If MacGyver was a Spec Editor.

Simple tools. Unbelievable results.
Testable assertions

- **MUST, SHOULD, MAY, etc.**
- The only bits that really matter.
- We need to guarantee they all get tested and verified.
MUST are expensive

- MUST = average ~3 tests+
- Redundancies are expensive.
- Variability and bugs are expensive.
So what would MacGyver do?

Spec (XHTML)

Test Suite (Selector API)

Tests (widgets)

Test Description (XML)

Test results (XML)
## Test Suite Visualized

### Compatibility table

Testable assertions are derived from the editor’s draft of the Widgets 1.0: Packaging and Configuration Specification.

TODO: define PASS, FAIL, etc.

<table>
<thead>
<tr>
<th>Testable Assertion</th>
<th>Test cases</th>
<th>Opera</th>
<th>Windows Mobile</th>
<th>BlackBerry</th>
<th>Samsung</th>
<th>LG</th>
<th>BOND</th>
<th>Wookie</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. ta-RRZxxvTPHx — If a user agent encounters a file matching a file name given in the file name column of the default start files table in an arbitrary folder, then user agent must treat that file as an arbitrary file.</td>
<td>widget / source — ✓ Tests that a UA does not go searching in an arbitrary folder (“abc123”) for default start files. To pass, the user agent must treat this widget as an invalid widget.</td>
<td>totest</td>
<td>incomplete</td>
<td>pass</td>
<td>totest</td>
<td>pass</td>
<td>pass</td>
<td>pass</td>
</tr>
<tr>
<td>2. ta-dxzVDWpaWg — A user agent must treat any file in an arbitrary folder or locale folders that uses the file name config.xml as an arbitrary file.</td>
<td>widget / source — ✓ Test to make sure that the UA only checks the root of the widget for config files, and not in an arbitrary folder. To pass, the user agent must treat this widget as an invalid widget (config file is not at the root).</td>
<td>totest</td>
<td>pass</td>
<td>pass</td>
<td>totest</td>
<td>fail</td>
<td>pass</td>
<td>pass</td>
</tr>
<tr>
<td>3. ta-GVvVxdeEUo — If a user agent supports acquisition of a potential Zip archive from a protocol that labels resources with a media type, then a user agent</td>
<td>widget / source — ✓ Test the ability for the user agent to handle correctly MIME labelled widgets. To pass, the widget start file must be index.htm</td>
<td>totest</td>
<td>cannotpass</td>
<td>cannotpass</td>
<td>totest</td>
<td>cannotpass</td>
<td>cannotpass</td>
<td>cannotpass</td>
</tr>
</tbody>
</table>
“If this is not the first name element encountered by the user agent, then the user agent must skip this element.”
Spec - XHTML

- `<p id="ta-LYL">
  ... the
  `<span class="product-ua">
    user agent
  </span>
  `<em class="ct">
    must
  </em>
  skip this element.
</p>`
<testsuite for="http://somespec/index.html">
  <test id="xyz" for="ta-YAY" src="xyz.wgt">
    Tests the UA’s ability to ignore subsequent repetitions of the name element. To pass, the name of the widget must be "PASS"
  </test>
  ...
</testsuite>
Test Results

- `<testsuite for="http://somespec/inde.html">

  <result
    for = "ta-YAY"
    verdict = "fail" >
    Test failed because
  </result>

  ...

</testsuite>`
Take Away

- Reduce your redundant MUSTs
- Use SHOULDs and MAYs with caution!
- Write in active voice
- Write for readers and write for testing.
- Keep things simple.