



# 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

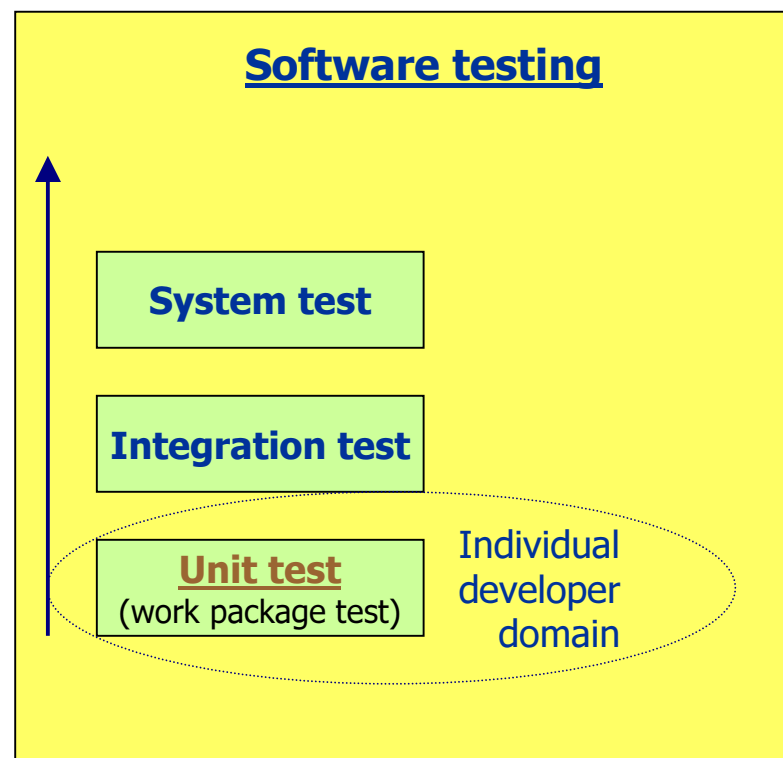
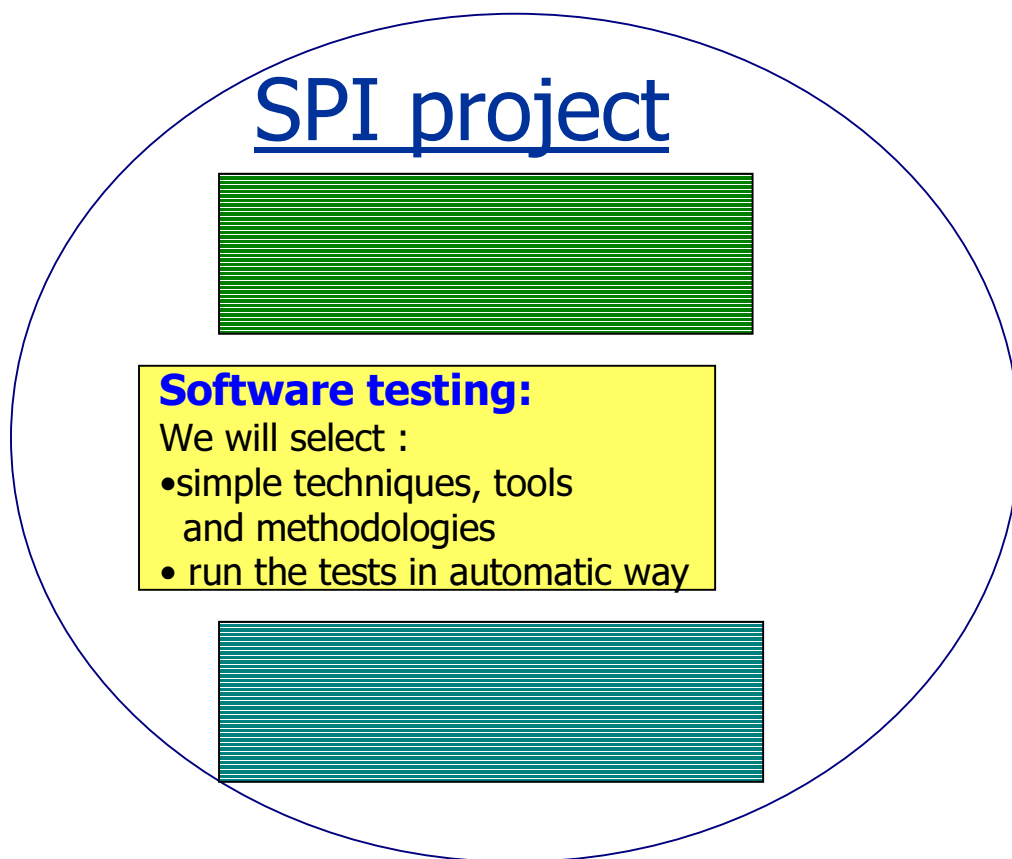


# Overview:



## 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



# Unit Test:

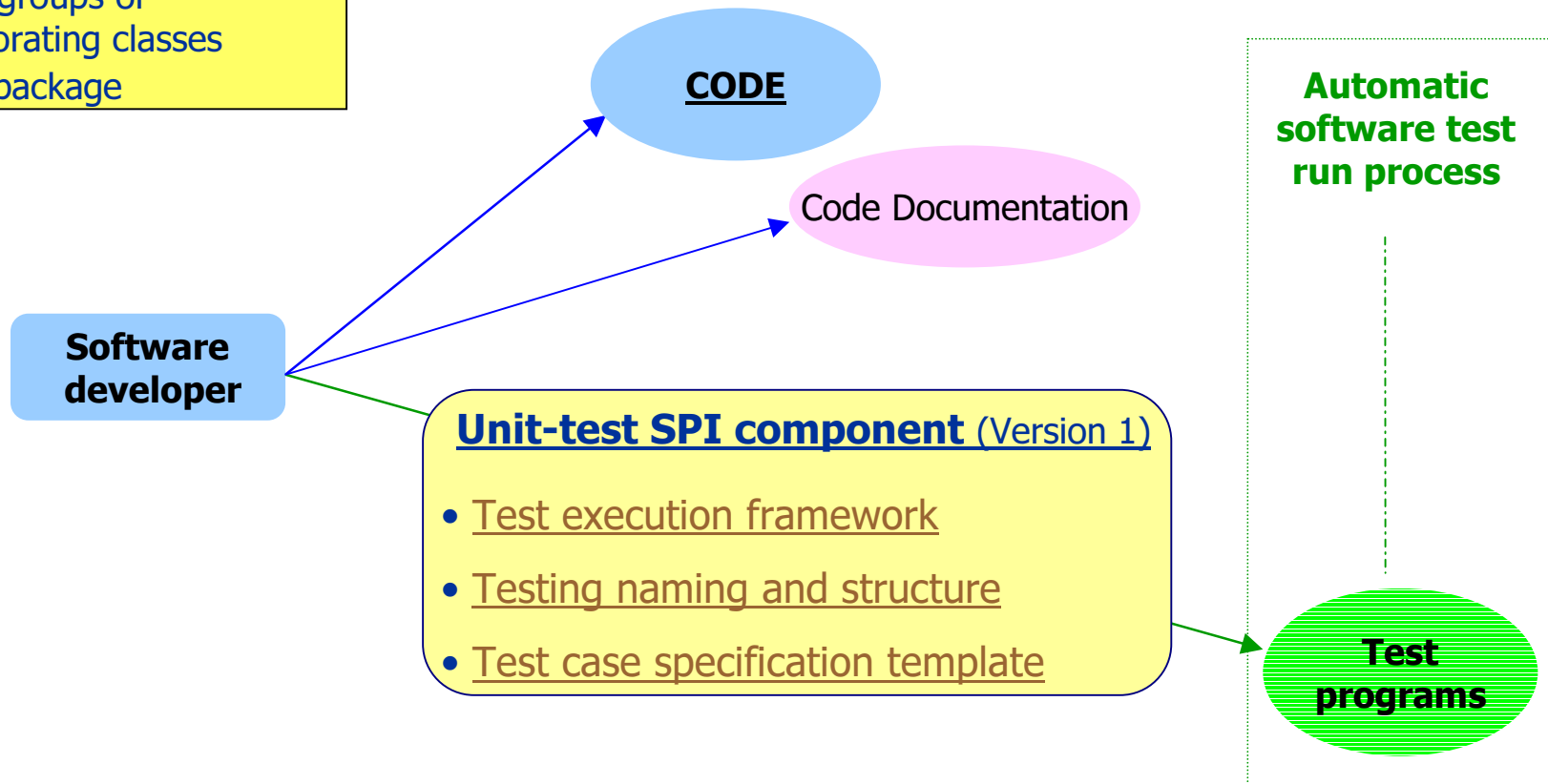


## Unit-tests

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**



# 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.

## **Our inputs:**

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

## **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 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.



# Test frameworks (II): CppUnit




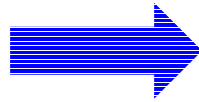
## CppUnit:

Like JUnit but for C++

<http://sourceforge.net/projects/cppunit>  
USED:

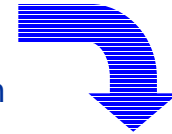
- in eXtreme Programming (XP)
- proposed for DataGrid

- 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



```
xterm <6>

!!!FAILURES!!!
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
=====
make[1]: Leaving directory `/home/user/mgallas/CppUnit-examples/newtest'
lpcitapi31] ~/CppUnit-examples/newtest > 
```



# Test frameworks (II<sub>cont</sub>): CppUnit



```
xterm <3>
#include<iostream.h>
#include<Complex.h>
#include<cppunit/TestCase.h>

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;
}

~
"ComplexTestCase.cpp" 25L, 683C
```



# Test frameworks (III): Oval



**Oval:**

- validation and regression
- used in CMS

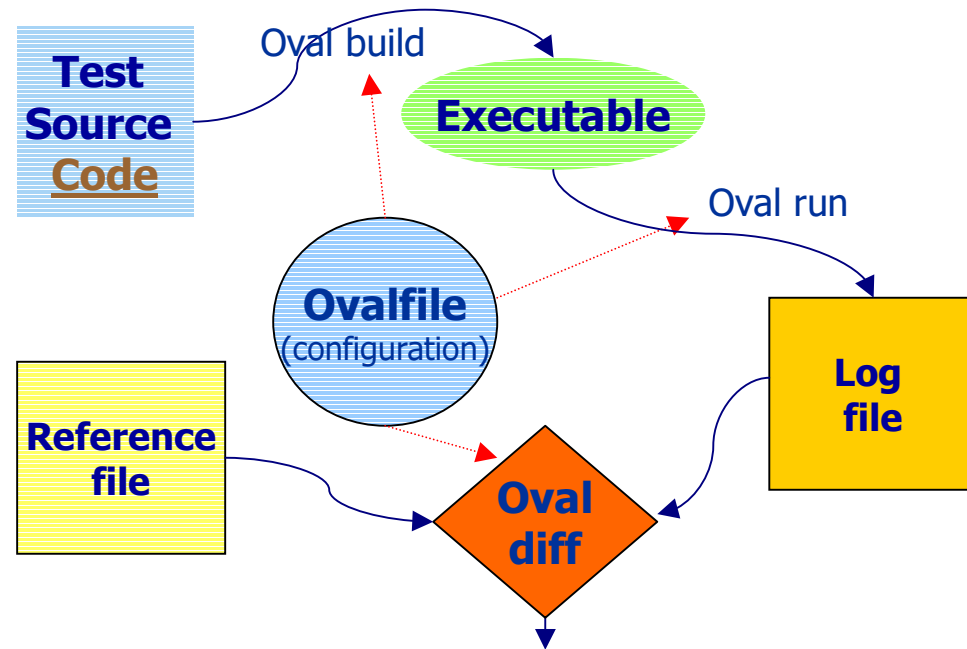
- Can be use for Unit-test.
- It is possible to set different run environments.
- Can run external scripts and external binaries.
- Authors modified it to run it with or without CMS specific environment (SCRAM ...)

*(Thanks to David Chamont for the changes)*

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



M. Gallas IT-API



```

xterm <6>
[pcitapi31] ~/OVAL-examples/subtest1 > oval p
Prog1: build, run, diff.
Prog2: build, run, diff (DIFFS).

xterm <6>
[pcitapi31] ~/OVAL-examples/subtest1 > oval diff Prog2
===== diff =====
(DIFFS)
=== log #29 !~ ref #65 (>15%)
log: [OVAL] result = 1.2
---
ref: [OVAL] result = 1
=====
[pcitapi31] ~/OVAL-examples/subtest1 >
  
```

LCG SPI project: testing





# Test frameworks (III<sub>cont</sub>): Oval



```
xterm <3>
#include <iostream.h>
#include <stdlib.h>

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

xterm <3>
[oval build] =====
[oval build] instruction: g++ Prog1.cpp -o Prog1
[oval build] eval `oval runtime -csh for_diff`
[oval build] INPUT = ok
[oval build] =====

g++ Prog1.cpp -o Prog1

[oval run] =====
[oval run] arguments:
[oval run] USER : mgallas
[oval run] HOST : pcitapi31
[oval run] eval `oval runtime -csh for diff`

xterm <3>
<watcher mail=Manuel.Gallas.Torreira@cern.ch>
<instructions build="g++ %s.cc -o %s" cshruntime="echo" shruntime="echo">
<runtime name="INPUT" value="ok">
<diffline regexpr="^\[OVAL]">

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

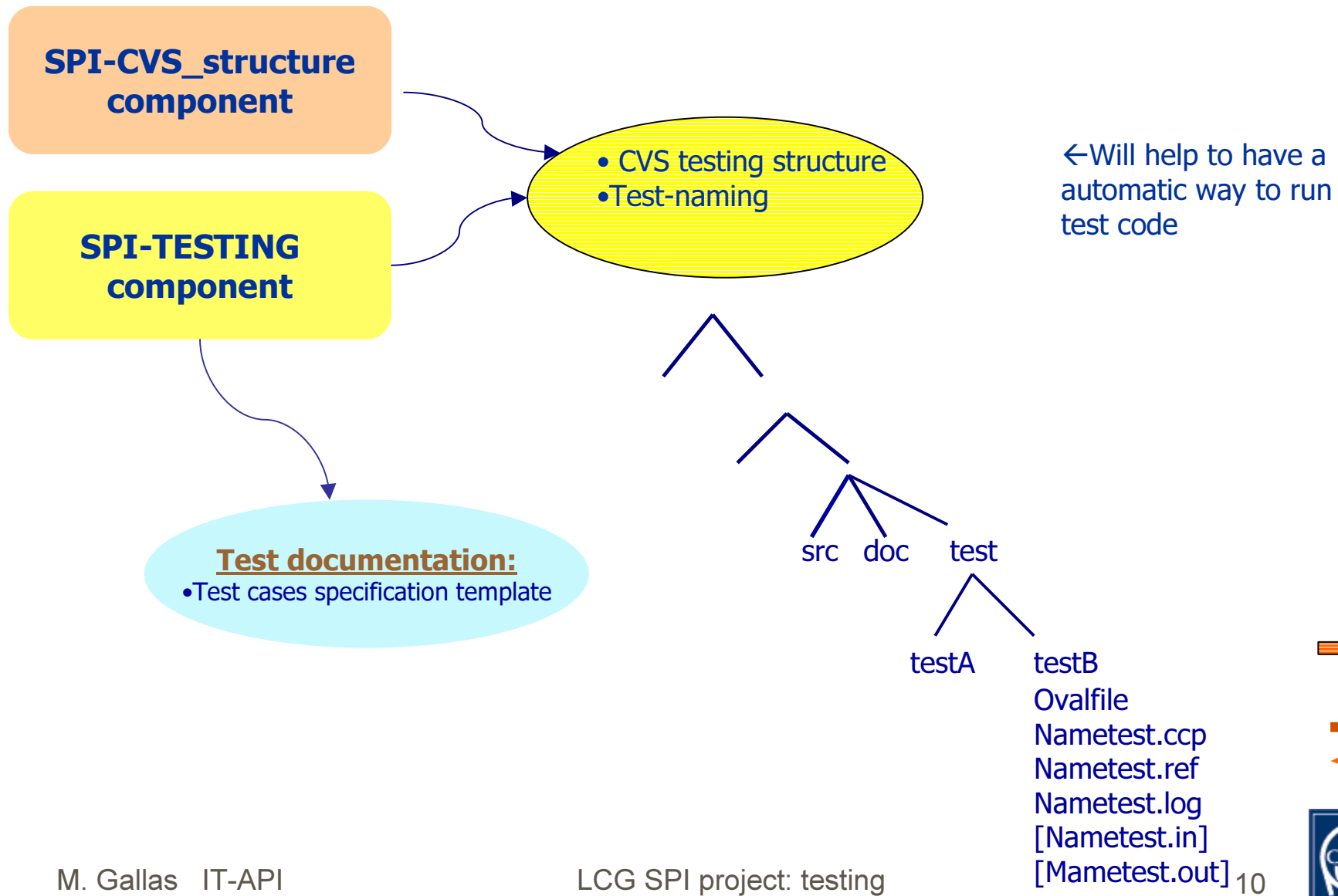
[oval diff] =====
[oval diff] diff line: /^(\[label\] input.*)$/
[oval diff] diff number: /^(\[label\] result = (.*)$/ ~ 15%
[oval diff] =====

=== log #24 != ref
log: [label] input ok
=== log #25 !~ ref (>15%)
log: [label] result = 1.1
"Prog1.ref" line 1 of 36 --2%-- col 1
```





# Unit-test: structure and documentation



# Unit-test: documentation



**LCG-Application Area:** Infrastructure template

**Title**

This document was generated by the POOL project to describe a specific set of steps and data along with the expected results.

Document Name: DataSvc Test Cases Specifications  
 Short Name: `<Component Name>-test`  
 Version: `<version number>`  
 Publication Date: `<mm/dd/yy>`  
 Author(s): Giacomo Govi  
 e-mail(s) contacts: Giacomo.Govi@cern.ch  
 Status: In progress

(\* - short name is a label for a cross-referencing the component name, version and revision information)

Test Suite ref.	TestCaseName	Status	Applicability	Short comment
DataSvc-suite-001	<a href="#">DataSvc1-Test</a>	Ready-Approved	v1r0	example with Oval
	<a href="#">DataSvc2-Test</a>	Ready-Approved	v1r0	example with Cpp Unit
<code>&lt;ComponentName&gt;-suite-002</code>				

DataSvc1-Test	Component	Author	Date	Version
Class DataSvc1-Test	DataSvc	G. Govi	10-20-02	1.0

**Test Case Description**

The goal is to test the DataSvc service in stand alone mode. The two functionalities tested are:

- Retrieve an object from the repository.
- Retrieve the token of a given object.

**Prerequisites and Adaptations to Standard Test Execution Environment**

Persistence using a dummy persistence service.

**Input/Output Specifications**

Use of a Oval reference file named DataSvc1-Test.ref. No differences between log (DataSvc1-Test.log) and reference file (DataSvc1-Test.ref) are expected.

**Test Procedure**

A OvalFile to drive the test is needed. It contains the right running environment. The Ovalfile, shared with another testcases of the same component, is in the test directory. "Oval p" command on the appropriate directory will build, run and establish the differences between the ref and log file.

**Expected results**

No differences between reference and log are expected.

**Failure Recovery**



# 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

## **Thanks to:**

- contacts within the experiments
- LCG-POOL team
- D. Chamont (Oval)

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

**Feedback and interaction are always welcome!!**

