Jubula Tutorial: Round 3

Alexandra Schladebeck
Achim Lörke
Zeb Ford-Reitz

BREDEX GmbH
Agenda

- Quick introduction to Jubula and short demo
- Exercises: hands-on test specification and execution
- Exercises: automated testing during the project lifecycle
- Further aspects
- Questions
What is Jubula?

- Eclipse Project
- Automated functional testing through the GUI
- Released from core parts of GUI\textit{dancer} commercial test tool
  (Eclipse Award Winner 2010)
- \textit{GUI}dancer = Jubula +
Architecture and definitions

- **Client – server application**
  - **Client**: ITE (Integrated test environment)
  - **Server**: Agent
    - For the tutorial: embedded agent
- **AUT = Application under test**
  - Started by the Agent
- **Tests stored in a database**
  - For the tutorial: embedded database
What makes Jubula different?

- Drag and drop test creation
  No recording
- No program code
- According to best practices known from software development
What can Jubula do?

- User perspective
- Complete workflows
- All layers tested
Constant feedback about acceptance

- Quality info close to development
  - User perspective at forefront
- Safety net
Quick demo: Jubula tests Jubula
Quick demo: how are tests made?

- Step 1: Specify (drag and drop)
- Step 2: Add data
- Step 3: Do Object Mapping
- Step 4: Execute
Version 1

Execute an existing test
Write a simple test
Write a less simple test
Finish off a test and run it
Pre-flight checks

- Firewalls off
- Write access to your home directory
- Default Java version is 1.6.x_xx
Setup

- Installation: setup file
- Start Jubula
- Import Project: EclipseCon2012_1.0.xml
  - Via Test → Import
- Connect to embedded AUT Agent
  - Check decorators
- Move component names view
  - To bottom of perspective

Thumbs up for all ok, hands up for help
Exercise 0: Run existing test

- Project is imported and opened
- Connected to embedded AUT Agent
- Start AUT for your system in version one:
  - E.g. DVDTTool (Win_V1): Win_V1@localhost
- Wait for AUT to start
- Start Test Suite FIRSTTEST
  - Yes to relevant
  - Yes to change perspective
  - Hands off the keyboard!
Exercise 1: Create simple test

- First test consists of actions on the *Application* component
  - → which means no object mapping 😊
- Use Case:
  - Open waiting dialog from Edit menu
  - Wait for dialog
  - Wait for dialog to close
Exercise 2: Create slightly less simple test

- This time with object mapping as well
- Use Case:
  - Open Help → Info
  - Wait for Info dialog
  - Check version number
  - Click OK

Sounds like we need a module!
Testing as a part of the process

- Software sometimes has bugs 😊
- Software sometimes changes 😊
- Misunderstandings can happen 😊

Following exercises look at development of tests and AUT over time
Exercise 3: Finish test case for name uniqueness

- Brand new feature in this version!
  - Category names should be unique in the DVD Tool
- Open Test Case: Check uniqueness of category names
- Look at what we’ve got already
- Add final test actions:
  - Check enablement of OK button
  - Close dialog with cancel
- Do mapping and run FULLTEST
Version 2
Check that previous bug is gone

Make any necessary changes

Write a new test
Exercise 4: Re-run uniqueness test on new version

- Stop Version 1 AUT
- Start Version 2 AUT
- Re-run FULLTEST (to check that the bug is fixed)

- Do any changes need to be made?
Exercise 5: Comment-in deletion test and run

- Brand new feature in version 2!
  - Categories can be deleted
- In the FULLTEST Test Suite, comment in the Delete Single Category Test Case
- Run the test

- Is everything ok?
What do we do about errors?

- Depending on error type, could lead to whole test stopping
- But what about our constant flow of information
  - For other use cases?
- Error within a Use Case in a Test Suite
  - Can we fix it?
    - Yes ➔ Fix and continue: Retry Event Handler
    - No / Don’t know ➔ Leave this use case and start the next one: Return Event Handler
About Event Handlers

- React to errors
- Action error
- Component not found
- Check failed
- Configuration error

- Continue test
- Continue
- Exit
- Return
- Retry
- Break
- Pause
Why do we need Event Handlers?

- What if an error occurs?

Use Case 1

- SetUp
- Workflow
- TearDown

Use Case 2

- SetUp
- Workflow
- TearDown
Why do we need Event Handlers?

Test Execution History

#Steps

5690

expected

executed
Why do we need Event Handlers?

**Test Execution History**

- **Unknown quality**
- **We haven't known about these test steps for a week**
- **Don't know if previous errors got fixed**
- **Looks like they didn't**
Using Event Handlers
Exercise 6: Add return Event Handler

- Create empty Test Case
- Add the Test Case as an Event Handler to the Deletion Test Case
  - Error type: Action error
  - Reentry type: Return

..if an action error occurs in this Test Case, then do nothing, but continue with the next Test Case in the hierarchy – it will be ready for you!
Version 3

Check that previous bug is gone

Check that there are no unwanted side effects
Exercise 7: Run test on new version

- Stop version 2
- Start version 3
- Re-run FULLTEST on latest version
Brief return to Event Handling

- What if the category to delete contains DVDs?
- Check for non-existence of confirmation dialog
  - If it is there, close it and check again
Exercise 7: Continue writing tests

- If we have time (or as homework assignment 😊):
  - Startup
  - Import DVD library
  - Select category (Science Fiction)
  - Select DVD from table
  - Select technics tab
  - Select region
  - Select languages

Examples of actions with content-based data
Further information: From the command line

- Continuous integration
  - Check out sources
  - Build software
  - Prepare test environment
  - Install software (AUT) on test systems
- Run tests – **Test Executor**
- Check results
  - HTML / XML
  - In ITE

```xml
<configuration>
  <project>YourProjectName</project>
  <version>1.0</version>
  <autconfig>YourAUTConfiguration</autconfig>
  <dbschema>YourDatabaseSchemeName</dbschema>
  <dbuser>YourDatabaseUsername</dbuser>
  <dbpw>YourDatabasePassword</dbpw>
  <server>YourAUTAgentHostname</server>
  <port>PortNumberForAUTAgent</port>
  <language>en_US</language>
  <testsuite>TestSuite</testsuite>
  <datadir>DirectoryWhereExternalDataIsStored</datadir>
  <resultdir>DirectoryToStoreTestResultFiles</resultdir>
</configuration>
```
Summary

- Complete workflows tested based on acceptance criteria
  - Even before application available
  - Automated regression testing
- Manual intelligence incorporated
  - Synchronization
  - Reacting to clues in the application
  - Error handling
- Reuse instead of copy
  - Tests made up of similar units
  - Reusing ensures structure & maintainability
  - Central place for any changes
Testing alongside development

- Tests can grow and change with the software
  - Accompany throughout lifecycle
  - Due to principles of good design
- Encourage communication between stakeholders
  - Questions while designing automated tests
  - Questions resulting from test results
- Provide feedback about the state of the software
  - Early acceptance testing
  - Continuous regression testing
Jubula project status

- Current release:
  - BREDEX release: 1.2 Standalone, 6.0 GUldancer
  - Next release: Juno (1.2 features also in Eclipse Jubula)
- Keep updated:
  @EclipseJubula
  ... or give us your card to be added to the newsletter
Further information

Community

- http://www.eclipse.org/jubula
- Forum
- Webinar
- FAQs
- Videos
- Documentation
- Mailing lists

Professional services

- http://www.bredexsw.com
- Proof of Concept
- Appetizer packages
- Training
- Workshops
- Support
- Test reviews
- Best practices
- GUldancer Demo Licenses
- Customized extensions
And finally...

- Special offer on training for Eclipse Con Tutorial participants:
  - Leave your business card and get 10% off any Jubula or GUIdancer training for your team (valid until 30th June 2012)

- Eclipse Testing Days 2012
  - Call for Sponsors
  - Call for Papers

- Tell us how you test with Jubula and get a goodie!
Give Feedback on the Sessions

1. Sign In:  www.eclipsecon.org

2. Select Session Evaluate

3. Vote  
   ![Vote Options]