Software Testing
Infrastructure
status

LCG Software Process & Infrastructure

(CERN, 10/23/02)
Index:

• Overview
• Unit-test
• Unit-test frameworks
  • CppUnit
  • Oval
• Unit-test structure and documentation
• Status & future plans
Overview:

**LCG applications area project:**
- software testing will be an integral part of the software development process
- All level of software testing should be run as part of an automatic process

**Software testing:**
- We will select:
  - simple techniques, tools and methodologies
  - run the tests in automatic way

**SPI project**

**Software testing**
- System test
- Integration test
- Unit test (work package test)

Individual developer domain
Unit Test:

**Unit-tests**
Should validate expected functionality at the level of:
- individual class
- small groups of collaborating classes
- work package

- Important code should have unit tests
- Tests should be written together with the code
- Code should pass all unit tests before it can be released

**CODE**

Code Documentation

**Unit-test SPI component (Version 1)**
- Test execution framework
- Testing naming and structure
- Test case specification template

**Automatic software test run process**

Test programs
Unit-test: Test frameworks (I)

**Aim:** to help developers:
- to produce code for unit-testing
- to run tests in automatic way

**Our constrains:**
- Avoid commercial software and licensing problems.
- Avoid “do it yourself solutions”
- Try to adopt commonly used open-source software.

**What we propose:**
- CppUnit
- Oval

**What we are doing:**
- Trying CppUnit and Oval in the LCG-POOL project
- Preparing “HowTo” documents to make easier the installation of these tools and the start with process

**Our inputs:**
- Contacts within HEP-community (CMS and G4 mainly, until now).
- What is available as free open source code.

**Our plans:**
- Analysis of the CppUnit and Oval tools in the POOL environment.
- Feedback from the experiments and big HEP software projects.
- Deliver and document the component soon.

What we propose:
Test frameworks (II): CppUnit

CppUnit:
Like JUnit but for C++
http://sourceforge.net/projects/cppunit

USED:
• in eXtreme Programing (XP)
• proposed for DataGrid

• Similar tools:
  Junit, PerlUnit, PyUnit, QtUnit
• Output in XML, compiler or text
• Windows version for MVC++6.0

• Just starting in LCG-POOL project
  (1 test running)

A simple test:
1. Subclass the TestCase CppUnit class
2. Override the method runTest().
3. When you want to check a value, call CPPUNIT_ASSERT(bool) and pass in an expression that is true if the test succeeds

Test results:
Run: 19 Failures: 2 Errors: 0

1) test: StringToolsTest::testToStringInt (F) line: 33 StringToolsTest.cpp
   equality assertion failed
   - Expected: 12345678
   - Actual : 123456789

2) test: ComplexNumberTest::testEquality (F) line: 22 ComplexNumberTest.cpp
   assertion failed
   - Expression: *m_10_1 == *m_1_1

PASS: test
---------------------
All 1 tests passed
---------------------

M. Gallas   IT-API
LCG SPI project: testing
Test frameworks (II\textsubscript{cont}): CppUnit

```cpp
class ComplexNumberTest : public CppUnit::_TestCase {
    public:
        ComplexNumberTest(std::string name) : CppUnit::_TestCase(name) {
        
    }

    void runTest() {
        CPPUNIT_ASSERT(Complex(10, 1) == Complex(10, 1));
        CPPUNIT_ASSERT(!(Complex(1, 1) == Complex(2, 2)));
    }

    int main() {
        ComplexNumberTest testcomplex("Test1_Name");
        testcomplex.runTest();
        cout<<"--------- This is the test we are running: \\
<<testcomplex.getName()<< " \ -----" \<<endl ;
        cout<<"--------- This is the test has number: \\
<<testcomplex.countTestCases()<< " \------" \<<endl ;
        return 0;
    }
};
```

M. Gallas   IT-API
LCG SPI project: testing
Test frameworks (III): Oval

Oval:
- validation and regression
- used in CMS

- Can be used for Unit-test.
- It is possible to set different run environments.
- Can run external scripts and external binaries.
- Authors modified it to run it with or without CMS specific environment (SCRAM ...)
  (Thanks to David Chamont for the changes)

- Just starting in LCG-POOL project (1 test running)

• Can be use for Unit-test.
• It is possible to set different run environments.
• Can run external scripts and external binaries.
• Authors modified it to run it with or without CMS specific environment (SCRAM ...)
  (Thanks to David Chamont for the changes)

• Just starting in LCG-POOL project (1 test running)

M. Gallas  IT-API
Test frameworks (III\textsubscript{cont}): Oval

```cpp
#include <iostream.h>
#include <stdlib.h>

int main()
{
    cout<<"hello\n";
    cout<<"[OVAL] input "
    cout<<"[OVAL] result return 0;"
}
```

```bash
[oval build]  ===============================
[oval build] instruction: g++ Progl.cpp -o Progl
[oval build] eval `oval runtime -csh for_diff`
[oval build] INPUT = ok
[oval build] ===============================
g++ Progl.cpp -o Progl

[oval run]  ==================================
[oval run] arguments:
[oval run] USER : mgallas
[oval run] HOST : pc:mpi31
[oval run] eval `oval runtime -csh for_diff`

"OvalFile" line 1 of 6 --16%-- col 22

[oval diff]  ================================
[oval diff] diff line: /*\{label\} input */
[oval diff] diff number: /*\{label\} result = (.)*/ ~ 15%
[oval diff]  ================================

### log #24 !="ref"
log: [label] input ok
### log #25 !="ref" (~15%)
log: [label] result = 1.1
"Progl.ref" line 1 of 36 --2%-- col 1
Unit-test: structure and documentation

SPI-CVS_structure component

• CVS testing structure
• Test-naming

Test documentation:
• Test cases specification template

Will help to have an automatic way to run test code

csv
doc
test

testA

testB
Ovalfile
Nametest ccp
Nametest ref
Nametest log
[Nametest in]
[Nametest out]
This document was generated by the POOL project to describe a specific set of steps and data along with the expected results. The document includes information on the short name, version, publication date, author(s), and status. The document is focused on testing the DataSvc service in an LCG environment, with a specific version and revision included.
Status & Future Plans:

**Status:**
- Installing the component
- Supporting the component at POOL project
- Doing some test cases with them
- Preparing Howto’s

**Future Plans:**
- Produce a first version of the **SPI-Testing Component (V1)** which will provide:
  - unit-test organization
  - test execution framework
  - test documentation and templates
  - complete user-documentation
- Feedback with the HEP experiments and big software projects.
INDEX review:

- Overview
- Unit-test
- Unit-test frameworks
  - CppUnit
  - Oval
- Unit-test structure and documentation
- Status & Future plans

Howto for installation and examples will be available soon...

Feedback and interaction are always welcome!!