


Testability – lessons learnt the hard way

Quality Meetup 14.10.2020

Tomasz Wierzchowski

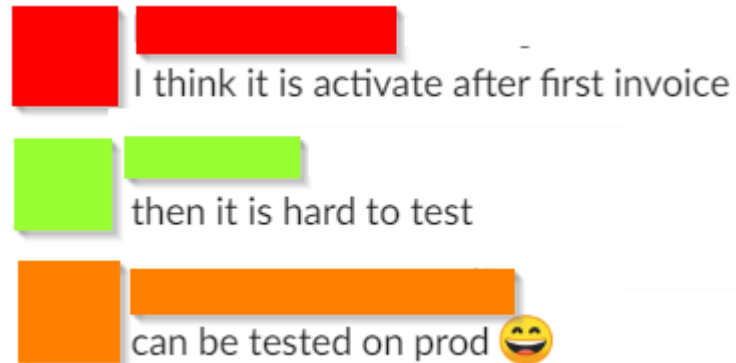
About me

- 10+ years in QA
- QA Lead @ [Northmill Bank](#)
- Program committee @ [.Quality Excites](#)
-  [/tomasz-wierzchowski](#)



When you hear:

- We need more time for tests (just before the deadline)
- We have legacy system, thus it cannot be tested
- This feature can only be 'tested' on production



When you hear:

- Our system is special one (special domain / business / technology / etc.), thus it cannot be tested
- There is too many bugs on production

Then you probably have issue with testability

Testability? What?

Theory - ISTQB / ISO 25010

The degree of effectiveness and efficiency with which tests can be designed and executed for a component or system.

In practise

How fast and easy can we test?

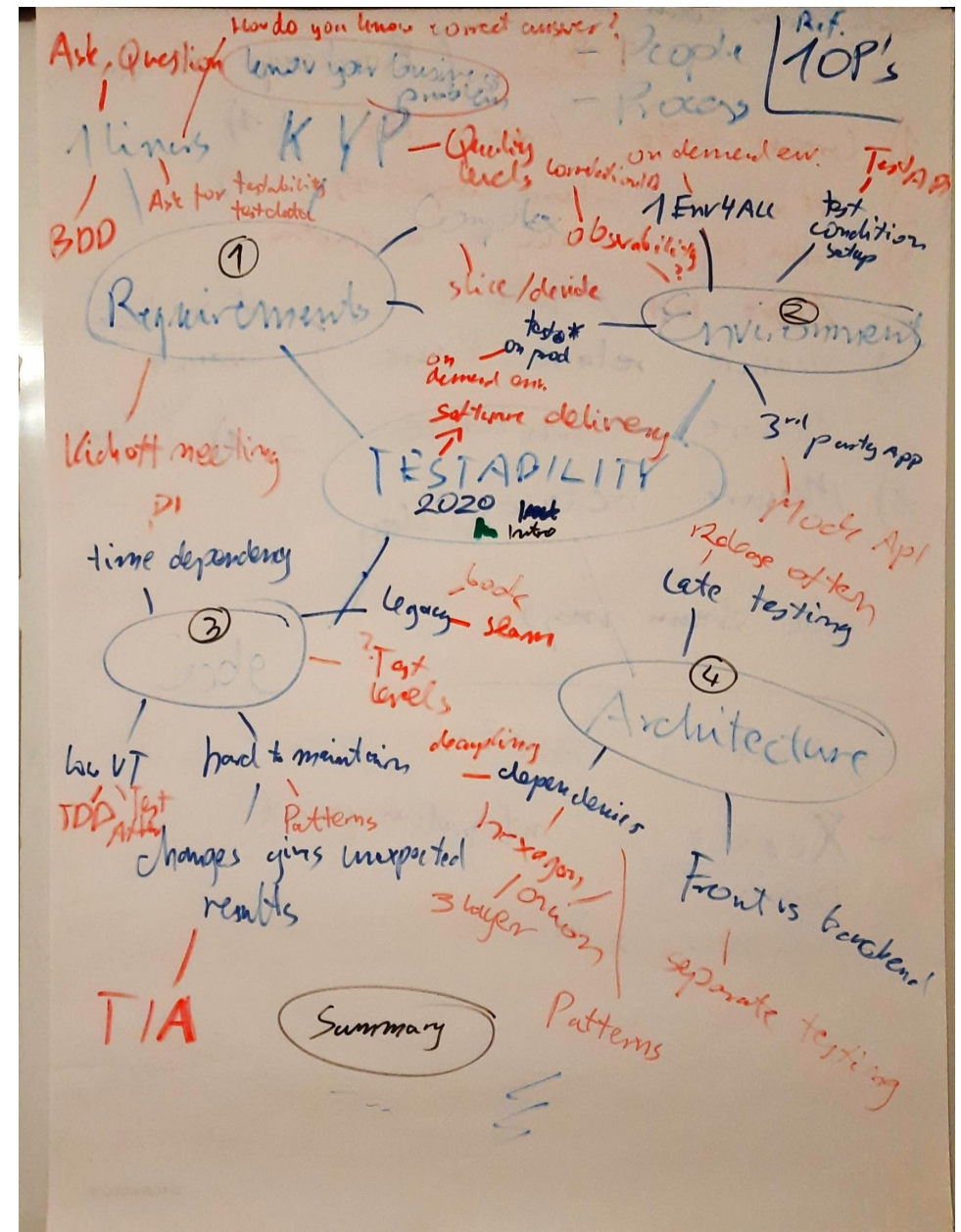
Why should you care?

- Speed
- Efficiency
- Control

| Time is money

Challenges:

- Requirements
- Environments
- Code
- Architecture



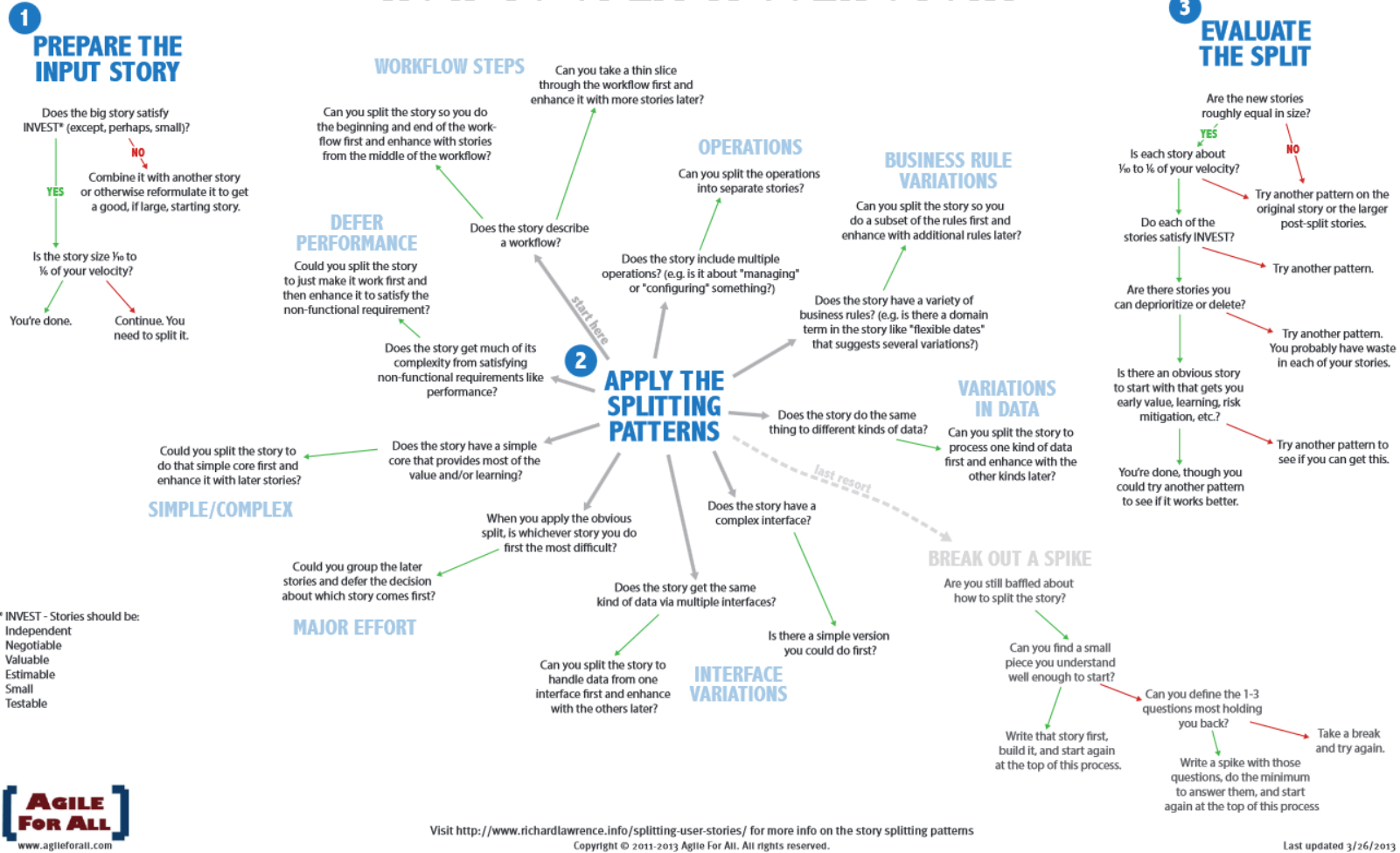
First challenge (requirements)

- One liner requirement
 - Do the same system as in X, but for Y country
 - Make sure the actual calculations works as in project A
- Team doesn't understand the business problem
- Complex/complicated systems are hard to test

What can help?

- Ask & question about the domain and possible solutions
- Do small features (Decomposition & INVEST!)

HOW TO SPLIT A USER STORY



What can help?

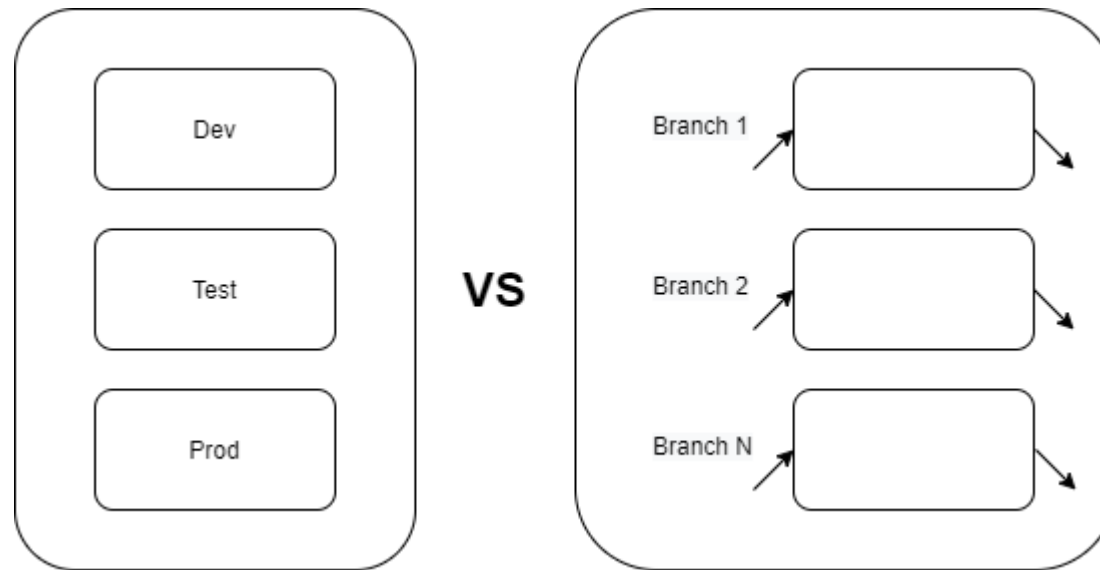
- Prepare test data during feature planning
- Think of BDD / ATDD / Specification By Example

Second challenge (environments)

- Everyone is using the same environment
- Hard to setup system in required test condition
- 3rd party apps
- Tests on production*

What can help?

- Script creation of environments (Infrastructure as a code)
- Make environments on demand



What can help?

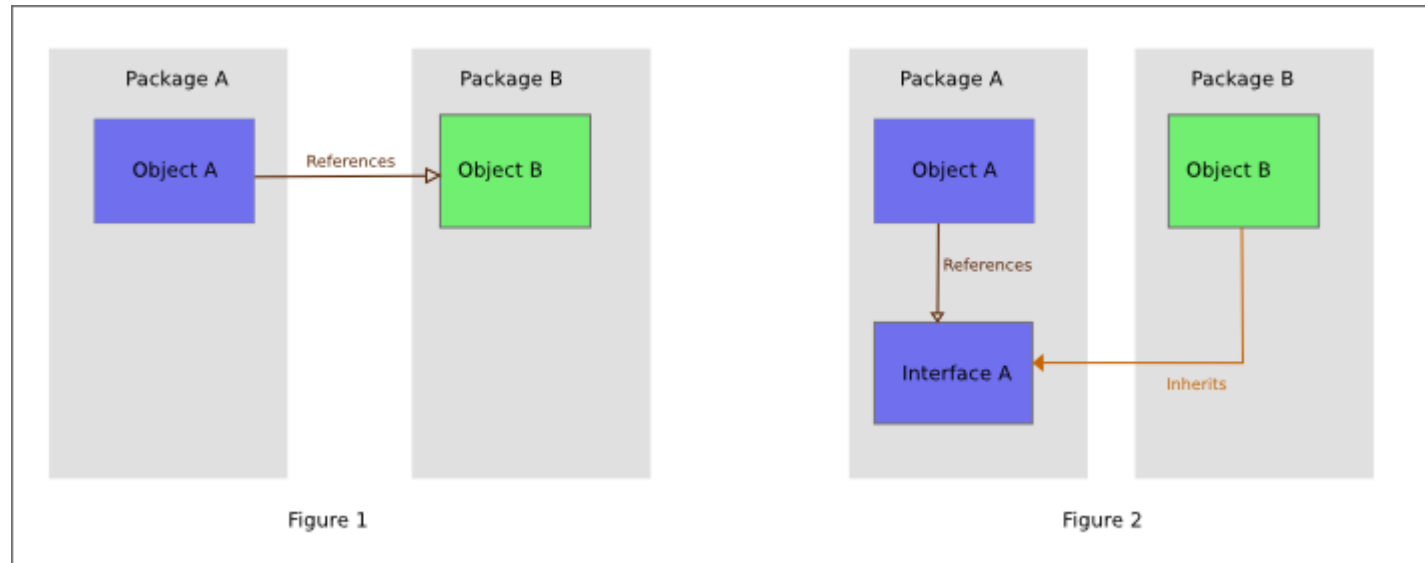
- Fake services
- Use dedicated API for testing purposes
- Log & monitor

Third challenge (code)

- Changing code in one area breaks functionality in another
- Team doesn't have confidence in changing code
- 'Hard' to write unit test -> Low level of unit tests -> low code coverage
- Testing time-dependent features
- Legacy code

What can help?

- Use SOLID rules (dependency inversion)



https://en.wikipedia.org/wiki/Dependency_inversion_principle

What can help?

- Don't mix objects creation with business logic

```
(...)  
public offer calculateOffer()  
{  
    var checker = new scoreCalculator(); // wrong !  
    var score = checker.calculate();  
  
    if (score >= 90)  
        MaxLoanAmount = 5000;  
  
    if (score >= 70)  
        (...)  
}
```

What can help?

- Don't mix objects creation with business logic

```
(...)  
public offer calculateOffer(IScoreCalculator scoreCalculator)  
{  
    var score = scoreCalculator.calculate(); // Better  
  
    if (score >= 90)  
        MaxLoanAmount = 5000;  
  
    if (score >= 70)  
        (...)  
}
```

What can help?

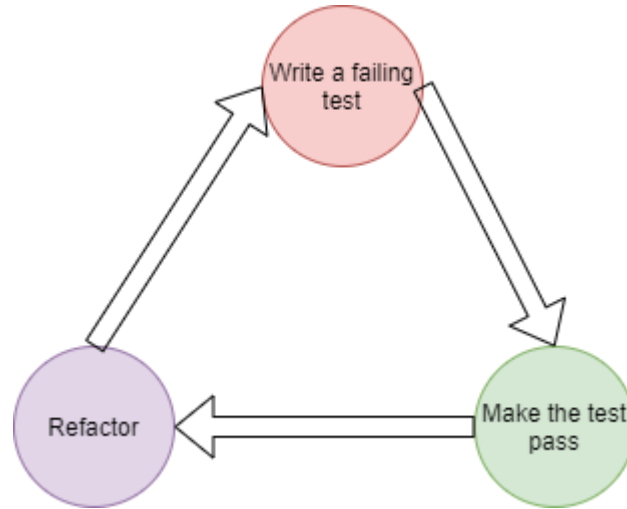
- Make abstraction for time, and all external/system dependencies

```
public interface IDateTimeService
{
    DateTime Now {get;}
}
```

```
public class FakeDateTimeService : IDateTimeService
{
    public DateTime Now {get; set;}
}
(...)
var longTimeAgo = new FakeDateTimeService {
    Now = new DateTime(2019,10,14) }
```

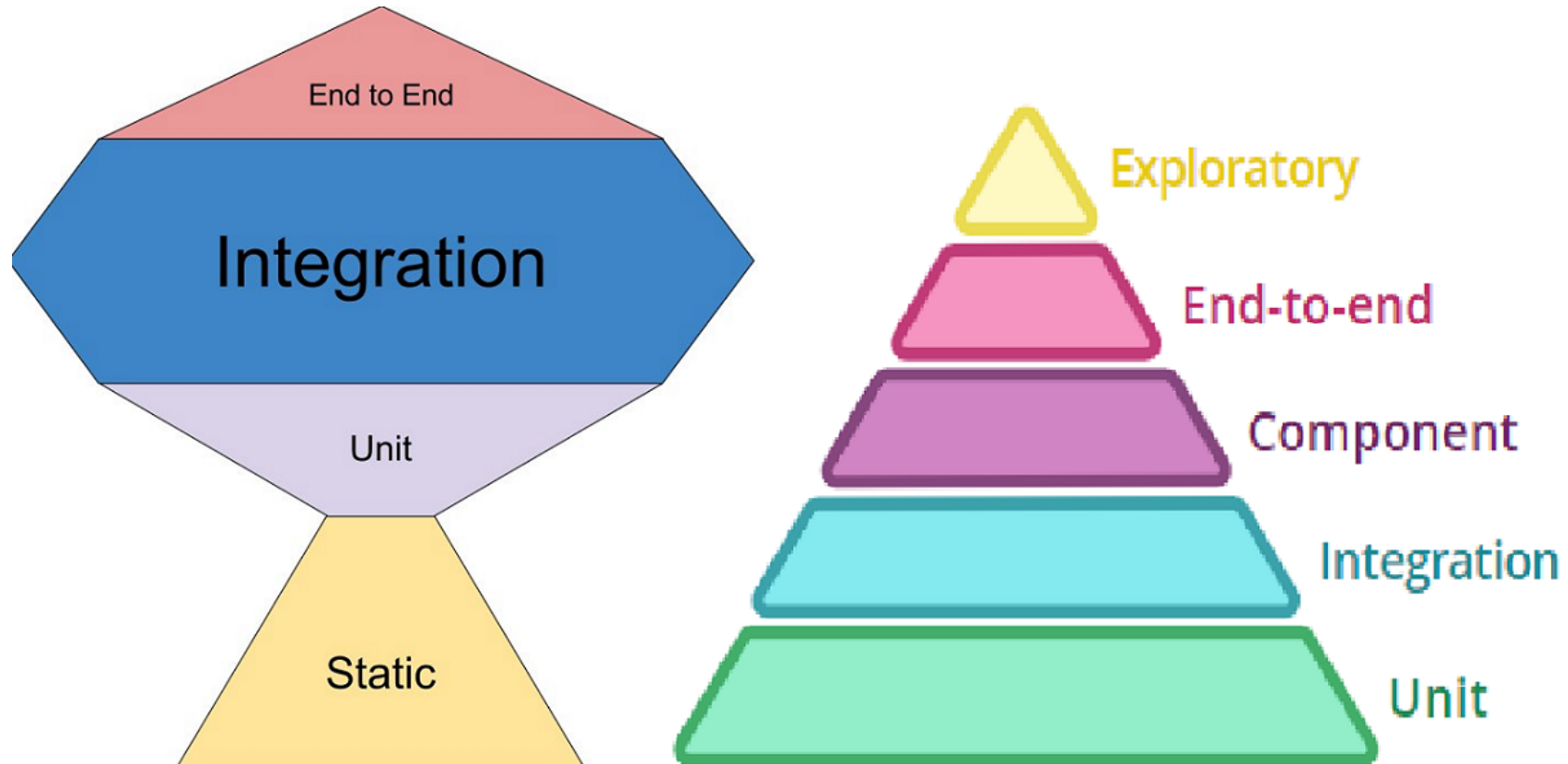
What can help?

- TDD



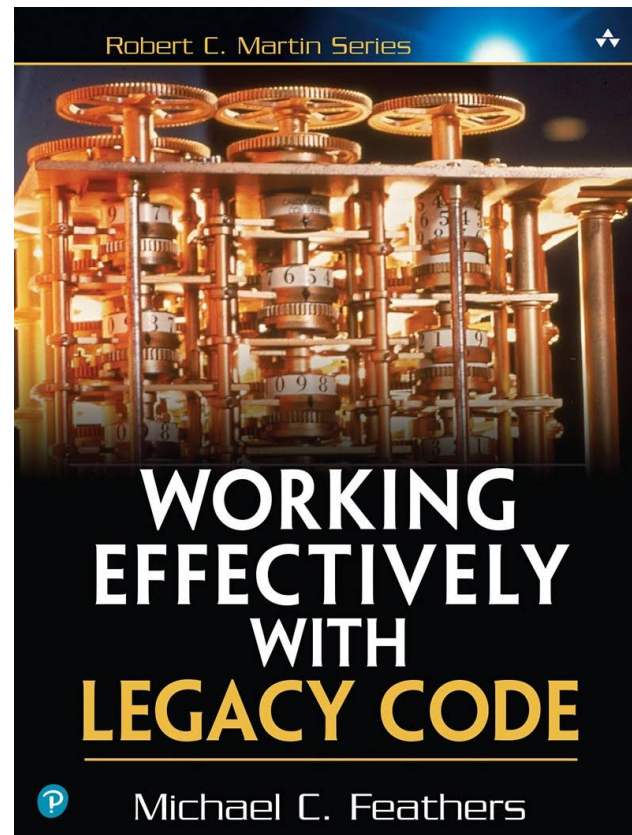
What can help?

- Know & use your test levels



What can help?

- Legacy is not an excuse – read book "Working effectively with legacy code" Michael Feathers (*published in 2004!*)

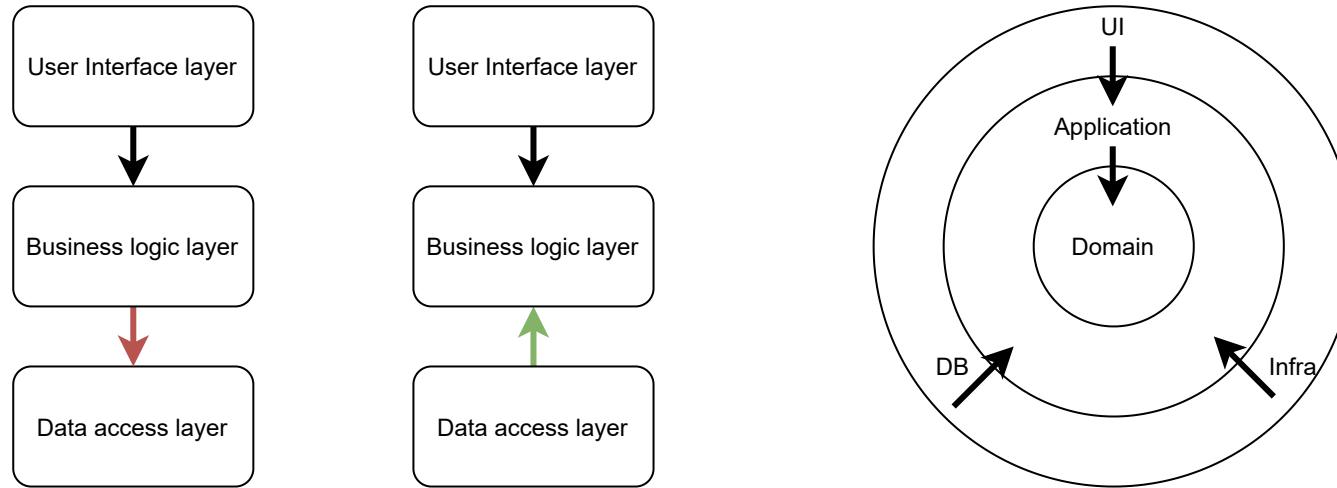


Fourth challenge (architecture)

- To test component A you have to run components B & C (and a few more)
- Cannot test front-end without testing whole application
- Tests can be done only at the end of the project schedule

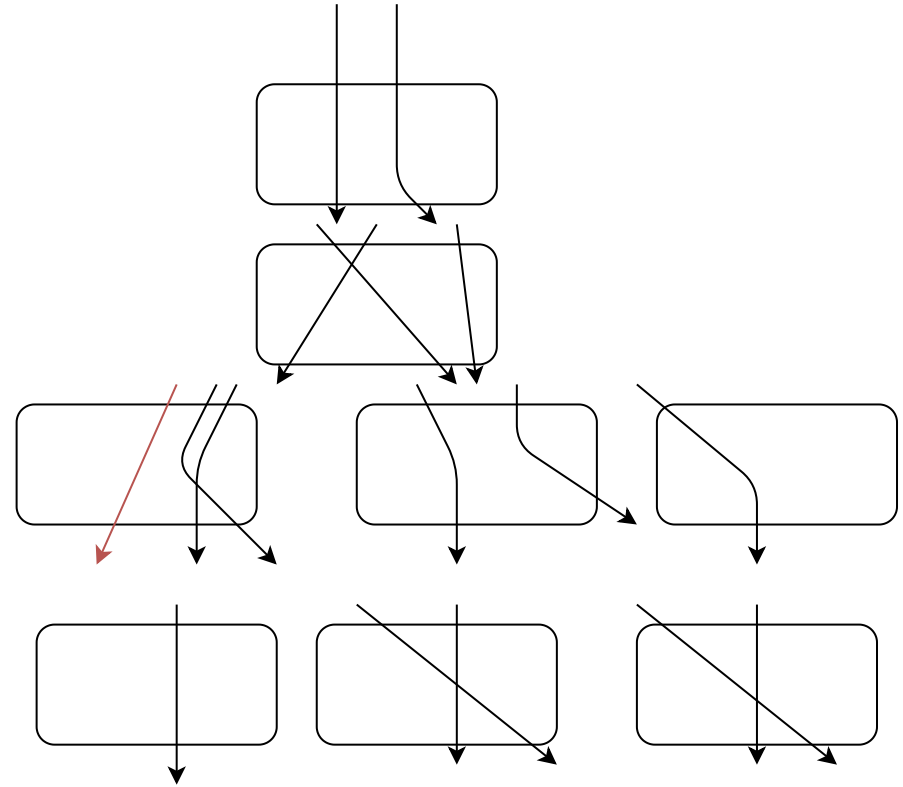
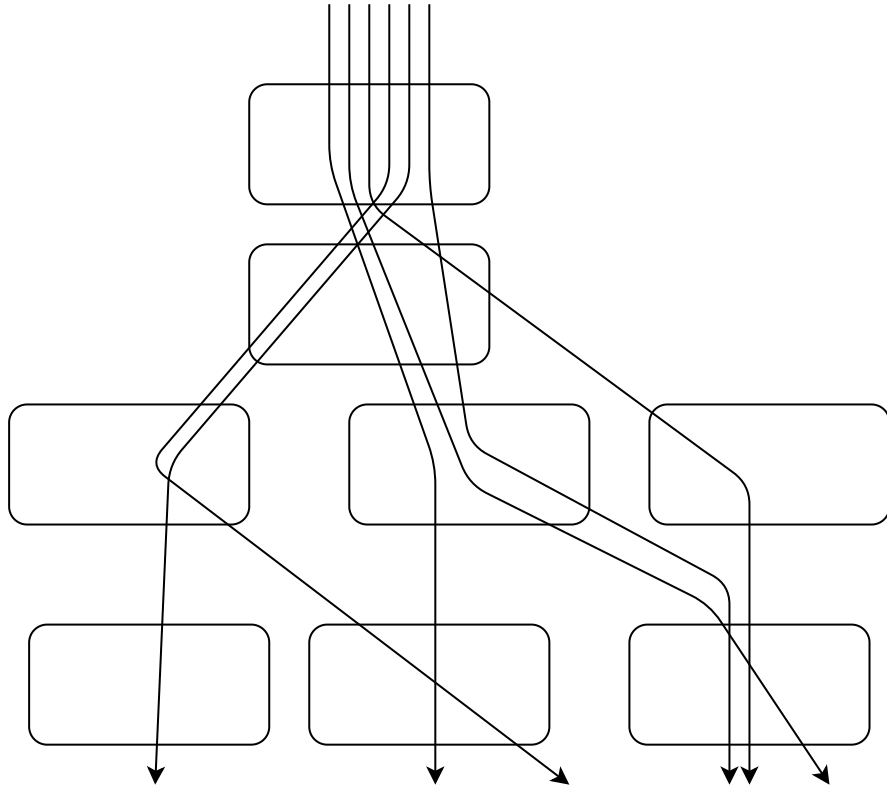
What can help?

- Keep business rules decoupled from UI / Database / Servers / etc.



What can help?

- Test components separately [#integratedtestsareasm](#)



What can help?

- Release often
- Cooperate with developers/architects

What to do when you're back in project?

- Know your domain & gain experience
- Question requirements status quo
- Control application and environments
- Enhance clean code & clean architecture
- Promote testability on all levels

The image features a vibrant red background with a series of concentric circles in varying shades of red and dark red, creating a tunnel-like effect. In the center, a dark blue circle serves as a focal point. Overlaid on this central circle is the text "That's all Folks!" in a white, elegant script font. The text is slightly tilted and has a subtle drop shadow, giving it a three-dimensional appearance as if it's floating above the circles.

That's all Folks!

Additional materials:

- [User Stories Applied](#)
- <https://www.satisfice.com/download/heuristics-of-software-testability>
- <https://www.slideshare.net/AshWinter/10-ps-of-testability>
- https://www.slideshare.net/AshWinter/architectural-testability-workshop-for-test-academy-barcelona?next_slideshow=1
- [Aslak Hellesøy - Testable Architectures](#)
- <https://understandlegacycode.com/>
- <https://martinfowler.com/articles/microservice-testing/>
- <http://misko.hevery.com/attachments/Guide-Writing%20Testable%20Code.pdf>
- <https://agileforall.com/resources/how-to-split-a-user-story/>
- <https://blog.thecodewhisperer.com/permalink/integrated-tests-are-a-scam>

Questions?

We are looking for

QA Engineer

Microsoft®
.NET

JS

aws

 Katowice

 10 000 - 15 000 PLN (gross)

northmill
bank