The future of testing in Pharo

Julien Delplanque
julien.delplanque@inria.fr
About me

• 2nd year of PhD in RMoD team

• Hacking Pharo around many aspects… just for fun :-)

• Hit by testing topic « by accident »
The one who work on this topic right now

• 2nd year of Master

• Doing an internship in RMoD team for 6 months

• Master thesis on the enhancements of tests in Pharo

Her other mentors on Github

Guille

Steph
Questions this talk addresses

• What infrastructure Pharo provides for testing?

• How can we enhance testing experience?
What infrastructure Pharo provides for testing?
What infrastructure Pharo provides for testing?

- SUnit-UI
- Jenkins-Tools
- System browser

Pharo’s SUnit
What infrastructure Pharo provides for testing?

In fact,

<table>
<thead>
<tr>
<th>SUnit-UI</th>
<th>Jenkins-Tools</th>
<th>System browser</th>
</tr>
</thead>
<tbody>
<tr>
<td>Custom layer for visiting, « deciding color » of results, …</td>
<td>Custom layer for visiting, « deciding color » of results, …</td>
<td>Custom layer for visiting, « deciding color » of results, …</td>
</tr>
</tbody>
</table>

Pharo’s SUnit
The infrastructure we want

Dr Tests  Jenkins-Tools  System browser

Common layer for visiting, «deciding color» of results, ...

Pharo’s SUnit
The infrastructure we want

DrTests Plugins

DrTests, Jenkins-Tools, System browser

Enhanced Pharo’s SUnit
How can we enhance testing experience?
What can we do with tests?

<table>
<thead>
<tr>
<th>Managed by SUnit-UI</th>
<th>Runner</th>
<th>Coverage</th>
<th>Profile</th>
<th>Parametrisable test</th>
<th>Mutation testing</th>
<th>Run examples in comment</th>
<th>Rotten Tests Finder</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>Test cases</td>
<td>Test cases</td>
<td>Test cases</td>
<td>Test cases + parameters</td>
<td>Test cases + mutations</td>
<td>Comments</td>
<td>Test cases</td>
<td>...</td>
</tr>
<tr>
<td>Output</td>
<td>Results of tests run</td>
<td>Percentage of methods covered</td>
<td>Time taken for each test run</td>
<td>Results of tests run depending on parameter</td>
<td>Mutants to be killed</td>
<td>Comments containing failing examples</td>
<td>Rotten tests</td>
<td>...</td>
</tr>
</tbody>
</table>
TestRunner UI

- Packages containing tests
- TestCases
- Results summary

Test Runner

- Run tests
- Profile test execution
- Analyse code coverage
- Run Failures
- Re-run failures or errors only
- Failed tests
- Run Errors
- Export results
- Errors

166 run, 164 passes, 2 skipped, 0 expected failures, 1 failures, 1 errors, 0 unexpected passes

Base64Tests#testRangeEncoding

Base64Tests#testPrimesEncoding
**TestRunner UI: coverage**

<table>
<thead>
<tr>
<th>Class Method</th>
<th>Description</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConnectionQueue</td>
<td>connectionCount</td>
<td>Network Kernel</td>
</tr>
<tr>
<td>ConnectionQueue</td>
<td>destroy</td>
<td>Network Kernel</td>
</tr>
<tr>
<td>ConnectionQueue</td>
<td>getConnectionOrNil</td>
<td>Network Kernel</td>
</tr>
<tr>
<td>ConnectionQueue</td>
<td>getConnectionOrNilLenient</td>
<td>Network Kernel</td>
</tr>
<tr>
<td>ConnectionQueue</td>
<td>initPortNumber:queueLength:</td>
<td>Network Kernel</td>
</tr>
<tr>
<td>ConnectionQueue</td>
<td>isValid</td>
<td>Network Kernel</td>
</tr>
<tr>
<td>ConnectionQueue</td>
<td>listenLoop</td>
<td>Network Kernel</td>
</tr>
<tr>
<td>ConnectionQueue</td>
<td>oldStyleListenLoop</td>
<td>Network Kernel</td>
</tr>
<tr>
<td>ConnectionQueue</td>
<td>portNumber:queueLength:</td>
<td>Network Kernel</td>
</tr>
<tr>
<td>ConnectionQueue</td>
<td>pruneStateConnections</td>
<td>Network Kernel</td>
</tr>
<tr>
<td>ConnectionRefused</td>
<td>host</td>
<td>Network Kernel</td>
</tr>
<tr>
<td>ConnectionRefused</td>
<td>HostPort:</td>
<td>Network Kernel</td>
</tr>
<tr>
<td>ConnectionRefused</td>
<td>HostPort:</td>
<td>Network Kernel</td>
</tr>
</tbody>
</table>

**connectionCount**

"Return an estimate of the number of currently queued connections. This is only an estimate since a new connection could be made, or an existing one aborted, at any moment."

```python
self pruneStateConnections.
^accessSchema critical: [connections size]
```
TestRunner UI: profile

```
5 - 7365 tallies, 7370 msec.

**Tree**
-------------------------------
Process: (40s) Morphic UI Process: nil
-------------------------------

10.1% [745ms] TestRunner>>runTestSuites:
  10.1% [745ms] TestExecutionEnvironment class>>runTestsBy:
    10.1% [745ms] DefaultExecutionEnvironment>>runTestsBy:
      10.1% [745ms] TestExecutionEnvironment(ExecutionEnvironment)>>beActiveDuring:
        10.1% [745ms] CurrentExecutionEnvironment class>>activate:for:
          10.1% [745ms] BlockClosure>>ensure:
            10.1% [745ms] CurrentExecutionEnvironment class>>activate:for:
              10.1% [745ms] TestRunner>>runTestSuites:
                10.1% [745ms] Set(Collection)>>do:displayingProgress:
                  10.1% [745ms] Set(Collection)>>do:displayingProgress:every:
```
What can we do with tests?

<table>
<thead>
<tr>
<th>Managed by SUnit-UI</th>
<th>No integration</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Runner</strong></td>
<td><strong>Parametrizable test</strong></td>
</tr>
<tr>
<td><strong>Coverage</strong></td>
<td><strong>Mutation testing</strong></td>
</tr>
<tr>
<td><strong>Profile</strong></td>
<td><strong>Run examples in comment</strong></td>
</tr>
<tr>
<td><strong>Input</strong></td>
<td><strong>Rotten Tests Finder</strong></td>
</tr>
<tr>
<td>Test cases</td>
<td>Test cases + parameters</td>
</tr>
<tr>
<td>Test cases</td>
<td>Test cases + mutations</td>
</tr>
<tr>
<td>Test cases</td>
<td>Comments</td>
</tr>
<tr>
<td>Test cases + parameters</td>
<td>Test cases</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Output</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Poor integration</strong></td>
</tr>
<tr>
<td>Results of tests run</td>
</tr>
<tr>
<td>Time taken for each test run</td>
</tr>
<tr>
<td>Results of tests run depending on parameter</td>
</tr>
<tr>
<td>Mutants to be killed</td>
</tr>
<tr>
<td>Comments containing failing examples</td>
</tr>
<tr>
<td>Rotten tests</td>
</tr>
</tbody>
</table>

- Managed by SUnit-UI: Runner, Coverage, Profile
- No integration: Parametrizable test, Mutation testing, Run examples in comment, Rotten Tests Finder
- Input: Test cases
- Output: Results of tests run, Percentage of methods covered, Time taken for each test run
- Poor integration: Results of tests run depending on parameter, Mutants to be killed, Comments containing failing examples, Rotten tests

What can we do with tests?

- Managed by SUnit-UI
- No integration

Test cases

Managed by SUnit-UI:
- Runner
- Coverage
- Profile

No integration:
- Parametrizable test
- Mutation testing
- Run examples in comment
- Rotten Tests Finder

Input:
- Test cases

Output:
- Results of tests run
- Percentage of methods covered
- Time taken for each test run

Poor integration:
- Results of tests run depending on parameter
- Mutants to be killed
- Comments containing failing examples
- Rotten tests

Results of tests run depending on parameter
- Test cases
- Parameters

Comments containing failing examples
- Test cases
- Parameters

Rotten tests
- Test cases
- Parameters
Proposal: Dr Tests

Power-up testing experience in Pharo by:

Developing and promoting DrTests as the new UI for testing

- Written in Spec
- Extensible via plugins
- Provides good model to configure, run and gather results from plugins
Dr Tests

Plugin selected
Packages under analysis
Plugin input
Results tree
Plugin-defined action(s)

Logging label
Start plugin
Browse result

Dr Tests - Tests Runner

Packages (1 selected):
- RottenTestsFinder-Tests
- Rubric-Tests
- STON Tests
- STUnit Core
- STUnit-Tests
- STUnit-UiTesting
- ScriptLoader-tests
- Shout-Tests
- Skit-Tests
- Smart/Suggestions-Tests

Test cases (7 selected):
- RTFFakeTestClass
- RTFFakeTestSuperClass
- RTFLeadsToAssertPrimitiveCallCheckerTest
- RTFMethodCallsCollectorTest
- RTFSelfCallInterpreterTest
- RFTSelfCallTreeCleanerTest
- RottenTestsFinderTest

Results:
- Errors(0)
- Failures(1)
- Re-run
- Skipped tests(0)
- Passed tests(17)

2018-07-02 14:45: Tests finished.
Mini Dr Tests

- Simplified view to use a plugin once it is configured

Button to re-run according to configuration

Go back to normal UI to configure plugins, input, etc.
Mini Dr Tests

Button’s color gives visual feedback about last result of plugin
Dr Tests model

Plugin Configuration
Created from UI or by scripting

Plugin
Plugin run, provides updates through announcements

Plugin Result
Can be queried from UI or by scripting
Conclusion
Vision

» We want a testing ecosystem able to evolve

★ Enhanced SUnit

★ Uniform API for SUnit clients

★ Plugin-based testing UI = Dr Tests

★ More tools to handle tests = Dr Tests plugins
What’s next?

Dayne is working on Dr Tests and SUnit enhancements

Clothilde will work on Smart Tests soon
We can test it!
You can help the effort!

We are interested in your inputs and contributions!

@juldelplanque
juliendelplanque/DrTests