LP DAM - GL

Introduction to Software Testing

05/01/2017

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 always 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

≠

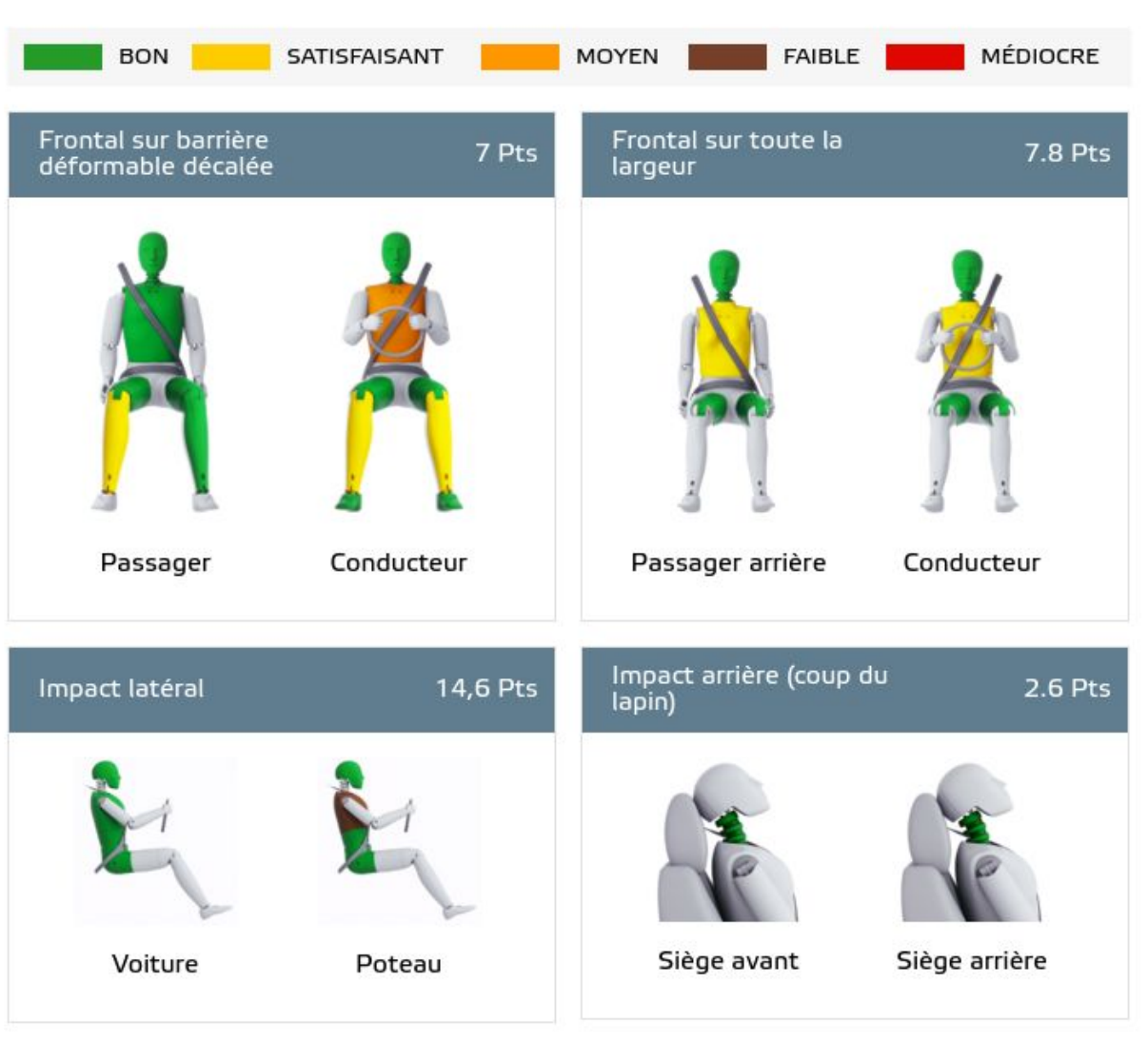
Trial

≠

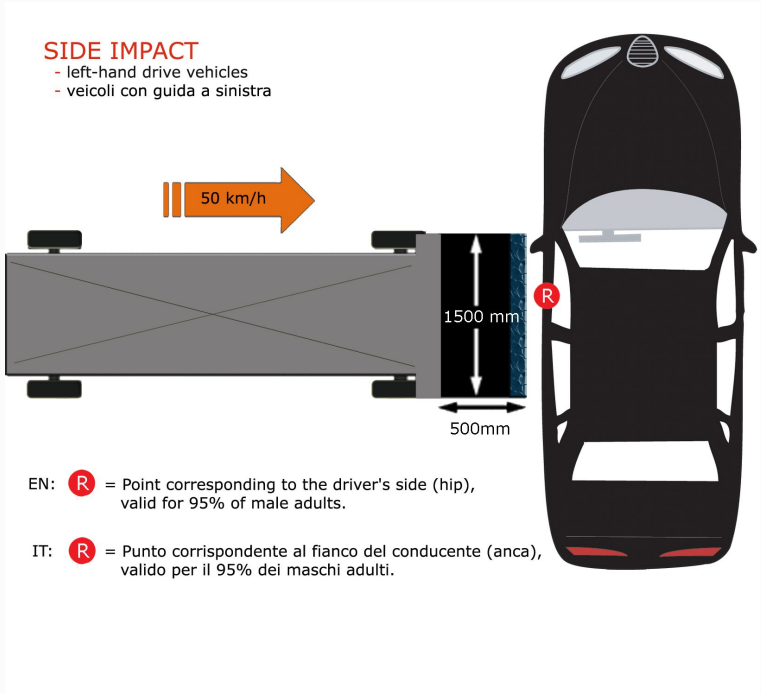
Debugging

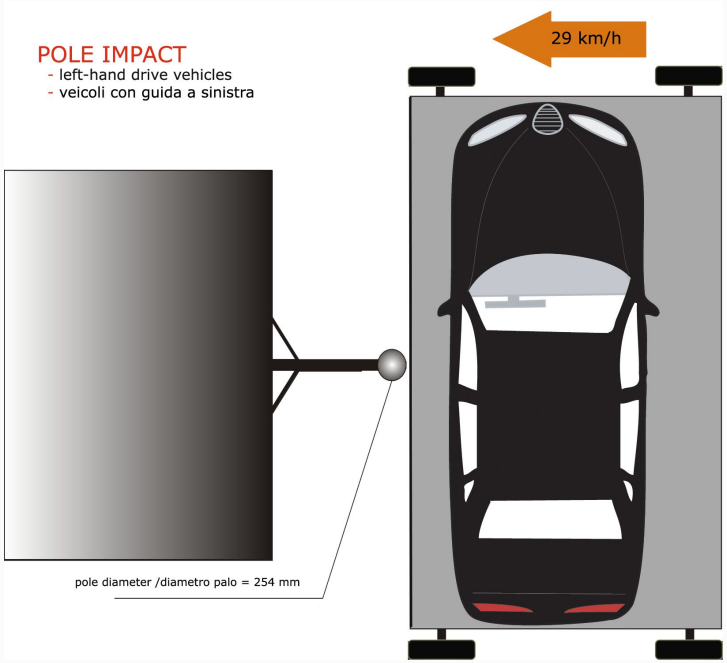
A trial, definitely not a test

Test oracles



Test cases


SIDE IMPACT
- left-hand drive vehicles
- veicoli con guida a sinistra
50 km/h
1500 mm
500mm
R
EN: R = Point corresponding to the driver's side (hip), valid for 95% of male adults.
IT: R = Punto corrispondente al fianco del conducente (anca), valido per il 95% dei maschi adulti.


POLE IMPACT
- left-hand drive vehicles
- veicoli con guida a sinistra
29 km/h
pole diameter /diametro palo = 254 mm

Test results compilation

Marque et modèle	Équipement de sécurité	Notation globale				
Toyota Prius	De série	★★★★★	92%	82%	77%	85%
Mercedes-Benz E-Class	De série	★★★★★	95%	90%	77%	62%
VW Tiguan	De série	★★★★★	96%	84%	72%	68%
Kia Niro	Pack sécurité	★★★★★	91%	80%	70%	81%
Toyota Hilux	Pack sécurité	★★★★★	93%	82%	83%	63%
Hyundai Ioniq	De série	★★★★★	91%	80%	70%	82%
Audi Q2	De série	★★★★★	93%	86%	70%	70%

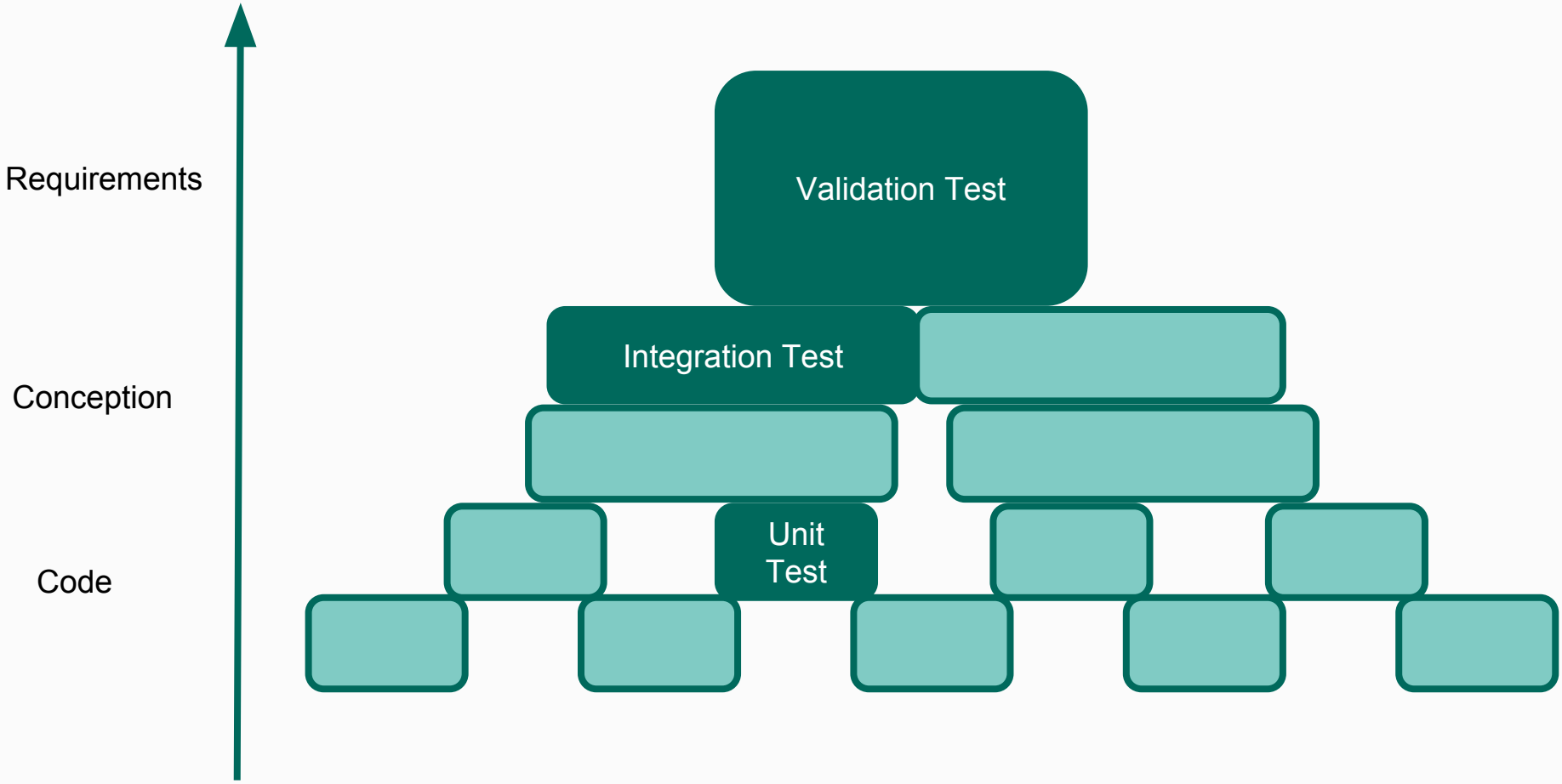
Test vs Trial vs Debugging

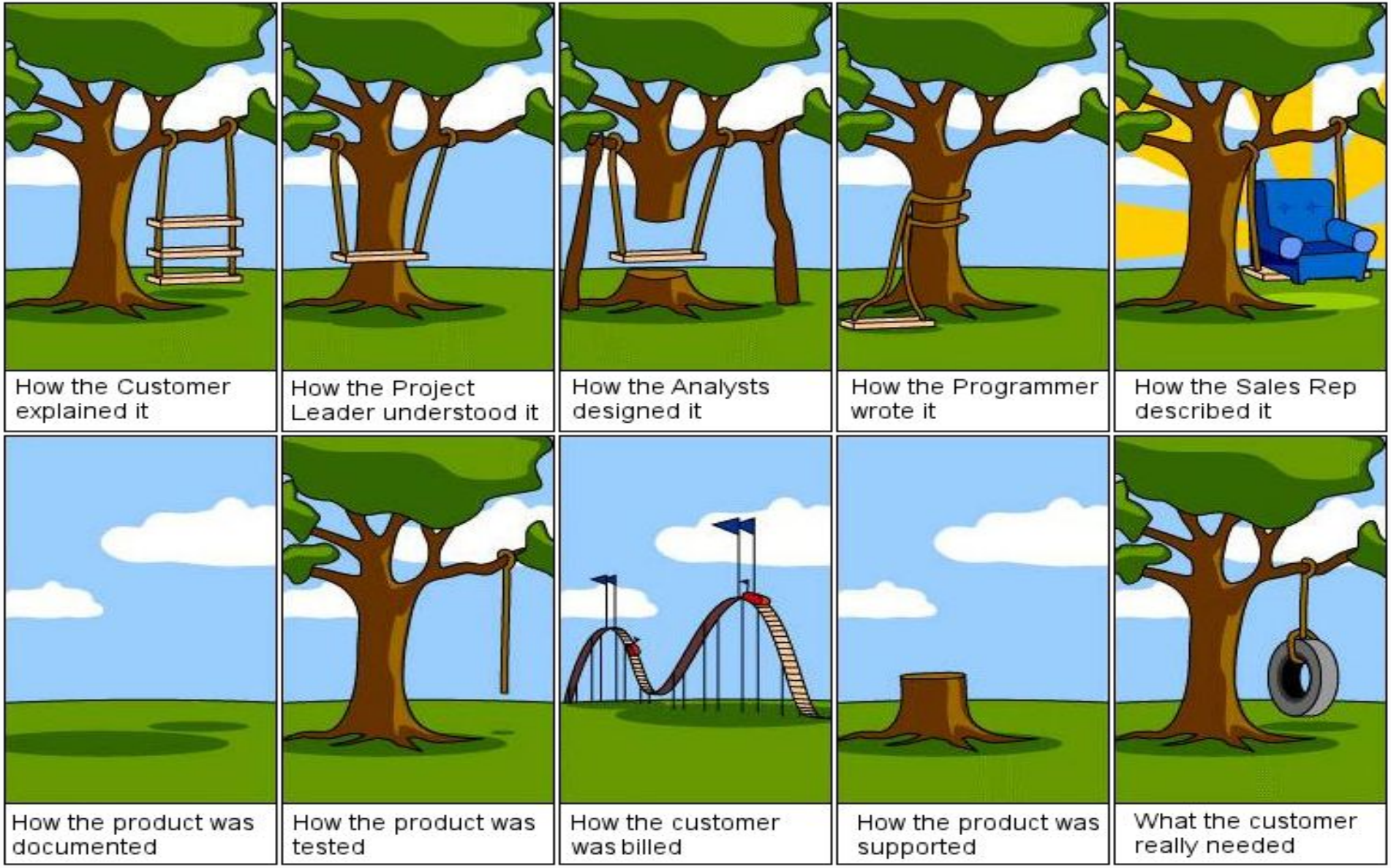
Test → Reproducible (mitigation costs)

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.

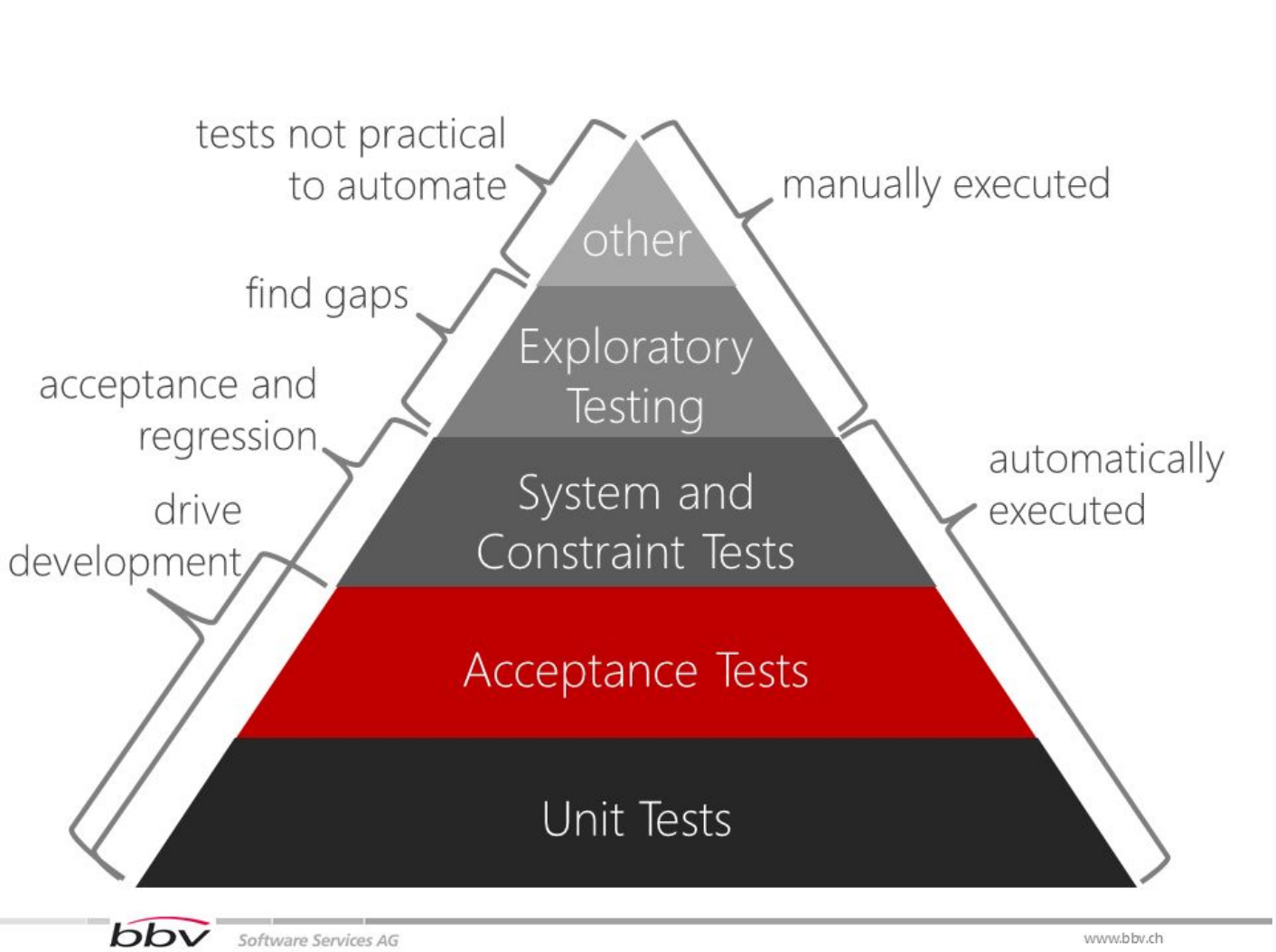
Acceptance Test

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



JUnit 5

Software Testing by example

Example with JUnit

```java
public class User {
    private final String username;
    private String pwd;
    private String token;
    private long tokenDate;
    private UserType userType;

    public User(String username, String pwd) { this(username, pwd, UserType.PLAYER); }

    public User(String username, String pwd, UserType userType) {
        this.username = username;
        this.pwd = pwd;
        this.userType = userType;
        this.tokenDate = -1;
    }

    public String getUsername() { return username; }

    public String getPwd() { return pwd; }

    public void setPwd(String pwd) { this.pwd = pwd; }

    public String getToken() { return token; }

    public void setToken(String token) {
        if (token != null) {
            this.token = token;
            this.tokenDate = System.currentTimeMillis();
        } else {
            this.tokenDate = -1;
        }
    }

    public long getTokenDate() { return tokenDate; }

    public UserType getUserType() { return userType; }
}
```

Test examples

```
public class UserTest {


    @Test
    public void checkTokenDateWithToken() {
        User user=new User("Test","test");
        long start=System.currentTimeMillis();
        user.setToken("AP");
        long stop=System.currentTimeMillis();
        assertEquals("AP",user.getToken());
        assertTrue(start <= user.getTokenDate()
                && user.getTokenDate() <= stop);
    }

    @Test
    public void checkTokenDateWithEmptyToken() {
        User user=new User("Test","test");
        user.setToken(null);
        assertEquals("Not a good toke, haven't to have a date",
                -1, user.getTokenDate());
    }

}
```

Clean test examples

```java
public class UserTest {
    public User user;

    @Before
    public void setup(){
        user=new User("Test","test");
    }


    @Test
    public void checkTokenDateWithToken() {
        long start=System.currentTimeMillis();
        user.setToken("AP");
        long stop=System.currentTimeMillis();
        assertEquals("AP",user.getToken());
        assertTrue(start <= user.getTokenDate()
                && user.getTokenDate() <= stop);
    }

    @Test
    public void checkTokenDateWithEmptyToken() {
        user.setToken(null);
        assertEquals("Not a good toke, haven't to have a date",
                -1, user.getTokenDate());
    }

}
```

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

Trace test example

```java
public class TraceTest {

    @BeforeClass
    public static void setUpBeforeClass(){
        System.out.println("BeforeClass");
    }

    @Before
    public void setUp() {
        System.out.println("Before");
    }

    @Test
    public void test1(){
        System.out.println("Test 1");
    }

    @Test
    public void test2(){
        System.out.println("Test 2");
    }

    @After
    public void tearDown() {
        System.out.println("After");
    }

    @AfterClass
    public static void tearDownAfterClass(){
        System.out.println("AfterClass");
    }
}
```

```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running fr.unice.iut.authentication.UserTest
Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.053 sec
Running fr.unice.iut.authentication.TraceTest
BeforeClass
Before
Test 1
After
Before
Test 2
After
AfterClass
Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.001 sec

Results :

Tests run: 4, Failures: 0, Errors: 0, Skipped: 0
```

Mock testing

Purpose :

Simulate the behaviour of real objects

Allows to deal with :

Testing at interface level

Blacked-box component

JUnit + Mockito example

```
public class AuthenticationServiceTest extends JerseyTest {

    @Mock
    private UserBase userBase;

    private AuthenticationService authenticationService;

    @Override
    public ResourceConfig configure(){
        MockitoAnnotations.initMocks(this);
        userBase=mock(UserBase.class);
        when(userBase.getUserByName("mock")).thenReturn(new User("mock","mock"));
        authenticationService=new AuthenticationService(userBase);
        return new ResourceConfig().register(authenticationService);
    }

    @Test
    public void test(){
        assertEquals(401,authenticationService.authenticateUser("t","test").getStatus());
        assertEquals(200,authenticationService.authenticateUser("mock","mock").getStatus());
    }

}
```

Free mobile kick-off : an epic journey

- January 10th, 2012 : kick-off
- January 10th, 2012 + 1h : server down
- January 10th, 2012 + 7d : server up

- Reason :
  - Bandwidth ? Scale issues ?
  - No, tests …

- January 10th, 2012 - 5h : A dev push a code without tests
  - First error : last minute commit → No way
  - Second error : no testing before production → No way
  - → Fired !

Let’s start this into an IDE !

Testing with Maven and Jenkins

- Maven
  - Test directory
  - Launched with test stage (test, package, install)
- Jenkins
  - Execute test stage automatically

→ Working in the same way with Gradle (Android), npm (Javascript) !

→ For sprint #1, you need to have **smart** tests on your project launched by Jenkins

Talking about Gradle and Jenkins ...

- Need to update .gitignore file .jar → ^(*gradle-wrapper)*.jar

- Generate a gradle wrapper (or use the one from Android Studio) and add it to git
  - Gradle wrapper = All the directory gradle + gradlew at android project root

- In the JenkinsFile gradle assemble → ./gradlew assemble

- git update-index --chmod=+x gradlew

- Commit and push to your repo

?