Software development with continuous integration

Martin Ettl (FESG/MPIfR)
ettl@fs.wettzell.de

Alexander Neidhardt (FESG)
neidhardt@fs.wettzell.de

FESG
A critical view on scientific software

- Tendency to become complex and unstructured
- Highly specialized for dedicated purposes (e.g. PhD-thesis)
- Undocumented, unreadable, not maintainable code
- Error-prone, not well tested, not optimized realizations
- Written in different computer languages

A survey of nearly 2000 researchers showed how coding has become important part of researchers toolkit:

- 38 % spend at least one fifth of their time developing software
- But only 47 % have a good understanding of software testing

Information taken from: Nature 467, 775-777 (2010) | doi:10.1038/467775a
General rules to improve quality

Tips to make scientific software more robust:

- Use a software version-control system (cvs, svn, git, …)
  - Put source code, raw data files, parameters and other primary material into it to record what you did, and when.

- Write testable software:
  - Build large code from smaller, easily testable modules.

- Test the software:
  - Collect tests and automate the testing.

- Encourage sharing the software:
  - Others will use your software on different machines for maybe different purpose. This normally reveals hidden bugs.

*Information taken from: Nature 467, 775-777 (2010) | doi:10.1038/467775a*
The principles we follow in software development:

- Reduction to a few software languages (C/C++, Perl)
- Style guide with coding layout and coding policies
- Usage of automated code formatting tool
- Use of documentation generation
- Automated code tests and inspections (standardized Makefiles)
- Write a unit-test for each change
- Test on several platforms and architectures with multiple compilers
- Activate all possible compiler -W flags and reduce warnings
- Version controlled software development
- Update and commit changes every day to version control system
- Offering of tested, open source toolboxes (e-RemoteCtrl, SysMon)
Scientific Software

The Continuous Integration software development method
The Continuous Integration software development principle
The Continuous Integration software development method

Usage of open-source tools:

C/C++

- Static program analysis (cppcheck)
- Spell check of program and text files (codespell)
- Extracting of development information (StatSVN)
- Find non-reentrant functions in code (nsiqcppstyle)
- Find security problems (flawfinder)
- Detect common printf/scanf format errors (pscan)
- Generate developer documentation (doxygen)
- Detect redundant files in the repository (own development)

Candidate under evaluation:

- Static analysis: splint,
- Dynamic analysis: clang (address sanity checker)
Continuous Integration as a supporting service

http://econtrol-software.de/
## The Continuous Integration software development method
- Current project state overview -

<table>
<thead>
<tr>
<th>Projekt</th>
<th>SVN Statistics (statsvn)</th>
<th>Developer Docu (doxygen)</th>
<th>Static analysis (cppcheck)</th>
<th>Redundant files</th>
<th>Duplicated code (simian)</th>
<th>Project checker</th>
<th>Beautify files (artistic style)</th>
<th>xsamba.wtz</th>
<th>Autobuild</th>
<th>Unit tests</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Run time</td>
<td>(Daily 04:00)</td>
<td>(Daily 12:00)</td>
<td>(Daily 22:00)</td>
<td>(Daily 23:00)</td>
<td>(Daily 03:30)</td>
<td>(Daily 04:00)</td>
<td>(Sa. 00:00)</td>
<td>(Daily 02:00)</td>
<td></td>
<td>(Daily 00:00)</td>
</tr>
<tr>
<td>sir</td>
<td>statistics</td>
<td>n.a.</td>
<td></td>
<td></td>
<td></td>
<td>n.a.</td>
<td>report</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tools</td>
<td>statistics</td>
<td>n.a.</td>
<td></td>
<td></td>
<td></td>
<td>n.a.</td>
<td>report</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>vbl</td>
<td>statistics</td>
<td>n.a.</td>
<td></td>
<td></td>
<td></td>
<td>n.a.</td>
<td>report</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sysmon</td>
<td>statistics</td>
<td>n.a.</td>
<td></td>
<td></td>
<td></td>
<td>n.a.</td>
<td>report</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wxgui</td>
<td>statistics</td>
<td>n.a.</td>
<td></td>
<td></td>
<td></td>
<td>n.a.</td>
<td>report</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The Continuous Integration software development method

Automatic cppcheck report: /home/subversion/codecheck/trunk/scripts/static_analysis/cppcheck

Generated: Sun Mar 4 05:07:28 UTC 2012

Cppcheck version: Cppcheck 1.53

Start of static checking

Statistc

<table>
<thead>
<tr>
<th>Category</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>error</td>
<td>4</td>
</tr>
<tr>
<td>warning</td>
<td>0</td>
</tr>
<tr>
<td>warning</td>
<td>0</td>
</tr>
<tr>
<td>style</td>
<td>42</td>
</tr>
<tr>
<td>performance</td>
<td>0</td>
</tr>
<tr>
<td>information</td>
<td>0</td>
</tr>
</tbody>
</table>

End of static checking
The Continuous Integration software development method

Automatic cppcheck report: /home/subversion/codecheck/trunk/scripts/static_analysis/cppcheck

Generated: Sun Mar 4 05:07:28 UTC 2012

Cpplint version: Cpplint 1.53

Start of static checking

End of static checking
The Continuous Integration software development method
The Continuous Integration software development method

Total Lines of Code: 31249
Most Recent Commit: 2012-10-08 12:22

vlbi: Lines of Code

vlbi: Commit Activity

vlbi: Author Activity

modifying  adding
The Continuous Integration software development method
The Continuous Integration software development method
- unit-testing -

```c
void strToUpper()
{
    ASSERT_EQUALS("XYZ", simple_string_util::strToUpper("xyz"));
}
```

Expected result!

Call of a function to test, here
```c
strToUpper()
```
The Continuous Integration software development method
- unit-testing -

Tests:
- simple_stl_util::bCountValues1
- simple_stl_util::bContainsValue
- simple_stl_util::bGetFirstKeyByValue
- simple_stl_util::bGetFirstKeyByValue1
- testsuite::GetListofTests
- testsuite::vGetRandomNumber
- testsuite::simple_testsuite_assert_results_bAddAssertResult
- testsuite::simple_testsuite_assert_results_Ctor
- testsuite::LCSLength
- testsuite::LCSLength1
- testsuite::MeanAngle
- testsuite::MeanAngle1
- Test::IsAlphaNum
- Test::IsAlpha
- Test::IsControlChar
- Test::IsDigit

Test statistics
#Asserts (#Tests)...: 20187 (1029)
#To do's.............: 7
#Successful asserts.: 20187
#Failed asserts.....: 0

martin@martin:/projects/tools/trunk/modules/simple_testsuite/make$
Experience

The code of the VVLJ DIFX software correlator is checked in regular intervals for common programming errors. These results are offered to the authorized, registered developers of the DIFX developer community and can be found here.

**DIFX Continuous Integration Code Check Results**

The code of the NASA Mark IV Field System re-archiving is checked in regular intervals for common programming errors. These results are offered to the authorized, registered developers of the F4 developer community and can be found here.

**NASA F4 Continuous Integration Code Check Results**

![Image of the E-Control SW interface](image-url)
Experience

Do not miss our e-RemoteCtrl demonstration at lunchtime today!

Wettzell Toolbox

e-RemoteCtrl
Thank you
The service is also available for your software.  
http://www.econtrol-software.de