



Unit Testing

DIEGO PACHECO

About me...



- ☐ Cat's Father
- ☐ Principal Software Architect
- ☐ Agile Coach
- ☐ SOA/Microservices Expert
- ☐ DevOps Practitioner
- ☐ Speaker
- ☐ Author



diegopacheco



@diego_pacheco



<http://diego-pacheco.blogspot.com.br/>



<https://diegopacheco.github.io/>

**MY CODE DOESN'T WORK AND
I DON'T KNOW WHY**

A man in a white shirt is shown from the side, resting his head on his hand next to a laptop. He appears to be exhausted or frustrated. The text "MY CODE WORKS AND I DON'T KNOW WHY" is overlaid on the image.

**MY CODE WORKS AND I DON'T
KNOW WHY**

Too busy to improve our tests...



The culture we need...



- ❑ *Tests allow SAFE Refactoring*
- ❑ *Tests are a SAFETY REQUIREMENT*
- ❑ *We should NEVER skip them*
- ❑ *We need to TRUST our tests, FLAKY tests are worse than NO TEST.*
- ❑ *We need stop the EXCUSES.*
- ❑ *Just do it.*

Dimensional quality == Dimensional Tests

XP Evolutionary Approach



Chão Batido



Paralelepípedo



Autoestrada



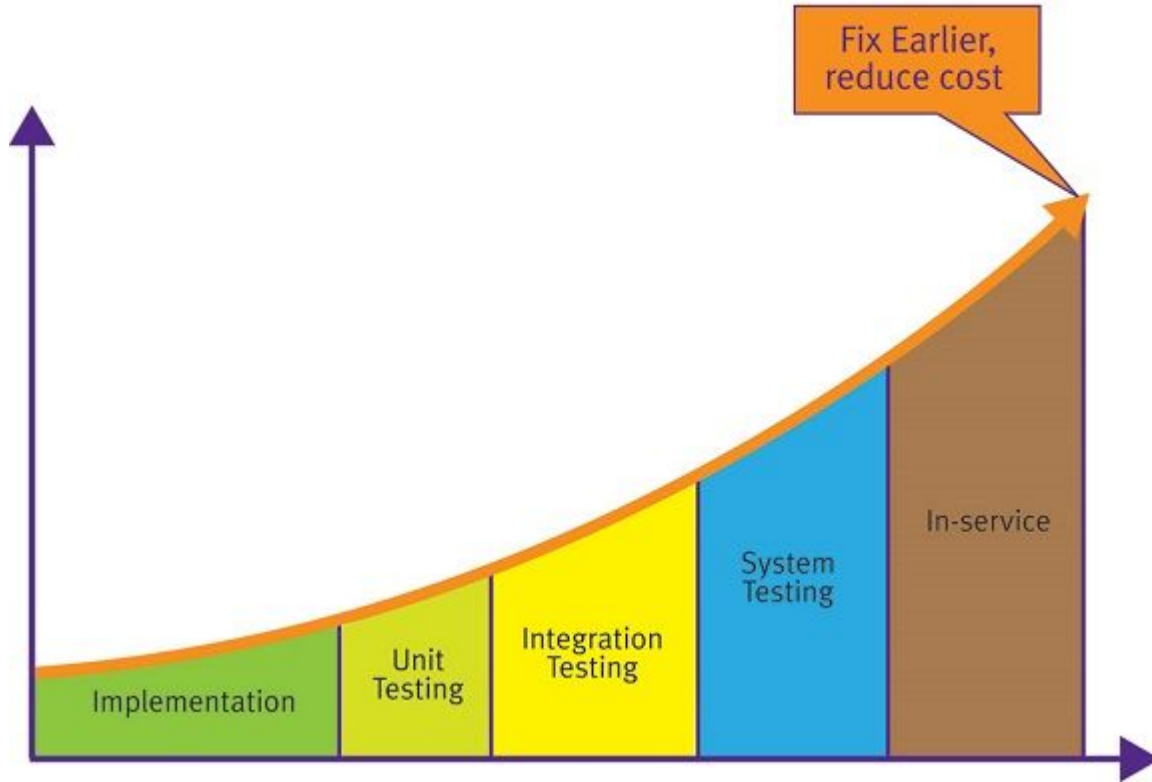
The TESTING Manifesto

we value:

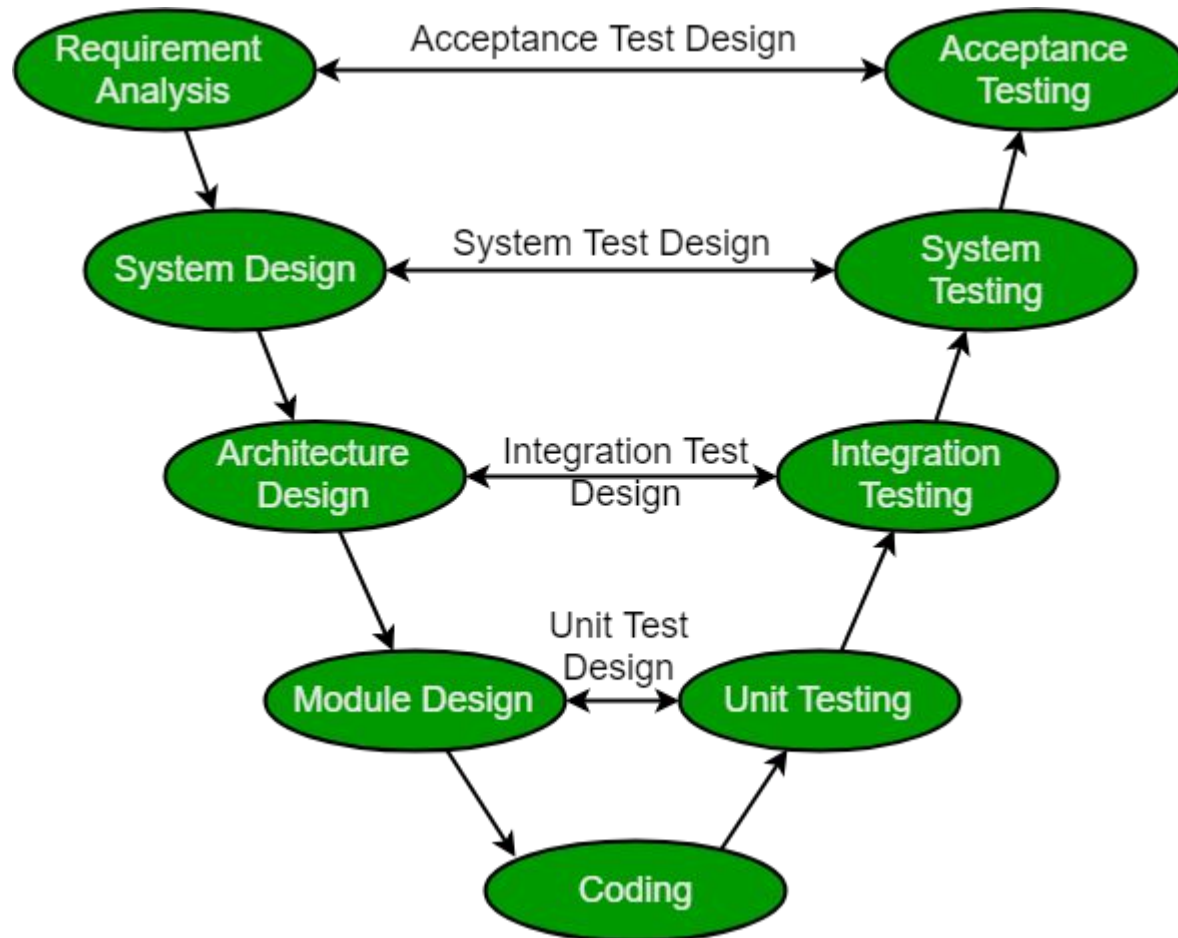
- Testing throughout **over** at the end
- Preventing bugs **over** finding bugs
- Testing understanding **over** checking functionality
- Building the best system **over** breaking the system
- Team responsibility for quality **over** tester responsibility

Testing == Economics

The Cost of Defects



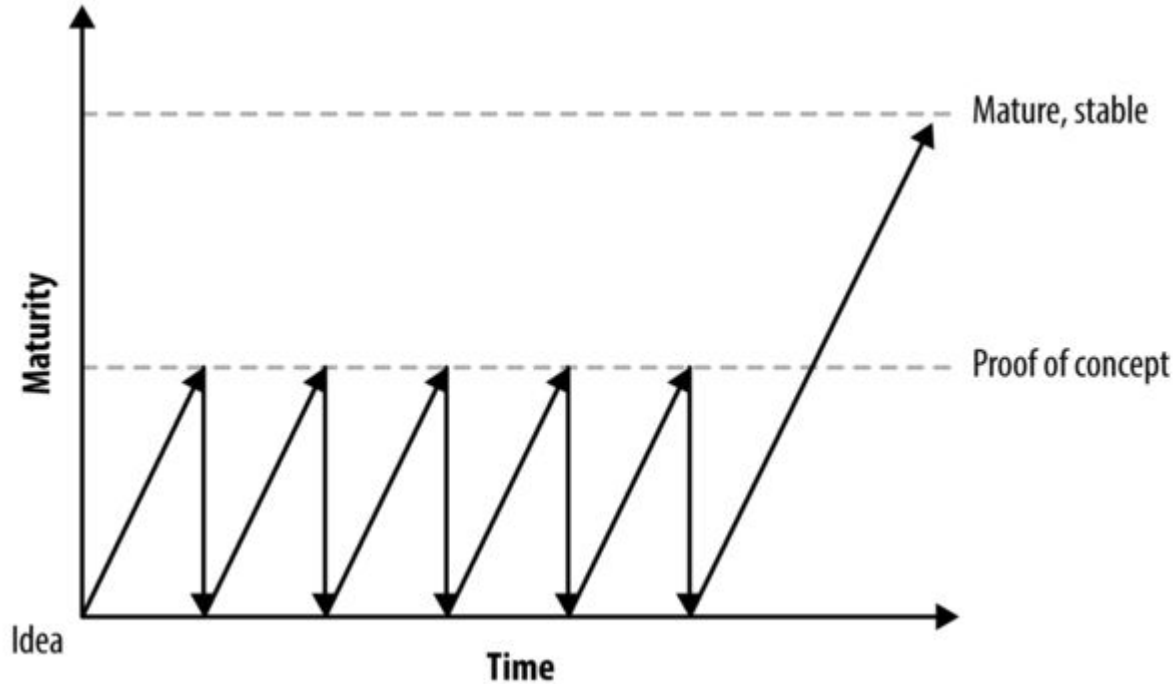
V MODEL



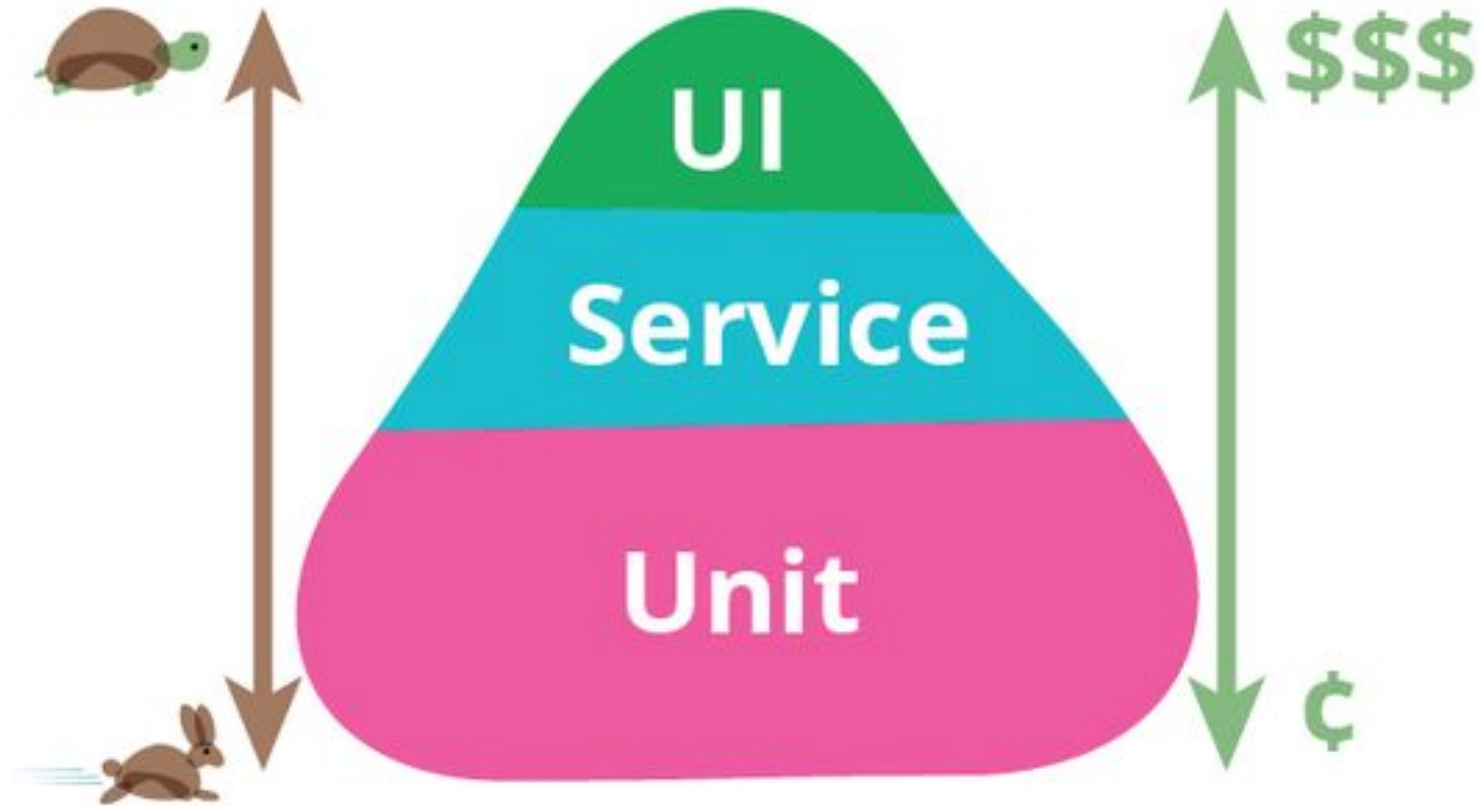
Testing is for everybody!

BUSINESS	
CHECK FOR EXPECTED OUTPUTS	SPEC BY EXAMPLE PERFORMANCE COMPLIANCE REGRESSION HYPOTHESIS (LS)
	EXPLORATORY USABILITY STAKEHOLDER IMPACT USAGE ANALYTICS A/B
	UNIT (TDD) INTEGRATION DATA FORMATS API COMPATIBILITY
	LOAD PENETRATION PRODUCTION TRENDS SMOKE
TECHNOLOGY	
ANALYSE UNDEFINED, UNKNOWN AND UNEXPECTED	

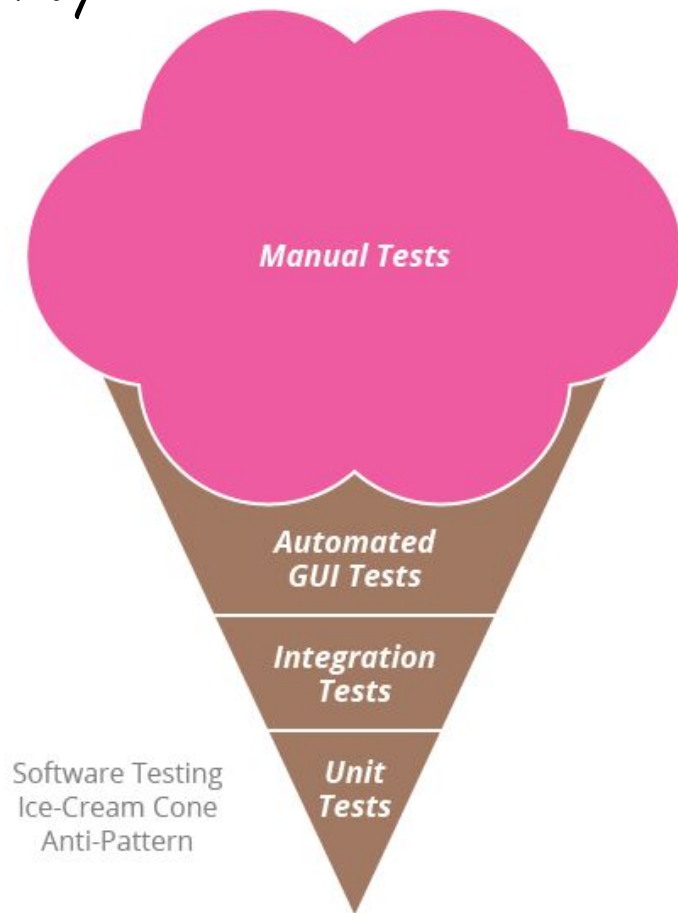
Discovery, MVP, Lean Startup, UX are about testing...



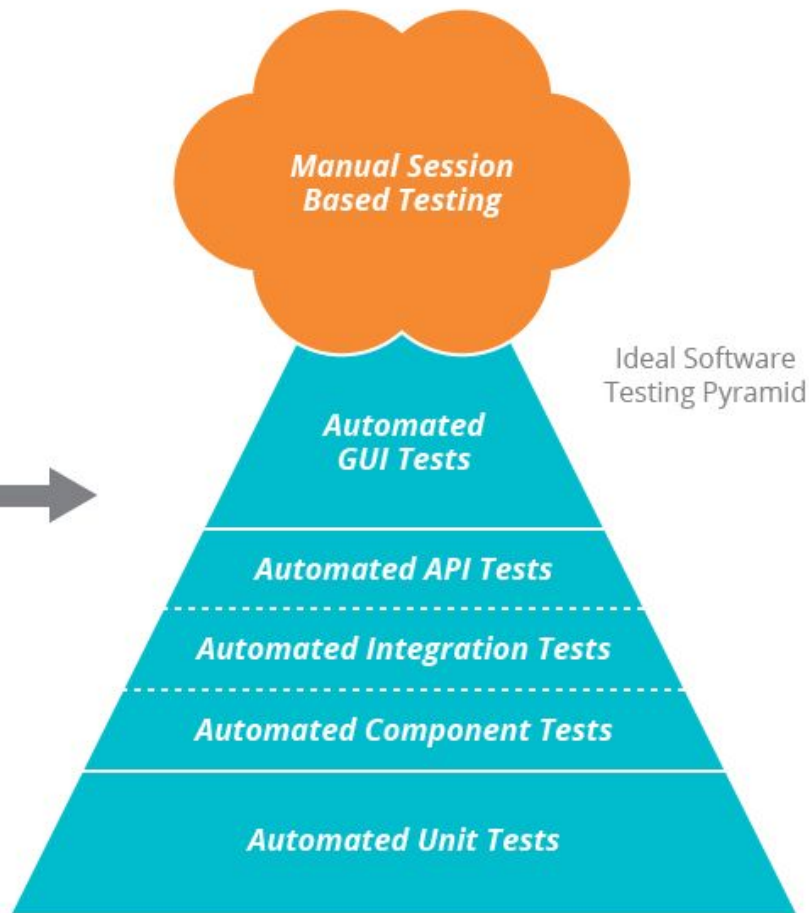
Testing Pyramid



Reality...



Software Testing
Ice-Cream Cone
Anti-Pattern



Ideal Software
Testing Pyramid

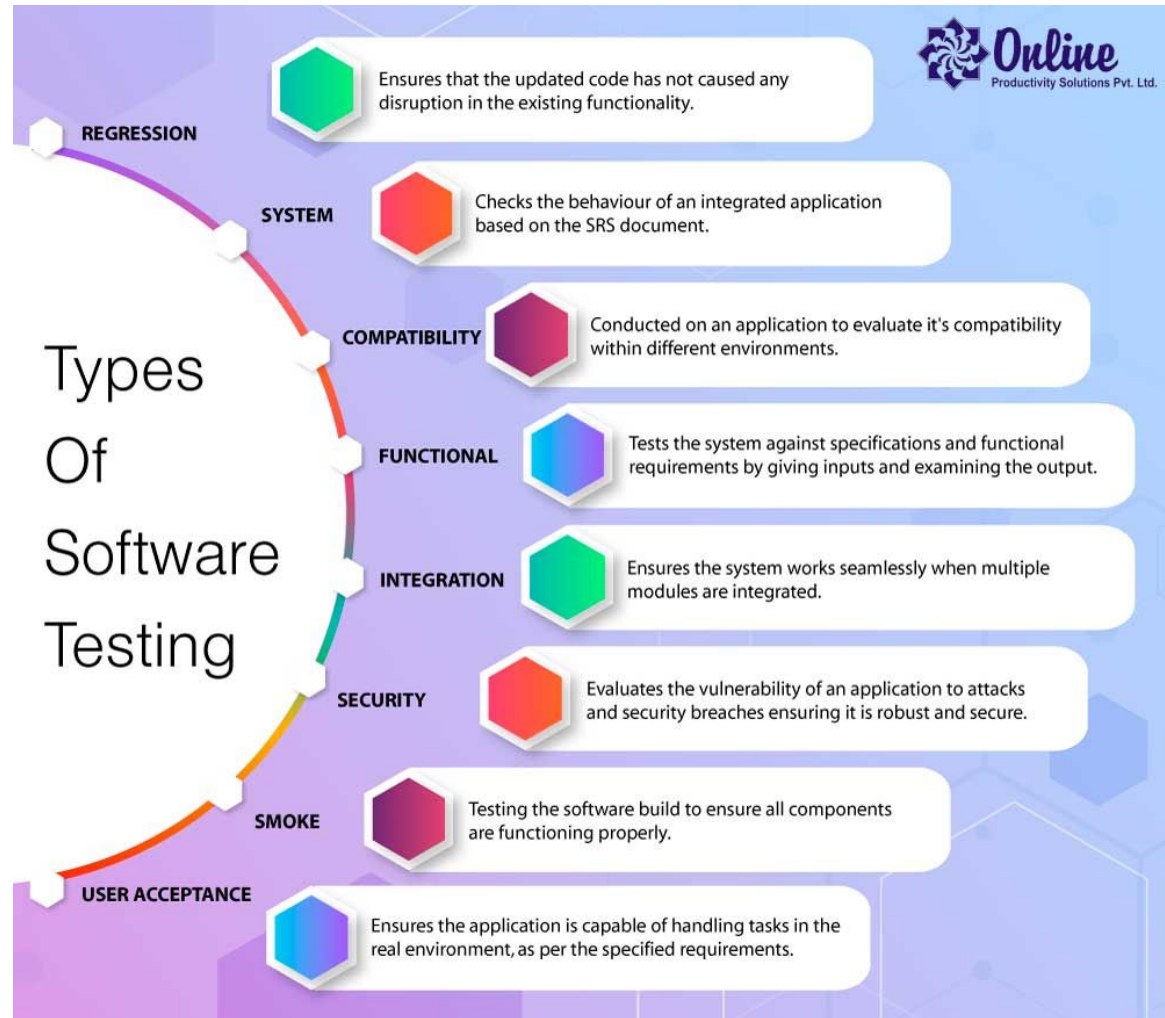
Level of testing...

Mock Contract

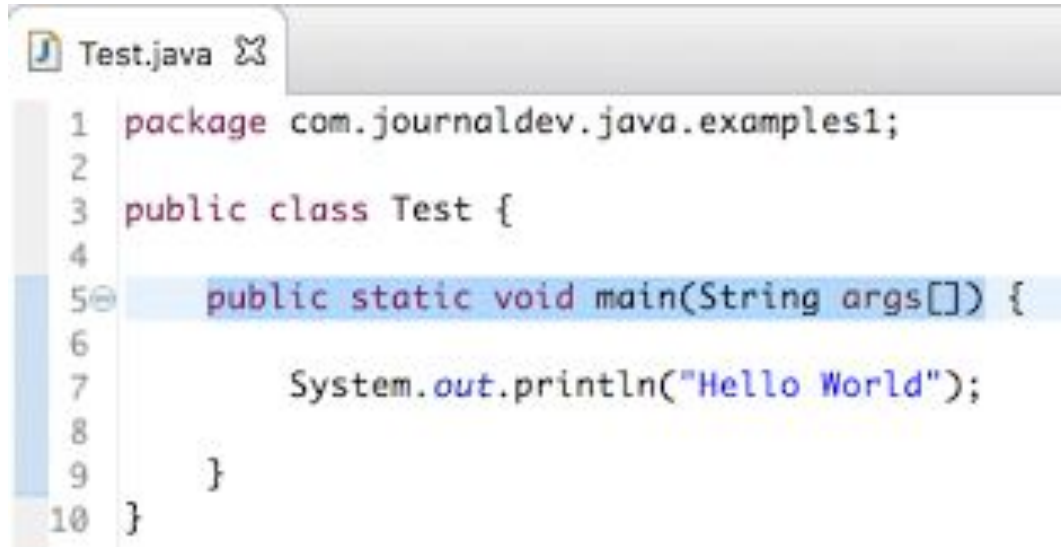
Stress Testing

Chaos Testing

Developer Testing



Developer Tests:: It's all about SPEED

A screenshot of a Java IDE window titled 'Test.java'. The code is as follows:

```
1 package com.journaldev.java.examples1;  
2  
3 public class Test {  
4  
5     public static void main(String args[]) {  
6  
7         System.out.println("Hello World");  
8  
9     }  
10 }
```

The line containing the `main` method signature is highlighted in blue.

- ❑ Faster discovery
- ❑ Faster validation
- ❑ Faster Debugging
- ❑ Faster Layouting
- ❑ It's a MUST !!!!
- ❑ Simplified "Service"
- ❑ Simple Main Class
- ❑ Debug tricks.

Mocked Contract



- ❑ *How do we know we don't break the CONSUMERS?*
 - ❑ *Frontend*
 - ❑ *Mobile*
 - ❑ *BFFs*
- ❑ *We should:*
 - ❑ *Version*
 - ❑ *Fast Regression props*
 - ❑ *Know before the CONSUMERS any breaking change.*

Stress Tests



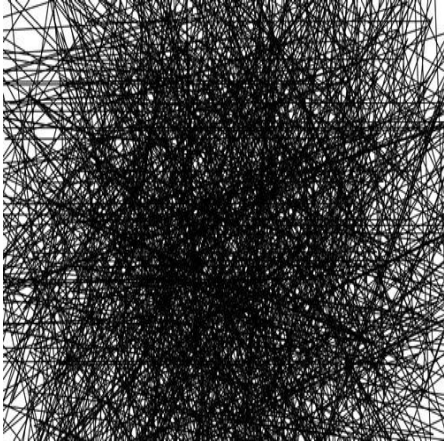
- ❑ Stress / Load Testing
- ❑ How I know my bottlenecks?
- ❑ When the system breaks? Why it breaks?
- ❑ We need to play with:
 - ❑ Sensitive Operations
 - ❑ Number of concurrent users
 - ❑ Load Patterns
- ❑ Best done via Gatling (<https://gatling.io/>)

Chaos Testing



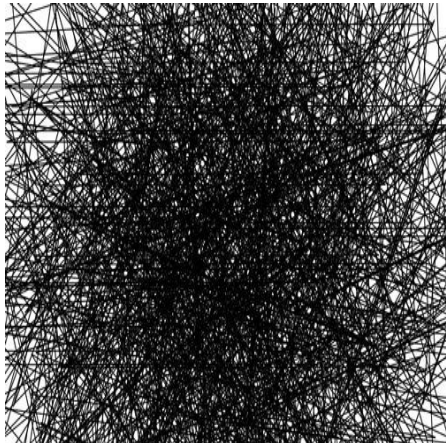
- ❑ *Software fails all the time.*
- ❑ *Hope is not a strategy*
- ❑ *Types of failure:*
 - ❑ *Datacenter is down.*
 - ❑ *AZ is down.*
 - ❑ *Service is down*
 - ❑ *Latency*
 - ❑ *Burn: CPU, network, Disk.*
 - ❑ *Bombos: Strings, JVM, volume...*
- ❑ *Best done via Netflix tools, linux and creativity*

Testing beyond the happy path....



- ❑ Null
- ❑ Invalid dates
- ❑ Special chars in Strings
- ❑ List corner cases
- ❑ Negative numbers
- ❑ Long ranges
- ❑ Wrong objects
- ❑ Think in scenarios before CODE.

Testing beyond the happy path....



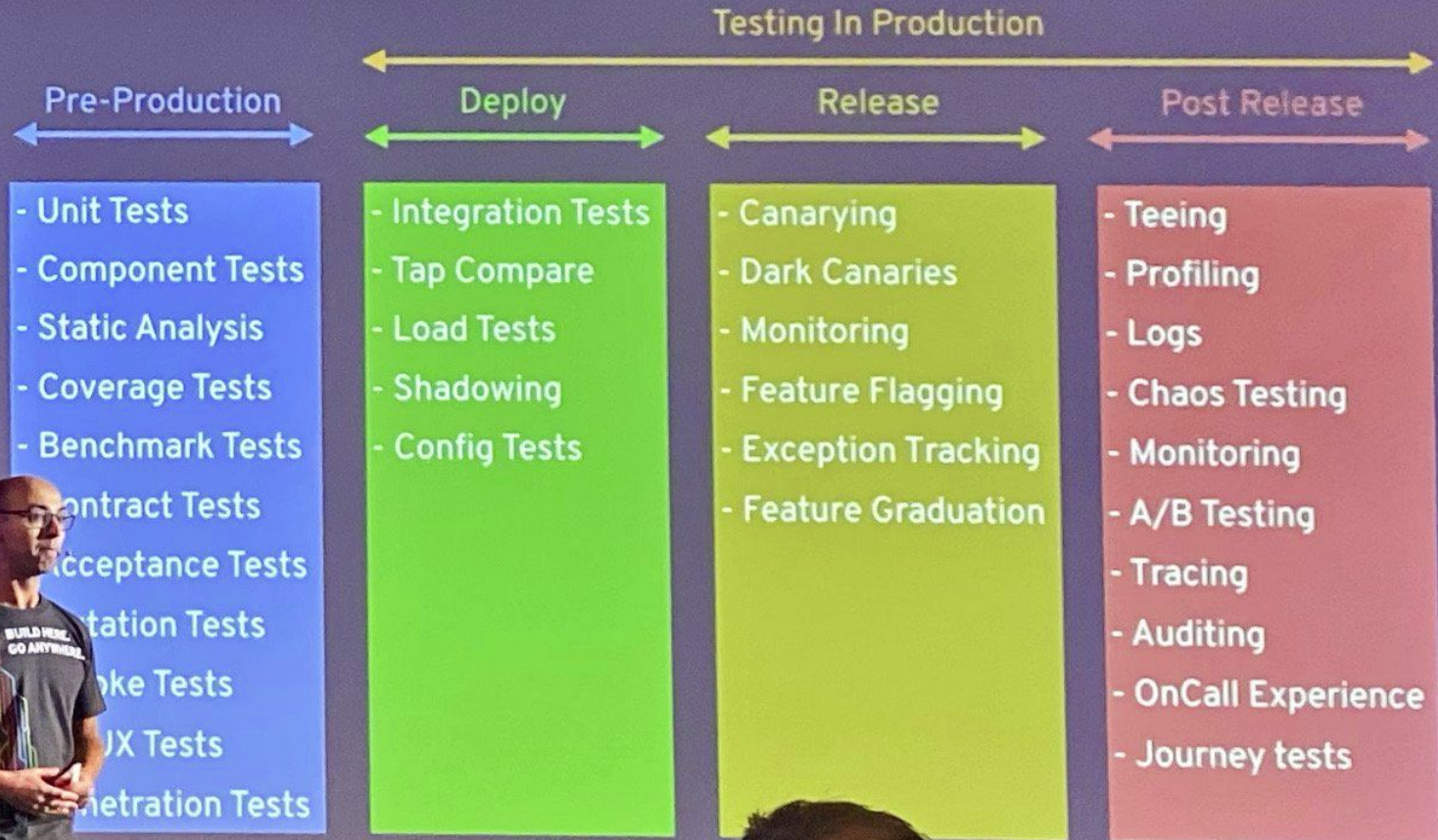
- ☐ Invalid Order
- ☐ Did you try exploratory ramon crazy tests?
- ☐ What if the dependencies are not there?
- ☐ How possible exceptions we might have?
- ☐ What if we it "works"...

Why is important to have *COVERAGE GOALS*



- ❑ *Most Engineers don't learn TESTS on university*
- ❑ *Some companies still have QAs*
- ❑ *Developer need to understand that is his responsibility to ship STABLE CODE.*
- ❑ *CODE Review Kanban queue policies +10 per class.*
- ❑ *XP BIG Visible charts on the WALL*
- ❑ *Feel the same of the lack of tests.*

The New ¿Pyramid?





We will implement some Features for a BANK customer and we will create tons of TESTs and we will refactor the code.

<https://github.com/diegopacheco/unit-testing-training>

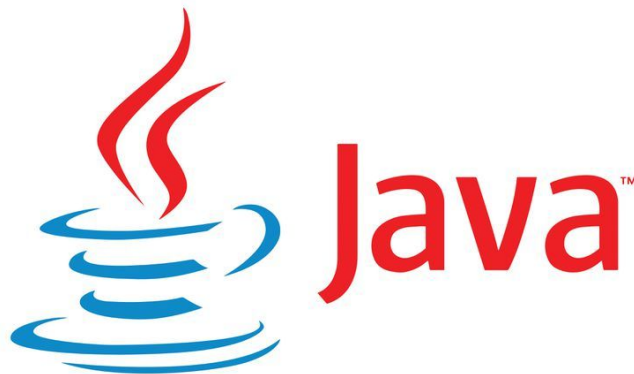
What we are building

So we are building a basic bank. We need to develop the following business capabilities:

- A) The USER must have the ability to *deposit* money.
- B) The USER must have the ability to *withdraw* money.
- C) The USER must have the ability to *check* his money balance.
- D) The USER must have the ability to *transf* his money to other account with 5% bank fees.
- E) The USER must have the ability to *save* his money into savings account where he will get %22 gains per minute with 2% bank fee on the *withdraw*.

What do we need todo

1. We will talk with the "Discovery" team who did a poor job discovering stuff to "extract" requirements.
2. We will DESIGN the minimal solution and grow as we need it. We need at least 1 diagram.
3. We need create proper unit tests that validate the core-features and also validate the proper happy path but also the corner-cases or failure scenarios as well.
4. As we implement the tests we will implemet the business capabilities and refactor the code. We should be running the tests all the time.





Unit Testing

DIEGO PACHECO