STREAMLINING PERFORMANCE VERIFICATION
THROUGH AUTOMATION AND DESIGN
Presented by Gabor Megyaszai
BACKGROUND – HISTORICAL PROBLEMS

• Legacy test cases
  • Long setup and tear down time
  • Non reusable automation
  • Huge amount of data collected, but not analyzed
• Low automation level in SUT deployment and configuration
  • Complex and greatly varying configuration
• No supervision over execution
  • Futile test case execution
RENEWAL ACTIONS
Refactoring test cases

- Differentiate between load testing and performance validation
- Test on lowest possible level
- Test with smallest possible configuration
- Create for reuse – keyword driven testing
- Infrastructure capability without deployment
  - KPI measurement against reference baseline
Deployment and configuration design

- Design principles
  - Deconstruct
  - Stabilze
  - Standardize
  - Automate

Original test cases

Lowest possible increment

Load test
Measurements
Validation
Provocative

Playbooks
Configuration
Keywords
Scripts
Automated virtual Environment Deploy

- Performance testing environment creation and on demand deploy
  - SUT and Non SUT
- Environment lifecycle management
  - VM and container
- Multiple triggering
  - Jenkins
  - GUI with manual triggering
Automated virtual Environment Deploy
### Testing Environment objects

<table>
<thead>
<tr>
<th>name</th>
<th>status</th>
<th>TAS</th>
<th>HSS</th>
<th>Generators</th>
<th>Terminate and Delete Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>vTAS19</td>
<td>deployed</td>
<td>vTAS19</td>
<td>HSS_vTAS19</td>
<td>GEN1901</td>
<td><img src="X" alt="Delete" /></td>
</tr>
<tr>
<td>vTAS24</td>
<td>deployed</td>
<td>vTAS24</td>
<td>HSS_vTAS24</td>
<td>GEN2401</td>
<td><img src="X" alt="Delete" /></td>
</tr>
<tr>
<td>vTAS30</td>
<td>deployed</td>
<td>vTAS30</td>
<td>HSS_vTAS30</td>
<td>GEN3001</td>
<td></td>
</tr>
</tbody>
</table>

### TAS objects

<table>
<thead>
<tr>
<th>TAS Name</th>
<th>Networks and IP Addresses</th>
<th>State in OS</th>
<th>Other info</th>
<th>Terminate or Update</th>
</tr>
</thead>
<tbody>
<tr>
<td>vtas05</td>
<td><img src="link" alt="Show Networks and IPs" /></td>
<td><img src="link" alt="CREATE COMPLETE" /></td>
<td><img src="link" alt="Created by admin" /></td>
<td><img src="X" alt="Delete" /> <img src="link" alt="Update System" /></td>
</tr>
</tbody>
</table>

### IPSL_Generator objects

<table>
<thead>
<tr>
<th>Stack Name</th>
<th>tas name</th>
<th>resource name</th>
<th>State in OS</th>
<th>Networks and IP Addresses</th>
<th>Templates</th>
<th>Terminate and Delete Generator</th>
</tr>
</thead>
<tbody>
<tr>
<td>GEN0601</td>
<td>vtas05</td>
<td>vtas06_ipsl_g1</td>
<td><img src="link" alt="CREATE COMPLETE" /></td>
<td></td>
<td><img src="link" alt="Download template" /></td>
<td><img src="X" alt="Delete" /> <img src="link" alt="Delete all vtas05 generators" /></td>
</tr>
</tbody>
</table>

Networks and IP Addresses:

**User Conference on Advanced Automated Testing**

26-28/10/2016
Test environment parameter handling

Python and Django based web application

- Store environment information
- Store traffic profiles / profile elements
- Environment selection for CI
- Environment configuration

User Conference on Advanced Automated Testing
Test execution

• ROBOT framework (generic automation framework for ATDD)
• Minimum viable unit
  • Test case
  • Environment
• System health check at start
• Continuous traffic and system check
• Collection of relevant data
• Automatic verdict
# Results in numbers

<table>
<thead>
<tr>
<th></th>
<th>Previous method</th>
<th>Renewed method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deployment time</td>
<td>8 hours</td>
<td>30 minutes</td>
</tr>
<tr>
<td>Configuration time</td>
<td>10 days</td>
<td>17 minutes</td>
</tr>
<tr>
<td>Test execution time</td>
<td>3.7 hours</td>
<td>1.3 hours</td>
</tr>
<tr>
<td>Futile test execution ratio</td>
<td>27% (release average)</td>
<td>&lt;3%</td>
</tr>
<tr>
<td>Number of parallel execution</td>
<td>Maximum 14</td>
<td>Maximum 58</td>
</tr>
</tbody>
</table>

© All rights reserved
Benefits

- Faster feedback to development
- Decreased turnover time
  - Broader test set
  - New test cases
    - Interface tests
    - Single container tests
  - More time for exploratory testing
    - Chaos and robustness