




Quality is not (only) testing!

BEER.EX 2021

6th October, 2021

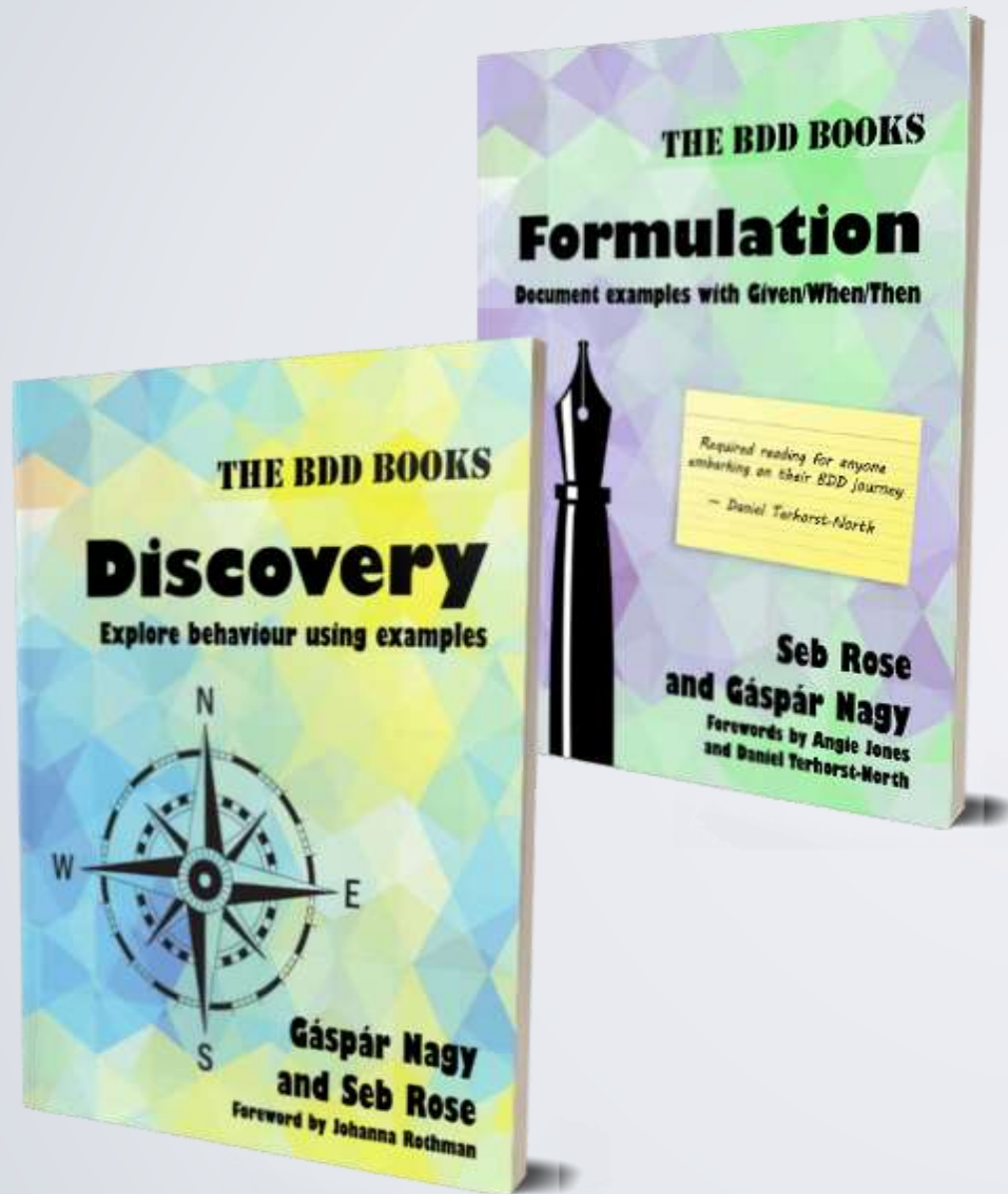
Gáspár Nagy

coach • trainer • bdd addict • creator of specflow
<http://bddbooks.com> • “Discovery” • “Formulation”
@gasparnagy • gaspar@specsolutions.eu

A large audience is seated at long wooden tables in a modern, high-ceilinged hall. The audience is facing a stage area where a presentation screen is visible. The screen displays a slide with a green logo and text. The hall has a high, industrial-style ceiling with exposed beams and lighting fixtures. The overall atmosphere is professional and focused.

**Its fantastic to be back on stage
after...**

606 days 22 hours 25 mins



Gáspár Nagy

coach, trainer and bdd addict
creator of SpecFlow


gaspar@specsolutions.eu

<https://specsolutions.eu>

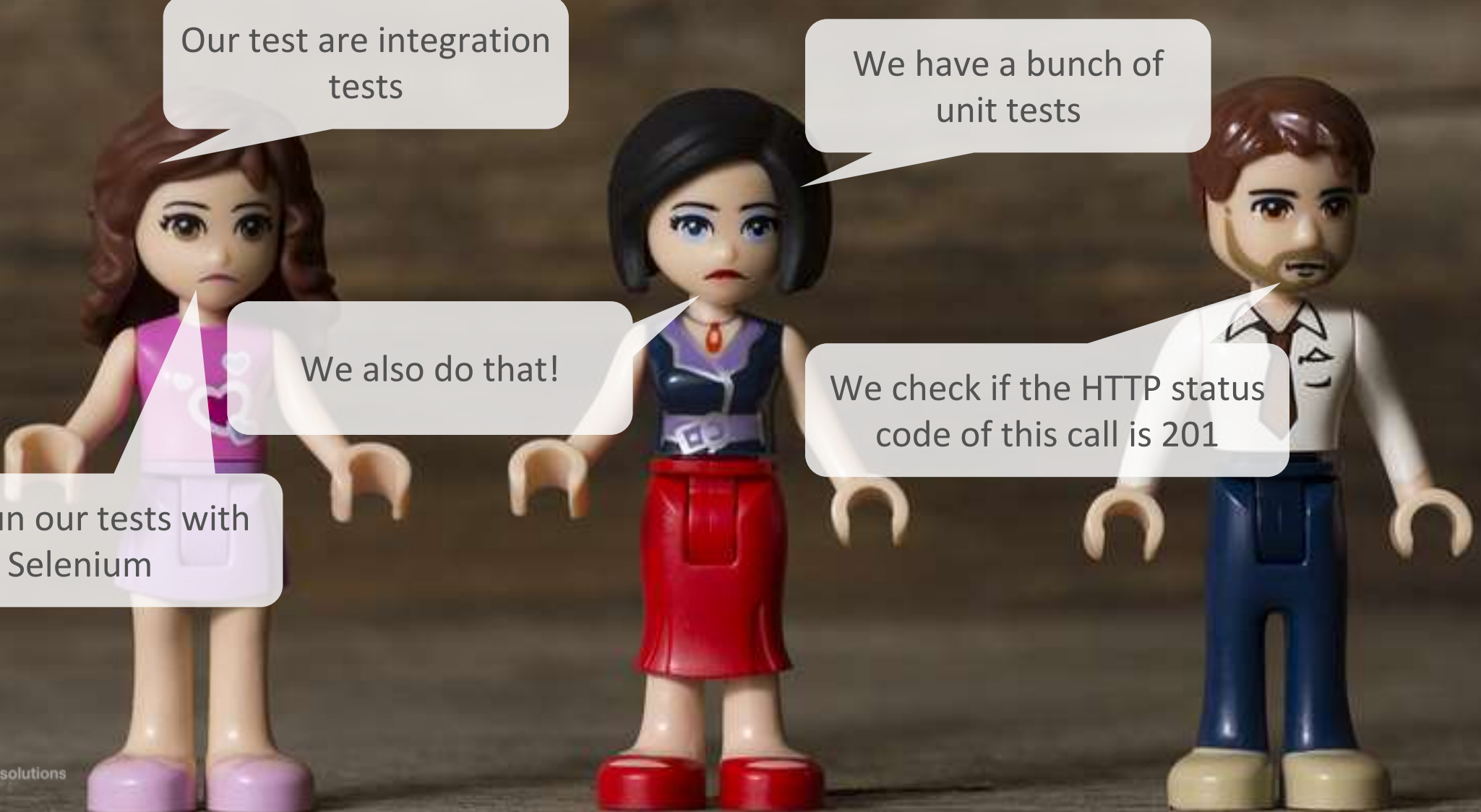
@gasparnagy

<http://bddbooks.com>

*Quality is not (only)
testing!*

A photograph of two dogs, one with blonde fur and one with dark fur, both bowing their heads and front paws down towards the floor. The blonde dog is on the left, and the dark dog is on the right. They are both wearing collars. The background is dark and out of focus.

We're not worthy



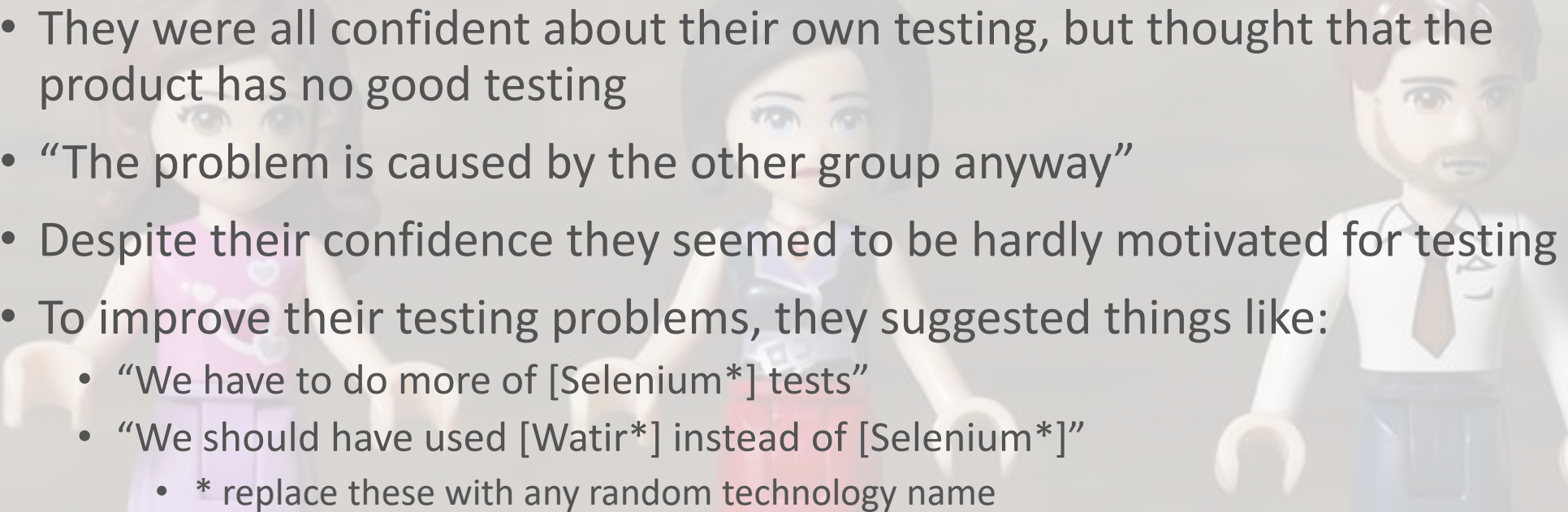
Our test are integration tests

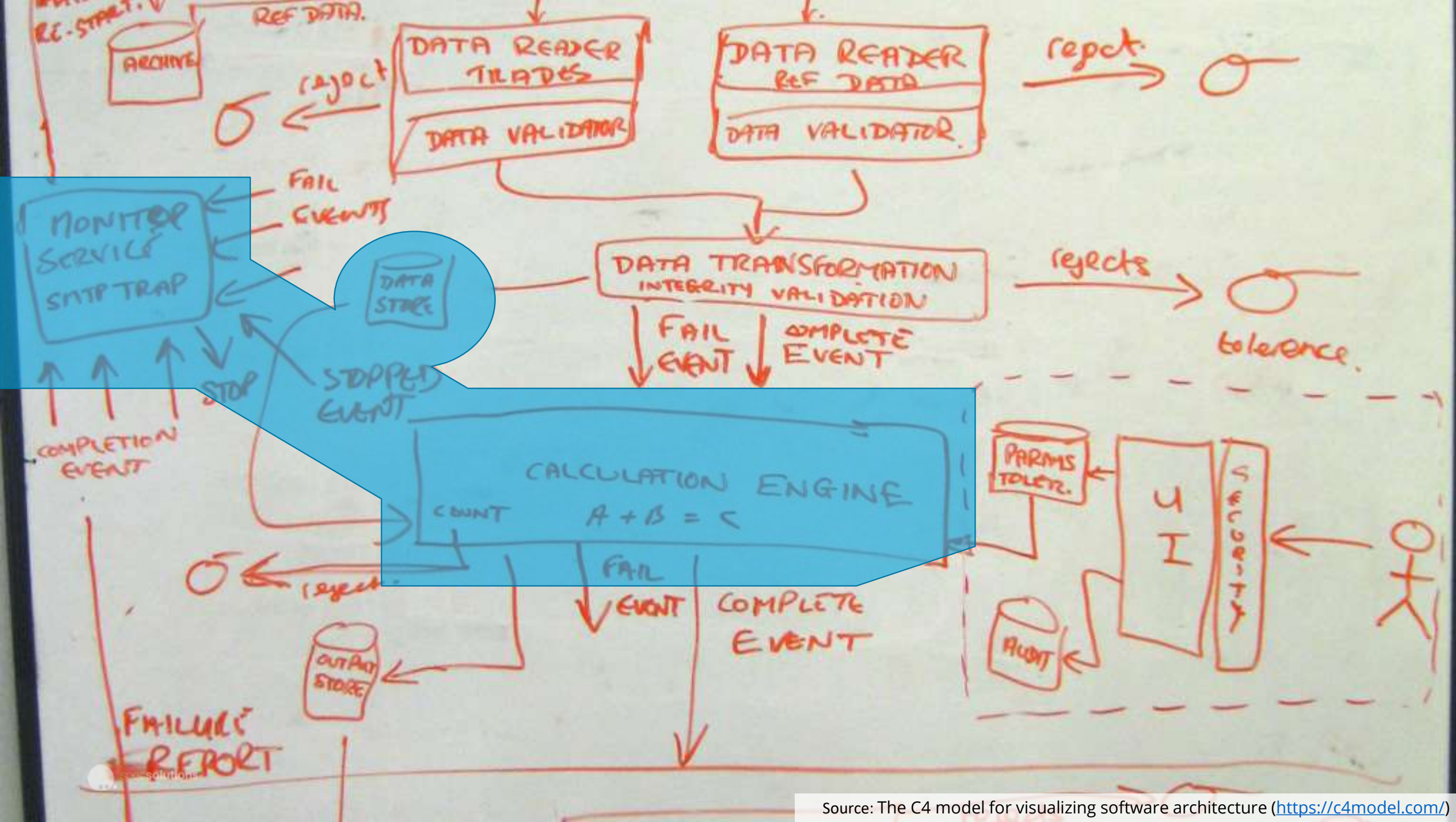
We have a bunch of unit tests

We also do that!

We run our tests with Selenium

We check if the HTTP status code of this call is 201

- 
- Three LEGO minifigures are standing behind a semi-transparent white text box. On the left is a female minifigure with brown hair, wearing a purple dress with a heart pattern and purple shoes. In the center is another female minifigure with black hair, wearing a pink dress and red shoes. On the right is a male minifigure with brown hair and a beard, wearing a white shirt, a brown tie, blue pants, and tan shoes.
- They were all confident about their own testing, but thought that the product has no good testing
 - “The problem is caused by the other group anyway”
 - Despite their confidence they seemed to be hardly motivated for testing
 - To improve their testing problems, they suggested things like:
 - “We have to do more of [Selenium*] tests”
 - “We should have used [Watir*] instead of [Selenium*]”
 - * replace these with any random technology name



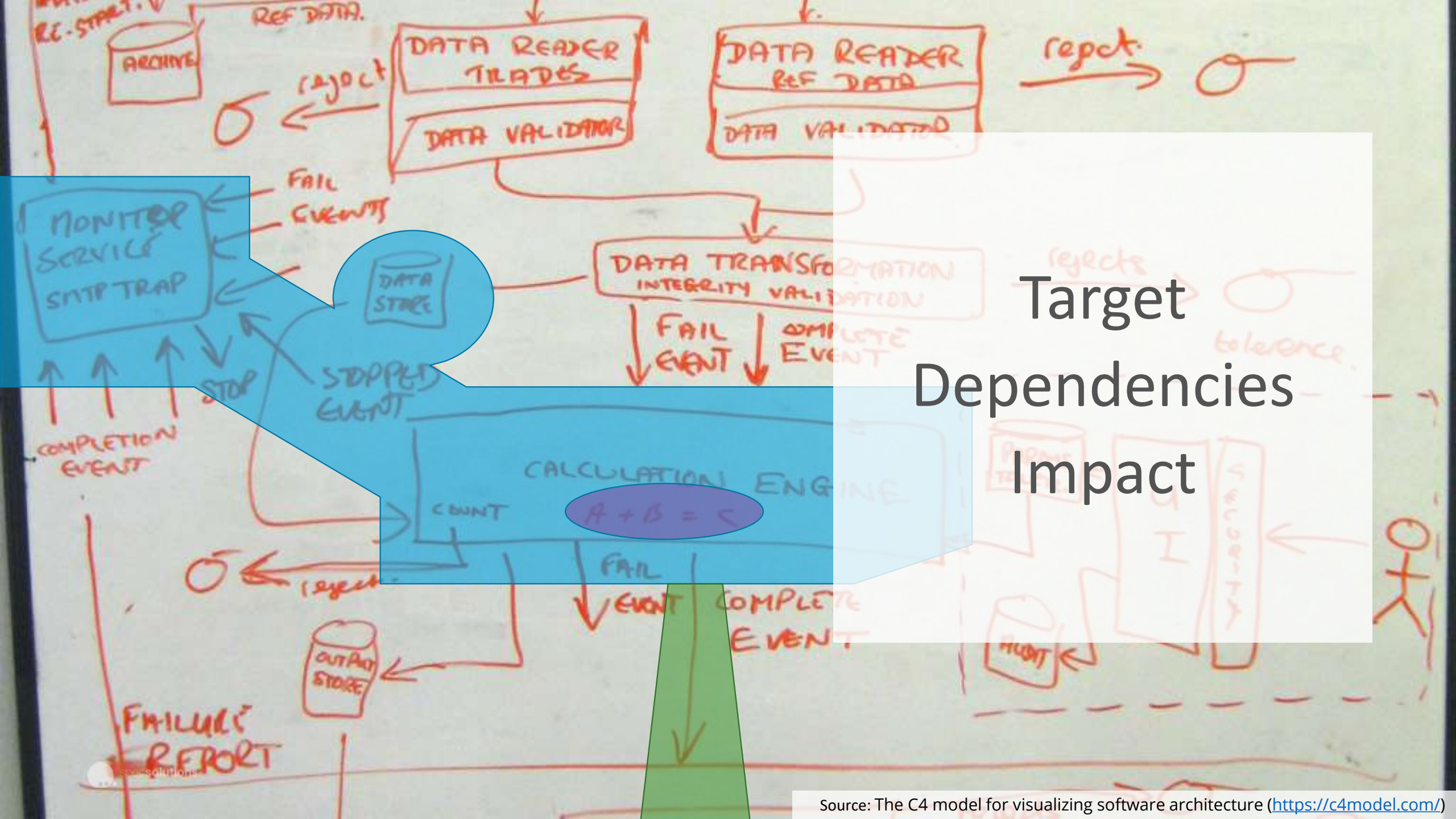
The measured coverage of the test
is not the same as the test target

Tests “exercise” much more code
than what it actually tests

What do you
test?

What does your
test focus on?

Target Dependencies Impact



PO

The system does not work
and we are already late!

The users often not
able to make doctor's
appointments

The registered users can go to the site,
select the doctor, choose an available
time slot in the calendar and finalize
their selection. As a result, they should
get a confirmation email with the
dates and the address.

The registered users can go to the site, select the doctor, choose an available time slot in the calendar and finalize their selection. As a result, they should get a confirmation email with the dates and the address.

Where is this tested?

Why do we check if the HTTP request returns with 201?

To verify compliance with the company standards

What has happened?

- A bunch of technologist have been assigned to testing – they produced a bunch of technology
- They were not collaborating, because they missed the common goal – the purpose
- They debated on technology, but what we need is quality

You can lead the horse to water
but you can't make him use
your favorite text editor...



The common goal is: Quality

How do you
test?

What do you
test?

“I get paid for code that works, not for tests”
(Kent Beck)

Quality is about confidence

Defining Testing

"to increase confidence
for stakeholders
through evidence"

@tastapod

Breaking down quality: quality aspects (sample)

Functional

- Works as expected
- Expectations are good
- Expectations are documented

Operational

- Secure
- Fast
- Convenient
- Pretty
- Consistent
- Predictable

Strategic

- Maintainable
- Architecture
- Code quality
- Easy to integrate
- Flexible

Understanding and documenting functional expectations

The registered users can go to the site, select the doctor, choose an available time slot in the calendar and finalize their selection. As a result, they should get a confirmation email with the dates and the address.

Scenario: An email confirmation is sent about a successful appointment booking

Given the user is registered

And they have initiated a booking process with

doctor	date	time range	
Dr. Who	10/6/2021	11:00-11:30	

When they finalize the booking

Then they should receive a confirmation email with the selected details

Quality expectations are always project specific

Functional

- Can be used by doctors
- Can be used by patients
- **Can provide informational content for anonymous users**
- Can be used for invoicing
- Provides public API for 3rd party integrators (B2B)
- **Handles time zones**
- Compliant with country-specific regulations

Operational

- **Can be used on small screens (mobile)**
- **Multi-language**
- Protects sensitive data of patients
- GDPR compliant
- Provides fast response time on mobile networks

Strategic

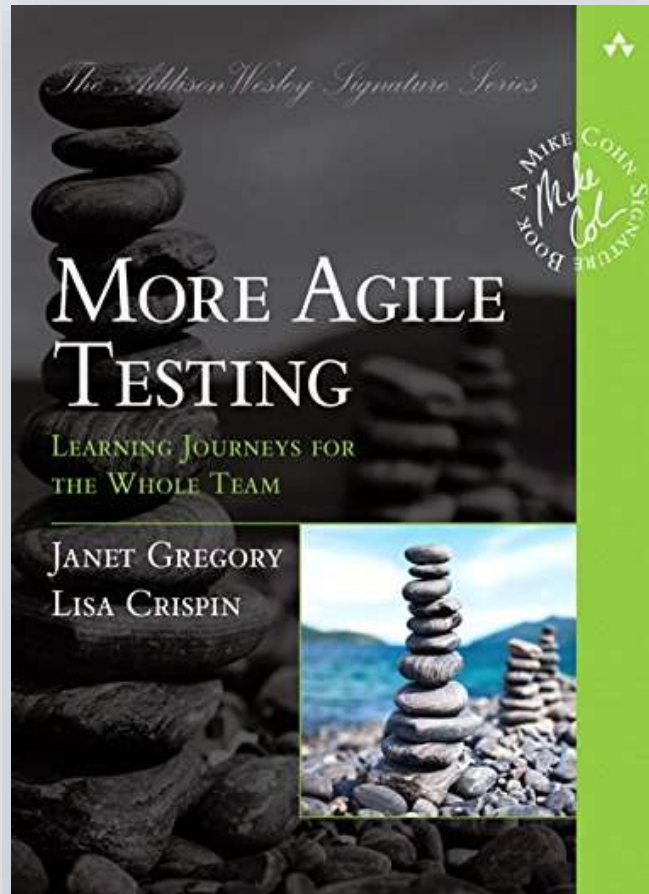
- Maintainable
- **Easy to handle changes of regulations**
- Easy to maintain long-term
- Can handle multiple supported version (hotfixes, etc.)
- **Scalable**

How to find the right test method for a particular purpose

- Many quality aspect can be “tested” (ensured) in multiple ways
- In many cases these might span across your boundary of expertise
- Finding the right test methods should be done collaboratively
- There are useful assets that can help you
 - System/component diagram
 - Test automation pyramid
 - Testing quadrant
- At the end this will become your testing strategy!

Testing Strategies

More Agile Testing: One-page test plan



Test Plan for Fitness Application Feature Adding BMI Calculator

Testing Scope

- A BMI (Body Mass Index) Calculator is being..... Initial testing will be performed using an iPhone, iPad mini, and a Nexus tablet.

Testing Approach

- The BMI calculator functionality will be tested based upon identifying the appropriate combinations and boundaries based upon the BMI requirements. Exploratory testing will be performed based upon how the users typically use the Fitness Application and anticipated usage of the new calculator. For this phase, testing is limited to functionality, integration, exploratory, and regression. Device specific testing such as low-battery or other device conditions is not part of this testing.

Risks

- Look and feel across devices may not be consistent. Additional devices may need to be tested beyond the initial sample.
- **Features Not to Test**
 - Waist to Hip Ratio Calculator
 - Weight Goal Tracker and Report
 - Food Tracking Plan
 - Settings Options

The four-page quality strategy

- A quality strategy is like an overall test plan – gives a guidance for the **whole team** about how we address quality
- Custom & specific to your project, but keep it short
- A strategy might contain the following sections:
 - Challenges (Risks)
 - Quality Aspects
 - Testing Methods
 - Feedback channels

Challenges (Risks) – sample

- This is a product that is used by stressful people
 - Doctor is always in rush
 - Patient is in trouble (hence seeking for doctor)
- Supports specialties of 4 countries and 3 languages
- People use it from 2 different time zones
- Having a doctor's appointment is very important for the patients
 - The application should be very clear if the booking really happened
 - There should be an alternative way for booking if the system is down
- Regulatory audits have to be served without down-time

Quality aspects – sample

Functional

- Can be used by doctors
- Can be used by patients
- Can provide informational content for anonymous users
- Can be used for invoicing
- Provides public API for 3rd party integrators (B2B)
- Handles time zones
- Compliant with country-specific regulations

Operational

- Can be used on small screens (mobile)
- Multi-language
- Protects sensitive data of patients
- GDPR compliant
- Provides fast response time on mobile networks

Strategic

- Maintainable
- Easy to handle changes of regulations
- Easy to maintain long-term
- Can handle multiple supported version (hotfixes, etc.)
- Scalable

Testing methods & Feedback channels

TESTING METHODS

AUTOMATED BDD SCENARIOS (SPEC SYNC.AZURE DEV OPS.SPECS)

- Purpose:
 - Test and document features & settings
 - Verify compatibility with different ADO versions
 - Document features not supported in specific ADO versions
 - Provide quick feedback on functionality using ADO Stubs
 - Test & document supported BDD framework test result formats
 - Provide guidance for configuring the different SpecSync features
- Automation strategy:
 - Automates SpecSync through the Synchronization
 - Integration tests with configurable ADO target (ADO Stub)
 - The same tests can run on for different targets.
 - The target can be specified in the configuration
 - Reconfigures the ADO interfaces used
 - ADO related context/outcome verification
 - When a scenario is not supported on a target, the test result is marked as `@notsupported-tfs2017`
 - Uses a stub local project, but real feature files
- Structure:
 - The structure should follow the structure of the project
 - Customizations should be placed in the main stub
 - Features & scenarios that explain the behaviour and marked with `@infrastructure`
 - Files that should be excluded from the public documentation (`!_licensing.feature`)
 - For specifying non-TRX test results, a full test result file can verify if their file follows the same structure
- **Do not include here** tests that verify:
 - the BDD test result file format variations (e.g., SpecSync Algorithmic Logic Tests)
 - scenario edge cases (e.g., special characters in scenario names)
 - configuration variations (e.g., config file vs command line)
 - ADO edge cases (e.g., Test Run with more than one ADO connection or authentication options)
 - configuration and user errors -> Exploratory

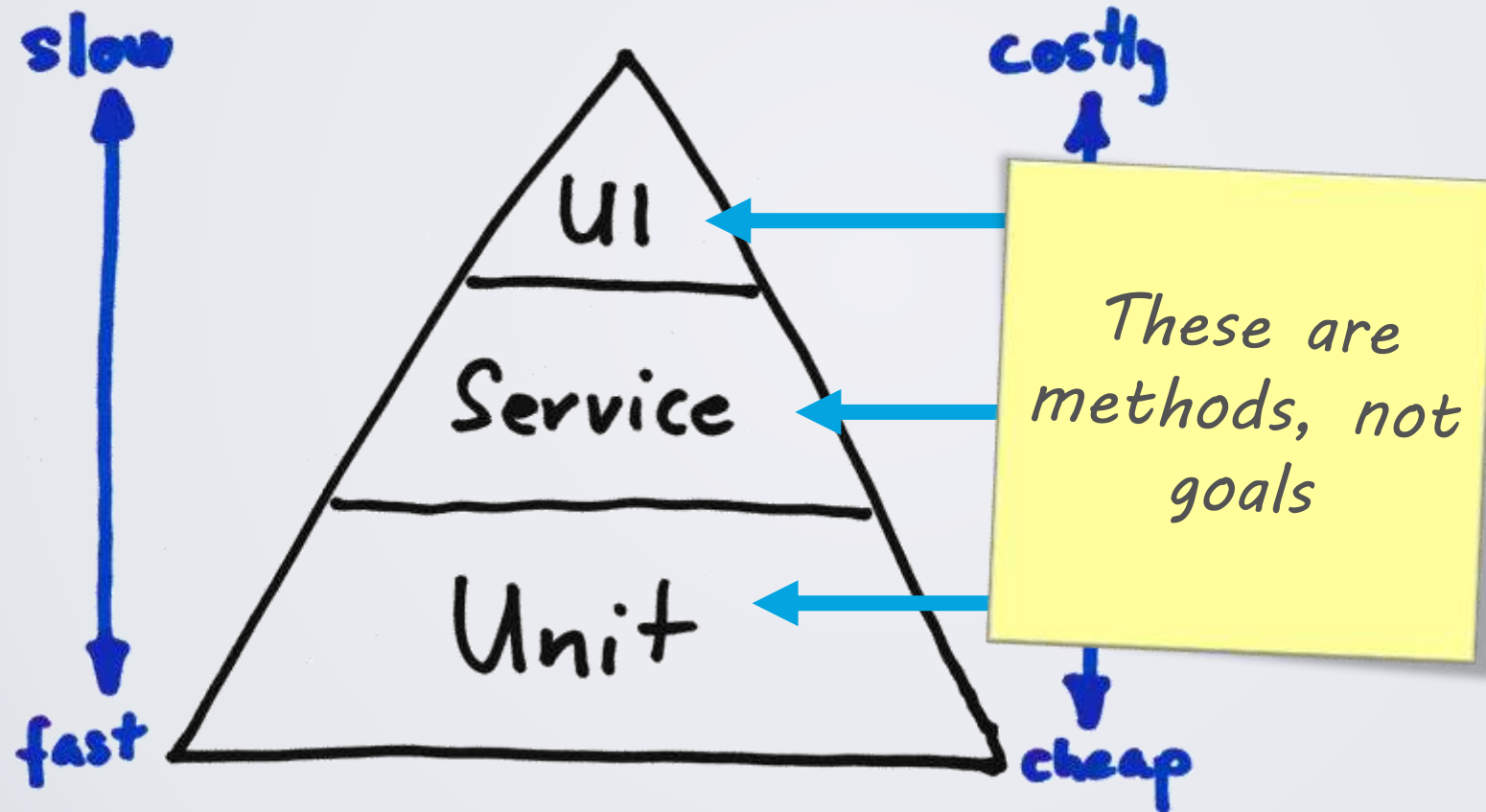
FEEDBACK CHANNELS

- Local test execution
 - Checks functional progress by running *Automated BDD Scenarios* and *SpecSync Algorithmic Logic Tests* regularly
 - Implement, diagnose and fix ADO-related integration issues (with a temporarily reconfigured *Automated BDD Scenarios* and with *ADO Integration Tests*)
 - Occasionally verify general end-to-end behaviour using *Smoke Tests*
- Exploratory testing results
 - Run occasionally and before completing a feature
 - Verify if the solution is usable for the end users
- CI build
 - Runs *SpecSync Algorithmic Logic Tests*, *ADO Integration Tests* and *Automated BDD Scenarios* with STUB and ADO targets
 - Verifies general health of the project in dev
 - Runs on every push and on Monday mornings even if there were no changes
 - Used as PR verification build
- Platform-Test builds (e.g., Platform-Test-TFS2018)
 - Runs on demand and on Monday mornings
 - Starts the ADO test server VM, runs *Automated BDD Scenarios* against it and shuts down the VM
 - Verifies SpecSync compatibility with older and in promises ADO servers
- Line builds (e.g., line-3.1)
 - Runs on demand and on Monday mornings
 - Like CI build, but for a specific supported version line
 - Verifies quality of older, but actively supported SpecSync versions

A wide-angle photograph of the Great Pyramids of Giza in Egypt. The Great Pyramid of Khufu is the central focus, with the Pyramids of Menkaure and Khafre visible to its left. The pyramids are constructed from light-colored limestone blocks. In the foreground, a camel is kneeling on the sandy desert floor, facing away from the camera. It is adorned with a vibrant, multi-colored patterned blanket. The sky is a clear, pale blue, and the overall scene is bathed in bright, natural light.

The Test Automation Pyramid

The Pyramid® is not enough as a strategy



There is a carnival of pyramids!

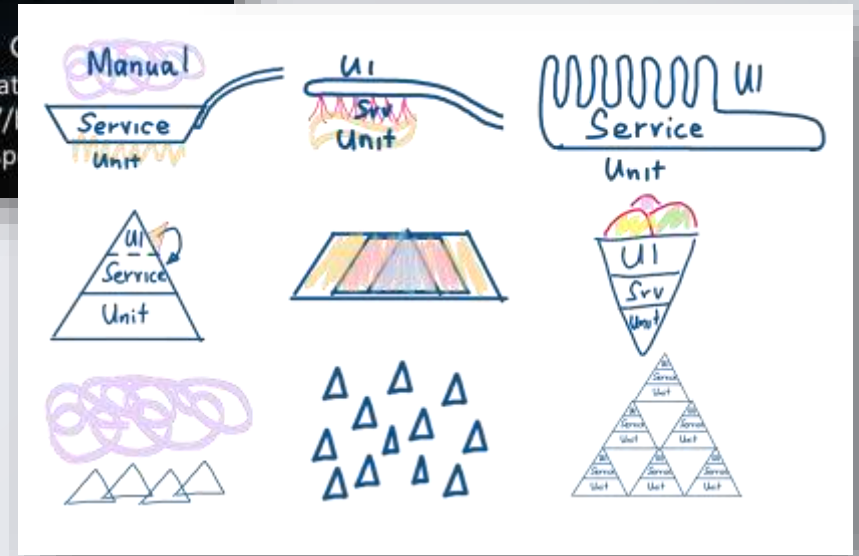
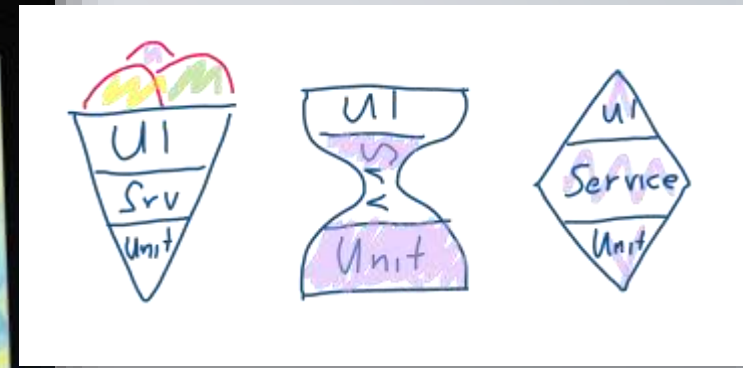
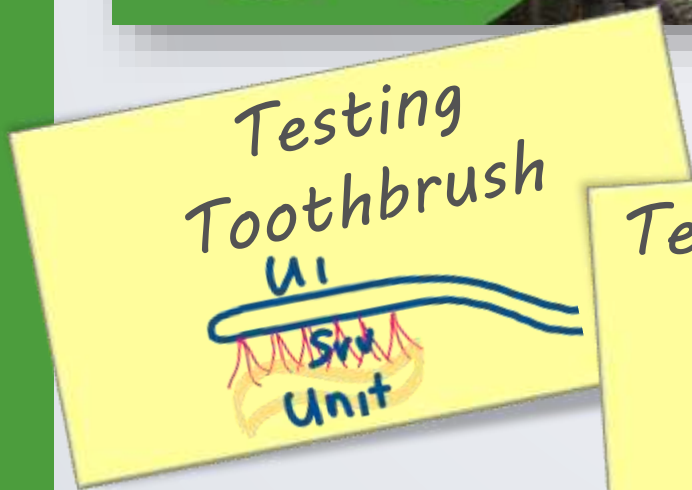
 spec solutions

We are sinking:
Hitting the testing
iceberg

HUSTEF, Budapest
30-31/10/2018

 THE BDD BOOKS
Discovery
Explore behaviour using examples
Gáspár Nagy
and Seb Rose
Foreword by Johannes Rudolph

coach • trainer • bdd addict • creator
"The BDD Books: Discovery" • <http://www.gasparnagy.com>
@gasparnagy • gaspar@spec-solutions.com



A classification of quality aspects

Functional

- Works as expected
- Expectations are good
- Expectations are documented

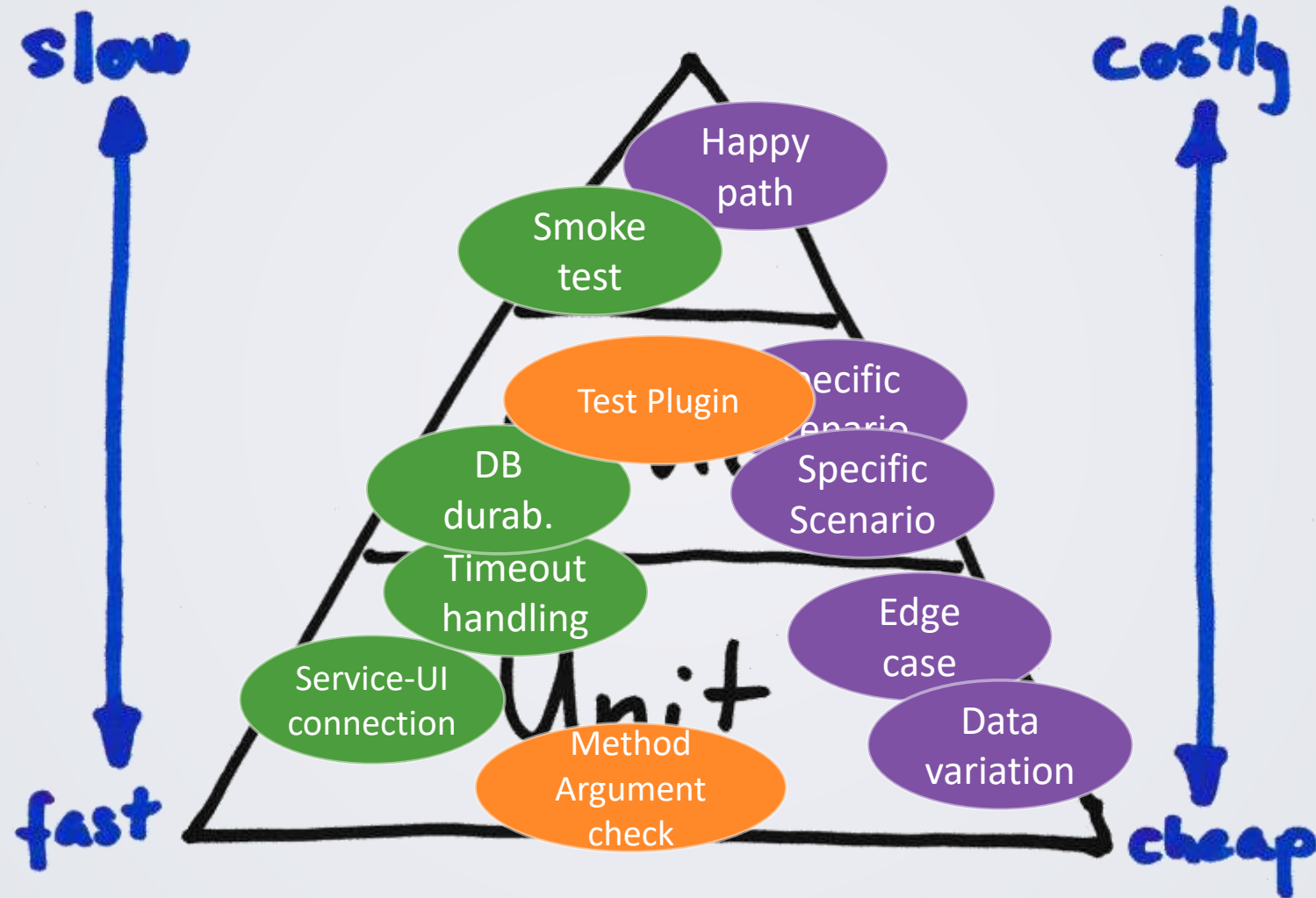
Operational

- Secure
- Fast
- Convenient
- Pretty
- Consistent
- Predictable

Strategic

- Maintainable
- Architecture
- Code quality
- Easy to integrate
- Flexible

Different kind of tests on the testing pyramid



Functional

Operational

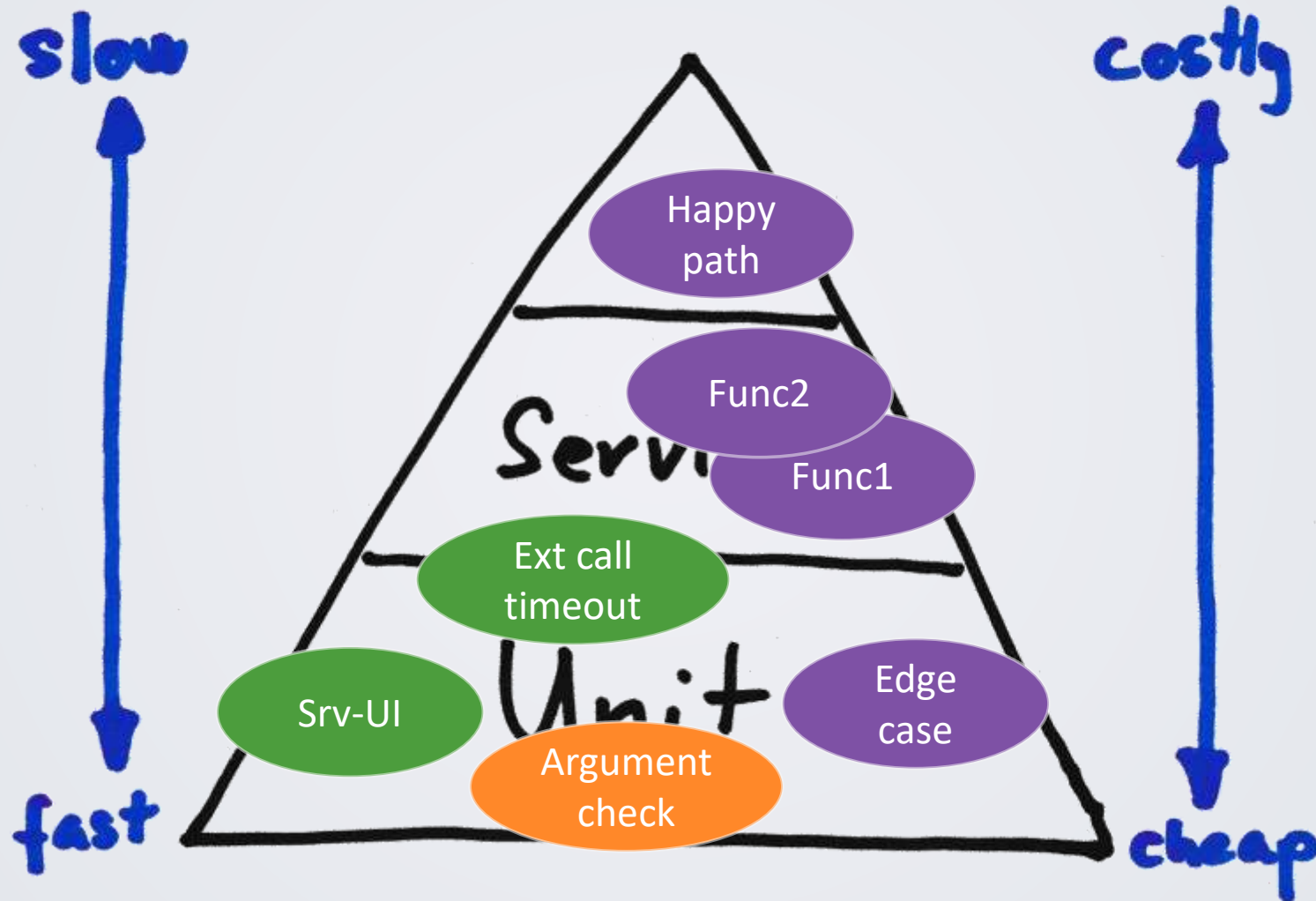
Strategic

testing the UI != UI testing
testing integration != integration test
testing a unit != unit test

A large, jagged iceberg with a prominent peak floats in a calm body of water. The ice has a deep blue hue, especially in the shadows and crevices. The sky is overcast and grey. The water is still, creating a clear reflection of the iceberg and the sky. The overall mood is serene and cold.

The Testing Iceberg

Functional tests in the pyramid

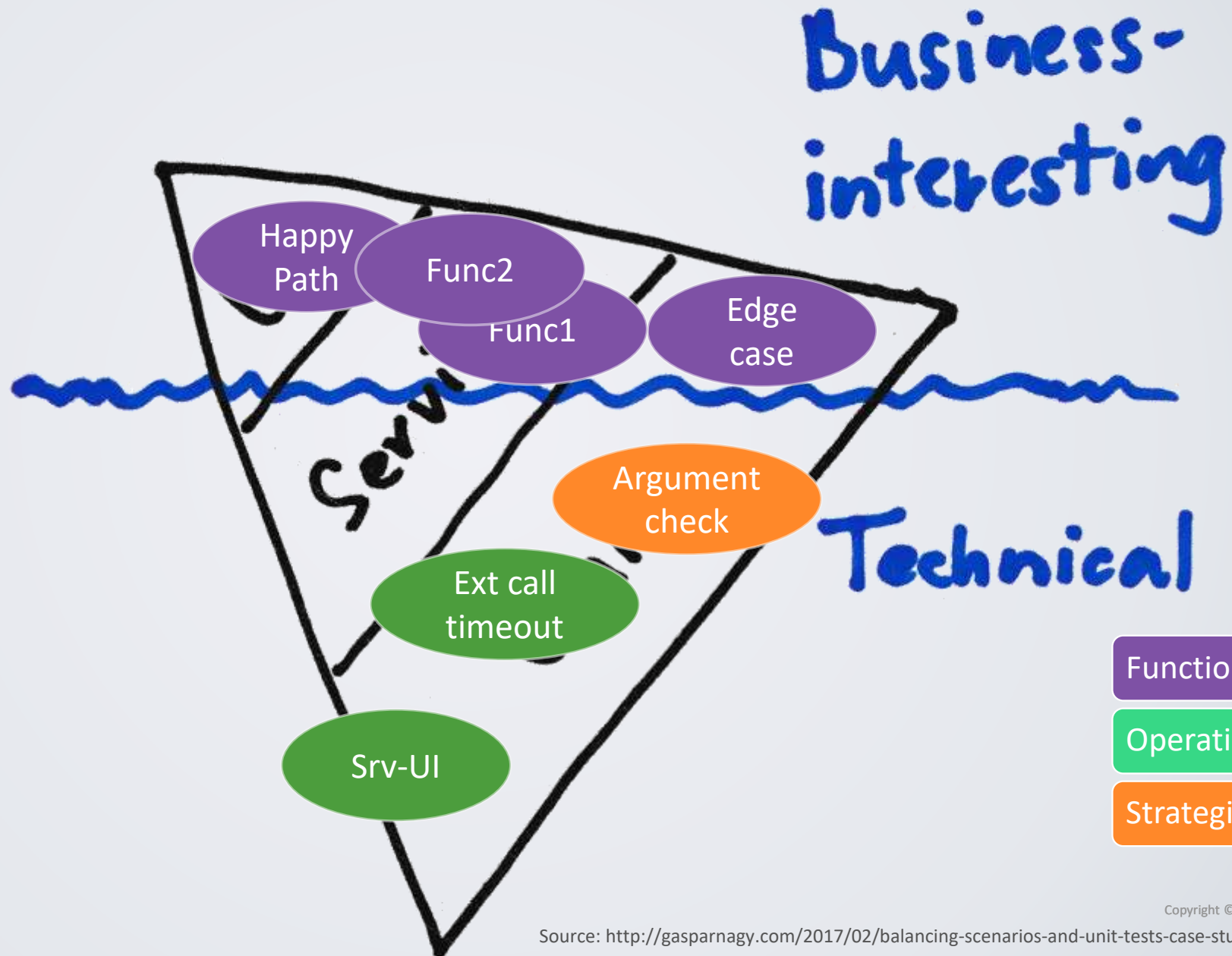


Functional

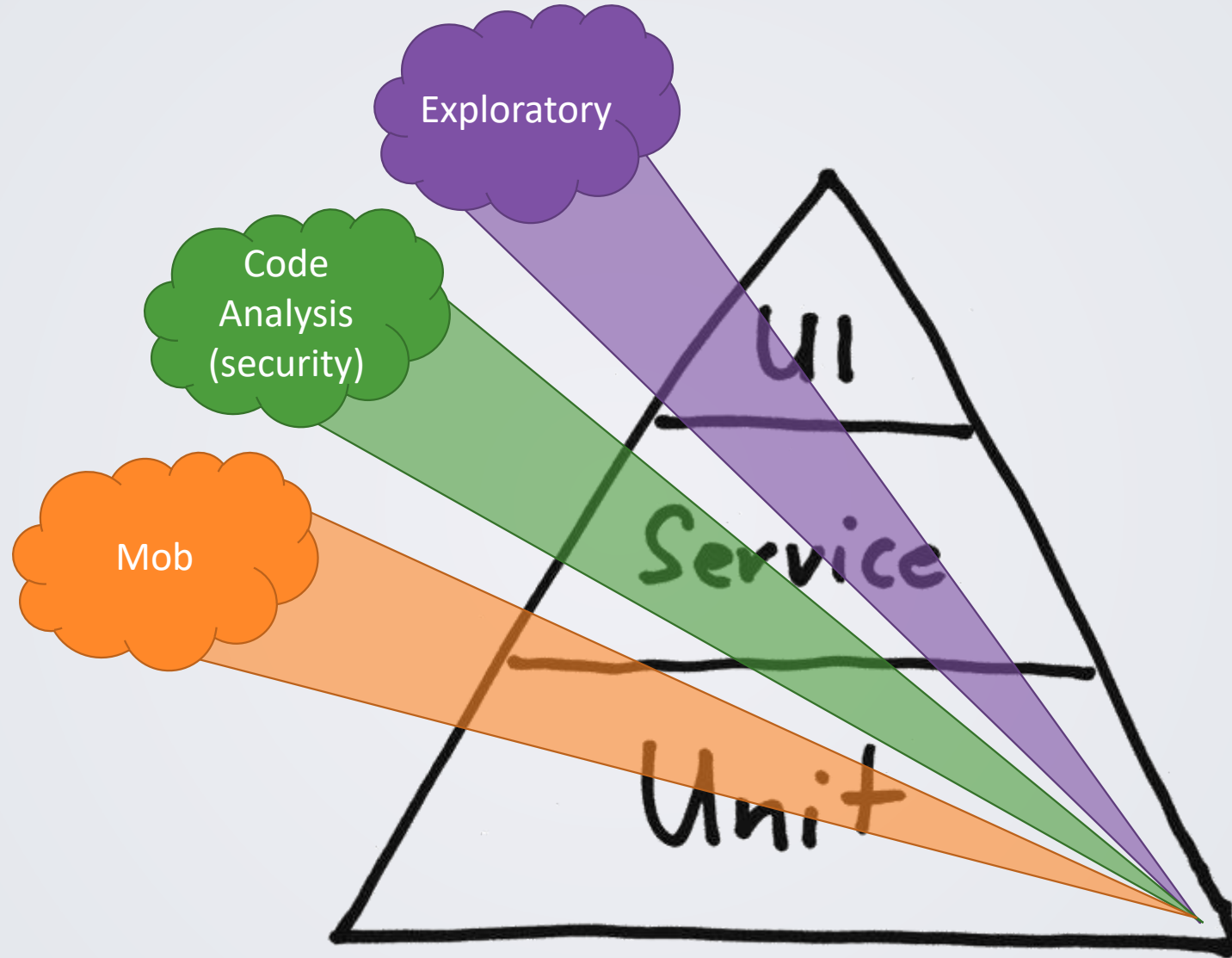
Operational

Strategic

Testing Iceberg!



The Test Automation Prism™



“Maybe pyramid is not a goal, but a use...”
/Mark Winteringham, @2bittester/

*Quality is not (only)
testing!*

Quality is about



And now...



It's your turn...



Thank you!

Upcoming ONLINE Courses:

BDD Vitals:

24 November

BDD with SpecFlow:

24-26 November

See more and book private courses at
<https://www.specsolutions.eu/courses/>

Gáspár Nagy

coach • trainer • bdd addict • creator of specflow
<http://bddbooks.com> • “Discovery” • “Formulation”
@gasparnagy • gaspar@specsolutions.eu



spec**solutions**
given.when.then.