

Agile + DevOps **WEST**

A TECHWELL EVENT

AD31

DevOps Engineering
10:00 AM

AD31 - Using Component Testing for Ultra-Fast Builds

Presented by:

Timothy Cochran

Thoughtworks

Brought to you by:



888-268-8770 · 904-278-0524 - info@techwell.com - <https://agiledevopswest.techwell.com/>

Timothy Cochran

Tim Cochran currently works for ThoughtWorks NYC, and has been full stack developer and architect for over 15 years. Working on everything from large distributed enterprise projects to small NGO visualization apps. He recently has been helping companies with digital transformation, moving towards continuous delivery and building a DevOps culture. He also is an automated testing zealot, practices TDD almost exclusively, he builds test strategy using the right combinations of unit, E2E and functional testing for clients.



Utilizing Component Testing for Ultra fast Builds



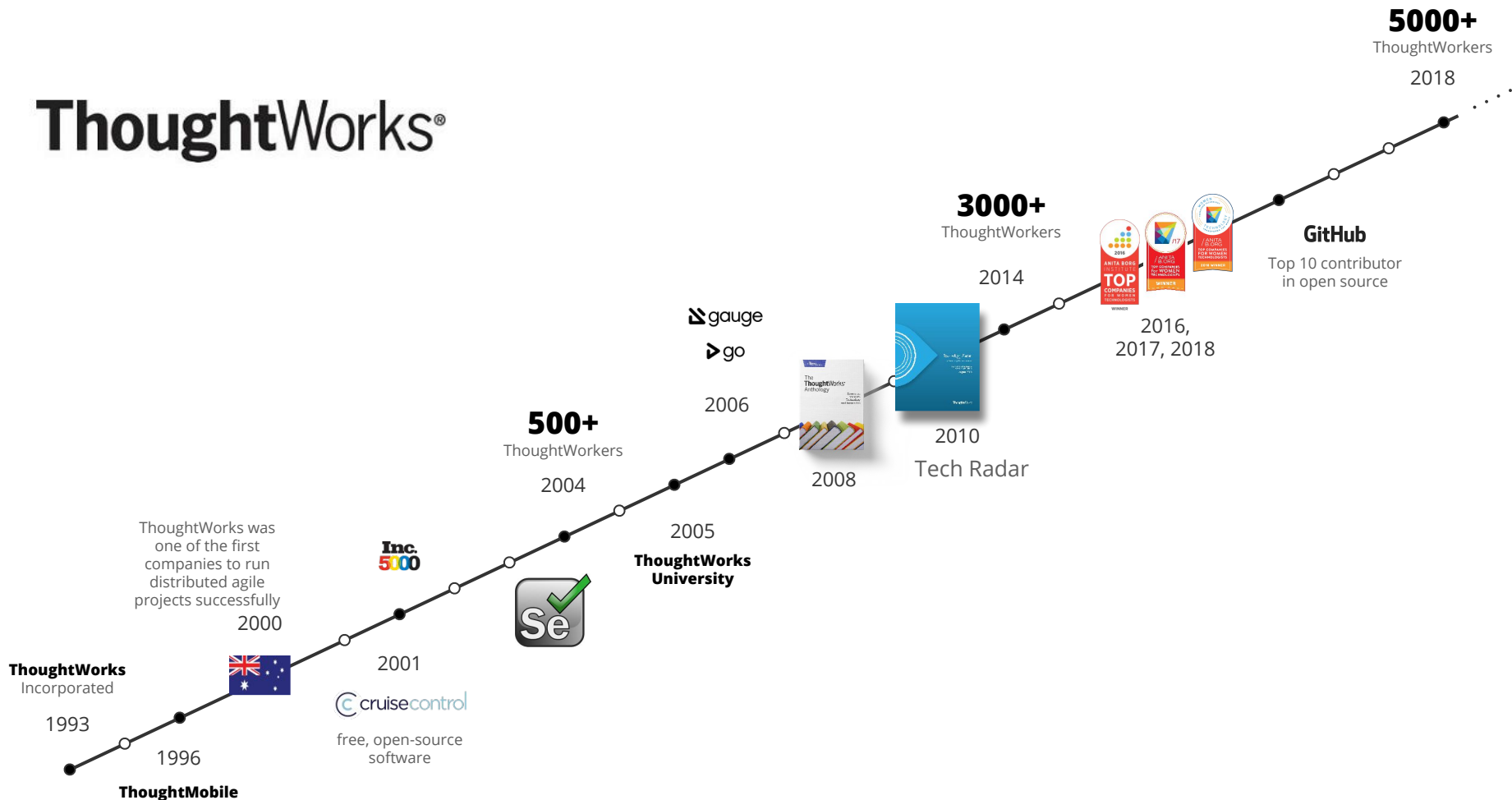
PERSONAL
BLOG
FEEDS

Contact to
Rebrand

ThoughtWorks®

GLOBAL SOFTWARE CONSULTANCY

ThoughtWorks®



Why Automated Testing



Cost of QA



Functional & Non-Functional Problems



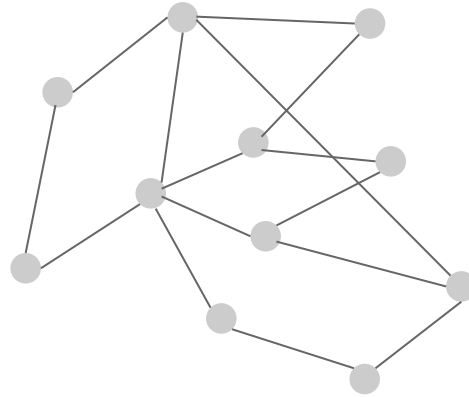
Productivity



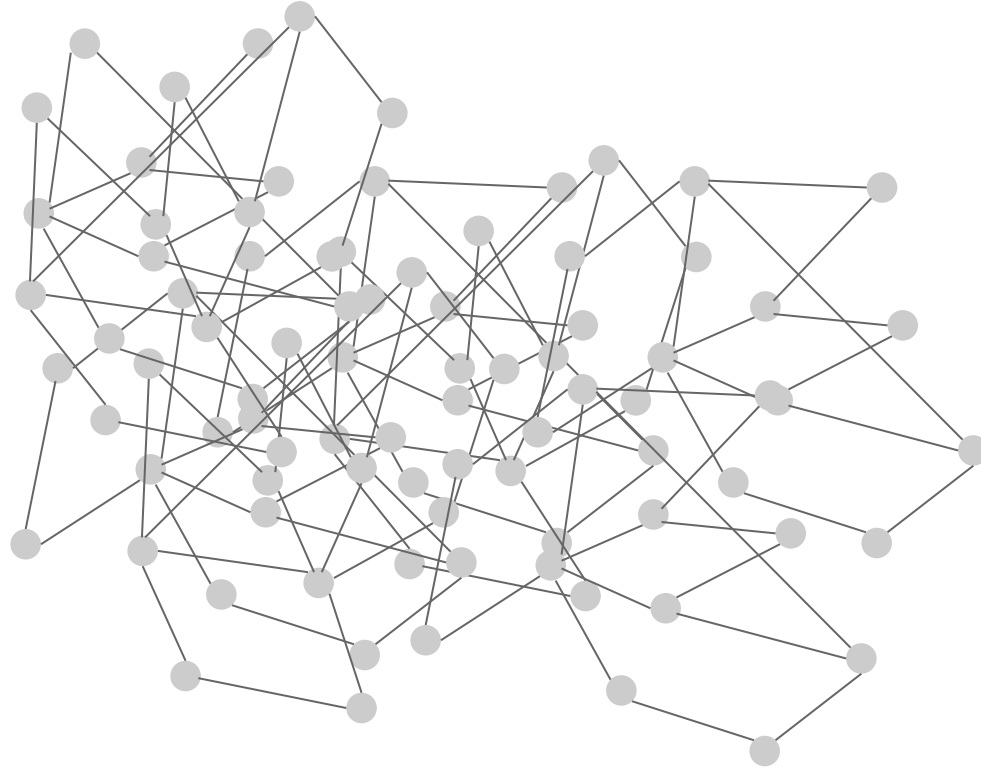
User Sentiment

The Problem

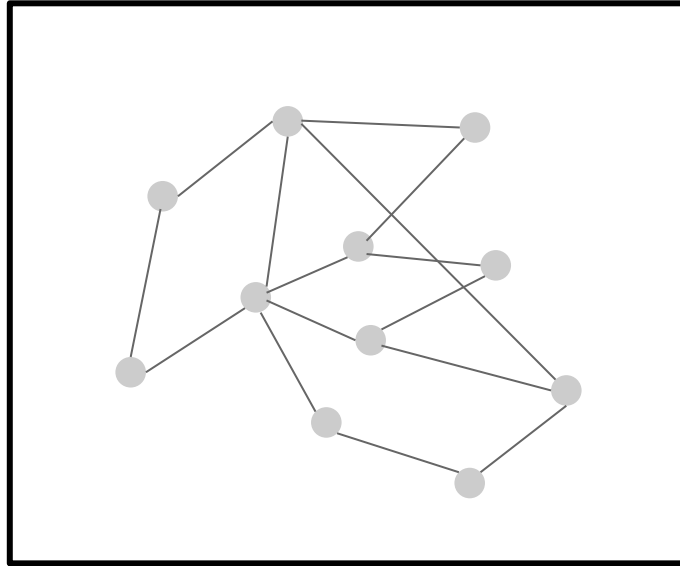
Software Design



Software Design

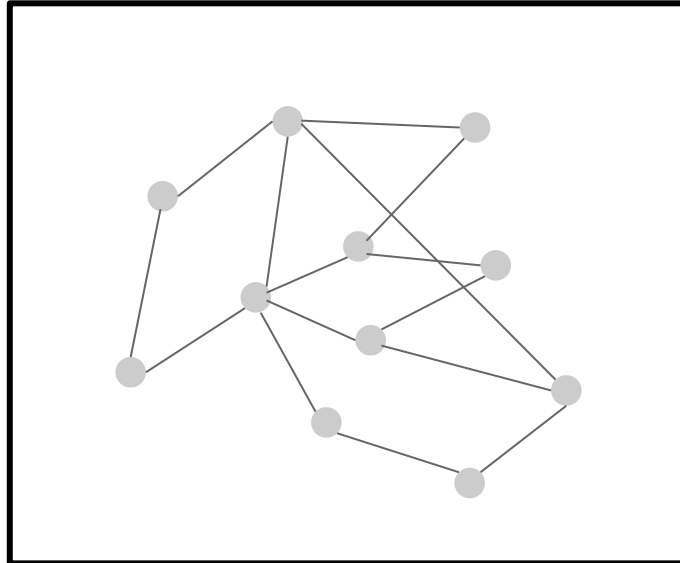


Black Box



Black Box

E2E tests



Build / Test pipeline (30 mins)



Deployment pipeline (15 mins)



E2E / Regression test pipeline (2 hours 5 mins)



Build / Test pipeline (30 mins)



Deployment pipeline (14 mins)



E2E / Regression test pipeline (35 mins)



Anti-patterns

Unit test - Deploy - Regression are separate pipelines, owned by different teams

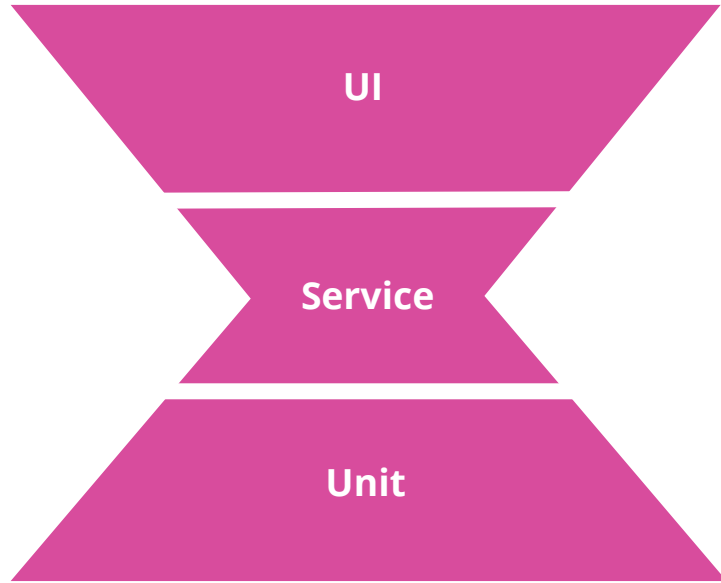
Flakey tests, Fails one in 5 tests

Build is never green, Deploying with failing tests

Environments that don't reflect Prod, Software still breaks in production

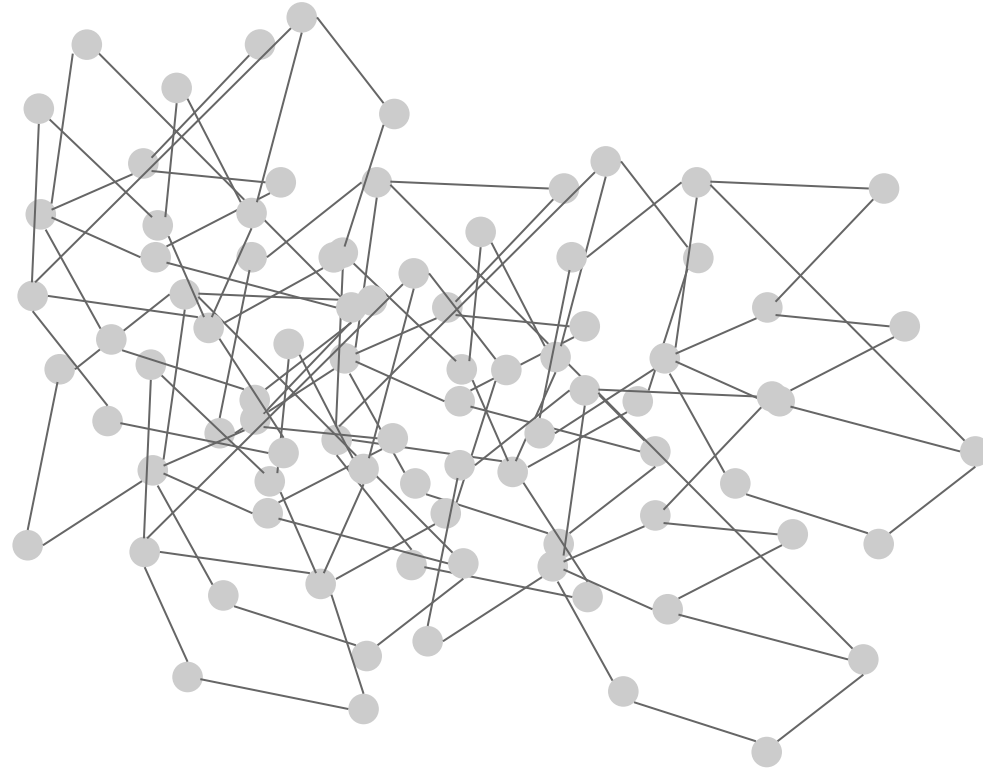
Broken Environments, Tests aren't run because env is down or misconfigured

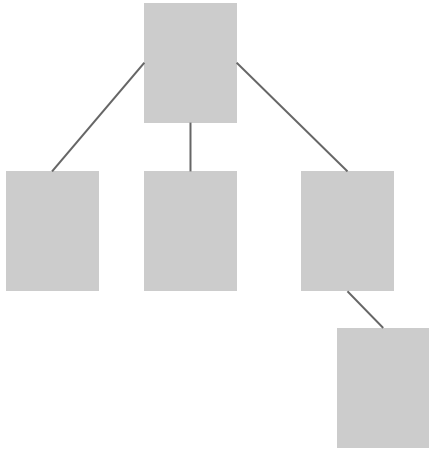




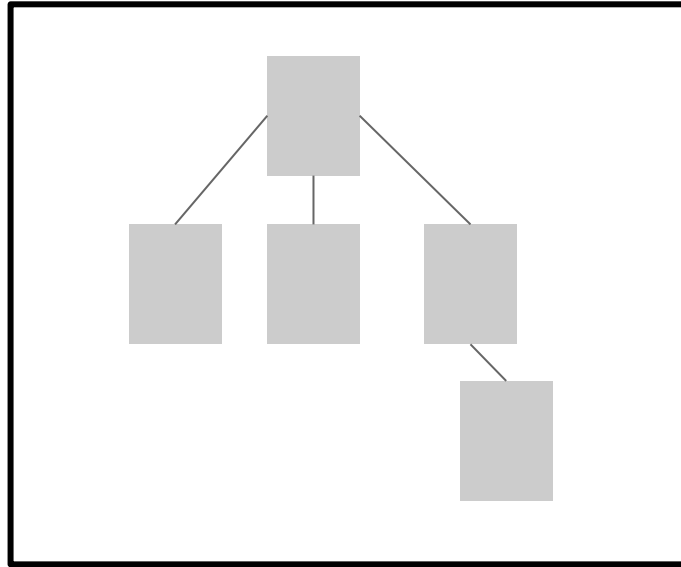
Component Testing

Software Design

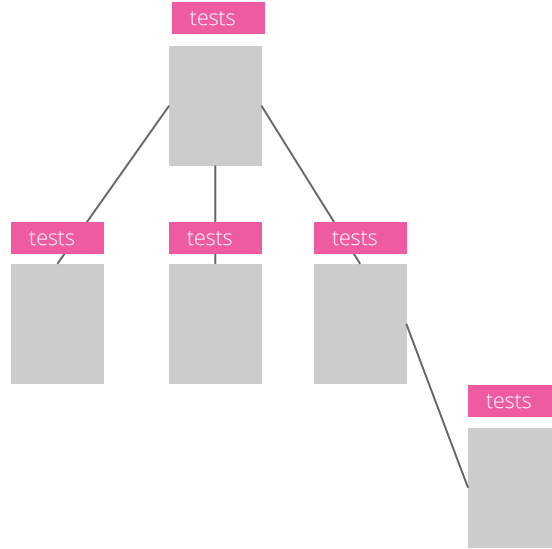




E2E tests



Component Testing



Types of Components

A code module

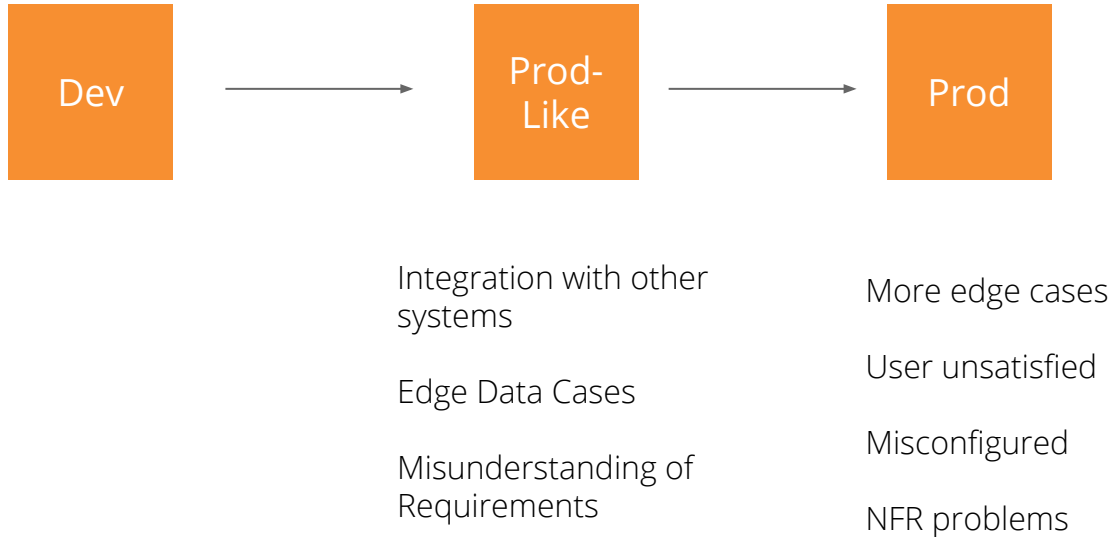
A data pipeline

Third party service

UI Component

Microservice

What goes wrong?



Component / Service

Edge Cases

Functionality (Acceptance)

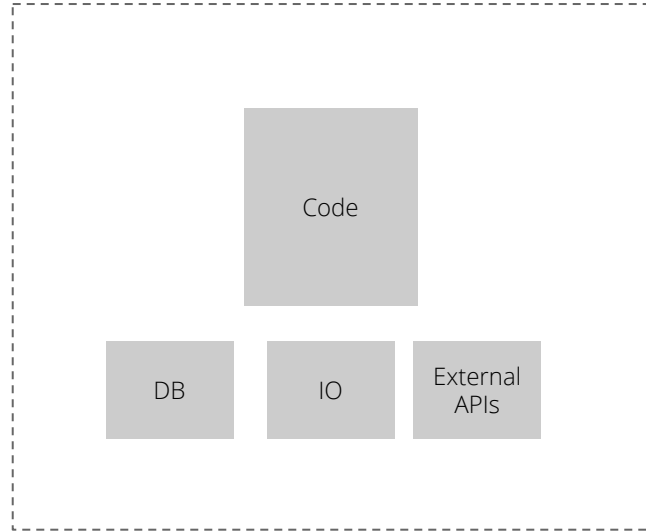
E2E tests

Integration between components

Critical Path

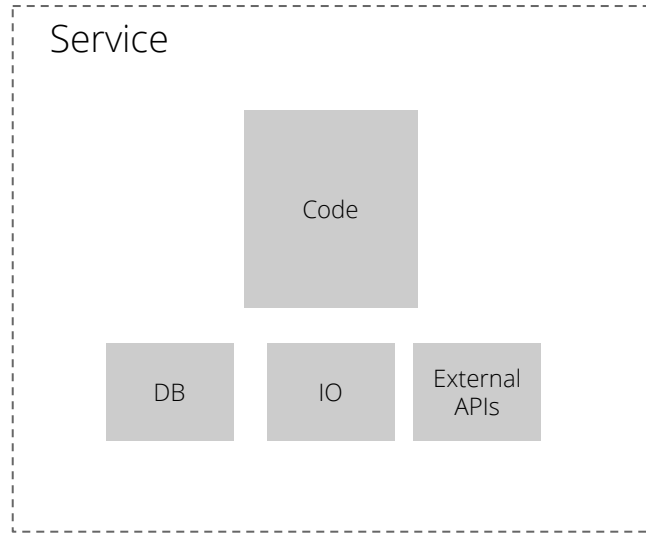
Configuration

What is a Component?



A microservice?

What is a Component? A service



CartService

OrderService

AddToCartService

PostReviewService

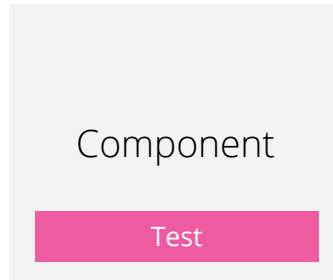
Domain Model

How to test? - Out of Process



Stub dependencies using
service virtualization

How to test? - In Process



Stub dependencies using
service virtualization

Or use in-memory stubs

SubcutaneousTest



Martin Fowler
14 February 2011

I use *subcutaneous test* to mean a test that operates just under the UI of an application.

Contrived Example

Charge Calculator

Service

Length (weeks)

Region

NA

EMEA

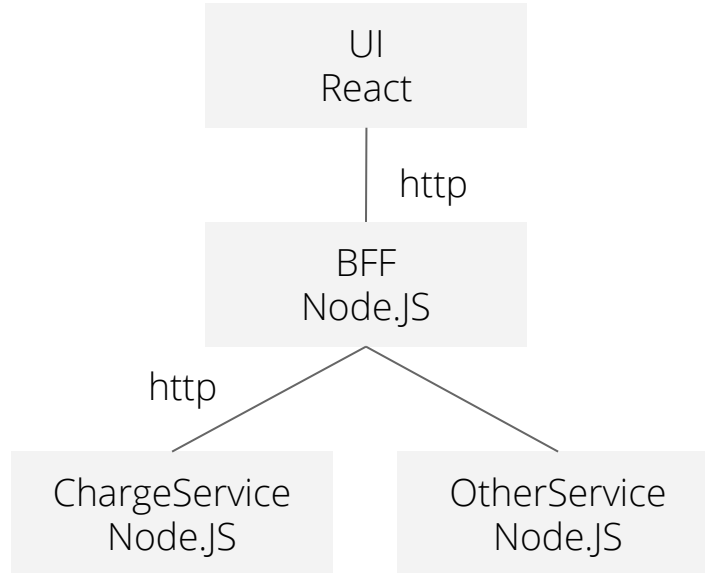
APAC

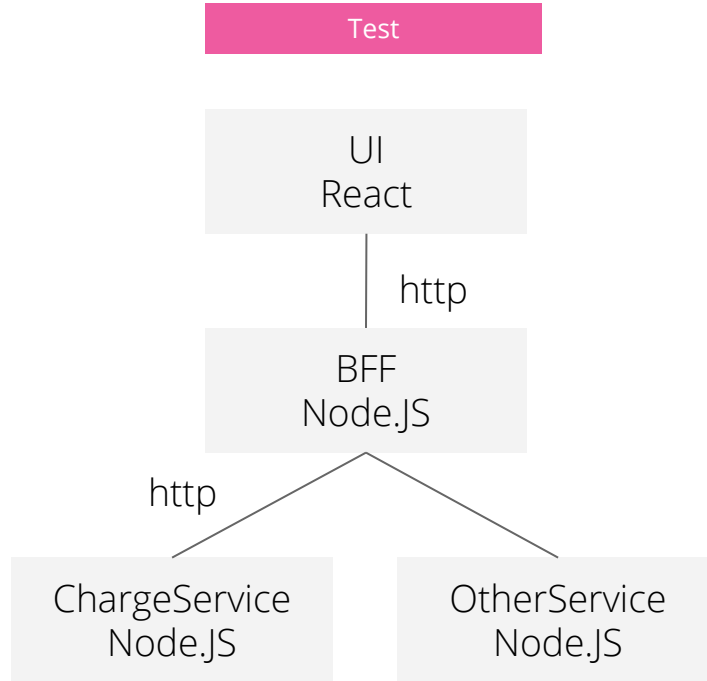
Country

City

The Charge for your service is: \$2,010.00

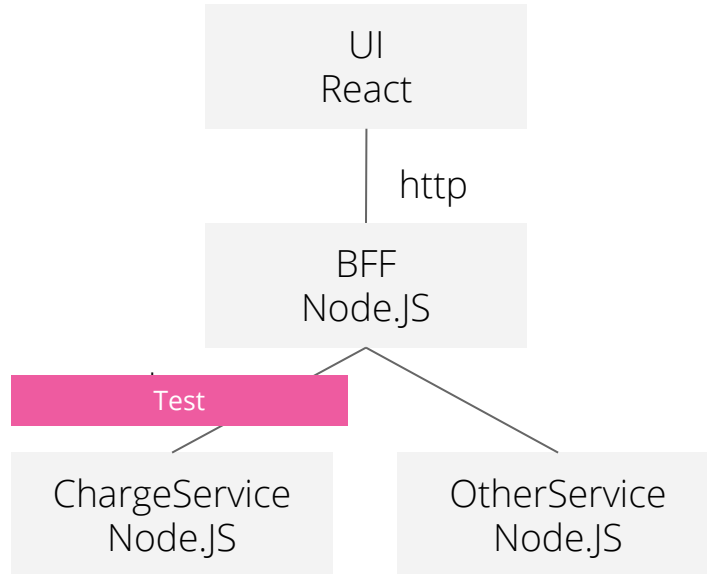
Contrived Example





```
21
22 describe('DefaultTest', () => {
23
24     const driver = new Builder()
25         .forBrowser('chrome')
26         .setChromeOptions(options)
27         .build();
28
29     it('Should calculate price for London and Service A', async () => {
30         await driver.get(url);
31         await driver.wait(until.elementLocated(By.css('form')), 1000);
32
33         // TODO change to select by visible text
34         await driver.findElement(By.css('#service > option:nth-child(1)')).click();
35         await driver.findElement(By.name('length')).sendKeys("12");
36         await driver.findElement(By.name('region-na')).click();
37         await driver.findElement(By.css('#country > option:nth-child(2)')).click();
38         await driver.findElement(By.css('#city > option:nth-child(2)')).click();
39         await driver.findElement(By.id('calculate-btn')).click();
40
41         const calculationResult = await driver.findElement(By.id("charge")).getText();
42         expect(calculationResult).toEqual("The Charge for your service is: $2,010.00");
43     }).timeout(5000);
44
45     after(async () => driver.quit());
46 });
```

~1000ms: headless, locally



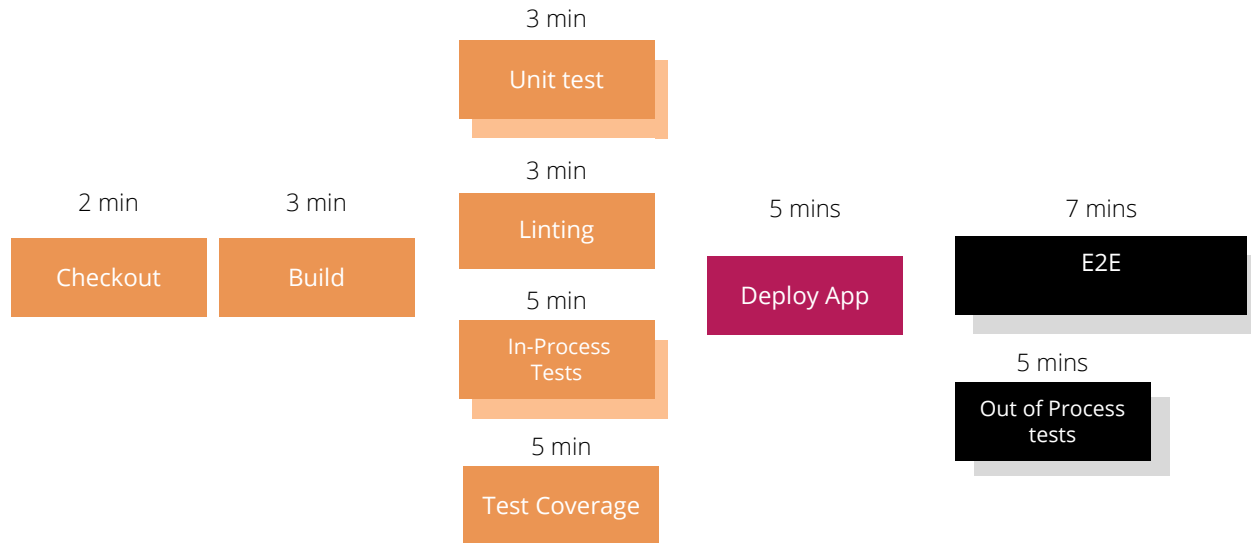
```
9
10   it('Should calculate price for Service A and London ', async () => {
11     |
12     |   chai.request(url)
13     |     .post('charge/new')
14     |     .send({
15     |       |   'service': 'serviceA',
16     |       |   'length': '12',
17     |       |   'city': 'london',
18     |     |   })
19     |     .end(function (err, res) {
20     |       |   expect(res).to.have.status(200);
21     |       |   expect(res.body).to.have.property('total').eql(3010)
22     |     |   });
23   });
24
25
```

~50ms: locally

How to test?

	Advantage	Disadvantage
In Process: Executed through Code	Fastest, Deterministic	Doesn't tests all the code (e.g. network, controller)
Out of Process: Executed through the API	Fast, Deterministic Forces API based ecosystem	Retests boiler plate code, network layer

Time for dev feedback - 22 mins



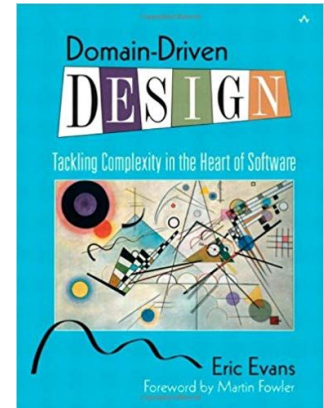
How to find entry points?

Understand what your application does, what capabilities does it provide?

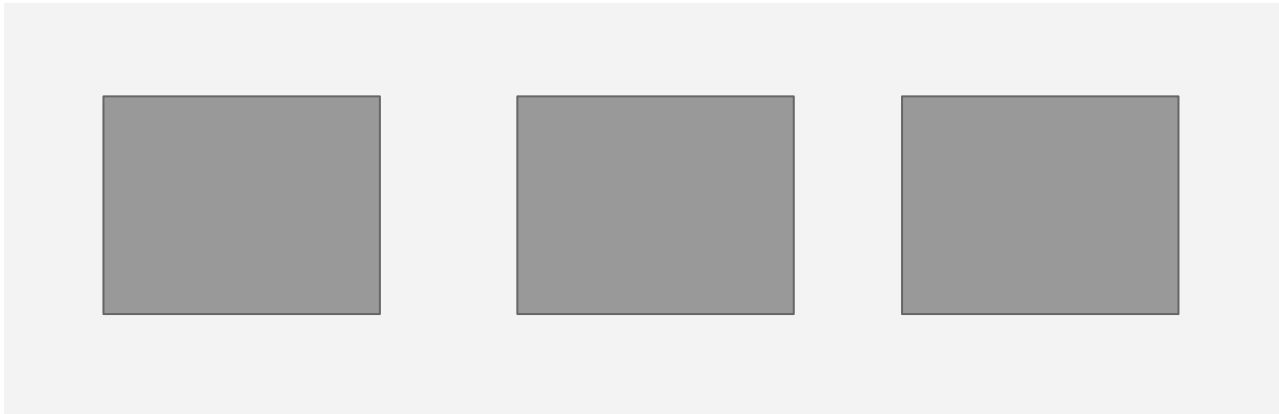
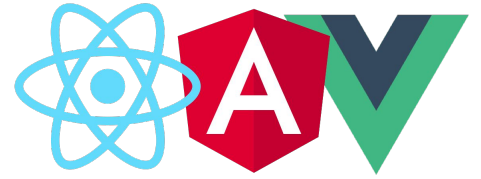
Where is the complexity in your domain?

Talk to your engineers and business

Domain Driven Design (DDD) techniques is good way, establish the domain events, bounded contexts, entities, behaviour, values.

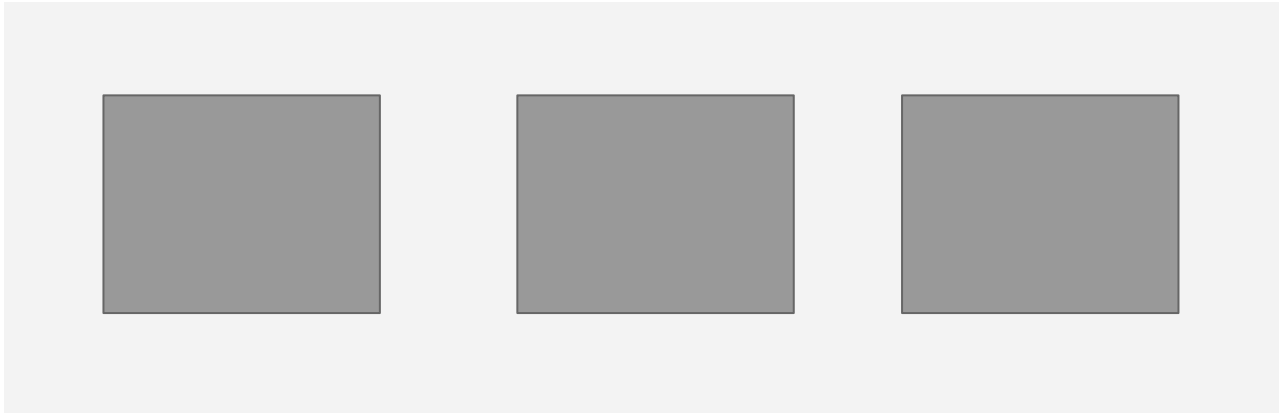
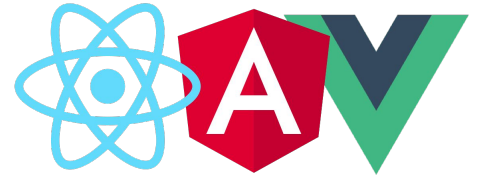


UI testing



API

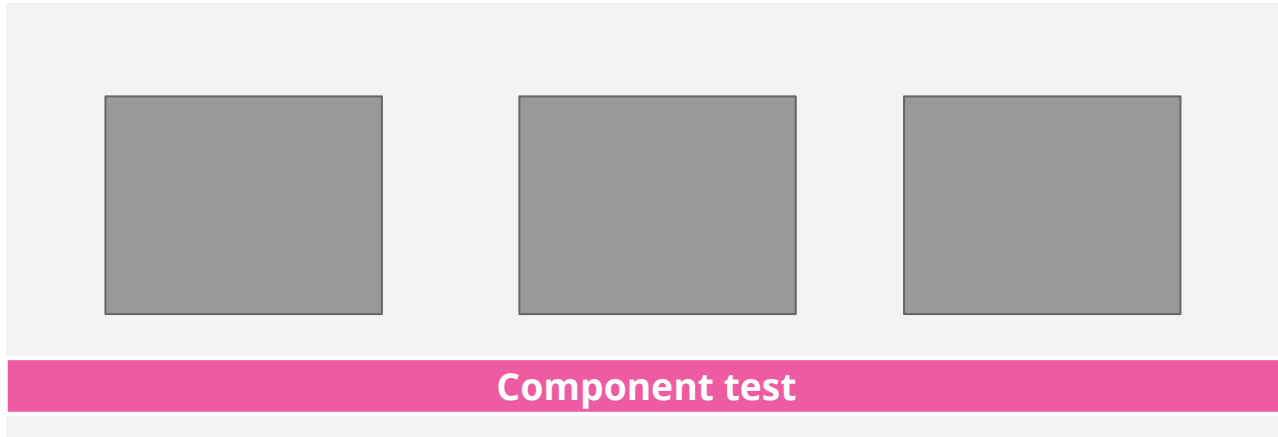
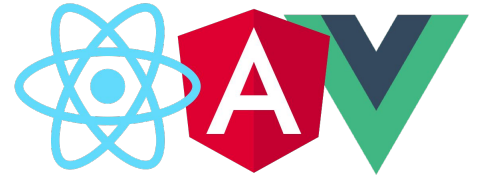
UI testing



Service virtualization



UI testing

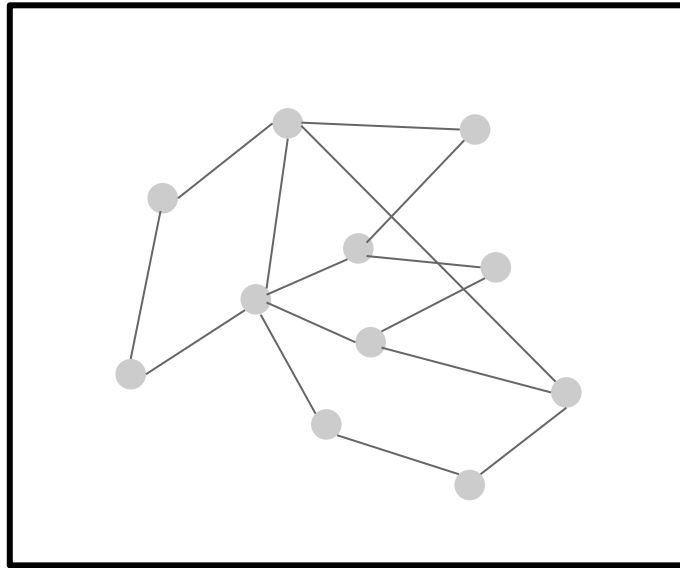


UI testing

<https://blog.pragmatists.com/genuine-guide-to-testing-react-redux-applications-6f3265c11f63>

```
1  it('should fetch and render a dog', async () => {
2    httpMock.onGet('https://dog.ceo/api/breeds/image/random').reply(200, {
3      status: 'success',
4      message: 'https://dog.ceo/api/img/someDog.jpg'
5    });
6
7    const wrapper = mount(<Provider store={store}><App /></Provider>);
8    wrapper.find('.dog-button').simulate('click');
9
10   await flushAllPromises();
11   wrapper.update();
12
13   expect(wrapper.find('img[src="https://dog.ceo/api/img/someDog.jpg"]').exists()).toBe(true);
14 });
15
```

E2E tests



Culture

Quality is everyone's responsibility

Cross functional teams

Deliberately design your application to be testable,

Think about testing from day one



ThoughtWorks®

Thank you

tcochran@thoughtworks.com