3rd UCAAT User Conference on Advanced Automated Testing

ETSI World Class Standards

Sophia Antipolis, French Riviera

20-22 October 2015

MODEL & TEST GENERATION FROM JUNIT

Presented by Pablo Lamela Seijas

© All rights reserved

PROWESS

Overview

- Introduction
- Architecture of the solution (James)
- Example
- Pilot study and lessons learned

© All rights reserved

INTRODUCTION

What is James and what does it do?

© All rights reserved

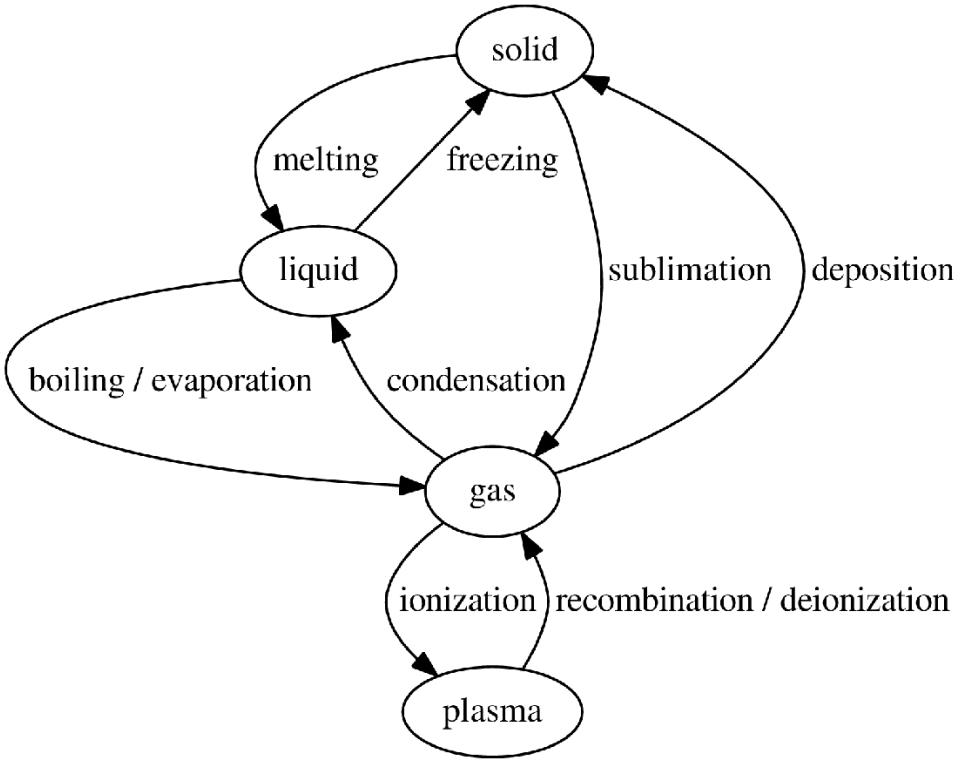
Introduction

- JUnit Tests → Model → Visual diagram
  → New JUnit Tests
- PROWESS project (testing Web Services)
- James tool: https://github.com/palas/james
- Main contribution: combines two approaches

© All rights reserved

Two approaches: Control vs Data

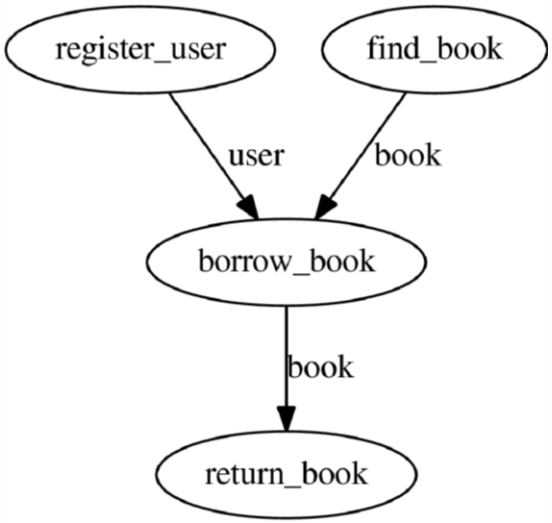
- **Control** flow: How events affect state



© All rights reserved

Two approaches: Control vs Data

- **Data** flow: How data is reused



© All rights reserved

Our approach James…

- Combines both approaches
- So far, targeted at Web Services
- Other interfaces could be tested

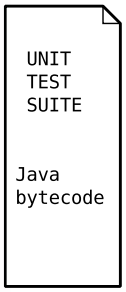
© All rights reserved

ARCHITECTURE OF JAMES

Components and output format

© All rights reserved

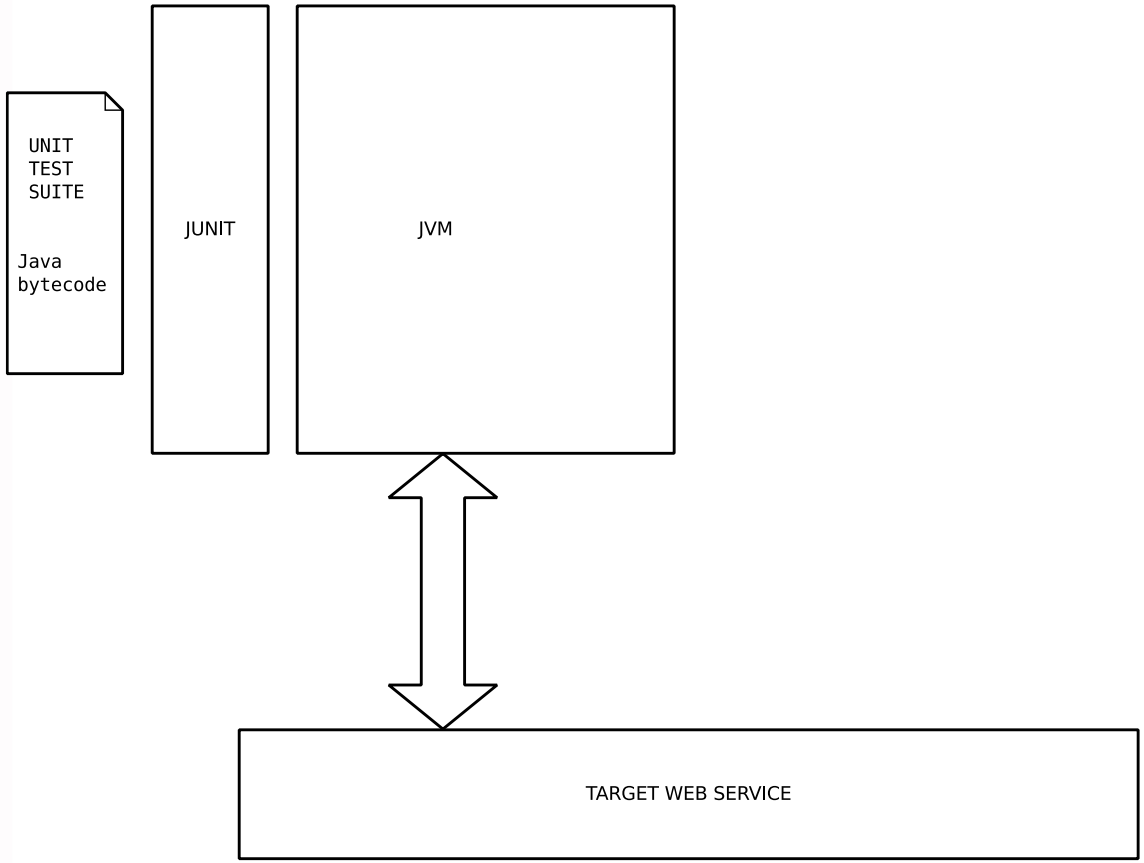
James structure



TARGET WEB SERVICE

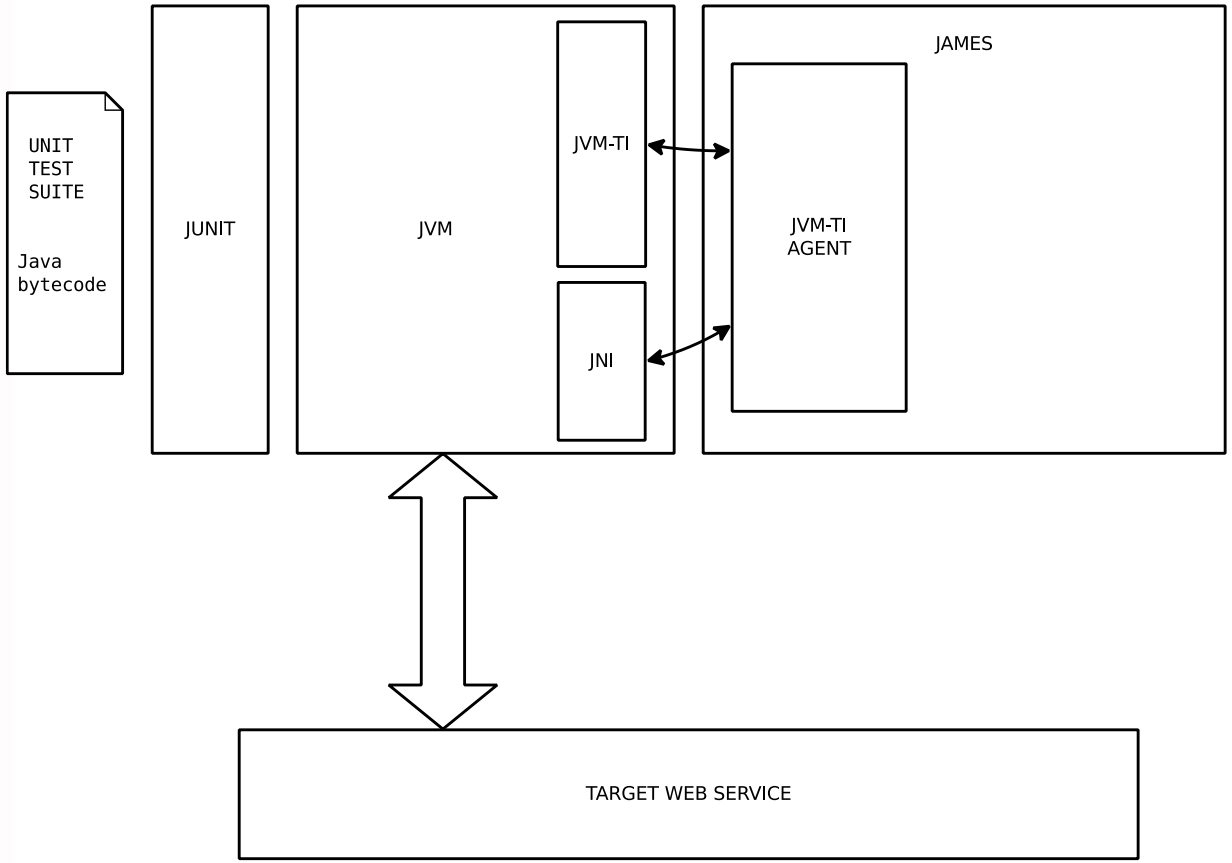
© All rights reserved

James structure



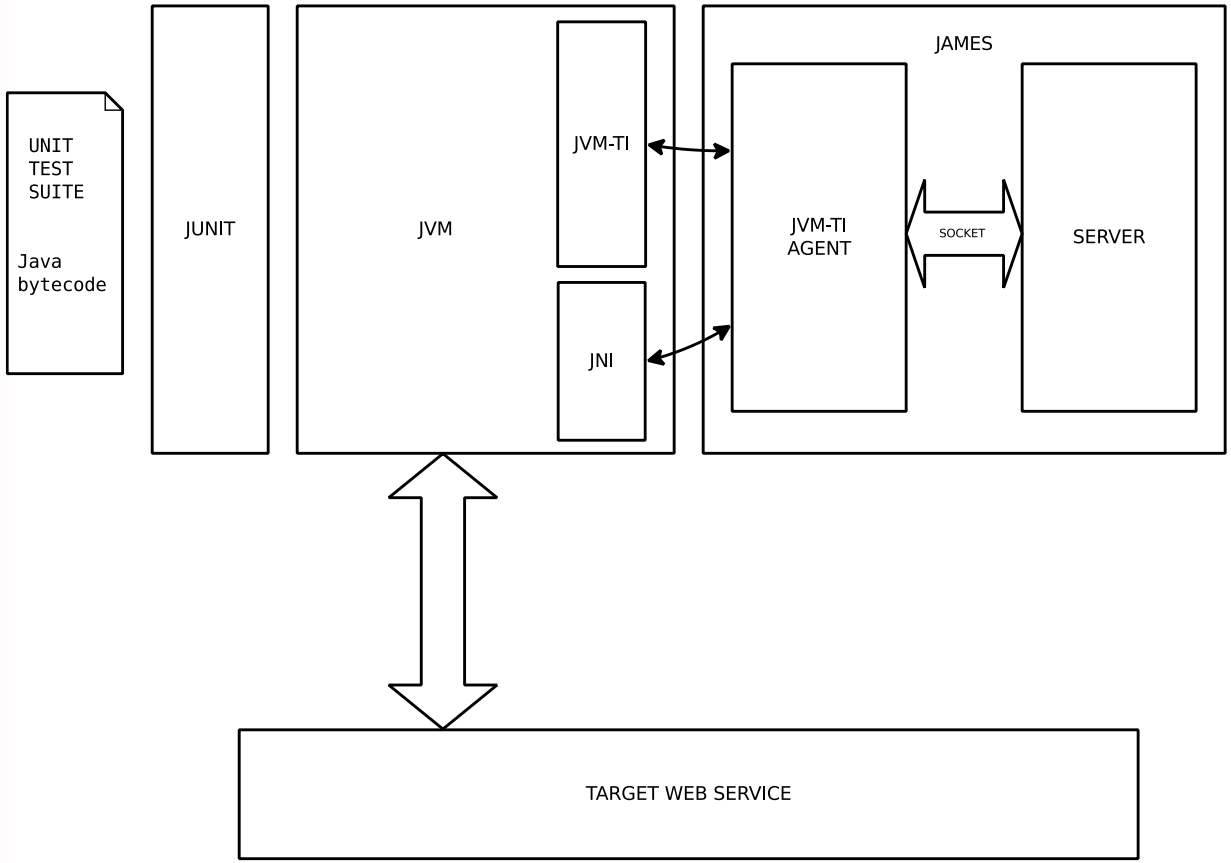
© All rights reserved

James structure



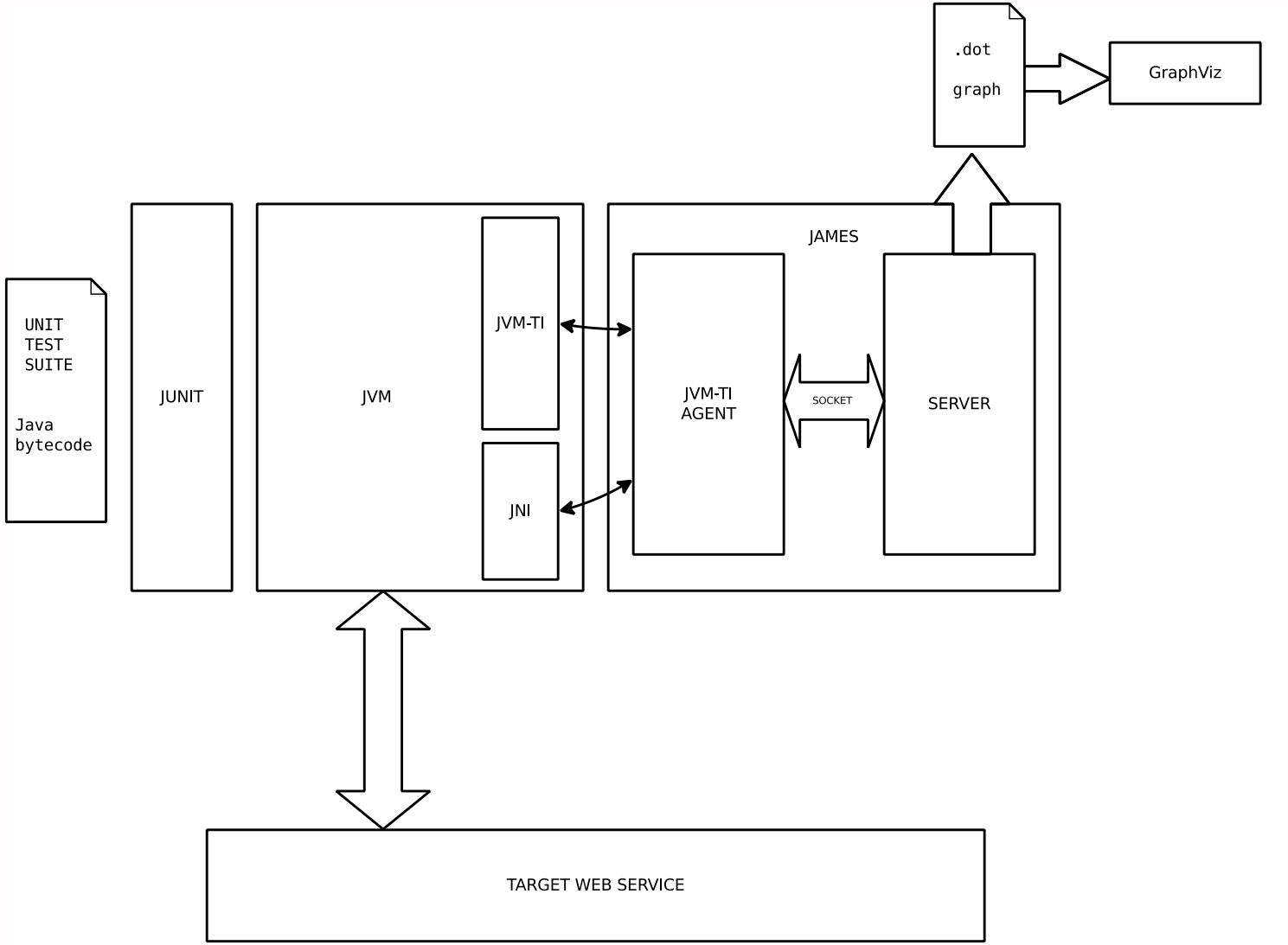
© All rights reserved

James structure



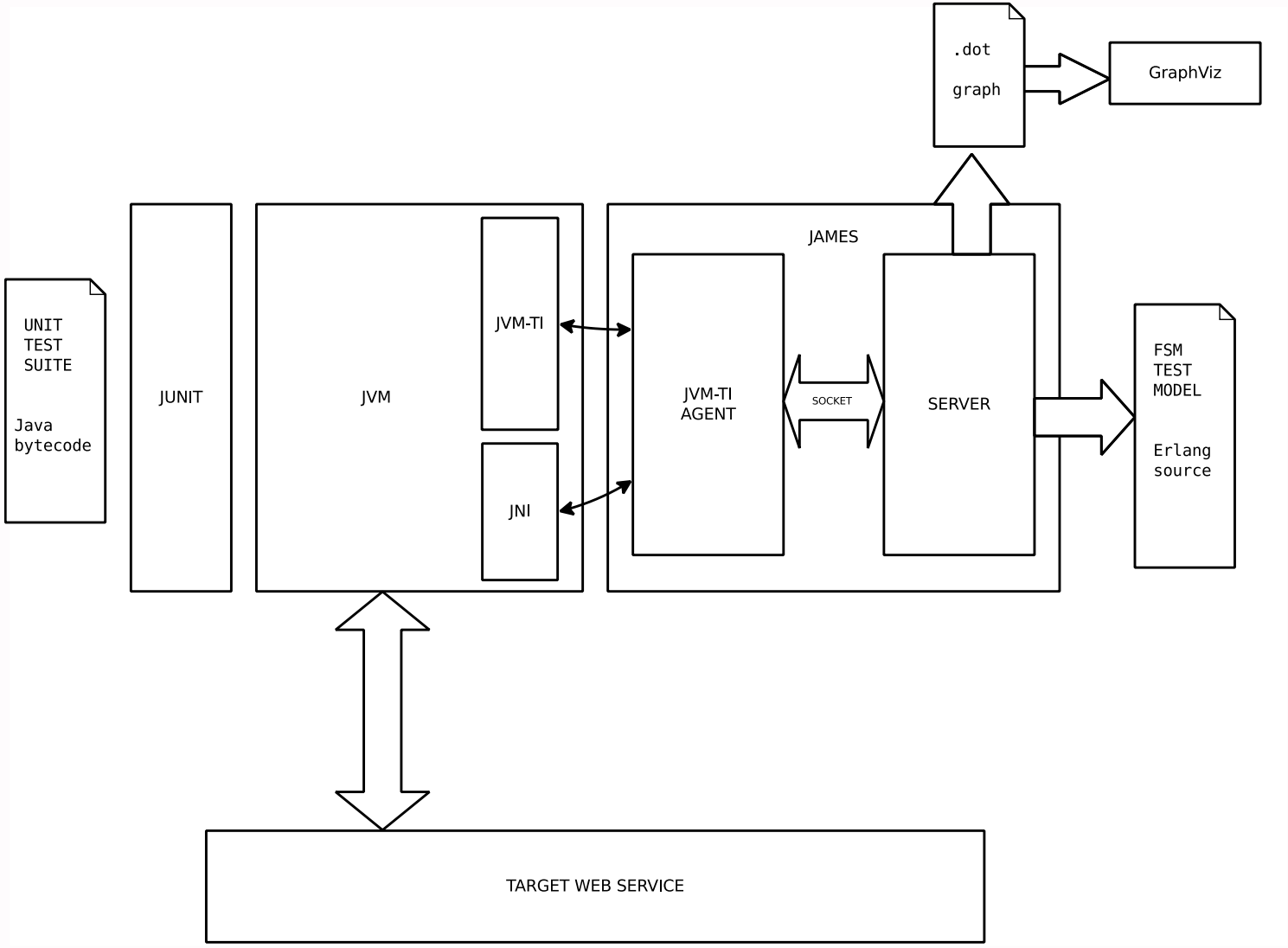
© All rights reserved

James structure



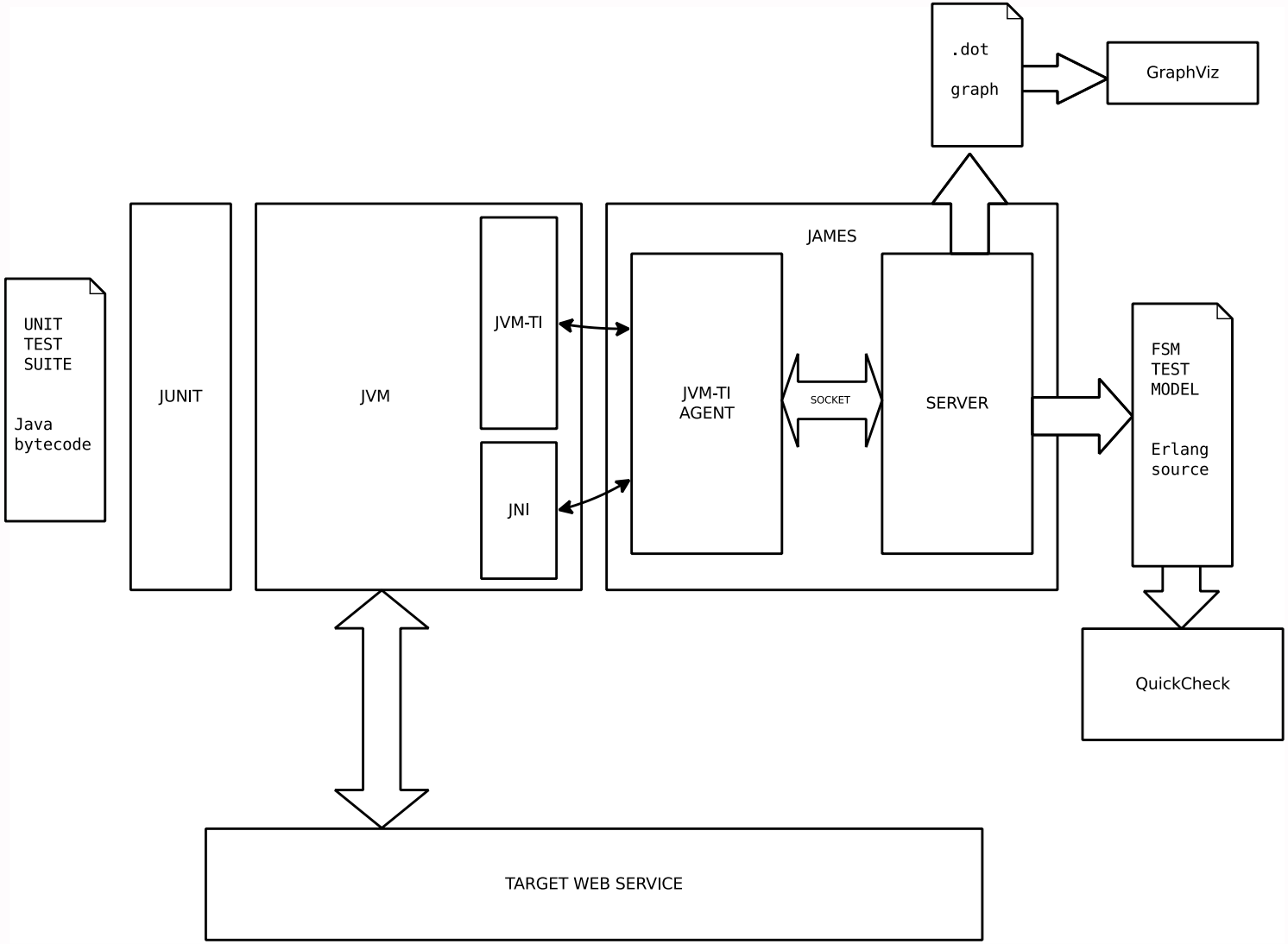
© All rights reserved

James structure



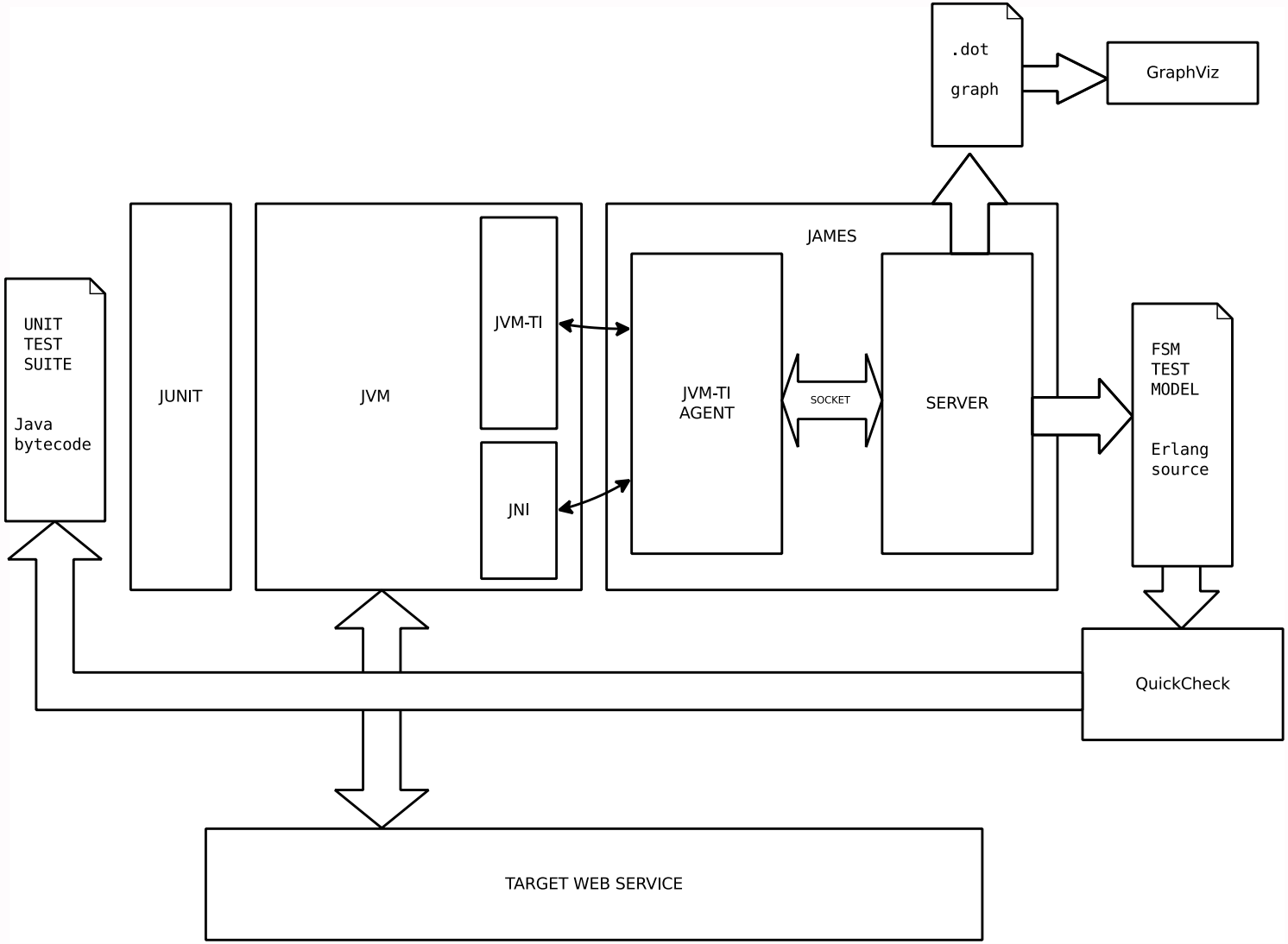
© All rights reserved

James structure



© All rights reserved

James structure



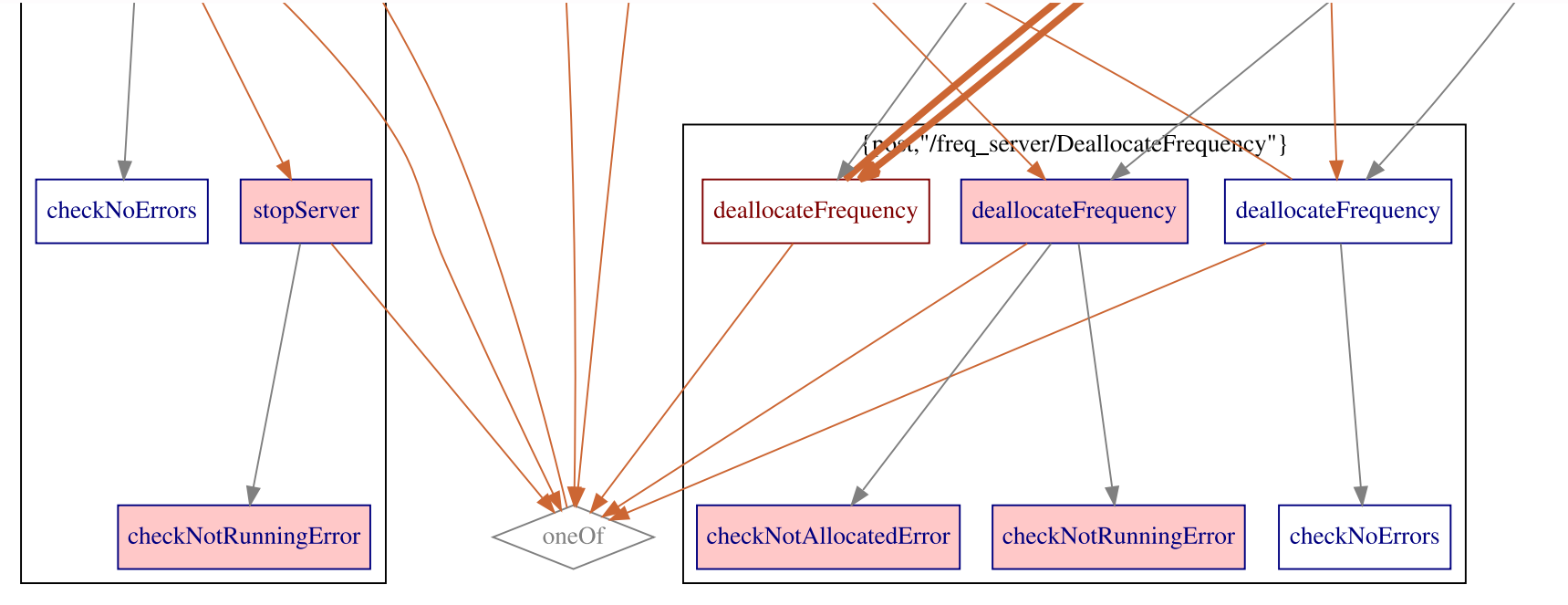
© All rights reserved

Generated diagrams

© All rights reserved

Generated diagrams

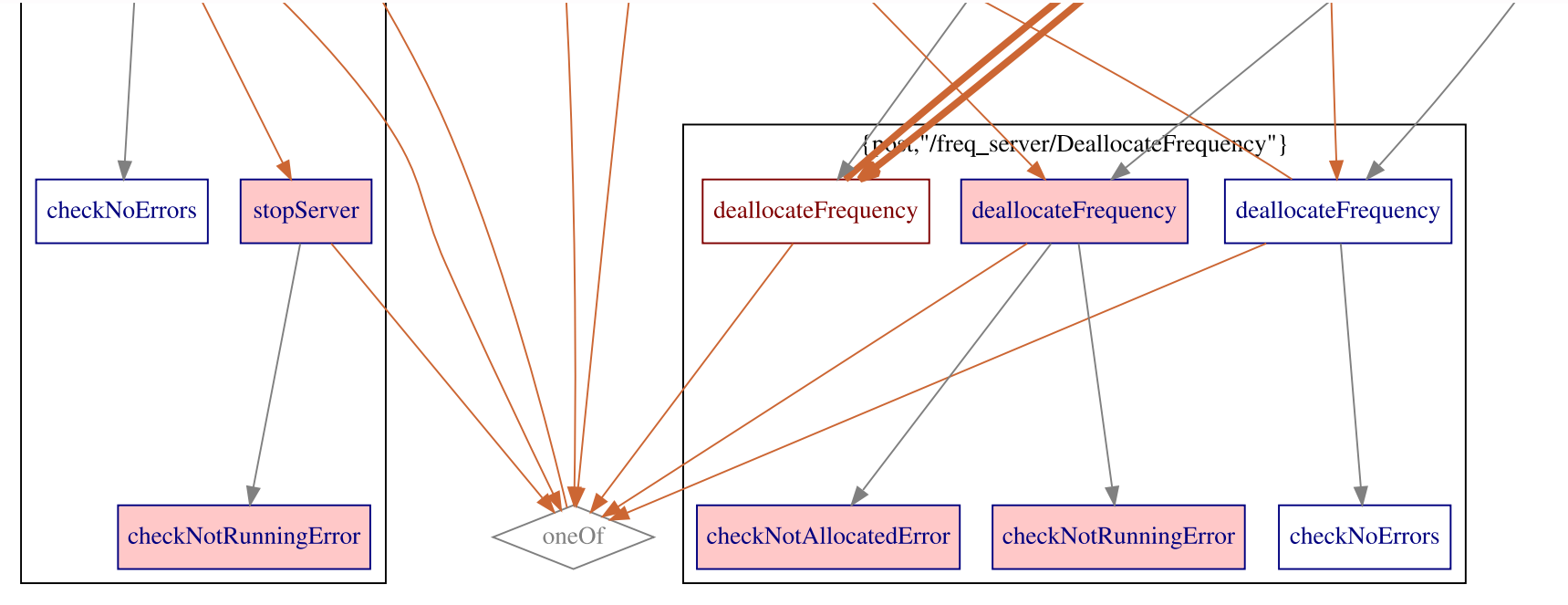
- Rectangular nodes represent:
  - Method calls



© All rights reserved

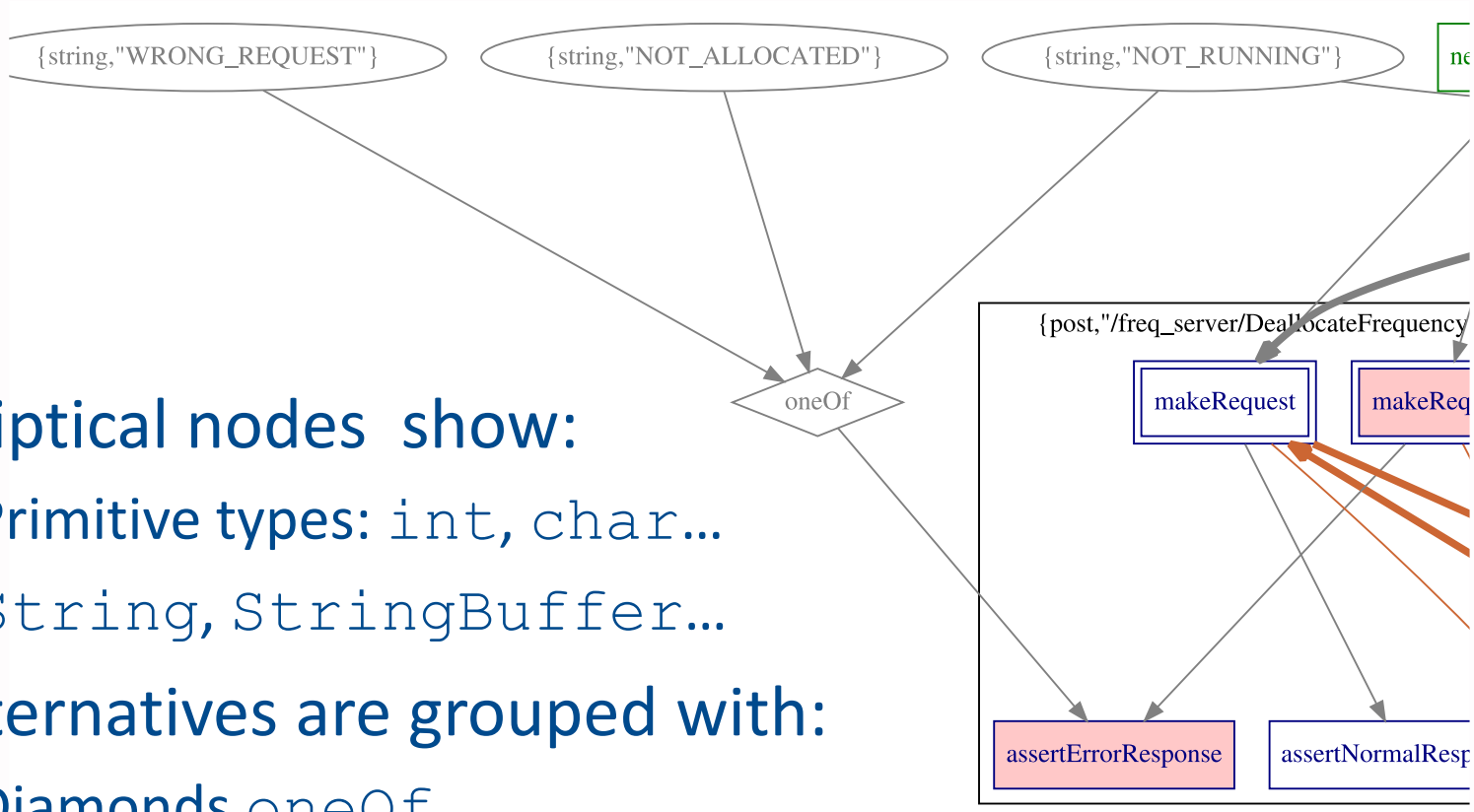
Generated diagrams

- Arrows:
  - Data dependencies (gray arrows)
  - Control dependencies (**brown** arrows)



© All rights reserved

Generated diagrams

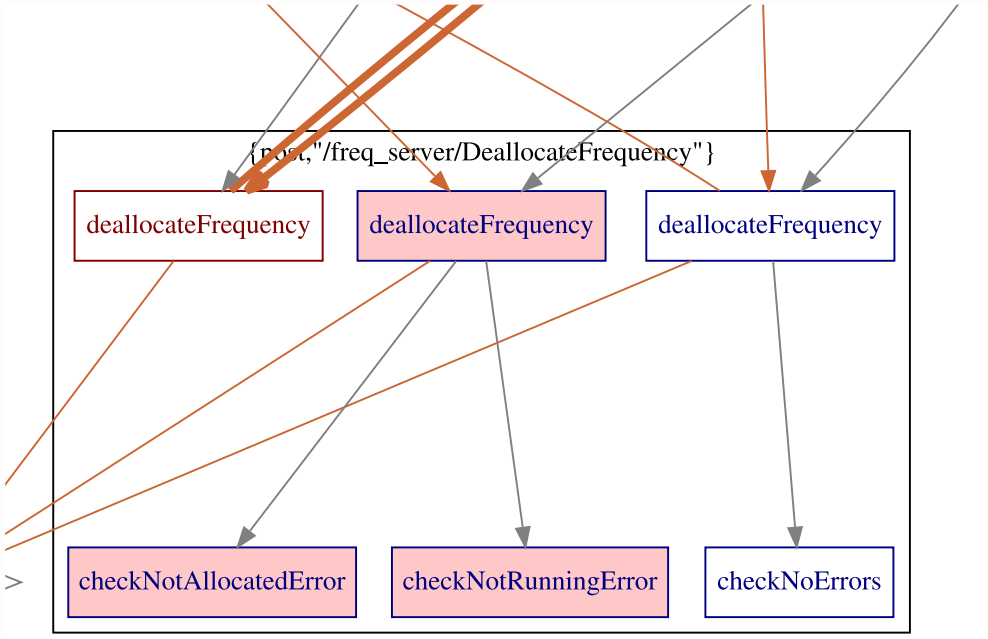


- Elliptical nodes show:
  - Primitive types: `int`, `char`…
  - `String`, `StringBuffer`…
- Alternatives are grouped with:
  - Diamonds `oneOf`

(Later translated to QuickCheck's `oneof` generator)

© All rights reserved

Generated diagrams



- Nodes grouped by
  - HTTP requests
  - With dependencies
- Classified in:
  - Positive
  - Negative (pink bg)

© All rights reserved

Generated diagrams

- Several other simplifications
  - Similar paths are merged
    - But not positive and negative nodes
- Consecutive numbers are generalized
- Constant strings are concatenated

© All rights reserved

EXAMPLE

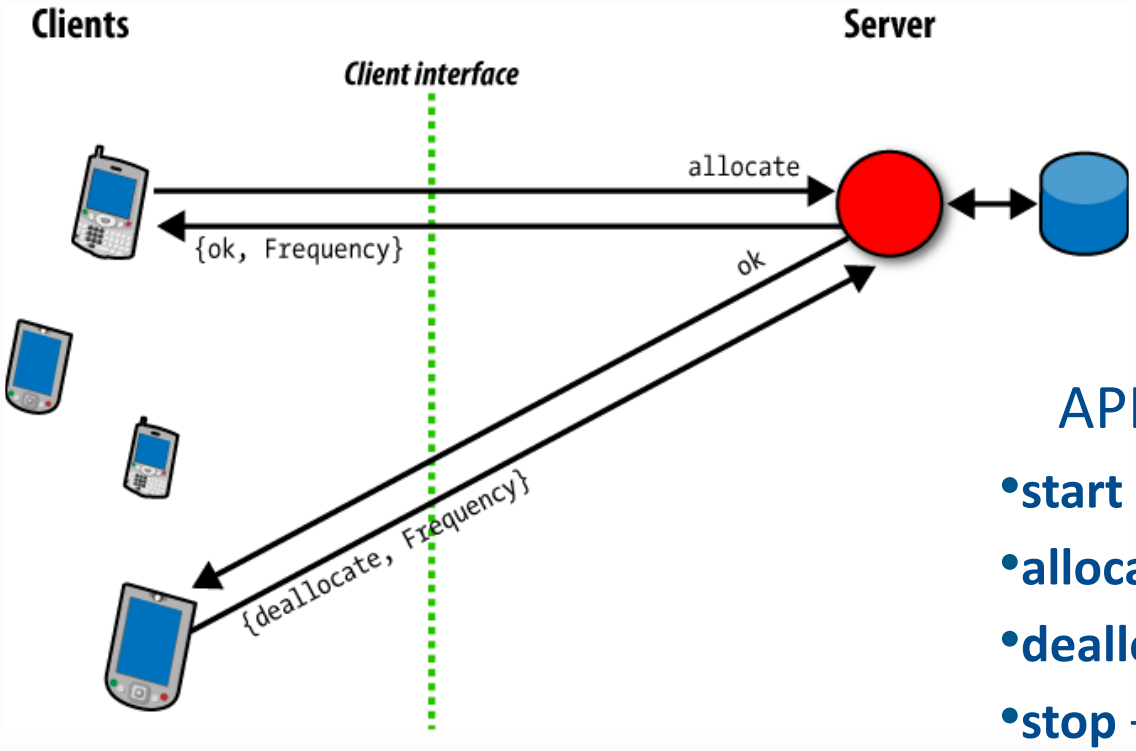
Applying James to a simple system

© All rights reserved

CONTENT SLIDE TITLE

- Example extracted from book:

  F. Cesarini and S. Thompson – *Erlang programming*

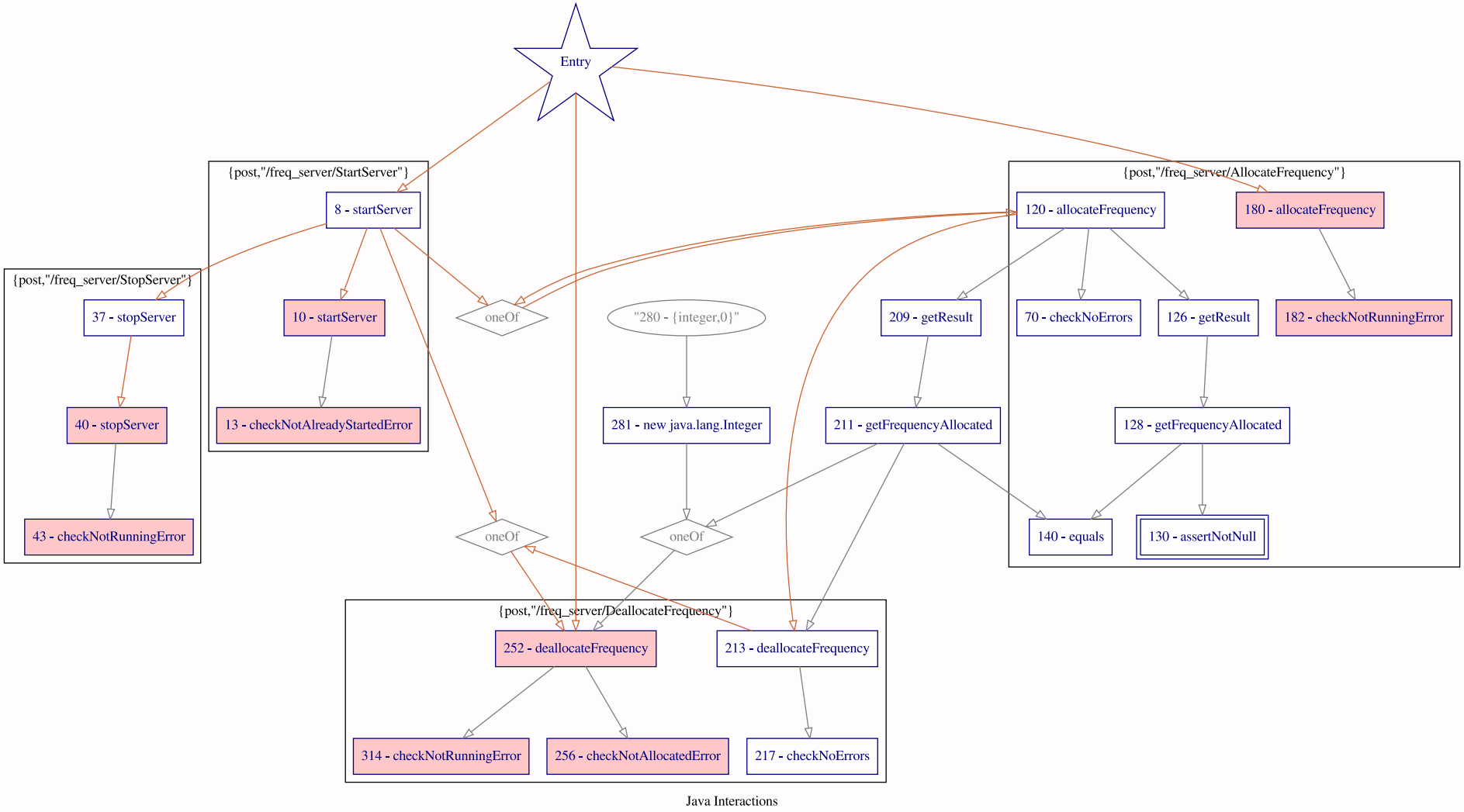


API:

- **start** – Starts the server
- **allocate** – Reserves a frequency
- **deallocate** – Frees a frequency
- **stop** – Stops the server, frees all

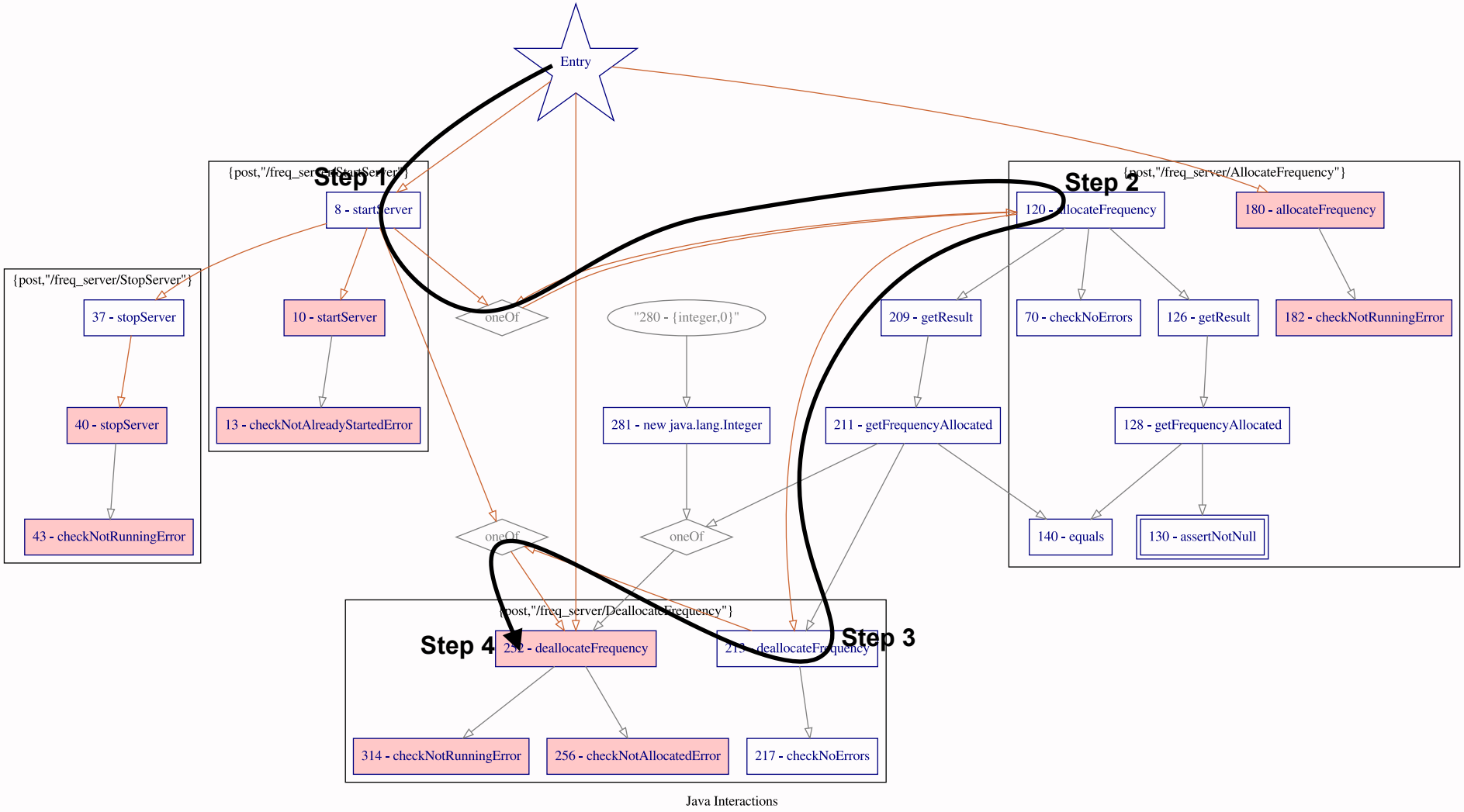
© All rights reserved

Example diagram



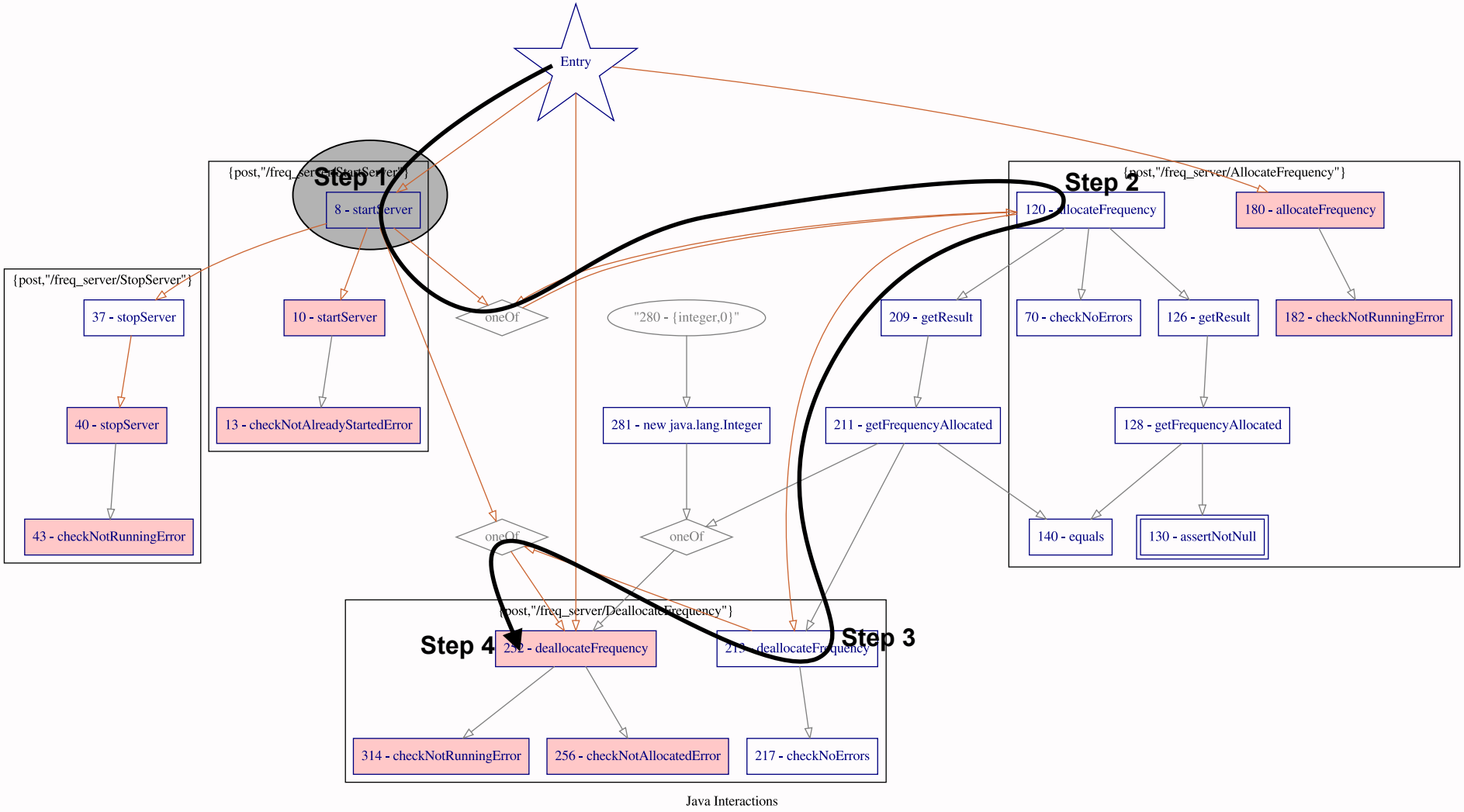
© All rights reserved

Example diagram



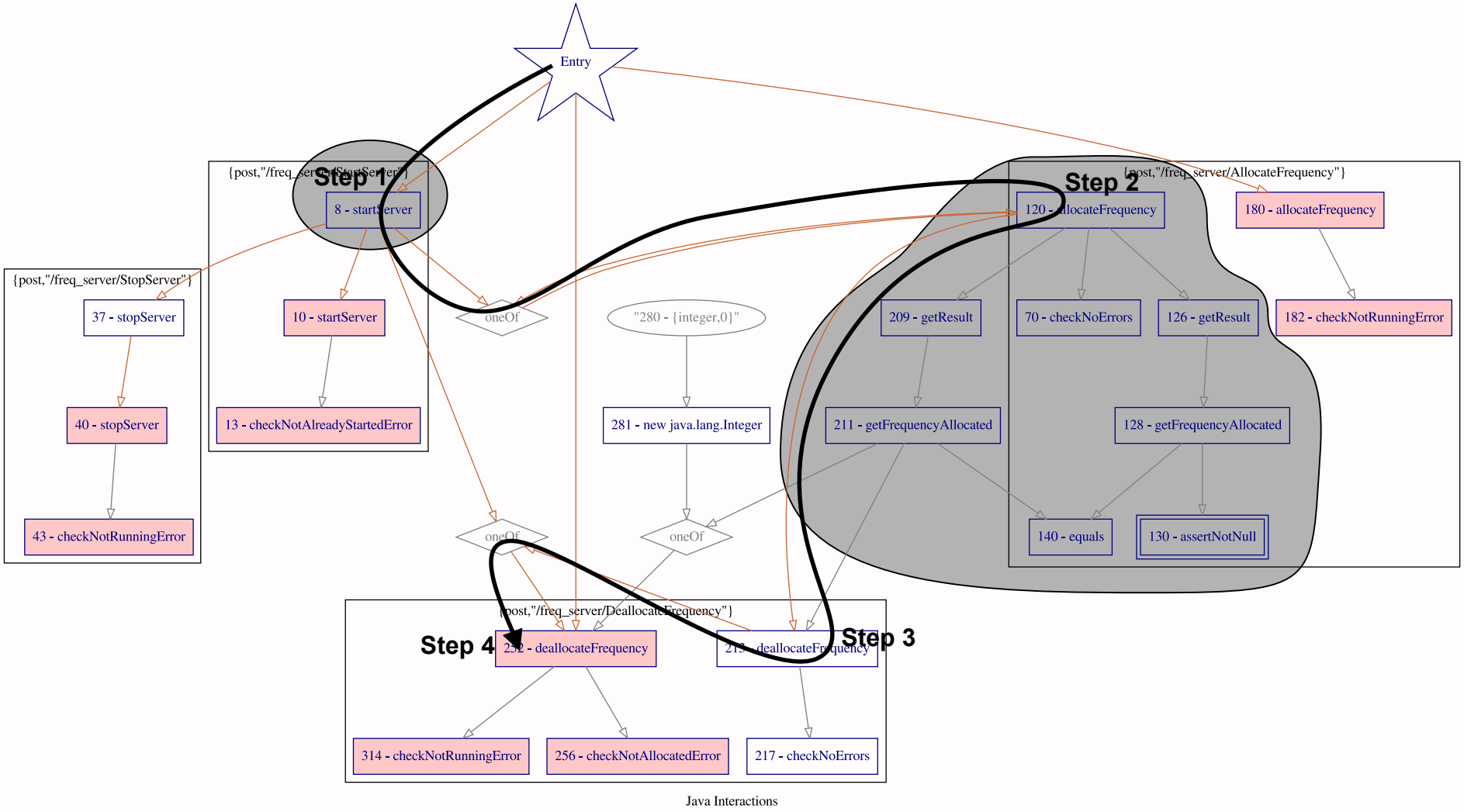
© All rights reserved

Example diagram



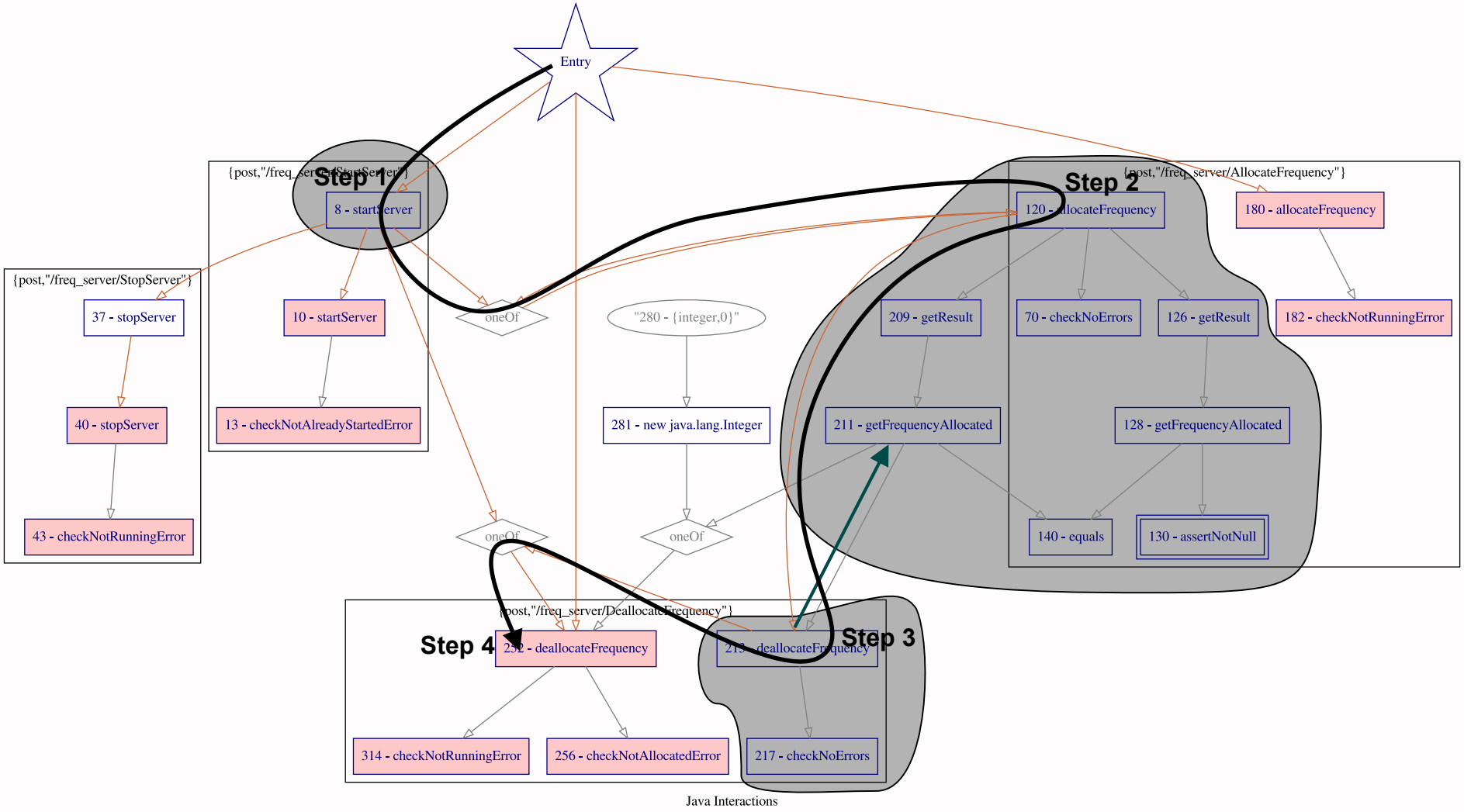
© All rights reserved

Example diagram



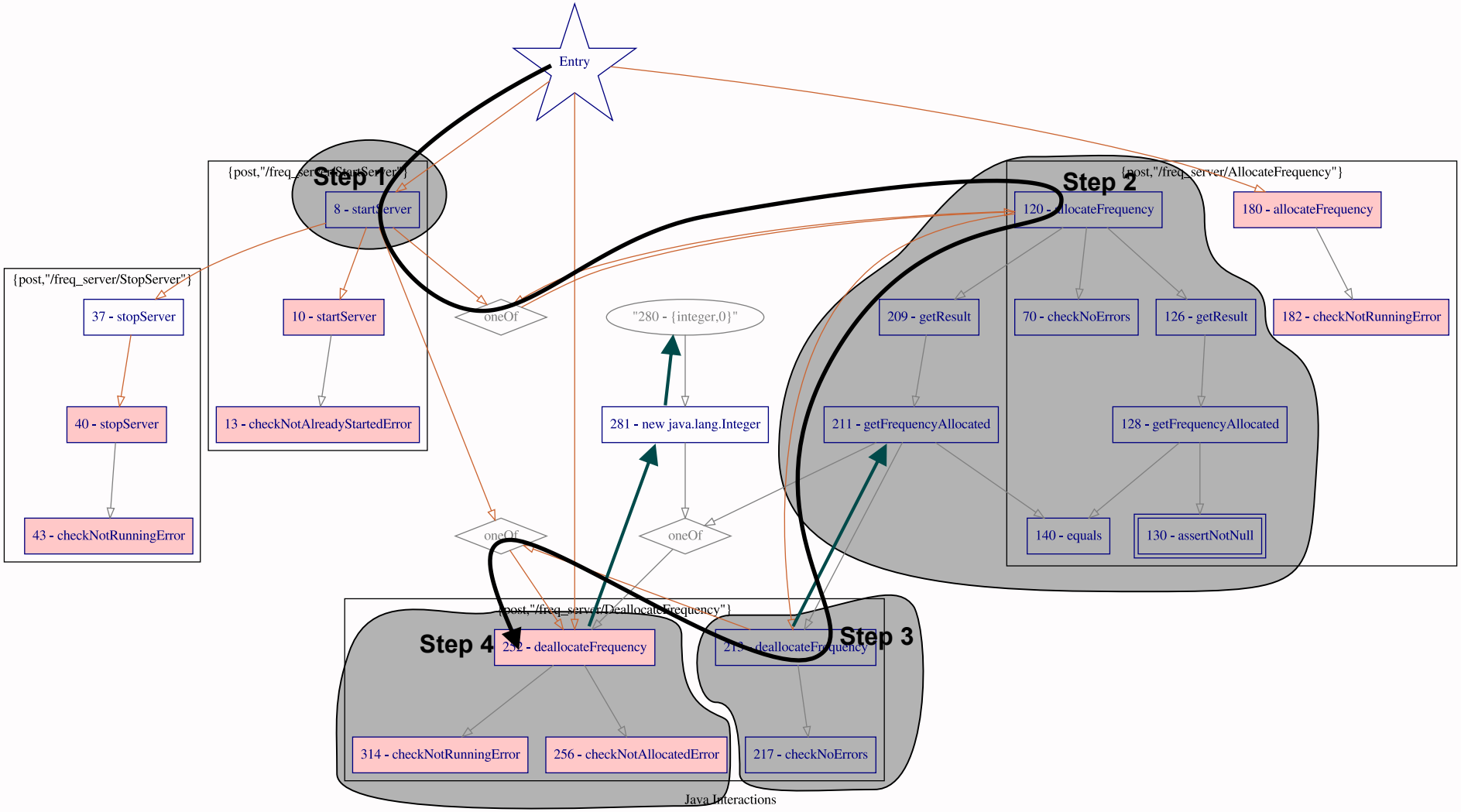
© All rights reserved

Example diagram



© All rights reserved

Example diagram



© All rights reserved

Example generated test

Step 1
```
com.interoud.freqserver.test.parser.FreqServerResponse var1 = this.startServer();
```

Step 2
```
com.interoud.freqserver.test.parser.FreqServerResponse var2 = this.allocateFrequency();
// Postcondition: 1
this.checkNoErrors(var2);
// Postcondition: 2
com.interoud.freqserver.test.parser.Result var4 = var2.getResult();
java.lang.Integer var5 = var4.getFrequencyAllocated();
com.interoud.freqserver.test.parser.Result var6 = var2.getResult();
java.lang.Integer var7 = var6.getFrequencyAllocated();
boolean var8 = var5.equals(var7);
// Postcondition: 3
com.interoud.freqserver.test.parser.Result var9 = var2.getResult();
java.lang.Integer var10 = var9.getFrequencyAllocated();
junit.framework.Assert.assertNotNull(var10);
// End of postconditions
```

Step 3
```
java.lang.Integer var12 = var6.getFrequencyAllocated();
com.interoud.freqserver.test.parser.FreqServerResponse var13 = this.deallocateFrequency(var12);
// Postcondition: 1
this.checkNoErrors(var13);
// End of postconditions
```

Step 4
```
int var15 = 0;
java.lang.Integer var16 = new java.lang.Integer(var15);
com.interoud.freqserver.test.parser.FreqServerResponse var17 = this.deallocateFrequency(var16);
// Postcondition: 1
this.checkNotAllocatedError(var17);
// Postcondition: 2
this.checkNotRunningError(var17);
// End of postconditions
```


© All rights reserved

Example generated test


```
Step 1  com.interoud.freqserver.test.parser.FreqServerResponse var1 = this.startServer();
Step 2  com.interoud.freqserver.test.parser.FreqServerResponse var2 = this.allocateFrequency();
        // Postcondition: 1
        this.checkNoErrors(var2);
        // Postcondition: 2
        com.interoud.freqserver.test.parser.Result var4 = var2.getResult();
        java.lang.Integer var5 = var4.getFrequencyAllocated();
        com.interoud.freqserver.test.parser.Result var6 = var2.getResult();
        java.lang.Integer var7 = var6.getFrequencyAllocated();
        boolean var8 = var5.equals(var7);
        // Postcondition: 3
        com.interoud.freqserver.test.parser.Result var9 = var2.getResult();
        java.lang.Integer var10 = var9.getFrequencyAllocated();
        junit.framework.Assert.assertNotNull(var10);
        // End of postconditions
        java.lang.Integer var12 = var6.getFrequencyAllocated();
Step 3  com.interoud.freqserver.test.parser.FreqServerResponse var13 = this.deallocateFrequency(var12);
        // Postcondition: 1
        this.checkNoErrors(var13);
        // End of postconditions
        int var15 = 0;
        java.lang.Integer var16 = new java.lang.Integer(var15);
Step 4  com.interoud.freqserver.test.parser.FreqServerResponse var17 = this.deallocateFrequency(var16);
        // Postcondition: 1
        this.checkNotAllocatedError(var17);
        // Postcondition: 2
        this.checkNotRunningError(var17);
        // End of postconditions
```


© All rights reserved

Example generated test


```
Step 1  com.interoud.freqserver.test.parser.FreqServerResponse var1 = this.startServer();
Step 2  com.interoud.freqserver.test.parser.FreqServerResponse var2 = this.allocateFrequency();
        // Postcondition: 1
        this.checkNoErrors(var2);
        // Postcondition: 2
        com.interoud.freqserver.test.parser.Result var4 = var2.getResult();
        java.lang.Integer var5 = var4.getFrequencyAllocated();
        com.interoud.freqserver.test.parser.Result var6 = var2.getResult();
        java.lang.Integer var7 = var6.getFrequencyAllocated();
        boolean var8 = var5.equals(var7);
        // Postcondition: 3
        com.interoud.freqserver.test.parser.Result var9 = var2.getResult();
        java.lang.Integer var10 = var9.getFrequencyAllocated();
        junit.framework.Assert.assertNotNull(var10);
        // End of postconditions
        java.lang.Integer var12 = var6.getFrequencyAllocated();
Step 3  com.interoud.freqserver.test.parser.FreqServerResponse var13 = this.deallocateFrequency(var12);
        // Postcondition: 1
        this.checkNoErrors(var13);
        // End of postconditions
        int var15 = 0;
        java.lang.Integer var16 = new java.lang.Integer(var15);
Step 4  com.interoud.freqserver.test.parser.FreqServerResponse var17 = this.deallocateFrequency(var16);
        // Postcondition: 1
        this.checkNotAllocatedError(var17);
        // Postcondition: 2
        this.checkNotRunningError(var17);
        // End of postconditions
```


© All rights reserved

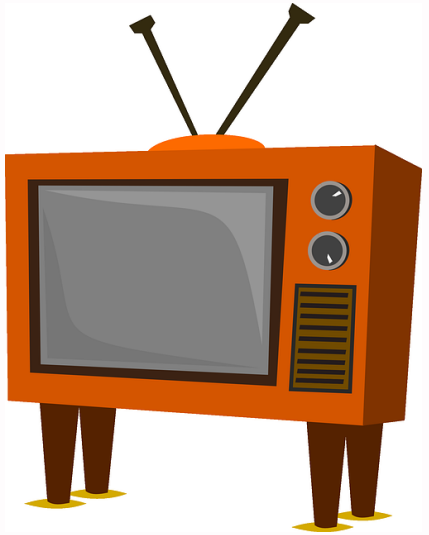
PILOT STUDY

Results and lessons learned

© All rights reserved

Pilot study in Interoud

- On an industrial Web Service: VoDKaTV
  - Video on Demand



- 20 operations related to:
  - Rooms and device management
  - Search by criteria

- Example operations

  CreateRoom, FindAllRoom, CreateDevice, FindDeviceById, FindDevicesByRoom, UpdateDevice, DeleteDevice, etc.

© All rights reserved

Pilot study in Interoud

- Evaluated possible threats to validity:
  - How many tests are needed?

    A single test-case already produces new results
  - Are requirements realistic?

    It requires some manual work, but yes
  - Are new tests useful?

    They helped find an unknown bug!
  - Are new tests meaningful?

    Not so much...

© All rights reserved

Pilot study in Interoud

- Implicit objects

```
id = createRoom(name, location);
checkRoomExists(id, name, location);
```

- Side effects for client objects
- Classify assertions further
- Primitive values are difficult to track

© All rights reserved

Questions?

James tool: https://github.com/palas/james

PROWESS: http://prowessproject.eu/

Presenter: Pablo Lamela Seijas

Email: P.Lamela-Seijas@kent.ac.uk

© All rights reserved