Software Testing with NA-MIC Kit

Xiaodong Tao, Ph.D.
Visualization and Computer Vision Lab
GE Global Research Center
Why Test Daily?

• Large code base too large for any single developer to understand
• Developers distributed around the world
• Identify problems as they occur
• Insure that object API remains unchanged
• Provide feedback to developers as they experiment with new implementations
Regression Testing

• Compare generated image against image from baseline implementation
  – pixel-by-pixel comparison
  – can use a threshold metric
  – adjusted for effects like dithering
Software Development Cycle

Developers worldwide

Code

CVS/SVN/
git

CTest

CDash

Web Browser
Testing Terminology

Sites
- dash5.kitware
- BillsBasement
- iris.elmtech

Builds
- Linux-g++
- Linux-gcc4
- Linux-gcc3

Dashboard
Tools in NA-MIC Kit
CMake, CTest, and CDash

- Testing client
- Distributed with CMake
- Submit test results to the dashboard
- XML submission
- Independent of CMake

- New generation of Dart
- Written in Php/ Javascript
- MySQL
- Open Source
- Run on top of a web server
- XSL implementation
- Web 2.0
Testing Submission

Compiler/Tester - CTest

HTTP PUT - XML Submission
- Update.xml
- Configure.xml
- Build.xml
- Test.xml
- Coverage.xml
- DynamicAnalysis.xml

XML Parsing

Backup XML File
Fill Database
Send Email
Update RSS Feed

CDash

National Alliance for Medical Image Computing
http://na-mic.org

MICCAI 2010, Beijing, China
# Slicer3 Dashboard

[![Slicer3 Dashboard](image)](http://www.cdash.org/CDash/index.php?project=Slicer3)

**Site Name:** District9.kitware  
**Build Name:** WinXP-64-VS2008-Kww-Qt4.6.2-Python-Release  
**Build Date:** 2010-08-15 03:27:15  
**Test Timing:** Passed  
**Rigid Registration Test 00** Passed

<table>
<thead>
<tr>
<th>Continuous Super Build</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Build Name</td>
<td>Update</td>
<td>Min</td>
<td>Error</td>
</tr>
<tr>
<td>dash21.kitware</td>
<td>Debian5-good-3.2-Kww-Qt4.6.2-Python-Release</td>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>dash21.kitware</td>
<td>Debian5-good-3.2-Kww-Qt4.6.2-Python-Release</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>dash21.kitware</td>
<td>Debian5-good-3.2-Kww-Qt4.6.2-Python-Release</td>
<td>7</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>dash21.kitware</td>
<td>Debian5-good-3.2-Kww-Qt4.6.2-Python-Release</td>
<td>5</td>
<td>0.1</td>
<td>0</td>
</tr>
<tr>
<td>District9.kitware</td>
<td>WinXP-64-VS2008-Kww-Qt4.6.2-Python-Release</td>
<td>6</td>
<td>0.2</td>
<td>0</td>
</tr>
</tbody>
</table>

**Execution Time (s):** 1.34 (mean: 1.33 std: 0.03)

**Command Line:**  
C:\Dashboards\Continuous\Slicer3-Superbuild-KwwAndQt-Python\Slicer3-build\Slicer3.exe -launch RigidRegistrationTest ModuleEntryPoint --resampledmovingfilename C:\Dashboards\Continuous\Slicer3-Superbuild-KwwAndQt-Python\Slicer3-build\Testing\Temporary\RigidRegistrationTest00.nhdr --outputtransform C:\Dashboards\Continuous\Slicer3-Superbuild-KwwAndQt-Python\Slicer3-build\Testing\Temporary\RigidRegistrationTest00Transform.txt --Initialtransform C:\Dashboards\Continuous\Slicer3-Superbuild\Testing\Data\Input\itkAffineTransform00.txt --spatielsamples 100 00 --histogrambins 32 --iterations 50 --learningrate 0.01 --translationscale 100  
C:\Dashboards\Continuous\Slicer3-Superbuild\Testing\Data\Input\CTHeadAxial.nhdr  
C:\Dashboards\Continuous\Slicer3-Superbuild\Testing\Data\Input\CTHeadAxial.nhdr

**Completion Status:** Completed

**Test output:**

c test needs: CTEST_FULL_OUTPUT
<filter-start>
<filter-name>OrientImageFilter</filter-name>
<filter-comment> "Orient Fixed Image" </filter-comment>
</filter-start>
<filter-end>

[Show Test Time Graph]  
[Show Failing/Passing Graph]
How to write a good test?

• Use empirical values (5/2=2.5)
• Test all of the use cases you can think of (even the boring/most difficult ones)
• Make your tests as efficient as possible
• Have someone else write the test for your algorithm
## Contributing Code to Slicer3

<table>
<thead>
<tr>
<th>Slicer Lbs</th>
<th>• ModuleDescriptionParser</th>
<th>Non-slicer specific support libraries</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• GenerateCLP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• vtkITK</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• MRML</td>
<td></td>
</tr>
<tr>
<td>Slicer Base</td>
<td>• Application logic</td>
<td>Common infrastructure for Slicer</td>
</tr>
<tr>
<td></td>
<td>• Widgets</td>
<td>applications</td>
</tr>
<tr>
<td>Built in modules</td>
<td>• Slice viewers</td>
<td>Full access to Slicer internals</td>
</tr>
<tr>
<td></td>
<td>• Models</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Fiducials</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Transforms</td>
<td></td>
</tr>
<tr>
<td>Loadable modules</td>
<td>• Query Atlas</td>
<td>Full access to Slicer internals</td>
</tr>
<tr>
<td></td>
<td>• QDEC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Volume rendering</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• ChangeTracker</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• EMSegment</td>
<td></td>
</tr>
<tr>
<td>Scripted modules</td>
<td>• Editor</td>
<td>Limited access to Slicer internals</td>
</tr>
<tr>
<td></td>
<td>• Teem Two Tensor Tractography</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• VMTK</td>
<td></td>
</tr>
<tr>
<td>Command line modules</td>
<td>• Registration</td>
<td>Restricted access to Slicer internals</td>
</tr>
<tr>
<td>Daemon</td>
<td>• OpenIGTLink</td>
<td>Access to MRML</td>
</tr>
<tr>
<td></td>
<td>• Stochastic Tractography</td>
<td></td>
</tr>
</tbody>
</table>
Steps to Creating CLM Tests

- Creating a Test Driver
- Designing a Test
- Configuring Build and Add the Test
- Configuring the Test
- Run the Test
Steps to Creating CLM Tests

- Creating a Test Driver
- Designing a Test
- Configuring Build and Add the Test
- Configuring the Test
- Run the Test
Creating a Test Driver

- **C++ Code**

```cpp
#include <iostream>
#include "itkTestMain.h"

void RegisterTests()
{
    REGISTER_TEST
        (CurvatureAnisotropicDiffusionTest);
}

#undef main
#define main CurvatureAnisotropicDiffusionTest
#include "CurvatureAnisotropicDiffusion.cxx"
```
Designing a Test

• Determining what functions to be tested
• Identifying test datasets
• Determining expected or baseline output for functions to be tested
• Covering as many functions as possible
• Covering as many use cases as possible
Configuring Build

• CMake Code

# CurvatureAnisotropicDiffusion tests
set (CLP CurvatureAnisotropicDiffusion)
add_executable(${CLP}Test ${CLP}Test.cxx)
add_dependencies(${CLP}Test ${CLP})
target_link_libraries(${CLP}Test ITKIO)
Add the Test

• CMake Code

add_test(${CLP}Test ${Slicer3_EXE})
  --launch ${CLP}Test
  --compare ${BASELINE}/${CLP}Test.nhdr
           ${TEMP}/${CLP}Test.nhdr

${CLP}Test
  --conductance 2
  --timeStep 0.0625
  --iterations 2

${TEST_DATA}/MRHeadResampled.nhdr
${TEMP}/${CLP}Test.nhdr  )
Configuring and Running the Test

• To configure the Test
  – Using CMake and your favorite compiler

• To run the Test
  ctest -V -R CurvatureAnisotropicDiffusionTest

• Start a new cycle
Acknowledgement

National Alliance for Medical Image Computing
NIH U54EB005149

NA-MIC community