Software Engineering and Architecture
Agile Testing

Olivier Liechti
HEIG-VD
olivier.liechti@heig-vd.ch

MSE | MASTER OF SCIENCE IN ENGINEERING

:(

Your PC ran into a problem that it couldn't handle, and now it needs to restart.

You can search for the error online: INITIALIZATION_OF_BASS_DROP_FAILED

Agenda

- **Testing in software development process**
  - Is it done as the **last phase** of the development cycle?
  - Is it done by an **independent organization**, which enforces quality levels?
- **Agile testing & quality**
  - How do testing activities fit in agile development methods?
  - What is the role of test engineers in agile organizations? Where do they fit?
  - How can we get apprehend the vast topic of “testing” and “validation”?
- **Agile testing quadrants**
  - Selected topics: unit tests, integration tests, automated acceptance tests, systemic tests

Product development cycle

Analyze

Specify

Build

Test

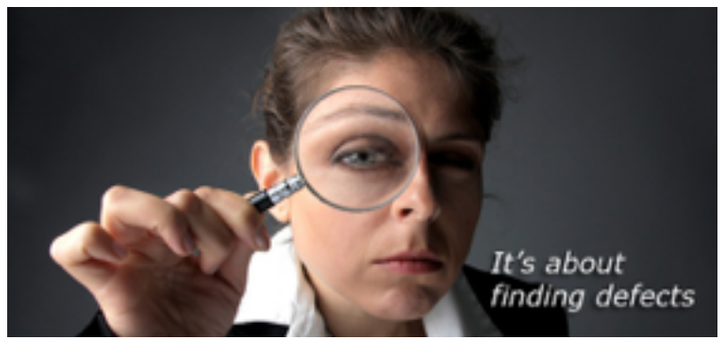
So... anything wrong with that?

Product development cycle

- How much **time** do we need to go through all the phases?
- How often do we go through these phases (is it a **cycle**)?
- How much "**functionality**" is moving through these phases?
- **Who** is working in each phase?
- How do people **collaborate**?

Product development cycle

Analyze → **Specify** → **Build** → **Test**




AUSTIN
AUSTIN

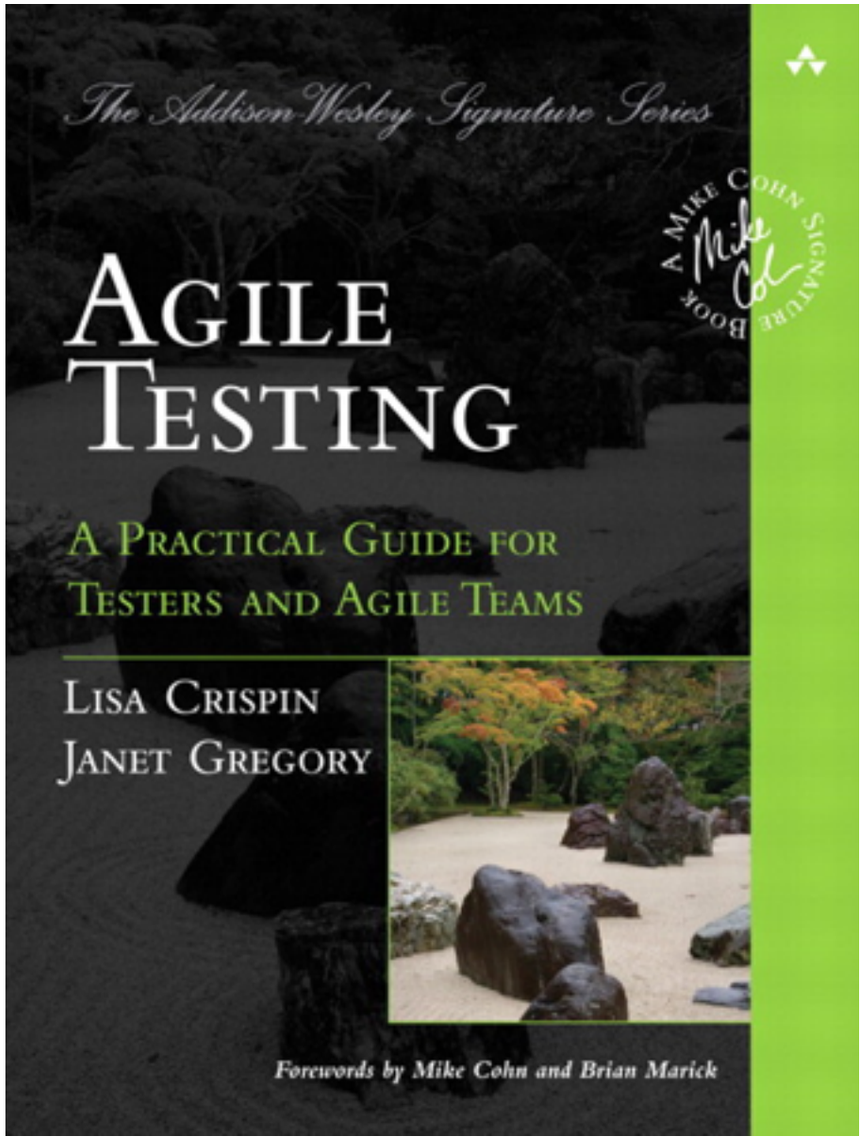

Don’t look at me, it’s his responsibility.
Browman

MSE | MASTER OF SCIENCE IN ENGINEERING

Agile Testing

*"One of the biggest differences in agile development versus traditional development is the "**whole team" approach**.*

*With agile, it's not only the testers or a quality assurance team who feel responsible for quality. [...]. **Everyone on an agile team gets "test-infected"**. Tests, from the unit level on up, drive the coding, help the team learn how the application should work, and let us know when we're "done" with a task or story."*



Janet Gregory and Lisa Crispin
http://www.informit.com/articles/article.aspx?p=1316250&seqNum=5

MSE | MASTER OF SCIENCE IN ENGINEERING

***Small autonomous teams**, which are given the responsibility of a complete "product" are very effective.*

*Communication, collaboration and coordination are much easier than across **organizational silos**.*

*People are much more likely to feel **empowered** and **accountable**.*

http://blog.jasoncrawford.org/two-pizza-teams

Elisabeth Hendrickson, Google Tech Talks

https://www.youtube.com/watch?v=bqrOnIECCSg

MSE | MASTER OF SCIENCE IN ENGINEERING

Agile Testing Quadrants

*"**Software quality**" is a **broad concept** and has many aspects (reliability, efficiency, usability, maintainability, etc.).*

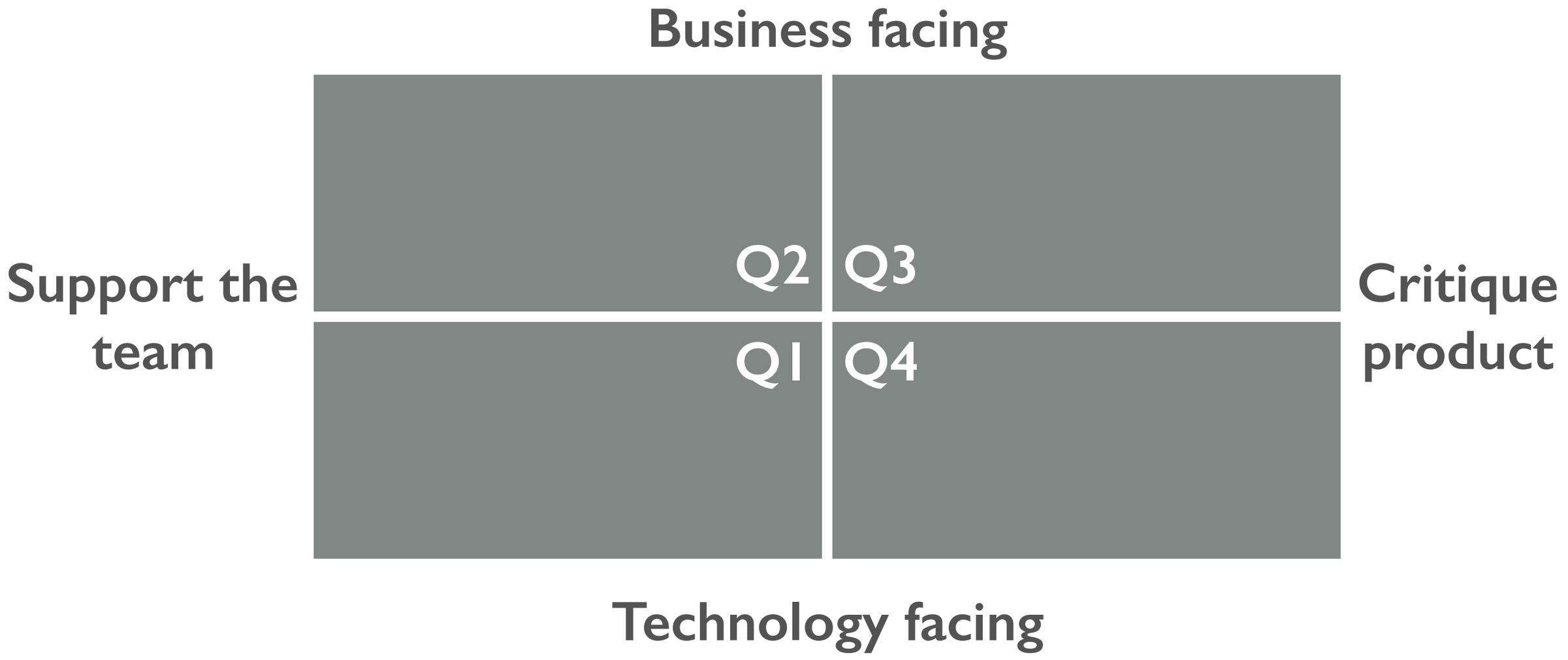
*"**Software testing**" refers to methods and techniques for **assessing** certain aspects of the quality of a software system. **There are many, many of them**.*

*Some "**software testing**" techniques do not only measure quality after the fact, but **help the team to proactively** maintain the quality of the software to an appropriate level.*

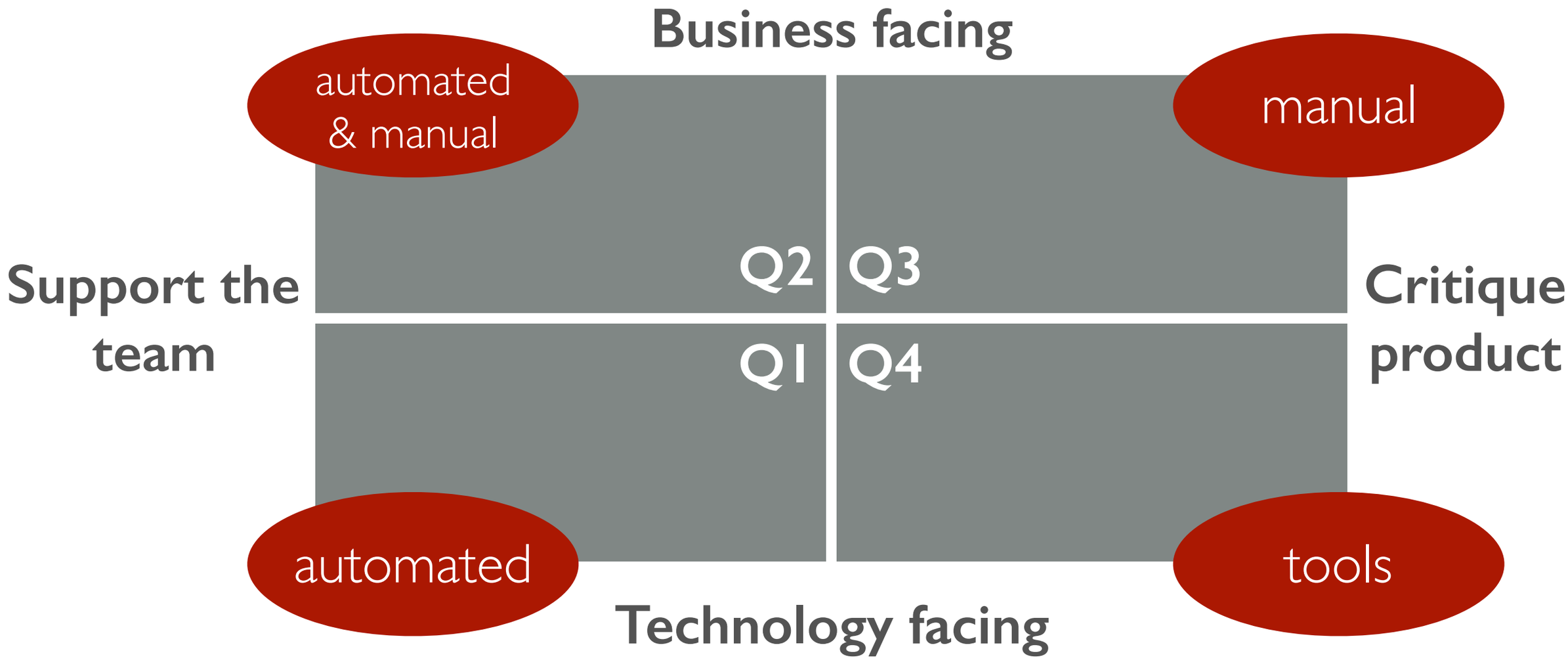
*Is there a way to **classify** all these methods, so that we can see how they relate to each other?*

*These techniques are aligned with the principles of **agile** software development.*

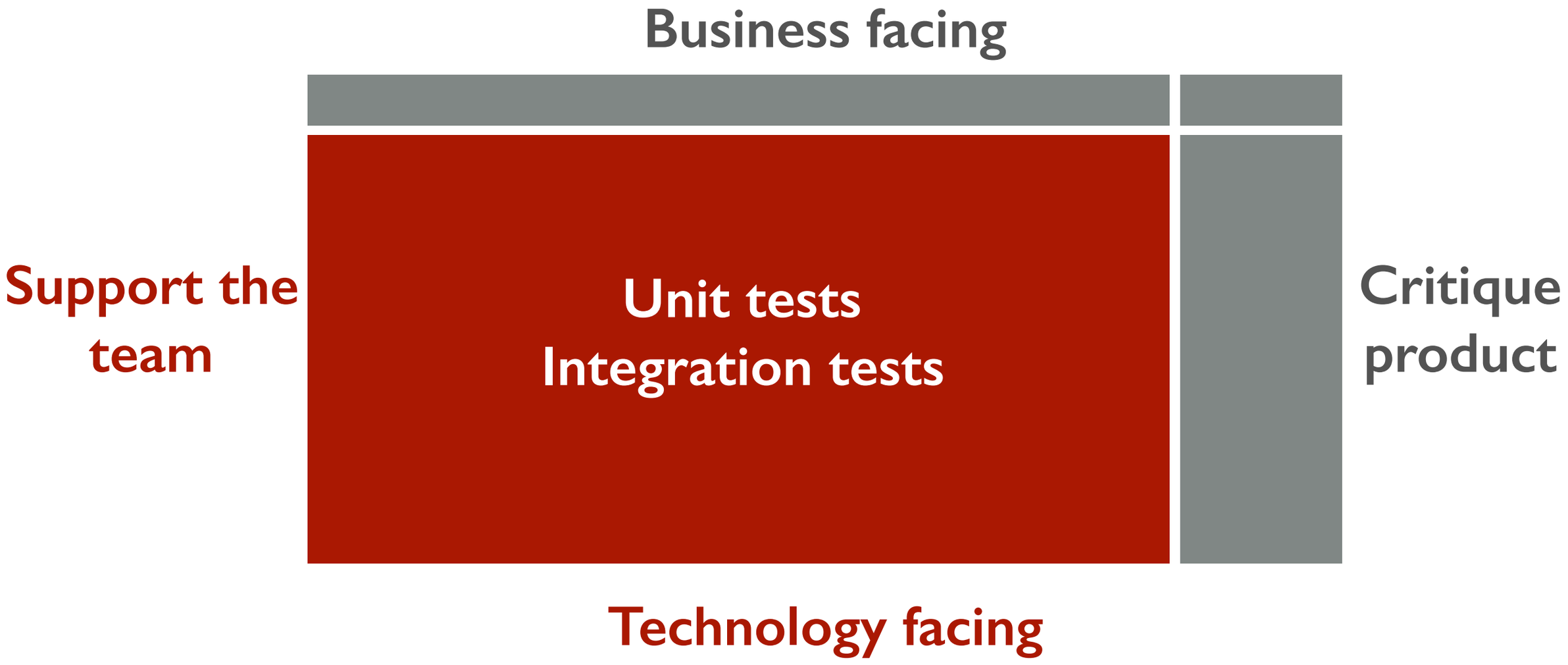
Agile testing quadrants



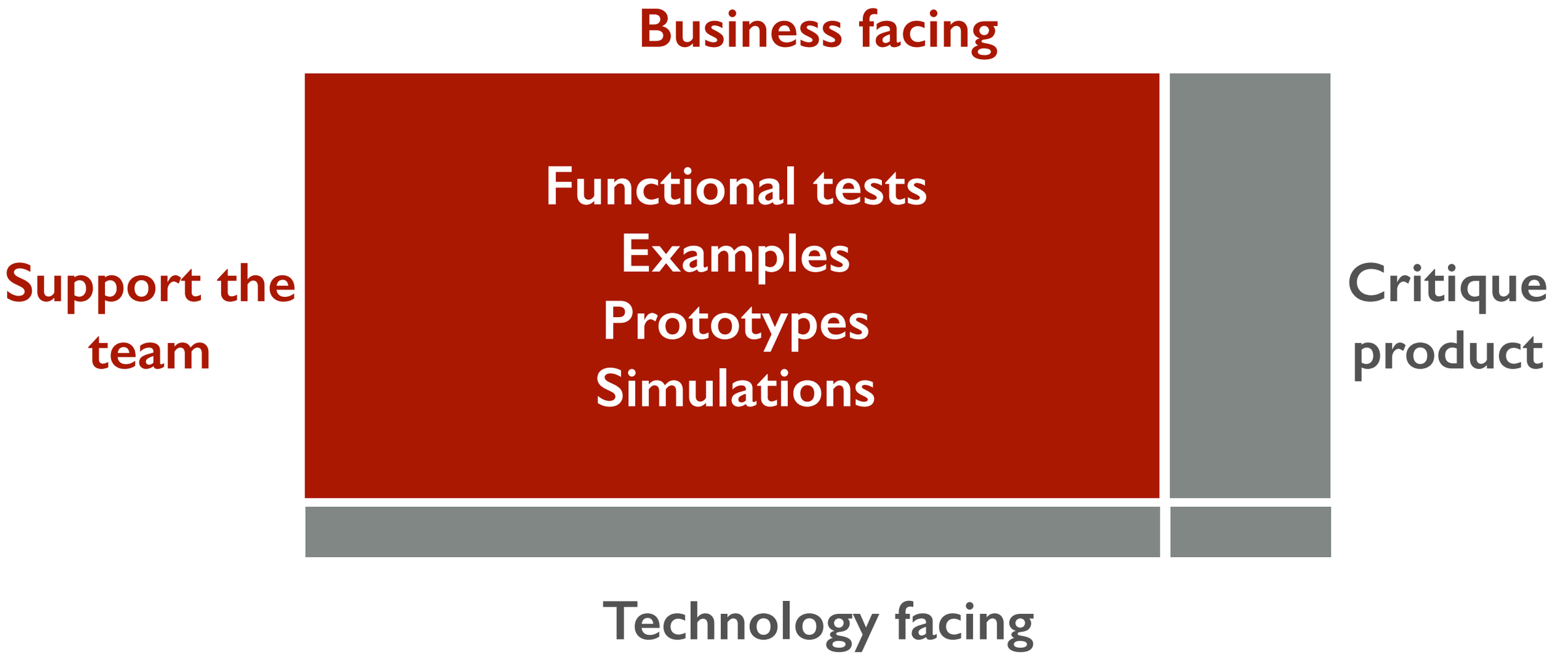
Support the team	Some of these tests **help individual team members** while they do their job. Sometimes, creating a "test" helps me **specify and/or design** the product. Other tests **facilitate team collaboration**, especially between "business" and "technical" people (shared language).
Critique product	Some of these tests allow humans to evaluate the quality of a software from the **users point of view** (is it easy to use? is it easy to learn? does it solve the user's problem?). Other tests aim to detect issues with **non-functional (systemic) qualities**.
Technology facing	Some tests are created and executed by **technical team members**. They are highly automated. They relate to the "Are we building the product right?" question.
Business facing	Some tests are created by (or at least with) **business-oriented team members**. They also relate to the "Are we building the right product?" question.


Business facing
automated & manual
manual
Support the team
Critique product
Q2
Q3
Q1
Q4
automated
tools
Technology facing

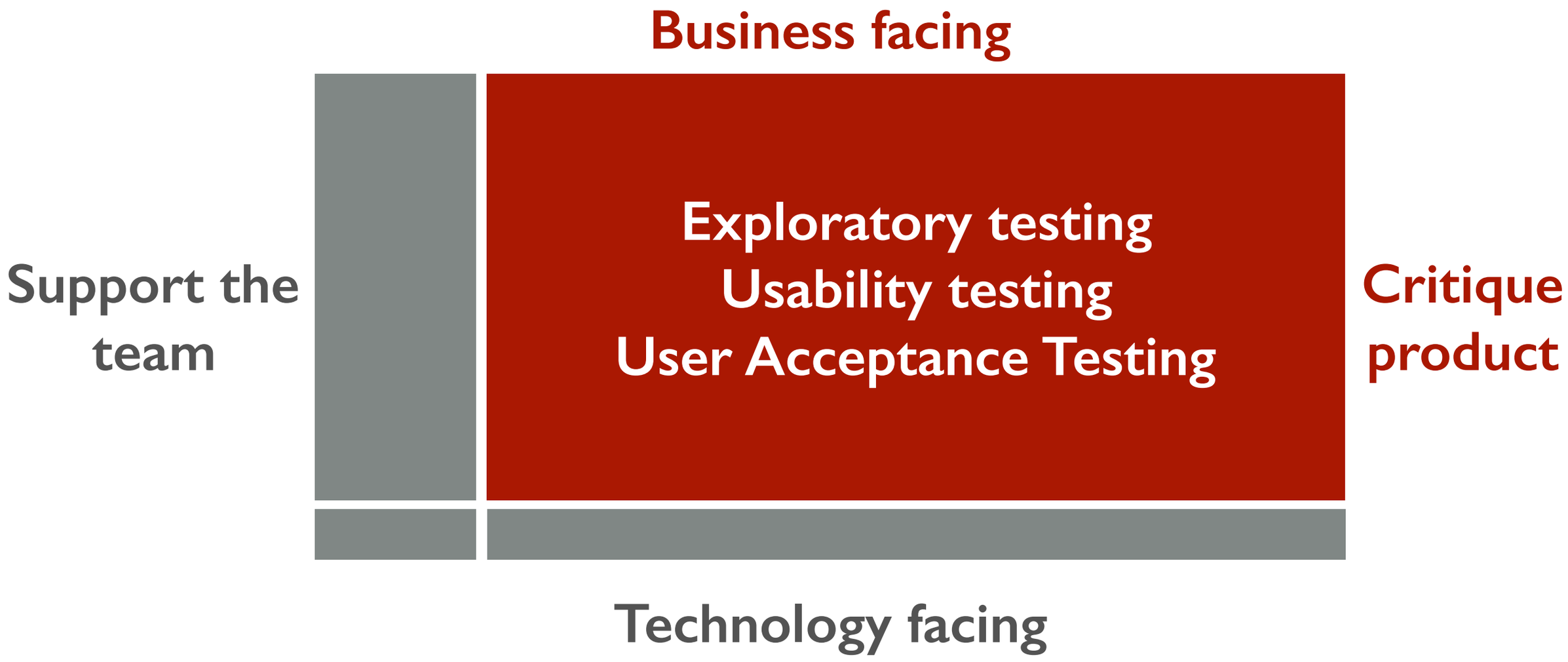
Agile testing quadrants: Q1



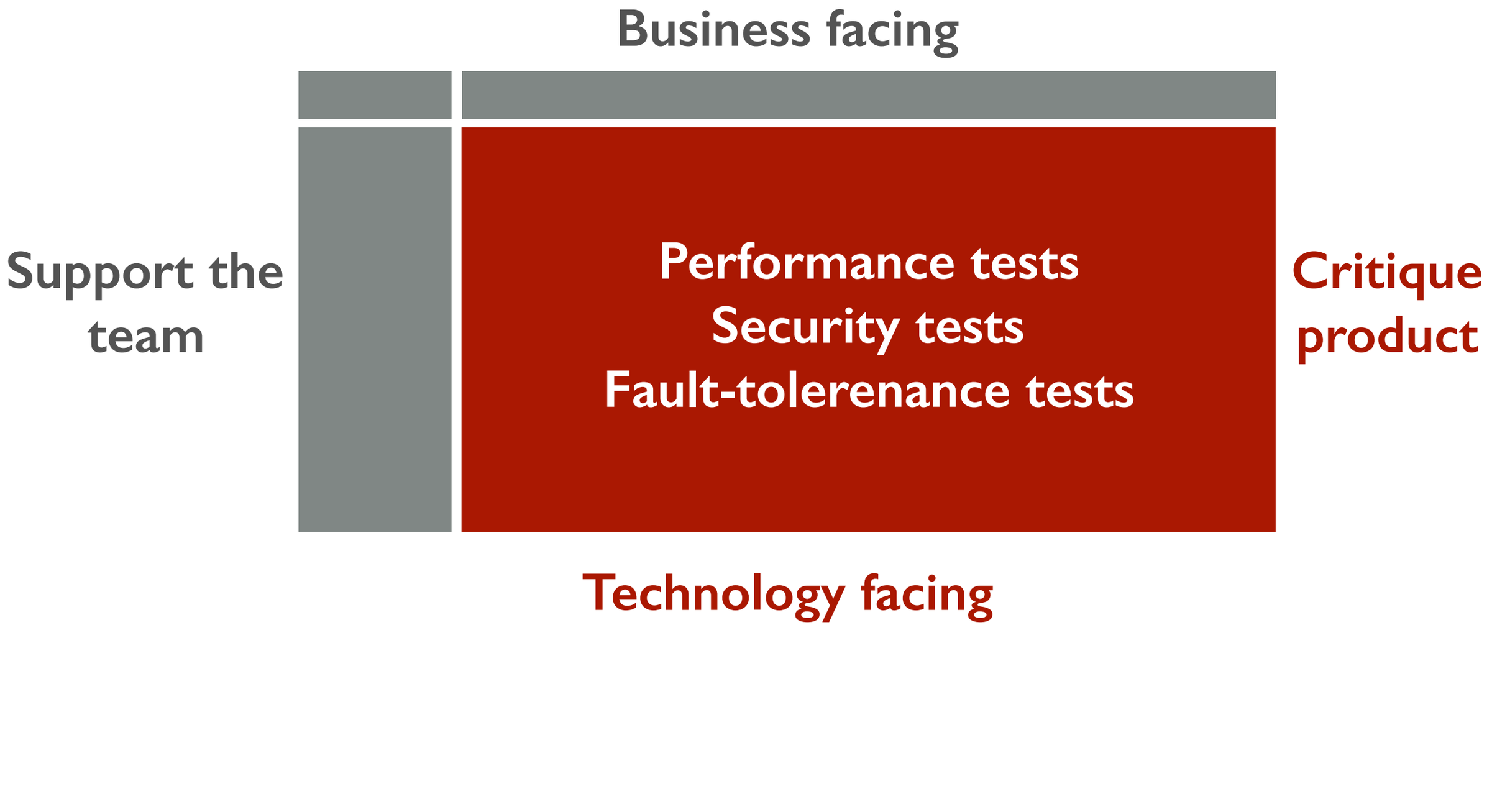
Agile testing quadrants: Q2



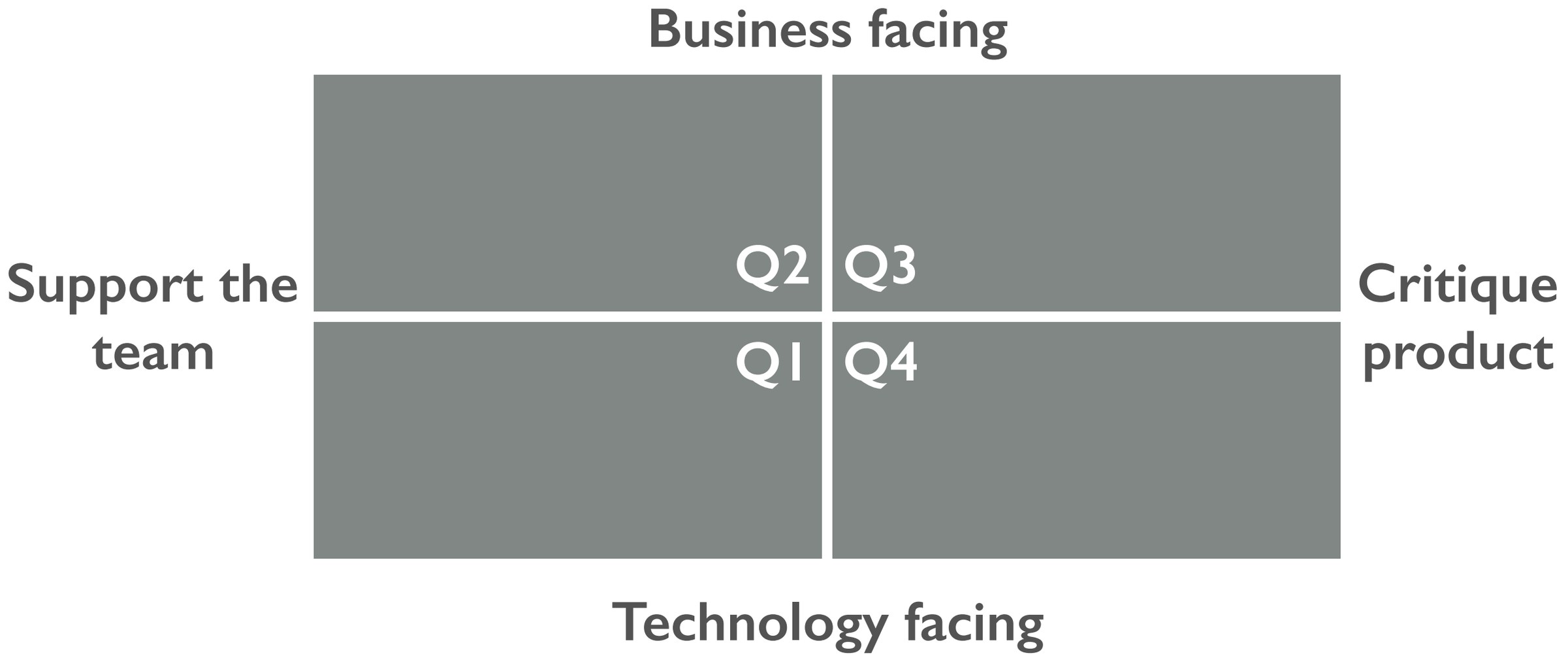
Agile testing quadrants: Q3



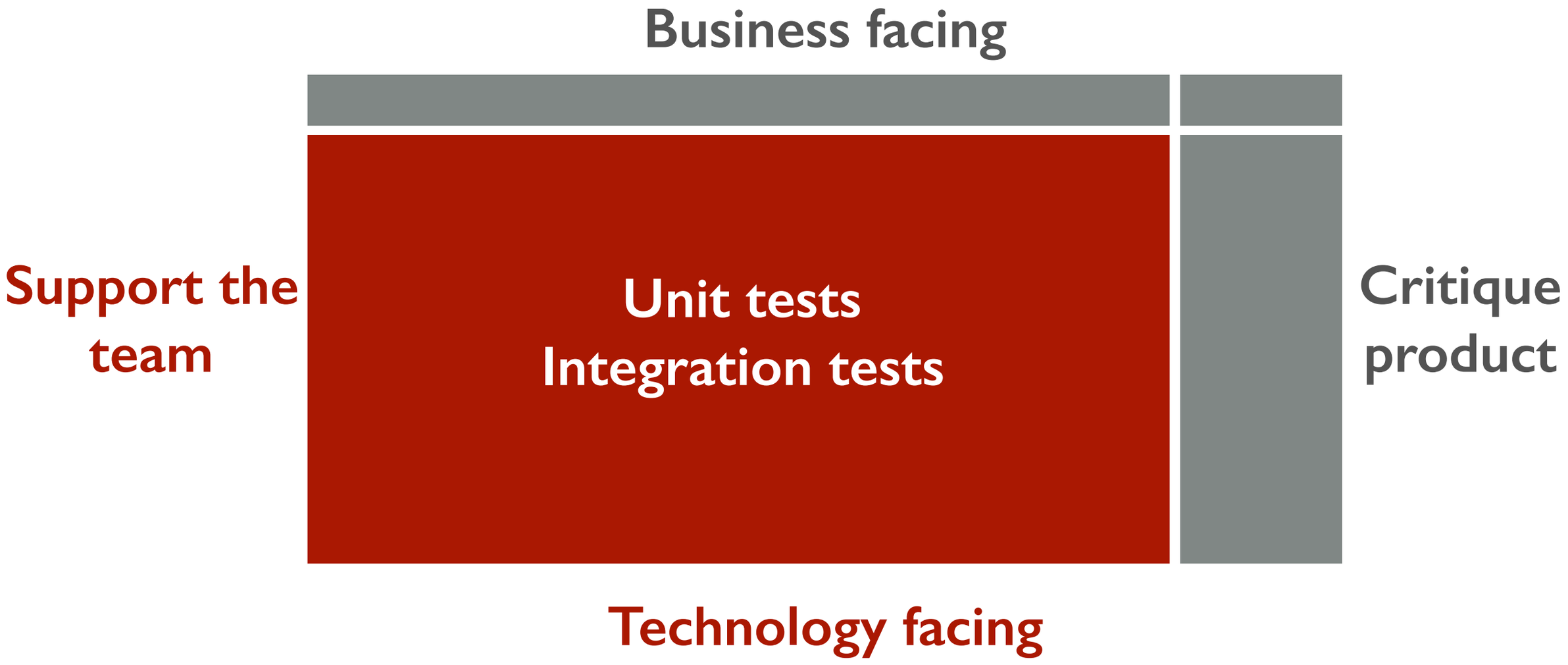
Agile testing quadrants: Q4



Agile testing quadrants



Agile testing quadrants: Q1



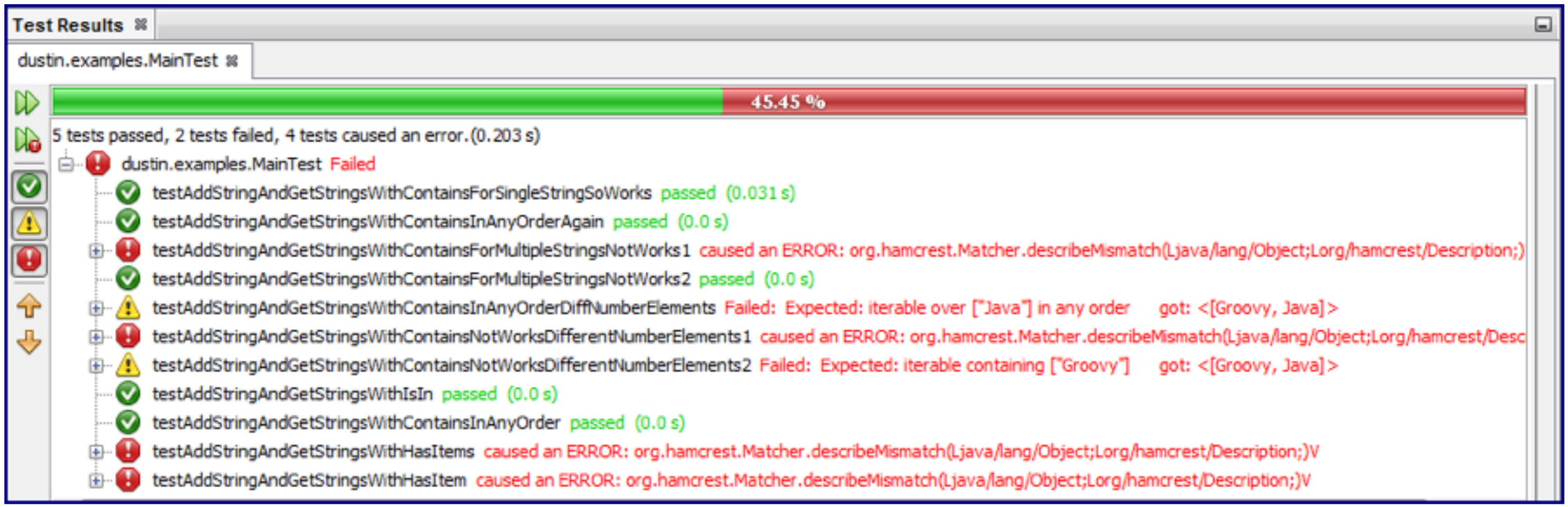
Unit tests: testing is not an after-the-fact activity

- **Writing tests** if not only about checking that a piece of code is doing what it is supposed to do.
- **Writing unit tests** helps us **design** better software (more modular, easier to maintain, easier to evolve). This is the core idea of Test-Driven Development (TDD).
- Writing unit tests, i.e. building a **test harness**, allows us to **refactor** our code with confidence.
- **If unit tests help us design, could other tests help us specify the... behavior of our system? Could they also give us confidence that we don't impact end-users with our changes?**

A harness can be useful and cool.

Whenever *they modify their code, developers can re-run the tests and get a "**green light / red light**" status in **milliseconds or seconds**. Even if there are thousands of unit tests.*

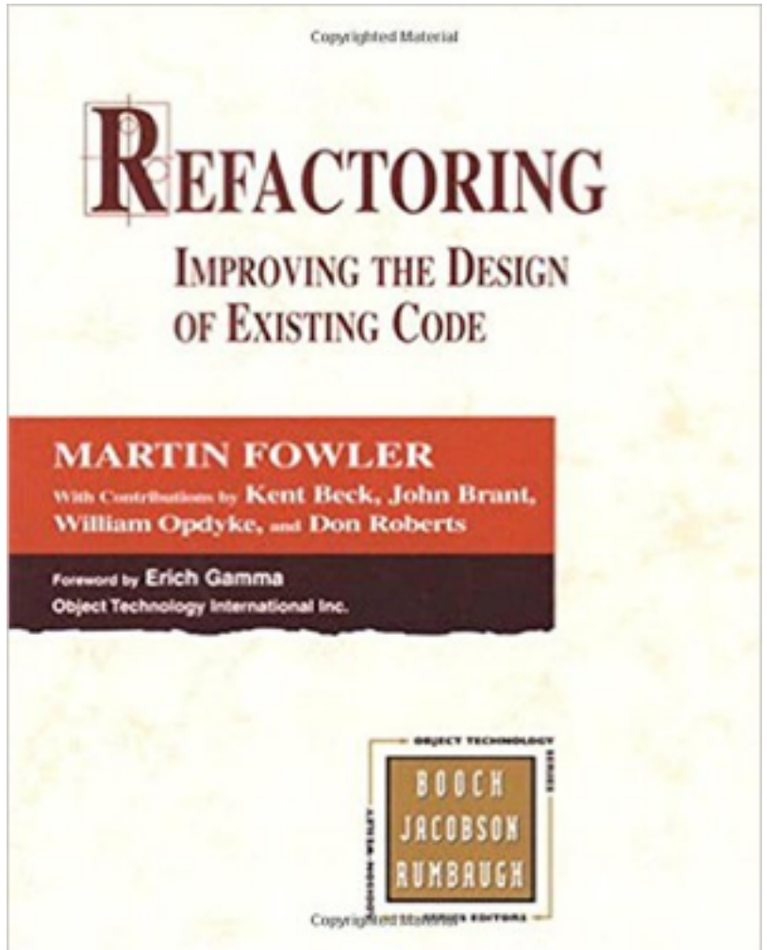
*A unit test **MUST BE extremely fast**. For this reason, a unit test MUST NOT do slow IO operations (no network calls, very few file operations).*



Unit tests enable refactoring

"Refactoring is a disciplined technique for ***restructuring an existing body of code****, altering its internal structure* ***without changing its external behavior****."*

http://www.refactoring.com/



*The **code works** (all tests are green) **but is a mess**. It is difficult to maintain and makes the development of new feature slow.*

*The code is now **nicely structured**. It is meant to do the same thing as before (no behavior change). Because all my unit tests are still green, I have some confidence that this is true.*

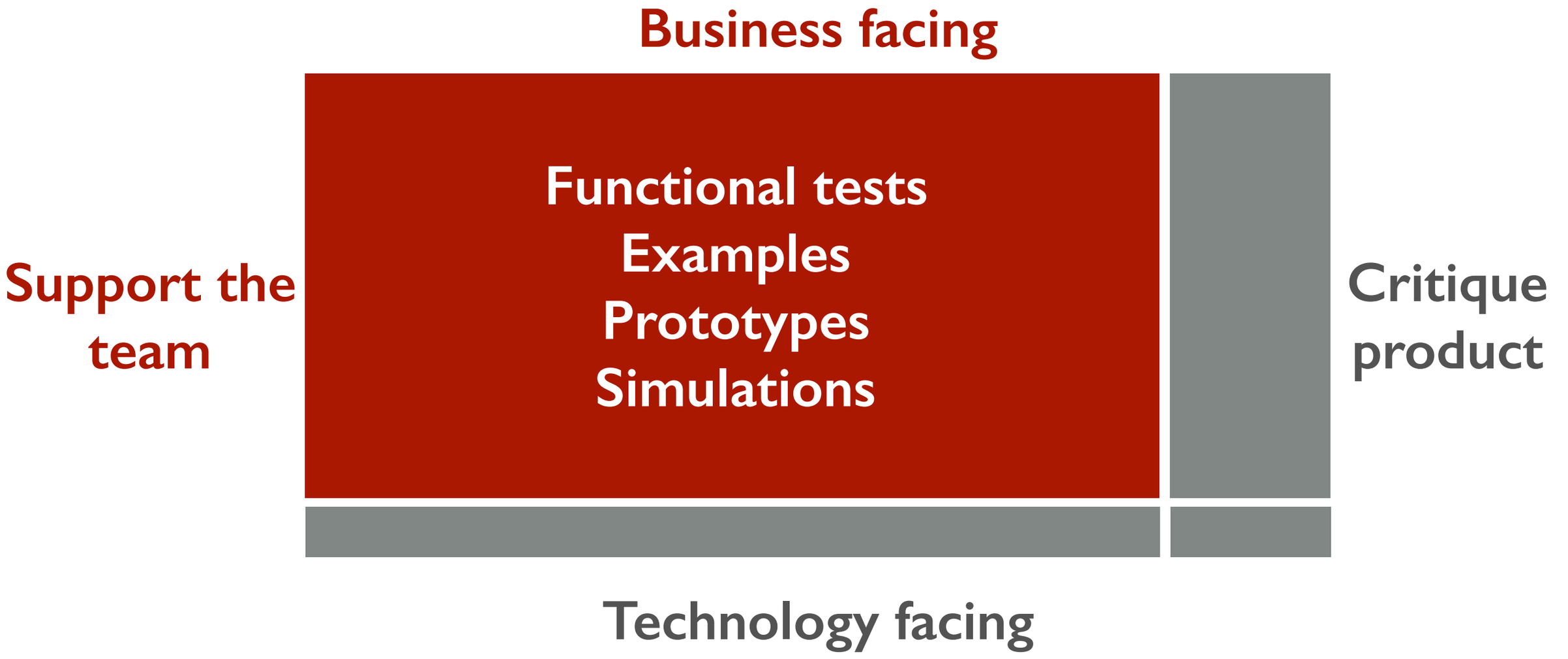
testArguments passed (3.592 s)
testInputOutput passed (0.924 s)
testWelcome passed (2.208 s)
testQuote passed (8.346 s)
testSubProjects passed (3.608 s)
testPi passed (0.598 s)
testFreeway passed (0.012 s)
testFractal passed (15.9 s)
testLexYacc passed (1.363 s)
testMP passed (4.338 s)
testHello passed (0.012 s)
testHelloQtWorld passed (0.009 s)
testProfilingDemo passed (0.025 s)

testArguments passed (3.592 s)
testInputOutput passed (0.924 s)
testWelcome passed (2.208 s)
testQuote passed (8.346 s)
testSubProjects passed (3.608 s)
testPi passed (0.598 s)
testFreeway passed (0.012 s)
testFractal passed (15.9 s)
testLexYacc passed (1.363 s)
testMP passed (4.338 s)
testHello passed (0.012 s)
testHelloQtWorld passed (0.009 s)
testProfilingDemo passed (0.025 s)

Integration tests

- Integration Tests are another type of **automated tests**.
- They validate the **interactions between several units**.
  - Does my “prediction algorithm” work with my “database service”?
  - Does my “web front-end” work with my “REST API”?
- Integration tests are **slower than unit tests**. We do not run them as often as unit tests (not to slow developers down), but we run them very often (at least before sharing one’s code with the team).
- ***In the project, we will implement integration tests that validate the behavior of our micro-services. These integration tests will use the REST APIs as their contract interface.***

Agile testing quadrants: Q2



Functional tests

- With **functional tests**, we want to validate that the system does what it it supposed to do **from the users point of view**.
- Very often, this means **defining usage scenarios** (**test cases**). We describe the steps to be followed by users and the expected results.
- When we evaluate a software release, we can **check** whether the defined test cases can be executed with success.

Manual functional tests

- In many organizations, test cases are documented in **test management software**. They are executed by **human operators**.
- This is a **repetitive process** with little added value.
- This is a **slow process**.
- It creates **overhead** and often gives a **false sense of confidence**.
- If you release every 3 months, it “might” be possible to do manual test campaigns. If you release on a weekly basis, it is just not possible.

Automated functional tests

- There are now **tools** that can be used to **simulate human users**.
- With these tools, you write scripts. When the scripts are executed, they **control a web browser** and check that the content of the pages is.
- **It is not a free lunch**. Writing these scripts takes time. Maintaining these scripts (when the UI changes) takes a lot of time.
- Integration tests are slower than unit tests. Automated functional tests are **a lot slower** than integration tests.
- For this reason, they are not executed as often (at a later stage in the continuous delivery pipeline).

Behavior Driven Development (BDD)

- With Unit Tests, developers have a way to **specify and check** the behavior of a tiny piece of code.
- The same principle can be applied with higher-level, **business oriented tests**. This is the idea of “behavior driven development” or BDD.
- BDD is a method that **facilitates the collaboration** between business analysis, developers and testers. It gives them a **common vocabulary**.

*"My response is **behaviour-driven development** (BDD). It has evolved out of established agile practices and is designed to make them more accessible and effective for teams new to agile software delivery.*

*Over time, BDD has grown to **encompass the wider picture of agile analysis and automated acceptance testing**."*

Dan North, 2006

BDD: Naming & Vocabulary Matters

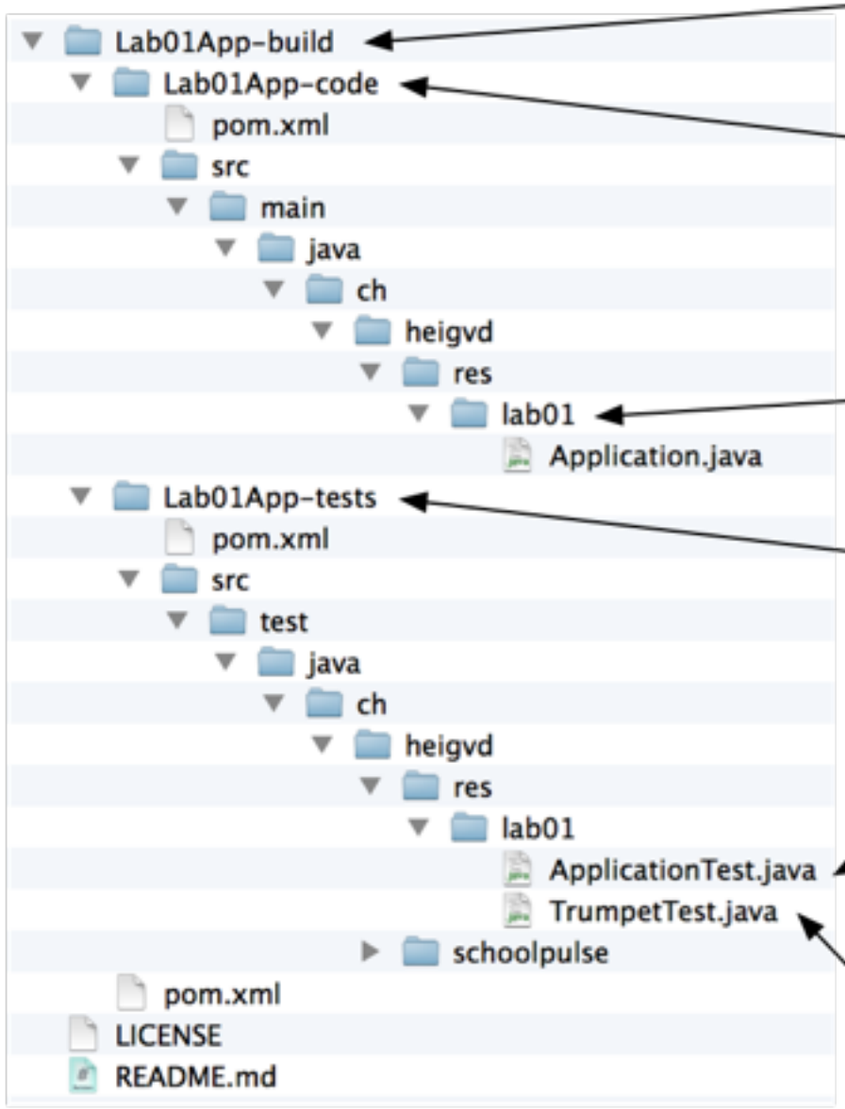
- “Test method names should be sentences”.
- Compare the two representations of the same “specification”. It suggests that tools can support communication by emphasizing a common language for the domain.

see http://agiledox.sourceforge.net/

```
public class FooTest extends TestCase {
    public void testIsASingleton() {}
    public void testAReallyLongNameIsAGoodThing() {}
}
```

```
Foo
- is a singleton
- a really long name is a good thing
```

Example: remember the first lab?



This is a "parent" project, which contains the two others. It is what we use to build and test our application. You can open this project in Netbeans.

This is a project that you can open in Netbeans. It is the project that contains the sources for your application.

We give you a partial implementation. At the beginning, there is only one source file, with one bug. ***You will add code here.***

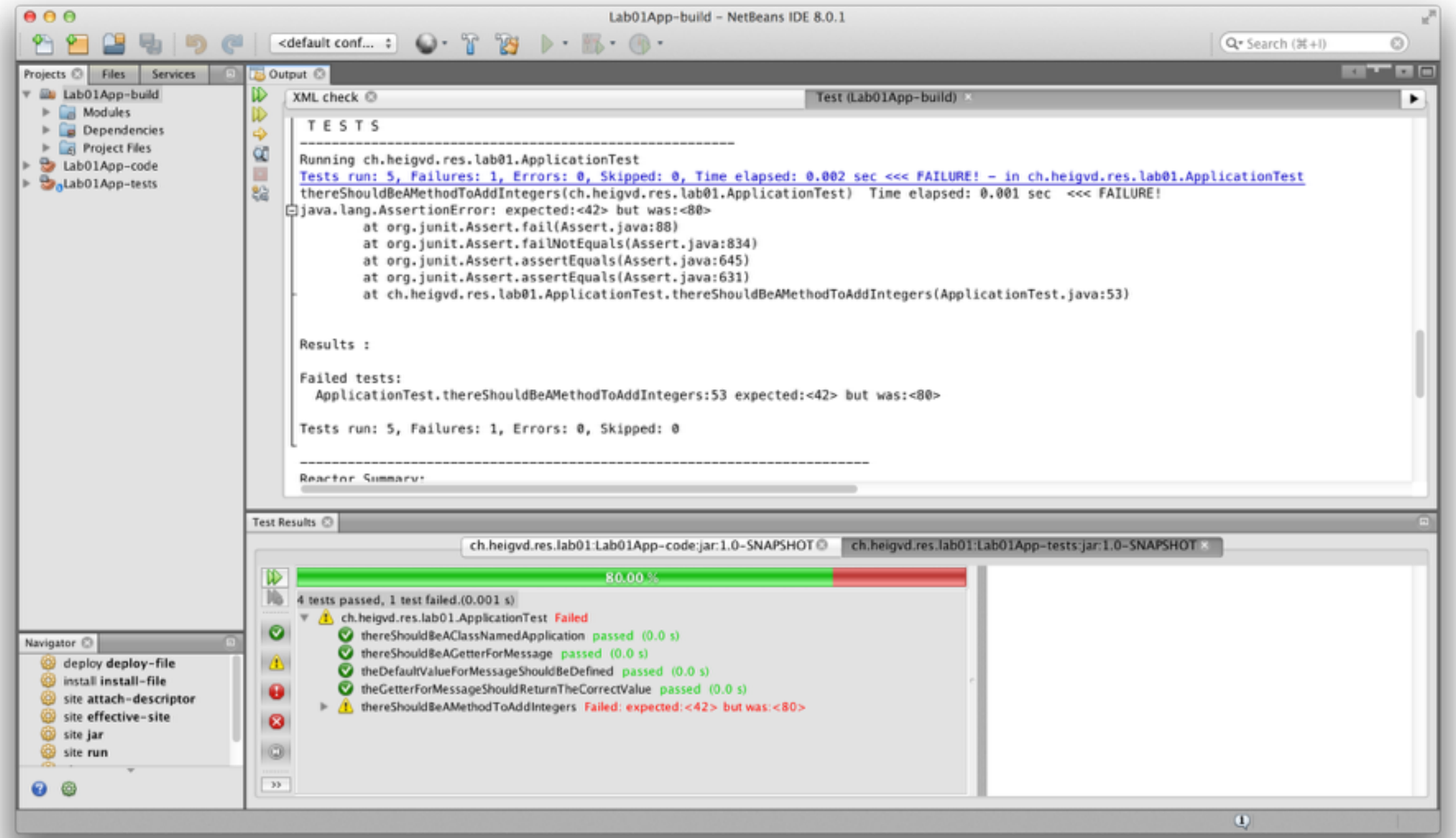
This is also a project that you can open in Netbeans. It contains the unit tests that specify what your application should do. The tests are used to validate your code.

This class contains unit tests to specify and validate the behavior of Application.java. At the beginning, one unit test fails (because we have one bug in Application.java).

This class contains unit tests for code that you have to write. The tests are commented so that you can compile the project after the clone.

```
public class ApplicationTest {

  @Test
  public void thereShouldBeAClassNamedApplication() {
    Application application = new Application();
    assertNotNull(application);
  }

  @Test
  public void thereShouldBeAGetterForMessage() {
    Application application = new Application();
    String message = application.getMessage();
    assertNotNull(message);
  }

  @Test
  public void theGetterForMessageShouldReturnTheCorrectValue() {
    String testValue = "does it work?";
    Application application = new Application(testValue);
    String message = application.getMessage();
    assertEquals(testValue, message);
  }

  @Test
  public void theDefaultValueForMessageShouldBeDefined() {
    Application application = new Application();
    String message = application.getMessage();
    assertEquals("HEIG-VD rocks!", message);
  }

  @Test
  public void thereShouldBeAMethodToAddIntegers() {
    Application application = new Application();
    int sum = application.add(40, 2);
    assertEquals(42, sum);
  }

}
```

```
@Test
public void thereShouldBeAnIInstrumentInterfaceAndATrumpetClas
  IInstrument trumpet = new Trumpet();
  assertNotNull(trumpet);
}

@Test
public void itShouldBePossibleToPlayAnInstrument() {
  IInstrument trumpet = new Trumpet();
  String sound = trumpet.play();
  assertNotNull(sound);
}

@Test
public void aTrumpetShouldMakePouet() {
  IInstrument trumpet = new Trumpet();
  String sound = trumpet.play();
  Assert.assertEquals("pouet", sound);
}

@Test
public void aTrumpetShouldBeLouderThanAFlute() {
  IInstrument trumpet = new Trumpet();
  IInstrument flute = new Flute();
  int trumpetVolume = trumpet.getSoundVolume();
  int fluteVolume = flute.getSoundVolume();
  Assert.assertTrue(trumpetVolume > fluteVolume);
}
@Test
public void aTrumpetShouldBeGolden() {
  IInstrument trumpet = new Trumpet();
  String color = trumpet.getColor();
  Assert.assertEquals("golden", color);
}
```


Lab01App-build - NetBeans IDE 8.0.1
<default conf...
Search (⌘+I)
Projects
Files
Services
Lab01App-build
Modules
Dependencies
Project Files
Lab01App-code
Lab01App-tests
Output
XML check
Test (Lab01App-build)
T E S T S
Running ch.heigvd.res.lab01.ApplicationTest
Tests run: 5, Failures: 1, Errors: 0, Skipped: 0, Time elapsed: 0.002 sec <<< FAILURE! - in ch.heigvd.res.lab01.ApplicationTest
thereShouldBeAMethodToAddIntegers(ch.heigvd.res.lab01.ApplicationTest) Time elapsed: 0.001 sec <<< FAILURE!
java.lang.AssertionError: expected:<42> but was:<80>
at org.junit.Assert.fail(Assert.java:88)
at org.junit.Assert.failNotEquals(Assert.java:834)
at org.junit.Assert.assertEquals(Assert.java:645)
at org.junit.Assert.assertEquals(Assert.java:631)
at ch.heigvd.res.lab01.ApplicationTest.thereShouldBeAMethodToAddIntegers(ApplicationTest.java:53)
Results :
Failed tests:
ApplicationTest.thereShouldBeAMethodToAddIntegers:53 expected:<42> but was:<80>
Tests run: 5, Failures: 1, Errors: 0, Skipped: 0
Reactor Summary:
Navigator
deploy deploy-file
install install-file
site attach-descriptor
site effective-site
site jar
site run
Test Results
ch.heigvd.res.lab01:Lab01App-code:jar:1.0-SNAPSHOT
ch.heigvd.res.lab01:Lab01App-tests:jar:1.0-SNAPSHOT
80.00 %
4 tests passed, 1 test failed.(0.001 s)
ch.heigvd.res.lab01.ApplicationTest Failed
thereShouldBeAClassNamedApplication passed (0.0 s)
thereShouldBeAGetterForMessage passed (0.0 s)
theDefaultValueForMessageShouldBeDefined passed (0.0 s)
theGetterForMessageShouldReturnTheCorrectValue passed (0.0 s)
thereShouldBeAMethodToAddIntegers Failed: expected:<42> but was:<80>

BDD: “Ubiquitous Language” for Analysis

- BDD proposes a template to describe the intended behavior of a system. The template is used to specify the acceptance criteria for a given user story.

```
Given some initial context (the givens),
When an event occurs,
then ensure some outcomes.
```

USER STORY
As a customer,
I want to withdraw cash from an ATM,
so that I don’t have to wait in line at the bank.

ACCEPTANCE CRITERIA
Given the account is in credit
And the card is valid
And the dispenser contains cash
When the customer requests cash
Then ensure the account is debited
And ensure cash is dispensed
And ensure the card is returned

BDD: Executable Specifications

- **“Acceptance criteria should be executable”**
- We need tools that allow:
  - **analysts** to write the acceptance criteria in plain english, following the previous template;
  - **developers** to write test fixtures that act as intermediary between the specification and the system to test;
  - the **continuous delivery pipeline** to execute the specifications automatically, to integrate the test results in the “live” specification, to notify the team about the results.

Process : When will be done?

```
Scenario:  trader is not alerted below threshold

Given a stock of symbol STK1 and a threshold of 10.0
When the stock is traded at 5.0
Then the alert status should be OFF
```

Executable Specifications

Acceptance criteria for stories are defined as scenarios.

Process : linking the specs with the system

Executable Specifications

```
Scenario:  trader is not alerted below threshold

Given a stock of symbol STK1 and a threshold of 10.0
When the stock is traded at 5.0
Then the alert status should be OFF
```

Test Fixtures

System Under Test (SUT)

```
public class TraderSteps { // look, Ma, I'm a POJO!!

    private Stock stock;

    @Given("a stock of symbol $symbol and a threshold
of $threshold")
    public void aStock(String symbol, double threshold)
{
        stock = new Stock(symbol, threshold);
    }

    @When("the stock is traded at $price")
    public void theStockIsTradedAt(double price) {
        stock.tradeAt(price);
    }

    @Then("the alert status should be $status")
    public void theAlertStatusShouldBe(String status) {
        ensureThat(stock.getStatus().name(),
equalTo(status));
    }

}
```

Process : let's see if we are done...

```
Scenario:  trader is not alerted below threshold

Given a stock of symbol STK1 and a threshold of 10.0
When the stock is traded at 5.0
Then the alert status should be OFF
```

Executable Specifications

The test results are displayed directly in the "living" specs (other reports and notifications are also useful!)

Process : yeah!!!!!!

```
Scenario:  trader is not alerted below threshold

Given a stock of symbol STK1 and a threshold of 10.0
When the stock is traded at 5.0
Then the alert status should be OFF
```

Executable Specificati

The test results are displayed directly in the "living" specs (other reports and notifications are also useful!)

Process : nooooooooooo....

```
Scenario:  trader is not alerted below threshold

Given a stock of symbol STK1 and a threshold of 10.0
When the stock is traded at 5.0
Then the alert status should be OFF
```

Executable Specifications

The test results are displayed directly in the "living" specs (other reports and notifications are also useful!)

I can't wait to get started... what should I do?

Tools

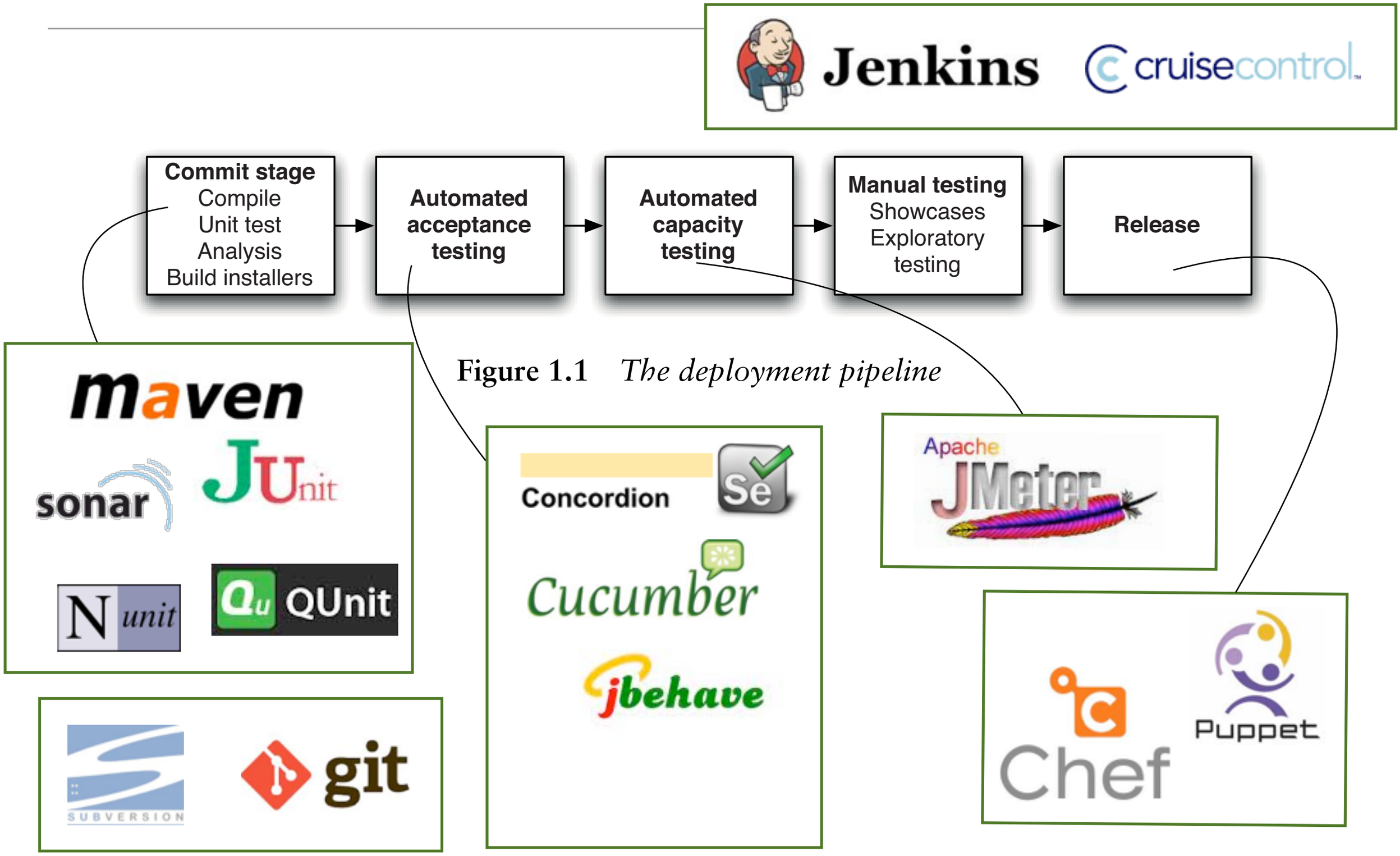


Figure 1.1 *The deployment pipeline*

Tools

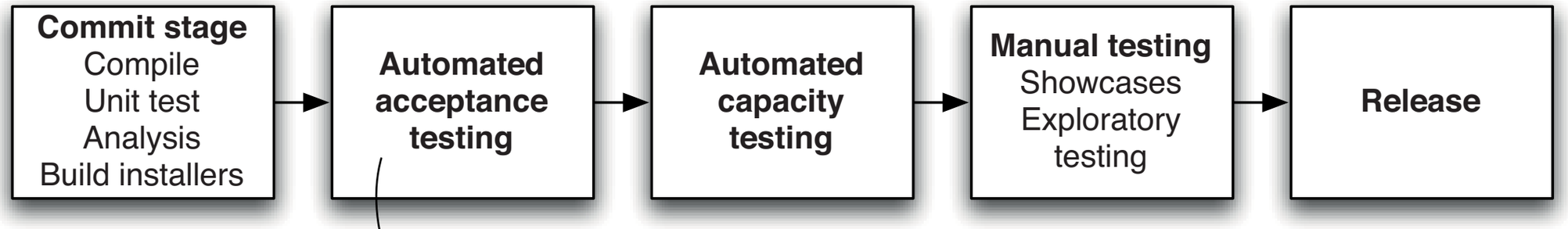
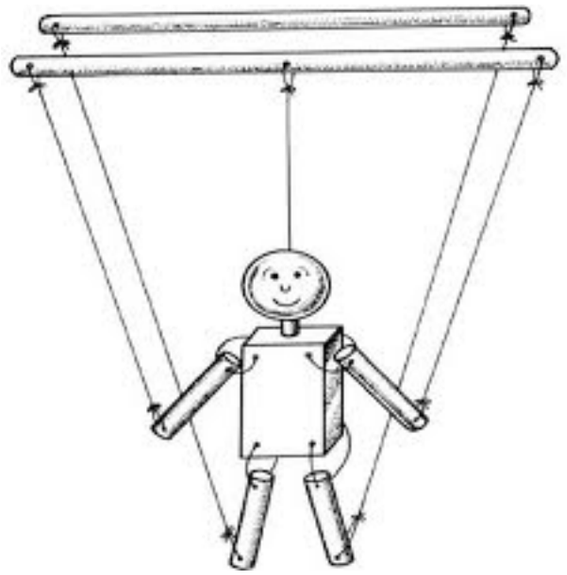


Figure 1.1 *The deployment pipeline*

Selenium

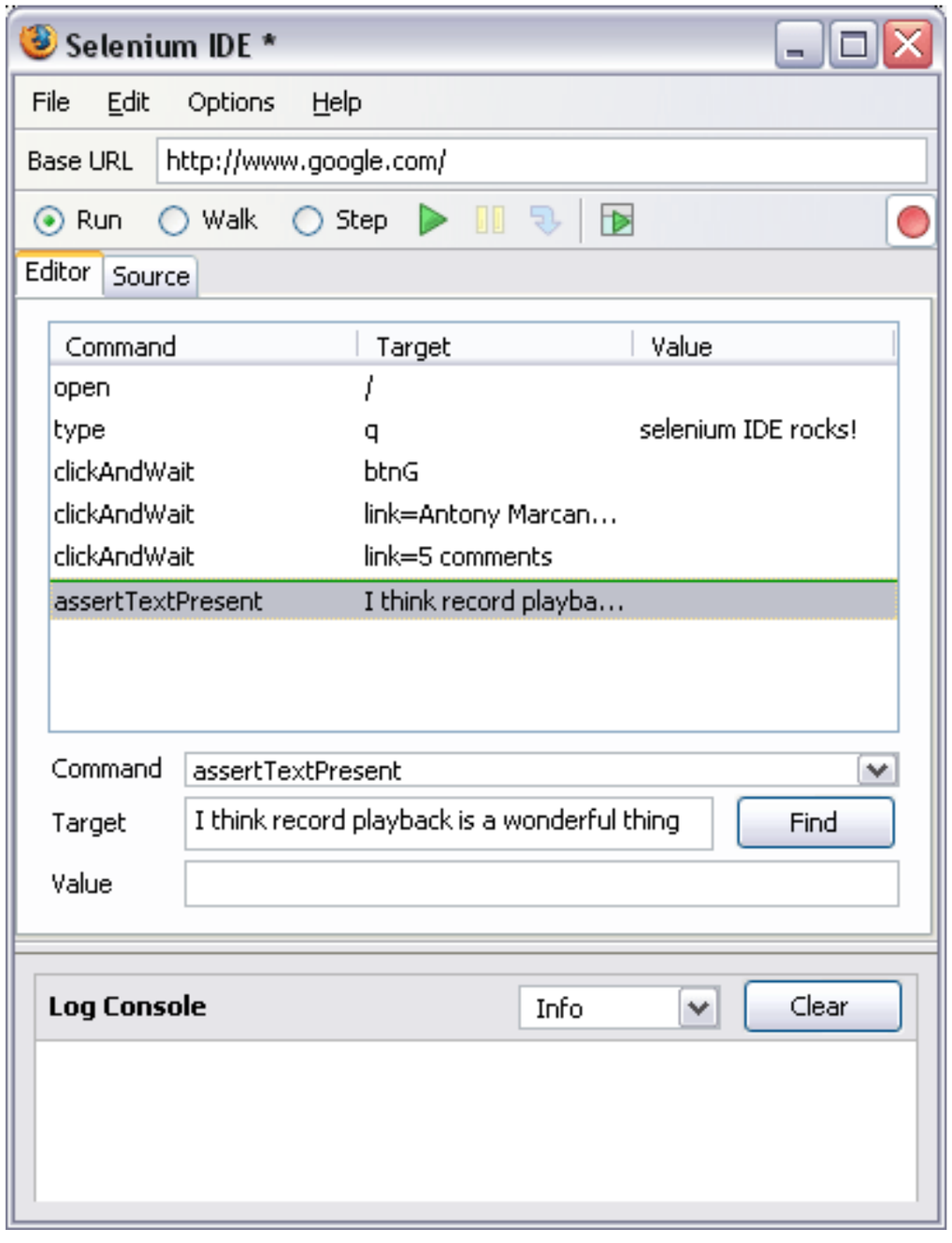
- Selenium provides means to **control web browsers** and to automate tasks, including testing.
- Selenium is a **suite of tools**: Selenium IDE, Selenium Web Driver, Selenium Server.
- With Selenium, it is possible to **record** browsing sessions and to **replay** them. An **API** is also available.
- Selenium is one of the most popular tools for **automated testing of web applications**.



http://seleniumhq.org/

Selenium IDE

- With Selenium IDE, you can record web browsing sessions.
- You can then replay them (Selenium is actually controlling the web browser, so this is testing the real user experience).
- You can also edit your scripts and add tests and assertions.



Selenium Web Driver

- Web Driver provides you with an API to write automated web tests.
- In this case, Selenium still controls a real web browser.
- There are patterns to write good, maintainable tests (in particular, the “Page Object” pattern)
- This is important, because the Web UI is often very dynamic (in the sense that it changes often). You don’t want to have to rewrite all of your tests when a page layout is redone...

```
public class Selenium2Example  {
    public static void main(String[] args) {
        // Create a new instance of the Firefox driver
        WebDriver driver = new FirefoxDriver();

        // And now use this to visit Google
        driver.get("http://www.google.com");

        // Find the text input element by its name
        WebElement element = driver.findElement(By.name("q"));

        // Enter something to search for
        element.sendKeys("Cheese!");

        // Now submit the form. WebDriver will find the form for us from
the element
        element.submit();

        // Check the title of the page
        System.out.println("Page title is: " + driver.getTitle());

        // Google's search is rendered dynamically with JavaScript.
        // Wait for the page to load, timeout after 10 seconds
        (new WebDriverWait(driver, 10)).until(new
ExpectedCondition<Boolean>() {
            public Boolean apply(WebDriver d) {
                return d.getTitle().toLowerCase().startsWith("cheese!");
            }
        });

        // Should see: "cheese! - Google Search"
        System.out.println("Page title is: " + driver.getTitle());

        //Close the browser
        driver.quit();
    }
```

```
public class Selenium2Example  {
    public static void main(String[] args) {
        // Create a new instance of the Firefox driver
        WebDriver driver = new FirefoxDriver();

        // And now use this to visit Google
        driver.get("http://www.google.com");

        // Find the text input element by its name
        WebElement element = driver.findElement(By.name("q"));

        // Enter something to search for
        element.sendKeys("Cheese!");

        // Now submit the form. WebDriver will find the form for us from the element
        element.submit();

        // Check the title of the page
        System.out.println("Page title is: " + driver.getTitle());

        // Google's search is rendered dynamically with JavaScript.
        // Wait for the page to load, timeout after 10 seconds
        (new WebDriverWait(driver, 10)).until(new ExpectedCondition<Boolean>() {
            public Boolean apply(WebDriver d) {
                return d.getTitle().toLowerCase().startsWith("cheese!");
            }
        });

        // Should see: "cheese! - Google Search"
        System.out.println("Page title is: " + driver.getTitle());

        //Close the browser
        driver.quit();
    }
```

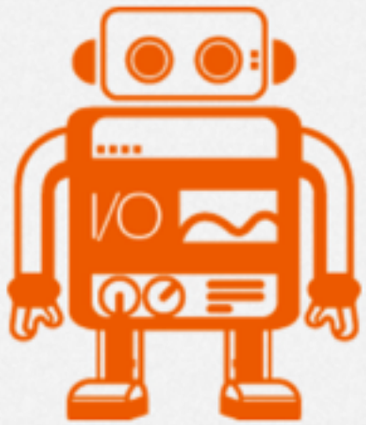
CIENCE
ING

WebdriverIO - Selenium 2.(×

Olivier

webdriver.io

I/O Home Developer Guide API Contribute API Version

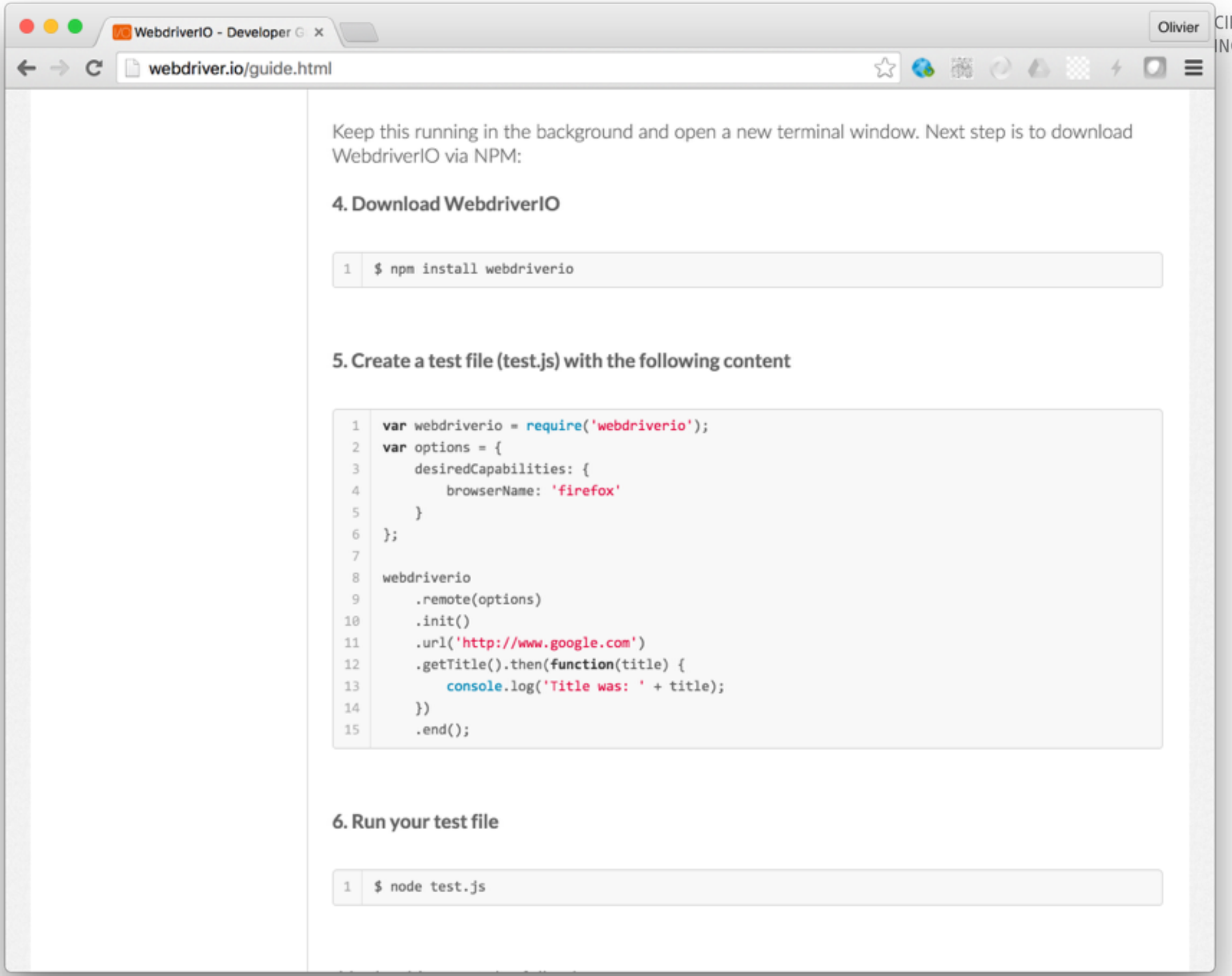


WEBDRIVER I/O

Selenium 2.0 bindings for NodeJS

npm package 4.0.7 | build failing | coverage 67%

Like 138 | G+1 56 | Star 1,851 | Tweet | Follow @webdriverio 615 followers


WebdriverIO - Developer G
Olivier
webdriver.io/guide.html
Keep this running in the background and open a new terminal window. Next step is to download WebdriverIO via NPM:
4. Download WebdriverIO
$ npm install webdriverio
5. Create a test file (test.js) with the following content
var webdriverio = require('webdriverio');
var options = {
desiredCapabilities: {
browserName: 'firefox'
}
};
webdriverio
.remote(options)
.init()
.url('http://www.google.com')
.getTitle().then(function(title) {
console.log('Title was: ' + title);
})
.end();
6. Run your test file
$ node test.js

CIENCE
ING

Nightwatch.js | Node.js po

Olivier

nightwatchjs.org

v0.8.18

Home Developer Guide API Reference Blog Contact

Nightwatch.js

Browser automated testing done easy.

google.js

```
module.exports = {
  tags: ['google'],
  'Demo test Google' : function (client) {
    client
      .url('http://www.google.com')
      .waitForElementVisible('body', 1000)
      .assert.title('Google')
      .assert.visible('input[type=text]')
      .setValue('input[type=text]', 'nightwatch')
      .waitForElementVisible('button[name=btnG]', 1000)
      .click('button[name=btnG]')
      .pause(1000)
      .assert.containsText('#main', 'The Night Watch')
      .end();
  }
};
```

Write End-to-End tests in Node.js quickly and effortlessly that run against a Selenium server.

Download (v0.8.18)

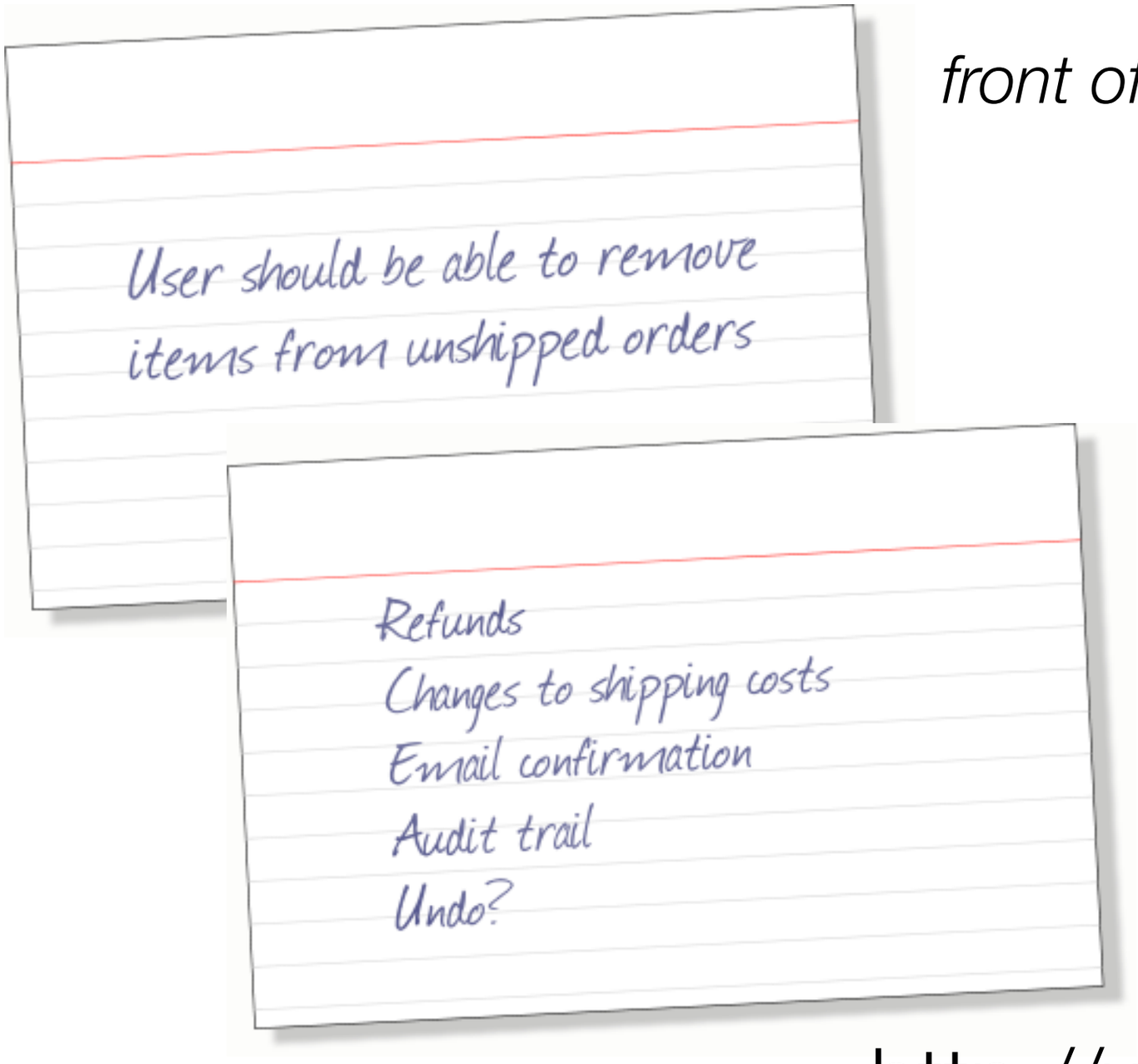
View on GitHub

Tweet Star 4,317

Browser Automation

Nightwatch.js is an easy to use *Node.js* based End-to-End (E2E)

Concordion

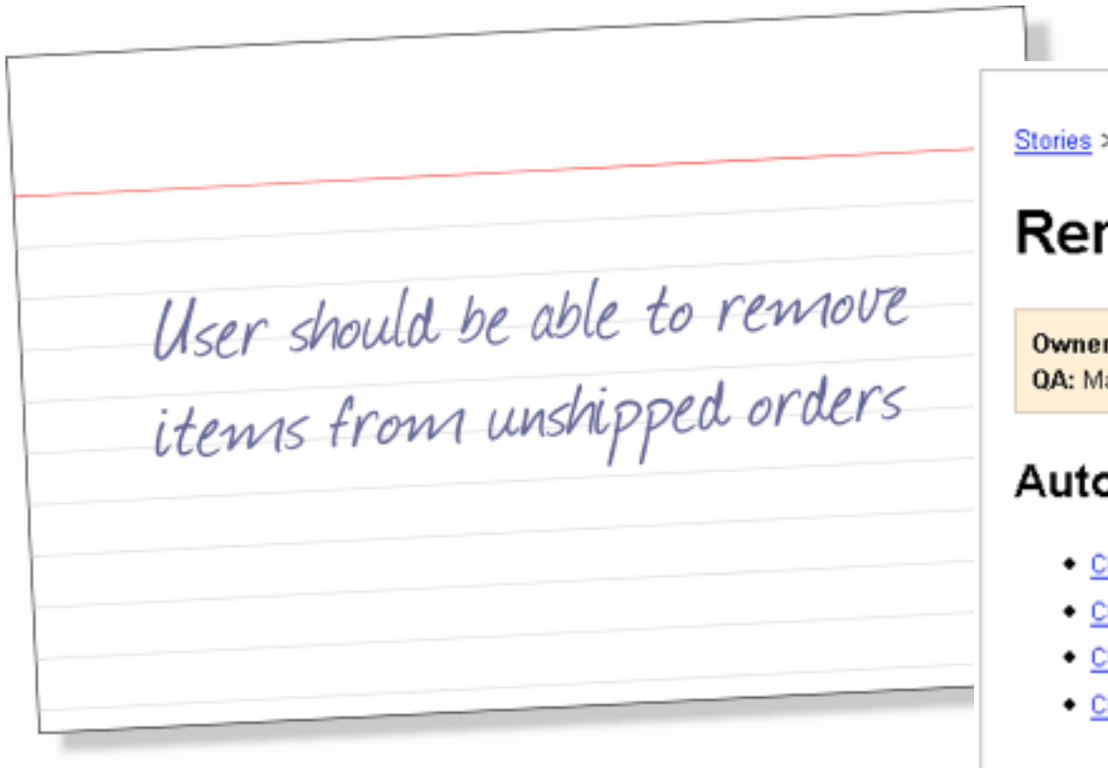


front of the story card

back of the story card (acceptance criteria)

http://www.concordion.org/

Concordion



Stories > Iteration 19 >

Removing Items From Unshipped Orders

Owner: Sharon Hargreaves
QA: Manish Gupta

Automated

- Check item removal is only available for unshipped orders
- Check price is updated (refunds / shipping costs)
- Check e-mail confirmation is sent
- Check changes are audited

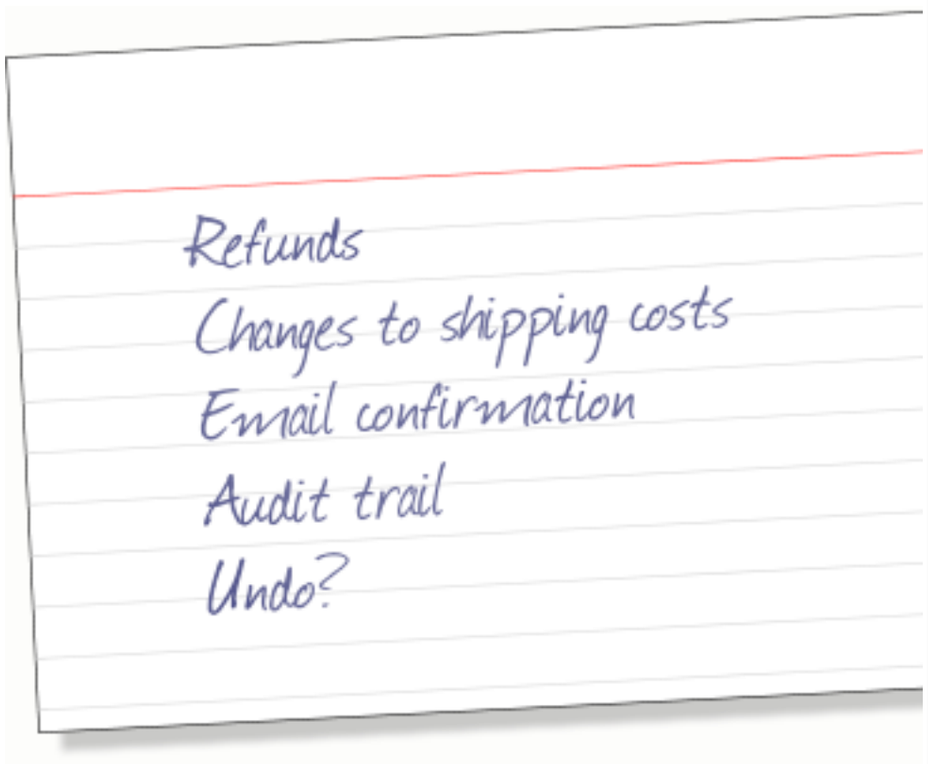
Manual

- Check user-interface changes are consistent with rest of system
- Attempt to remove items from a shipped order (incl. URL manipulation)

Out of Scope

- "Undo" functionality
- Analysis / reporting of suspicious activity
- Handling bounced e-mail

Concordion



Online Shop > Orders > Unshipped > Item Removal >

Item Refund

For orders that are not subject to shipping charges, removal of an item results in a refund of the item's price.

Example

Given an unshipped order with no shipping charges, containing the following items:

Item Description	Price Incl. Tax (£)
Kettle	33.25
Dictionary	18.49
Camera	249.95

If the shopper removes the **Dictionary** from her order then she will be refunded **£18.49**.

The order now contains:

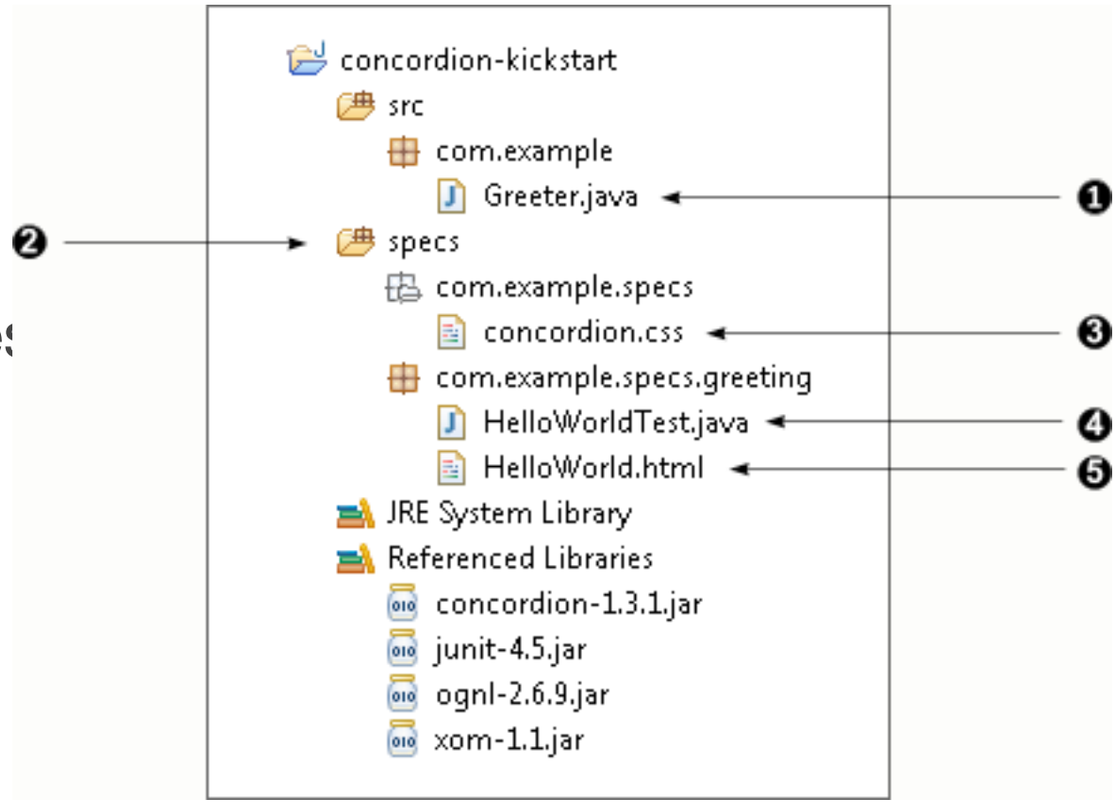
Item Description	Price Incl. Tax (£)
Kettle	33.25
Camera	249.95

Further details

- How are shipping charges handled?
- What happens if the order only contains one item?
- How are the refunds processed?

Concordion

- You write **acceptance tests in HTML** documents.
- You use **special HTML tags** to invoke te fixtures and to make assertions.
- You write **text fixtures as JUnit** tests.
- Executing the JUnit tests generates standard reports AND generates a collection of HTML documents (the living spec).
- You can use (and write) extensions to integrate **Selenium tests** (and even include snapshots in the living specs!)



Acceptance Test & Fixture

Hello World!

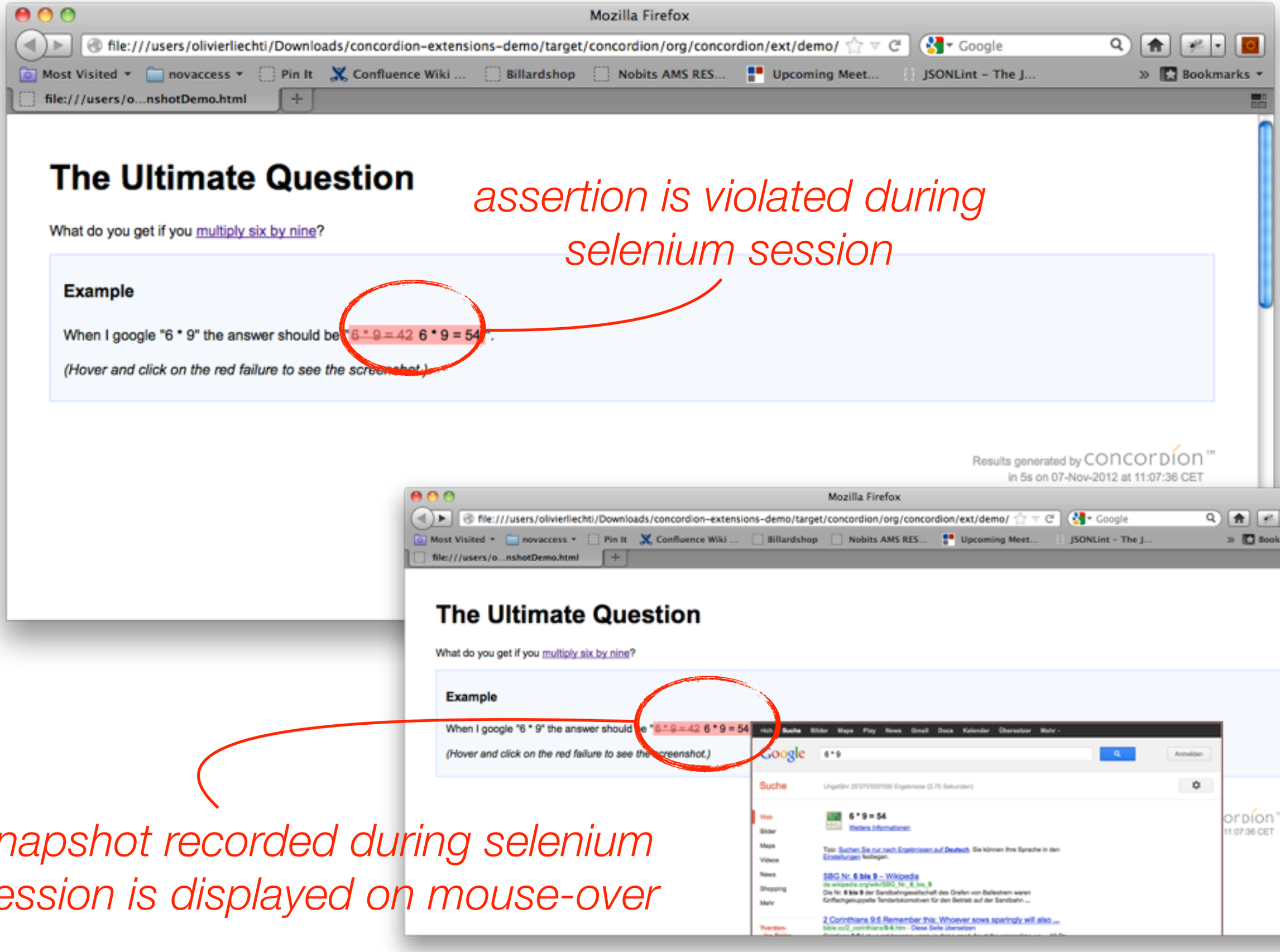
Results generated by concordion™
in 97 ms on 18-Sep-2007 at 16:20:49 BST

HelloWorld.html

```
<html xmlns:concordion="http://www.concordion.org/2007/concordion">
    <body>
        <p concordion:assertEquals="getGreeting()">Hello World!</p>
    </body>
</html>
```

HelloWorldTest.java

```
package example;

import org.concordion.integration.junit3.ConcordionTestCase;

public class HelloWorldTest extends ConcordionTestCase {

    public String getGreeting() {
        return "Hello World!";
    }
}
```


Mozilla Firefox
file:///users/olivierliechti/Downloads/concordion-extensions-demo/target/concordion/org/concordion/ext/demo/
Most Visited
novaccess
Pin It
Confluence Wiki ...
Billardshop
Nobits AMS RES...
Upcoming Meet...
JSONLint - The J...
Bookmarks
file:///users/o...nshotDemo.html
The Ultimate Question
What do you get if you multiply six by nine?
Example
When I google "6 * 9" the answer should be "6 * 9 = 42 6 * 9 = 54".
(Hover and click on the red failure to see the screenshot.)
assertion is violated during selenium session
Results generated by concordion
in 5s on 07-Nov-2012 at 11:07:36 CET
snapshot recorded during selenium session is displayed on mouse-over

jbehave

1. Write story

Plain text

```
Scenario: A trader is alerted of status

Given a stock and a threshold of 15.0
When stock is traded at 5.0
Then the alert status should be OFF
When stock is traded at 16.0
Then the alert status should be ON
```

2. Map steps to Java

POJO

```
public class TraderSteps {
    private TradingService service;   // Injected
    private Stock stock; // Created

    @Given("a stock and a threshold of $threshold")
    public void aStock(double threshold) {
        stock = service.newStock("STK", threshold);
    }
    @When("the stock is traded at price $price")
    public void theStockIsTraded(double price) {
        stock.tradeAt(price);
    }
    @Then("the alert status is $status")
    public void theAlertStatusIs(String status) {
        assertThat(stock.getStatus().name(), equalTo(status));
    }
}
```

jbehave

3. Configure Stories

Only once

```
public class TraderStories extends JUnitStories {

    public Configuration configuration() {
        return new MostUsefulConfiguration()
            .useStoryLoader(new LoadFromClasspath(this.getClass()))
            .useStoryReporterBuilder(new StoryReporterBuilder()
                .withCodeLocation(codeLocationFromClass(this.getClass()))
                .withFormats(CONSOLE, TXT, HTML, XML));
    }

    public List<CandidateSteps> candidateSteps() {
        return new InstanceStepsFactory(configuration(),
              new TraderSteps(new TradingService())).createCandidateSteps();
    }

    protected List<String> storyPaths() {
        return new StoryFinder().findPaths(codeLocationFromClass(this.getClass()),
"**/*.story");
    }
}
```

4. Run Stories

With any of

<APACHE ANT>

eclipse THE ECLIPSE PROJECT

JUnit

IntelliJIDEA

maven

jbehave

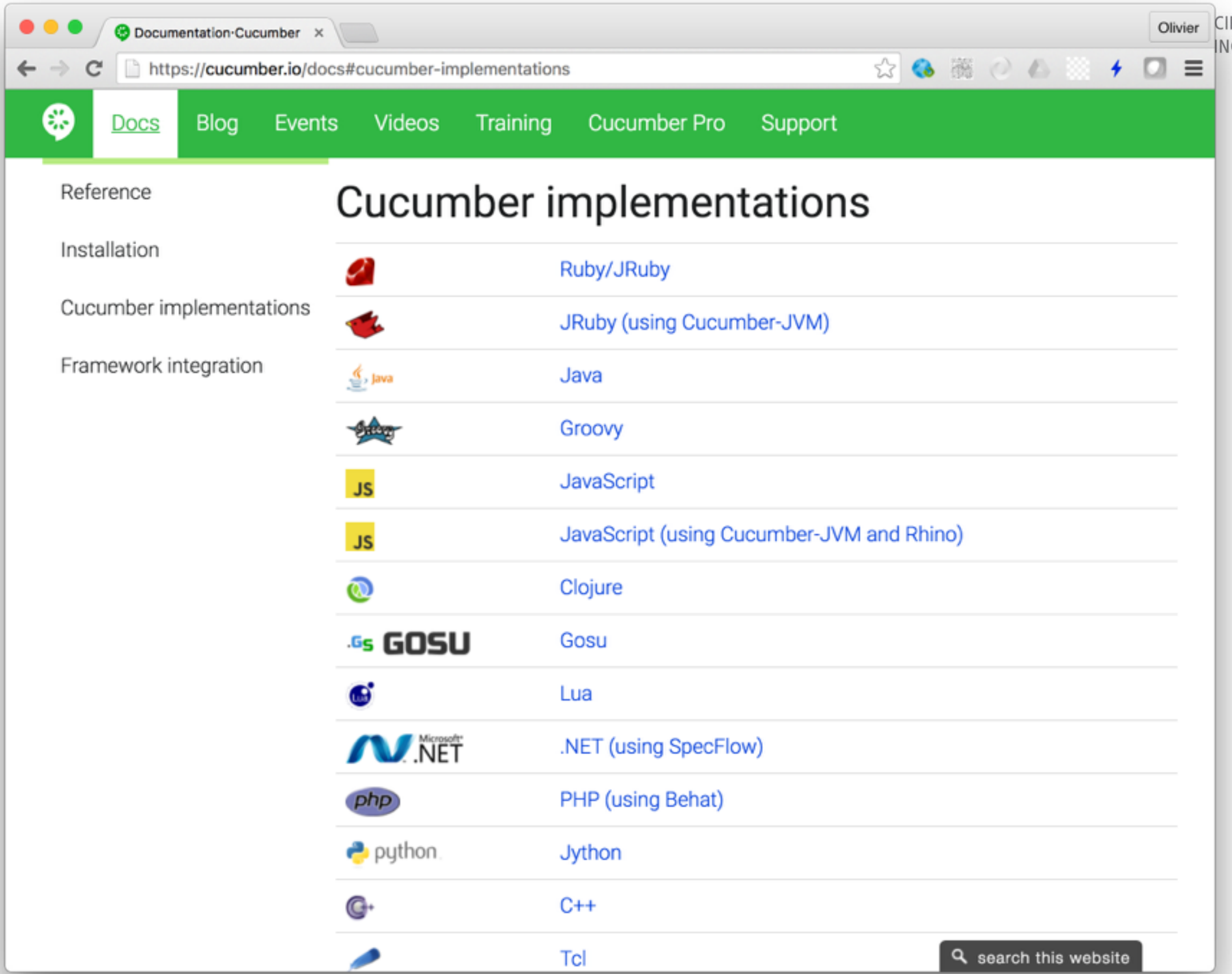
5. View Reports

HTML

Scenario: A trader is alerted of status

Given a stock and a threshold of 15.0
When stock is traded at 5.0
Then the alert status is OFF
When stock is traded at 16.0
Then the alert status is ON

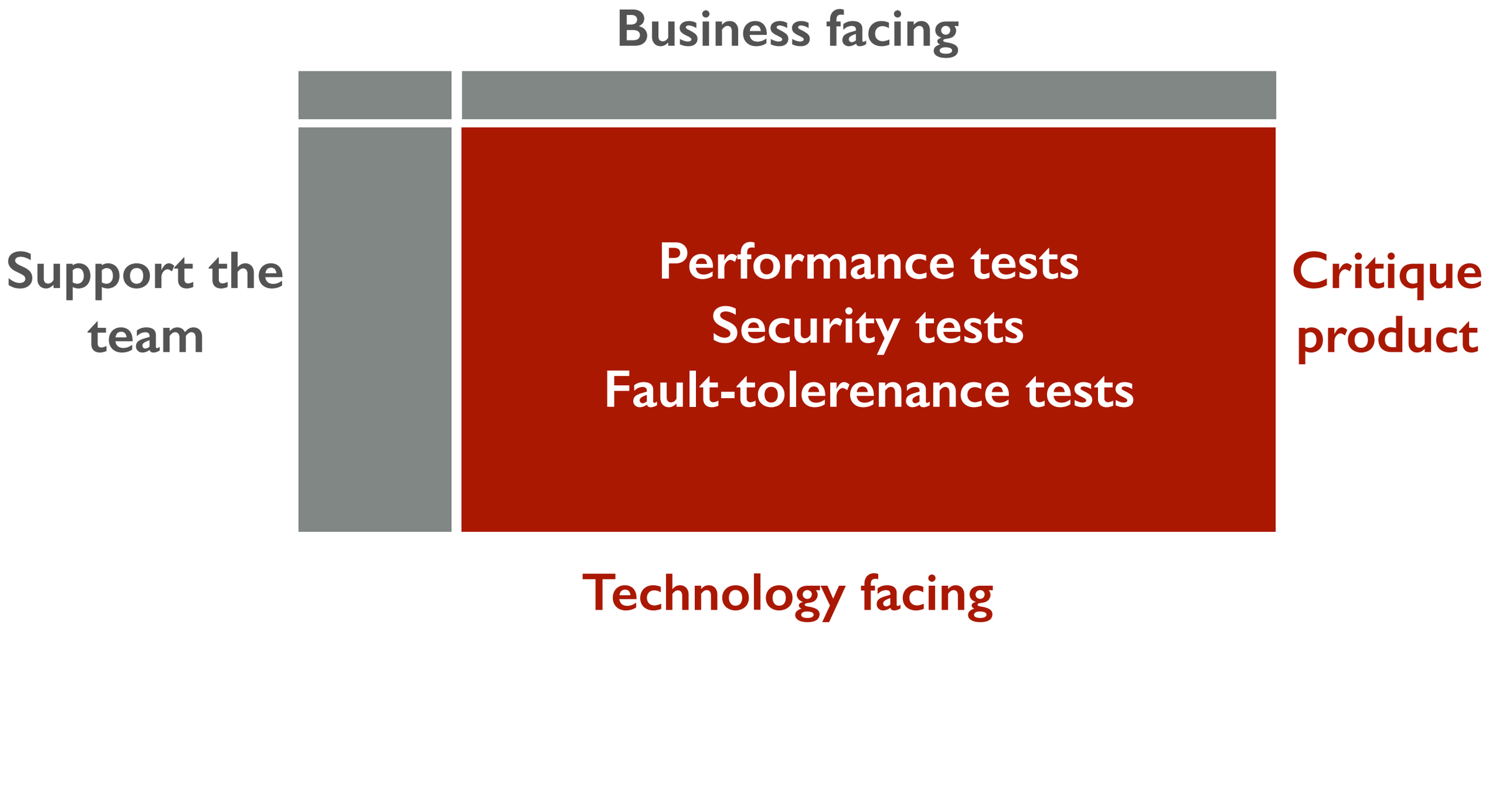

Cucumber
Olivier
https://cucumber.io
Docs
Blog
Events
Videos
Training
Cucumber Pro
Support
cucumber
Simple, human collaboration
CukeUp! AU - Sydney, Australia - 17 & 18 Nov 2016
An open-source tool for executable specifications
A vibrant community
An ingenious company
search this website


Documentation·Cucumber
Olivier
https://cucumber.io/docs#cucumber-implementations
Docs
Blog
Events
Videos
Training
Cucumber Pro
Support
Reference
Installation
Cucumber implementations
Framework integration
Cucumber implementations
Ruby/JRuby
JRuby (using Cucumber-JVM)
Java
Groovy
JavaScript
JavaScript (using Cucumber-JVM and Rhino)
Clojure
Gosu
Lua
.NET (using SpecFlow)
PHP (using Behat)
Jython
C++
Tcl
search this website

```
# features/my_feature.feature

Feature: Example feature
  As a user of Cucumber.js
  I want to have documentation on Cucumber
  So that I can concentrate on building awesome applications

  Scenario: Reading documentation
    Given I am on the Cucumber.js GitHub repository
    When I go to the README file
    Then I should see "Usage" as the page title
```

```
module.exports = function () {
  this.Given(/^I am on the Cucumber.js GitHub repository$/, function (callback) {
    this.visit('https://github.com/cucumber/cucumber-js', callback);
 });

  this.When(/^I go to the README file$/, function (callback) {
    callback(null, 'pending');
  });

  this.Then(/^I should see "(.*)" as the page title$/, function (title, callback) {
   var pageTitle = this.browser.text('title');
    if (title === pageTitle) {
      callback();
    } else {
      callback(new Error("Expected to be on page with title " + title));
    }
  });
};
```

Agile testing quadrants: Q4



Systemic qualities

- A common problem is that developers focus on the functionality (**what** the software does), but forget about the non-functional aspects (**how** it does it).
- Things like performance, availability, security, scalability, etc.
- We often talk about “**systemic qualities**” or “-ilities”

A sad story...

Developer	I have **modified** the "search feature", so that it now ranks the results based on the user profile
Product Owner	Cool, let's **release** it!
User	Why is it search feature so **slow** today?
System Administrator	Why is all the **CPU** and **memory** in red on the server?

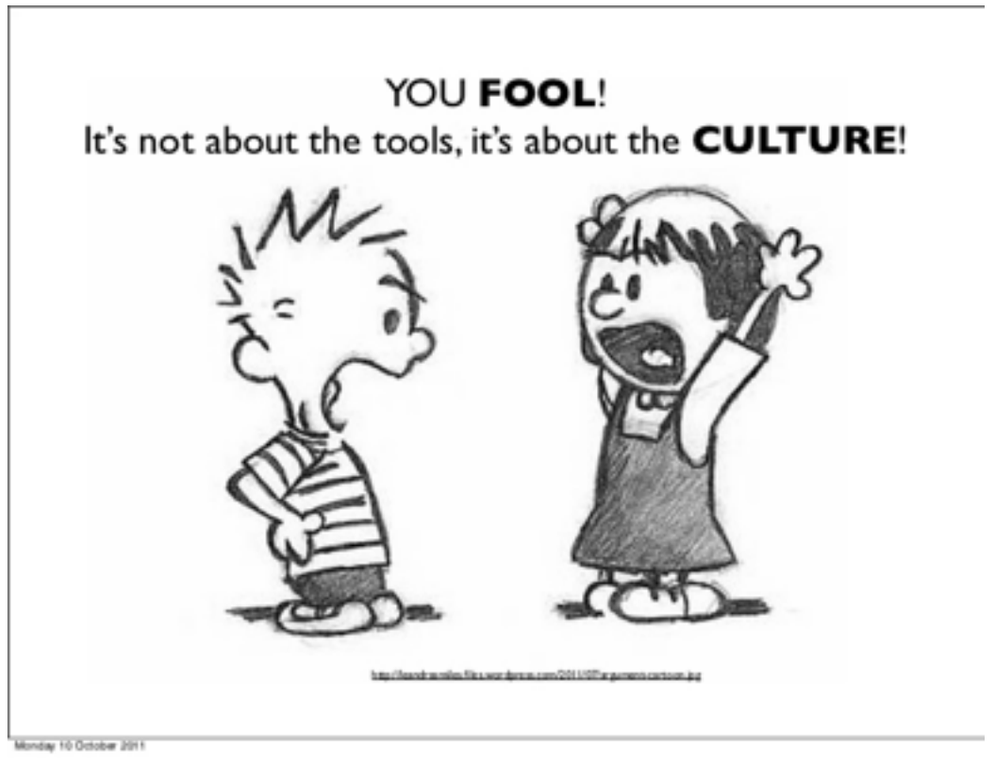
*An algorithm may work well if it is applied on a database that contains **1'000** entries. But it may break if it contains **1'000'000** entries.*

*A feature may be fast if it is used by a single user at the time. But it may extremely slow if is used by 10 **concurrent users**. It may even return wrong results in this case.*

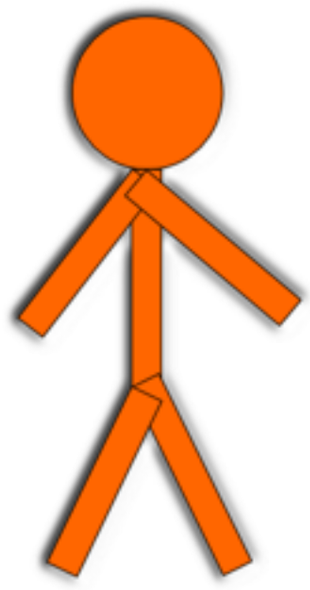
*A web app may appear to be fast when the browser, the server and the database are all on the same (developer) machine. But may appear very slow when the components are **distributed across the Internet**.*

How to avoid these issues (1)

- Increase **team awareness**
  - Developers and product owners should be reminded about "**systemic qualities**" and the importance of **non-functional aspects**.
  - With a "**whole-team approach**", you are more likely to hear the point of view of system administrators very early. This will help a lot.



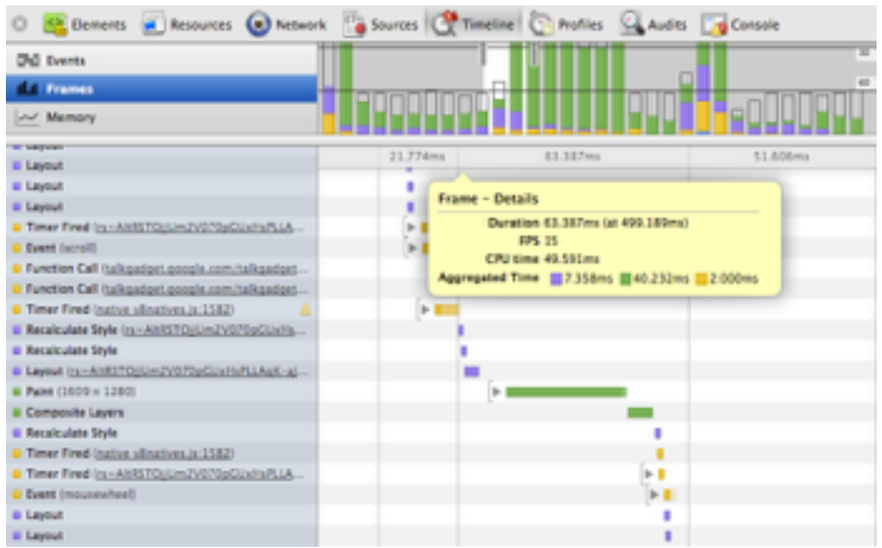
http://codeignition.co/blog/2015/09/17/dev_to_ops/



The System Administrator

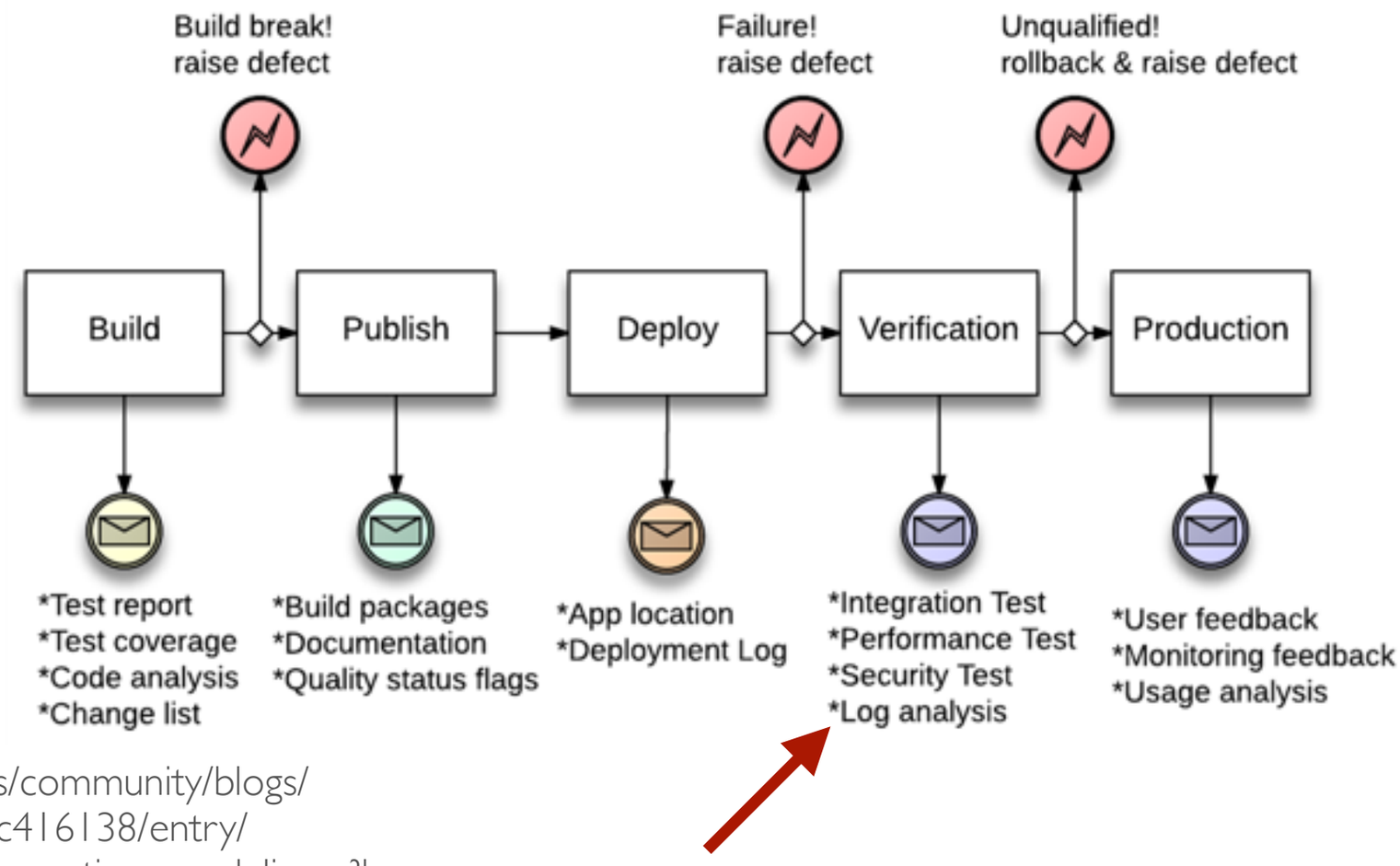
How to avoid these issues (2)

- **Measure** early, measure often
  - There are **many tools to measure performance** (response time) of a software system. There are also **many tools that simulate a large number of users** and to generate a heavy load on a test system.
  - Engineers can also **write custom programs and scripts** to do these experiments.
- **Allocate time and resources** to support these activities.



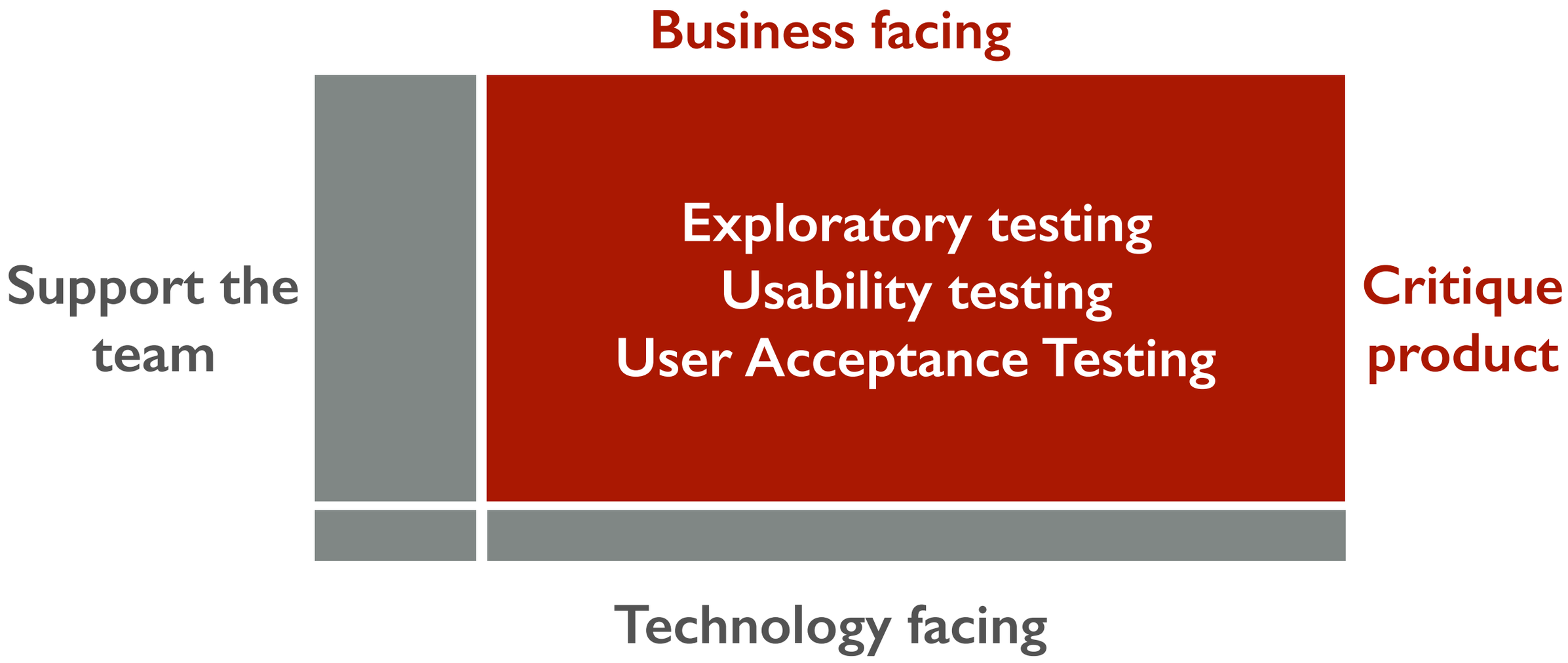
How to avoid these issues (3)

- Integrate automated performance tests in your **continuous delivery pipeline**.
- Performance is only one aspect, you should also think about security, fault tolerance, etc.



https://www.ibm.com/developerworks/community/blogs/c914709e-8097-4537-92ef-8982fc416138/entry/devops_in_practice_best_practices_for_adopting_continuous_delivery?lang=en

Agile testing quadrants: Q3



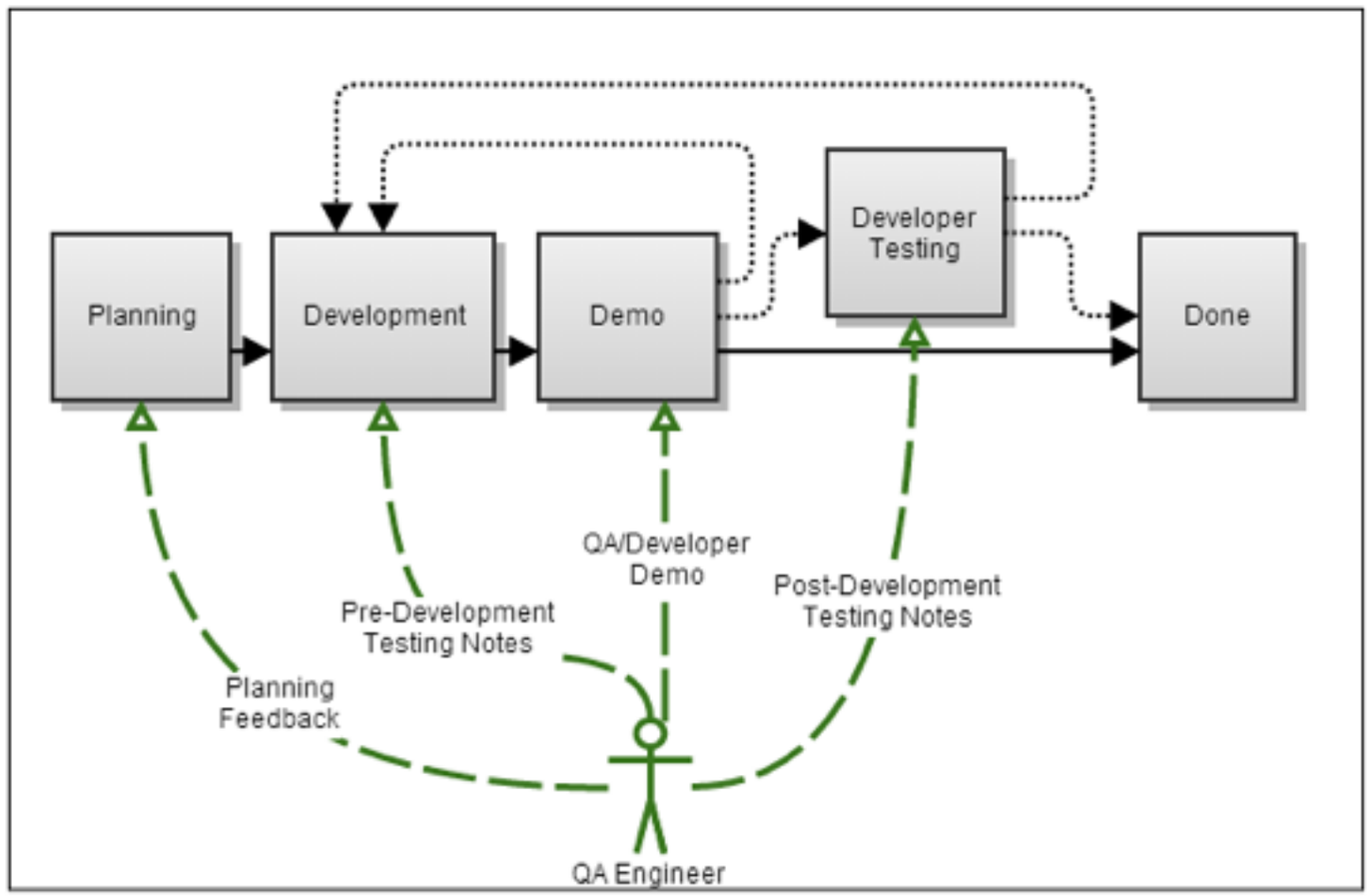
Exploratory testing

- Exploratory testing is a **human activity** (which may involve the use of automated tests).
- Unlike scripted manual testing, it requires **strong skills and expertise**.
- It fits well with the "**whole-team**" approach (handing over the system to an external testing team is not efficient).

*"A style of software testing that emphasizes the **personal freedom and responsibility** of the individual tester to continually optimize the quality of his/her work by treating **test-related learning**, **test design**, **test execution**, and **test result interpretation** as **mutually supportive activities** that run in parallel throughout the project."*

Cem Kaner

Example: JIRA development process



http://blogs.atlassian.com/2013/12/jira-qa-process/

Example: JIRA development process

"Once a developer is happy that they have completed a story, he or she will call over their QA engineer for a ***demo session****.*

The demo is ***a discussion between equals****, and not a test that the developer or story can "pass" or "fail." However, sometimes concerns are noticed during the demo, and these get quickly noted as comments on the story.*

At the end of the demo, the QA engineer and developer will make a ***joint decision*** *on what should happen to the story next."*

Example: JIRA development process

"The Atlassian QA strategy is more about prevention and trust.

As QA, we know the roads really well – we know what causes accidents, and where the accidents have been in the past. ***So instead of sitting by the road with a radar gun to catch speeding drivers, we sit in the cab with the drivers and say, "careful in this next bit, there's a huge rock in the middle of the road after that blind corner."*** *In safer stretches, we say, "There've been no nasty accidents in this stretch in the past, full speed ahead!""*

http://blogs.atlassian.com/2014/01/atlassian-qa-makes-development-faster/

Where did we leave our pipeline?

SEA Build Pipeline [Jenkins

Olivier

192.168.99.100:18080/job/SEA%20Build%20Pipeline/

Jenkins

search

Jenkins › SEA Build Pipeline ›

ENABLE AUTO REFRESH

- Back to Dashboard
- Status
- Changes
- Build with Parameters
- Delete Pipeline
- Configure
- Move
- Full Stage View

Build History trend

find x

#1
May 19, 2016 11:32 AM

RSS for all RSS for failures

Pipeline SEA Build Pipeline

add description

Recent Changes

Stage View

	Setup	Commit	Validation	End
Average stage times: (Average full run time: ~71ms)	2ms	3ms	5ms	61ms
#1 May 19 13:32 — No Changes	2ms master	3ms master	5ms master	61ms master

Permalinks

- Last build (#1), 13 sec ago
- Last stable build (#1), 13 sec ago
- Last successful build (#1), 13 sec ago
- Last completed build (#1), 13 sec ago

Help us localize this page

Page generated: May 19, 2016 11:32:30 AM UTC REST API Jenkins ver. 1.651.1

SEA Build Pipeline [Jenkins

Olivier

192.168.99.100:18080/job/SEA%20Build%20Pipeline/

Jenkins

search

Jenkins › SEA Build Pipeline ›

ENABLE AUTO REFRESH

- Back to Dashboard
- Status
- Changes
- Build with Paramet
- Delete Pipeline
- Configure
- Move
- Full Stage View

Stage Logs (Commit)

Print Message

```
We don't know what to do in the commit stage...
```

add description

Stage View

Setup	Commit	Validation	End
Average stage times: (Average full run time: ~71ms)	2ms	5ms	61ms
#1 May 19 13:32 — No Changes	2ms	3ms	5ms	61ms

Success — Logs

Build History trend

find

#1

May 19, 2016 11:32 AM

RSS for all RSS for failures

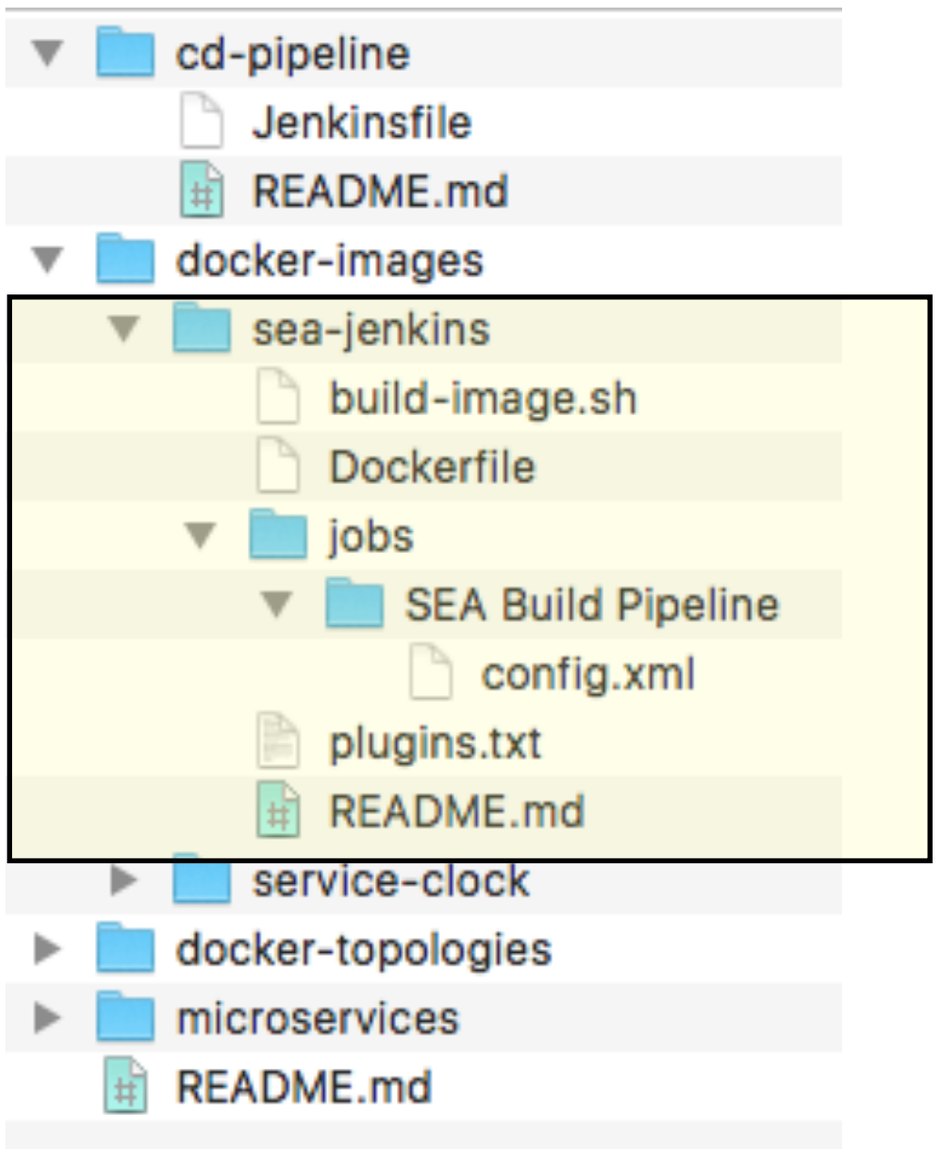
Permalinks

- Last build (#1), 13 sec ago
- Last stable build (#1), 13 sec ago
- Last successful build (#1), 13 sec ago
- Last completed build (#1), 13 sec ago

Help us localize this page

Page generated: May 19, 2016 11:32:30 AM UTC REST API Jenkins ver. 1.651.1

Custom Docker image for Jenkins

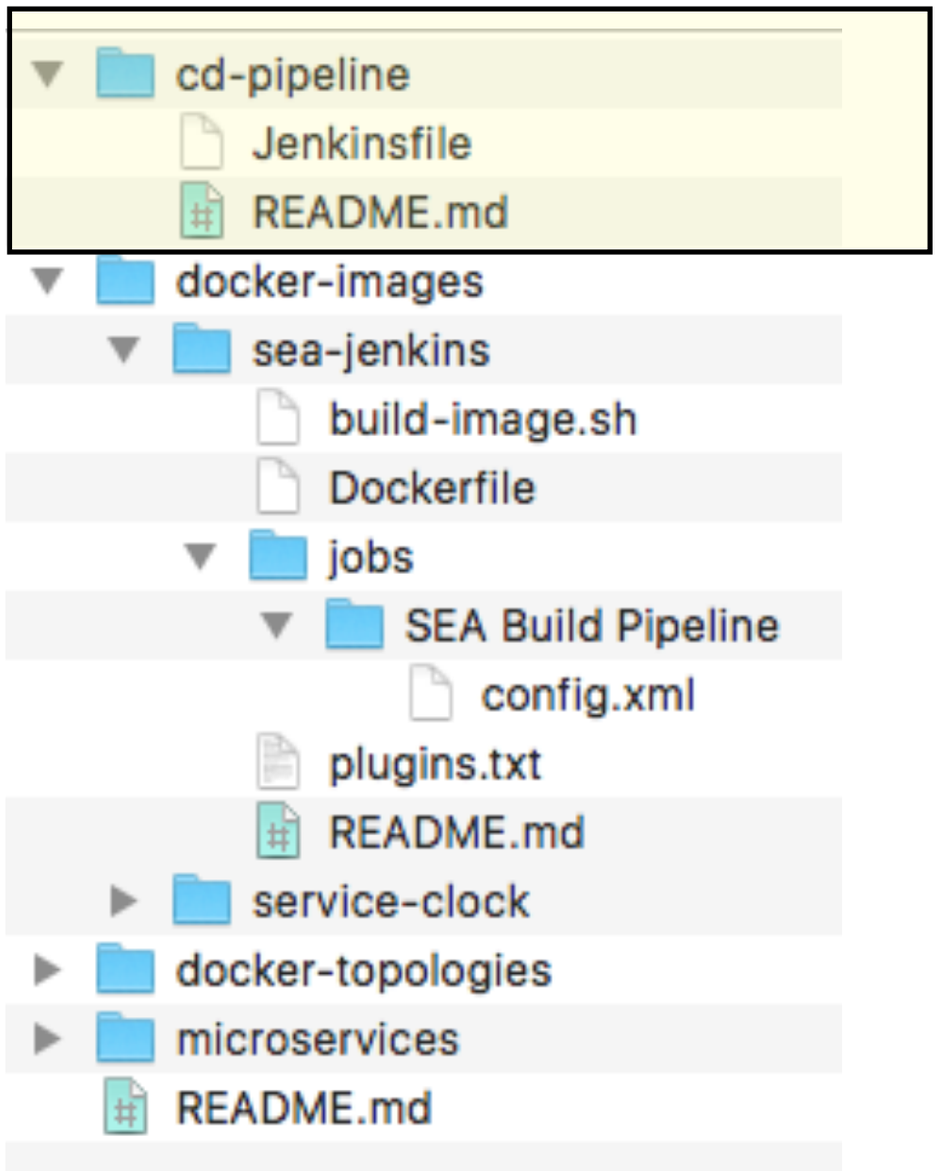


In the Dockerfile, we have defined the steps to build our own version of Jenkins, ready to use.

We have installed a list of optional plugins (in particular the pipeline and stage view plugins).

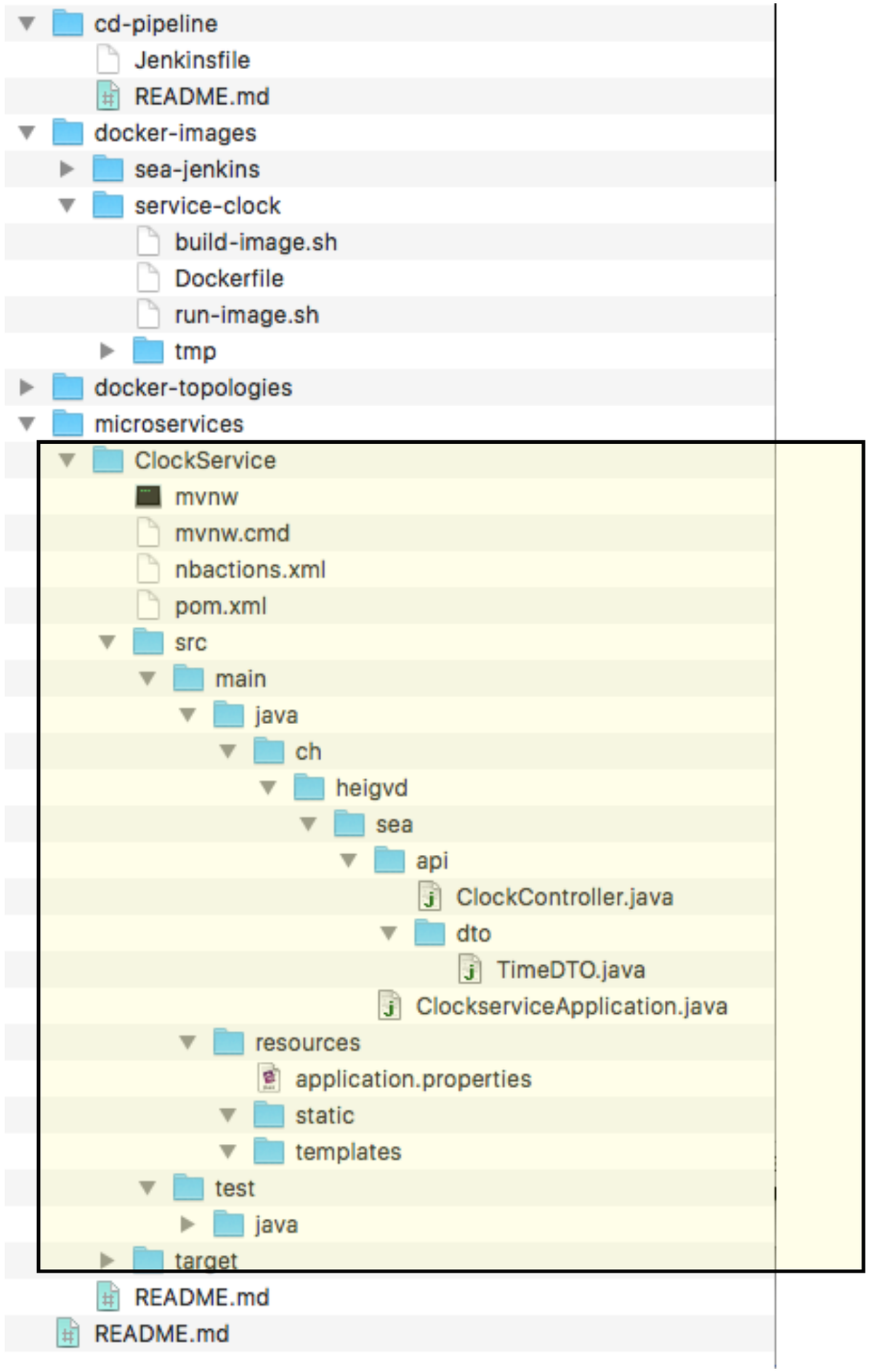
We have copied jobs/SEA Build Pipeline/config.xml into the image file system. This file defines the Jenkins job. It is not a “maven” job, it is a “pipeline job”. In the config.xml, we have specified where Jenkins can fetch the Jenkinsfile script, which defines the pipeline steps (i.e. we have specified in which git repo Jenkinsfile is stored).

Jenkinsfile dynamically fetched when pipeline starts



We have created a first version, almost empty, of the Jenkinsfile. We could have stored it in another git repo, but we have decided to keep it in the same repo as the whole project.

```
node {
        /*-----------------------------------------------------------------------------*/
        stage 'Setup'
        /*-----------------------------------------------------------------------------*/

         // We have received the IP of the Docker host from a Jenkins job parameter
        echo "We will contact our micro-service via: ${MICROSERVICE_URL}"

        /*-----------------------------------------------------------------------------*/
        stage 'Commit'
        /*-----------------------------------------------------------------------------*/

        echo "We don't know what to do in the commit stage..."

        /*-----------------------------------------------------------------------------*/
        stage 'Validation'
        /*-----------------------------------------------------------------------------*/

        echo "We don't know what to do in the validation stage..."

        /*-----------------------------------------------------------------------------*/
        stage 'End'
        /*-----------------------------------------------------------------------------*/

        echo "We have been through the entire pipeline"

}
```



We have implemented a first, very simple micro-service with Spring Boot.

When we are in the ClockService directory, we can do a **mvn clean install** to build the executable jar.

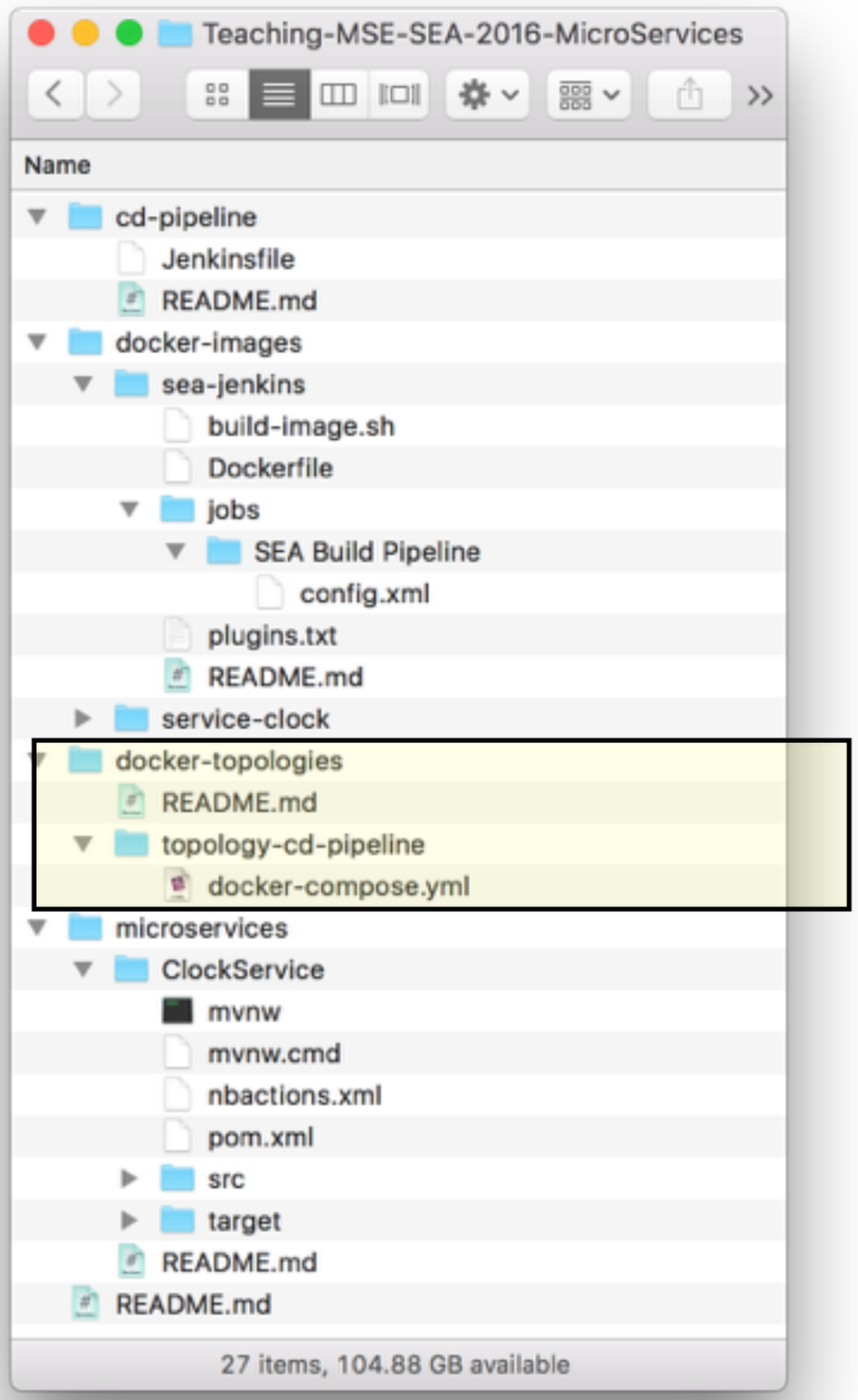
We can then run the app with **java -jar target/*xxx*.jar**.

What’s next for our pipeline?

Next steps

- We know that at some point, we will want to launch (at least) one Docker container for each one of our micro-services. Some micro-services will depend on some kind of database, also running in a container.
    - Is there a way to manage these “topologies” of Docker containers, instead of doing a “docker run” on every single container? Is there a way to declare dependencies between containers?
    - Yes, there is. Docker Compose a tool that will allow us to do just that. As a first step, we will define a topology for our custom Jenkins container (a topology with a single container).
- We also want to actually get the code for our micro-services from GitHub, build them with maven, create a new image.
    - To do that, we have to do a bit of additional setup, so that we have a maven installation available.

Docker Compose



We have a folder that will contain one sub-folder for every docker topology.

Today, we create a first one for our pipeline.

We will have an alternative way to get our Jenkins server up and running. No more “docker run -p 18080:8080 sea/jenkins” after that.

Docker Compose

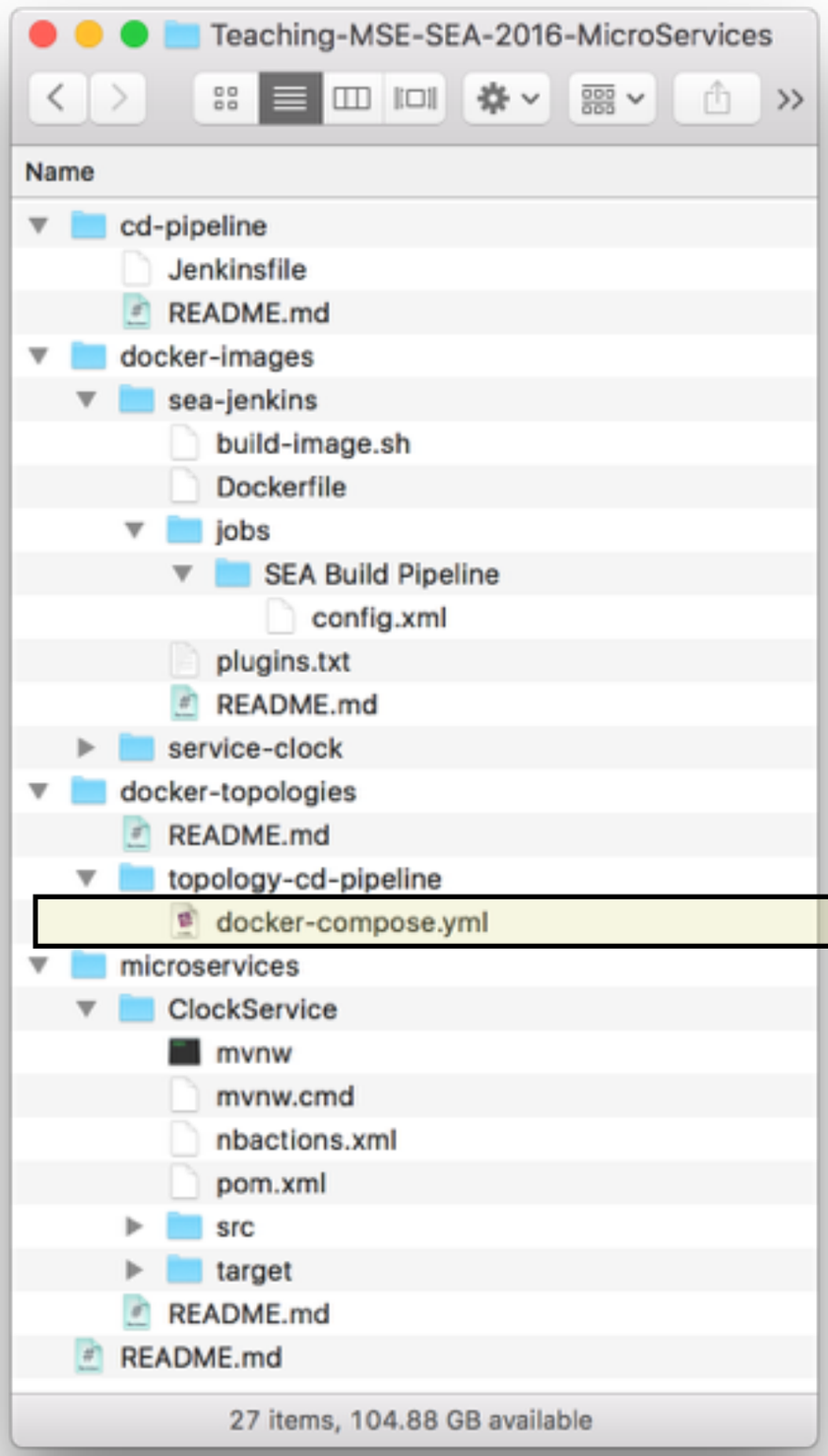
- “Compose is a tool for defining and running multi-container Docker applications.
- With Compose, you use a Compose file to configure your application’s services.
- Then, using a single command, you create and start all the services from your configuration.”

1. Define your app's environment with a `Dockerfile` so it can be reproduced anywhere.
2. Define the services that make up your app in `docker-compose.yml` so they can be run together in an isolated environment.
3. Lastly, run `docker-compose up` and Compose will start and run your entire app.

A `docker-compose.yml` looks like this:

```
version: '2'
services:
  web:
    build: .
    ports:
    - "5000:5000"
    volumes:
    - .:/code
    - logvolume01:/var/log
    links:
    - redis
  redis:
    image: redis
volumes:
  logvolume01: {}
```

Docker Compose

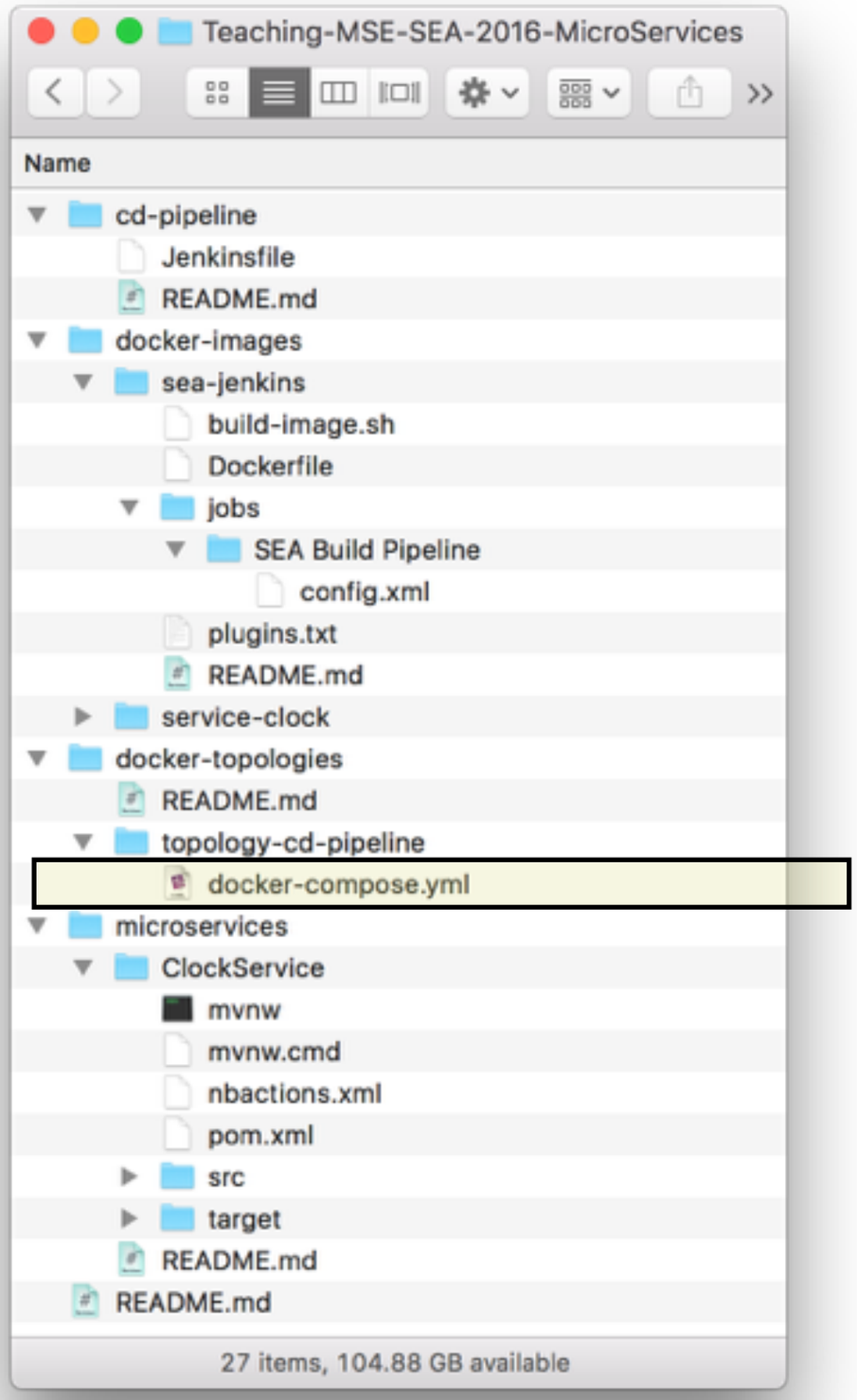



```
version: '2'
services:
  jenkins:
    build: ../../docker-images/sea-jenkins
    image: sea/jenkins
    restart: always
    dns: "8.8.8.8"
    ports:
     - "50000:50000"
     - "18080:8080"
    volumes:
#     - jenkins-data:/var/jenkins_home
     - /var/run/docker.sock:/var/run/docker.sock
```

```
$ docker-compose up
```

```
$ docker-compose --help
```

Docker Compose

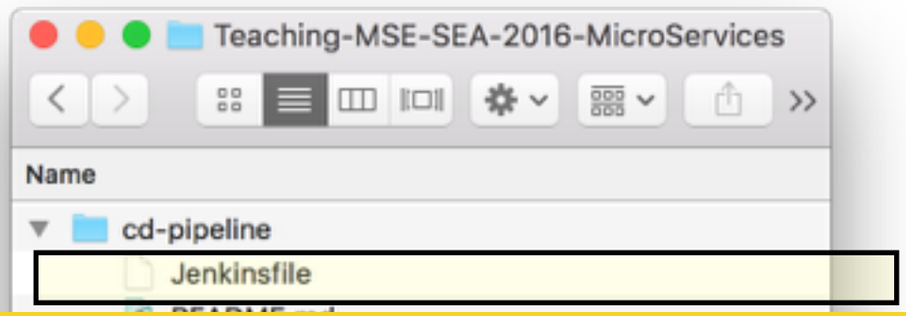



```
version: '2'
services:
  jenkins:
    build: ../../docker-images/sea-jenkins
    image: sea/jenkins
    restart: always
    dns: "8.8.8.8"
    ports:
     - "50000:50000"
     - "18080:8080"
    volumes:
#     - jenkins-data:/var/jenkins_home
     - /var/run/docker.sock:/var/run/docker.sock
```

I had issues on the HESSO network without this DNS parameter

This allows the Docker client running ***in*** the Jenkins container to communicate with the Docker daemon running outside (in the VM). This will allow us to start containers during the pipeline execution.

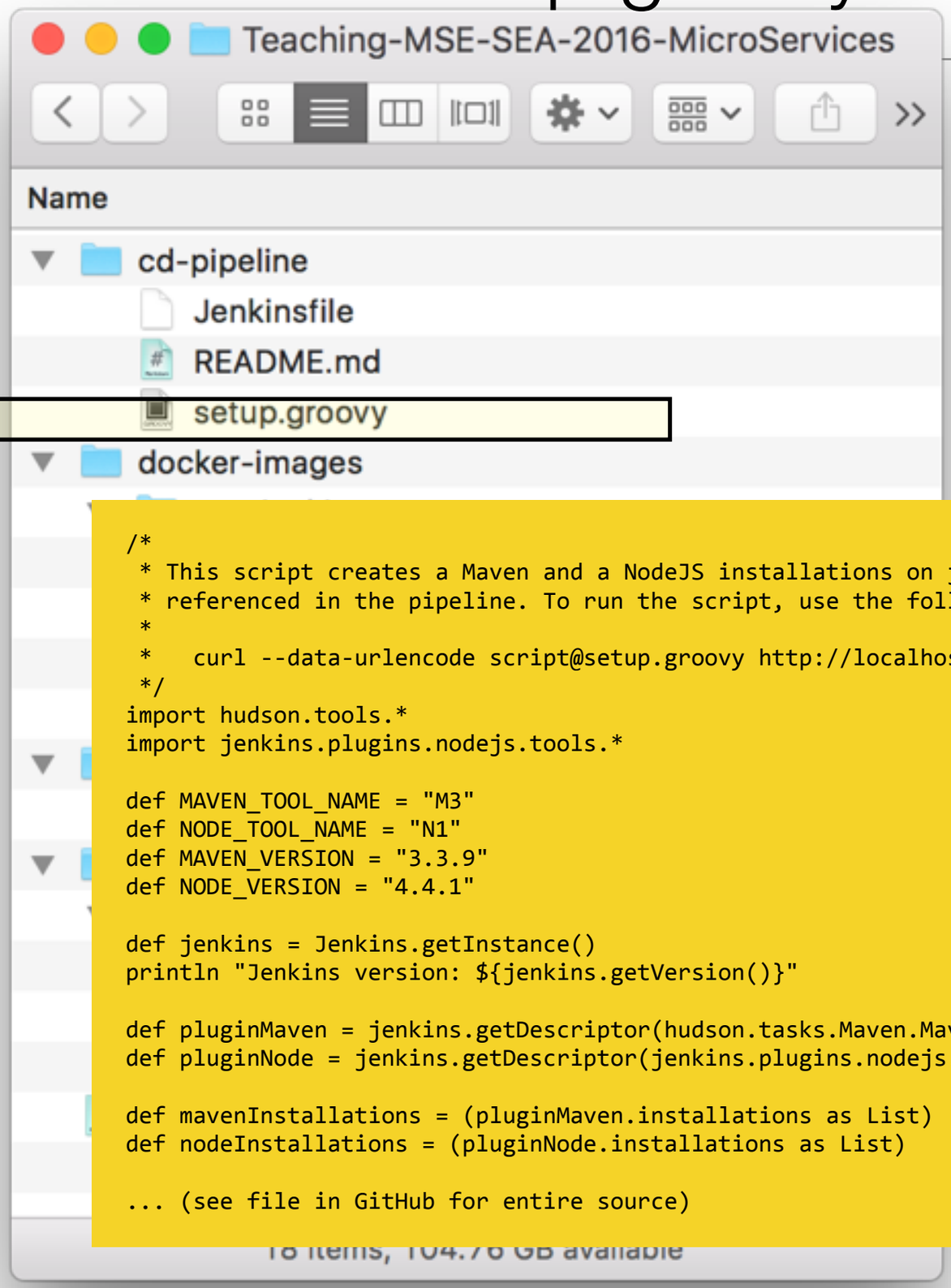
Docker Compose



Jenkins exposes a REST API, allowing us to send a groovy script that is executed by the server. We use this feature to configure the environment (add a maven installation) when the pipeline starts.

```
/*-----------------------------------------------------------------------*/
stage 'Setup'
/*-----------------------------------------------------------------------*/
 // We have received the IP of the Docker host from a Jenkins job parameter
echo "We will contact our micro-service via: ${MICROSERVICE_URL}"

// Get some code from a GitHub repository
git url: 'https://github.com/wasadigi/Teaching-MSE-SEA-2016-MicroServices.git'

// Ask Jenkins to execute a groovy script to setup the Maven and NodeJS tools (could not do
   it in Jenkinsfile because of the sandbox)
dir('cd-pipeline') {
     sh 'curl --data-urlencode script@setup.groovy http://localhost:8080/scriptText'
}
// Get a ref to the Maven and Node tools configured in Jenkins (so far, they need to be
    configured manually after booting the Docker container!)
def mvnHome = tool 'M3'
env.PATH = "${mvnHome}/bin:${env.PATH}"
def npmHome = tool 'N1'
env.PATH = "${npmHome}/bin:${env.PATH}"
```

Invoke a setup.groovy script to configure maven



You might remember that earlier during the semester, we had a manual configuration step, where we had to add a “maven installation” in Jenkins.

This script does that automatically. It also adds a “Node.js” installation, which we will use late.

```
/*
 * This script creates a Maven and a NodeJS installations on jenkins, so that they can be
 * referenced in the pipeline. To run the script, use the following command:
 *
 *   curl --data-urlencode script@setup.groovy http://localhost:8080/scriptText
 */
import hudson.tools.*
import jenkins.plugins.nodejs.tools.*

def MAVEN_TOOL_NAME = "M3"
def NODE_TOOL_NAME = "N1"
def MAVEN_VERSION = "3.3.9"
def NODE_VERSION = "4.4.1"

def jenkins = Jenkins.getInstance()
println "Jenkins version: ${jenkins.getVersion()}"

def pluginMaven = jenkins.getDescriptor(hudson.tasks.Maven.MavenInstallation)
def pluginNode = jenkins.getDescriptor(jenkins.plugins.nodejs.tools.NodeJSInstallation)

def mavenInstallations = (pluginMaven.installations as List)
def nodeInstallations = (pluginNode.installations as List)

... (see file in GitHub for entire source)
```