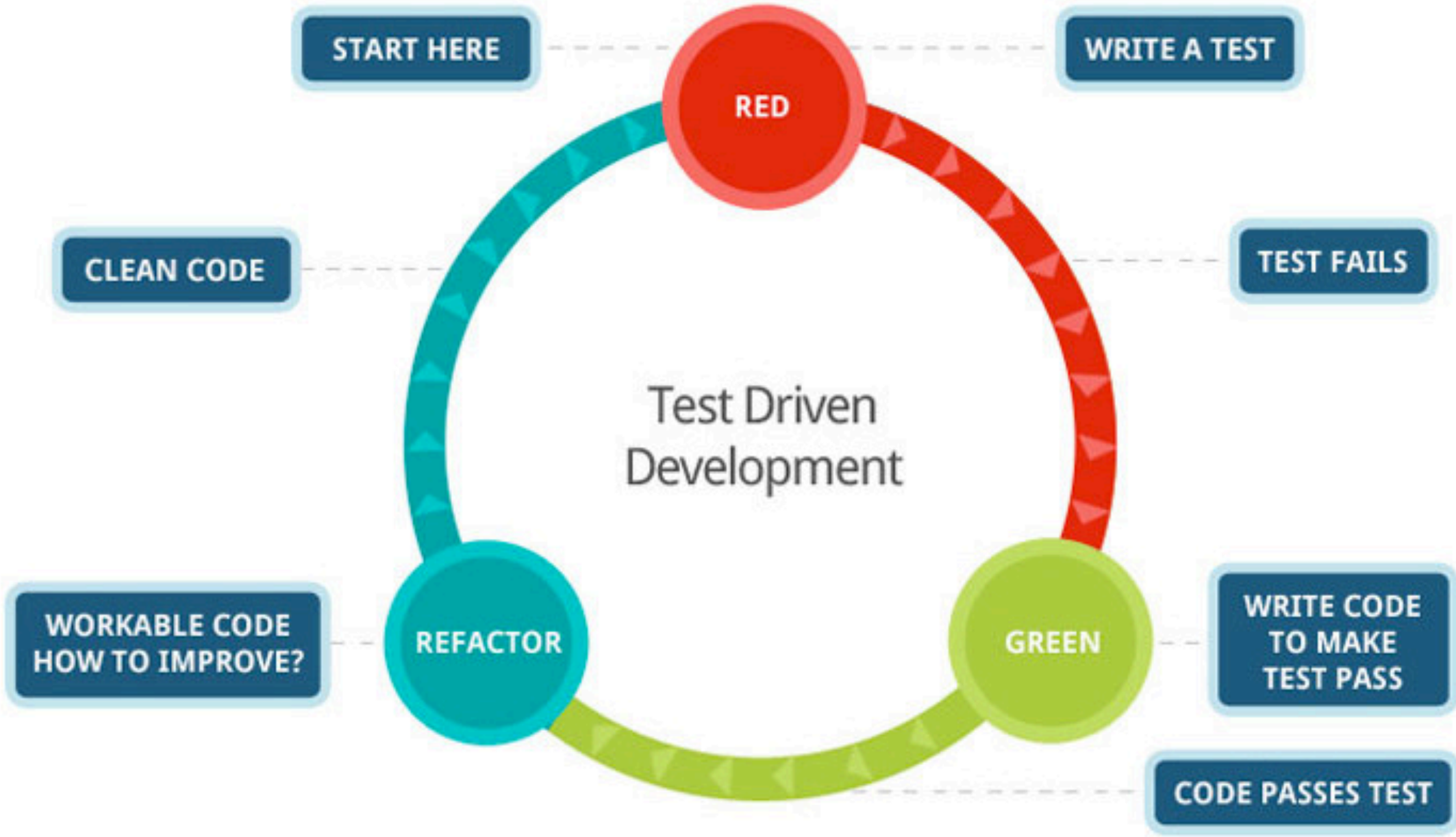

TDD with Java

# Workshop day 2

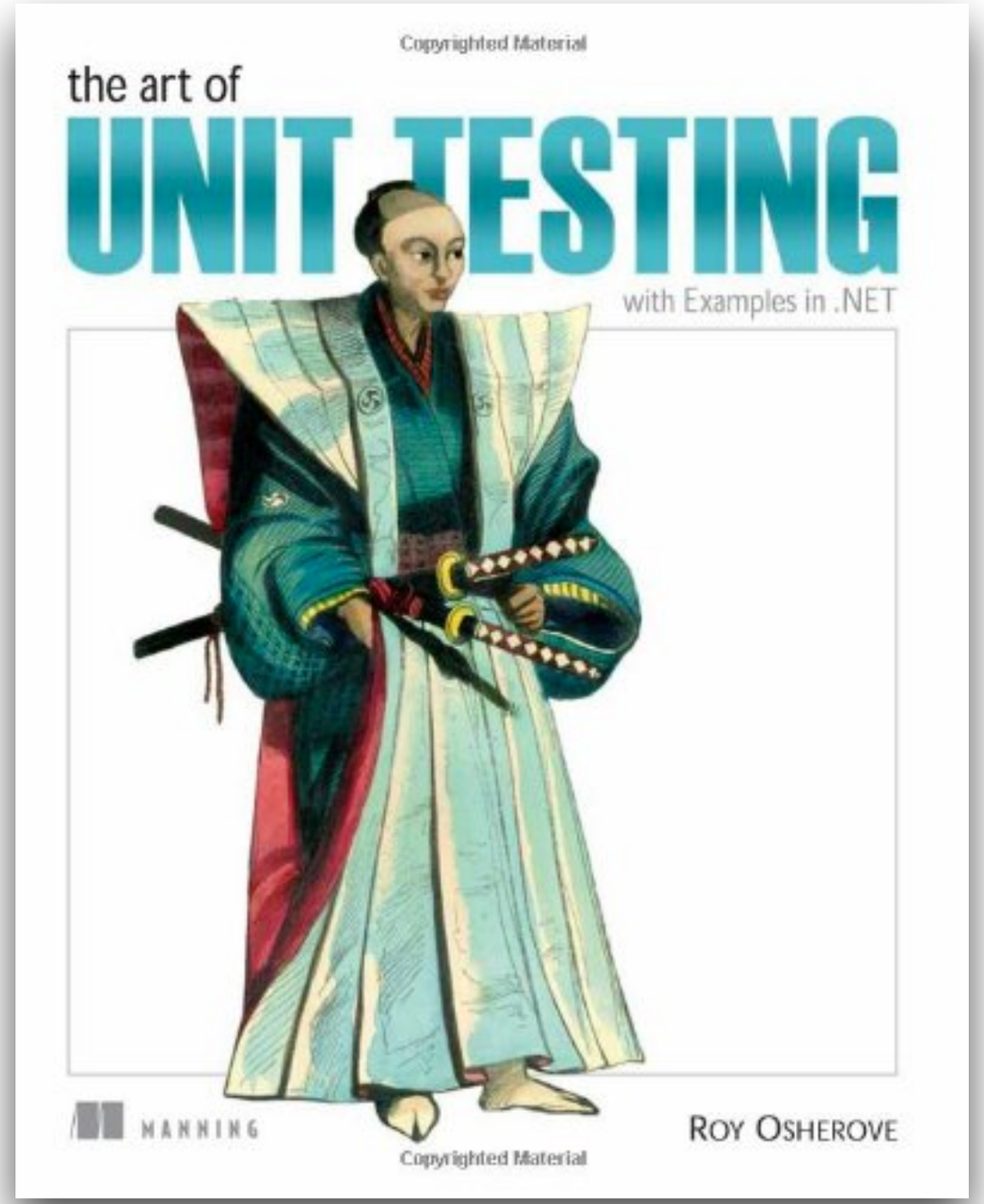
Legacy workshop

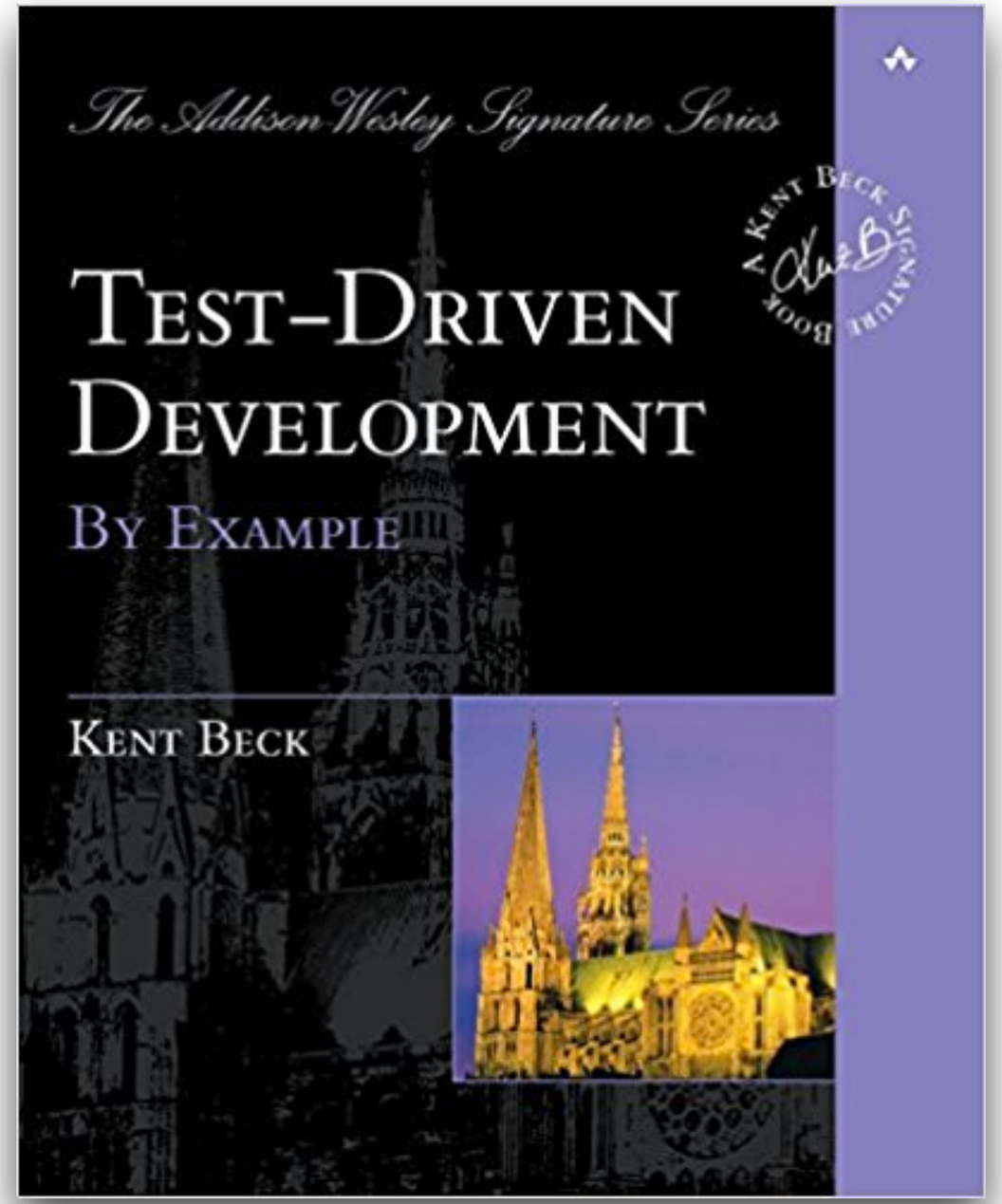
December 13, 2017

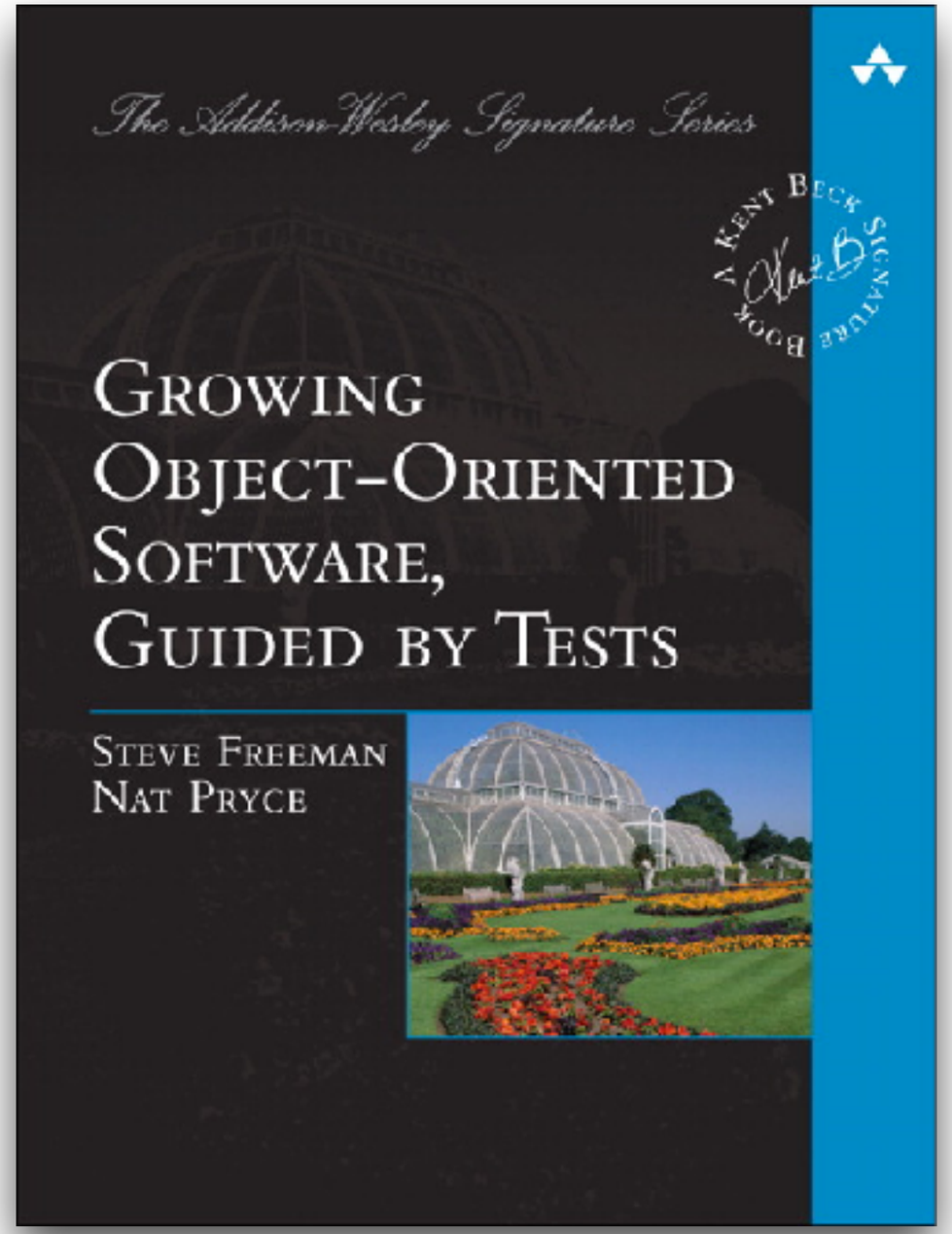
# Test-Driven Development

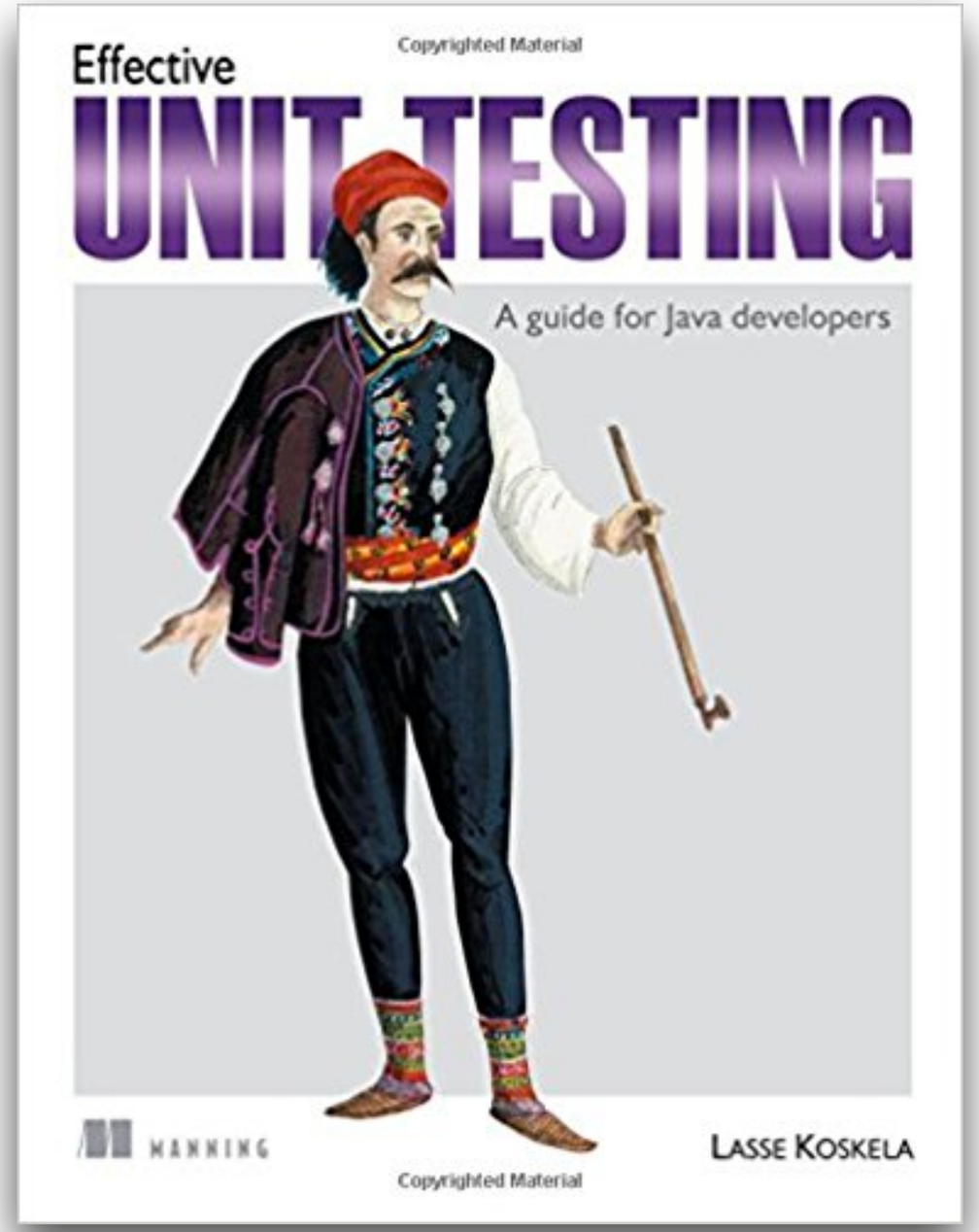



http://haselt.com/coding-dojo-with-tdd/





# SOLID workshop

# FizzBuzz

| No. | Input | Expected Result |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 2 | 2 |
| 3 | 3 | Fizz |
| 4 | 4 | 4 |
| 5 | 5 | Buzz |
| 6 | 6 | Fizz |
| 7 | 7 | 7 |
| 8 | 8 | 8 |
| 9 | 9 | Fizz |
| 10 | 10 | Buzz |
| 11 | 15 | FizzBuzz |
| 12 | 30 | FizzBuzz |

# Refactoring Production code

# Refactoring Test code

# Fizz

| No. | Input | Expected Result |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 2 | 2 |
| 3 | 3 | **Fizz** |
| 4 | 4 | 4 |
| 5 | 5 | Buzz |
| 6 | 6 | **Fizz** |
| 7 | 7 | 7 |
| 8 | 8 | 8 |
| 9 | 9 | Fizz |
| 10 | 10 | Buzz |
| 11 | 15 | FizzBuzz |
| 12 | 30 | FizzBuzz |

# Buzz

| No. | Input | Expected Result |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 2 | 2 |
| 3 | 3 | Fizz |
| 4 | 4 | 4 |
| 5 | 5 | **Buzz** |
| 6 | 6 | Fizz |
| 7 | 7 | 7 |
| 8 | 8 | 8 |
| 9 | 9 | Fizz |
| 10 | 10 | **Buzz** |
| 11 | 15 | FizzBuzz |
| 12 | 30 | FizzBuzz |

# FizzBuzz

| No. | Input | Expected Result |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 2 | 2 |
| 3 | 3 | Fizz |
| 4 | 4 | 4 |
| 5 | 5 | Buzz |
| 6 | 6 | Fizz |
| 7 | 7 | 7 |
| 8 | 8 | 8 |
| 9 | 9 | Fizz |
| 10 | 10 | Buzz |
| 11 | 15 | **FizzBuzz** |
| 12 | 30 | **FizzBuzz** |

# Business rules

Fizz = หาร 3 ลงตัว
Buzz = หาร 5 ลงตัว
FizzBuzz = หาร 3 และ 5 ลงตัว
แสดงคำเดิม

# Data-Driven Testing

# Data

| No. | Input | Expected Result |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 2 | 2 |
| 3 | 3 | Fizz |
| 4 | 4 | 4 |
| 5 | 5 | Buzz |
| 6 | 6 | Fizz |
| 7 | 7 | 7 |
| 8 | 8 | 8 |
| 9 | 9 | Fizz |
| 10 | 10 | Buzz |
| 11 | 15 | FizzBuzz |
| 12 | 30 | FizzBuzz |

# Workshop

https://github.com/junit-team/junit4/wiki/parameterized-tests

# Step 1 : RunWith Parameterized

```
import org.junit.runner.RunWith;
import org.junit.runners.Parameterized;

@RunWith(Parameterized.class)
public class FizzBuzzWithDataTest {

}
```

# Step 2 : Setup data for testing

```java
@RunWith(Parameterized.class)
public class FizzBuzzWithDataTest {

    @Parameters
    public static Collection<Object[]> data() {
        return Arrays.asList(new Object[][] {
                { 1, "1" },
                { 2, "2" },
                { 3, "Fizz" }
        });
    }

}
```

# Step 3 : Create constructor

```
@RunWith(Parameterized.class)
public class FizzBuzzWithDataTest {

    private int input;
    private String expectedResult;

    public FizzBuzzWithDataTest(int input,
                                String expectedResult) {
        this.input = input;
        this.expectedResult = expectedResult;
    }
```

# Step 4 : Create test case

```
@RunWith(Parameterized.class)
public class FizzBuzzWithDataTest {

    @Test
    public void ทดสอบ() {
        FizzBuzz fizzBuzz = new FizzBuzz();
        String actualResult = fizzBuzz.say(this.input);
        assertEquals(this.expectedResult, actualResult);
    }
```

# More : Not use constructor

```
@RunWith(Parameterized.class)
public class FizzBuzzWithDataTest {

    @Parameter
    public int input;
    @Parameter(1)
    public String expectedResult;

    @Test
    public void ทดสอบ() {
        FizzBuzz fizzBuzz = new FizzBuzz();
        String actualResult = fizzBuzz.say(this.input);
        assertEquals(this.expectedResult, actualResult);
    }
```

# More : Improve test name

```
@Parameters(name = "{index}: say({0})={1}")
public static Collection<Object[]> data() {
        return Arrays.asList(new Object[][] {
                { 1, "1" },
                { 2, "2" },
                { 3, "Fizz" }
        });
}
```

# SOLID workshop

# เพิ่ม หาร 7 ลงตัวแสดง TDD

# กฎกติกา

ห้ามแก้ไข code เดิม (OCP)

# คำถาม

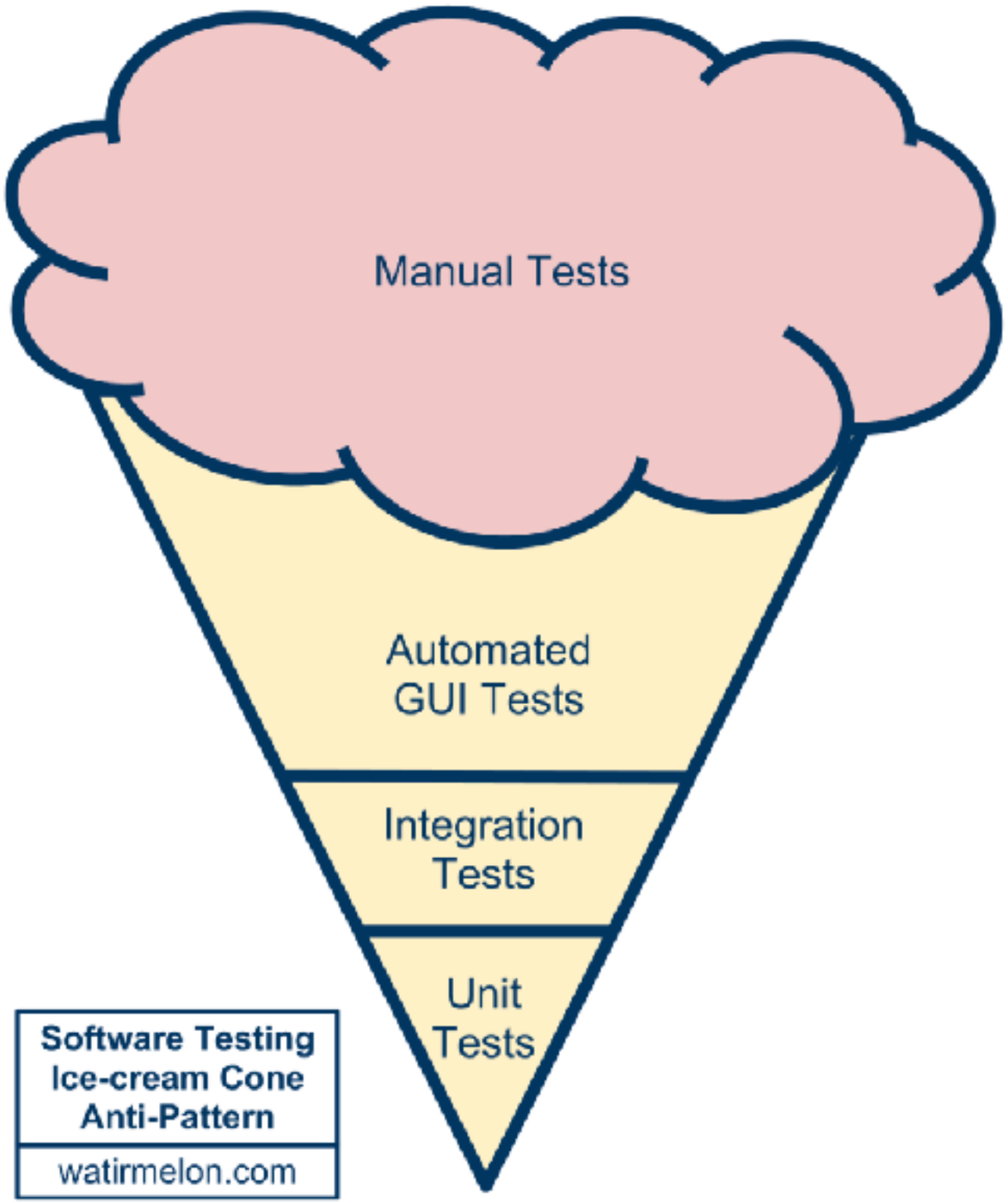
ทำได้ไหม ?

# ถ้าทำไม่ได้

ต้องทำการปรับปรุง code ปัจจุบันก่อน

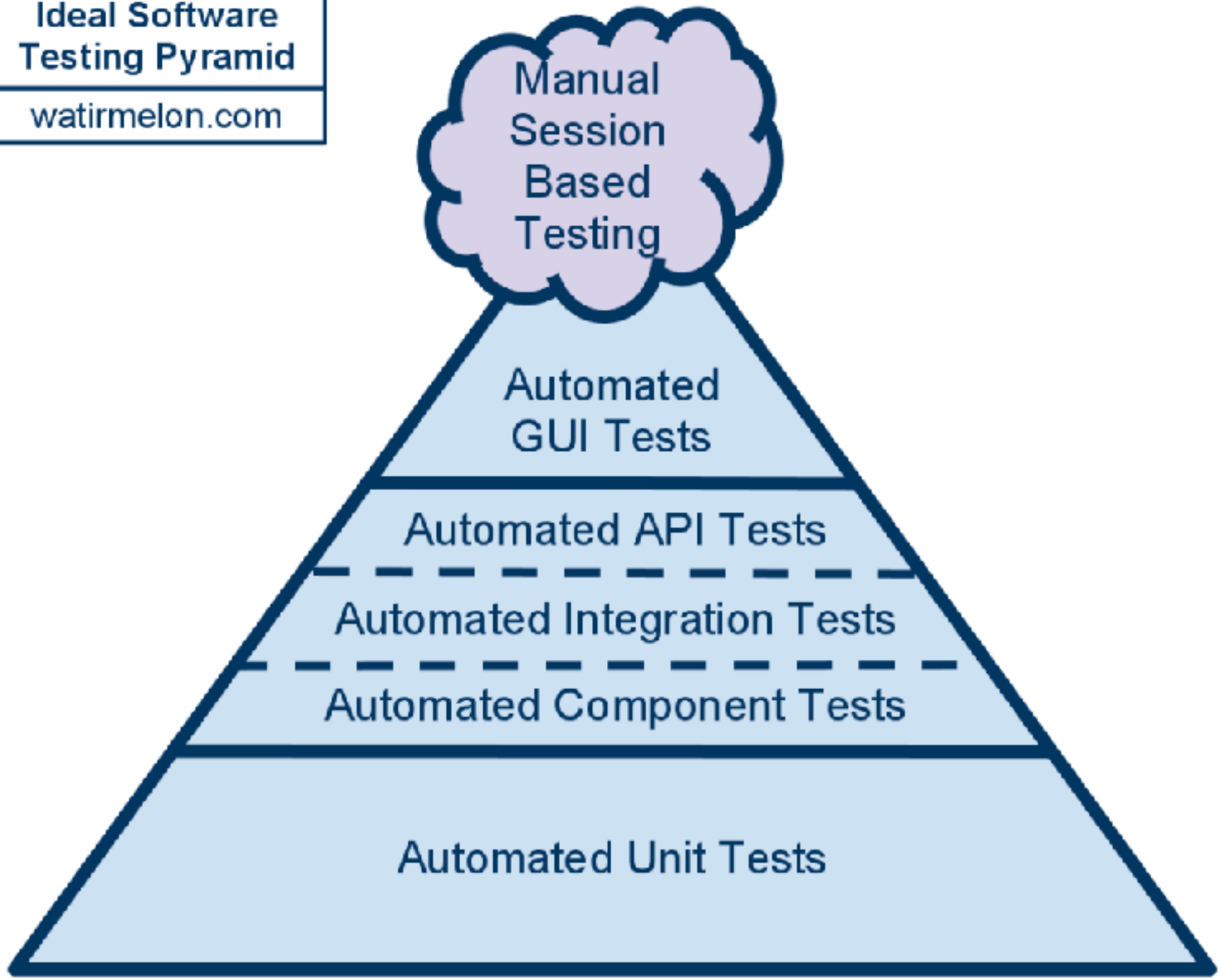
# Let’s go

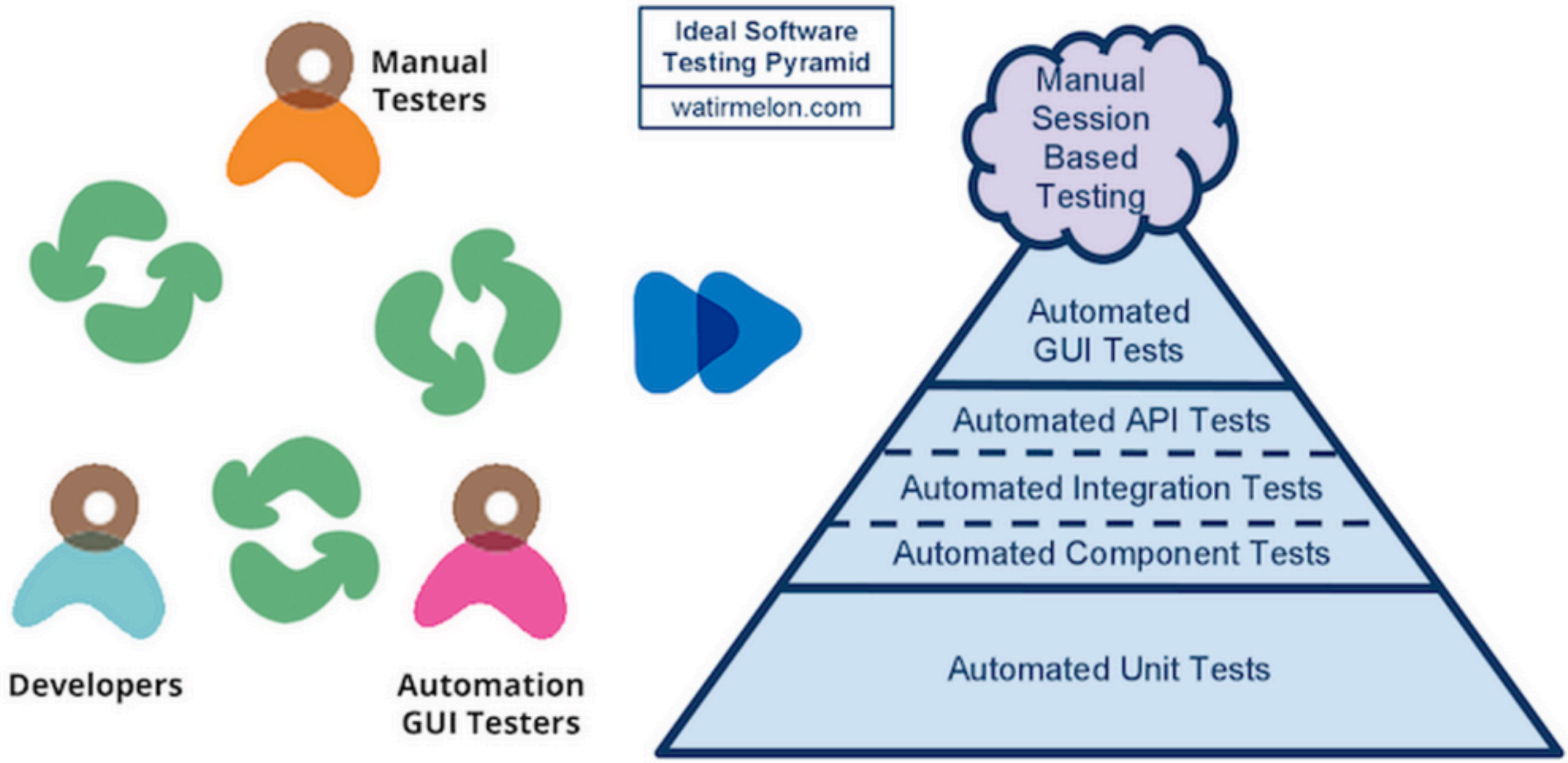
# Software Testing



**Ideal Software Testing Pyramid**

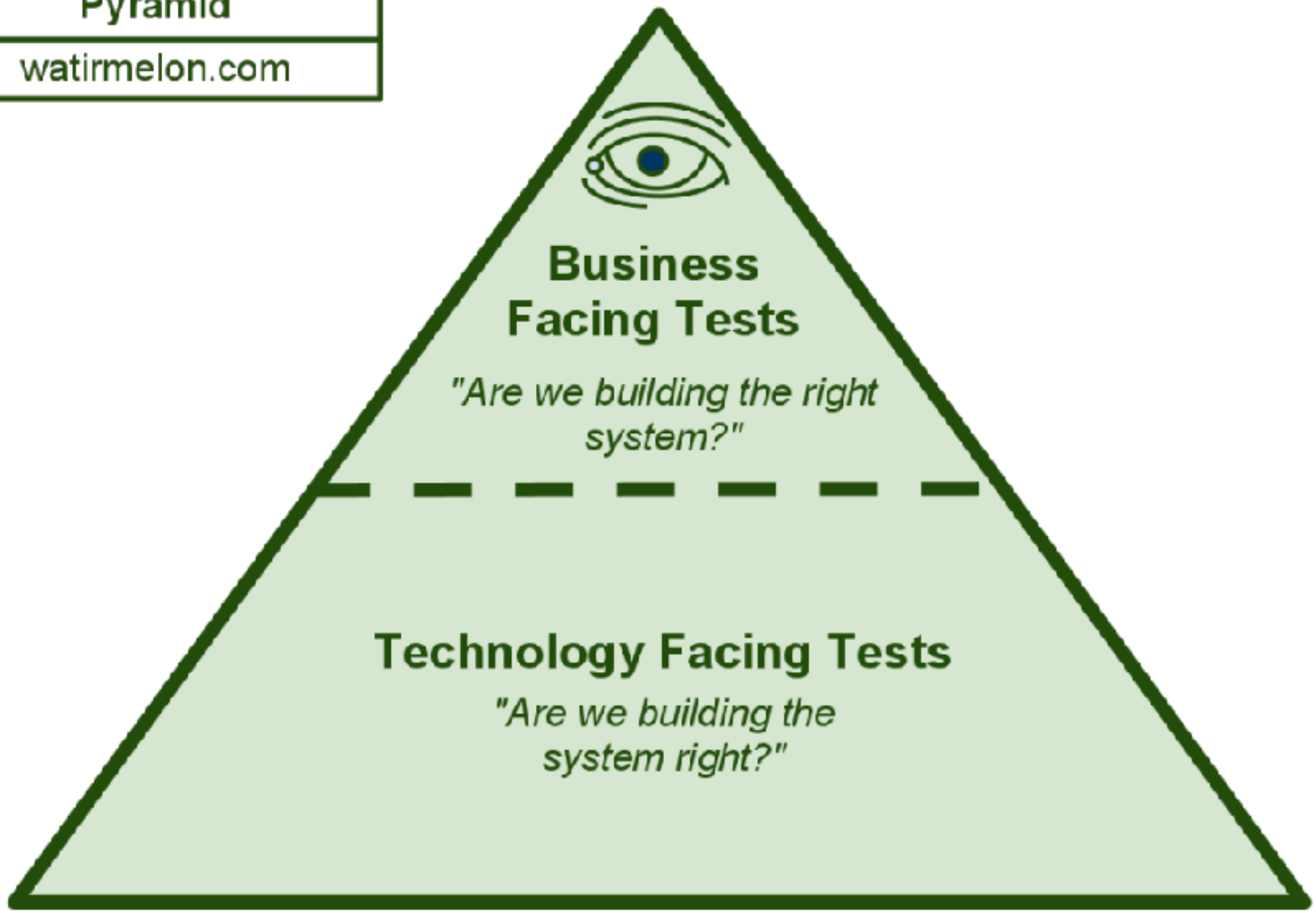
watirmelon.com

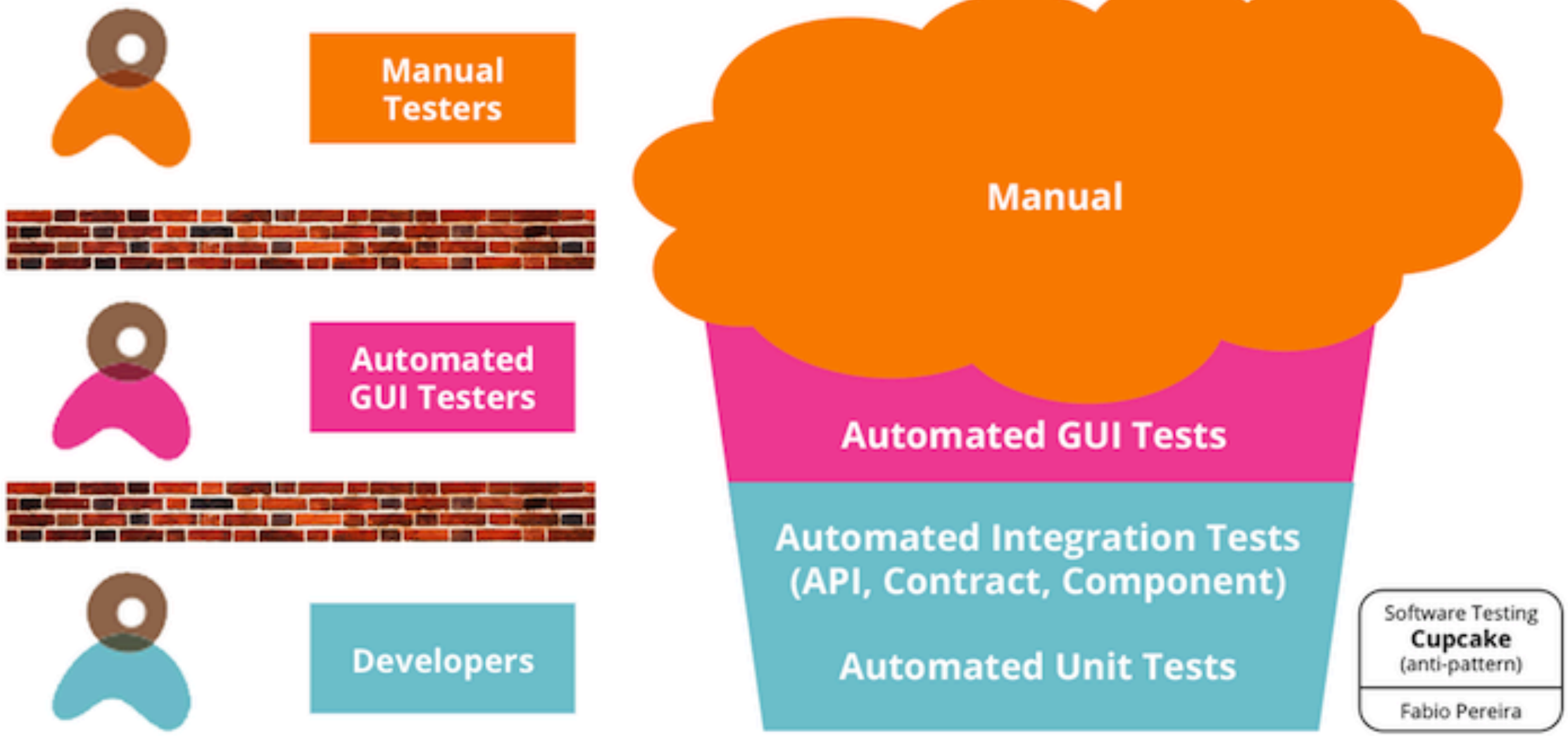


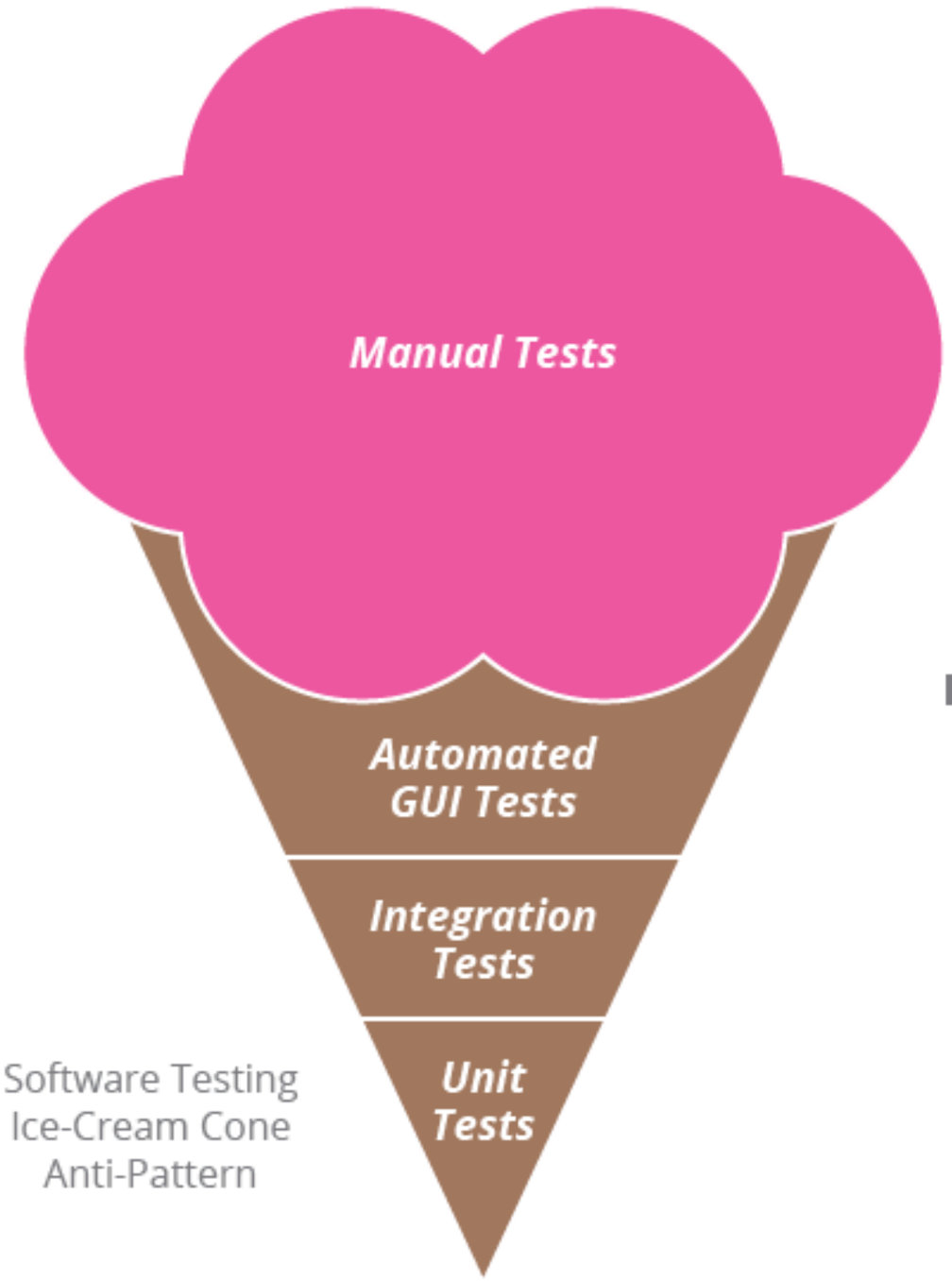


**Software Testing Pyramid**

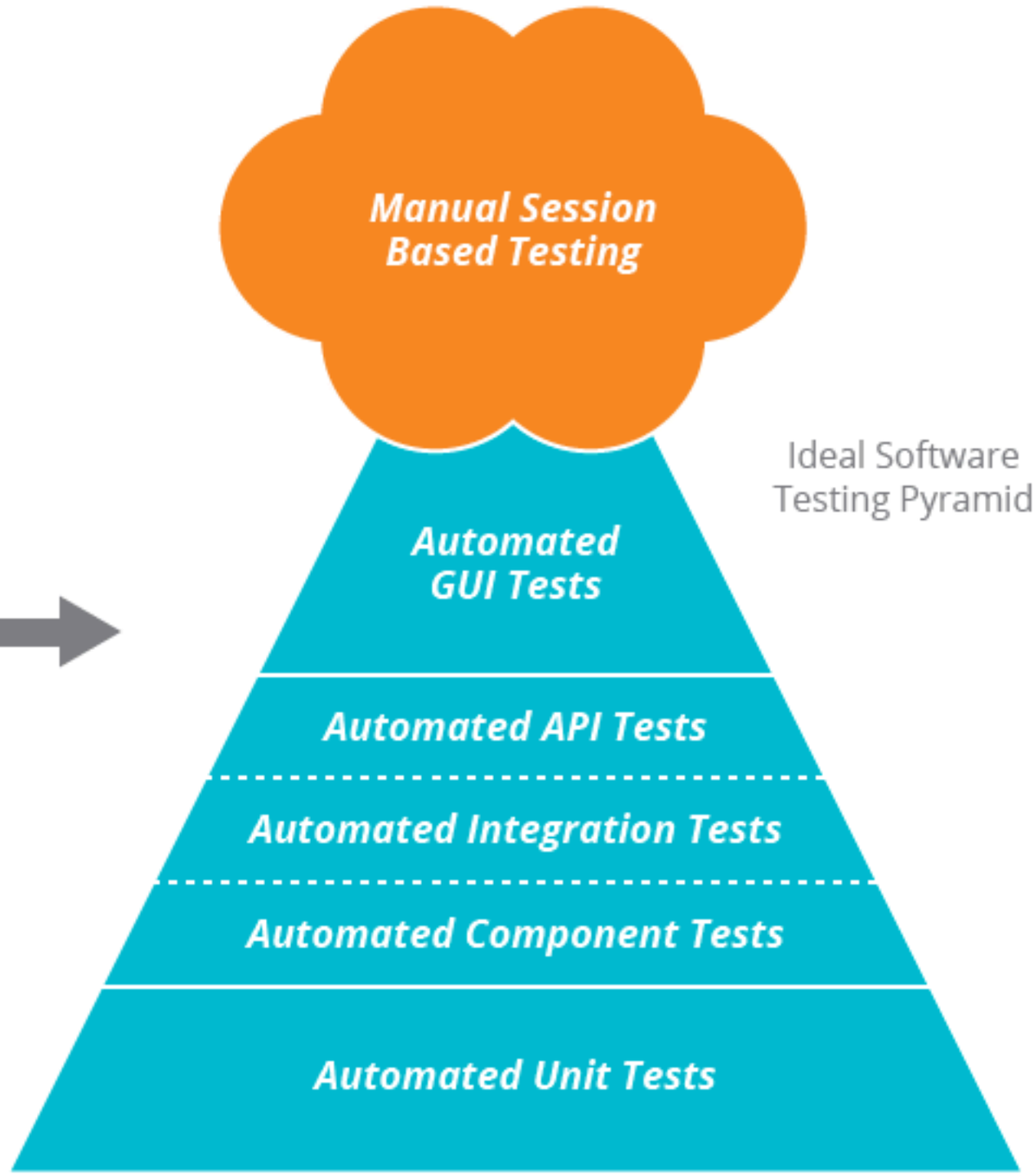
watirmelon.com



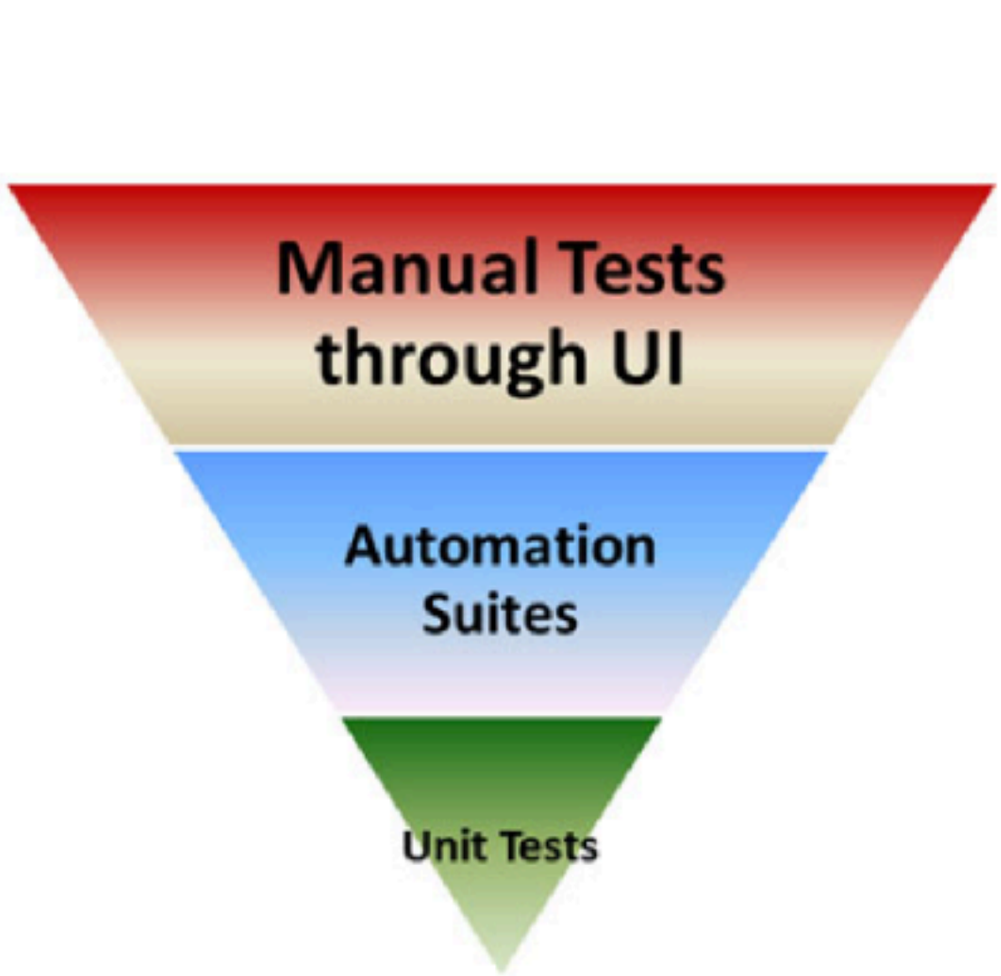




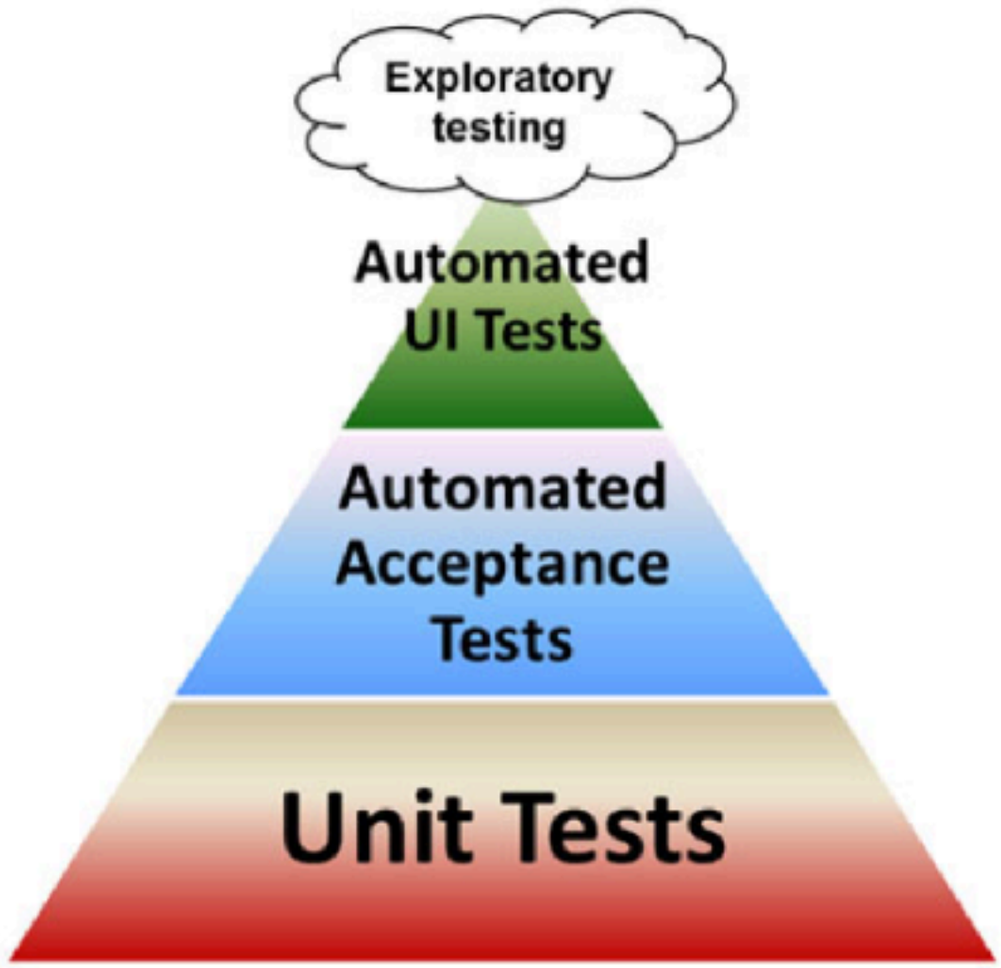

Software Testing Ice-Cream Cone Anti-Pattern




Ideal Software Testing Pyramid

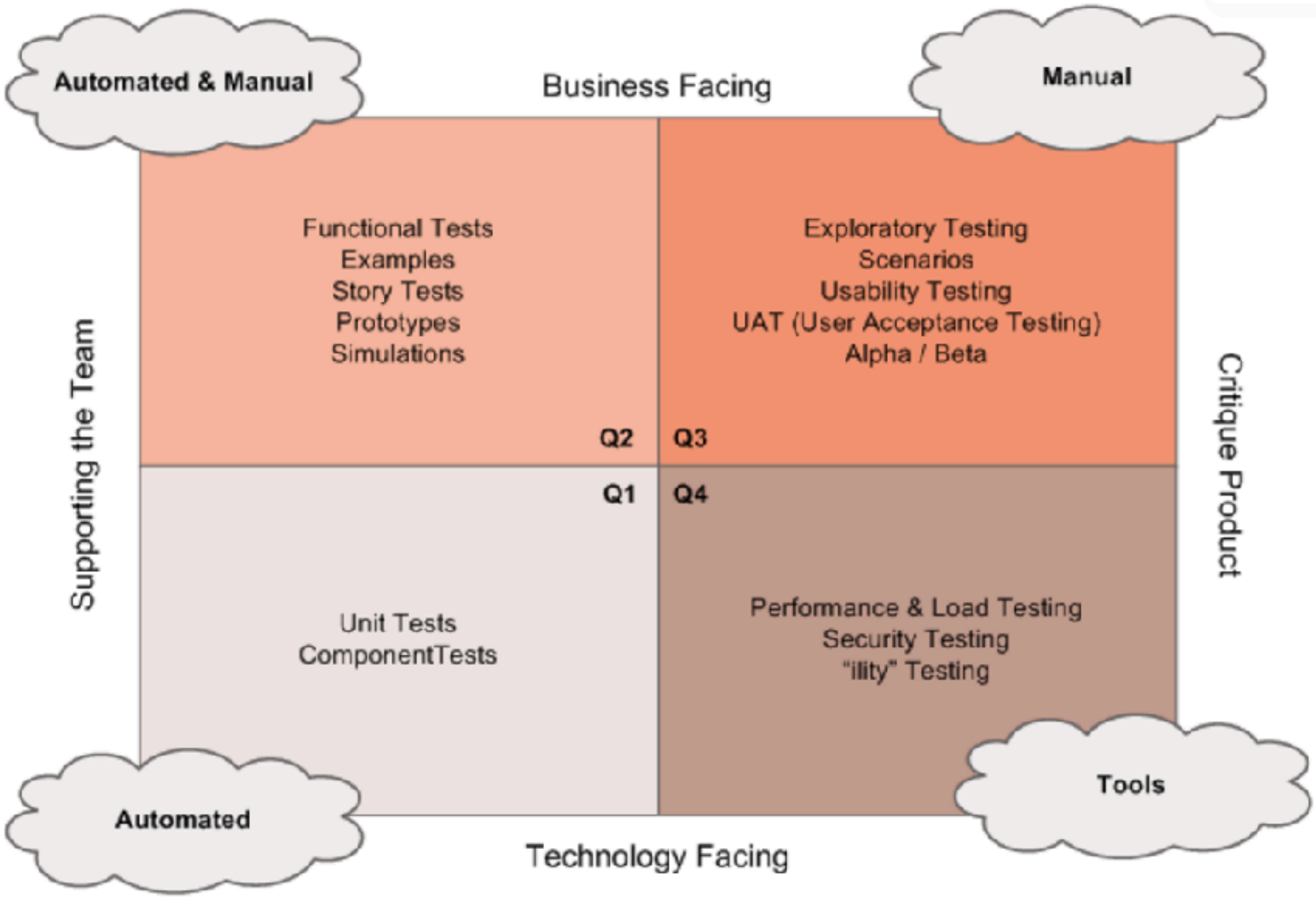



Traditional
(find bugs)

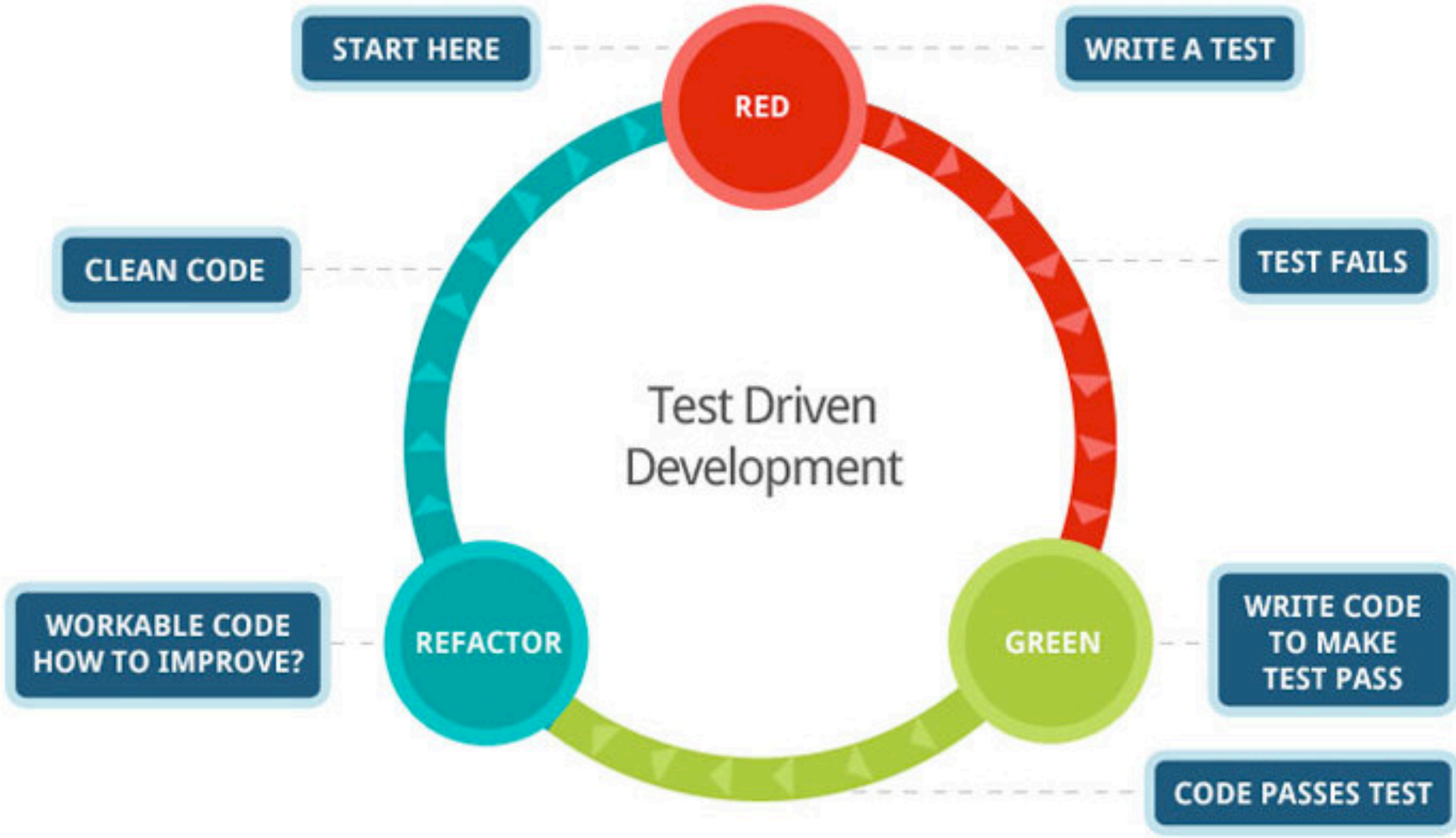



Agile
(prevent bugs)

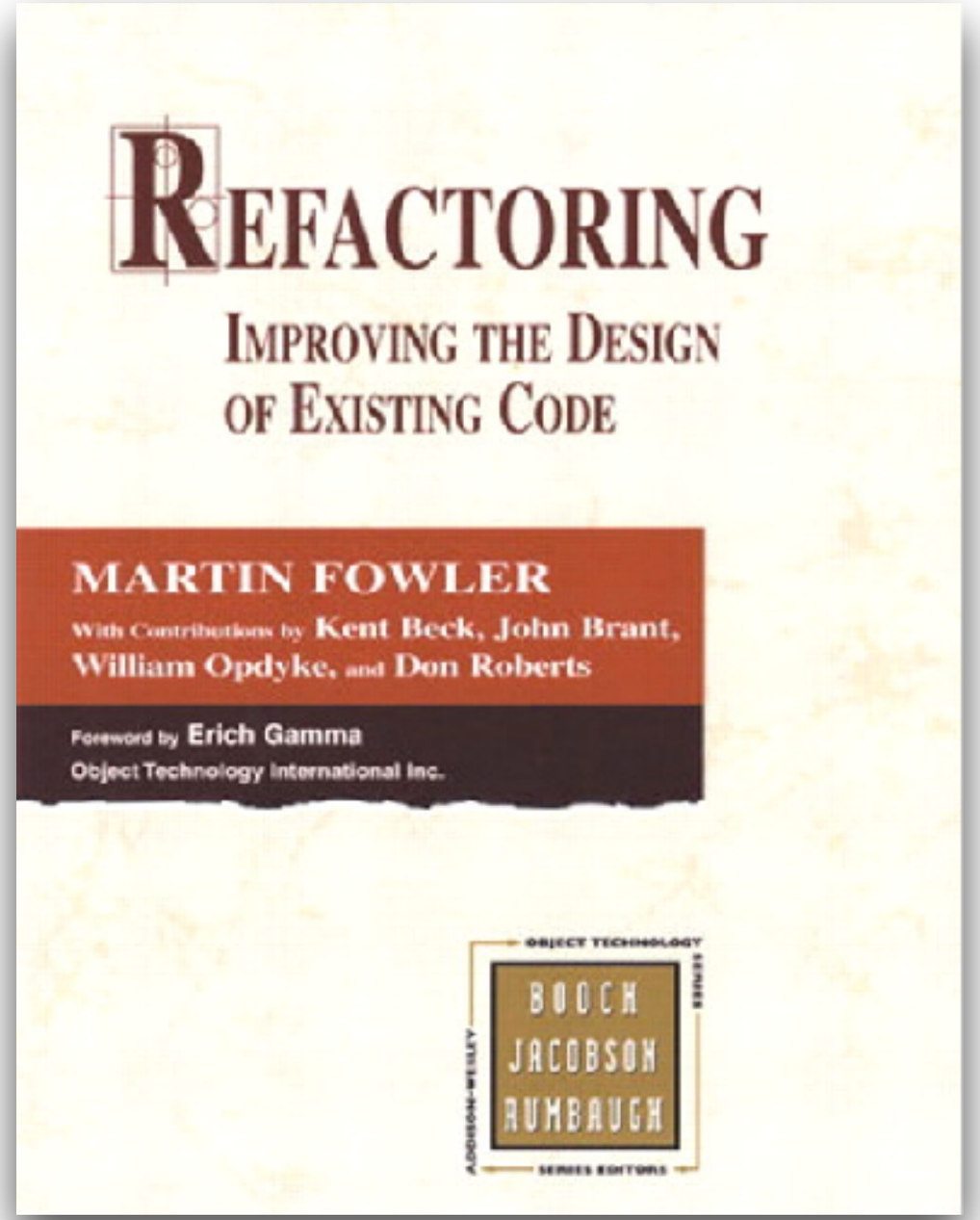
# Agile testing quadrant

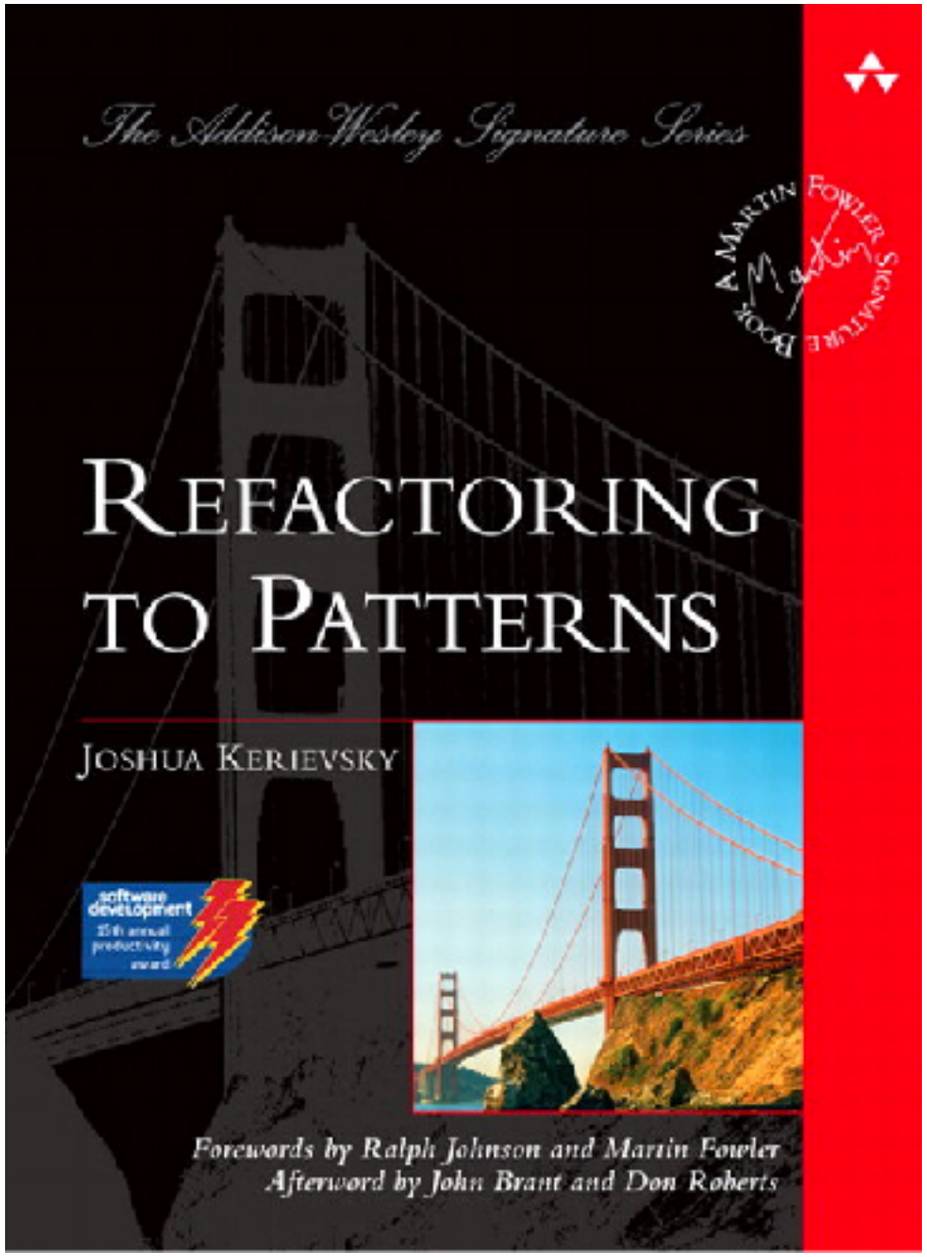


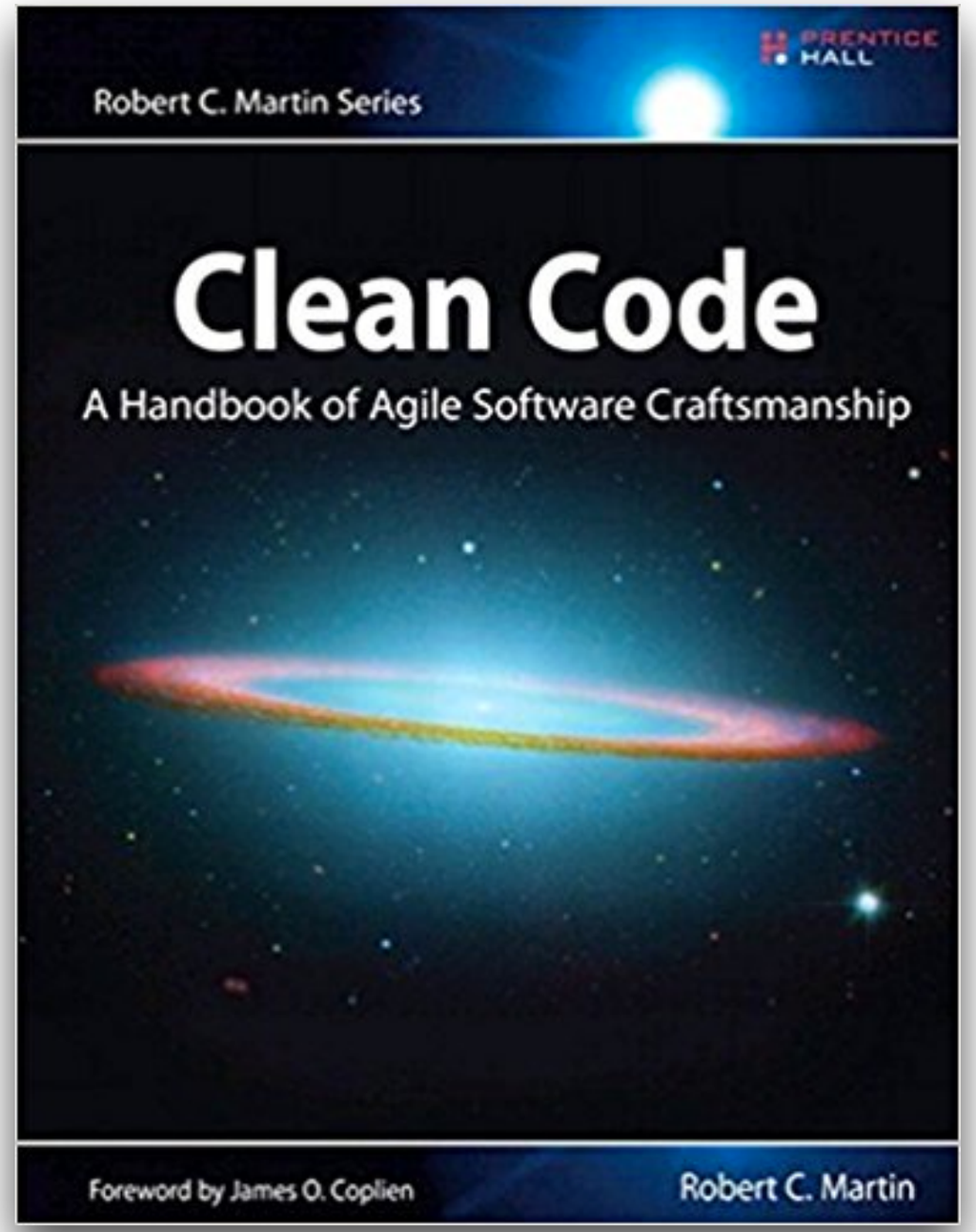
# Refactoring

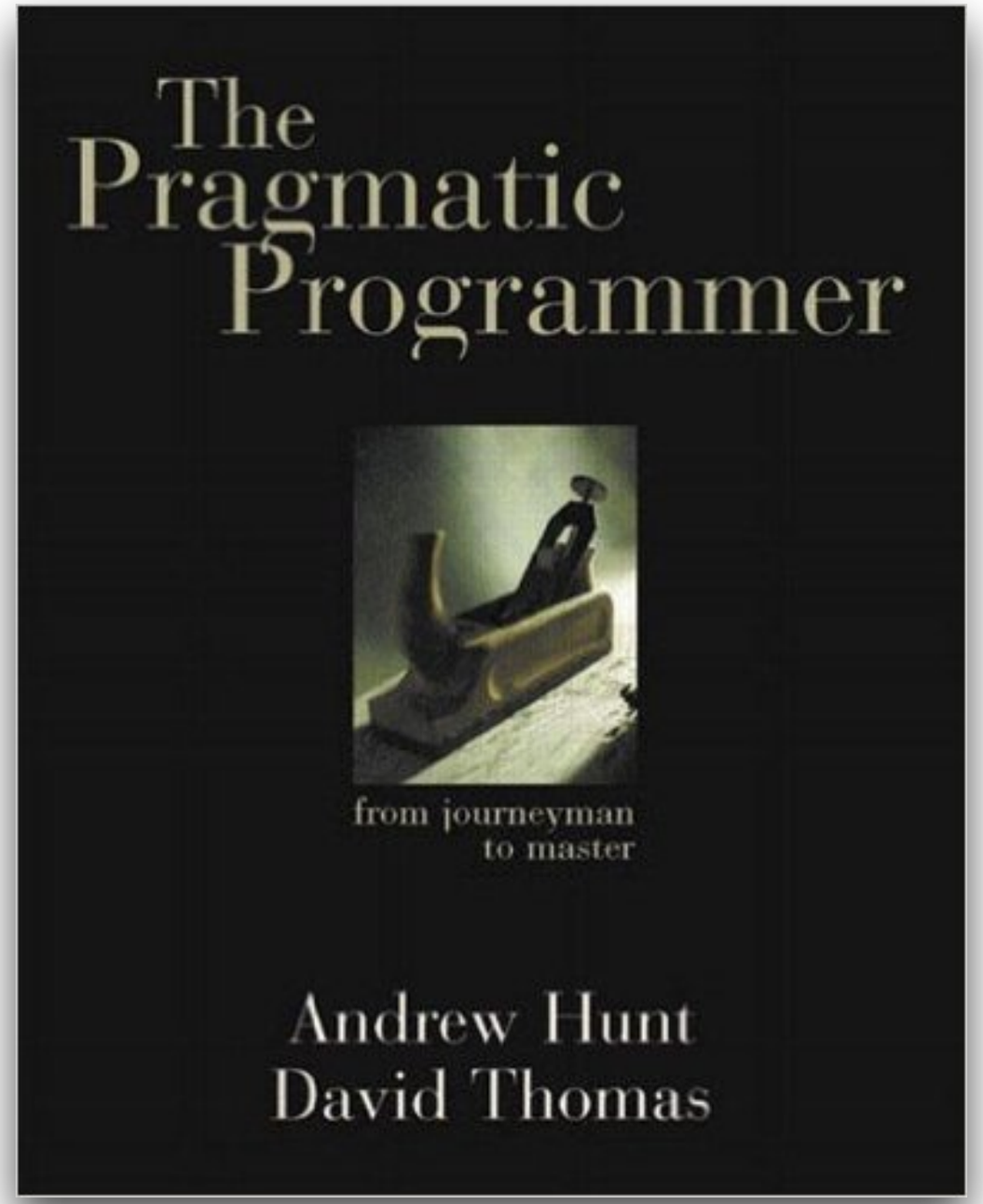



http://haselt.com/coding-dojo-with-tdd/

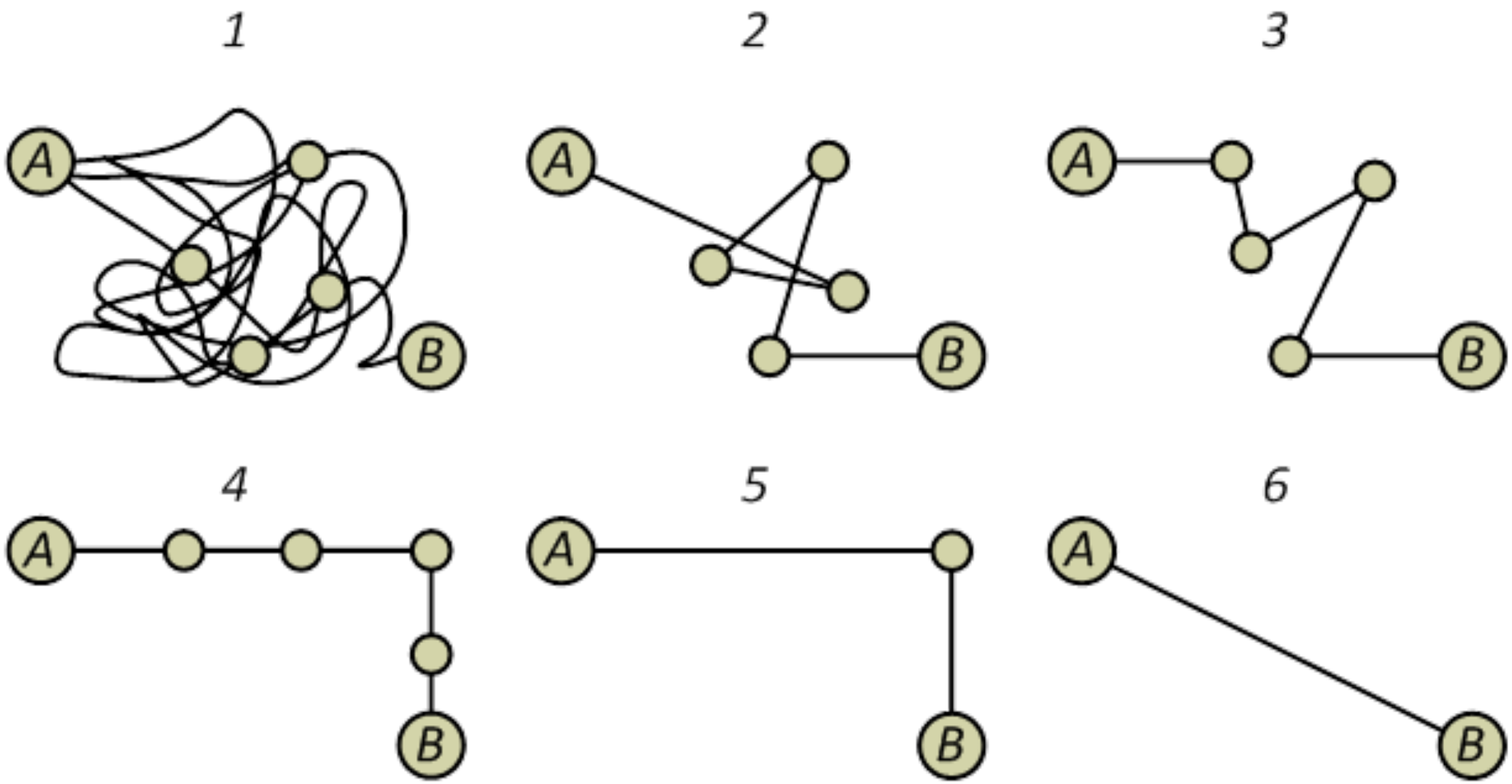




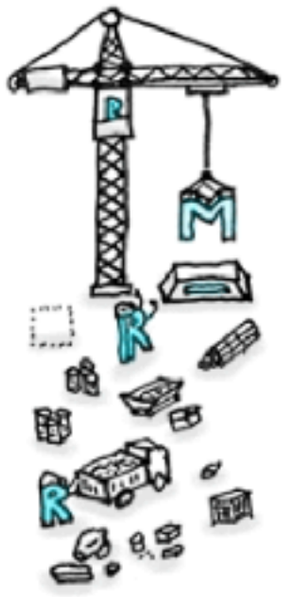



# Refactoring




https://codeandwork.github.io/courses/common/writingMaintainableCode.html

# Refactoring techniques



## Composing methods

Much of refactoring is devoted to correctly composing methods. In most cases, excessively long methods are the root of all evil. The vagaries of code inside these methods conceal the execution logic and make the method extremely hard to understand – and even harder to change.

- Extract Method
- Inline Method
- Extract Variable
- Inline Temp
- Replace Temp with Query
- Split Temporary Variable
- Remove Assignments to Parameters
- Replace Method with Method Object
- Substitute Algorithm

https://sourcemaking.com/refactoring/refactorings

# Code Smells

– What? How can code "smell"??

– Well it doesn't have a nose... but it definitely can stink!

## Bloaters

Bloaters are code, methods and classes that have increased to such gargantuan proportions that they are hard to work with. Usually these smells do not crop up right away, rather they accumulate over time as the program evolves (and especially when nobody makes an effort to eradicate them).

- Long Method
- Large Class
- Primitive Obsession
- Long Parameter List
- Data Clumps

https://sourcemaking.com/refactoring/smells

# Refactoring workshop

https://github.com/emilybache/Tennis-Refactoring-Kata

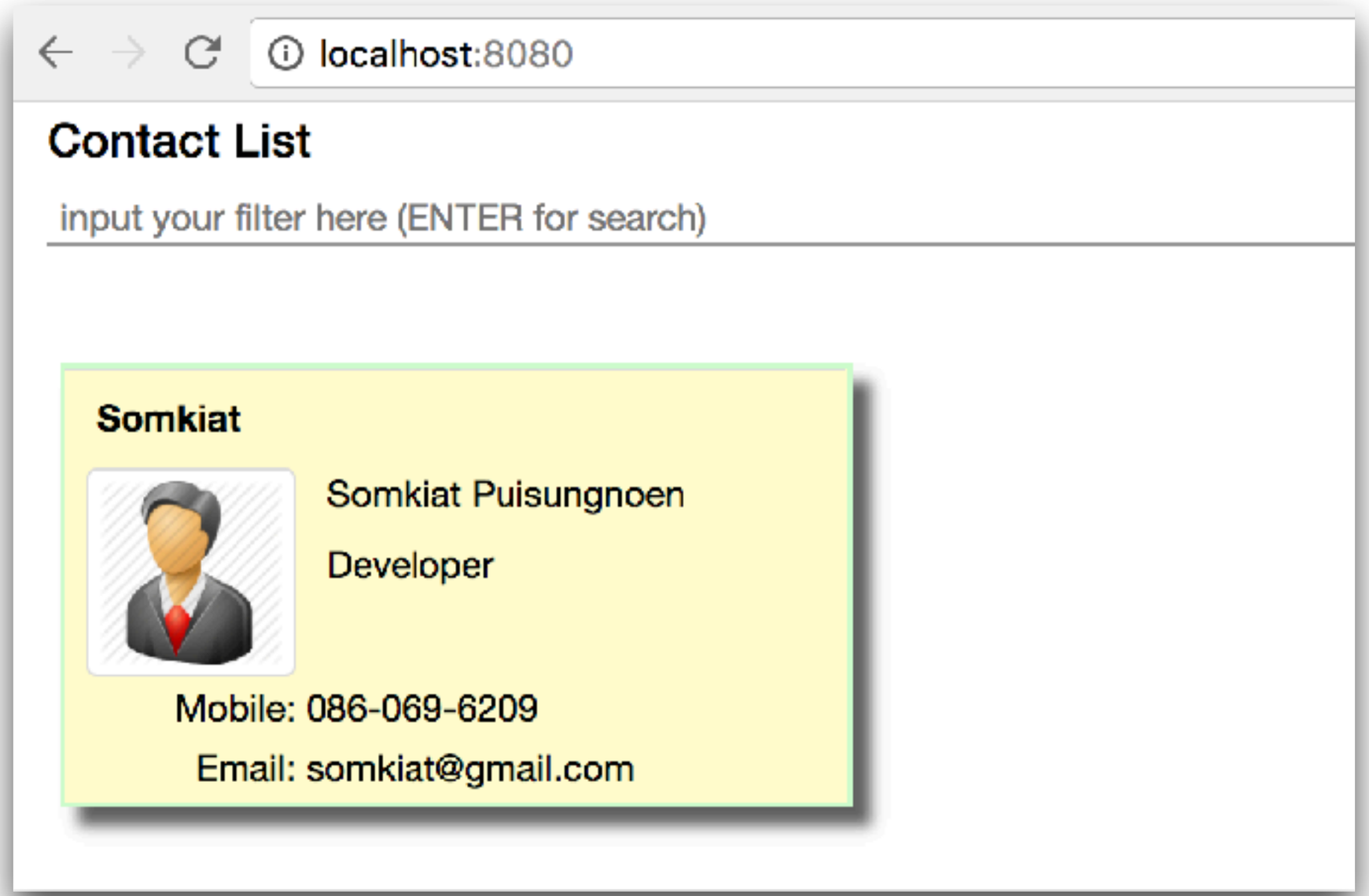
# Let’s workshop

# TDD workshop with SpringBoot

https://github.com/up1/workshop_java_legacy

# Let’s start

$mvn spring-boot:run



# Code coverage

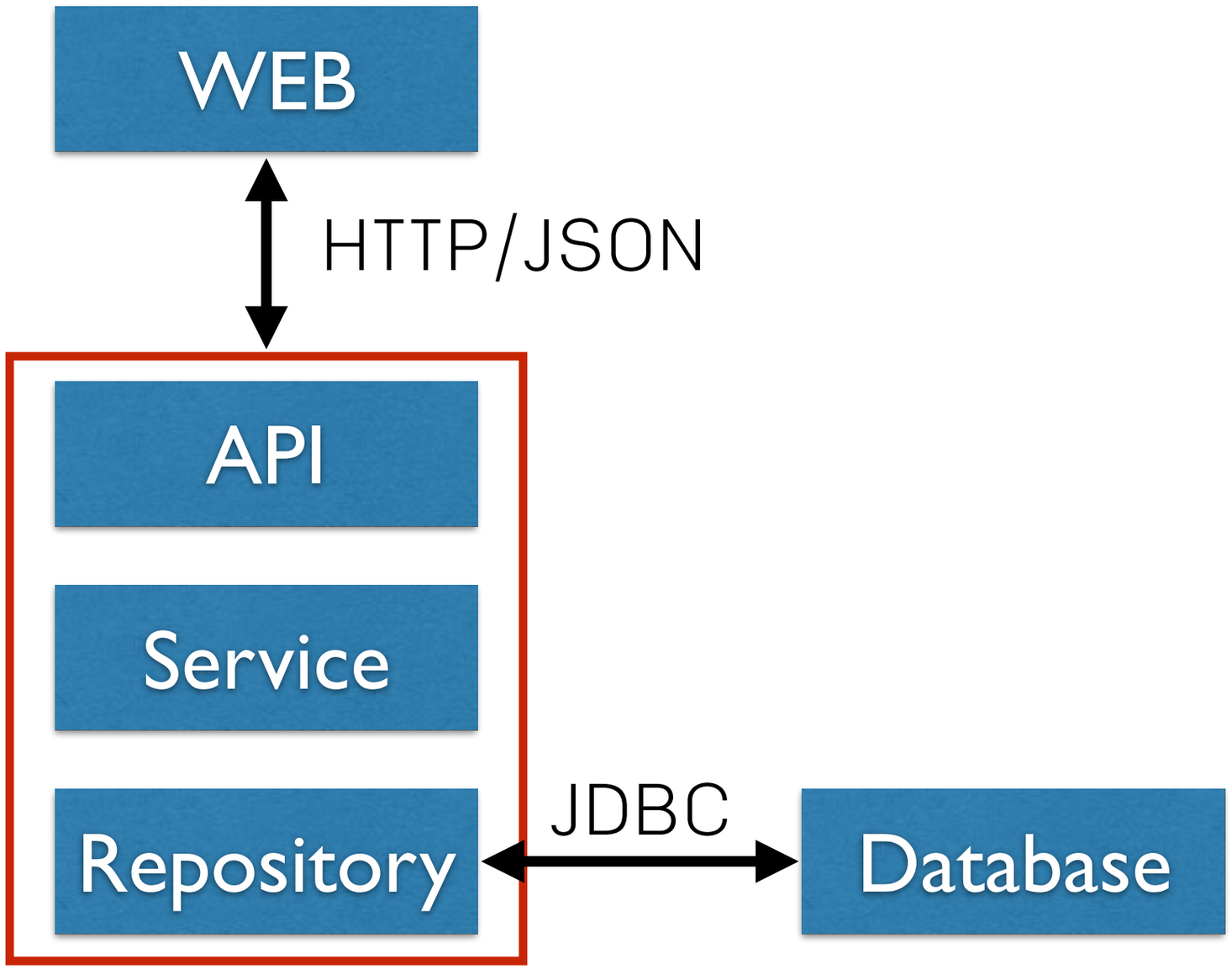
$mvn jacoco:report

demo-legacy

## demo-legacy

| Element | Missed Instructions | Cov. | Missed Branches | Cov. | Missed | Cxty | Missed | Lines | Missed | Methods | Missed | Classes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| demo.api | | 25% | | 0% | 6 | 10 | 15 | 22 | 5 | 9 | 0 | 2 |
| demo | | 17% | | n/a | 2 | 3 | 3 | 4 | 2 | 3 | 0 | 1 |
| demo.service | | 78% | | 50% | 3 | 8 | 2 | 12 | 1 | 6 | 0 | 1 |
| demo.config | | 60% | | n/a | 1 | 3 | 3 | 8 | 1 | 3 | 0 | 1 |
| demo.domain | | 94% | | n/a | 1 | 13 | 1 | 22 | 1 | 13 | 0 | 1 |
| demo.sample | | 100% | | n/a | 0 | 2 | 0 | 5 | 0 | 2 | 0 | 1 |
| Total | 100 of 268 | 62% | 4 of 6 | 33% | 13 | 39 | 24 | 73 | 10 | 36 | 0 | 7 |

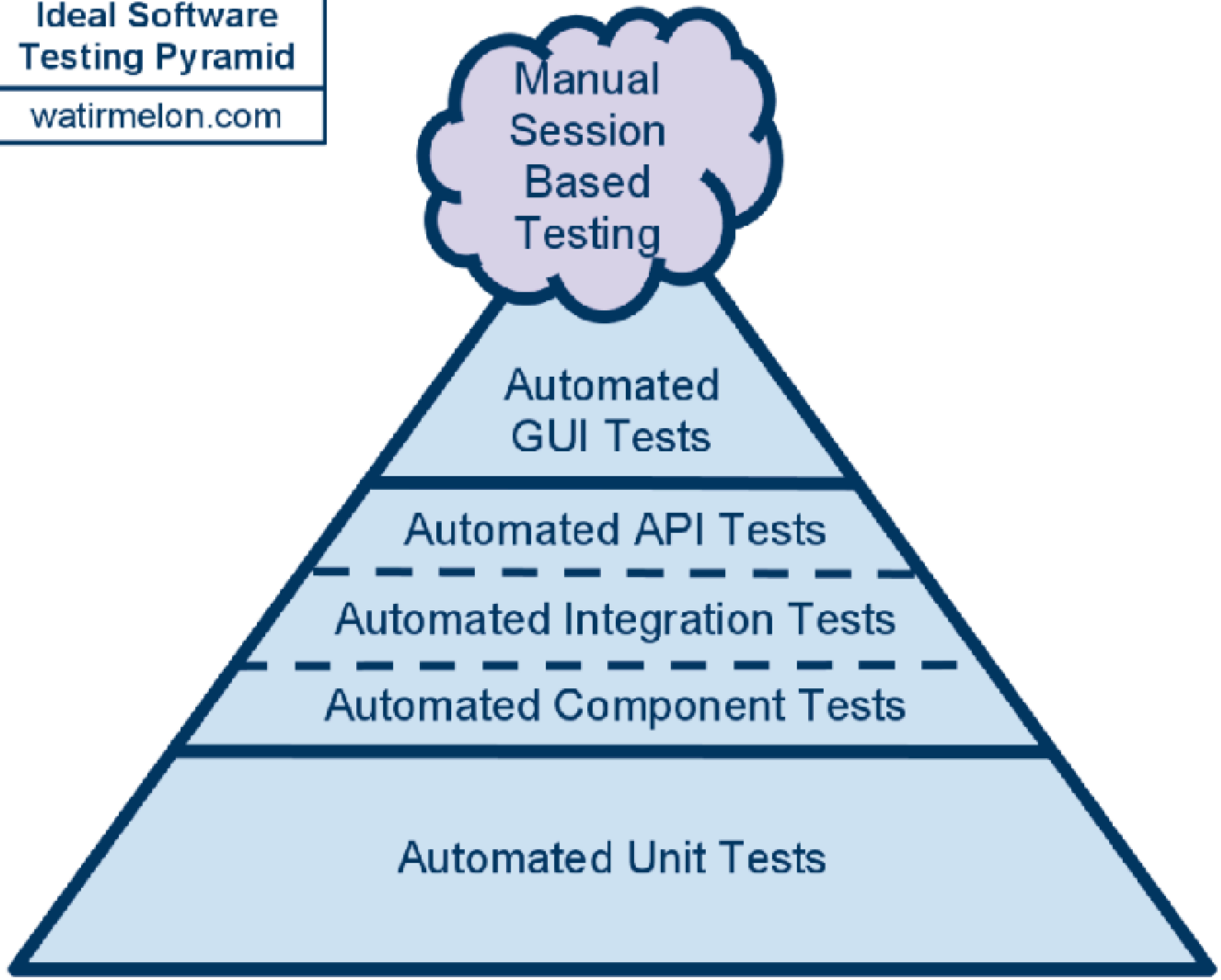
# Architecture




# How to testing ?

**Ideal Software Testing Pyramid**

watirmelon.com



# Goal

Code coverage 100%

# Compile

$mvn clean package -DskipTests=true

# Unit tests

# Unit tests + Test double

Domains
Services

# domain/ContactTest.java

```
public class ContactTest {

    @Test
    public void can_be_update() {
        Contact contact = new Contact( name: "name",  fullName: "fullname"
        Contact newContact = new Contact( name: "new name",  fullName: "ne
        contact.updateWith(newContact);

        assertEquals( expected: "new name", contact.getName());
        assertEquals( expected: "new fullname", contact.getFullName());
    }
```

# domain/ContactTest.java

```
@Test
public void can_not_be_update() {
    Contact contact = new Contact( name: "name",  fullName: "full
    Contact newContact = new Contact( name: null,  fullName: null
    contact.updateWith(newContact);

    assertEquals( expected: "name", contact.getName());
    assertEquals( expected: "fullname", contact.getFullName());
}
```

# service/ContactServiceTest.java

```java
@RunWith(MockitoJUnitRunner.class)
public class ContactServiceTest {

    @Mock
    ContactRepository contactRepository;

    @InjectMocks
    ContactService contactService;

    @Test
    public void success_with_save_new_contact() {
        Contact newContact
                = new Contact( name: "Somkiat",  fullName: "Somkiat Puisungnoen",
                               jobTitle: "Developer",  email: "somkiat@gmail.com",
                               mobile: "086-869-6209");

        contactService.saveContact(newContact);

        verify(contactRepository).save(newContact);
    }
```

# Run unit tests

$mvn surefire:test

# Integration tests

# Integration tests

Services
Use the SpringTest

# service/integration/ContactServiceTest.java

```java
@RunWith(SpringRunner.class)
@SpringBootTest(classes = Application.class)
public class ContactServiceTest {

    @Autowired
    private ContactService contactService;

    @Before
    public void tearDown() { contactService.deleteAllContacts(); }

    @Test
    public void saves_new_contact() {//given
        Contact newContact
                = new Contact( name: "Somkiat", fullName: "Somkiat Puisungnoen",
                jobTitle: "Developer", email: "somkiat@gmail.com",
                mobile: "086-869-6209");

        Contact savedContact = contactService.saveContact(newContact);
        Contact actualContact = contactService.load(savedContact.getId());

        assertEquals(newContact, actualContact);
    }
```

# Integration tests

Services
Use the SpringTest

# Run integration tests

$mvn failsafe:integration-test

# API tests

# API tests

API/Controllers
Use the SpringTest

# api/integration/ContactControllerTest.java

```java
@RunWith(SpringRunner.class)
@SpringBootTest(classes = Application.class)
public class ContactControllerTest {

    @Autowired
    ContactRepository repository;

    @Autowired
    ContactController controller;

    @Before
    public void setup() {
        repository.deleteAll();
        mockMvc = buildMockMvc(controller);
    }

    protected MockMvc mockMvc;

    protected MockMvc buildMockMvc(Object... controllers) {
        return MockMvcBuilders
                .standaloneSetup(controllers)
                .build();
    }
```

# api/integration/ContactControllerTest.java

```java
@Test
public void should_save_contact() throws Exception {
    MvcResult result = mockMvc.perform(
            post( urlTemplate: "/api/contacts").contentType(APPLICATION_JSON)
            .content("{\"name\":\"Somkiat\",\"fullName\":\"Somkiat Puisungnoen\"}"))
            .andExpect(status().isCreated())
            .andReturn();

    Contact contact = repository.findOne(result.getResponse().getContentAsString());
    assertThat(contact.getName()).isEqualTo("Somkiat");
    assertThat(contact.getFullName()).isEqualTo("Somkiat Puisungnoen");
}
```

# UI tests

# UI tests

Web application

# Let’s workshop

# Continuous Integration with Jenkins

# Start Jenkins server

$java -jar jenkins.war

# Let’s workshop

สยามชำนาญกิจ
SIAM CHAMNAN KIT