

# Testing a Scout Application with JUnit and Jubula (Presented by BSI)

EclipseCon 2013

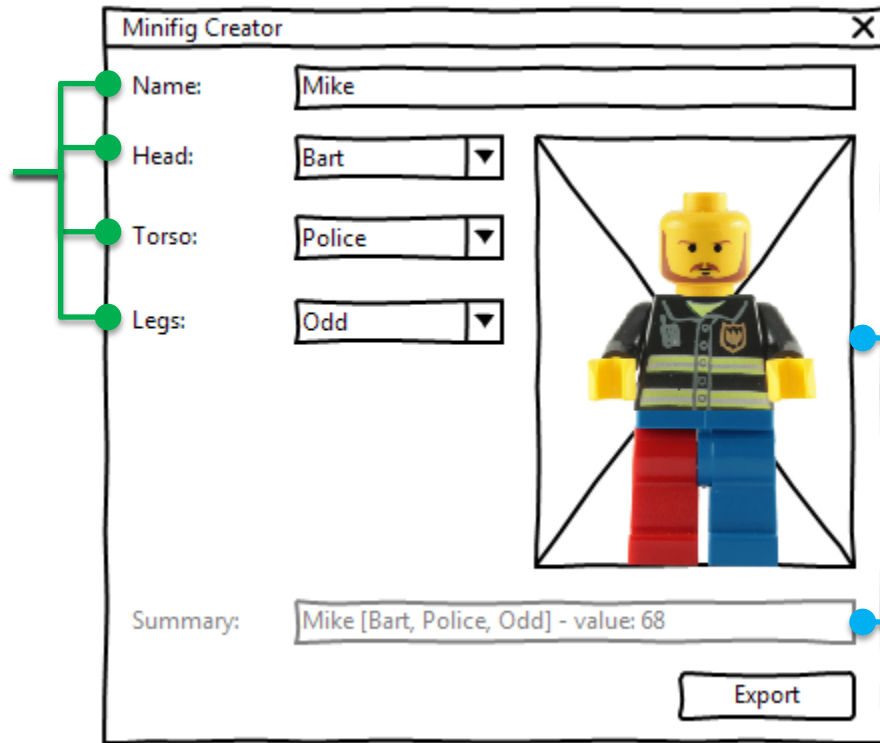
Alexandra Schladebeck [BREDEX GmbH]

Jérémie Bresson [BSI Business Systems Integration AG]

# APPLICATION UNDER TEST

# Requirements (1)

Modification of the input fields...

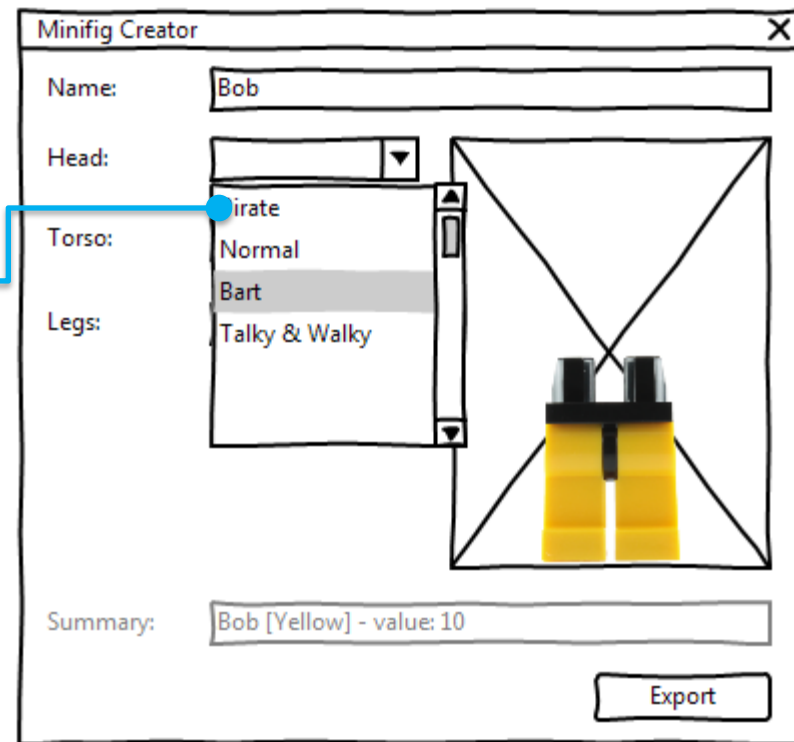


... will update the image and summary fields

## Requirements (2)

Only the available parts are listed in the field

Is only a part available, the field is disabled



## Requirements (3)

Minifig Creator

Name:

Head:

Torso:

Legs:

Summary:

Export

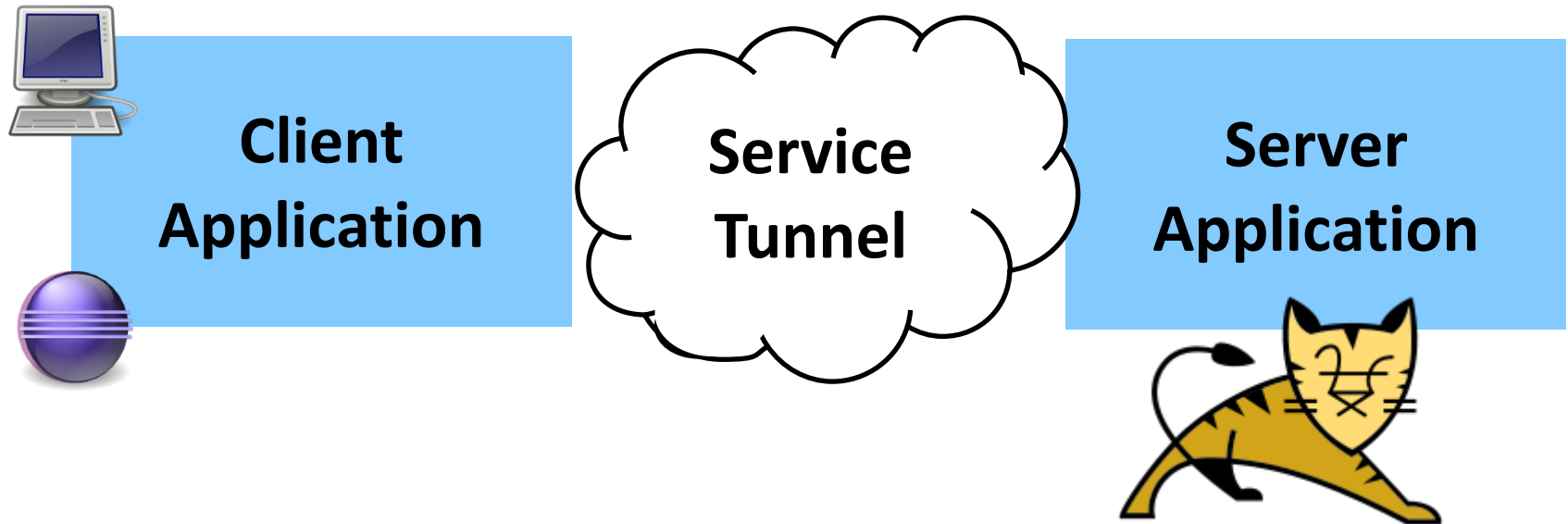
Export button:

- validate the form
- register the minifig in the server
- reset the form

# ECLIPSE SCOUT

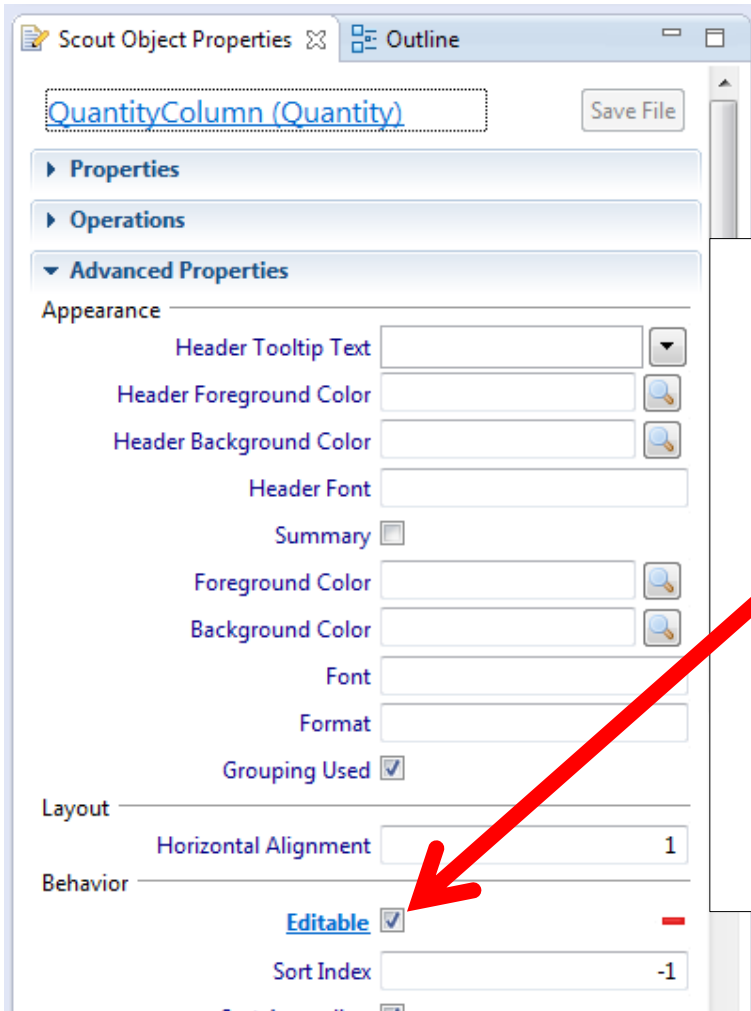


# Client / Server integration





# Eclipse Scout SDK: quick results



Scout Object Properties Outline

QuantityColumn (Quantity) Save File

Properties

Operations

Advanced Properties

Appearance

Header Tooltip Text

Header Foreground Color

Header Background Color

Header Font

Summary

Foreground Color

Background Color

Font

Format

Grouping Used

Layout

Horizontal Alignment 1

Behavior

Editable

Sort Index -1

```

@Order(10.0)
public class QuantityColumn extends
    AbstractIntegerColumn{

    @Override
    protected boolean getConfiguredEditable() {
        return true;
    }

    @Override
    protected String getConfiguredHeaderText() {
        return Texts.get("Quantity");
    }
}
    
```





# Nice user interface fields

Legs

Blue

Gray

Overall

Jeans

**Odd**

Yellow



Type	Name	Quantity
HEAD	Normal	2
HEAD	Pirate	3
HEAD	Talky & Walky	2
HEAD	Bart	5
LEGS	Blue	3
LEGS	Gray	9
LEGS	Overall	8
LEGS	Jeans	5
LEGS	Odd	7
LEGS	Yellow	4
TORSO	Yellow	8
TORSO	Police	1
TORSO	Suit	1

# JUBULA

## Automated testing – through the GUI

As a user would work – passing through all layers



Test creation, execution, analysis

Drag and drop test creation:

- *No recording*
- *No programming*
- *With the same ♥ as development code*

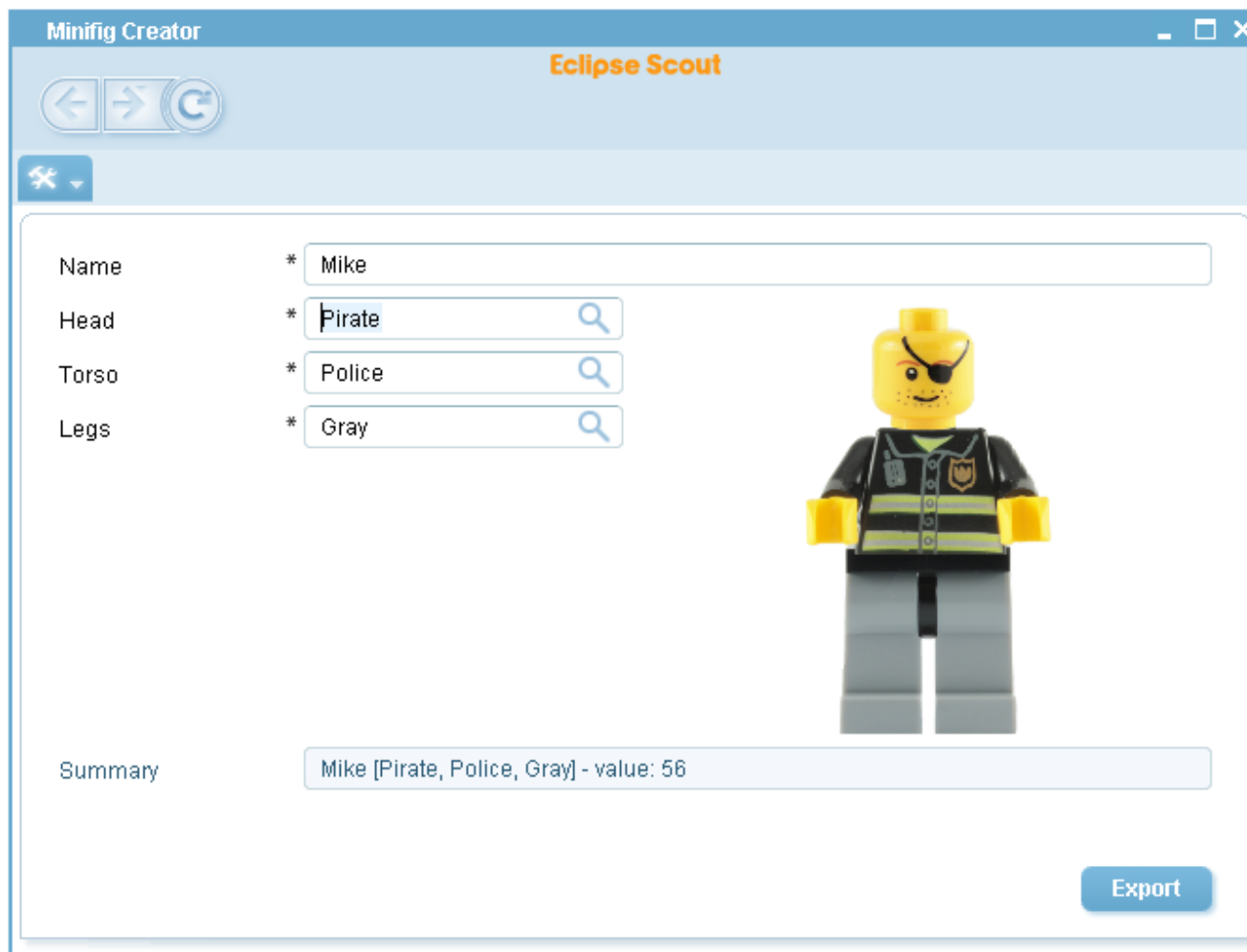
Constant feedback about quality

- *Acceptance testing*
- *Regression testing*



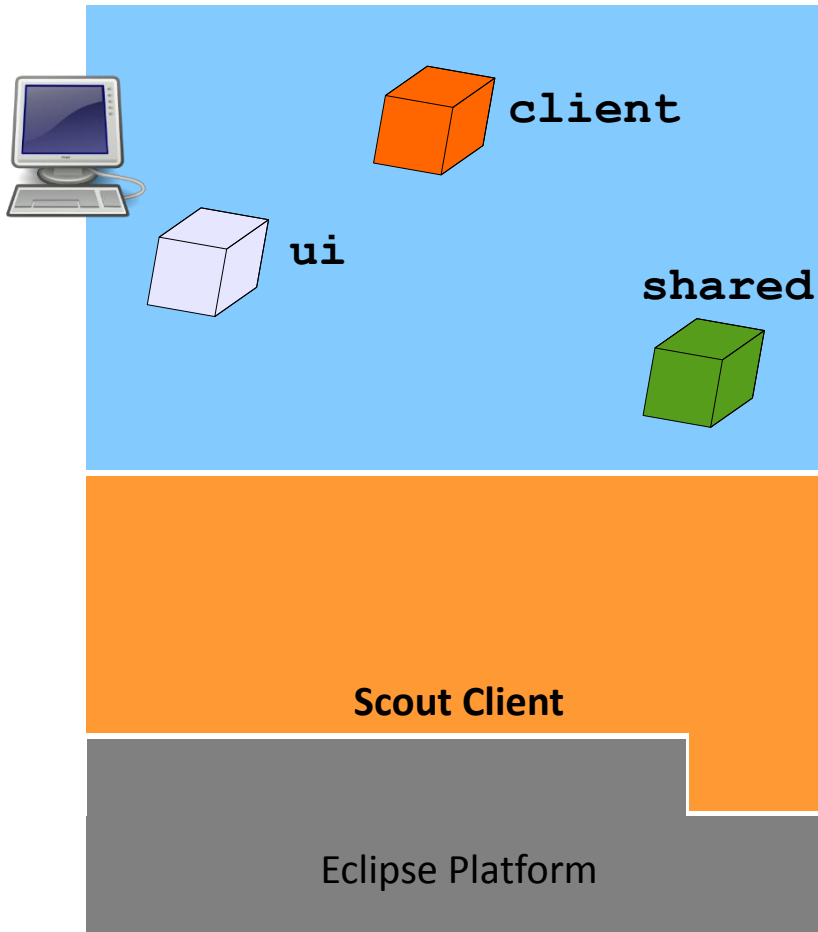
# BACKGROUND

# The application under test

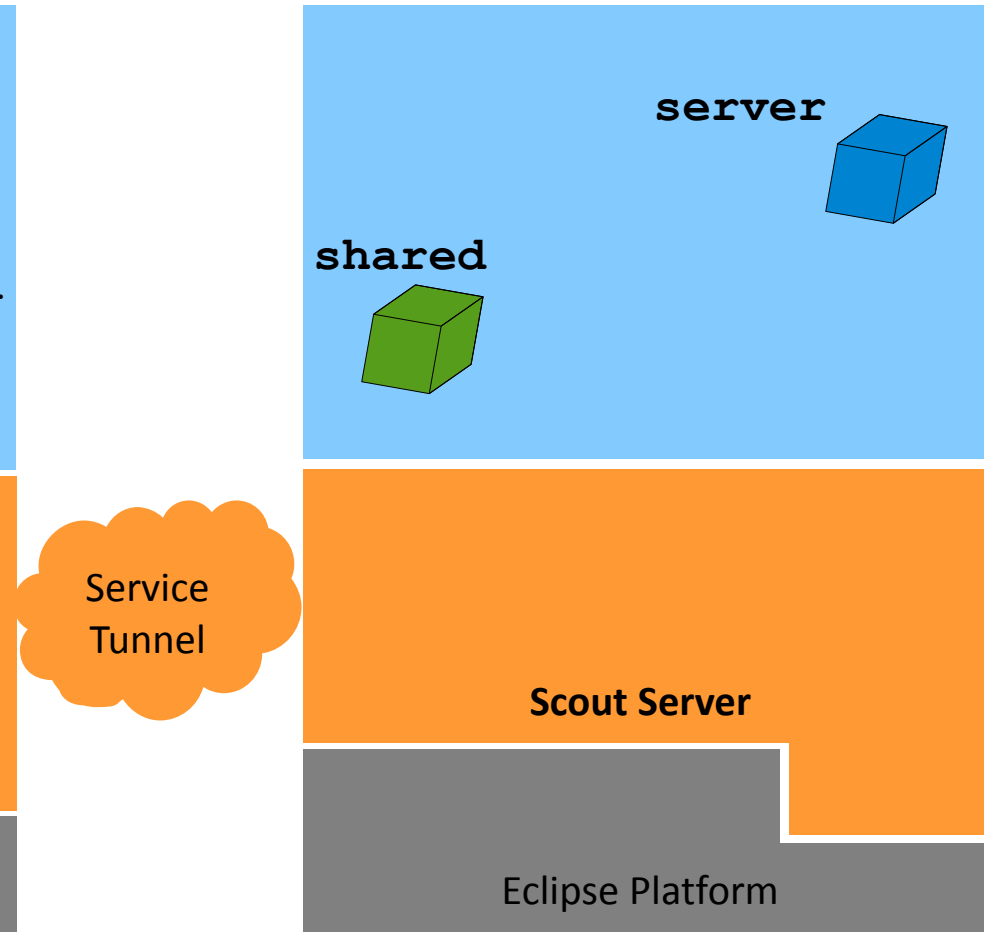


# A Scout application

## Client Application



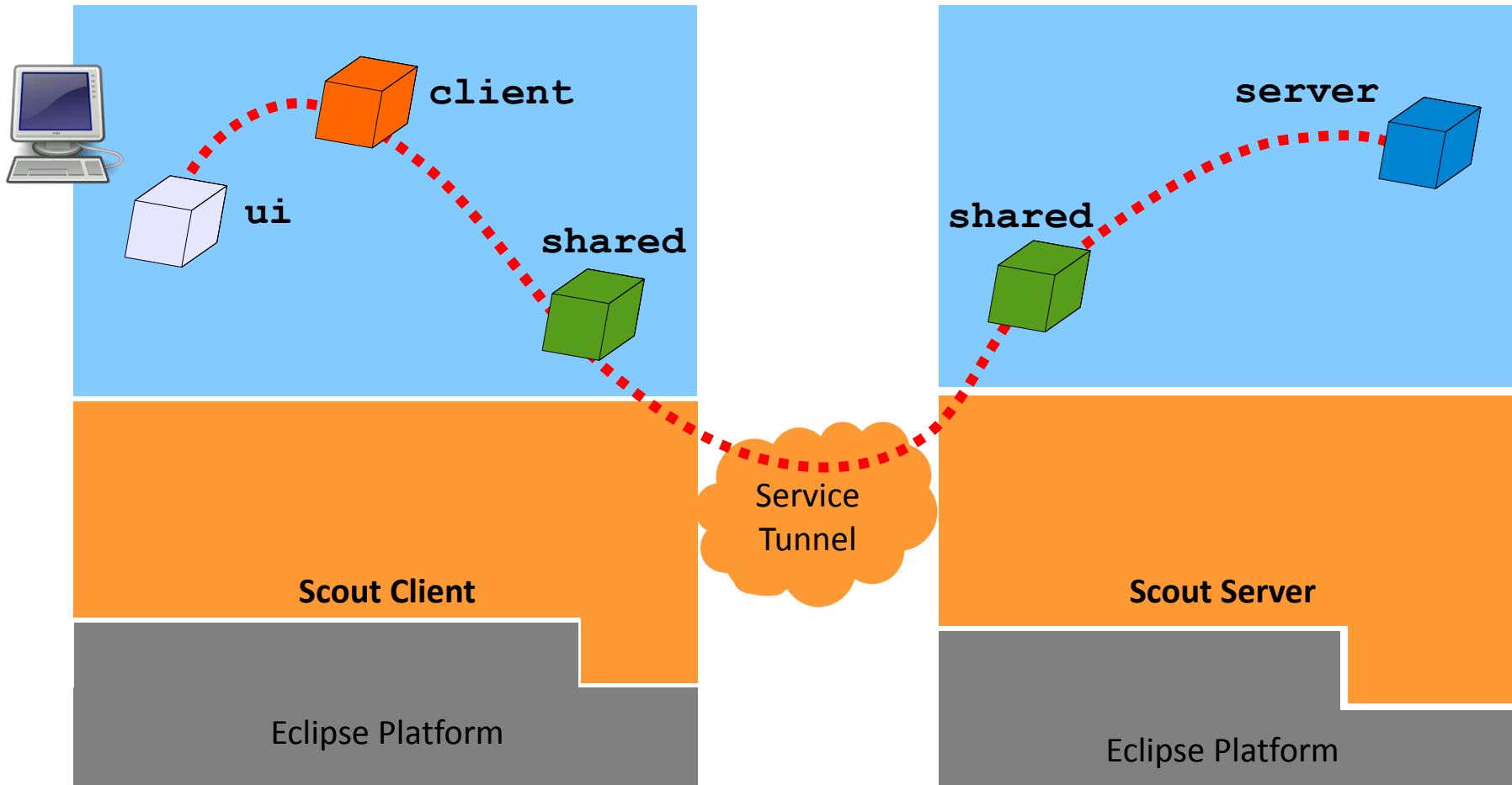
## Server Application



# A Scout application

## Client Application

## Server Application

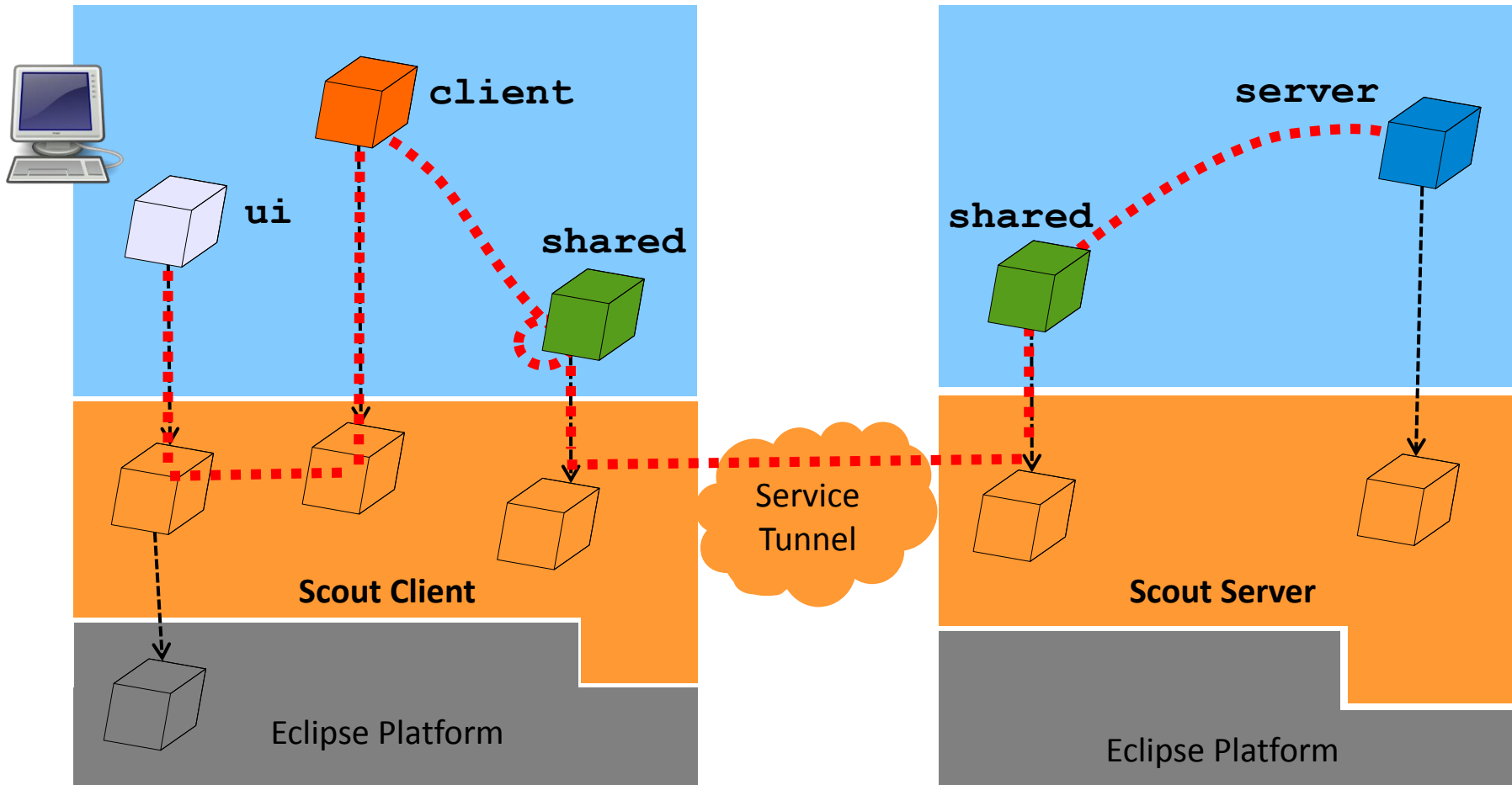




# A Scout application

## Client Application

## Server Application

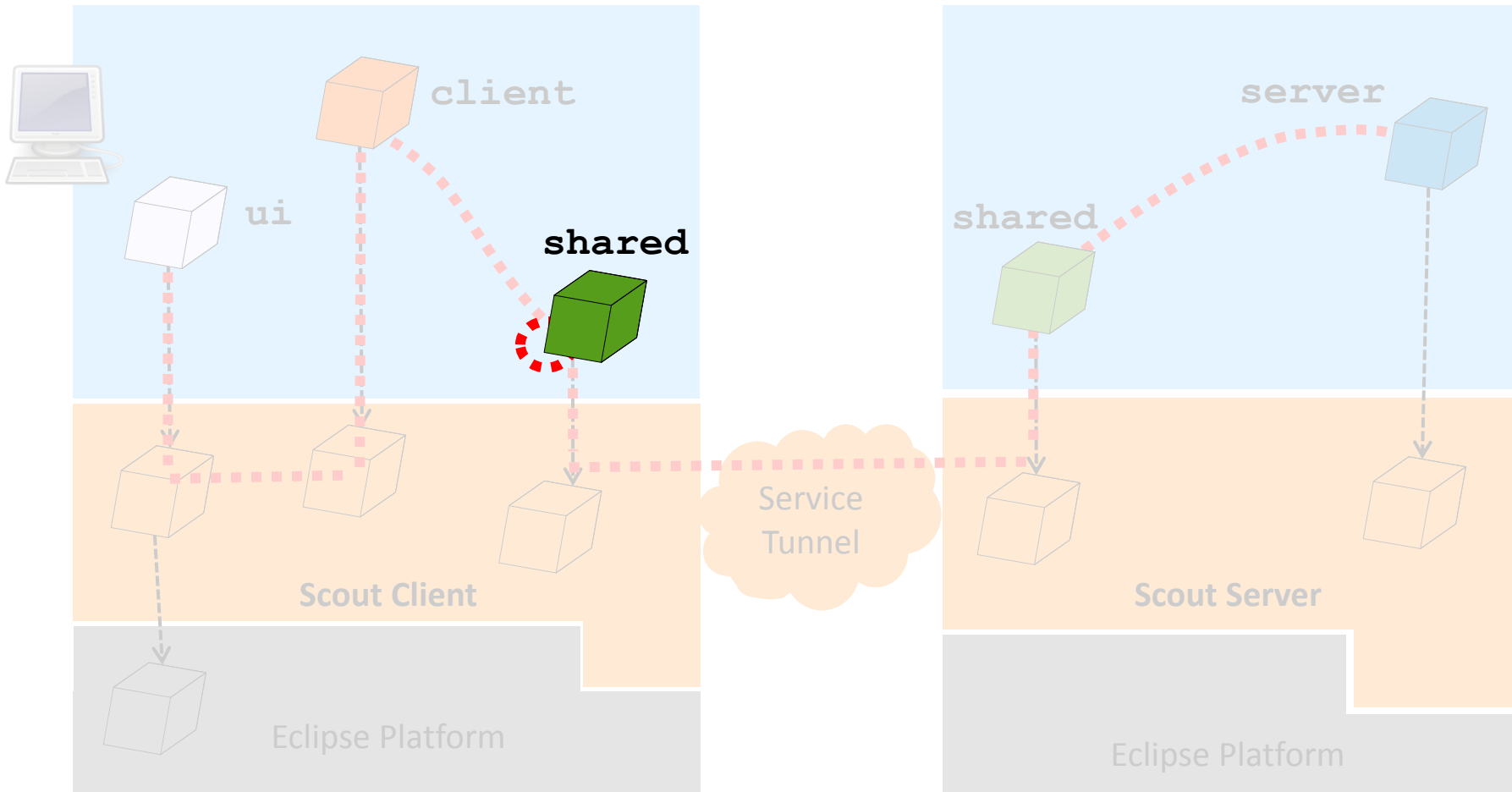


# JUNIT

# Test for logic in the shared plugin

## Client Application

## Server Application



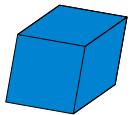
# Where do I put my tests?



- `<app>.client`



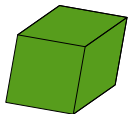
- `<app>.client.test`



- `<app>.server`



- `<app>.server.test`

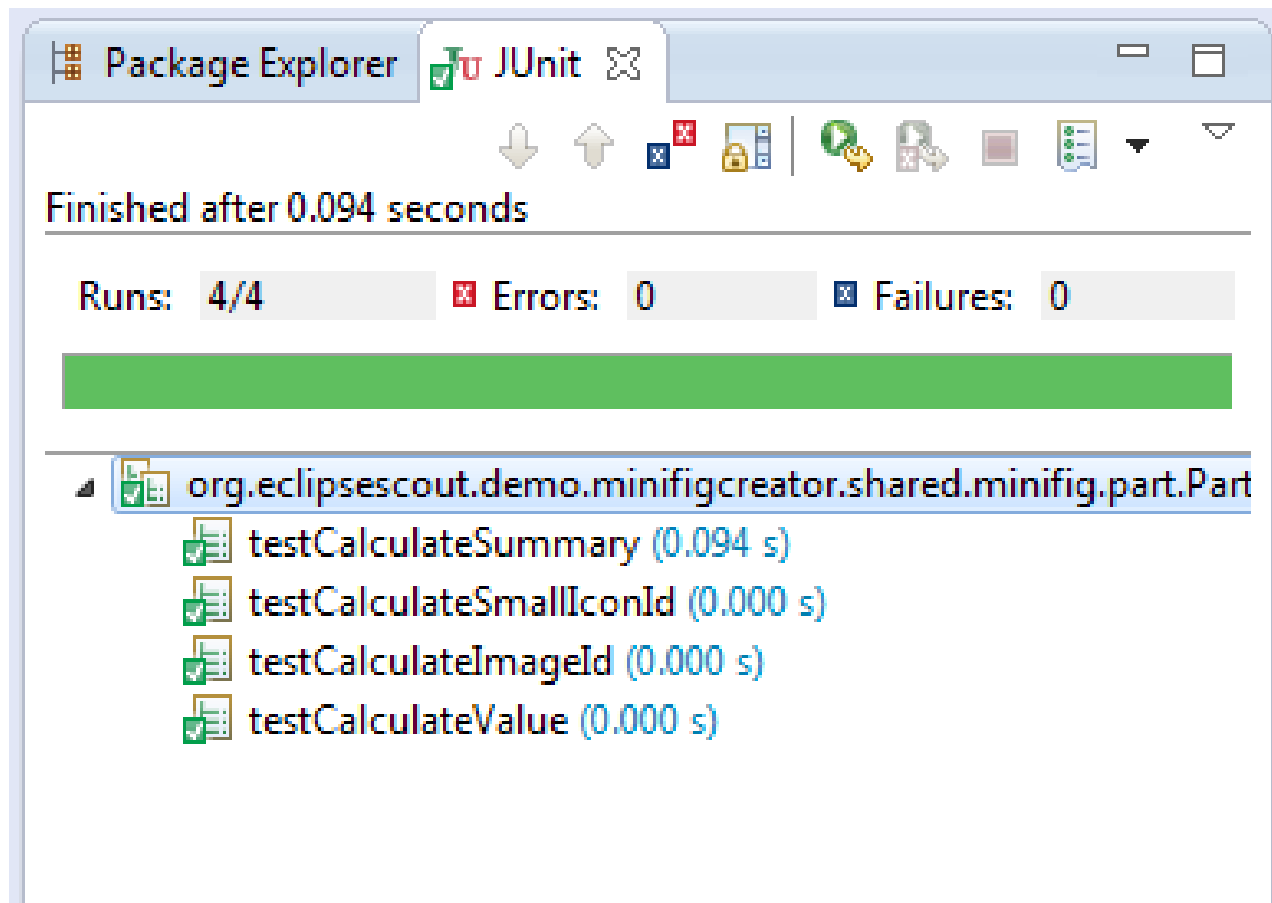


- `<app>.shared`



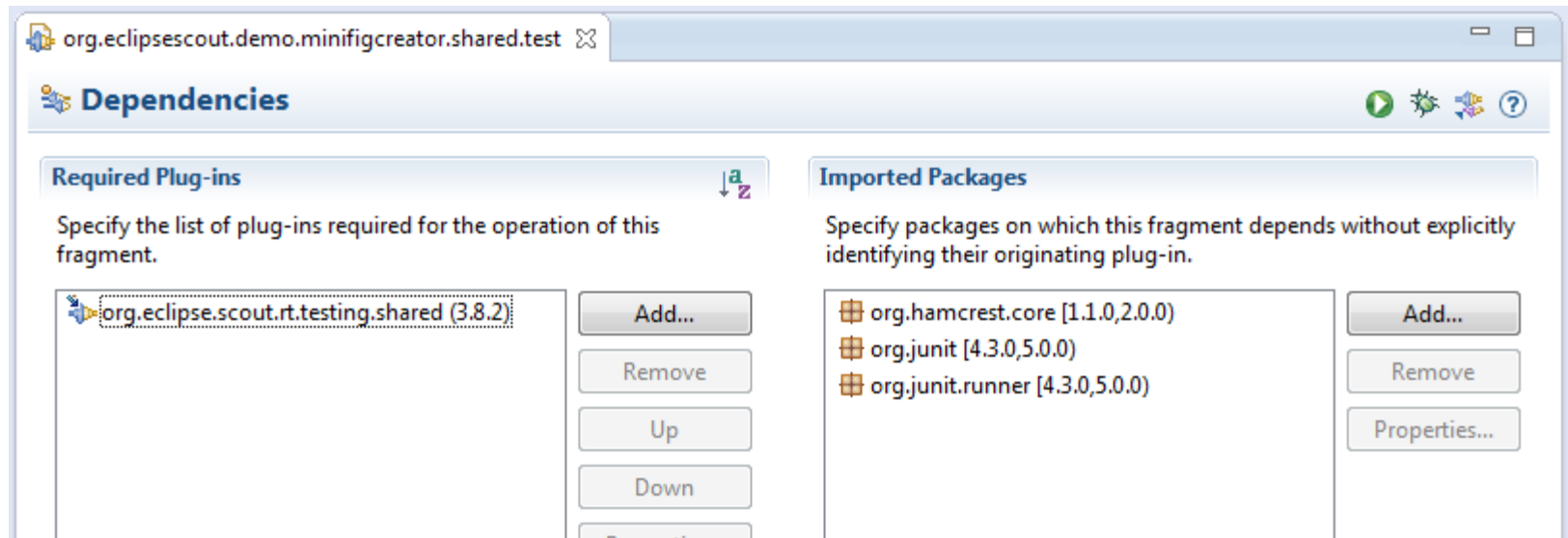
- `<app>.shared.test`

# Running the tests from Eclipse



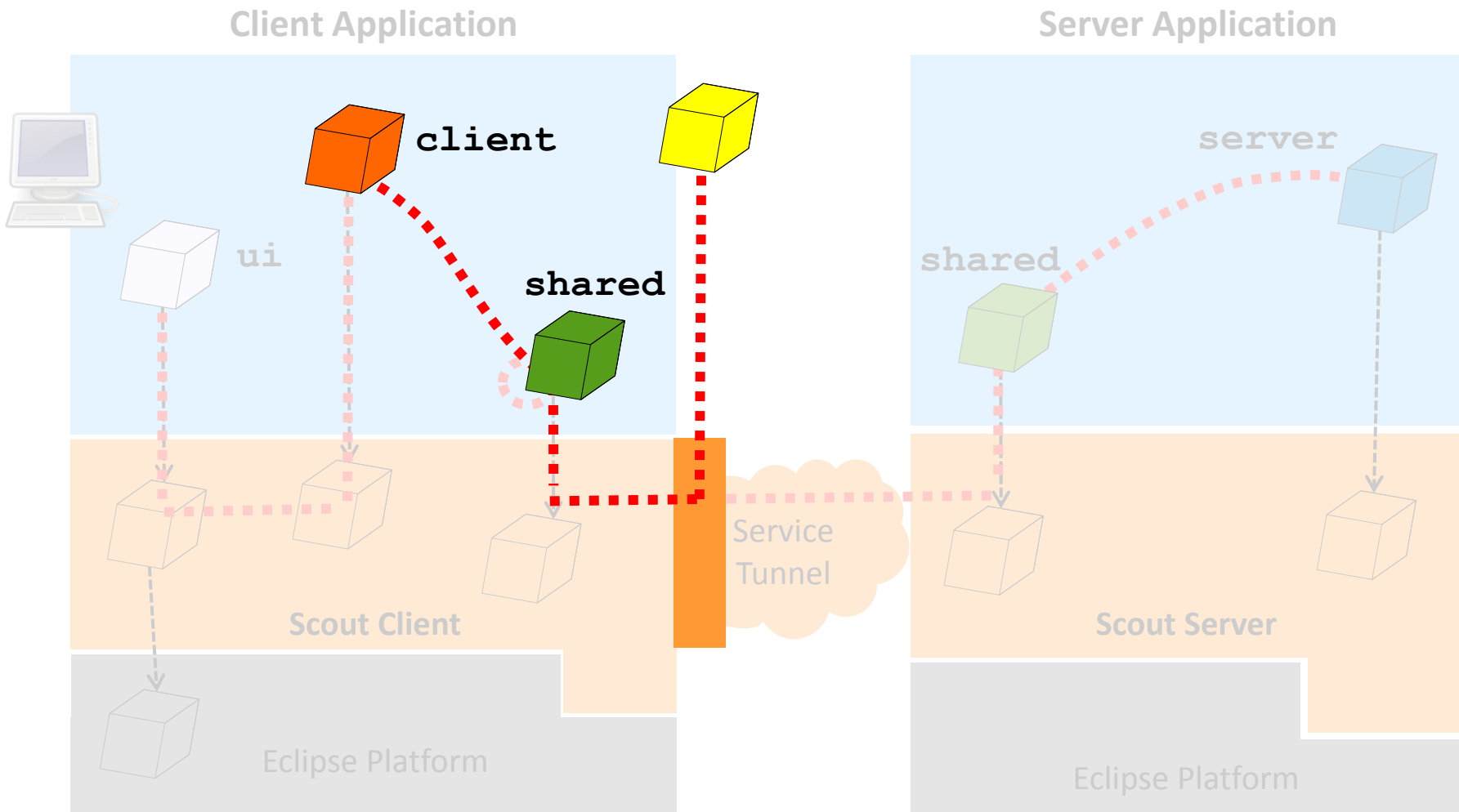
# Using maven + tycho

- Add the maven nature to the project. Maven packaging:
  - `eclipse-plugin` for plugins
  - `eclipse-test-plugin` for test fragments
  
- Import JUnit as imported Packages:



# JUNIT IN A SCOUT CONTEXT

# Test with the Scout services





## @RunWith annotation

Annotate the test class with the Annotation:

```
@RunWith(ScoutClientTestRunner.class)
public class DesktopFormTest {

    //...

}
```

It adds:

- Equinox OSGi Runtime
- Scout Context, Services, ...

## TestingUtility.registerServices(..)

TestingUtility allows to register other implementation for services.

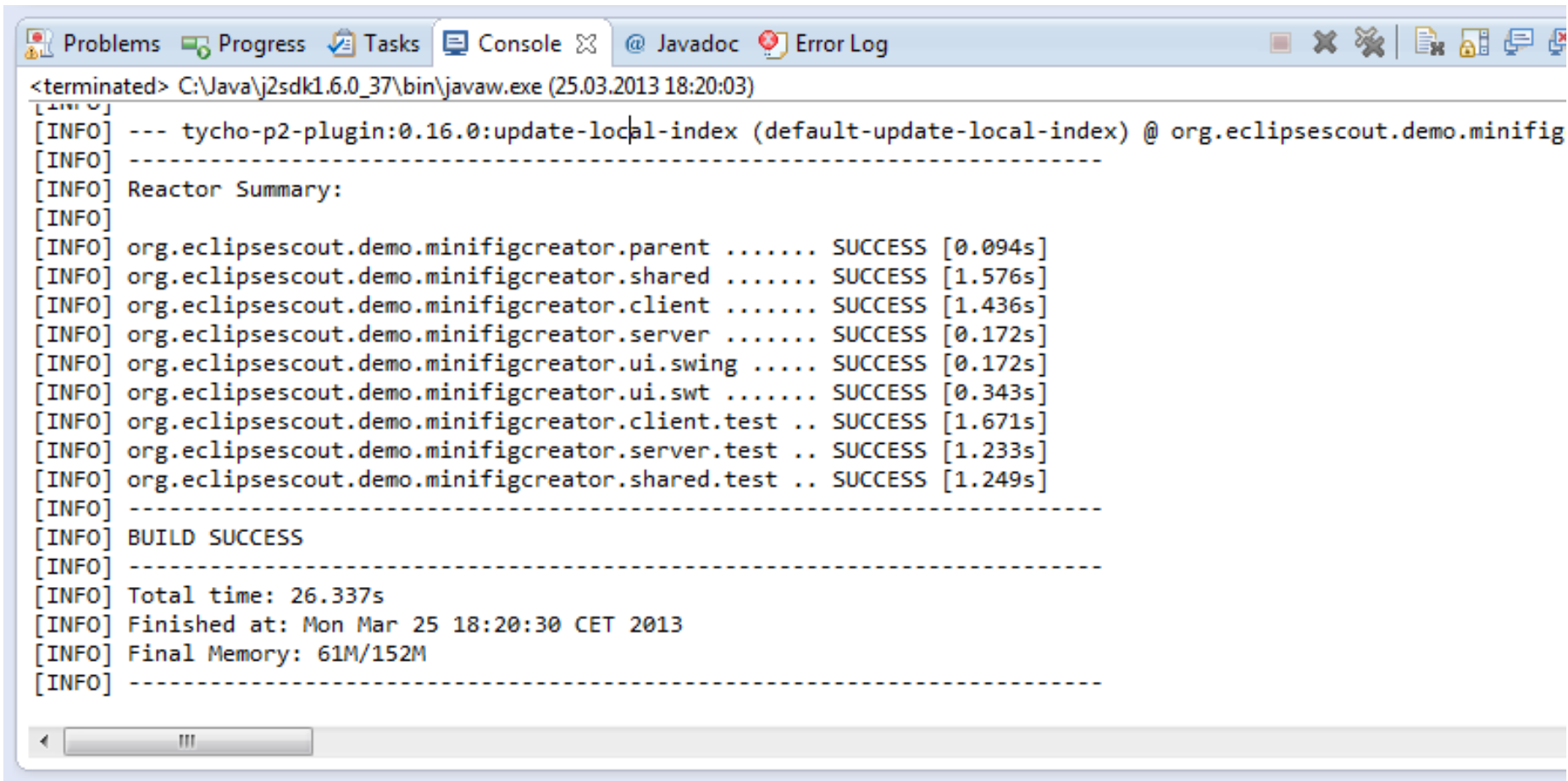
```
public class P_DesktopProcessService
    implements IDesktopProcessService {

    //Controlled implementation
}
```

The Service is

- registred before each tests @Before
- unregistred after each test @After

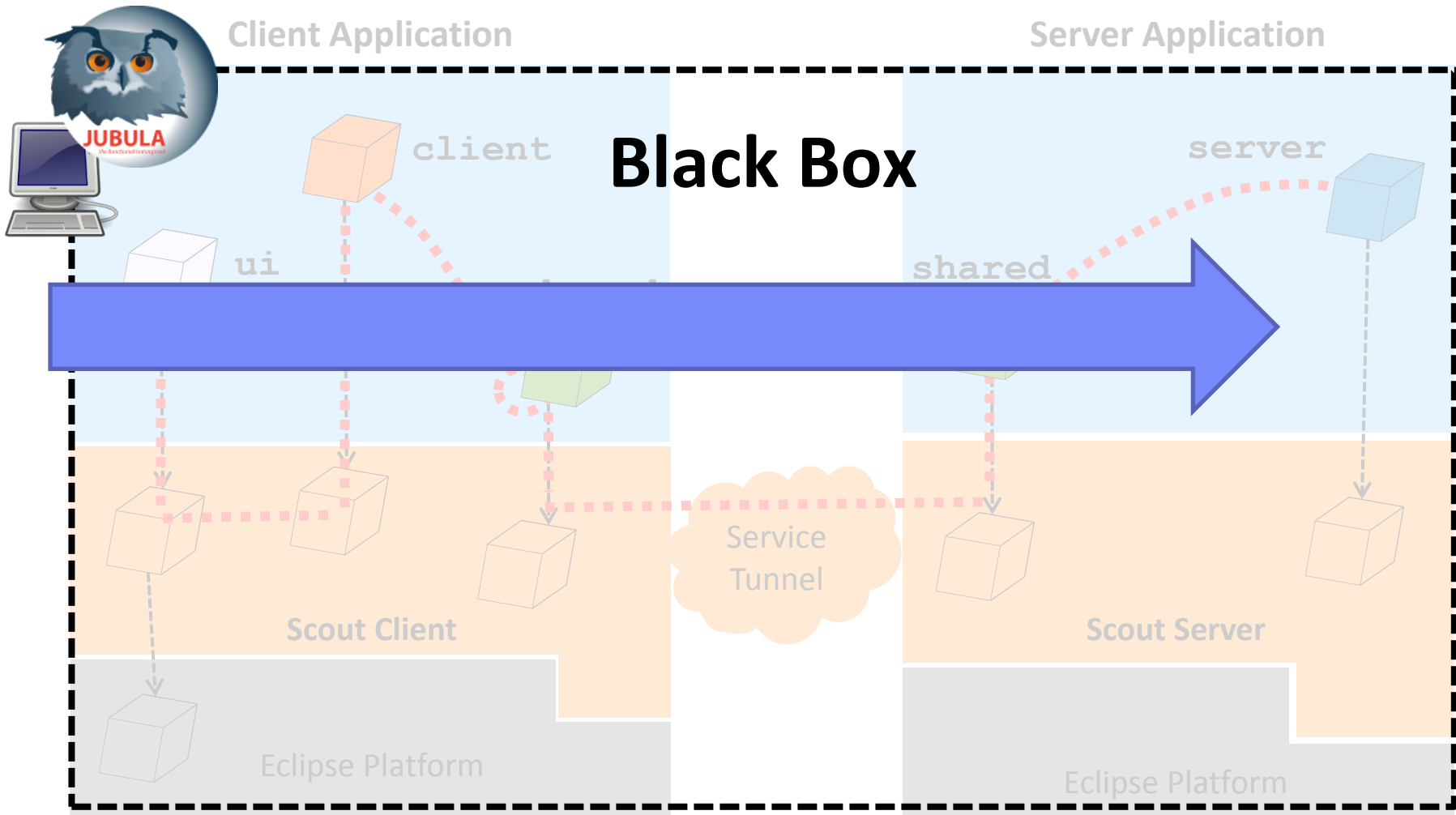
# Running the tests with maven + tycho build



```
<terminated> C:\Java\jdk1.6.0_37\bin\javaw.exe (25.03.2013 18:20:03)
[INFO] --- tycho-p2-plugin:0.16.0:update-local-index (default-update-local-index) @ org.eclipsescout.demo.minifig
[INFO] -----
[INFO] Reactor Summary:
[INFO]
[INFO] org.eclipsescout.demo.minifigcreator.parent ..... SUCCESS [0.094s]
[INFO] org.eclipsescout.demo.minifigcreator.shared ..... SUCCESS [1.576s]
[INFO] org.eclipsescout.demo.minifigcreator.client ..... SUCCESS [1.436s]
[INFO] org.eclipsescout.demo.minifigcreator.server ..... SUCCESS [0.172s]
[INFO] org.eclipsescout.demo.minifigcreator.ui.swing ..... SUCCESS [0.172s]
[INFO] org.eclipsescout.demo.minifigcreator.ui.swt ..... SUCCESS [0.343s]
[INFO] org.eclipsescout.demo.minifigcreator.client.test .. SUCCESS [1.671s]
[INFO] org.eclipsescout.demo.minifigcreator.server.test .. SUCCESS [1.233s]
[INFO] org.eclipsescout.demo.minifigcreator.shared.test .. SUCCESS [1.249s]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 26.337s
[INFO] Finished at: Mon Mar 25 18:20:30 CET 2013
[INFO] Final Memory: 61M/152M
[INFO] -----
```

# JUBULA

# Test with Jubula



# Using the specification to automate tests

module

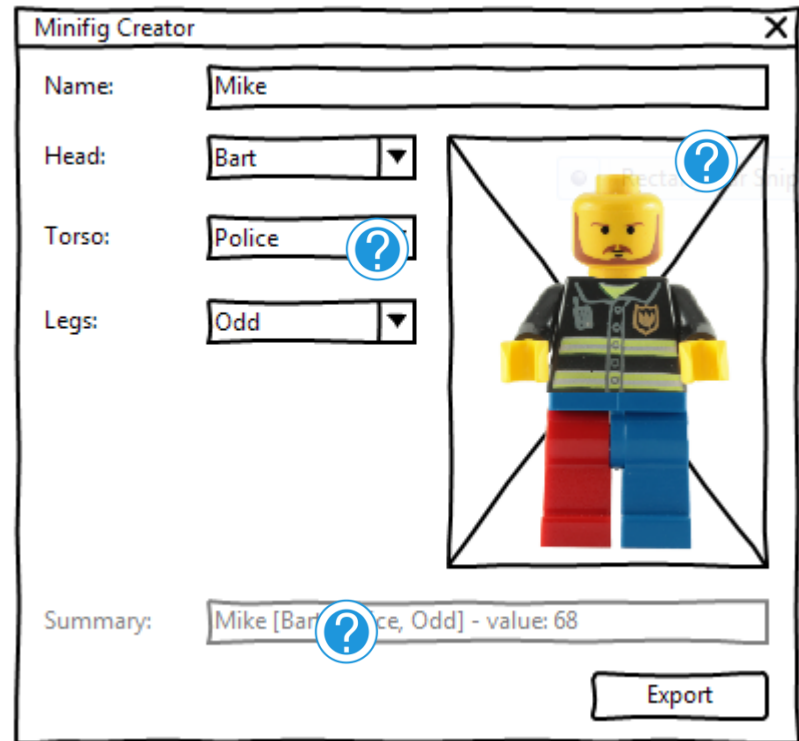
Replace text

Select from smart field

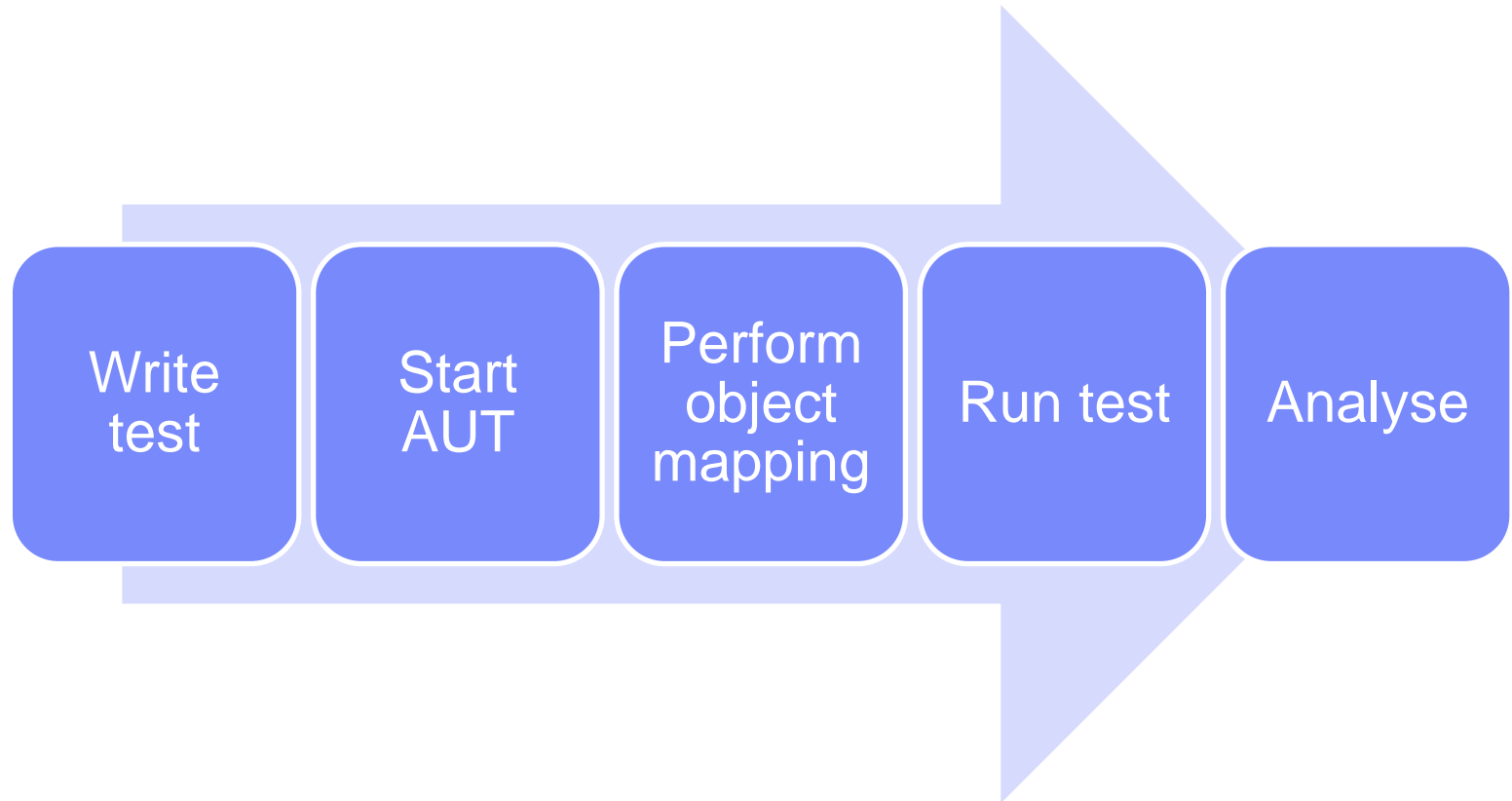
Select from smart field

Select from smart field

Check text



# Testing an application with Jubula



# Demo





# SUMMARY

# Summary

- Developer tests (unit level)
  - Plain JUnit
  - JUnit in a Scout context
  
- User-centric tests
  - Jubula

Sources: [http://wiki.eclipse.org/Scout/Demo#Minifig\\_Application](http://wiki.eclipse.org/Scout/Demo#Minifig_Application)

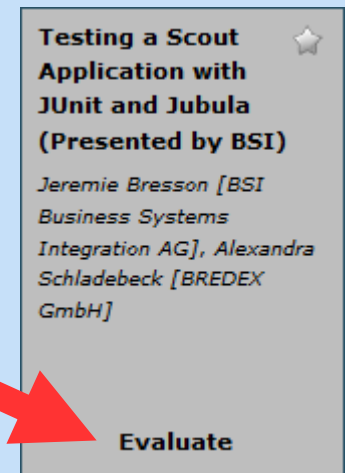
# Give Feedback on the Sessions

1

Sign In: [www.eclipsecon.org](http://www.eclipsecon.org)

2

Select Session Evaluate



Testing a Scout Application with JUnit and Jubula (Presented by BSI)

Jeremie Bresson [BSI Business Systems Integration AG], Alexandra Schladebeck [BREDEX GmbH]

Evaluate

A red arrow points from the 'Evaluate' button in the screenshot to the '2 Select Session Evaluate' step.

3

Vote

