ACME’s testing process

Jeff Johnson, Jim Foucar
Software Engineering Team
ACME All-Hands Meeting
May 5, 2015
What do we mean by “testing?”

• A test either PASSes or FAILs
  • Establishing criteria is challenging, but worthwhile
  • Result is interpretable by non-experts
• A test should be easy for a non-expert to run
• All important features should be tested
  • Only tested features can be changed reliably!
• It should be easy to run tests frequently
  • Defects are localized in time
• It should be easy for someone to run subsets of tests
Why is testing important?

- Prevents bugs from creeping into the master branch
- Accelerates bug fixes
  - Failing tests provide an “imprint” that gives developers hints about the nature, whereabouts, and timeline of defects
- Builds confidence needed to make transformational software / algorithm / science changes
- Keeps “the model” in a good, reliable (and releasable) state for doing science
## ACME's testing machinery

<table>
<thead>
<tr>
<th>Running tests</th>
<th>Maintaining test database</th>
<th>Continuous integration</th>
</tr>
</thead>
<tbody>
<tr>
<td>create_test</td>
<td>testlist.xml</td>
<td>Jenkins / CDash</td>
</tr>
<tr>
<td>cs.status.*</td>
<td>update_acme_tests</td>
<td></td>
</tr>
<tr>
<td></td>
<td>manage_xml_entries</td>
<td></td>
</tr>
</tbody>
</table>
Running a test suite with create_test

- Run within ACME/scripts directory
- Many options — not all needed all the time (run without arguments to get help)

```bash
cd projects/ACME/scripts
mkdir $SCRATCH/acme-baseline
./create_test -xml_mach edison \
   -xml_compiler intel \
   -xml_category acme_developer \
   -testid acme_dev \
   -testroot $SCRATCH/acme_dev \
   -baselineroot $SCRATCH/acme-baseline \
   -project acme
```
Checking the status of a running test suite with `cs.status.*` scripts

- “Test root” directory contains a `cs.status.testid.machine` script

```bash
cd $SCRATCH/acme_dev
./cs.status.acme_dev.edison
...
PAS ERS.f19_g16_rx1.A.edison_intel
PAS ERS.f19_g16_rx1.A.edison_intel.memleak
PAS ERS.f19_g16_rx1.A.edison_intel.generate./scratch2/scratchdirs/johnson/acme-baseline-testcases
FAIL ERS_IOP4c.f19_g16_rx1.A.edison_intel
BFAIL ERS_IOP4c.f19_g16_rx1.A.edison_intel.generate./scratch2/scratchdirs/johnson/acme-baseline-testcases
RUN PEA_P1_M.f45_g37_rx1.A.edison_intel.G.acme_dev
PEND SMS.ne30_f19_g16_rx1.A.edison_intel
...
```
# Interpreting test results

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>PASS</td>
<td>Test passed!</td>
</tr>
<tr>
<td>GEN</td>
<td>Test was generated, not run yet</td>
</tr>
<tr>
<td>PEND</td>
<td>Test is waiting in a queue</td>
</tr>
<tr>
<td>RUN</td>
<td>Test is running</td>
</tr>
<tr>
<td>CHECK</td>
<td>Manual review of data required</td>
</tr>
<tr>
<td>ERROR</td>
<td>Checking of test result failed</td>
</tr>
<tr>
<td>EXPFAIL</td>
<td>Expected test failure</td>
</tr>
<tr>
<td>FAIL</td>
<td>Test failed (run failure or inexact restart)</td>
</tr>
<tr>
<td>BFAIL</td>
<td>Baseline result could not be found to compare</td>
</tr>
<tr>
<td>TFAIL</td>
<td>Test setup error</td>
</tr>
<tr>
<td>SFAIL</td>
<td>Generation of test failed in scripts</td>
</tr>
<tr>
<td>CFAIL</td>
<td>check_case script failed (env or build problem)</td>
</tr>
</tbody>
</table>
You can add your own tests to existing ACME test suites

1. Edit script to rebuild acme_developer and acme_integration test suites:
   
   `ACME/scripts/acme/update_acme_tests`

2. Modify TEST_SUITES near top of file

3. Change to the directory containing the testing database:
   
   `cd ACME/scripts/ccsm_utils/Testlistxml`

4. Execute the script, rebuilding the database:
   
   `../../acme/update_acme_tests`
You can also add a test case to another pre-existing test suite

1. Change to the directory containing the testing database:
   `cd ACME/scripts/ccsm_utils/Testlistxml`

2. Extract the existing tests in the desired suite:
   
   `./manage_xml_entries -query -outputlist -machine machine -compiler compiler -category suite > my_tests.txt`

3. Edit the extracted tests in the file (my_tests.txt), adding one line for each new test.

4. Add the tests in the file to those in the database:
   
   `./manage_xml_entries -addlist -file my_tests.txt -category suite`

5. Replace the database with the new database:
   
   `mv testlist-date-time.xml testlist.xml`
You should generate a baseline for a
new test running on a machine

```
cd projects/ACME/scripts
./create_test -testname new-test-case.machine.compiler \
    -testroot $SCRATCH/new-test-case \
    -generate new-test-baseline-name \
    -baselineroot acme-baseline-dir \
    -project account
```
ACME’s testing policy is built into our procedure for code changes

1. Make your code changes in a feature branch
2. Run the acme_developer test suite on your branch and check the results to make sure the tests pass
3. If necessary, add your own test case(s) to an existing or new ACME test suite and run it to make sure these new tests pass
4. Issue a pull request and assign an integrator, who
   • integrates the branch into ‘next’
   • waits for results of acme_integration test suite
   • works with you to address issues
   • integrates the branch into master
Nightly test results are available on CDash (http://my.cdash.org/index.php?project=ACME_Climate)
Nightly test results are available on CDash (http://my.cdash.org/index.php?project=ACME_Climate)
Nightly test results are available on CDash
(http://my.cdash.org/index.php?project=ACME_Climate)
Upcoming enhancements will make testing easier

- Multiple, coexisting test databases (make your own!)
- Simplified tools for maintaining test databases
- A set of shorter-running test cases for some components (better coverage)
- Better (more specific) diagnostic reporting for troubleshooting test results
You can find more information on ACME Confluence

- Documentation: https://acme-climate.atlassian.net/wiki/display/Docs/Testing
- Support page: https://acme-climate.atlassian.net/wiki/display/SE/Help%3A++Testing (monitored by SE team)
- Your feedback is appreciated!