Tutorial
Methodologies for Test-Driven Development of OSGi enabled Embedded Devices
"How Fit is Your Device?"
About the Tutorial

- The tutorial includes an overview, some real life examples, demos and hands-on exercises

- Requirements
  - Notebook
  - Java ME
  - Eclipse 3.3
  - Eclipse FitNesse Plug-In

- We provide the contents of the tutorial on CD-ROM and USB stick
  - Slides
  - PDF-Article
  - eFitNesse software
About us – MicroDoc

- Located in Munich – Germany
- Test Driven Process Model using Java and OSGi
- Cross industry expertise
  - Financial
  - Airline
  - Automation
  - Automotive
  - Logistics
  - IT
- First european business partner for IBM’s embedded Java platform
  - Customized platform ports
  - Distribution and licensing for the IBM embedded product range.
- Active member of the
  - Eclipse Foundation
  - Open Source Business Foundation

www.microdoc.com
About our partners and projects in the embedded space

• IBM Business Partner
• AMD
• SKIDATA AG
• Daimler Fleetboard
• Lufthansa
• Trolltech
• Emlix
• Feig Electronic
• Banksys
• Gumstix
Challenges in Embedded Development

- Development environment versus runtime environment
- Device unit costs versus software development costs
- High end user expectations
- Bug fixing
  - much more complicated and expensive
  - no or minimal access to customer devices
- Requirements
  - High stability and quality
  - Advanced requirements for recovery strategies
  - Remote update possibilities
  - Remote analyzing features
Our experiences in real-life projects

- Testing and Features
  - Exponential relation if testing is done manually
  - A lot of tests are performed manually on the embedded device.
  - Customers complain about time-consuming quality assurance
  - Customers tend to reduce full test coverage
  - Software quality suffers

![Graph showing exponential relation between number of features and time needed for testing.](image-url)
Another experience in a real-life project

- **Legacy Systems**
  - Customer with a long-running embedded system in the area of access control systems
  - Grown legacy software with all known problems:
    - Hard to maintain/extend
    - No testing possibilities besides manual testing
    - Each change leads to a quality risk

- **Customer Decision**
  - New software architecture → open source, OSGi
  - Focus on → changeability, maintainability, extendibility, testability
  - Introduction of → Agile methods, test-driven development
Need for a testing framework

- Ideally usable for all testing layers
- Tests deployable and runnable on the target device
- Communication basis with the customer
  - Requirement specification (acceptance tests)
  - Bug reporting
  - Better understanding of business use cases
- Test automation support
  - Less manual test effort
  - Continuous testing
- Tests as application documentation
- Reduce time to market
Our Solution

- **Story Tests / Acceptance Tests**
  - Encourage your customer to define expectations and requirements as story tests
  - Clear communication through concrete examples
  - Business rules as executable tests
  - Implement story-driven / test-driven
  - Acceptance procedure = test verification

- **Continuous Testing**
  - Reduce Risk / Improve Software Quality
  - Test from a business perspective
  - Integration in a Continuous Build process
  - Test on the target platform / on the embedded device
FIT/FitNesse

- FIT - Framework for Integrated Tests
  - captures business rules in a simple table format

- FitNesse
  - Collaborative Wiki for building and executing tests
  - Runs tests by reading HTML files, looks for tables, uses data in the tables to execute tests and compare results to expectations
  - Easy to learn Markup Language

- Business rules fall into two broad categories
  - Rules that calculate something or make decisions
  - Business process or workflow specification on how something gets done and what the outcomes should be
FitNesse – Two Minute Example

**Test Case**
- Test division of calculator application
- Tests are expressed as table of:
  - Input
  - Expected Output
  - Input: numerator, denominator
  - Expected output: quotient
  - $10 : 2 = 5$
FitNesse – Two Minute Example

Test Case

- Wiki Page (Test Table)
- Java Code (Fixture)
FitNesse – Two Minute Example

- **Eclipse Setup**
  - Install BandXI FitNesse Plug-In
  - Create a new Project
  - Create source folder called “src”
    - Add FitNesse to project class path (right click -> FitNesse)
  - Create a wiki folder called “FitNesseRoot”
  - Configure the root directory of the FitNesse plugin
    - point to your “FitNesseRoot”
  - Start FitNesse Plug-In
FitNesse – Two Minute Example

- Wiki page
  - Edit wiki page and enter test table
  - Set page properties to “test”
FitNesse – Two Minute Example

| examples.Division |
| numerator | denominator | quotient() |
| 10 | 2 | 5 |
| 10 | 1 | 10 |

```java
package examples;
import com.microdoc.efitnesse.fit.ColumnFixture;

public class Division extends ColumnFixture {

    public double numerator, denominator;

    public double quotient() {
        return numerator/denominator;
    }
}
```
FitNesse – Two Minute Example

**ExampleTests**

**BasicTests**

**TEST RESULTS**

**Assertions:** 13 right, 0 wrong, 0 ignored, 0 exceptions

<table>
<thead>
<tr>
<th>com.microdoc.fitnesse.fit.import</th>
<th>examples</th>
</tr>
</thead>
</table>

**BASIC FITNESSE TESTS**

**COLUMN FIXTURE**

<table>
<thead>
<tr>
<th>Division</th>
<th>numerator</th>
<th>denominator</th>
<th>quotient()</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>2</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>13</td>
<td></td>
</tr>
</tbody>
</table>
FitNesse – Wiki Pages

- Wikis, SubWikis
  - Page Properties, Refactoring Pages
  - PageHeader, PageFooter
  - SetUp, TearDown

- Useful built-in support
  - Index of contained subwiki pages: !contents
  - Include other pages: !include
  - Collapsible sections: !*****>
  - Variables: !define
  - Keyword support: null, blank, error
FitNesse – Basic Fixture Types

- **ColumnFixture**
  - Rows of data represent inputs and expected outputs
  - `execute()`, `reset()`

- **RowFixture**
  - Check the exact set of result objects of a query
  - `missing / surplus-marker`

- **ActionFixture**
  - Useful for emulating a series of events
  - `start()`, `press()`, `check()`

- **Additional support in fixtures**
  - Standard Java data types, Arrays
  - Parameters
  - Comparison expressions in numeric table cells
- Java Micro Edition
- OSGi Applications
- Remote Test Execution
- Remote Debugging

- System Analysis
- JUnit Test Cases
- Timeout behaviour
Architecture – Developer’s View
Architecture – Technical View

- Webserver hosting Wiki
- OSGi container
- HTTP Service
- Testing Framework (Fit)
- Application

- HTTP
- FitServlet
Extended Features – QueryRowFixture

- Extended RowFixture
  - Assertion mode (default behaviour)
  - Query/Analyze mode (argument „query“)

- Fixtures already available for analysis of
  - OSGi bundles
  - OSGi services
  - System properties

<table>
<thead>
<tr>
<th>id()</th>
<th>description()</th>
<th>state()</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>org.eclipse.osgi</td>
<td>ACTIVE</td>
</tr>
<tr>
<td>1</td>
<td>org.eclipse.osgi.services</td>
<td>ACTIVE</td>
</tr>
<tr>
<td>3</td>
<td>org.eclipse.equinox.http.servlet</td>
<td>ACTIVE</td>
</tr>
<tr>
<td>4</td>
<td>javax.servlet</td>
<td>ACTIVE</td>
</tr>
<tr>
<td>5</td>
<td>org.eclipse.equinox.http</td>
<td>ACTIVE</td>
</tr>
<tr>
<td>6</td>
<td>com.microdoc.efitnesse</td>
<td>ACTIVE</td>
</tr>
</tbody>
</table>
Extended Feature – JUnitFixture

- Generic fixture to run JUnit Tests within FitNesse
- Advantage:
  - One user interface for different test levels

```
<table>
<thead>
<tr>
<th>com.microdoc.fitnesse.fitnesse.fixtures.junitJUnitFixture</th>
<th>examples.SimpleTestCase</th>
</tr>
</thead>
<tbody>
<tr>
<td>getTestCase()</td>
<td>getFailures()</td>
</tr>
<tr>
<td></td>
<td>getErrors()</td>
</tr>
<tr>
<td>testSimpleTrue</td>
<td></td>
</tr>
<tr>
<td>testExceptionFailed</td>
<td>testException(examples.SimpleTestCase): My Exception</td>
</tr>
<tr>
<td>testSimpleFalse: failed</td>
<td>testSimpleFalse(examples.SimpleTestCase): expected: /77/ but was: /88/</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
Extended Feature – TimeoutColumnFixture

- Extends ColumnFixture

- Abstract superclass for userdefined fixtures
  - Waits for the expected results within a distinct timeout interval without canceling the test
  - Redos assertion of the results until the assertion is passed or the max timeout is reached
  - The interval between two assertions is configurable

- Useful for
  - Asynchronous operations
  - Slow devices
  - Device specific delays
Demo 1: **Automated Tests**

- **SKIDATA AG**  [www.skidata.com](http://www.skidata.com)

- Access Control Systems
  - Several thousands of access transactions per day

- Methodology was adopted within the customer’s QA department

- Used for automated regression tests
Demo 1: Automated Tests (cont’d)
Demo 2: GUI Testing

- FleetBoard DispoPilot [http://www.fleetboard.com](http://www.fleetboard.com)
- Automatic GUI Testing
  - GUI Fixtures with timeout behaviour
    - Edit
    - Button-Click
    - Hardkey-Click
    - Select
    - Assert Pages, Dialogs
  - Test Recorder Tool
  - GUI Analyze Feature
Demo 3: Hands-On

- Embedded Devices
  - Gumstix device
  - Windows Mobile Smartphone
- Software
  - eFitNesse including runtime, documentation, wiki, and examples
Outlook

- Continuous Build Integration
- Wiki improvements
  - Extended Keywords
  - Embedded HTML, PHP
  - Flash
Useful Links

- Technical Paper – eFitnesse
  - [http://www.microdoc.com/efitnesse](http://www.microdoc.com/efitnesse)
- FitNesse Testing Framework
  - [http://fitnesse.org](http://fitnesse.org)
- Patang Testing Framework
  - [http://patang.sourceforge.net](http://patang.sourceforge.net)
- Eclipse FitNesse Plug-In
- General information about embedded Java and OSGi
- Custom Embedded Java Virtual Machines
Trademarks & Copyrights

- **Java** is a trademark or registered trademark of Sun Microsystems, Inc. in the United States and other countries.

- **IBM**, and **WebSphere** are trademarks or registered trademarks of IBM Corporation in the United States and other countries.

- **OSGi** is a trademark of the OSGi Alliance.

- All other trademarks mentioned are trademarks of the respective owners
Thanks