



SW-Testing: SPI policies and support

LCG Software Process & Infrastructure

Manuel.Gallas@cern.ch

LCG/SPI

LCG AA meeting (07/05/03)

SPI SW-Testing: overview



GOAL

"Software testing should be an integral part of the software development process"



Automated testing

SW- testing support:

- Test frame works
- HowTo, examples ...
- Test plan & test case documentation templates

SW- testing polices

S
P
I

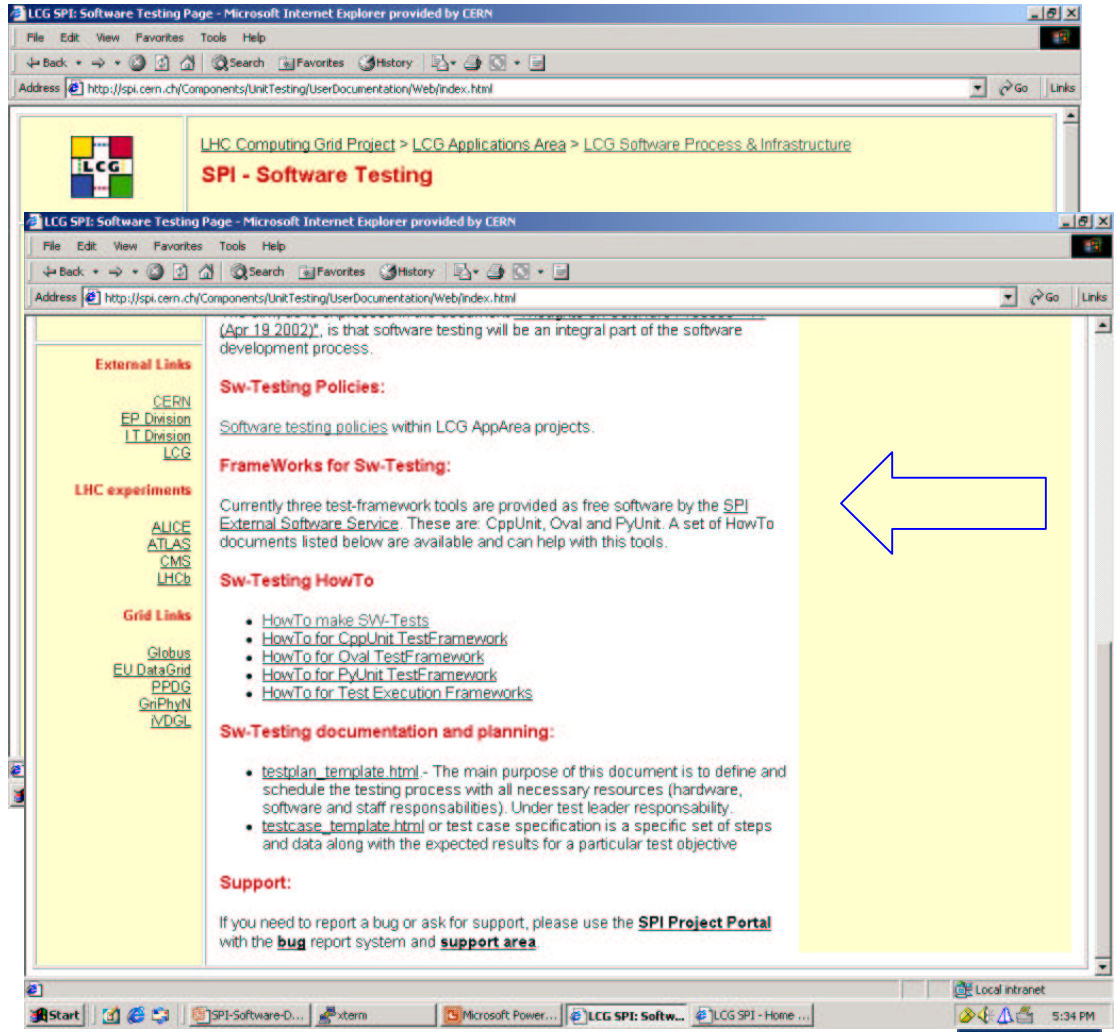
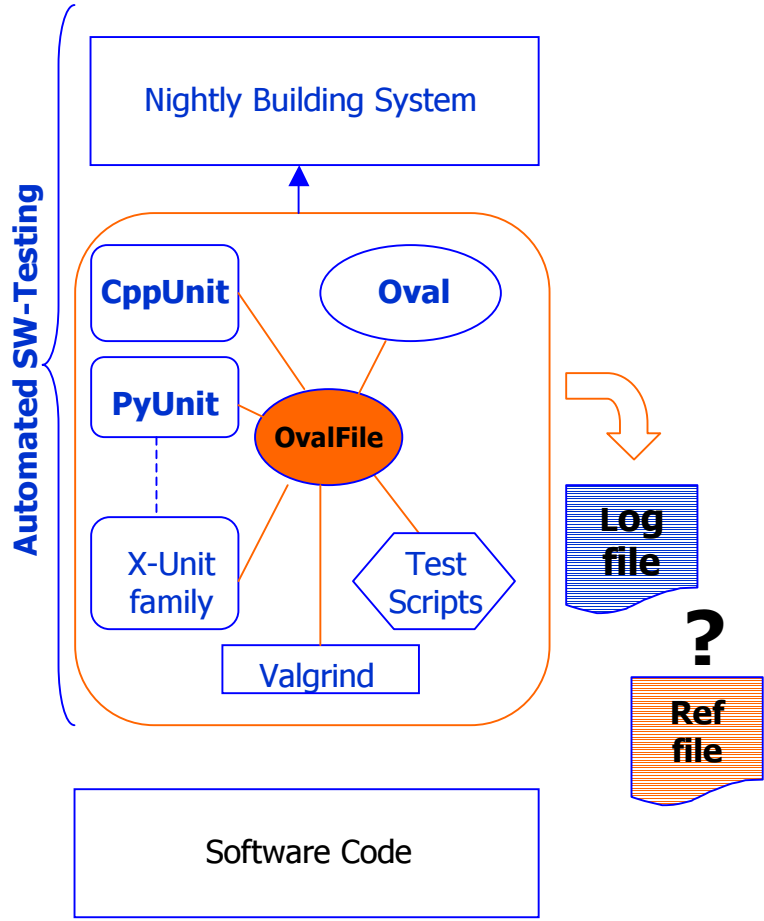
The screenshot shows the SPI website interface. The main content area is divided into several columns:

- SPI Quick Links:** SPI Home, SPI Index, Savannah Portal, LCG App. Area (Home Page, LCG Agenda), MATH Project, PL Project, POOL Project, SEAL Project, SPI Project.
- Index Page:** Infrastructure, CVS Service, Project Portal, Software Library, LCG External Software service, Workbook, SPI Templates.
- Software Development:** Policy and Tools Page (Setting up environment, CVS Directory Policy, Build Directory Policy, FAQ), SCRAM web site, **Testing** (circled in blue), Software Testing in LCG App.Area, Coding conventions, Code documentation.
- App. Area Projects:** POOL Project, SEAL Project, PI Projects.
- LCG Application Area:** News, Overview, RTAGs, Mailing lists, Meetings Agenda, Architecture Blueprint.

The page is updated on 06 May 2003 16:21. The browser window title is "LCG SPI - Home Page - Microsoft Internet Explorer provided by CERN".



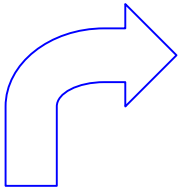
SW-Testing support: Web information



SW-Testing support: Tools supported



Provided as free software by the
[SPI External Software Service](http://spi.cern.ch/extsoft/)



CppUnit

Oval

PyUnit

- Different platforms:
(Linux/Solaris/Windows)
- Different compilers
- Integrated with SCRAM
and ready to be used
- Installed in AFS/DFS
- Instructions to install
them in a local machine

How to use them ?

M Gallas
CERN EP-SFT

The screenshot shows a Microsoft Internet Explorer browser window displaying the SPI External Software Service page for CppUnit. The browser address bar shows the URL <http://spi.cern.ch/extsoft/cppunit.html>. The page content includes a navigation menu with links for Home, News, How to, Contact us, and Search. The main content area is titled "CppUnit" and "The C++ Unit Test Library". It provides a description: "The C++ port of the famous JUnit framework for unit testing. Test output is in XML or text format for automatic testing and GUI based for supervised tests." It also lists availability paths for different compilers and platforms, and documentation links.



SW-Testing support: HowTos



1 Support:

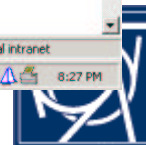
2 HowTo for CppUnit TestFramework

3 Dedicate time to make tests

4 Follow sw-testing policies

M Gallas
CERN EP-SFT

SPI: SW-Testing



SW-Testing support: Examples



SEAL

seal/Foundation/SealKernel/tests - Microsoft Internet Explorer provided by CERN

Address: <http://lcgapp.cern.ch/cgi-bin/viewcvs/viewcvs.cgi/seal/Foundation/SealKernel/tests/?cvsroot=SEAL>

LHC Computing Grid > LCG Applications Area > LCG Software Process & Infrastructure

seal/Foundation/SealKernel/tests

Powered by **APACHE** ViewCVS and CVS Help

Current directory: [SEAL] / seal / Foundation / SealKernel / tests

Files shown: 2

File	Rev.	Age	Author	Last log entry
Exception/				
MessageSvc/				
TestMessageSvc/				
BuildFile	1.2	4 weeks	rado	Updated OVAL files after discussion with Manuel. Corrected SPI unit test names
OvalFile	1.1	4 weeks	rado	Updated OVAL files after discussion with Manuel. Corrected SPI unit test names

Show files using tag: - Non-branch tags -

Show

seal/Foundation/SealKernel/tests/MessageSvc - Microsoft Internet Explorer provided by CERN

Address: <http://lcgapp.cern.ch/cgi-bin/viewcvs/viewcvs.cgi/seal/Foundation/SealKernel/tests/MessageSvc/?cvsroot=SEAL>

LHC Computing Grid > LCG Applications Area > LCG Software Process & Infrastructure

seal/Foundation/SealKernel/tests/MessageSvc

Powered by **APACHE** ViewCVS and CVS Help

Current directory: [SEAL] / seal / Foundation / SealKernel / tests / MessageSvc

Files shown: 3

File	Rev.	Age	Author	Last log entry
BuildFile	1.2	4 weeks	rado	Updated OVAL files after discussion with Manuel. Corrected SPI unit test names
MessageSvcCppUnitSuite.cpp	1.1.1.1	5 weeks	rado	Imported seal::MessageSvc CppUnit test.
OvalFile	1.1	4 weeks	rado	Updated OVAL files after discussion with Manuel. Corrected SPI unit test names

Show files using tag: - Non-branch tags -

Show

Powered by **APACHE** ViewCVS and CVS Help

SPI support

Powered by ViewCVS 0.9.2



SW-Testing support: Examples (II)



POOL

LHC Computing Grid > LCG Applications Area > LCG Software Process & Infrastructure

pool/Tests/FileCatalog_Functionality

Current directory: [POOL] / pool / Tests / FileCatalog_Functionality
Files shown: 5

File	Rev.	Age	Author	Last log entry
Attic/ [show contents]				
src/				
BuildFile	1.3	14 hours	xiezhen	removed dependency on sub catalogs
OvalFile	1.2	3 weeks	girone	import a cppunit test
pool.env	1.2	14 hours	xiezhen	new contact string
test_FileCatalog_Functionality.log	1.1	3 weeks	girone	import a cppunit test
test_FileCatalog_Functionality.ref	1.1	3 weeks	girone	import a cppunit test

Show files using tag: Show



SW-Testing policies: where to find them?



http://spi.cern.ch/software_development.html

Software Testing Policies:

- Design phase:
 - Perform a [Testing Planning](#)
- Code phase:
 - Populate, document and execute [Unit Test](#) cases
 - Populate and document [Integration-Tests](#) and System-Tests
- Test phase:
 - Conduct unit, integration and systems tests
 - Perform regression testing as needed
 - Conduct acceptance tests (alfa tests)

Software Testing Resources:

- [Test frameworks](#)
- [Test frameworks howto](#)
- [Test doc templates](#)
- [Sw-Testing Web](#)
- [SPI Web](#)
- [Savannah SPI Portal](#)

Document Status Sheet

The following information is being used to control and track modifications made to this document. It is the reader's responsibility to ensure they have the latest version of this document placed at [SPI Software Development Policies](#) page.

Version	Date	Author	section(s)	reason
1.1	29/04/03	M. Gallas	Initial version: - Unit and Integration testing already accepted - Test Planning proposal	



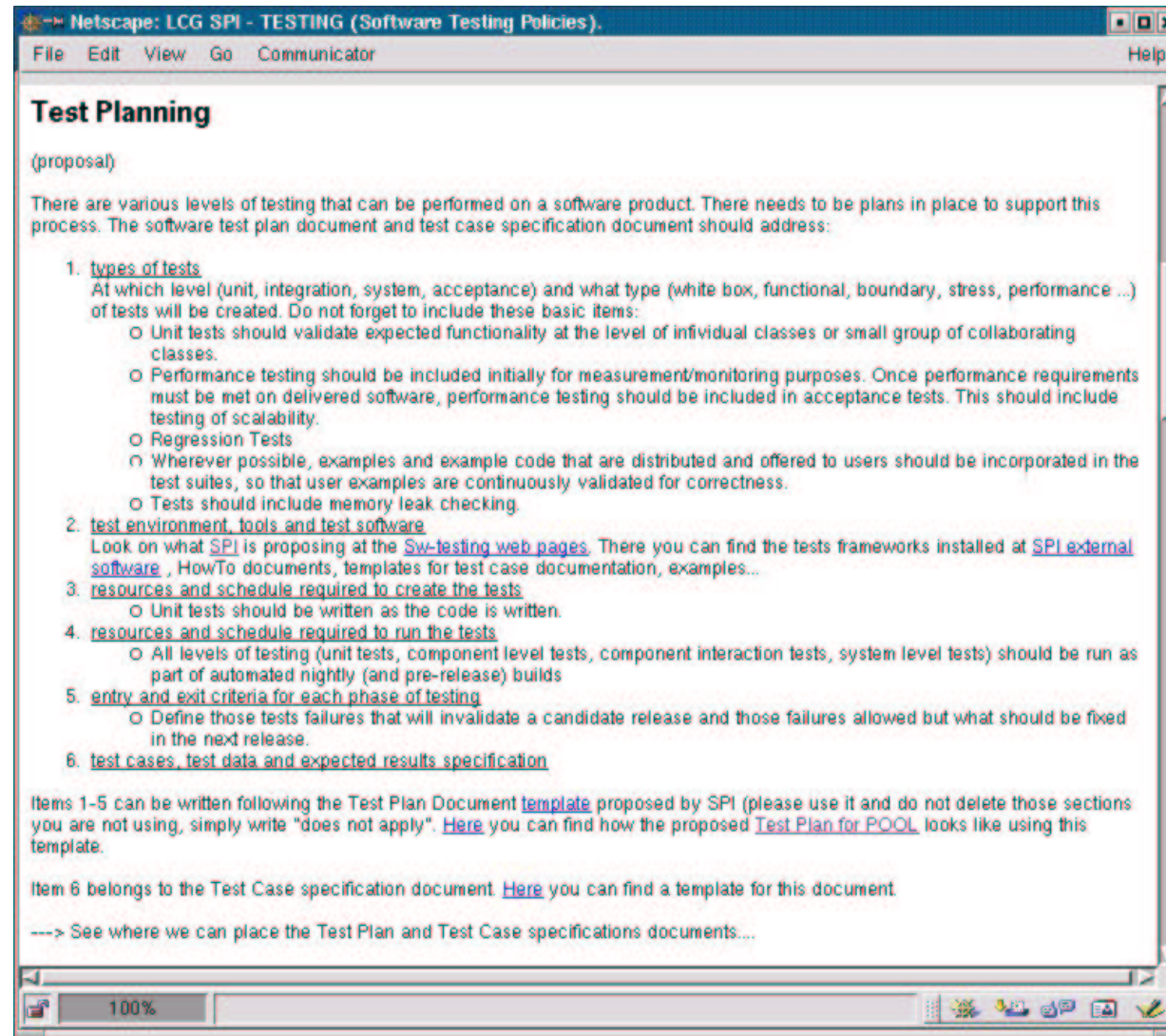
SW-Testing policies: Test Plan



For all projects is needed a Test Plan

Let us use a template

POOL Test Plan is a good example and a starting point



SW-Testing policies: for developers



Unit Tests
(proposal already accepted)

The Unit testing or component unit has been correctly implemented.

1. All the unit tests corresponding to a component should be placed in a subdirectory. If it is not created, it should be created under the **tests** directory. The name of the subdirectory should hint the test cases which are stored into them.
2. Under **tests** directory you have to write a "OvalFile" in which you should specify the tests (written in Oval, CppUnit, PyUnit or your own scripts) you would like to run automatically, the order in which they will be run and the environment and arguments needed for each of them. Remember that all your test should be run by Oval.
3. For write the unit-tests you can use: [CppUnit](#), [Oval](#) or [PyUnit](#). You can follow the specific howto at the [SW-testing Web page](#) to get help. You must follow all the conventions written in there.
4. For each test subdirectory you have to write a "OvalFile" in which you should specify the tests (written in Oval, CppUnit, PyUnit or your own scripts) you would like to run automatically, the order in which they will be run and the environment and arguments needed for each of them. Remember that all your test should be run by Oval. Under "tests" directory you have to place a "OvalFile" which describes the active test subdirectories. The content of this file is as simply as follow:


```
<dir name="test1"> My test suite number 1, intended for... </dir>
<dir name="test2"> My test suite number 2, intended for... </dir>
.....
```
5. For each specific test Oval will need a reference file **test_Package_X_test1.ref** which must be placed at the specific test subdirectory and committed to the CVS repository. For CppUnit or PyUnit tests you only need to copy the "test_Package_X_test1.log" file once you have not Cppunit or PyUnit failures in your tests. For these cases the OVAL tag is **[OVAL] Cppunit-result=0** and **[OVAL] PyUnit-result=0**. If you use Oval to produce your tests or a script language is up to you decide which result will be compared by Oval.
6. At this point your "tests" subdirectory structure is like:


```
SPROJECTtop/
  src/
    Package_X/
      tests/
        BuildFile
        OvalFile
        test1/
          OvalFile
          test_Package_X_test1.cpp
          test_Package_X_test1_input.dat
          test_Package_X_test1_ref
```

The "test_Package_X_test1_input.dat" file contains the input fat for your tests if they are needed.

NOTE: For help you to follow this conventions and to start with the testing of your component there is a script [distributeTest.py](#) which makes the needed steps to have running OVAL, CppUnit, TestScripts and PyUnit tests. Execute it in your test subdirectory and you will receive all the needed stuff.

HELP !!



SW-Testing policies: for developers (II)



Integration Tests
(proposal already accepted)

- The integration tests will go into the **Tests** subdirectory under "src". There you can make as many subdirectories as needed, again a good naming is required. Test libraries (useful when one or more classes are shared between different tests) will go into **Tests/Libraries**.

```

$PROJECTtop/
  src/
    Examples/
      Libraries/
        ExampleLib1/ <-----
      Example1/ <-----
    Tests/
      Libraries/
        TestLib1/ <-----
      Test1/
        BuildFile
        OvalFile
        test_Test1.ref
        test_Test1_input.data
        src/
          test_Test1.cpp
    Package_X/
      tests
      dict/
      src/
      utilities/
      doc/
      scripts/
      data/
    utilities/
      utilityset1/ <-----
  $PLATFORM/
    bin/
    data/
    lib/
    libPackage_X.so <-----
    libPackage_XDict.so <-----
    tests/
      bin/
        test_Package_X_test1 <-----
        test_Test1 <-----
      lib/
        libTestLib1.so <-----
    examples/
      bin/
        Example1 <-----
      lib/
        libExampleLib1.so <-----
  
```

<<< suggested for unit tests
 <<< suggested for integration tests

- In each test or lib subdirectory you have to create a "BuildFile" which defines the dependencies to the relevant packages.
- In each test directory you have to create a "OvalFile" to run automatically your tests. The needed reference file should be placed at the same level and not inside the "src" directory where the sources of the tests are.
- If input data is needed for you tests, it must be at the same level as the BuildFile, inside the specific test directory.



SPI SW-Testing



Thanks to:

- LCG-POOL team
- LCG-SEAL team

SPI SW-Testing

SW- testing support:

- Test frame works
- HowTo, examples ...
- Test plan & test case documentation templates

SW- testing polices

Feedback and interaction are always welcome!!

