LabVIEW Unit Testing

Outlook & Tutorial
Unit Testing – Why?
Correctness & Completeness

Test your SW for **correctness** (common sense, V model etc.)

**EXAMPLE**
Divide(6,3) = 2  **PASSED**

Test your SW for **completeness/”coverage”** (ISO etc.)

**EXAMPLE**
Divide(6,0) = 0  **FAILED**

Test your SW **units** (divide and conquer/modularity)

**EXAMPLE**
Calc(Add(3,Divide(6,0))) = 3  **FAILED** (‘Add’ or ‘Divide’ failed?)
A LabVIEW example

Challenge 1: write test for critical ("cover critical")
Challenge 2: measure that critical is tested
Coverage

- Avoid **untested code**
- Prove **reachability** (no “dead code”)
- Required by
  - **Standards**, e.g., ISO 26262
  - **Companies**
Unit Testing LabVIEW code – *Tool options*
# Unit testing tool alternatives for LabVIEW

<table>
<thead>
<tr>
<th>Unit testing tool for LabVIEW</th>
<th>Model(1)</th>
<th>Price</th>
<th>Code Coverage Measurement</th>
<th>Scalability(4)</th>
<th>Real-time target</th>
<th>Source code availability</th>
<th>API (CI support)</th>
<th>NXG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit Test Framework</td>
<td>xUnit-style</td>
<td>$1,555 or bundled(2)</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>None (proprietary)</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>VI Tester</td>
<td>xUnit-style</td>
<td>Free</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>Yes (BSD-like)</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Caraya</td>
<td>Assertion-based</td>
<td>Free</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>Yes (BSD-like)</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>InstaCoverage Core</td>
<td>xUnit-style</td>
<td>Free</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>Inspection(3) (available on request)</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>InstaCoverage Pro</td>
<td>xUnit-style</td>
<td>€279/seat</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Inspection(3) (available on request)</td>
<td>✓</td>
<td>✗</td>
</tr>
</tbody>
</table>

1. Only UTF and InstaCoverage support explicit test configurations (lvtest and .instacov resp.).
3. Allows in-house customization.
4. The Unit Test Framework may slow down considerably with increasing project size.
MEMS Mics Test – Running example
**Setup**
- configure in-range clock rate
- configure out-range clock rate

**DUT**
BMP 280

**Teardown**
- expected = no error + chip ID
- expected = error

**TEST CASE**
- TEST CASE
  - Expected = no error + chip ID
  - Expected = error

**Unit test**
- VI under test

**Coverage**
100%

**WATCH DEMO**
Demo – InstaCoverage for NXG