State of Automated Testing

- Michael Stahl, Red Hat, Inc.
- 2015-09-23
Overview

1. Introduction

2. Regression Tests Developers Run Every Day

3. Even More Tests
Introduction

- Every release: ~10,000 commits, ~1 million LOC changed
- Regressions?
- Automated testing
- Goal: developers find bugs before they push them to master

"Your developers—or worst case, test organization—produce tests."
— Michael Klepikov
Goal: developers find bugs before they push them to master

Requirements

- Standard unit test libraries
- Must be run as part of standard build – make check
- Reliable – avoid false positives
- Fast
- Good defect localization
- Debuggable
CppUnit Tests

- C++
- Standard CppUnit library (Thanks to Markus Mohrhard)
- In-process (except smoketest)
- Kinds:
  - Unit tests
    - C++ classes
    - Components via UNO API
  - Integration tests
    - Filter crash-tests (CVE-tests)
    - Filter tests
    - Etc.
- System tests: smoketest
Filter Tests

- Filter test:
  - Import file
  - Check some properties were imported correctly
  - Export file, import again
  - Check properties were round-tripped correctly
  - **--with-export-validation** validates every exported file
  - ODF Validator
  - OOXML Validator
CppUnit Test Growth

![CppUnit Tests chart]

LO version

- CPPUNIT_TEST
- CPPUNIT_ASSERT
- CppunitTest*.mk

LibreOffice Aarhus 2015: State of Automated Testing
JUnit Tests

- Java
- Standard JUnit library
- Kinds:
  - Unit tests – in-process
    - URE Java binding
  - “Complex” tests – out-of-process – Remote UNO
    - Unit tests of components
    - Integration tests
JUnit Test Growth

![JUnit Tests](chart.png)

- JUnit @Test
- assert
- JUnitTest*.mk

LO version

- 3.3
- 3.4
- 3.5
- 3.6
- 4.0
- 4.1
- 4.2
- 4.3
- 4.4
- 5.0
- (5.1)
qadevOOo “unoapi” Tests

- Java
- Custom test framework
- Out-of-process – Remote UNO
- Obscure test code
- “black-box”, overly generic tests

Kinds
- Unit tests – components via UNO API
qadevOOo Test Non-Growth

qadevOOo UnoApi Tests

- unoapi components
- unoapi interfaces
- unoapi*.mk

LO version:
- 3.5
- 3.6
- 4.0
- 4.1
- 4.2
- 4.3
- 4.4
- 5.0
- (5.1)
Python Tests

- Python
- Standard unittest library
- In-process
- Thanks to David Ostrovsky
- Kinds:
  - Unit tests – PyUNO binding
  - Unit tests – components via UNO API
Tests Run by make check

"make check"

LO version

CPPUNIT_TEST
JUnit @Test
unoapi interfaces
Python test*

LO version

3.5 3.6 4.0 4.1 4.2 4.3 4.4 5.0 (5.1)
<table>
<thead>
<tr>
<th>Requirements Checklist</th>
<th>Std. libs</th>
<th>make check</th>
<th>Reliable</th>
<th>Fast</th>
<th>Defect localization</th>
<th>Debug-able</th>
</tr>
</thead>
<tbody>
<tr>
<td>AutoTest / testtool</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>?</td>
</tr>
<tr>
<td>CppUnit „unit-test“</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>CppUnit „filter-test“</td>
<td>✔</td>
<td>✔</td>
<td>(-)</td>
<td>✔</td>
<td>✗</td>
<td>✔</td>
</tr>
<tr>
<td>JUnit „complex“</td>
<td>✔</td>
<td>✔</td>
<td>(-)</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>qadevOOo „unoapi“</td>
<td>✗</td>
<td>✔</td>
<td>(-) / ✗</td>
<td>(-) / ✔</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Python unittest</td>
<td>✔</td>
<td>✔</td>
<td>✔ / ?</td>
<td>✔</td>
<td>✔</td>
<td>(-)</td>
</tr>
</tbody>
</table>
Better Testing With Assertions

- Use `assert()` liberally in the product code to detect invalid states
- Assertion failure $\rightarrow$ abort and test failure
- But not necessarily good defect localization
Code Coverage

- GCC -fprofile-arcs -ftest-coverage
- LCOV
- http://lcov.libreoffice.org/
- Updated daily
- Thanks to Maarten Hoes
LCOV Code Coverage

Lines Hit
Lines Total
Lines Coverage

01.04.2014
01.09.2014
01.02.2015
01.07.2015
More Tests

- “Crash Testing”
  - Import and export 80k documents
  - See talk by Caolán McNamara
- Performance test
  - Callgrind profiling
  - make perfcheck
  - [http://perf.libreoffice.org/](http://perf.libreoffice.org/)
- Additional out-of-tree tests in test-files.git repo
- Thanks to Matúš Kukan, Laurent Godard, Norbert Thiebaud
Now … go forth and write more tests!

Thanks for listening.