ApacheCon Europe
Sinsheim, Germany 5th-8th November 2012

Apache OpenOffice Automated Testing

by Liu Zhe
(presented by Herbert Dürr)

Agenda

- Brief Introduction
- Background
- New Solution
- Next Plan

Brief Introduction

- Test automation saves a lot of human effort and provides faster feedback, especially in regression testing
- Test automation is very appropriate for OpenOffice
  - Has a long maintenance life
  - Core function and UI don't change frequently
  - Heavy regression testing requirement
- OpenOffice had built-in test automation for a long time already, but it needed to evolve

Background

- OpenOffice has a lot of testing code
- Testing code can be categorized to 3 levels

Module	Description
testgraphical	Test tool to test documents by it's graphical representation
testautomation	All test scripts for the old VCL Testtool. Nobody maintains it now
smoketestdoc	It's used to generate the test documents required by smoke test
test	Includes reusable code for UNO API test
smoketestoo_native	A small test suite to verify the working of basic functionality
qadevOOo	UNO API test, application wide complex tests
{Module}/qa	UNO API test, module-specific complex tests

Low Level - Unit Testing for Code

- Verify the correctness of functions, methods and interfaces
- White box testing
- Executed during the build process
- Test failure leads to build break

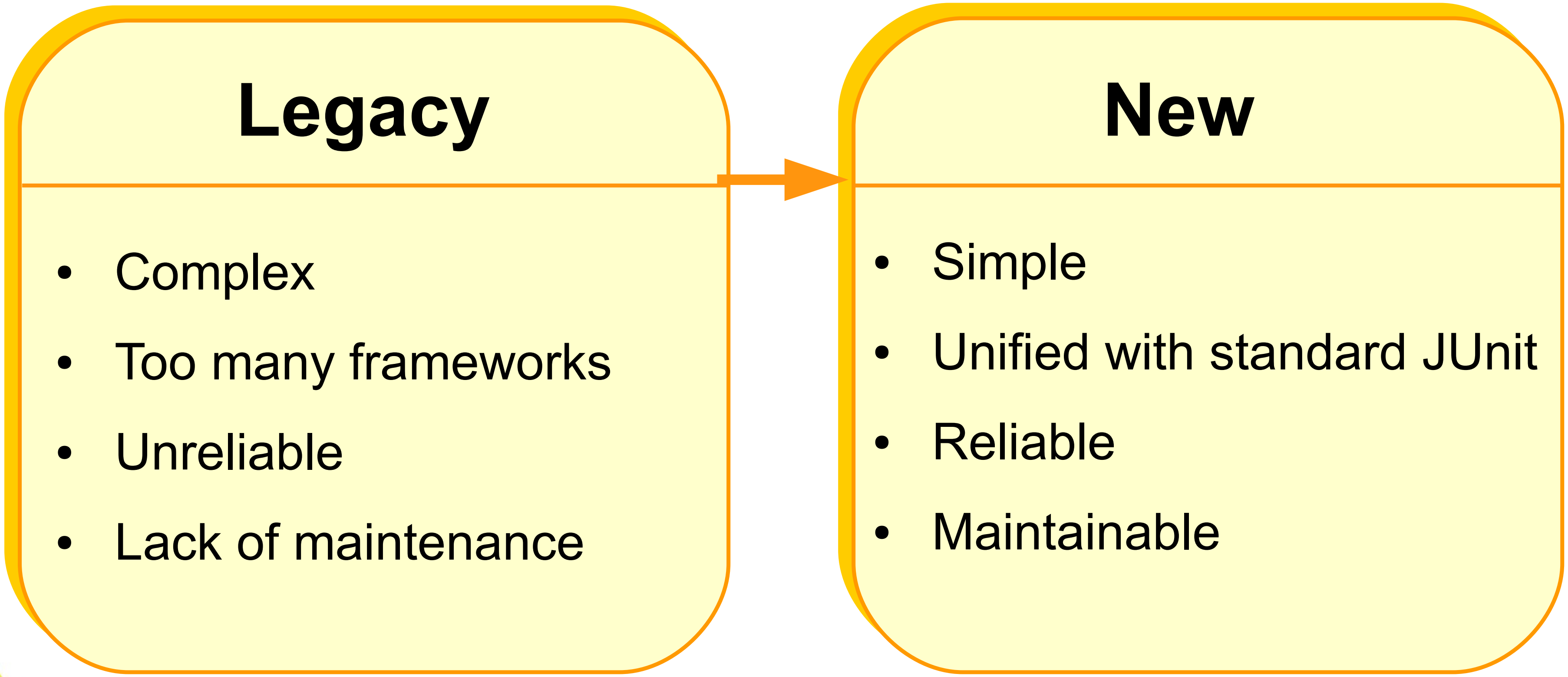
Middle Level - UNO API Testing

- Verify the correct behavior of the product Application Programming Interface (API)
- Gray box testing
- Requires a running OpenOffice instance
- e.g. qadevooo, smoketestoo_native

High Level - GUI Testing

- Simulates a real user to perform testing
- Generates keyboard/mouse events to GUI actions and get information from the GUI to validate the product
- Requires a running OpenOffice instance
- e.g. vcltesttool, testautomation

Evolving Test Framework



Legacy

- Complex
- Too many frameworks
- Unreliable
- Lack of maintenance

New

- Simple
- Unified with standard JUnit
- Reliable
- Maintainable

(Only covering UNO API testing and GUI testing here)

New Testing Framework

- Test code is separated from product code
  - **SVN Repository:** https://svn.apache.org/repos/asf/incubator/ooo/trunk/test/
- Test projects
  - **testcommon:** Reusable code for both GUI test and API test
  - **testuno:** UNO API test cases
  - **testgui:** GUI test cases
- Pure Java projects which can be imported, edited, debugged and run in Eclipse
- All test cases are implemented with JUnit

New Testing Framework

- The testing framework can automatically install builds
  - from the local build environment
  - from the internet, e.g. public buildbots
- Command line to start the tests
- Nice HTML/XML test report
- Uploads test report to testdashboard

Run Test in Build Environment

- Tests can be started easily with the following commands

```
# build OpenOffice
cd main/instsetoo_native
build –all
cd ../../test
# run build verification test (BVT)
ant
# run functional verification test (FVT)
ant "-Dtest.args=-tp fvt"
# run both
ant "-Dtest.args=-tp bvt -tp fvt"
```

Testcase Terminology

- BVT: Build Verification Testing
- FVT: Functional Verification Testing
- PVT: Performance Verification Testing
- SVT: System Verification Testing

Building the Test Code as Standalone Package

- Prerequisites to build

  JDK 1.5 or above.

  Apache Ant 1.8.2 or above.

  Apache OpenOffice 3.4.1 or above.

  JUnit 4.10 or above (automatically downloaded if needed)

- Commands to build

```
cd test
# Generate aoo_test_*.zip to dest.dir
ant "-Ddest.dir={Any Directory}" "-Dopenoffice.home={OpenOffice Directory}" dist
```

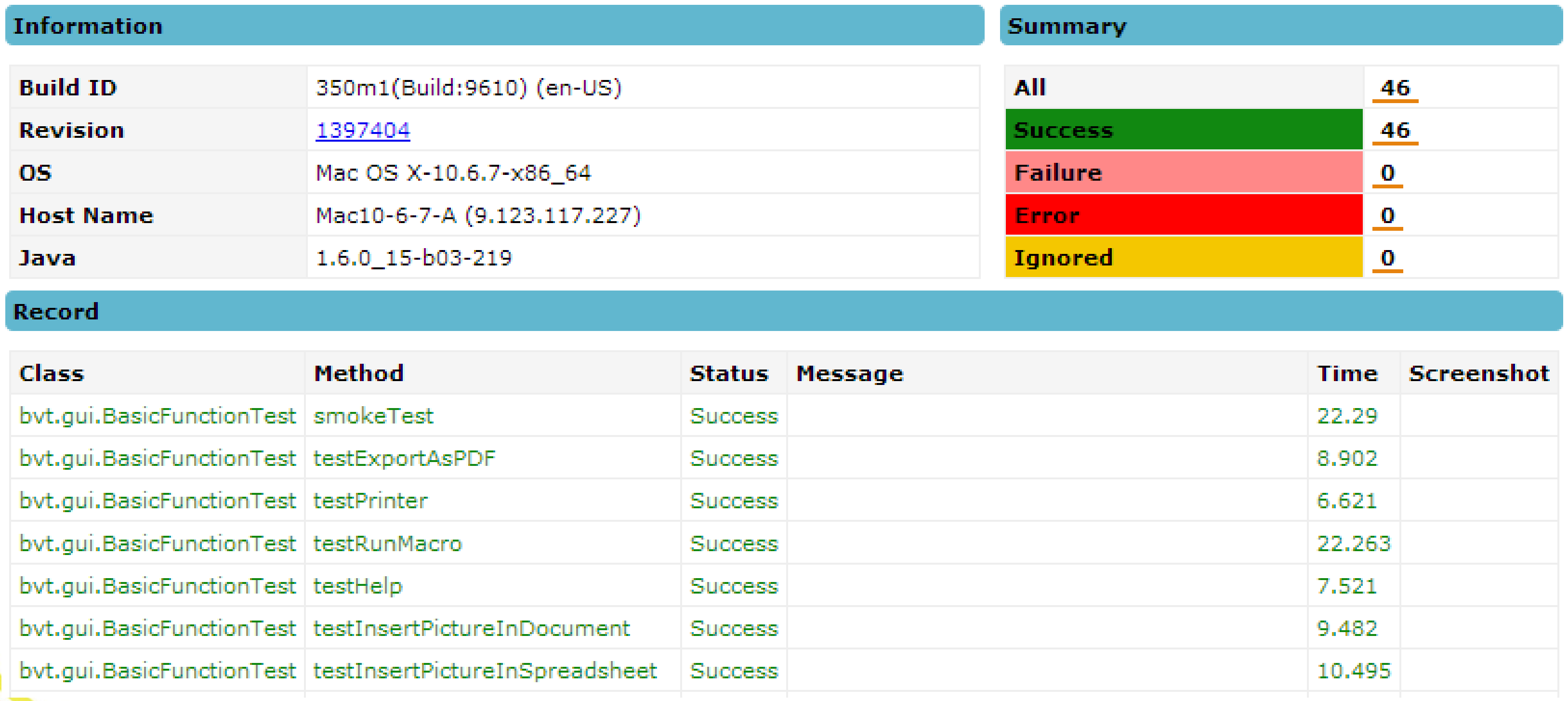
Running the Test Code as Standalone Package

- To run test with standalone package, only JRE 1.5 or above is required
- Extract aoo_test.zip to testing machine, then use the script "run" to start tests

```
cd aoo_test
# Run build verification test on one pre-installed openoffice
./run -Dopenoffice.home="/Applications/OpenOffice.org.app/Contents" -tp bvt
# Automatically download and install a build, then run functional verification test
./run -Dopenoffice.pack=
http://somehost/Apache_OpenOffice_3.5.0_Linux_x86-64_install-arc_en-US.tar.gz -tp fvt
```

Test Result Output

- Test output is stored in "testspace/output.*"
- Open "result.html" in browser to view the test report



Information

Build ID	350m1(Build:9610) (en-US)
Revision	1397404
OS	Mac OS X-10.6.7-x86_64
Host Name	Mac10-6-7-A (9.123.117.227)
Java	1.6.0_15-b03-219

Summary

All	46
Success	46
Failure	0
Error	0
Ignored	0

Record

Class	Method	Status	Message	Time	Screenshot
bvt.gui.BasicFunctionTest	smokeTest	Success		22.29	
bvt.gui.BasicFunctionTest	testExportAsPDF	Success		8.902	
bvt.gui.BasicFunctionTest	testPrinter	Success		6.621	
bvt.gui.BasicFunctionTest	testRunMacro	Success		22.263	
bvt.gui.BasicFunctionTest	testHelp	Success		7.521	
bvt.gui.BasicFunctionTest	testInsertPictureInDocument	Success		9.482	
bvt.gui.BasicFunctionTest	testInsertPictureInSpreadsheet	Success		10.495	

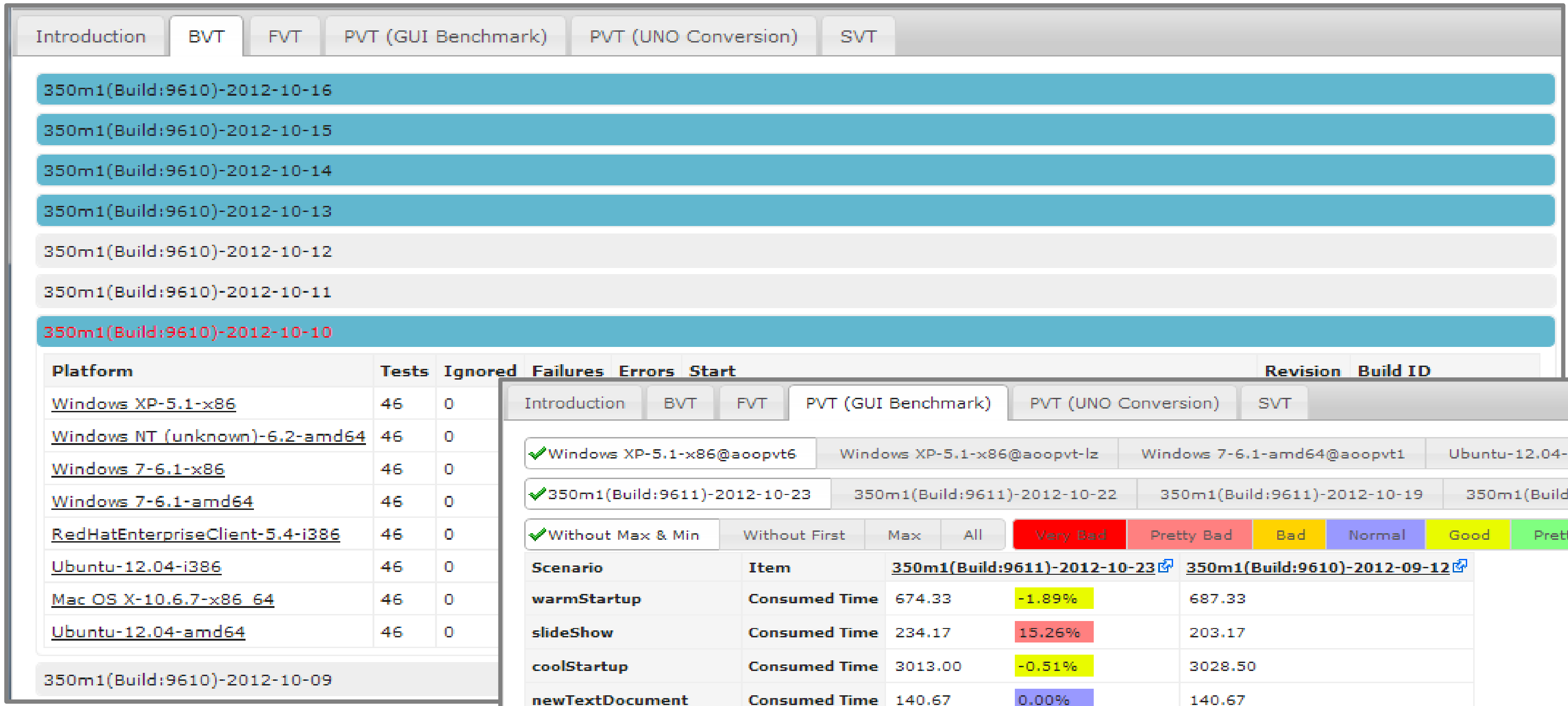
Develop Tests

- It is easy to get the testing framework started in Eclipse with just two steps:
  - Import all projects into Eclipse
  - Set one classpath variable "openoffice.home" to the openoffice directory
- Write, debug and run tests all in Eclipse
- Detail reference
  - http://wiki.openoffice.org/wiki/QA/test_automation_guide

Test Dashboard

- A web app to view and track test result
- Compare performance data between builds
- Anyone can upload test results to it
- Demo address
  - http://people.apache.org/~liuzhe/testdashboard/

Test Dashboard



Introduction | BVT | FVT | PVT (GUI Benchmark) | PVT (UNO Conversion) | SVT

350m1(Build:9610)-2012-10-16

350m1(Build:9610)-2012-10-15

350m1(Build:9610)-2012-10-14

350m1(Build:9610)-2012-10-13

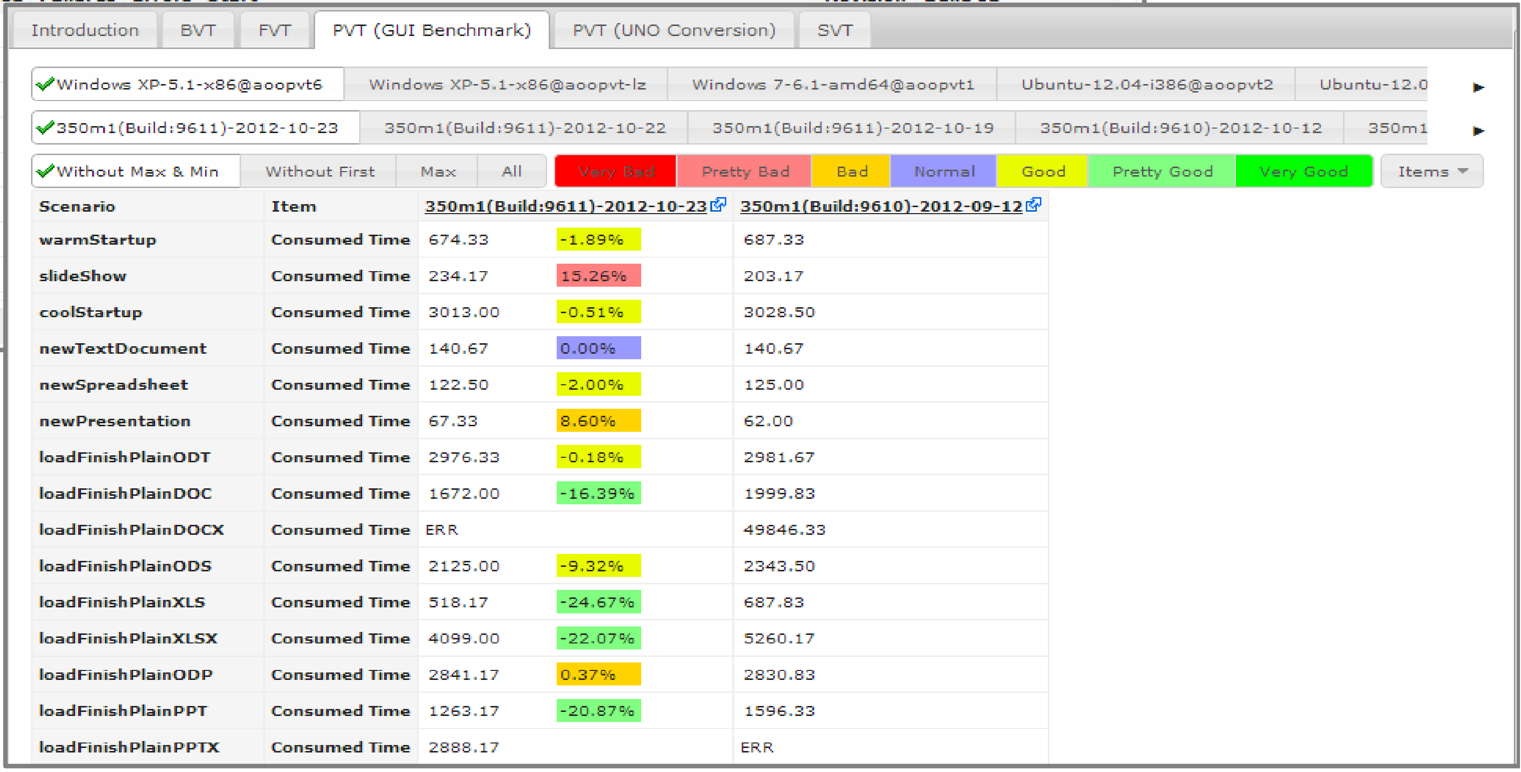
350m1(Build:9610)-2012-10-12

350m1(Build:9610)-2012-10-11

350m1(Build:9610)-2012-10-10

Platform	Tests	Ignored	Failures	Errors	Start	Revision	Build ID
Windows XP-5.1-x86	46	0					
Windows NT (unknown)-6.2-amd64	46	0					
Windows 7-6.1-x86	46	0					
Windows 7-6.1-amd64	46	0					
RedHatEnterpriseClient-5.4-i386	46	0					
Ubuntu-12.04-i386	46	0					
Mac OS X-10.6.7-x86_64	46	0					
Ubuntu-12.04-amd64	46	0					

350m1(Build:9610)-2012-10-09



Introduction | BVT | FVT | PVT (GUI Benchmark) | PVT (UNO Conversion) | SVT

✓Windows XP-5.1-x86@aoopvt6 | Windows XP-5.1-x86@aoopvt-lz | Windows 7-6.1-amd64@aoopvt1 | Ubuntu-12.04-i386@aoopvt2 | Ubuntu-12.0

✓350m1(Build:9611)-2012-10-23 | 350m1(Build:9611)-2012-10-22 | 350m1(Build:9611)-2012-10-19 | 350m1(Build:9610)-2012-10-12 | 350m1

✓Without Max & Min | Without First | Max | All | Very Bad | Pretty Bad | Bad | Normal | Good | Pretty Good | Very Good | Items

Scenario	Item	350m1(Build:9611)-2012-10-23		350m1(Build:9610)-2012-09-12
warmStartup	Consumed Time	674.33	-1.89%	687.33
slideShow	Consumed Time	234.17	15.26%	203.17
coolStartup	Consumed Time	3013.00	-0.51%	3028.50
newTextDocument	Consumed Time	140.67	0.00%	140.67
newSpreadsheet	Consumed Time	122.50	-2.00%	125.00
newPresentation	Consumed Time	67.33	8.60%	62.00
loadFinishPlainODT	Consumed Time	2976.33	-0.18%	2981.67
loadFinishPlainDOC	Consumed Time	1672.00	-16.39%	1999.83
loadFinishPlainDOCX	Consumed Time	ERR		49846.33
loadFinishPlainODS	Consumed Time	2125.00	-9.32%	2343.50
loadFinishPlainXLS	Consumed Time	518.17	-24.67%	687.83
loadFinishPlainXLSX	Consumed Time	4099.00	-22.07%	5260.17
loadFinishPlainODP	Consumed Time	2841.17	0.37%	2830.83
loadFinishPlainPPT	Consumed Time	1263.17	-20.87%	1596.33
loadFinishPlainPPTX	Consumed Time	2888.17		ERR

Future

- Clean up old test codes
- Improve the test dashboard
  - Add SVT result
- Improve test stability
- Promote the new framework in community