Introduction to Software Testing

11/10/2016

Clément Duffau

Verification & Validation

V&V

Verification → Answer to technical specifications

Are we building the good product ?

Validation → Answer to functional specifications

Are we building the product correctly ?

“Testing is the process of executing a program with the intent of finding errors”

Glen Myers

“If test results are also green, what’s the purpose of tests ?”

Marc Rougé (AXONIC CEO)

Are you ok to walk up in a plane if the airman says to you “That’s the first time this plane will take off” ?

1h of coding

~

1h of testing

Test

Trial

Debugging

Test

≠

Trial

≠

Debugging

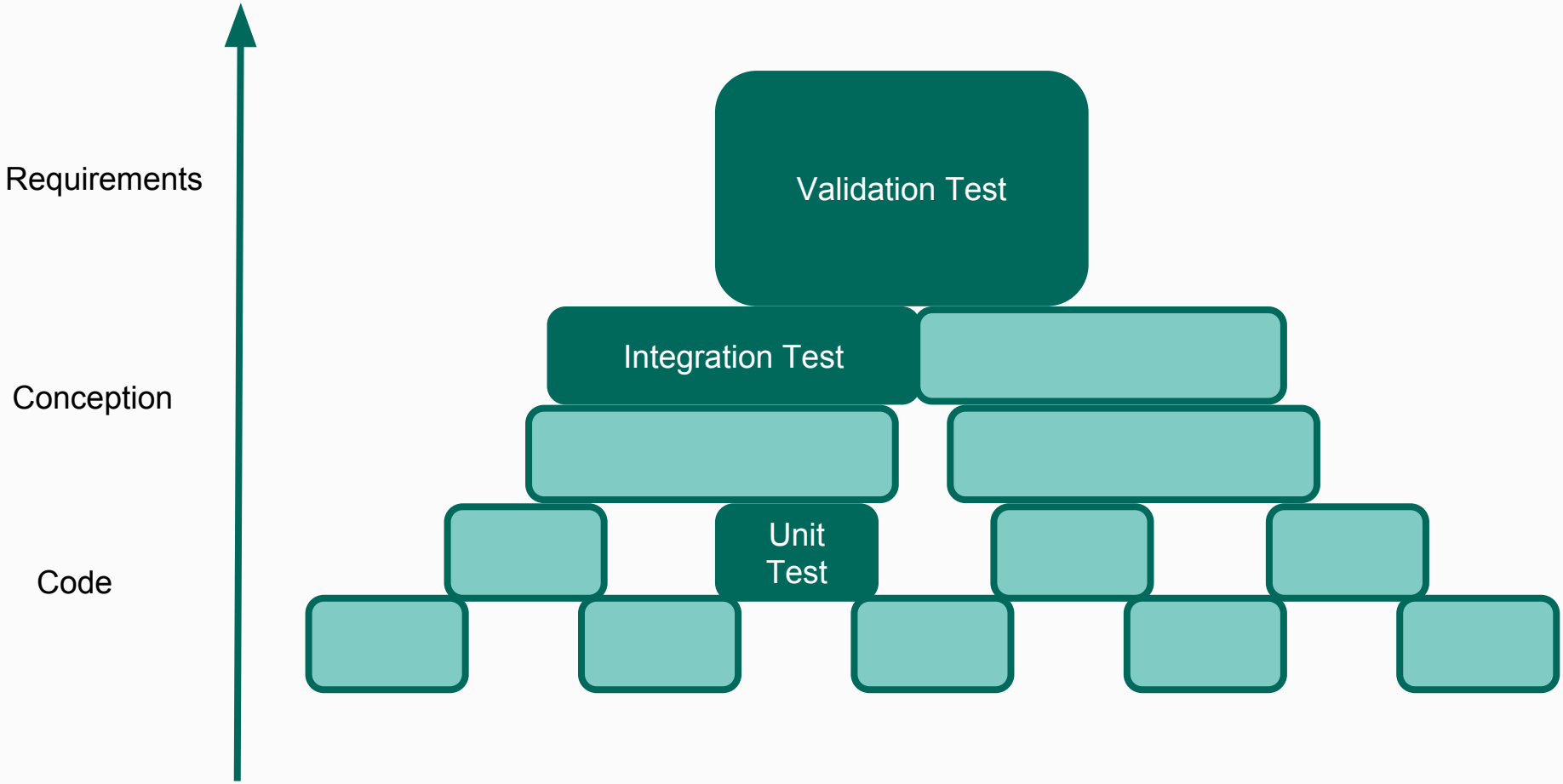
Test vs Trial vs Debugging

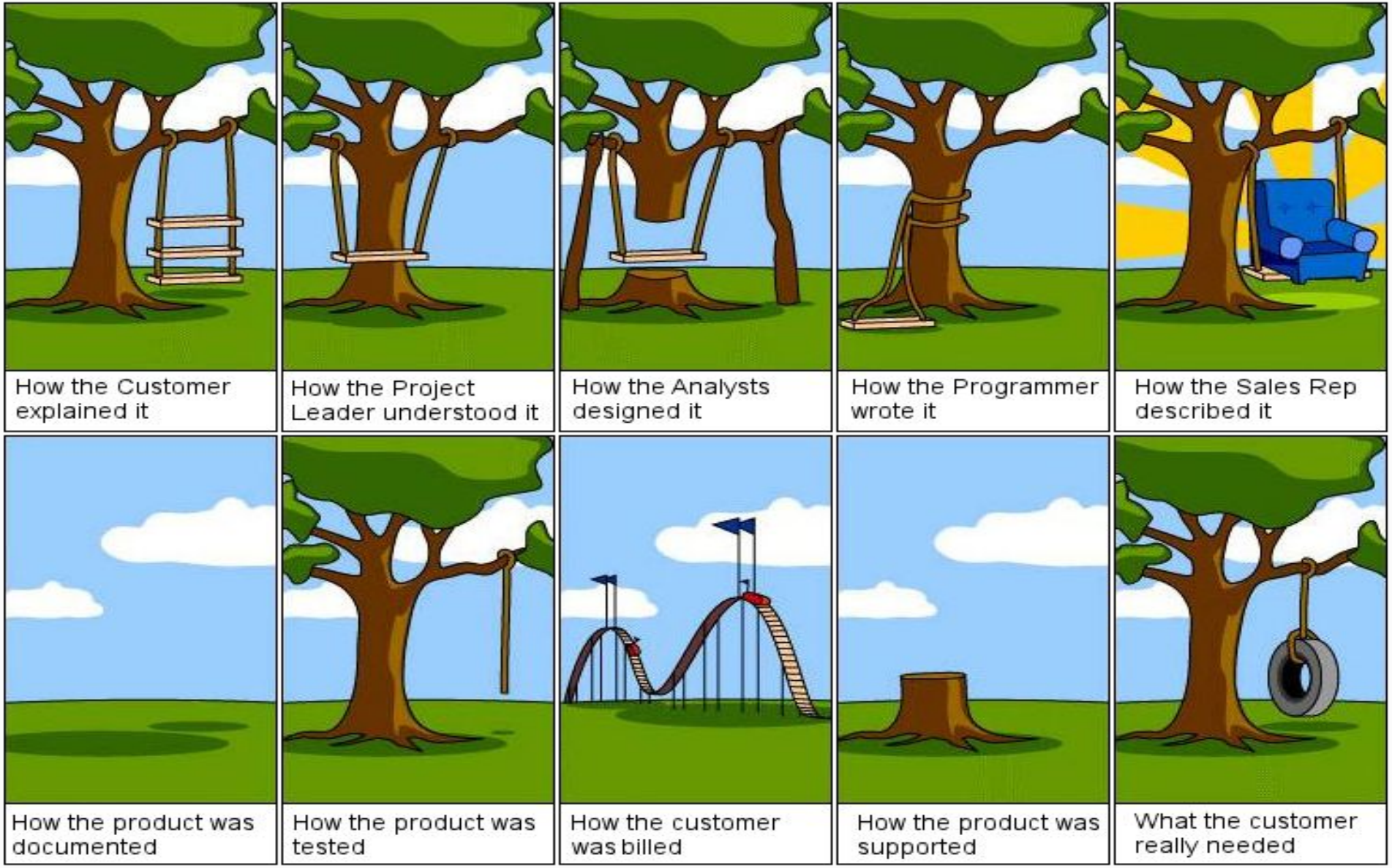
Test → Reproducible (mitigation costs)

Trial → Manual

Debugging → Investigation

Tests strategies





Software Testing and Agility

Acceptance Criteria

Set of conditions for user story validation

Example :

User Story : As an Administrator, I want to be able to create User Accounts so that I can grant users access to the system

Acceptance critera :

- If I am an Administrator, I can create User Accounts.
- I can create a User Account by entering the following information about the User: a) Name, b) Email address. The system notifies me that it sent an email to the new User's email address, containing a system-generated initial password and instructions for the person to log in and change their password.
- I am able to verify with the intended recipient of the email that it was received.

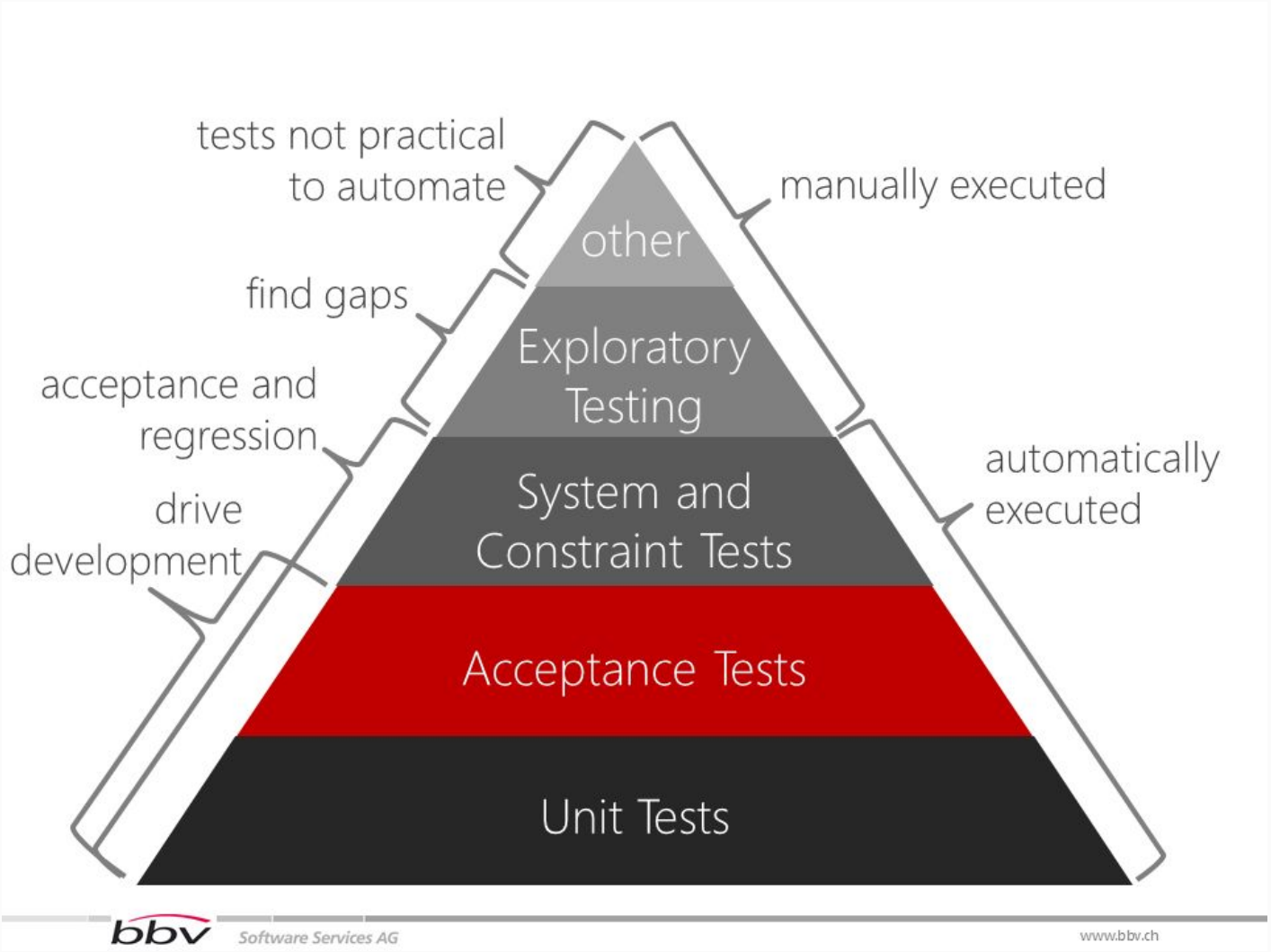
Acceptance Test

One (or more) scenario(s) for one condition of the acceptance criteria

Examples :

- Create a user with a name and email
- Try to create a user just with email → failure testing
- Check the list the people who received the confirmation email

Tests pyramid



Software Testing by example

Example with JUnit

```java
public class Point {

    private double x;

    private double y;

    // On a besoin d'un constructeur par defaut et des getter/setter pour la serialisation automatique en json/xml.
    public Point() {
    }

    public Point(double x, double y) {
        this.x = x;
        this.y = y;
    }
public class Shape {

    private String id;
    private List<Point> points;

    public Shape() {
    }

    public Shape(String id) { this.id = id; }

    public String getId() { return id; }

    public void setId(String id) { this.id = id; }

    public List<Point> getPoints() { return points; }

    public void setPoints(List<Point> points) { this.points = points; }

    public void addPoint(Point point) { throw new UnsupportedOperationException("Not Yet Implemented"); }
}
```

Test examples

```
private static final double delta=0.0;

@Test
public void testDefaultConstructor(){
    Point point=new Point();
    assertEquals(point.getX(),0, delta);
    assertEquals(point.getY(),0, delta);

}

@Test
public void testSetters(){
    Point point=new Point();

    point.setX(2);
    point.setY(3);

    assertEquals(point.getX(),2, delta);
    assertEquals(point.getY(),3, delta);

}

@Test
public void testConstructor(){
    Point point=new Point(2,3);
    assertEquals(point.getX(),2, delta);
    assertEquals(point.getY(),3, delta);

}
```

More complex test examples

```java
private static final double delta=0.01;
private Point point;
private Shape shape;

@Before
public void setup(){
    point=new Point(200,300);
    shape=new Shape();
}

@Test
public void testRotate360(){
    Point origin=new Point(0,0);
    Point startPoint=point;
    point.rotate(origin,360);
    assertEquals(point.getX(),startPoint.getX(), delta);
    assertEquals(point.getY(),startPoint.getY(), delta);
}

@Test(expected = UnsupportedOperationException.class)
public void testAddPointToShape(){
    try{
        shape.addPoint(point);
        fail();
    }
    catch (UnsupportedOperationException e){
        throw e;
    }

}
```

JUnit Tag words

@AfterClass / @BeforeClass

@After / @Before

@Test

assert*

fail()

expected

Mock testing

Purpose :

Simulate the behaviour of real objects

Allows to deal with :

Testing at interface level

Blacked-box component

JUnit + Mockito example

```java
public class ShapeServiceMockedTest extends JerseyTest{

    private static Shape shape;
    static {
        shape = new Shape("star");
        shape.addPoint(new Point(3.0, 6.0));
        shape.addPoint(new Point(4.0, 4.0));
        shape.addPoint(new Point(6.0, 4.0));
        shape.addPoint(new Point(4.5, 2.5));
        shape.addPoint(new Point(5.75, 0.0));
        shape.addPoint(new Point(3.0, 1.25));
        shape.addPoint(new Point(0.25, 0.0));
        shape.addPoint(new Point(1.5, 2.5));
        shape.addPoint(new Point(0.0, 4.0));
        shape.addPoint(new Point(2.0, 4.0));
    }
    @Mock
    private ShapesService service;

    @Override
    public ResourceConfig configure() {
        MockitoAnnotations.initMocks(this);
        List<Shape> shapes=new ArrayList<Shape>();
        shapes.add(shape);
        service=mock(ShapesService.class);
        when(service.getAllShapes()).thenReturn(Response.ok(shapes).build());

        return new ResourceConfig().register(service);
    }
    @Test
    public void testShapeList() {
        List<Shape> shapeList = (List<Shape>) service.getAllShapes().getEntity();
        assertTrue(shapeList.size()==1);
    }


}
```

Let’s start this into an IDE !