Tools and rules to encourage quality for C/C++ software

Katarina Sigerud, Wojciech Sliwinski, Vito Baggiolini, Jean-Claude Bau, Stephane Deghaye, Jeremy Nguyen Xuan, Xavier Piroux, Gennady Sivatskiy, Ilia Yastrebov
CERN, CH-1211, Geneva 23, Switzerland

Background

In view of improving the quality and integrity of the products released in operations, the CERN accelerator Controls group decided in 2011 to apply a systematic approach to quality assurance for C/C++ software developed in the group.

Objectives

- Agree on and establish best software quality practices.
- Choose tools for quality and integrate these tools in the software development process.
- Identify appropriate tools corresponding to our criteria: open-source, easy to use, active developer community, good documentation.

Future plans

The next steps will be to:

- Draw conclusions from the early adopters of the Coverity tool for static code analysis.
- Encourage all projects to use the CMX library for exposure of runtime, in-process metrics.
- Identify and agree on a common tool for code coverage analysis, an important metric to encourage developers to do quality assurance.

References

[1] GoogleTest: https://code.google.com/p/googletest/
[2] GoogleMock: https://code.google.com/p/googlemock/
[3] Bamboo: http://www.atlassian.com/software/bamboo
[4] Coverity: http://www.coverity.com/products/coverity-save.html
[5] F. Ehm et al, "CMX - A Generic In-Process Monitoring Solution for C and C++ Applications", ICALEPCS'13, San Francisco, CA, USA, October 2013

Results

A common build tool – "Make.generic"

- Based on GNU Make
- Common targets for all projects

```
MAKE_CONF = /acc/src/dsc/co/mw/Make.cmw

PRODUCT = myproduct
LIB_NAME = mylib
VERSION = 1.0.0

COMPILER_FLAGS = ...

# Include the generic make file
include /acc/src/dsc/co/cpp/2.x.x/Make.generic
```

Example product Makefile including Make.generic

Integrates common compile options

Static code analysis

- Common compile options to all C/C++ projects in the group.
- Automatically spot common mistakes and bug patterns with Coverity [4].

Example Coverity report with highlighted code defect

```
78 //------------------------------------------------------------------------------
79
80 string StringUtils::toString(float number, size_t digitsAfterDot)
81 {
82     char buf[50];
CID 10635: Calling risky function (SECURE_CODING)
[VERY RISKY]. Using "sprintf" can cause a buffer overflow when done incorrectly. Because sprintf() assumes an arbitrarily long string, callers must be careful not to overflow the actual space of the destination. Use snprintf() instead, or correct precision specifiers.
83     sprintf(buf, "%.*f", static_cast<int>(digitsAfterDot), number);
84     return buf;
85 }
86
87 //------------------------------------------------------------------------------
88
89 string StringUtils::toString(double number, size_t digitsAfterDot)
90 {
91     char buf[360];
CID 10634: Calling risky function (SECURE_CODING) [select defect]
92     sprintf(buf, "%.*f", static_cast<int>(digitsAfterDot), number);
93     return buf;
94 }
95
```

Generates manifest file

Manifest

- Build-time and dependency information for a binary.
- Easy to retrieve at runtime (using CMX) [5].

Example Manifest as extracted by ident

```
[cwe-513-vml142] /user/wsliwins/workspace/cmw/cmw-p
cmw-proxy:
    $Manifest: product=cmw-proxy;version=2.1.8;bui
ompiler=g++ (GCC) 4.4.7 20120313 (Red Hat 4.4.7-3);
;executable=true $
    $Manifest: product=cmw-log-stomp;version=1.5.0;build_date=2013-05-27 14:49:45;user=wsliwi
ns;compiler=g++ (GCC) 4.4.7 20120313 (Red Hat 4.4.7-3);os=Linux 2.6.32-358.2.1.el6.x86_64;cpu=
L866 $
    $Manifest: product=cmw-stomp;version=2.1.0;build_date=2013-05-27 14:39:04;user=wsliwins;c
ompiler=g++ (GCC) 4.4.7 20120313 (Red Hat 4.4.7-3);os=Linux 2.6.32-358.2.1.el6.x86_64;cpu=L866
 $
    $Manifest: product=cmw-util;version=1.5.0;build_date=2013-05-27 13:47:38;user=wsliwins;co
mpiler=g++ (GCC) 4.4.7 20120313 (Red Hat 4.4.7-3);os=Linux 2.6.32-358.2.1.el6.x86_64;cpu=L866
$
    $Manifest: product=cmw-log;version=2.7.6;build_date=2013-05-27 14:03:24;user=wsliwins;com
piler=g++ (GCC) 4.4.7 20120313 (Red Hat 4.4.7-3);os=Linux 2.6.32-358.2.1.el6.x86_64;cpu=L866 $
    $Manifest: product=cmw-rbac;version=5.3.0;build_date=2013-05-28 15:23:41;user=wsliwins;co
mpiler=g++ (GCC) 4.4.7 20120313 (Red Hat 4.4.7-3);os=Linux 2.6.32-358.2.1.el6.x86_64;cpu=L866
$
    $Manifest: product=cmw-rda;version=2.12.1;build_date=2013-05-28 15:56:28;user=wsliwins;co
mpiler=g++ (GCC) 4.4.7 20120313 (Red Hat 4.4.7-3);os=Linux 2.6.32-358.2.1.el6.x86_64;cpu=L866
$
    $Manifest: product=cmw-directory-client;version=1.4.0;build_date=2013-05-27 16:20:20;user
=wsliwins;compiler=g++ (GCC) 4.4.7 20120313 (Red Hat 4.4.7-3);os=Linux 2.6.32-358.2.1.el6.x86_
64;cpu=L866 $
```

Info retrieved via CMX

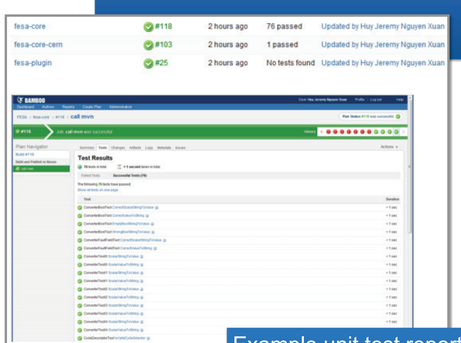
Runtime in-process metrics – "CMX"

- Inspired by JMX.
- Exposes runtime information, used by external diagnostics tools for checking thresholds, sending alerts or trending.

Runs unit tests

Unit testing, mocking and CI

- Unit tests written using Google Test [1] and Google Mock [2].
- Continuous integration using Atlassian Bamboo [3].



Example unit test report in CI server

```
[ RUN      ] TestAccessRulesMap.TestAuthorization20Roles
[       OK ] TestAccessRulesMap.TestAuthorization20Roles (28 ms)
[ RUN      ] TestAccessRulesMap.TestAccessMapSize1
[       OK ] TestAccessRulesMap.TestAccessMapSize1 (22 ms)
[ RUN      ] TestAccessRulesMap.TestAccessMapSize2000
[       OK ] TestAccessRulesMap.TestAccessMapSize2000 (50 ms)
[ RUN      ] TestAccessRulesMap.TestRoles1
[       OK ] TestAccessRulesMap.TestRoles1 (19 ms)
[ RUN      ] TestAccessRulesMap.TestRoles20
[       OK ] TestAccessRulesMap.TestRoles20 (28 ms)
[----------] 13 tests from TestAccessRulesMap (168 ms total)

[----------] Global test environment tear-down
[==========] 45 tests from 10 test cases ran. (11030 ms total)
[  PASSED  ] 36 tests.
[  FAILED  ] 9 tests, listed below:
[  FAILED  ] TestExpiration.TestDefaultLifetime
[  FAILED  ] TestExpiration.TestDefaultMasterLifetime
[  FAILED  ] TestExpiration.TestCustomLifetime
[  FAILED  ] TestExpiration.TestCustomMasterLifetime
[  FAILED  ] TestLocationLogin.TestLocationLogin
[  FAILED  ] TestRolePicker.TestRolePicker
[  FAILED  ] TestExplicitLogin.TestBasic
[  FAILED  ] TestExplicitLogin.TestAdvanced
[  FAILED  ] TestRelogin.TestRelogin

9 FAILED TESTS
```

Example output from running C++ unit tests written using the Google Test framework