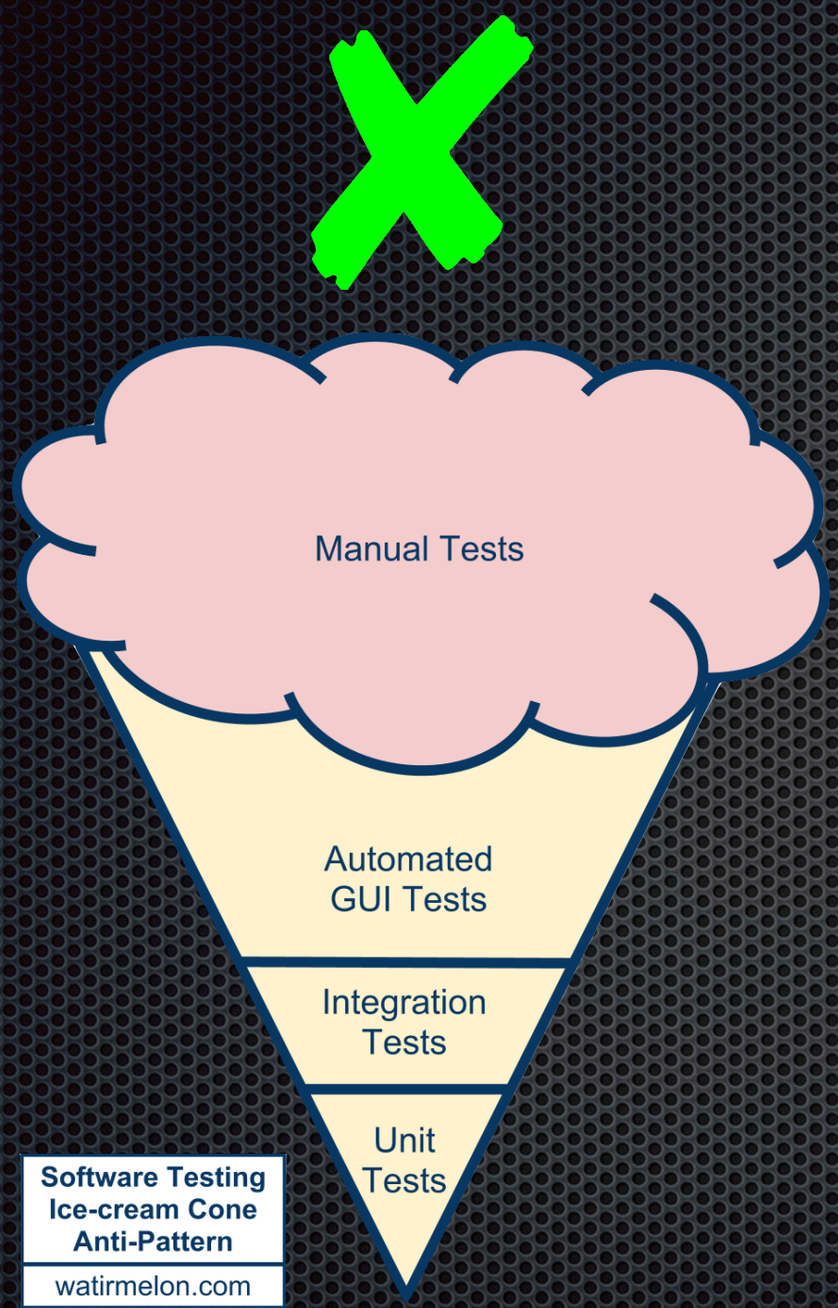
Dependency / Environment bugs

Unit Tests

- Test smallest unit of functionality
- Isolate SUT (System Under Test)
- Focus on particular feature
- Validate unit performs as designed
- Traceable errors
- Should be extremely fast
- Are extremely reliable

- Leave out (non-trivial) dependencies
- No network access
- No database access
- Do not spin threads

Testing Pyramid

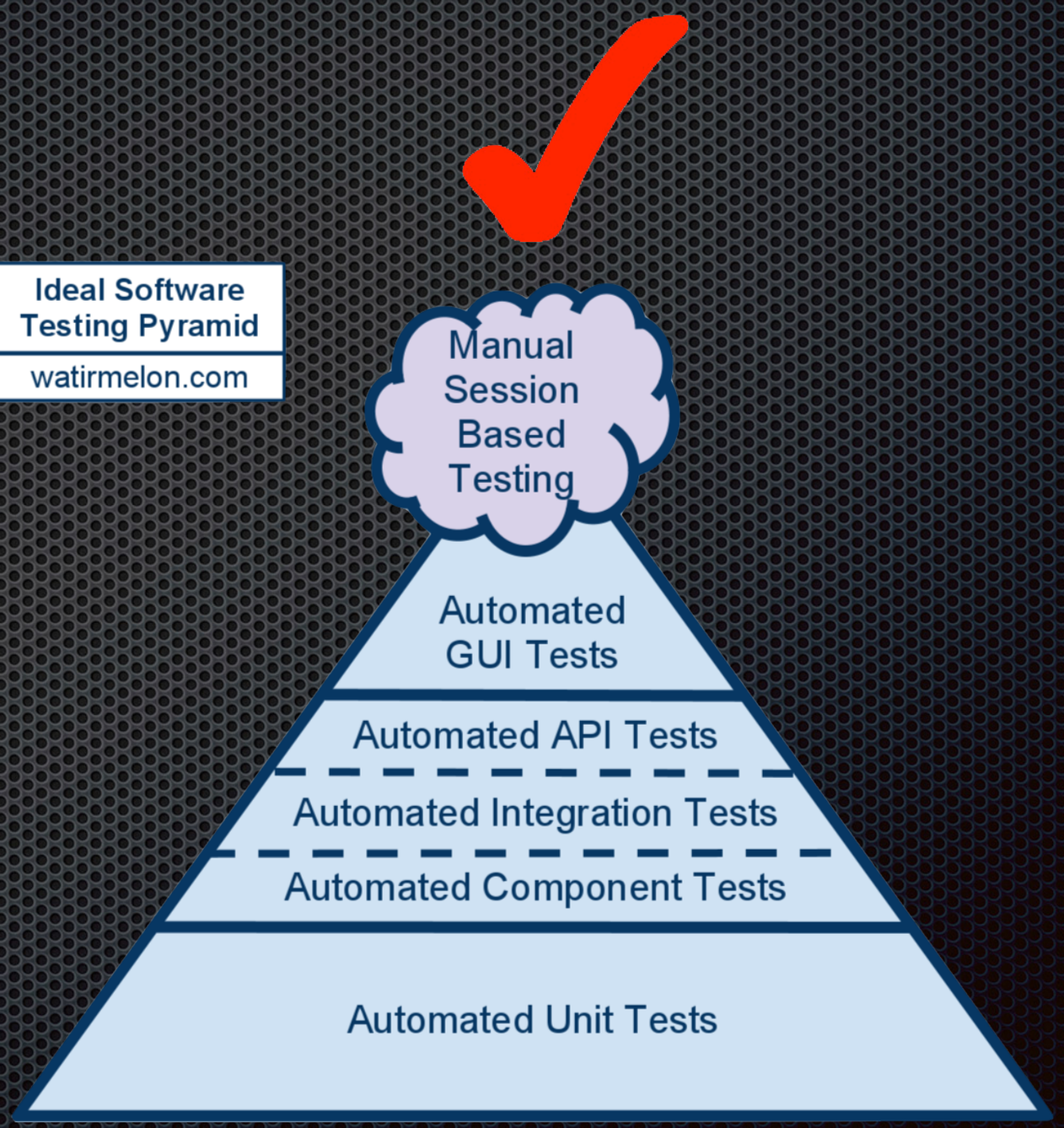


Software Testing Ice-cream Cone Anti-Pattern

watirmelon.com

Ideal Software Testing Pyramid

watirmelon.com



CI / CD

What is it?

- Development methodology
- Integrate new code
- Verify by automated build
  - and automated testing

Why use it?

- Build is automated and self-testing
- Test production clones
- Everyone gets early feedback
- Automated deployment

First Steps Test Case #1

How do I go about testing this?

```
public class MapDataRetriever {
    private final MapProvider provider;
    private final DataSource datasource;
    public MapDataRetriever(MapProvider provider, DataSource datasource) {
        this.provider = provider;
        this.datasource = datasource;
    }
    public MapData getMapData(int latitude, int longitude) {
        MapProperties props = datasource.fetchMapProperties();
        String response = provider.retrieveDataAPI(props, latitude, longitude);
        return parseData(response);
    }
    private MapData parseData(String dataResponse) {
        //data representation mapping
    }
}

public class MapDataRetrieverTest {@Test
    public void testXXX() throws Exception {
        //assert?
    }
}
```

First Steps Test Case #2

How do I go about testing this?

```
public class MapDataFileParser {
    private final File mapDatafile;
    public MapDataFileParser(File mapDatafile) {
        this.mapDatafile = mapDatafile;
    }
    public List < MapData > parseFile() throws Exception {
        ImmutableList.Builder < MapData > listBuilder = ImmutableList.builder();
        try (BufferedReader br = new BufferedReader(new FileReader(mapDatafile))) {
            MapData mapData = parseLine(br.readLine());
            listBuilder.add(mapData);
        }
        return listBuilder.build();
    }
    MapData parseLine(String line) {
        //split line to pieces and convert to map data
    }
}


public class MapDataFileParserTest {@Test
    public void testXXX() throws Exception {
        //assert?
    }
}
```

First Steps Test Case #3

How do I go about testing this?

```
public class SimpleDataCollector extends AbstractDataCollector {
    private final Executor executor;
    private final HttpClient client;
    private final JSONParser parser;
    private final DataSource dataSource;

    public SimpleDataCollector() {
        executor = Executors.newScheduledThreadPool(4);
        client = new HttpClient(new MultiThreadedHttpConnectionManager());
        parser = new JSONParser();
        dataSource = createDataSource();
    }

    public void performTask() {
        //combine all of these to perform a task
    }
}

public class SimpleDataCollectorTest {@Test
    public void testXXX() throws Exception {
        //???
    }
}
```

The Value in Testing, for Management

a.k.a. how to convince your boss

But WHY do you even need to convince them?

The Ignorant Boss Theorem

"Your code should be working already!!
Why do you need tests?"

– *George Boss*

The Blaming Manager Constant

"Well, Why haven't you been doing it up till now?"

– John Manager

The Venture Capital-less Dilemma

"So… Let me get this right…

You are asking me to invest more money?"

– José Cheapita Nomoneda

The Busy Wait Loop

"We don't have time for that now"

– Antonio Bizi

CHANGE MAKERS

How Much Time Is It Taking to Manually Test ?

Multiply by # Releases

How Frequent Are Your Releases?

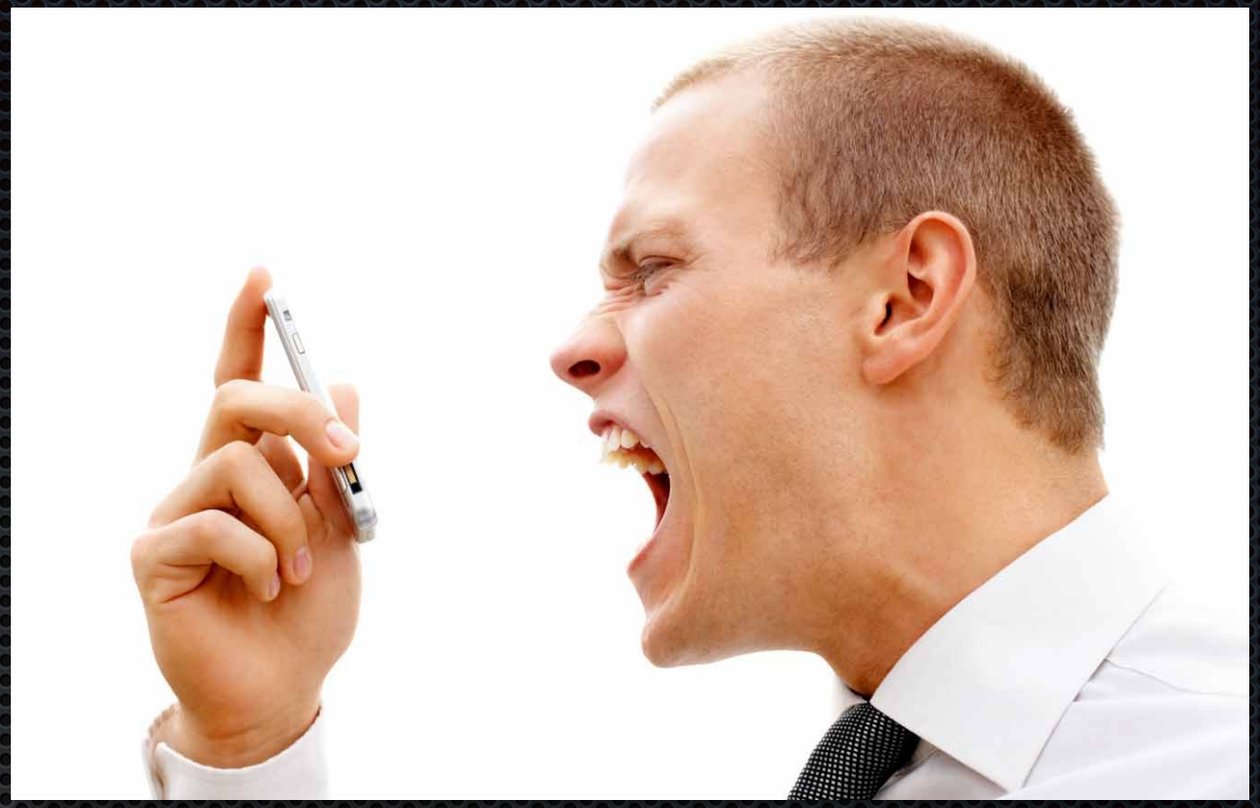
Let's say, in the last year?

Do they count customer complaints owing to bugs?

Do you even use a ticketing system?

How many weekends have you worked extra?

...and they paid you for?



Have they had at least one client tell them:

…didn’t we fix that again a couple of weeks ago?

Last but not least...

This is really special.....

Wait for it....

Ok, here goes…

their competition does it.

For a Happily Ever After...

- Try to explain in business terms
  (free tips: ‘save money’, ‘retain customers’, ‘reduce time to market’…)
- Don’t tell them how to run their business
  (so they don’t tell you how to code…)
- Don’t expect it all at once. Change takes time =>
  Baby Steps

Test - Driven Development

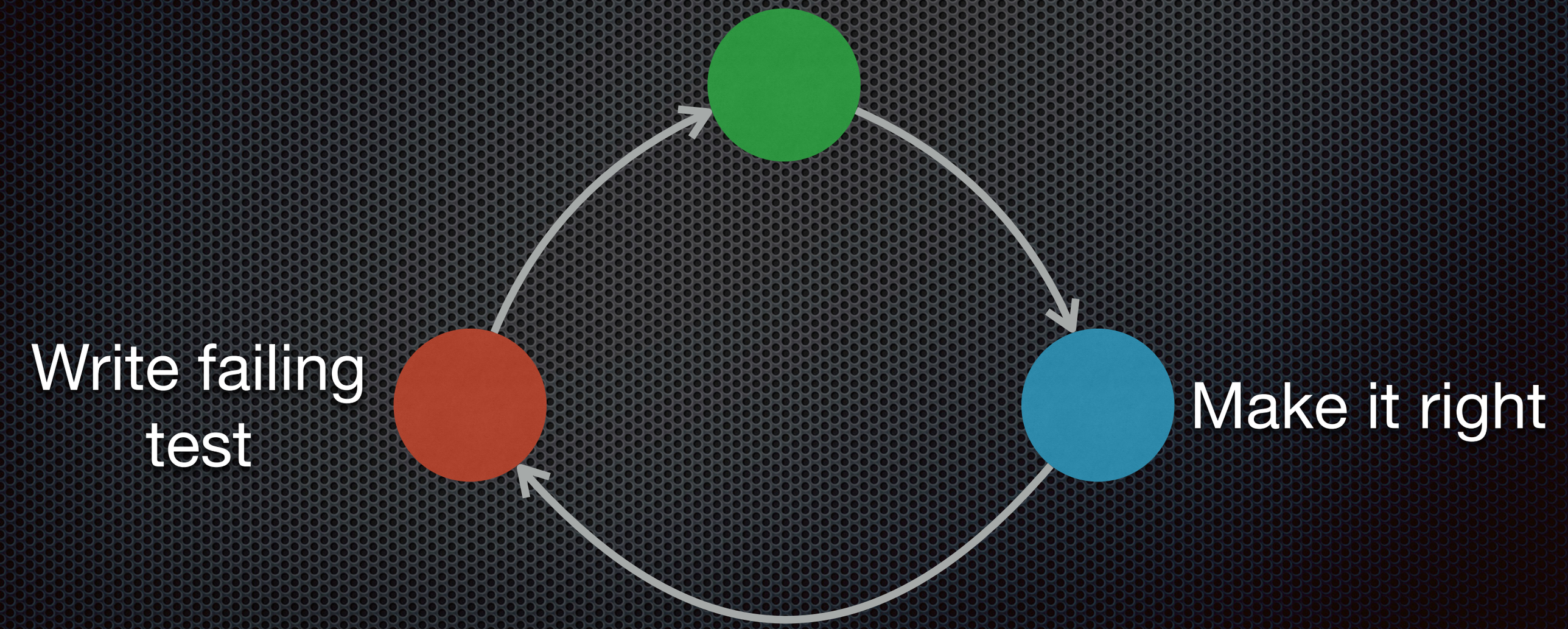
Let's Remove The Fear

Nikolas Vourlakis

The Laws

- You are not allowed to write any production code unless it is to make a failing unit test pass
- You are not allowed to write any more of a unit test than is sufficient to fail; and compilation failures are failures
- You are not allowed to write any more production code than is sufficient to pass the one failing unit test

Red - Green - Refactor



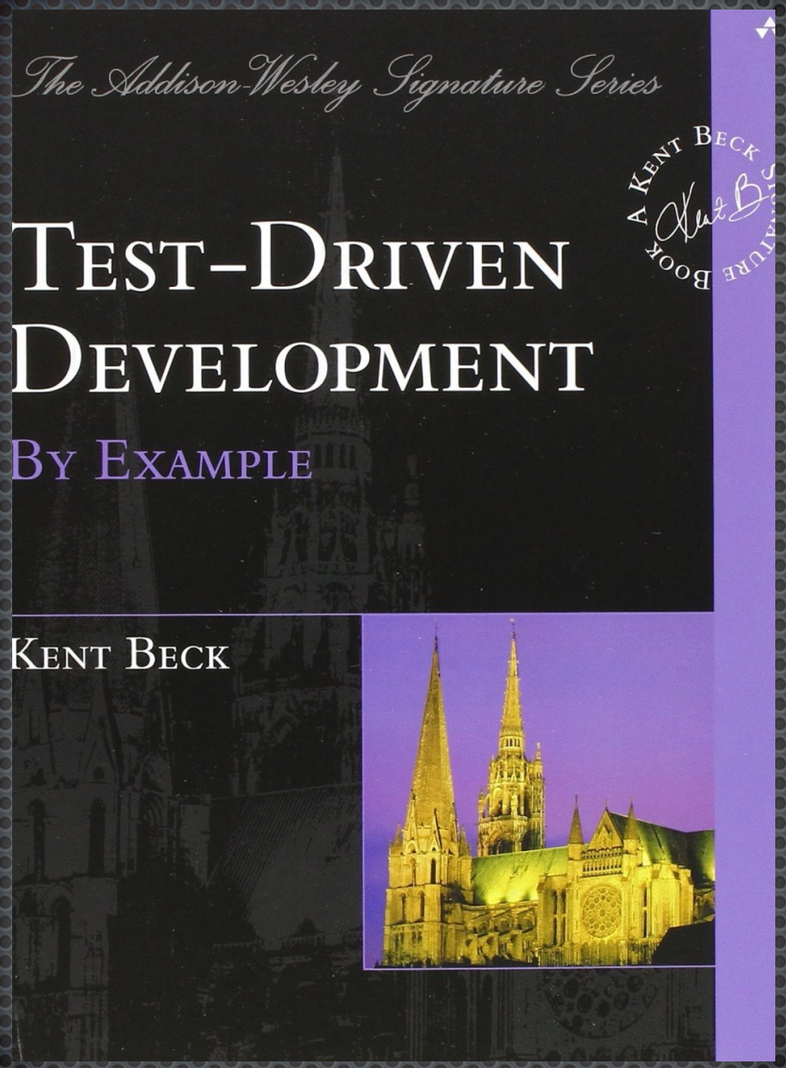
What we gain

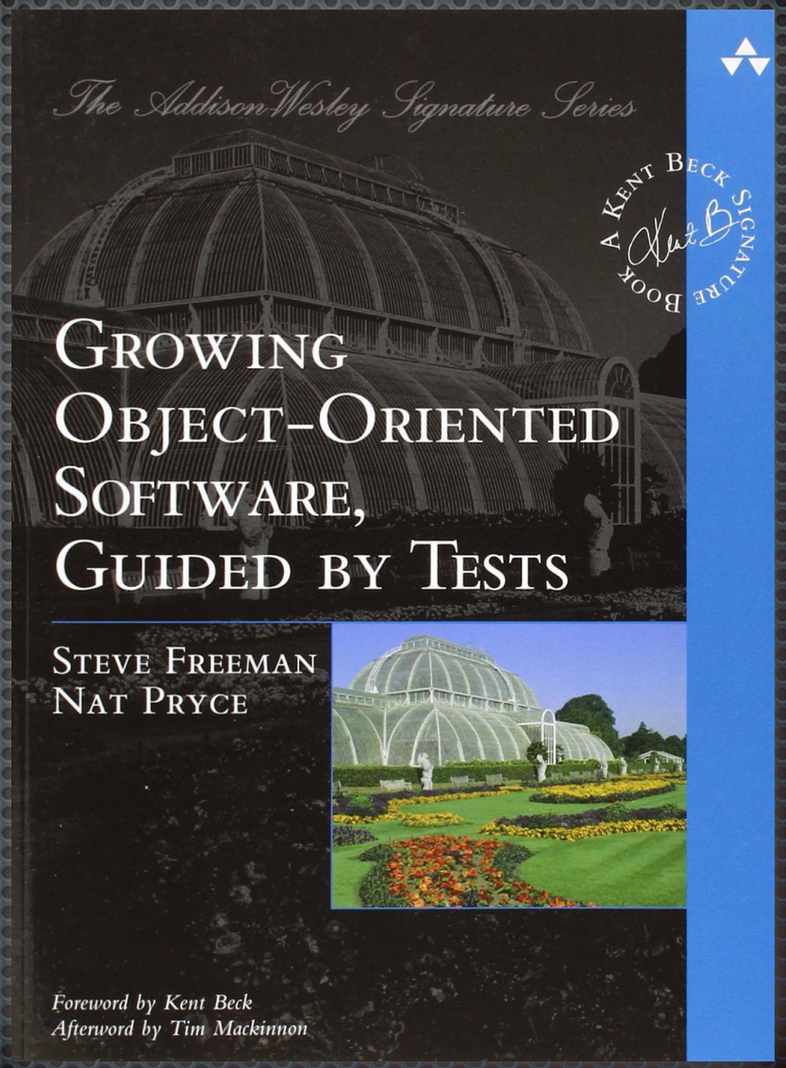
- I am in control of the code
- Fearless refactoring and thus agility
- Decoupled components
- Debugging? What’s that? (sort of)
- Low level documentation

Tips to get started

- Test behavior, not just functions or classes
- ALWAYS write the test first
- Quickly getting to green dominates everything else
- Don’t forget the “make it right” part, aka clean code
- Listen to your tests

What to read?


The Addison-Wesley Signature Series
A Kent Beck Signature Book
Test-Driven Development
By Example
Kent Beck


The Addison-Wesley Signature Series
A Kent Beck Signature Book
Growing Object-Oriented Software, Guided by Tests
Steve Freeman
Nat Pryce
Foreword by Kent Beck
Afterword by Tim Mackinnon

Let’s do a Code Kata

http://osherove.com/tdd-kata-1/