QMTest
Status and New features

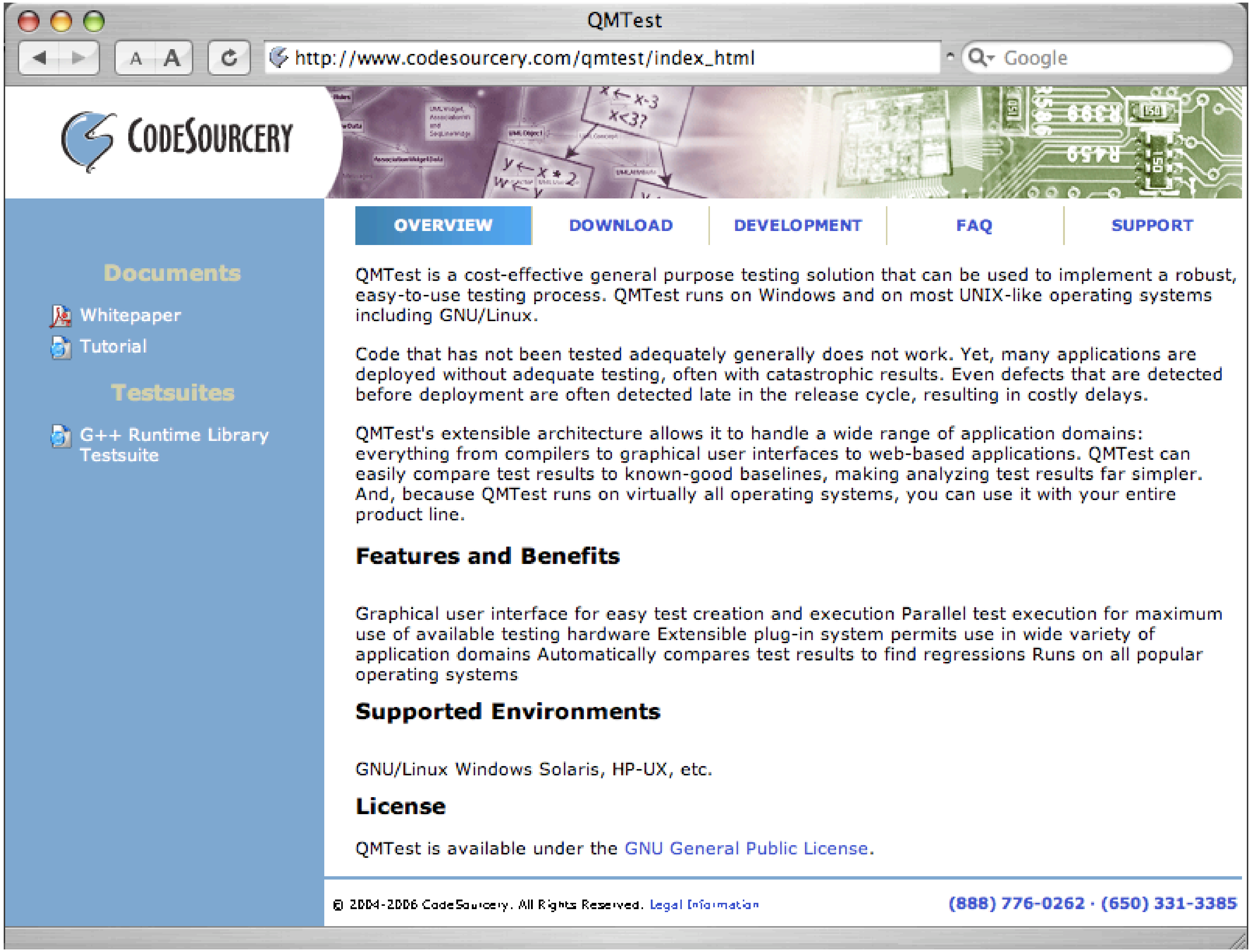
Applications Area Meeting
08 November 2006, CERN

M. Gallas (mgallas@mail.cern.ch)
CERN/PH-SFT-SPI

Outline:

- QMTest
- QMTest in SPI
- QMTest test-domain extension example
- Conclusions

QMTest



QMTest is a cost-effective general purpose testing solution that can be used to implement a robust, easy-to-use testing process. QMTest runs on Windows and on most UNIX-like operating systems including GNU/Linux.

Code that has not been tested adequately generally does not work. Yet, many applications are deployed without adequate testing, often with catastrophic results. Even defects that are detected before deployment are often detected late in the release cycle, resulting in costly delays.

QMTest's extensible architecture allows it to handle a wide range of application domains: everything from compilers to graphical user interfaces to web-based applications. QMTest can easily compare test results to known-good baselines, making analyzing test results far simpler. And, because QMTest runs on virtually all operating systems, you can use it with your entire product line.

Features and Benefits

Graphical user interface for easy test creation and execution Parallel test execution for maximum use of available testing hardware Extensible plug-in system permits use in wide variety of application domains Automatically compares test results to find regressions Runs on all popular operating systems

Supported Environments

GNU/Linux Windows Solaris, HP-UX, etc.

License

QMTest is available under the GNU General Public License.

© 2004-2006 CodeSourcery. All Rights Reserved. Legal Information

(888) 776-0262 · (650) 331-3385

QMTest

- **QMTest** stands for "Quality management" tests
- It is (open source) test framework in Python from http://www.codesourcery.com/qmtest/
- Started as project on Jan 2002 (qmtest 1.1). Last pub-release on July 2005 (qmtest 2.3)
- **Supported platforms**: linux, macOsX, windows
- **Tests:** can be written in almost any language and can deal with unit, integration, system, acceptance tests.
- **Hierarchy & organization**: tests can be placed in suites and a suite can contain another suites, dependences among tests are possible
- **Interaction**: command line or web browser (IE, Safari, Mozilla, Opera)
- **Encapsulation**: tests can run in own threads, process and hosts
- **Test DB:** xml (default) or any other DB of your choice
- **Test results:** presented in Web pages or saved in a file (inspected in command line)
- **Extensibility:domain,** storage, execution and display (the type of tests, the storage of test, schedule of the execution and display of results)

QMTest: tests (*.qmt)

- Tests inherit the running environment in which qmtest is started. Suitable to be used in a experiment sw-framework ($PATH, $LD_LIBRARY_PATH and other environment variables are there)
- Individual tests can have a set of properties: environment variables, target (platforms,compilers, parallel running), resources (which are run before the test)
- Tests are boolean (return SUCCESS or FAILURE)
- In addition tests can return ERROR (problems in the test execution environment) or UNTESTED (qmtest did not run the test)
- QMtest accepts a previous results file (results.qmr) as expected output.
- Actual test programs (in whatever language), binaries, shell, python scripts and it can run another programs inside shells.
- User can create his own test classes or customize the default ones (see in this talk): command.ShellCommandTest, command.ExecTest, command.ShellScriptTest, file.FileContentsTest, python.Exception, python.ExecTest, python.StringExceptionTest

QMTest: test suites (*.qms)

- Suites are collections of tests; good for grouping and ordering tests
- Suites can be used to provide context variables to a set of tests
- Suites can call other suites. Suites, as tests, can be run individually
- Directories in the QMTest database path (QMTEST_DB_PATH) are treated as automatic suites.
- Test suites and tests are stored in XML format by default (possible automatic generation)
- Tests can depend on other tests (failure in cascade). A test with a prerequisite test is called a dependent test.
- A dependent test is executed only after its prerequisite tests are executed and have the specified outcome.
- Dependent tests --> attempt to diagnose failures in more detail
- User can create new suites with new properties.

QMTest: test targets

- Default QMTest execution engine executes tests sequentially on a single machine but there are more possibilities using "targets"
- Types of targets:
  - Standard -> synchronous
  - Thread -> One test per process
  - Process -> One test per process
  - RemoteShell -> One test per chosen host

- May want to run tests on separate machines
- Or you may want to create your own targets

QMTest: resources

- Resources are external system resources that you don't want to reallocate for each test:
  - DB connections
  - set of input files,
  - etc
- Tests may require common setup and cleanup code. A resource is an object with SetUp and CleanUp methods
- When a test depends on a resource it is guaranteed that the resource's SetUp method will be executed before the test is executed and the CleanUp method will be executed after wards.
- If a resource is share among several tests the scheduler share the resource rather than set the resource each time.

QMTest: test execution & display of results

- Execution either in command line, through web interface or as remote server
- Context can be provided at runtime
- Can select to run any subset of tests or individual tests
- Targets and new resources are not chosen at runtime

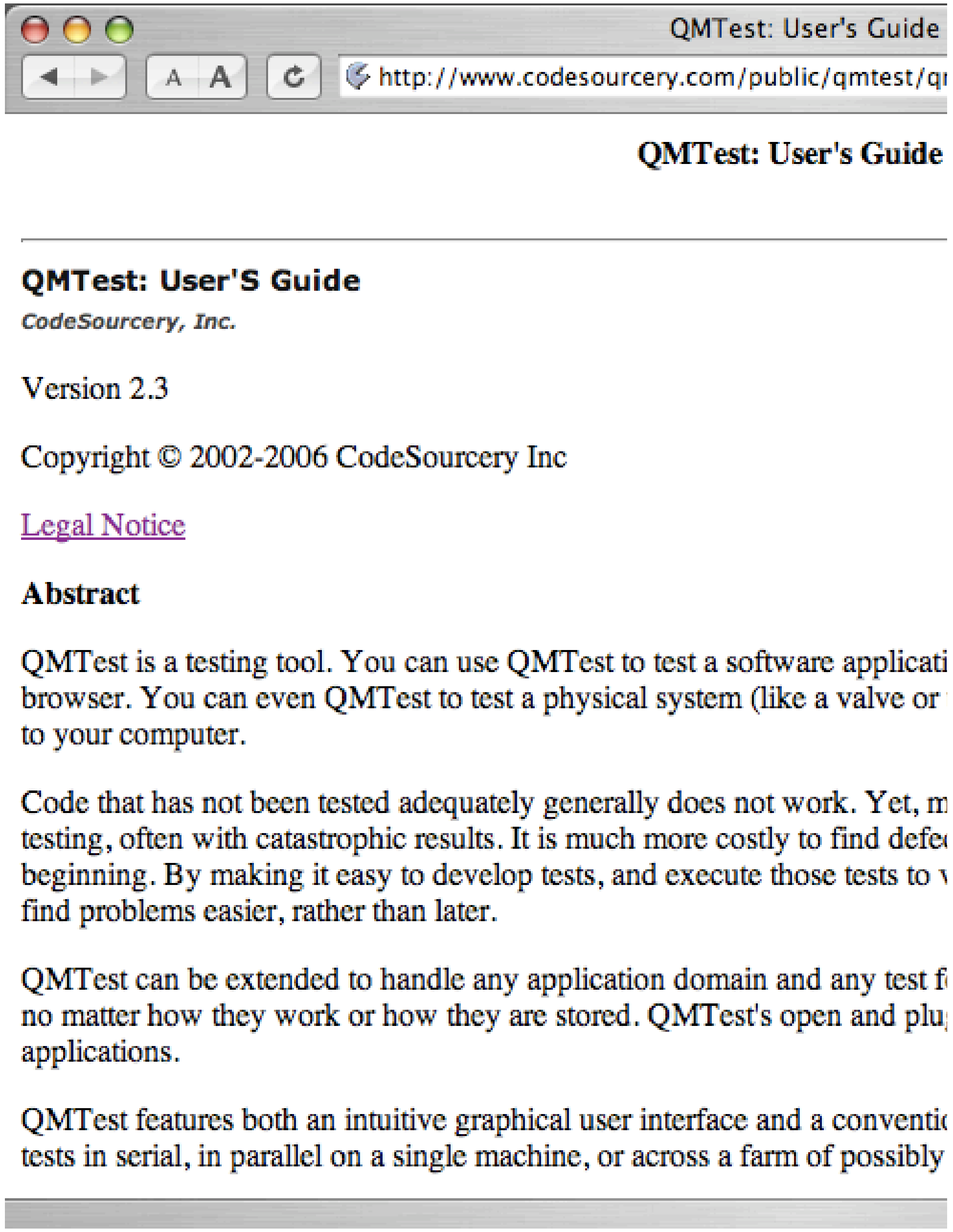
- Results from tests are return codes, error messages and in our case the stdout
- The output result from the tests can be in different formats (stats,brief,full) and is also customizable.
- Since the test results are stored in a file it can be also retrieved (qmtest summarize/qmtest report) and inserted in another views (example ATLAS-NICOS)

QMTest: 2.3 release news

- Requires Python 2.2 or greater
- Needs xml.dom Python module which is not present in SuSE dist (distributed separately)
- On Windows it needs Win32 extensions
- There is now a "qmtest report" command-line option that can be used to create test reports from multiple result files.
- There is a new "host" extension kind (with a built-in set of predefined host types such as 'localhost.LocalHost', 'ssh_host.SSHHost', etc.)
- QMTest has been made robust and more flexible.
- QMTest now runs with Python 2.4

- Improved a lot the documentation (recently, ~ July 2006):
  - API: http://www.codesourcery.com/public/qmtest/qm-snapshot/share/doc/qmtest/html/manual/index.html
  - Tutorial: http://www.codesourcery.com/public/qmtest/qm-snapshot/share/doc/qmtest/html/tutorial/index.html

QMTest: user guide



QMTest: User's Guide

http://www.codesourcery.com/public/qmtest/q

QMTest: User's Guide

QMTest: User'S Guide

CodeSourcery, Inc.

Version 2.3

Copyright © 2002-2006 CodeSourcery Inc

Legal Notice

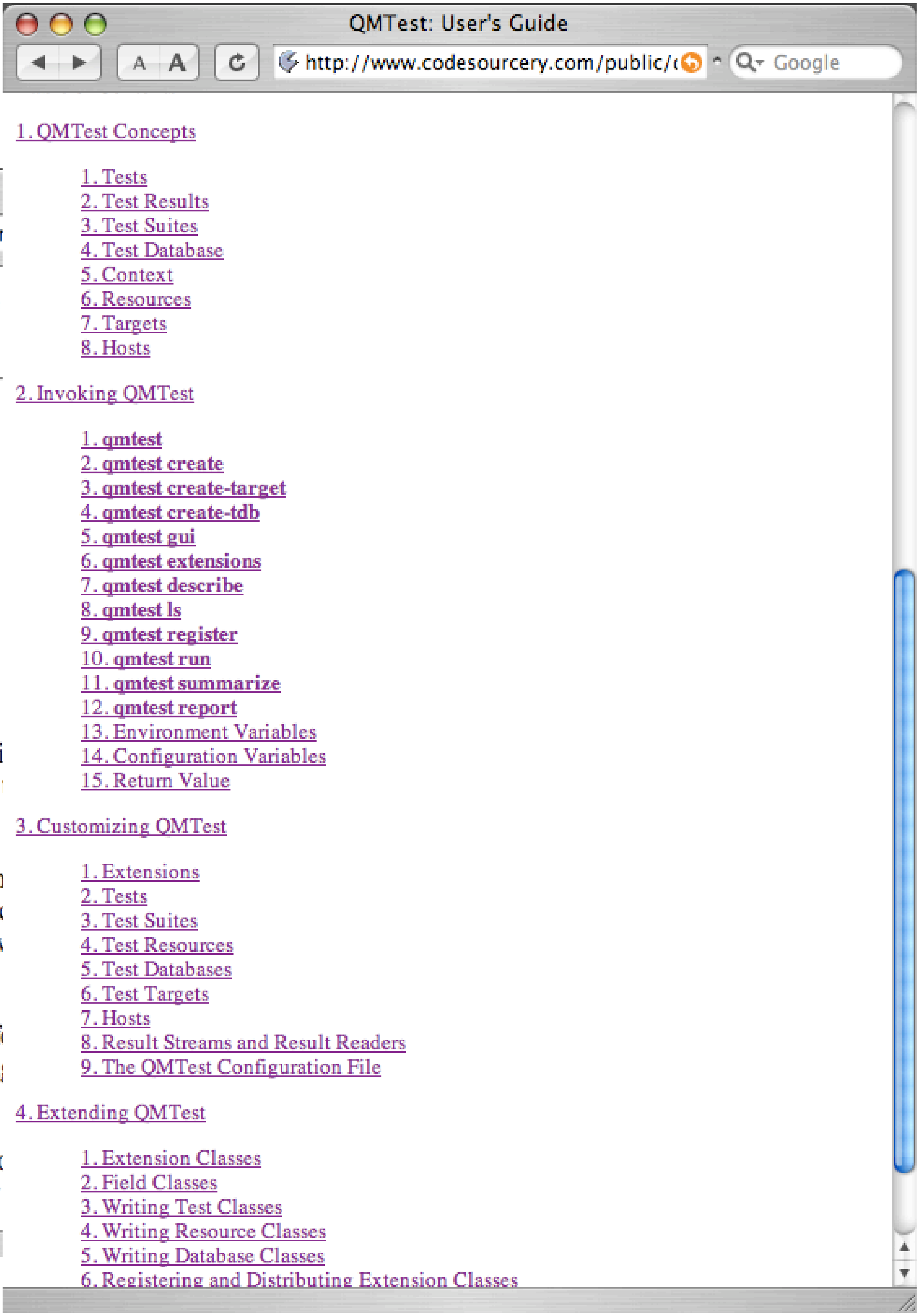
Abstract

QMTest is a testing tool. You can use QMTest to test a software applicati
browser. You can even QMTest to test a physical system (like a valve or
to your computer.

Code that has not been tested adequately generally does not work. Yet, m
testing, often with catastrophic results. It is much more costly to find defe
beginning. By making it easy to develop tests, and execute those tests to
find problems easier, rather than later.

QMTest can be extended to handle any application domain and any test f
no matter how they work or how they are stored. QMTest's open and plu
applications.

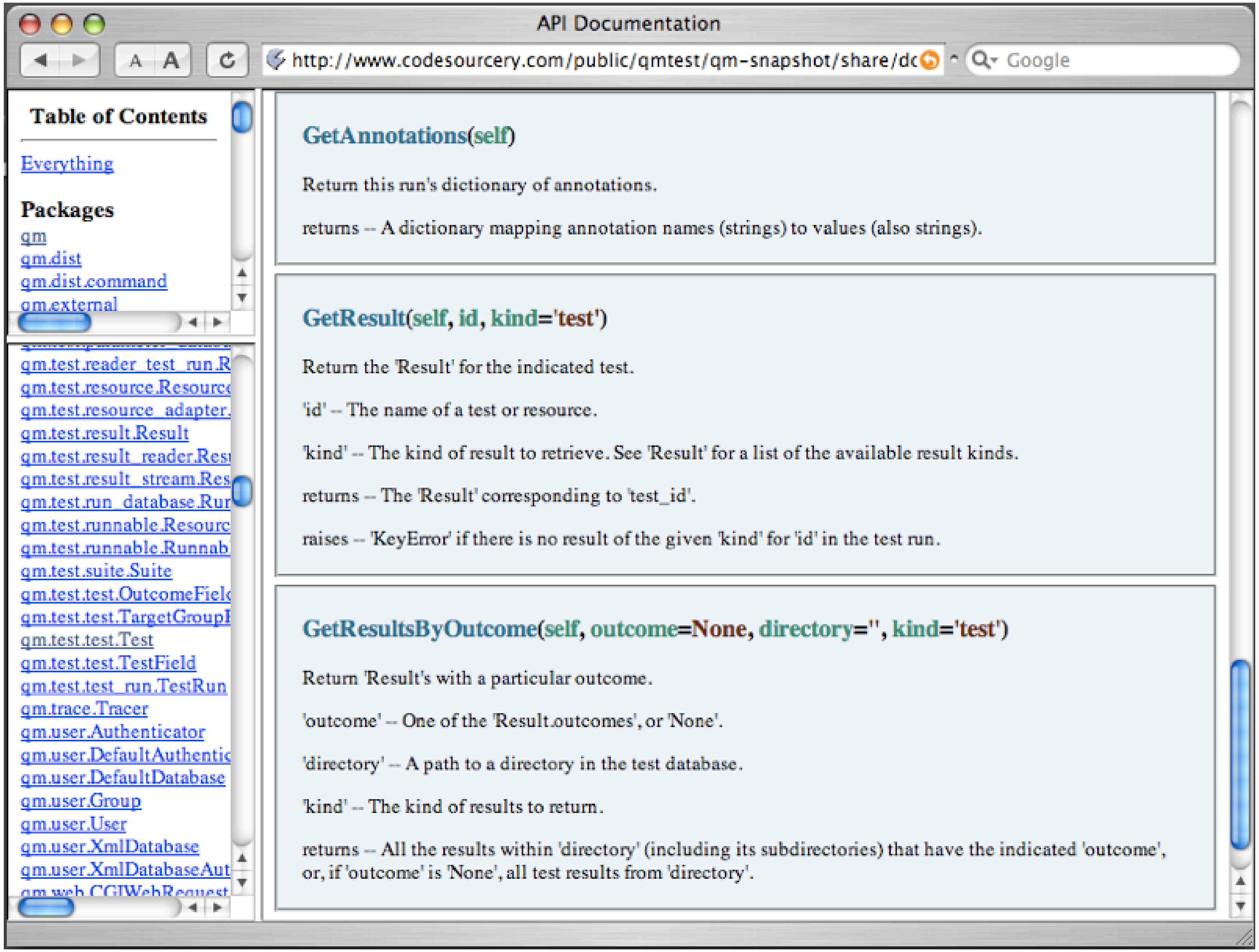
QMTest features both an intuitive graphical user interface and a conventi
tests in serial, in parallel on a single machine, or across a farm of possibly



QMTest: User's Guide

http://www.codesourcery.com/public/

QMTest: api-doc (developers and user-extenders)


API Documentation
http://www.codesourcery.com/public/qmtest/qm-snapshot/share/dc
Google

GetAnnotations(self)

Return this run's dictionary of annotations.

returns -- A dictionary mapping annotation names (strings) to values (also strings).

GetResult(self, id, kind='test')

Return the 'Result' for the indicated test.

'id' -- The name of a test or resource.

'kind' -- The kind of result to retrieve. See 'Result' for a list of the available result kinds.

returns -- The 'Result' corresponding to 'test_id'.

raises -- 'KeyError' if there is no result of the given 'kind' for 'id' in the test run.

GetResultsByOutcome(self, outcome=None, directory='', kind='test')

Return 'Result's with a particular outcome.

'outcome' -- One of the 'Result.outcomes', or 'None'.

'directory' -- A path to a directory in the test database.

'kind' -- The kind of results to return.

returns -- All the results within 'directory' (including its subdirectories) that have the indicated 'outcome', or, if 'outcome' is 'None', all test results from 'directory'.

QMTest in SPI

- Working since 2003 within the LCG-I projects and later for ATLAS NICOS-tests and EGEE.

- Installed versions : 2.0.3, 2.2.1, 2.3.0 (since LCG_42)

- Supported platforms (linux/macOsX/windows):

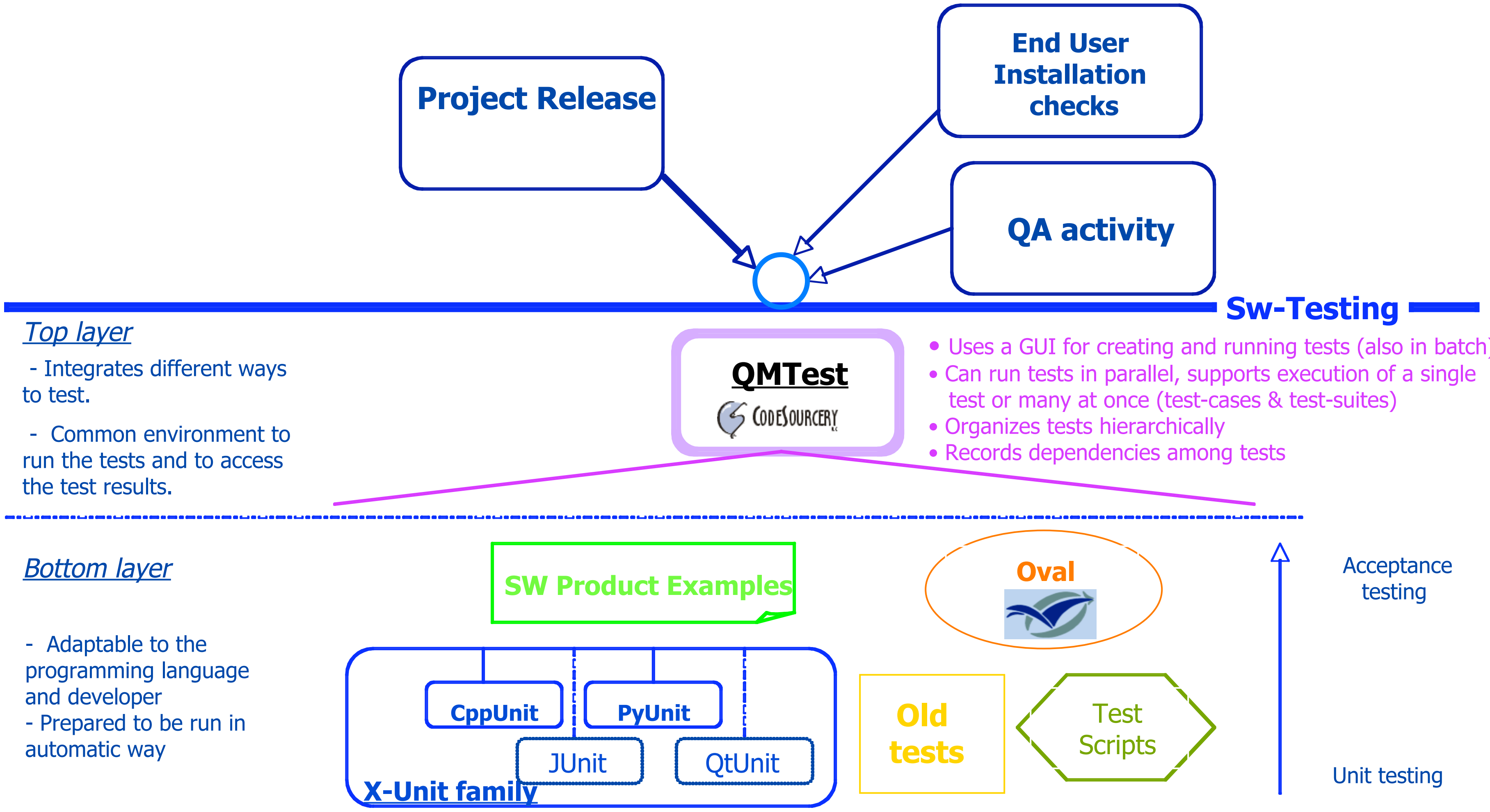
  osx104_ppc_gcc401 slc3_ia32_gcc323 slc4_amd64_gcc345 slc3_amd64_gcc344
  slc3_ia32_gcc344 slc4_ia32_gcc34 slc4_ia32_gcc346 slc3_gcc323
  slc4_amd64_gcc34 slc4_ia32_gcc345 win32_vc71

- As it is used today for the above mentioned projects a very small part of the QMTest functionality is used.

- We use the test-class **ShellCommandTest** wich runs a shell command and checks the stderr and exit code. As it is show in the next slide it used as framework which glues other testing pieces.

- SPI has a set of QMTest tests which are testing the expected and present QMTest used functionality (stderr, exit code, time-limit, pyunit integration).

QMTest in SPI: (general picture)



Top layer

- Integrates different ways to test.
- Common environment to run the tests and to access the test results.

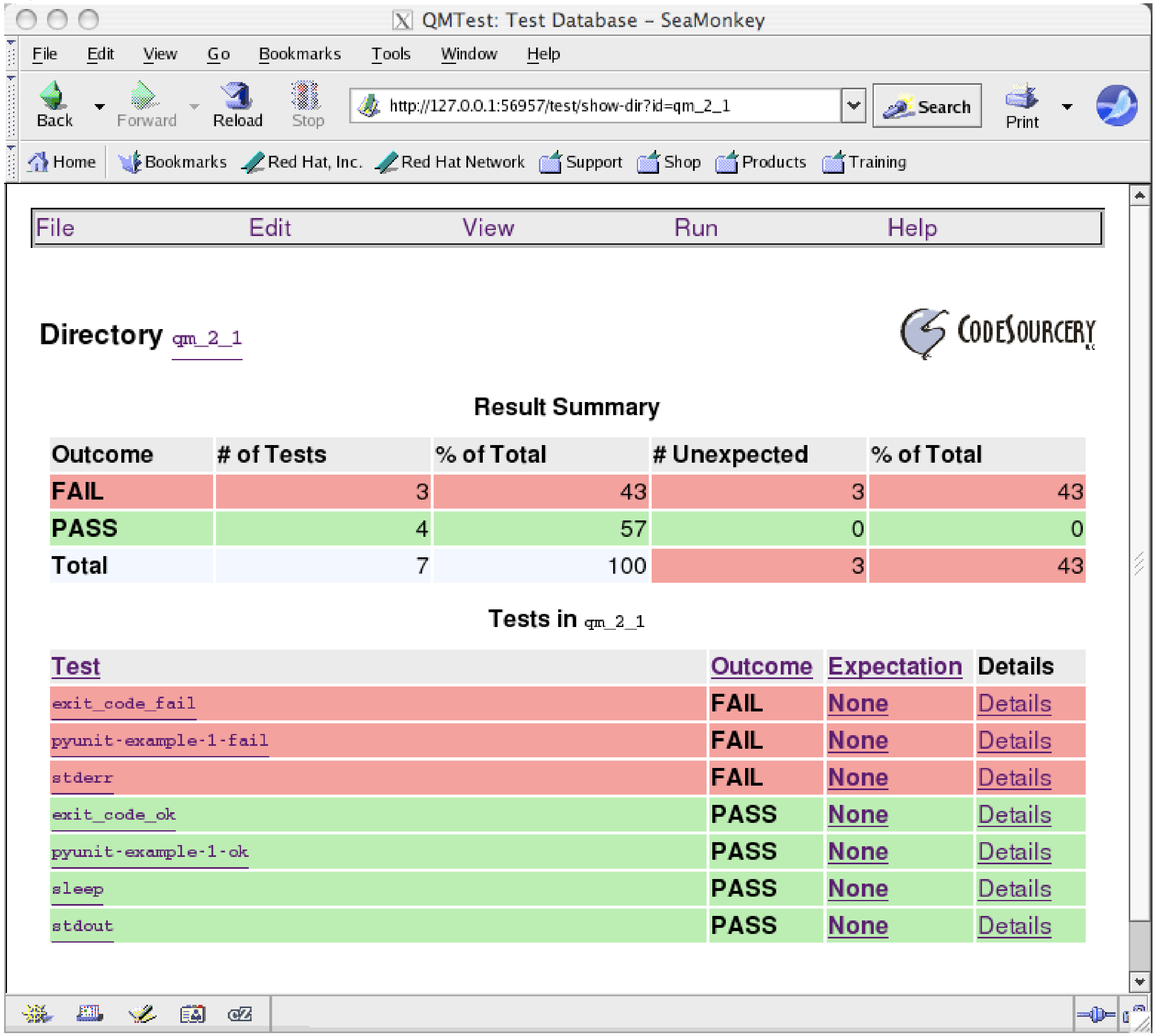
- Uses a GUI for creating and running tests (also in batch).
- Can run tests in parallel, supports execution of a single test or many at once (test-cases & test-suites)
- Organizes tests hierarchically
- Records dependencies among tests

Bottom layer

- Adaptable to the programming language and developer
- Prepared to be run in automatic way

QMTest in SPI (qmtest checks)

- Basic functionality we are using from qmtest.
- These are a set of test we have used when we migrated from qmtest_2_1 to qmtest_2_3
- These tests are used today to check the SPI installations in different platforms.



QMTest: Test Database - SeaMonkey
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop Search Print
http://127.0.0.1:56957/test/show-dir?id=qm_2_1
Home Bookmarks Red Hat, Inc. Red Hat Network Support Shop Products Training
File Edit View Run Help
CODESOURCERY

Directory qm_2_1

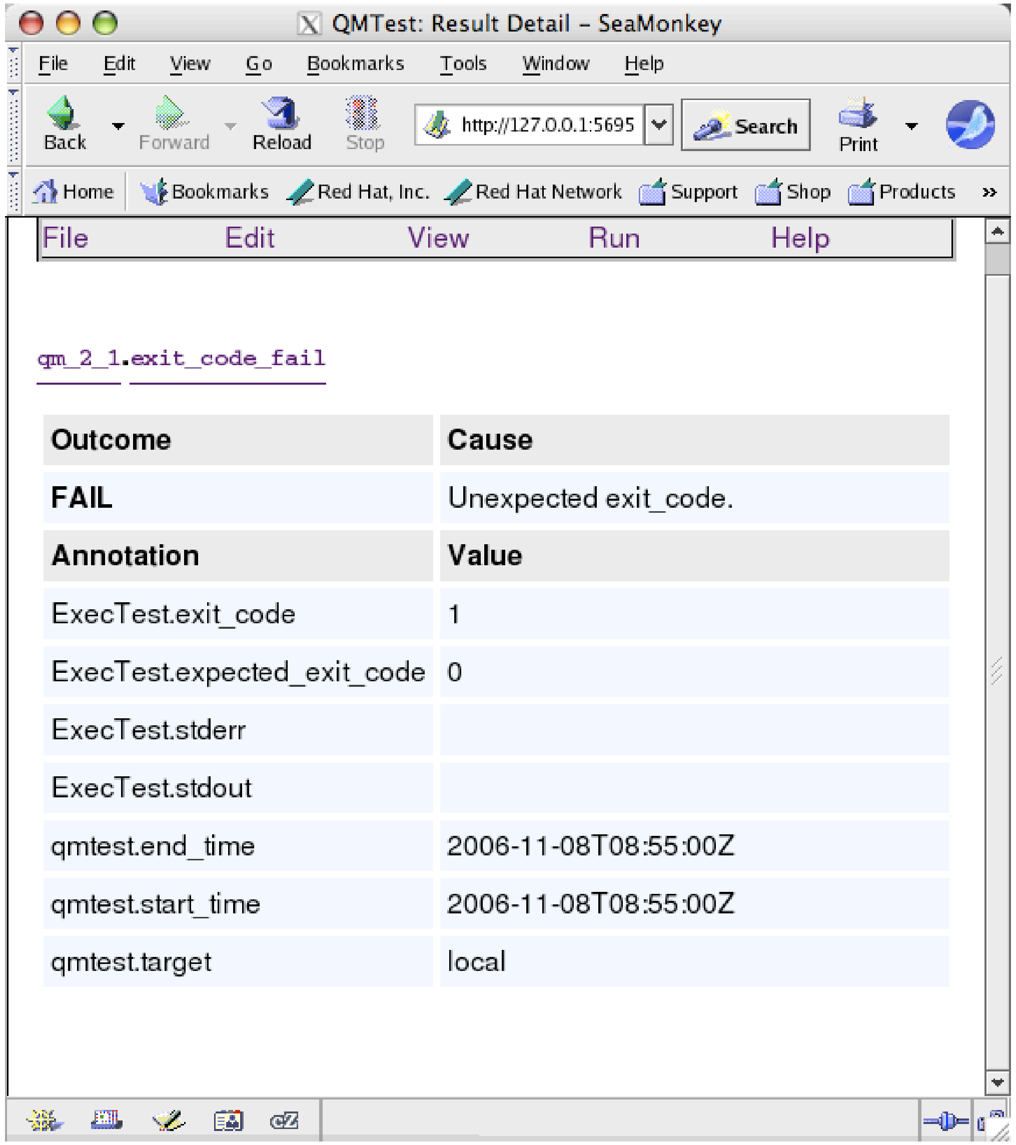
Result Summary

Outcome	# of Tests	% of Total	# Unexpected	% of Total
FAIL	3	43	3	43
PASS	4	57	0	0
Total	7	100	3	43

Tests in qm_2_1

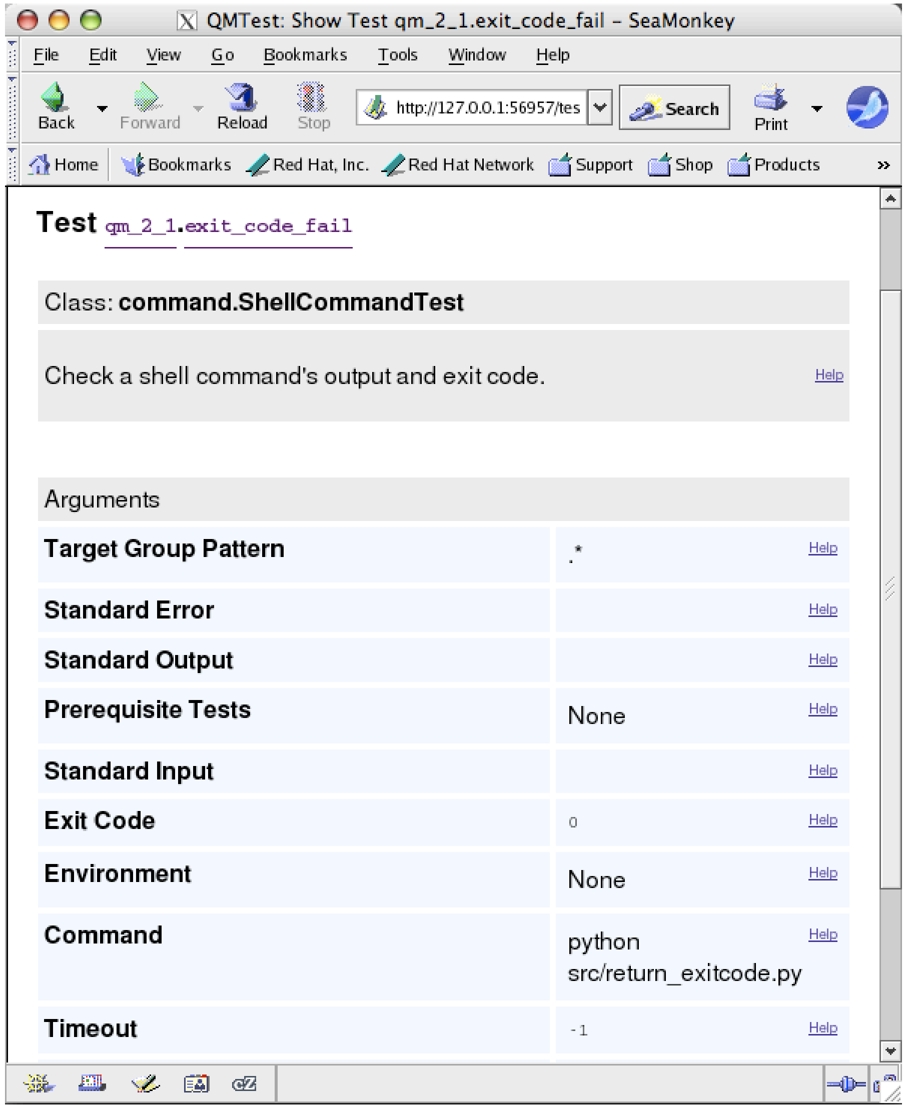
Test	Outcome	Expectation	Details
exit_code_fail	FAIL	None	Details
pyunit-example-1-fail	FAIL	None	Details
stderr	FAIL	None	Details
exit_code_ok	PASS	None	Details
pyunit-example-1-ok	PASS	None	Details
sleep	PASS	None	Details
stdout	PASS	None	Details

QMTest in SPI


QMTest: Result Detail – SeaMonkey
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop http://127.0.0.1:5695 Search Print
Home Bookmarks Red Hat, Inc. Red Hat Network Support Shop Products
File Edit View Run Help

qm_2_1.exit_code_fail

Outcome	Cause
FAIL	Unexpected exit_code.
Annotation	**Value**
ExecTest.exit_code	1
ExecTest.expected_exit_code	0
ExecTest.stderr	
ExecTest.stdout	
qmtest.end_time	2006-11-08T08:55:00Z
qmtest.start_time	2006-11-08T08:55:00Z
qmtest.target	local


QMTest: Show Test qm_2_1.exit_code_fail – SeaMonkey
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop http://127.0.0.1:56957/tes Search Print
Home Bookmarks Red Hat, Inc. Red Hat Network Support Shop Products

Test qm_2_1.exit_code_fail

Class: **command.ShellCommandTest**

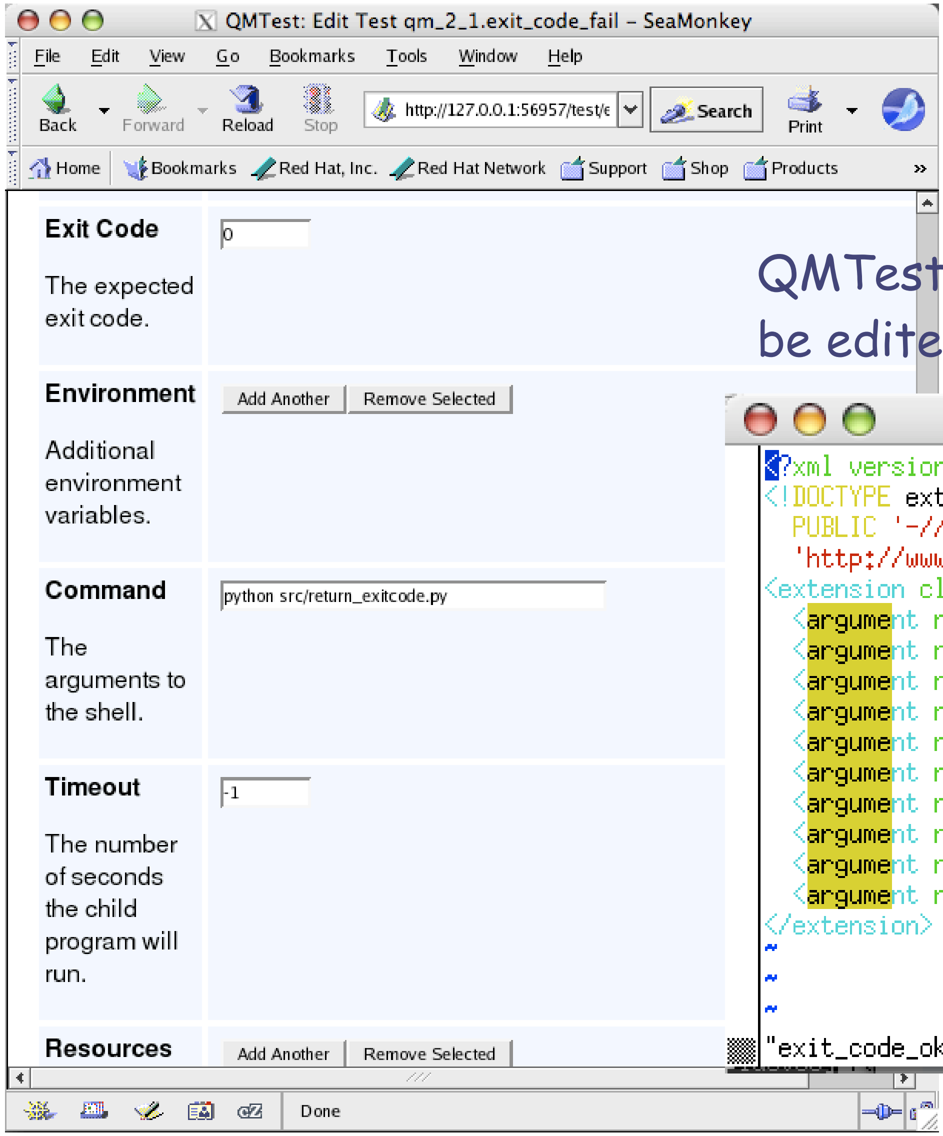
Check a shell command's output and exit code. Help

Arguments		
Target Group Pattern	.*	Help
Standard Error		Help
Standard Output		Help
Prerequisite Tests	None	Help
Standard Input		Help
Exit Code	0	Help
Environment	None	Help
Command	python src/return_exitcode.py	Help
Timeout	-1	Help

Inspecting the test failure

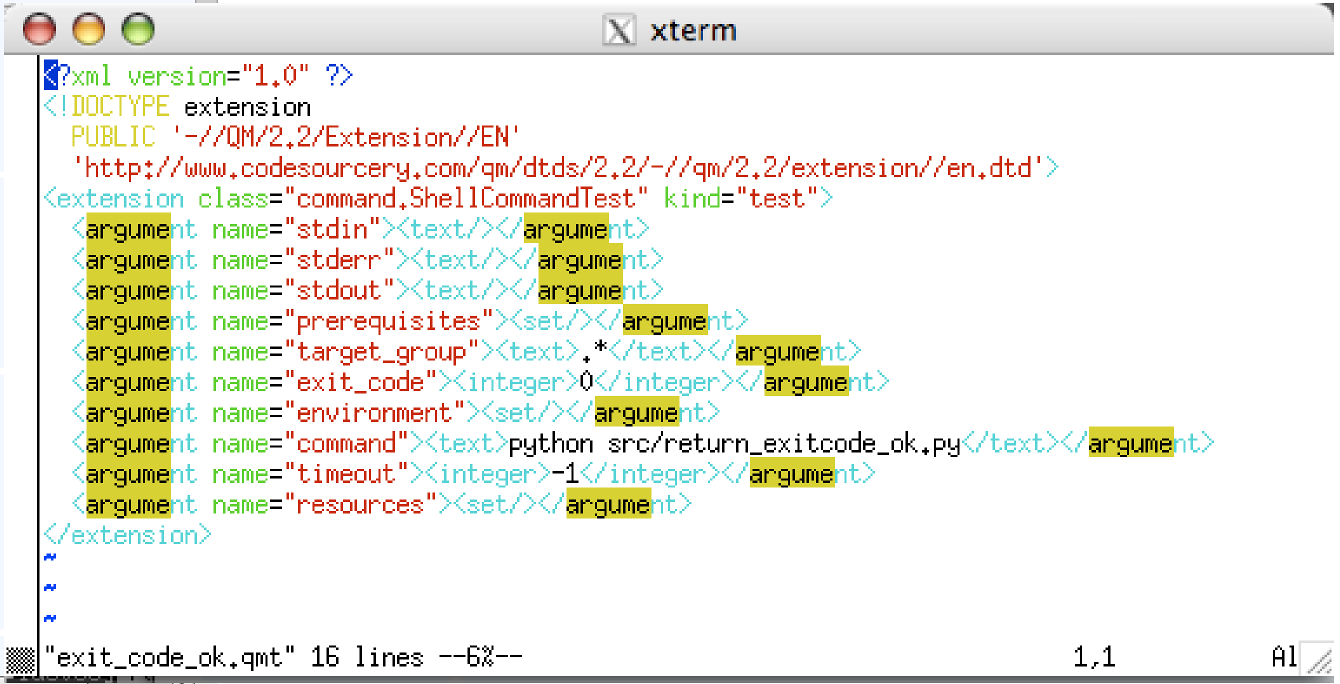
Inspecting the test definition

QMTest in SPI


QMTest: Edit Test qm_2_1.exit_code_fail - SeaMonkey
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop Search Print
http://127.0.0.1:56957/test/e
Home Bookmarks Red Hat, Inc. Red Hat Network Support Shop Products
Done

Exit Code The expected exit code.	0
Environment Additional environment variables.	Add Another / Remove Selected
Command The arguments to the shell.	python src/return_exitcode.py
Timeout The number of seconds the child program will run.	-1
Resources	Add Another / Remove Selected

QMTest test-case storage in XML: readable, can be edited and generated automatically

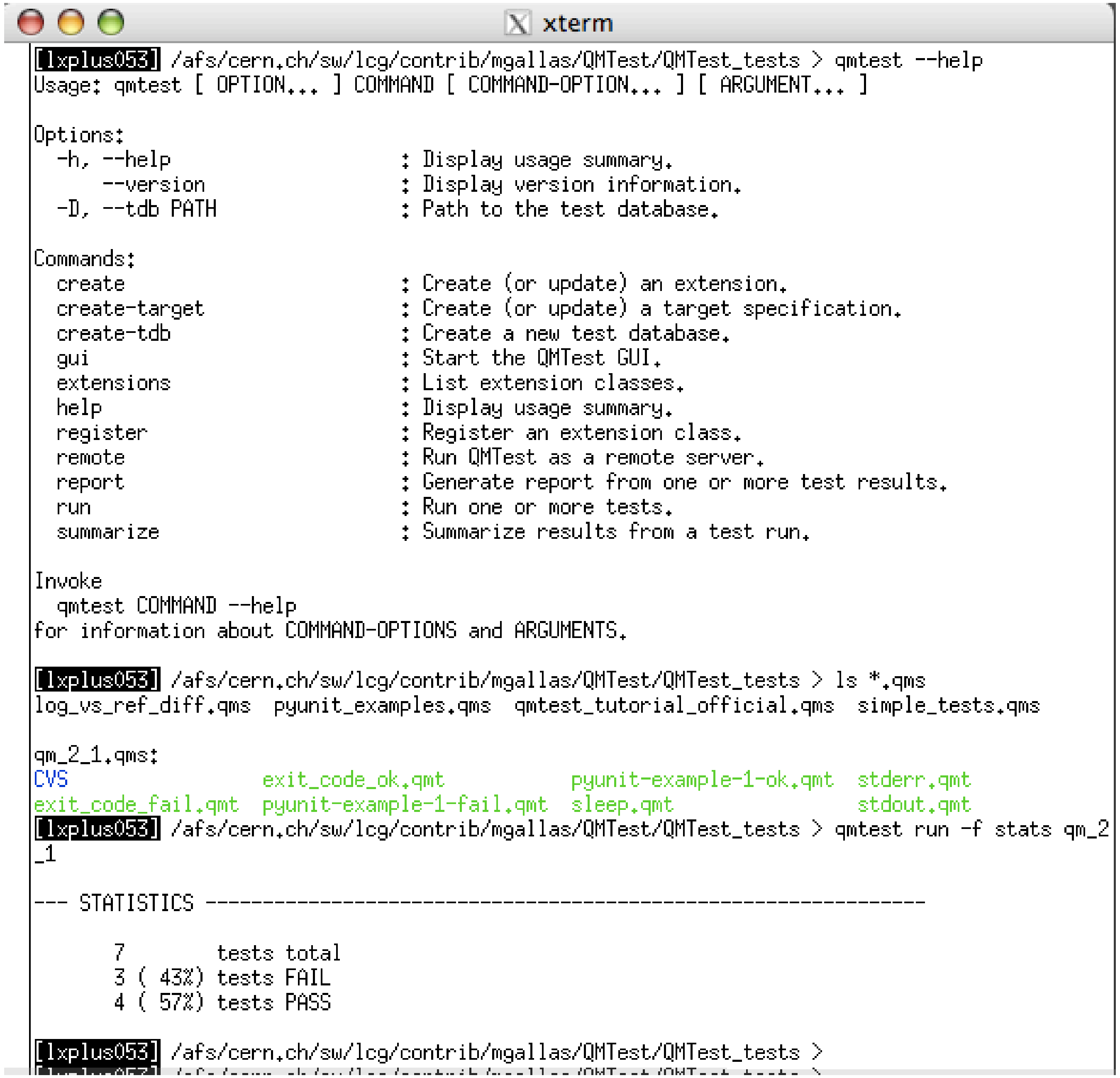


```
<?xml version="1.0" ?>
<!DOCTYPE extension
  PUBLIC '-//QM/2.2/Extension//EN'
  'http://www.codesourcery.com/qm/dtds/2.2/-//qm/2.2/extension//en.dtd'>
<extension class="command.ShellCommandTest" kind="test">
  <argument name="stdin"><text/></argument>
  <argument name="stderr"><text/></argument>
  <argument name="stdout"><text/></argument>
  <argument name="prerequisites"><set/></argument>
  <argument name="target_group"><text>.*</text></argument>
  <argument name="exit_code"><integer>0</integer></argument>
  <argument name="environment"><set/></argument>
  <argument name="command"><text>python src/return_exitcode_ok.py</text></argument>
  <argument name="timeout"><integer>-1</integer></argument>
  <argument name="resources"><set/></argument>
</extension>
~
~
~
"exit_code_ok.qmt" 16 lines --6%--                              1,1           All
```

Editing the test definition (Web interface)

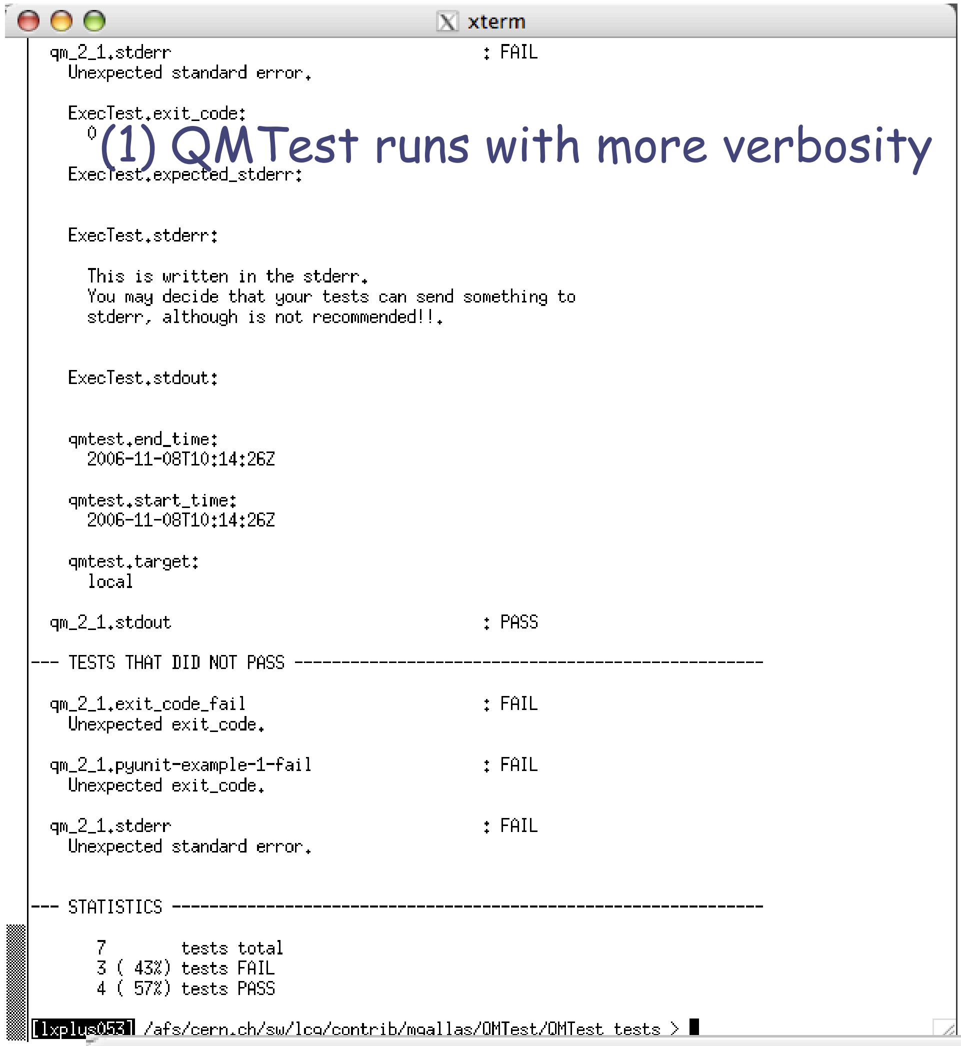
QMTest in SPI

- QMTest in the command line
- QMTest help
- QMTest runs a test suite with minimal output



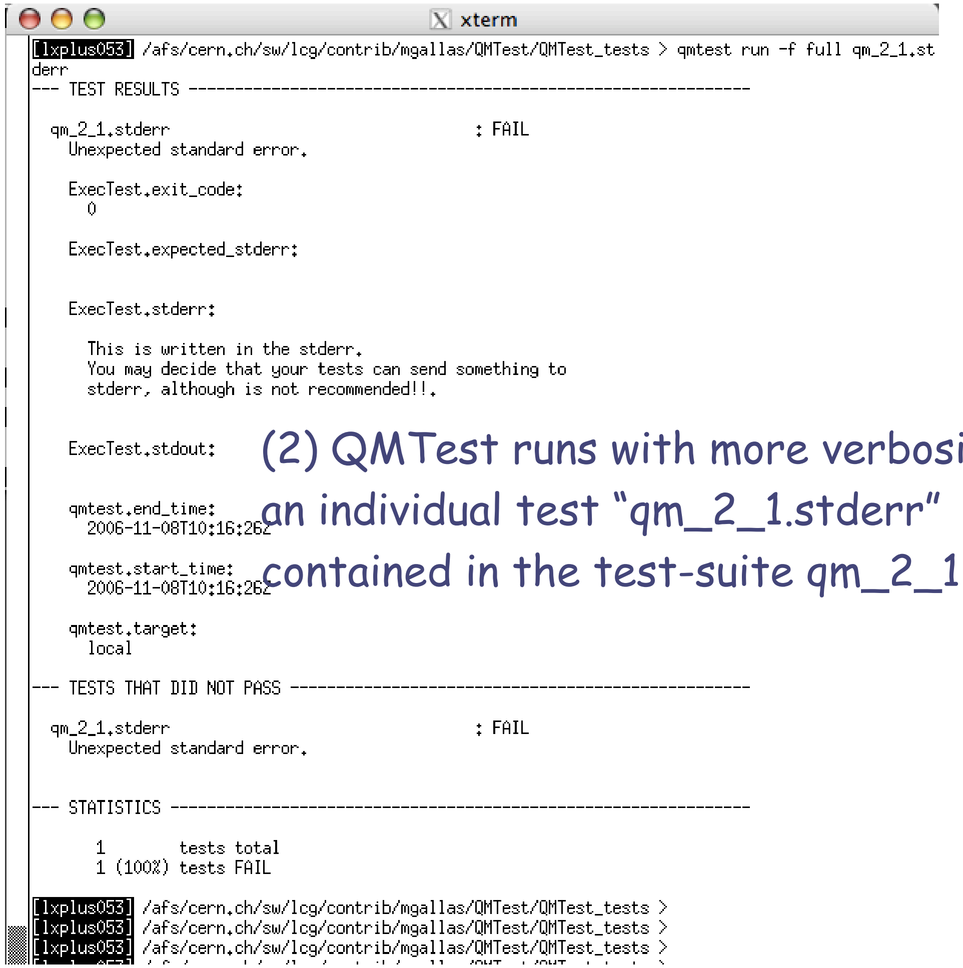
```
[lxplus053] /afs/cern.ch/sw/lcg/contrib/mgallas/QMTest/QMTest_tests > qmtest --help
Usage: qmtest [ OPTION... ] COMMAND [ COMMAND-OPTION... ] [ ARGUMENT... ]

Options:
  -h, --help                      : Display usage summary.
      --version                   : Display version information.
  -D, --tdb PATH                  : Path to the test database.

Commands:
  create                          : Create (or update) an extension.
  create-target                   : Create (or update) a target specification.
  create-tdb                      : Create a new test database.
  gui                             : Start the QMTest GUI.
  extensions                      : List extension classes.
  help                            : Display usage summary.
  register                        : Register an extension class.
  remote                          : Run QMTest as a remote server.
  report                          : Generate report from one or more test results.
  run                             : Run one or more tests.
  summarize                       : Summarize results from a test run.

Invoke
  qmtest COMMAND --help
for information about COMMAND-OPTIONS and ARGUMENTS.

[lxplus053] /afs/cern.ch/sw/lcg/contrib/mgallas/QMTest/QMTest_tests > ls *.qms
log_vs_ref_diff.qms  pyunit_examples.qms  qmtest_tutorial_official.qms  simple_tests.qms

qm_2_1.qms:
CVS                 exit_code_ok.qmt            pyunit-example-1-ok.qmt  stderr.qmt
exit_code_fail.qmt  pyunit-example-1-fail.qmt  sleep.qmt                stdout.qmt
[lxplus053] /afs/cern.ch/sw/lcg/contrib/mgallas/QMTest/QMTest_tests > qmtest run -f stats qm_2
_1

--- STATISTICS ---------------------------------------------------------------

       7        tests total
       3 ( 43%) tests FAIL
       4 ( 57%) tests PASS

[lxplus053] /afs/cern.ch/sw/lcg/contrib/mgallas/QMTest/QMTest_tests >
```

QMTest in SPI



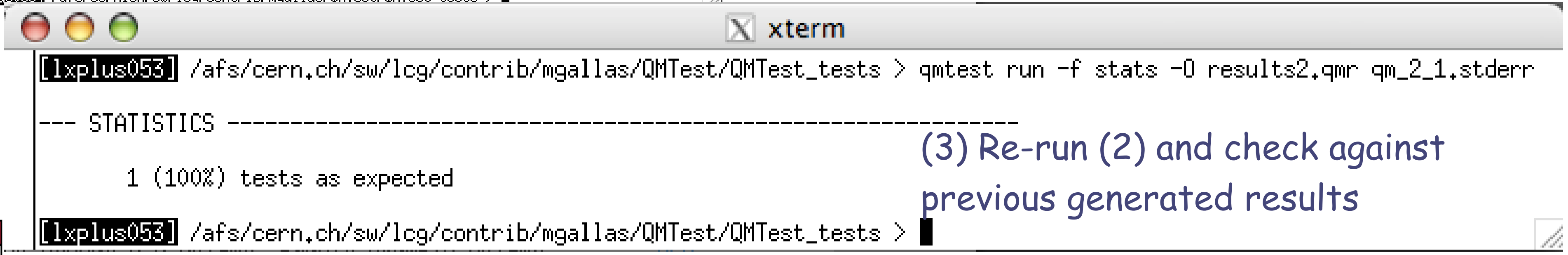
```
 qm_2_1.stderr                               : FAIL
   Unexpected standard error.

   ExecTest.exit_code:
     0

   ExecTest.expected_stderr:


   ExecTest.stderr:

     This is written in the stderr.
     You may decide that your tests can send something to
     stderr, although is not recommended!!.


   ExecTest.stdout:


   qmtest.end_time:
     2006-11-08T10:14:26Z

   qmtest.start_time:
     2006-11-08T10:14:26Z

   qmtest.target:
     local

 qm_2_1.stdout                               : PASS

--- TESTS THAT DID NOT PASS --------------------------------------------------

 qm_2_1.exit_code_fail                       : FAIL
   Unexpected exit_code.

 qm_2_1.pyunit-example-1-fail                : FAIL
   Unexpected exit_code.

 qm_2_1.stderr                               : FAIL
   Unexpected standard error.


--- STATISTICS ---------------------------------------------------------------

       7        tests total
       3 ( 43%) tests FAIL
       4 ( 57%) tests PASS

[lxplus053] /afs/cern.ch/sw/lcg/contrib/mgallas/QMTest/QMTest_tests >
```

(1) QMTest runs with more verbosity



```
[lxplus053] /afs/cern.ch/sw/lcg/contrib/mgallas/QMTest/QMTest_tests > qmtest run -f full qm_2_1.st
derr
--- TEST RESULTS -------------------------------------------------------------

 qm_2_1.stderr                               : FAIL
   Unexpected standard error.

   ExecTest.exit_code:
     0

   ExecTest.expected_stderr:


   ExecTest.stderr:

     This is written in the stderr.
     You may decide that your tests can send something to
     stderr, although is not recommended!!.


   ExecTest.stdout:


   qmtest.end_time:
     2006-11-08T10:16:26Z

   qmtest.start_time:
     2006-11-08T10:16:26Z

   qmtest.target:
     local

--- TESTS THAT DID NOT PASS --------------------------------------------------

 qm_2_1.stderr                               : FAIL
   Unexpected standard error.


--- STATISTICS ---------------------------------------------------------------

       1        tests total
       1 (100%) tests FAIL

[lxplus053] /afs/cern.ch/sw/lcg/contrib/mgallas/QMTest/QMTest_tests >
[lxplus053] /afs/cern.ch/sw/lcg/contrib/mgallas/QMTest/QMTest_tests >
[lxplus053] /afs/cern.ch/sw/lcg/contrib/mgallas/QMTest/QMTest_tests >
```

(2) QMTest runs with more verbosity an individual test "qm_2_1.stderr" contained in the test-suite qm_2_1



```
[lxplus053] /afs/cern.ch/sw/lcg/contrib/mgallas/QMTest/QMTest_tests > qmtest run -f stats -O results2.qmr qm_2_1.stderr

--- STATISTICS ---------------------------------------------------------------

       1 (100%) tests as expected

[lxplus053] /afs/cern.ch/sw/lcg/contrib/mgallas/QMTest/QMTest_tests >
```

(3) Re-run (2) and check against previous generated results

QMTest in SPI

- In LCG-I projects there was a set of test policies:
  - all the unit-tests of a given package (Package_X) must be in "Package/test"and named test_PacKage_X_myname
  - integration tests under Test directory
  - etc ...

  which allows for automatization.

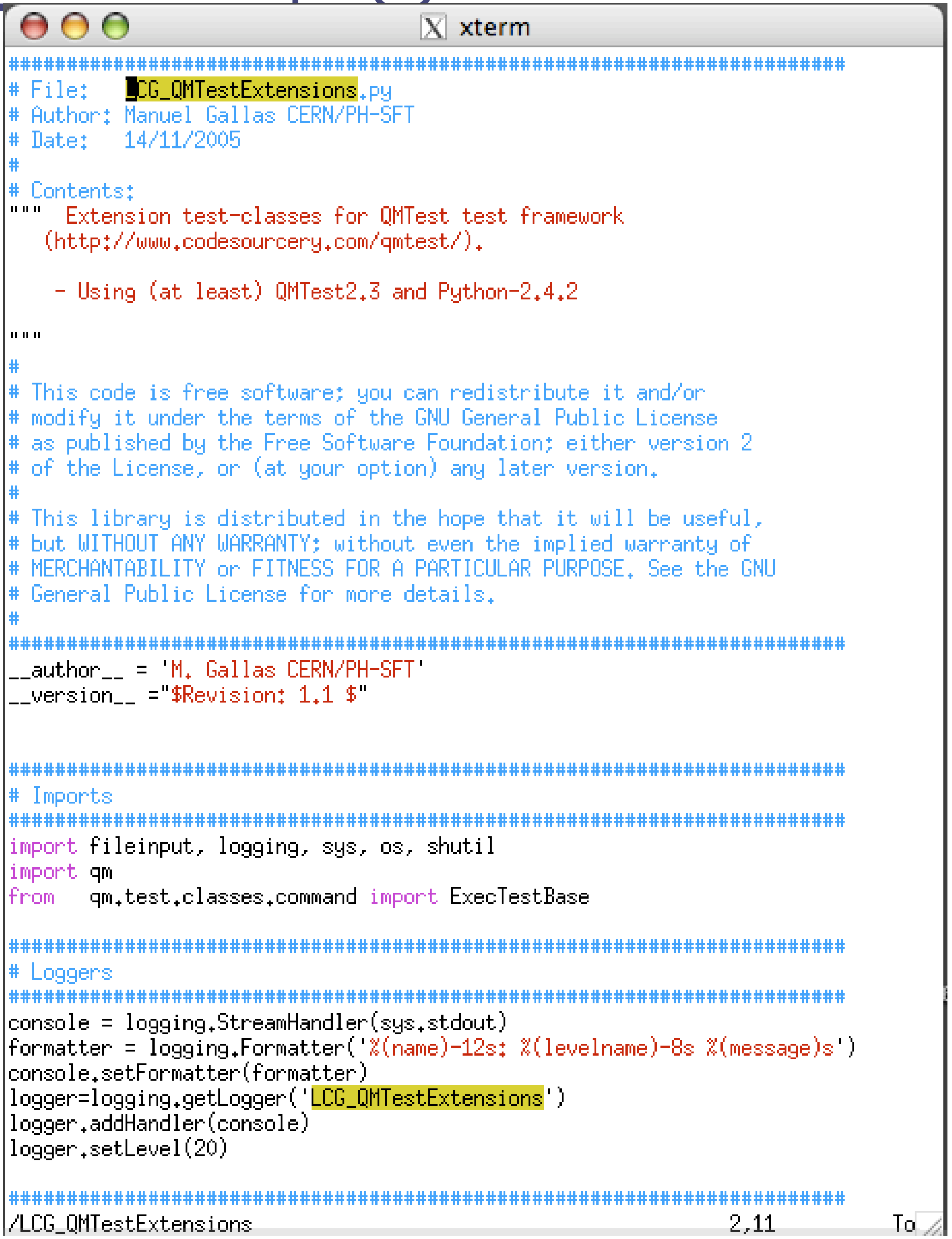
- A python script was used the "first time" or "each release time" to generate automatically the qmtest test cases (*.qmt) and suites (*.qms). The script was scanning the directory structure of the project.
- ATLAS NICOS-tests (nightly-tests integrated with the nightly-built) use the same approach and looks into the "Package/test/" for the scripts (name_shell_script.sh) suitable to be used within QMTest.
- If we need to add dependences among tests, specific context variables, resources is better to keep the qmtest configuration files (*.qms,*.qmt) in CVS and tag them with the release.

QMTest: test-domain extension example(1)

- **Steps to follow:**

1. create the extension (users can do they own extensions)
2. register the extension against the central qmtest distributions
3. create new tests using the new test-domain extension
4. use them and profit

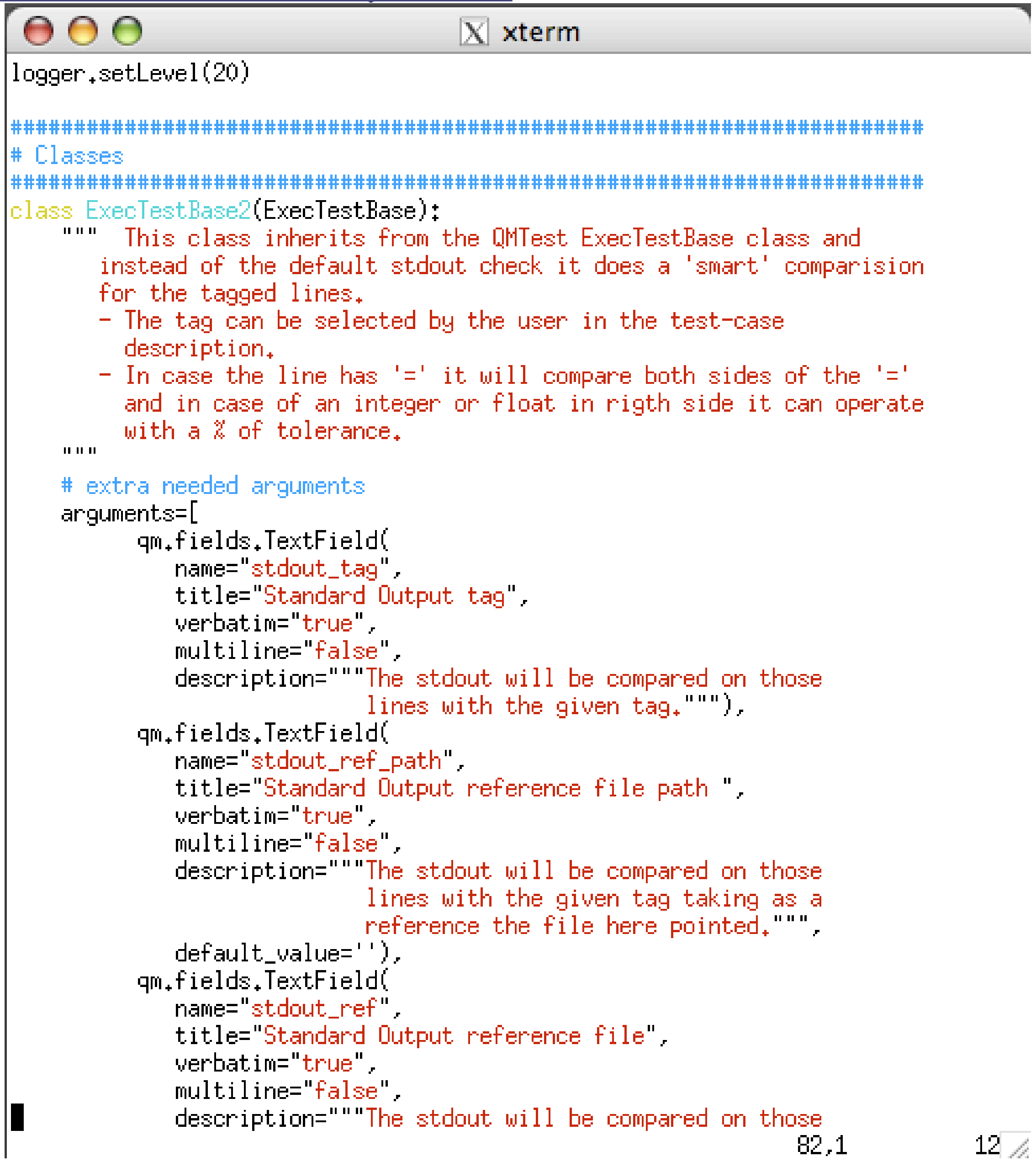
- **Step-1**:
  - Here we show the imports we need from Python and QMTest (qm)
  - We add logger for dbg



```
############################################################################
# File:    LCG_QMTestExtensions.py
# Author: Manuel Gallas CERN/PH-SFT
# Date:    14/11/2005
#
# Contents:
"""  Extension test-classes for QMTest test framework
   (http://www.codesourcery.com/qmtest/).

    - Using (at least) QMTest2.3 and Python-2.4.2

"""
#
# This code is free software; you can redistribute it and/or
# modify it under the terms of the GNU General Public License
# as published by the Free Software Foundation; either version 2
# of the License, or (at your option) any later version.
#
# This library is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
# General Public License for more details.
#
############################################################################
__author__ = 'M. Gallas CERN/PH-SFT'
__version__ ="$Revision: 1.1 $"


############################################################################
# Imports
############################################################################
import fileinput, logging, sys, os, shutil
import qm
from    qm.test.classes.command import ExecTestBase

############################################################################
# Loggers
############################################################################
console = logging.StreamHandler(sys.stdout)
formatter = logging.Formatter('%(name)-12s: %(levelname)-8s %(message)s')
console.setFormatter(formatter)
logger=logging.getLogger('LCG_QMTestExtensions')
logger.addHandler(console)
logger.setLevel(20)

############################################################################
/LCG_QMTestExtensions                                          2,11        To
```

QMTest: test-domain extension example(1)

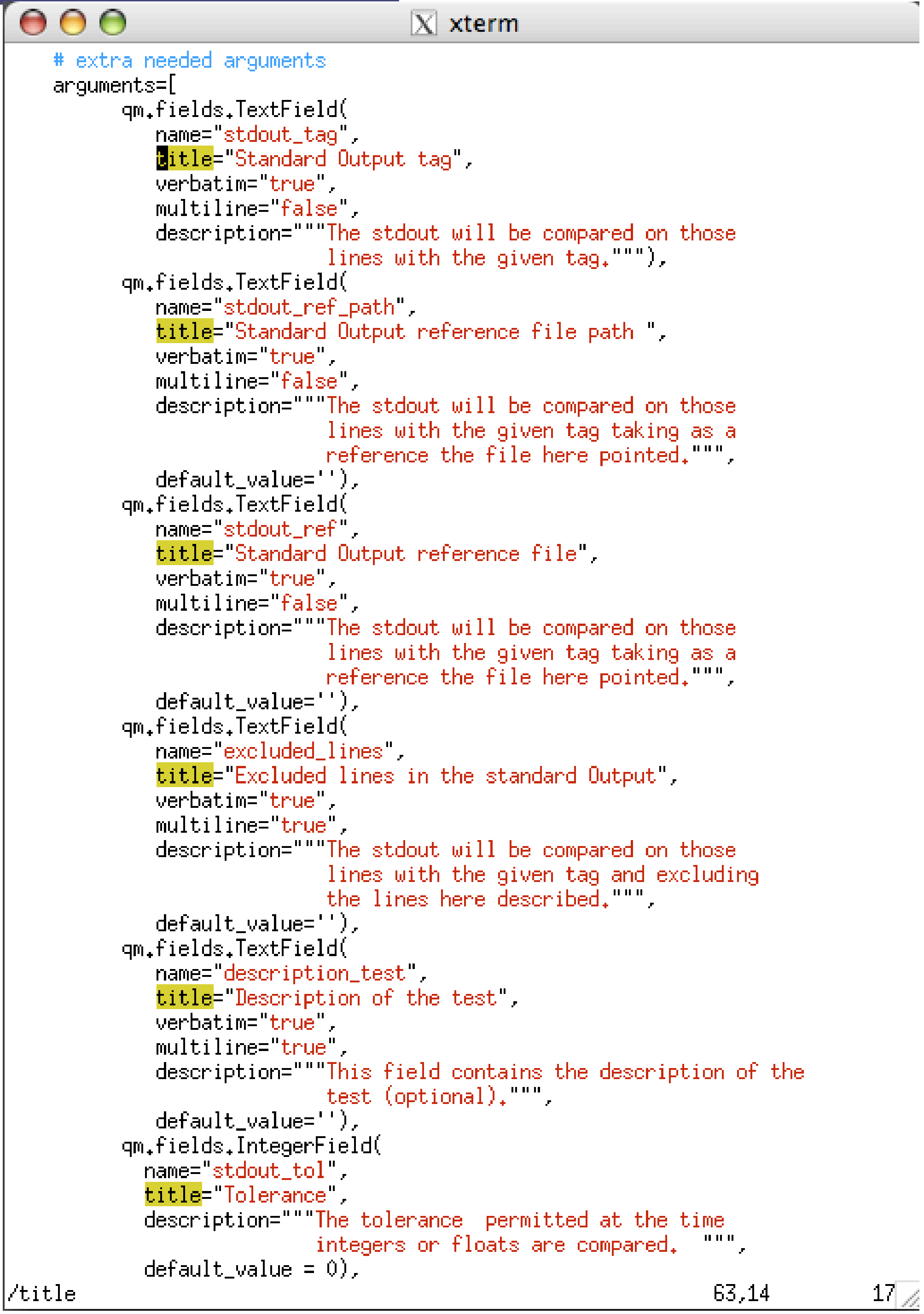
- **Step-1 (continuation)** :
  - We want to customize the command.ShellCommandTest class we use which inherits from "ExecTestBase".
  - Goal: to modify the test execution and be able to compare the stdout with a reference only for the "tagged lines"
  - We create a new class "ExecTestBase" and we add our modifications starting form the new fields we may need.



```
logger.setLevel(20)

###########################################################################
# Classes
###########################################################################
class ExecTestBase2(ExecTestBase):
    """  This class inherits from the QMTest ExecTestBase class and
       instead of the default stdout check it does a 'smart' comparision
       for the tagged lines.
       - The tag can be selected by the user in the test-case
         description.
       - In case the line has '=' it will compare both sides of the '='
         and in case of an integer or float in rigth side it can operate
         with a % of tolerance.
    """
    # extra needed arguments
    arguments=[
          qm.fields.TextField(
             name="stdout_tag",
             title="Standard Output tag",
             verbatim="true",
             multiline="false",
             description="""The stdout will be compared on those
                            lines with the given tag."""),
          qm.fields.TextField(
             name="stdout_ref_path",
             title="Standard Output reference file path ",
             verbatim="true",
             multiline="false",
             description="""The stdout will be compared on those
                            lines with the given tag taking as a
                            reference the file here pointed.""",
             default_value=''),
          qm.fields.TextField(
             name="stdout_ref",
             title="Standard Output reference file",
             verbatim="true",
             multiline="false",
             description="""The stdout will be compared on those
                                                          82,1          12
```

QMTest: test-domain extension example(1)

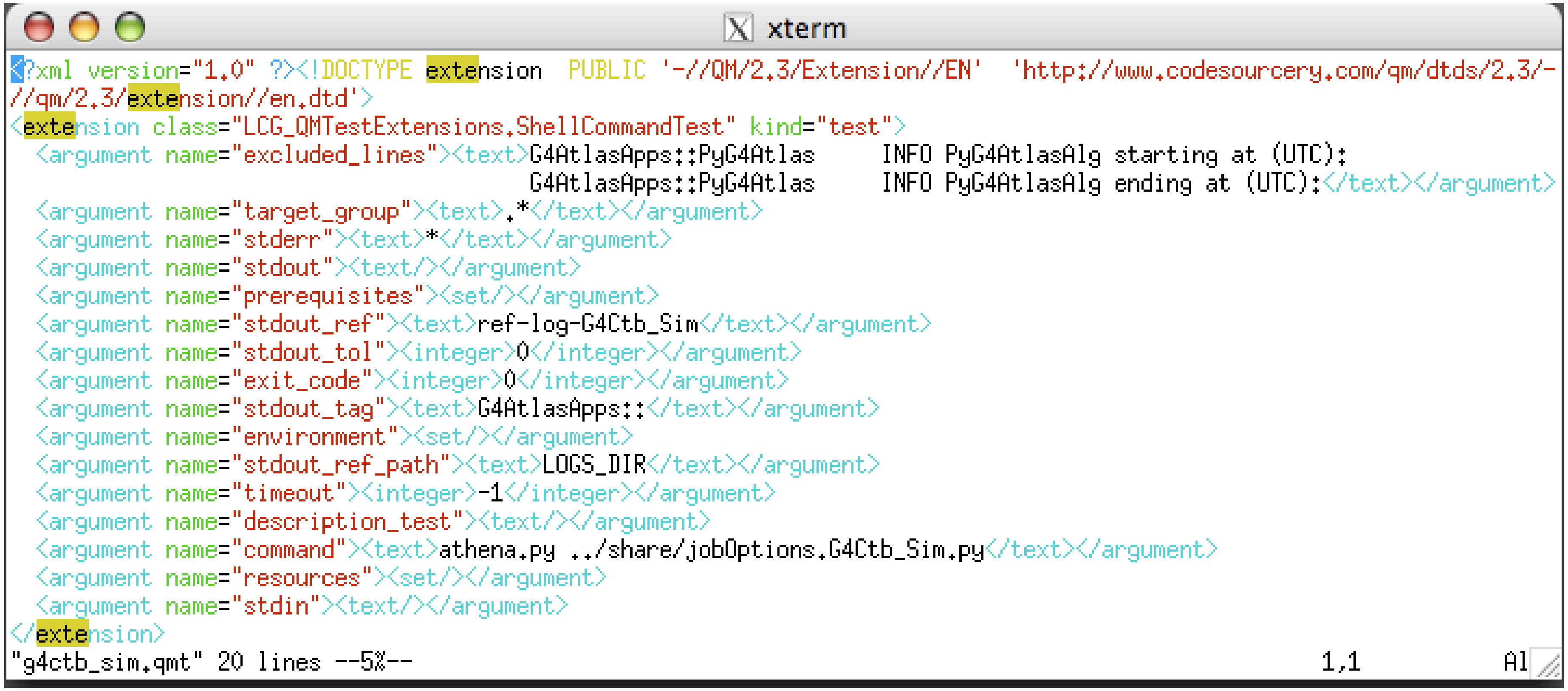
- **Step-1 (continuation)** :
  - Full list of additional arguments that will be added to the test-case definition at the creation time
  - See in the next slide how the test-case XML looks like now



```
    # extra needed arguments
    arguments=[
          qm.fields.TextField(
             name="stdout_tag",
             title="Standard Output tag",
             verbatim="true",
             multiline="false",
             description="""The stdout will be compared on those
                            lines with the given tag."""),
          qm.fields.TextField(
             name="stdout_ref_path",
             title="Standard Output reference file path ",
             verbatim="true",
             multiline="false",
             description="""The stdout will be compared on those
                            lines with the given tag taking as a
                            reference the file here pointed.""",
             default_value=''),
          qm.fields.TextField(
             name="stdout_ref",
             title="Standard Output reference file",
             verbatim="true",
             multiline="false",
             description="""The stdout will be compared on those
                            lines with the given tag taking as a
                            reference the file here pointed.""",
             default_value=''),
          qm.fields.TextField(
             name="excluded_lines",
             title="Excluded lines in the standard Output",
             verbatim="true",
             multiline="true",
             description="""The stdout will be compared on those
                            lines with the given tag and excluding
                            the lines here described.""",
             default_value=''),
          qm.fields.TextField(
             name="description_test",
             title="Description of the test",
             verbatim="true",
             multiline="true",
             description="""This field contains the description of the
                            test (optional).""",
             default_value=''),
          qm.fields.IntegerField(
            name="stdout_tol",
            title="Tolerance",
            description="""The tolerance  permitted at the time
                           integers or floats are compared.  """,
            default_value = 0),
/title                                                         63,14         17
```

Manuel Gallas (SPI)
CERN PH-SFT

QMTest: test-domain extension example(1)



xterm

```
<?xml version="1.0" ?><!DOCTYPE extension  PUBLIC '-//QM/2.3/Extension//EN'  'http://www.codesourcery.com/qm/dtds/2.3/-
//qm/2.3/extension//en.dtd'>
<extension class="LCG_QMTestExtensions.ShellCommandTest" kind="test">
  <argument name="excluded_lines"><text>G4AtlasApps::PyG4Atlas     INFO PyG4AtlasAlg starting at (UTC):
                                        G4AtlasApps::PyG4Atlas     INFO PyG4AtlasAlg ending at (UTC):</text></argument>
  <argument name="target_group"><text>.*</text></argument>
  <argument name="stderr"><text>*</text></argument>
  <argument name="stdout"><text/></argument>
  <argument name="prerequisites"><set/></argument>
  <argument name="stdout_ref"><text>ref-log-G4Ctb_Sim</text></argument>
  <argument name="stdout_tol"><integer>0</integer></argument>
  <argument name="exit_code"><integer>0</integer></argument>
  <argument name="stdout_tag"><text>G4AtlasApps::</text></argument>
  <argument name="environment"><set/></argument>
  <argument name="stdout_ref_path"><text>LOGS_DIR</text></argument>
  <argument name="timeout"><integer>-1</integer></argument>
  <argument name="description_test"><text/></argument>
  <argument name="command"><text>athena.py ../share/jobOptions.G4Ctb_Sim.py</text></argument>
  <argument name="resources"><set/></argument>
  <argument name="stdin"><text/></argument>
</extension>
"g4ctb_sim.qmt" 20 lines --5%--                                                                   1,1           All
```

- The reference-file for this individual test will be "ref-log-G4Ctb-Sim" at $LOGS_DIR (the lines can also included in here instead of read them from ref-files)
- The tag will be "GAtlasApps::" from Athena/Gaudi message service
- Lines with time-output will be excluded
- And the job is: ' athena.py ../share/jobOptions.G4Ctb_Sim.py'

QMTest: test-domain extension example(1)

- **Step-1 (continuation)** :
  - We re-write the method ExecTestBase.ValidateOutput

    - to know about: the “tag” for comparisons, the reference file, the tolerance & to copy the stdout into a log (as future reference)

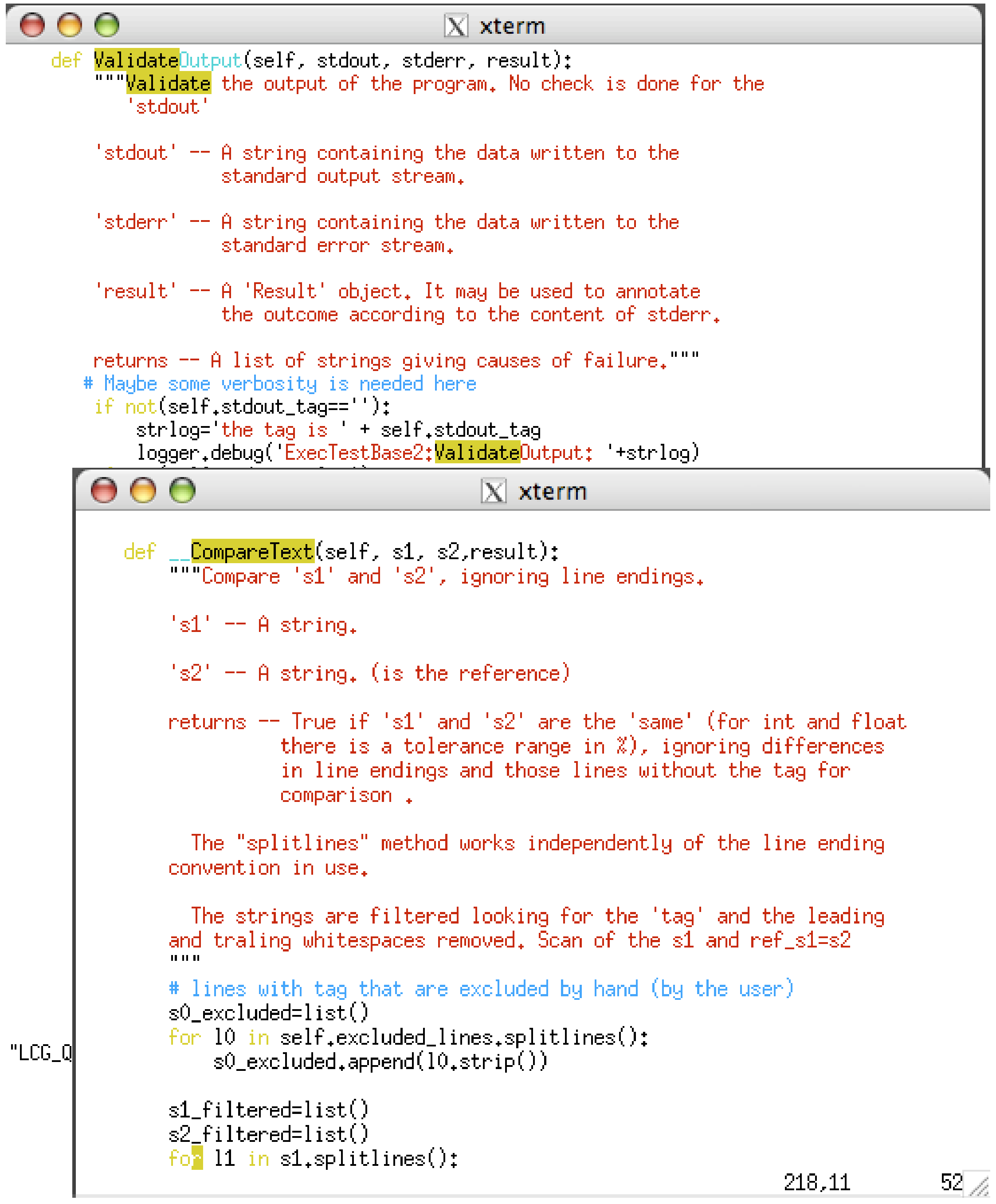
  - We re-write the method ExecTestBase.CompareText to do our smart comparisons

  - Last step!!. We define:

    class ShellCommandTest(ExecTestBase2)

    and it will be:

  - LCG_QMTestExtensions. ShellCommandTest

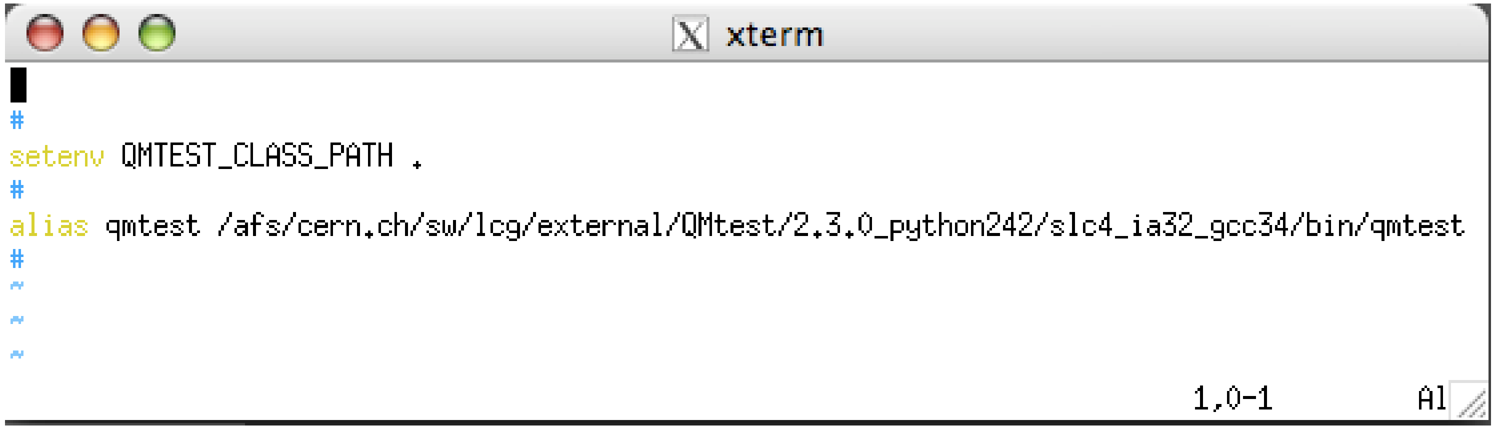


```
    def ValidateOutput(self, stdout, stderr, result):
        """Validate the output of the program. No check is done for the
           'stdout'

        'stdout' -- A string containing the data written to the
                    standard output stream.

        'stderr' -- A string containing the data written to the
                    standard error stream.

        'result' -- A 'Result' object. It may be used to annotate
                    the outcome according to the content of stderr.

        returns -- A list of strings giving causes of failure."""
       # Maybe some verbosity is needed here
        if not(self.stdout_tag==''):
            strlog='the tag is ' + self.stdout_tag
            logger.debug('ExecTestBase2:ValidateOutput: '+strlog)
"LCG_Q
```

```
    def __CompareText(self, s1, s2,result):
        """Compare 's1' and 's2', ignoring line endings.

        's1' -- A string.

        's2' -- A string. (is the reference)

        returns -- True if 's1' and 's2' are the 'same' (for int and float
                   there is a tolerance range in %), ignoring differences
                   in line endings and those lines without the tag for
                   comparison .

          The "splitlines" method works independently of the line ending
        convention in use.

          The strings are filtered looking for the 'tag' and the leading
        and traling whitespaces removed. Scan of the s1 and ref_s1=s2
        """
        # lines with tag that are excluded by hand (by the user)
        s0_excluded=list()
        for l0 in self.excluded_lines.splitlines():
            s0_excluded.append(l0.strip())

        s1_filtered=list()
        s2_filtered=list()
        for l1 in s1.splitlines():
                                                                218,11        52
```

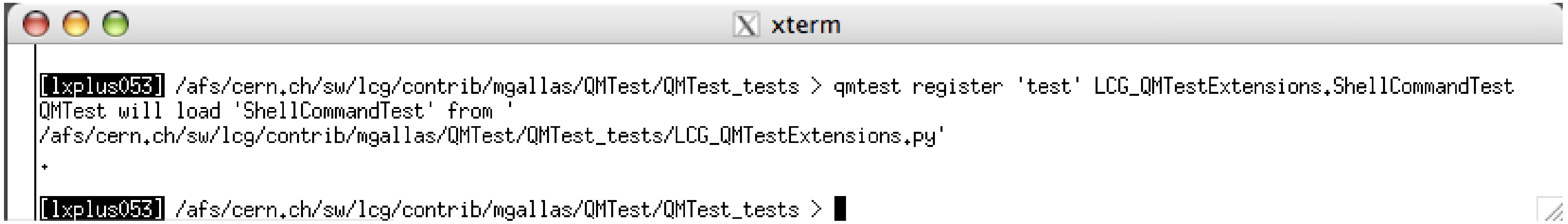
QMTest: test-domain extension example(2)

- Set **$QMTEST_CLASS_PATH** to the directory with the **LCG_QMTestExtensions.py** python-module



```
#
setenv QMTEST_CLASS_PATH .
#
alias qmtest /afs/cern.ch/sw/lcg/external/QMtest/2.3.0_python242/slc4_ia32_gcc34/bin/qmtest
#
~
~
~
                                                              1,0-1         All
```

- register the new test "ShellCommandTest"



```
[lxplus053] /afs/cern.ch/sw/lcg/contrib/mgallas/QMTest/QMTest_tests > qmtest register 'test' LCG_QMTestExtensions.ShellCommandTest
QMTest will load 'ShellCommandTest' from '
/afs/cern.ch/sw/lcg/contrib/mgallas/QMTest/QMTest_tests/LCG_QMTestExtensions.py'
.
[lxplus053] /afs/cern.ch/sw/lcg/contrib/mgallas/QMTest/QMTest_tests >
```

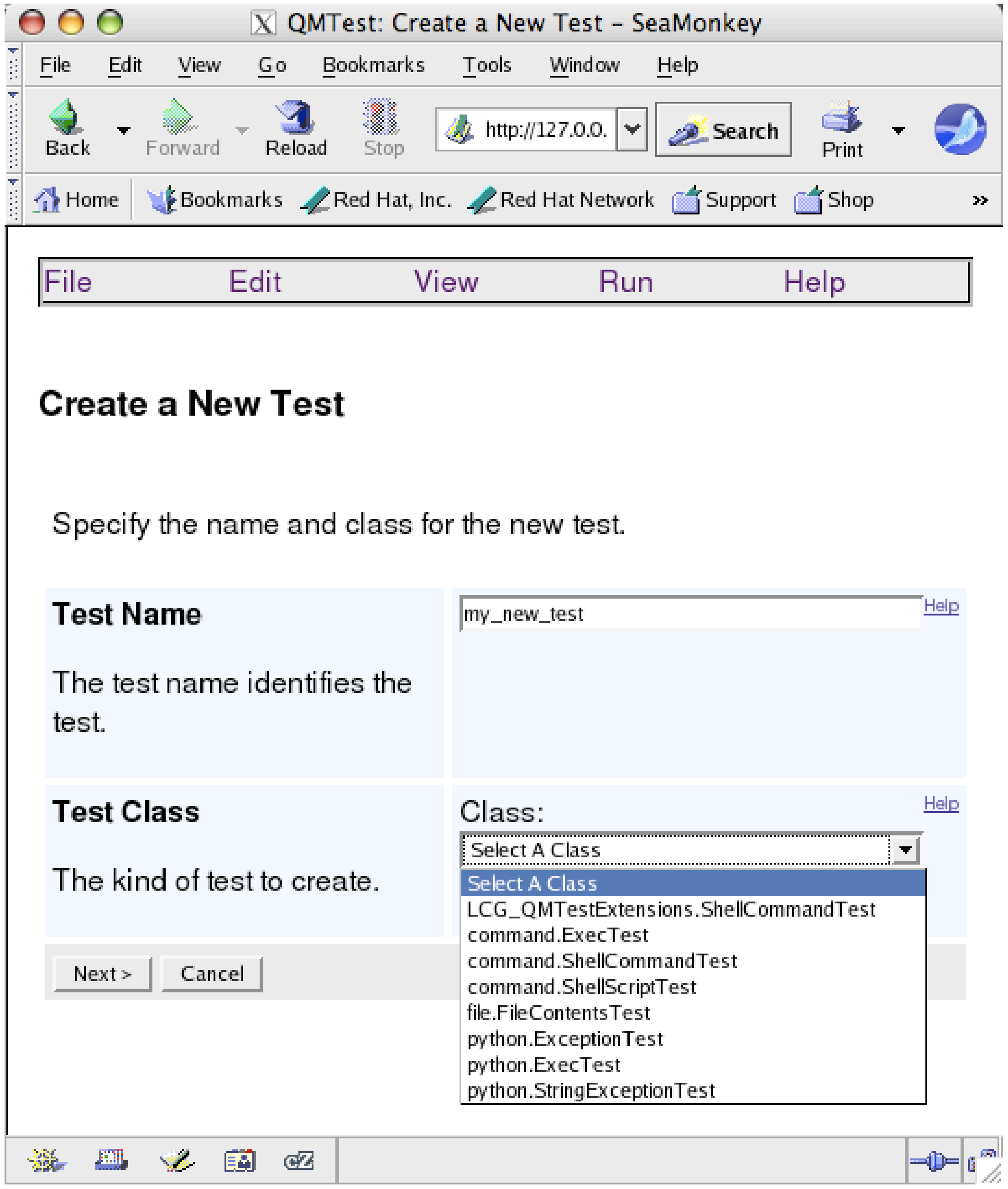
QMTest: test-domain extension example(3)

- Create a new test selecting the new

  LCG_QMTestExtensions.ShellCommand

  test-class

- Old tests can still use the old test-class from QMTest


QMTest: Create a New Test - SeaMonkey
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop http://127.0.0. Search Print
Home Bookmarks Red Hat, Inc. Red Hat Network Support Shop
File Edit View Run Help

Create a New Test

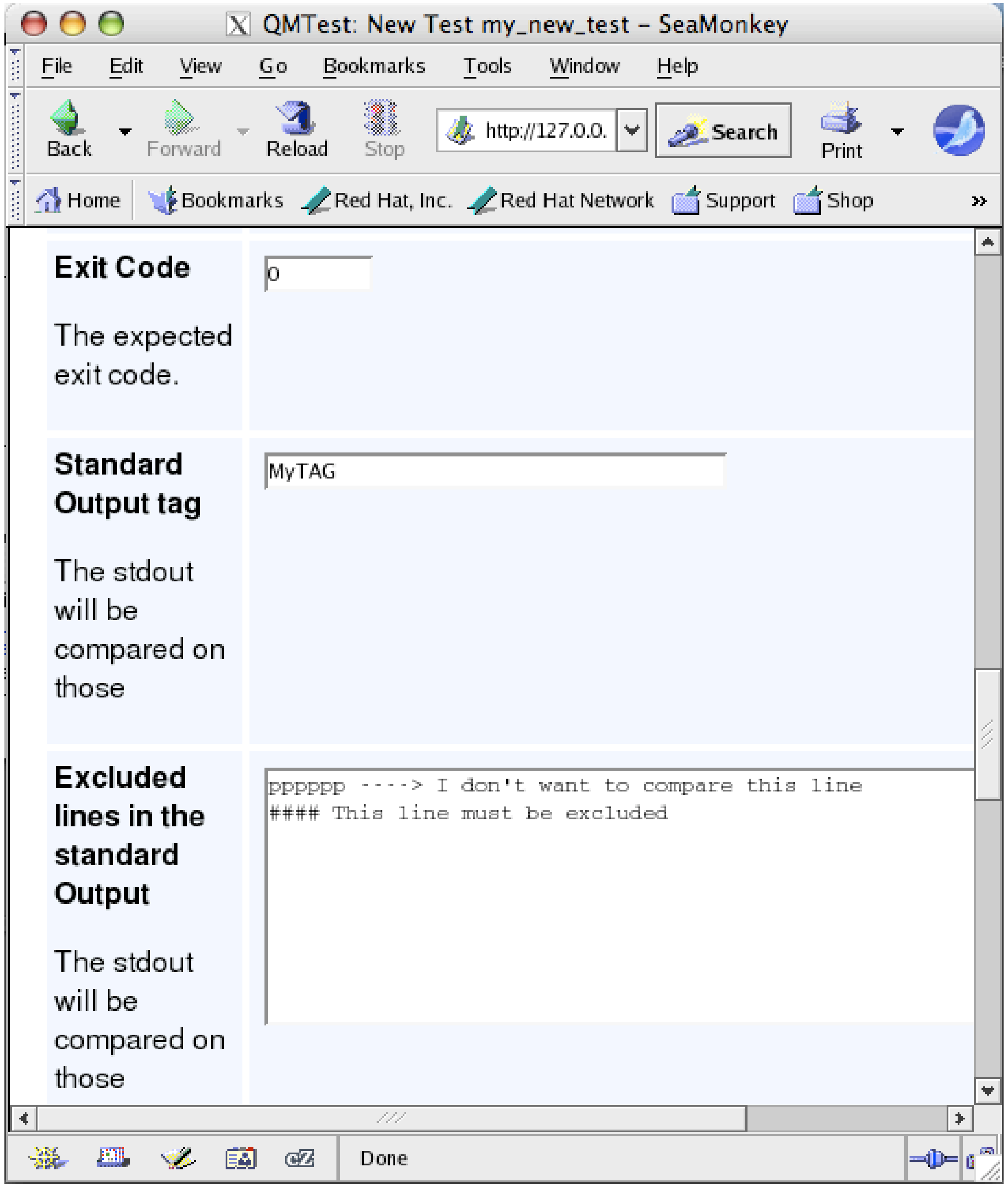
Specify the name and class for the new test.

Test Name The test name identifies the test.	my_new_test
Test Class The kind of test to create.	Class: Select A Class

Select A Class
LCG_QMTestExtensions.ShellCommandTest
command.ExecTest
command.ShellCommandTest
command.ShellScriptTest
file.FileContentsTest
python.ExceptionTest
python.ExecTest
python.StringExceptionTest

QMTest: test-domain extension example(3)

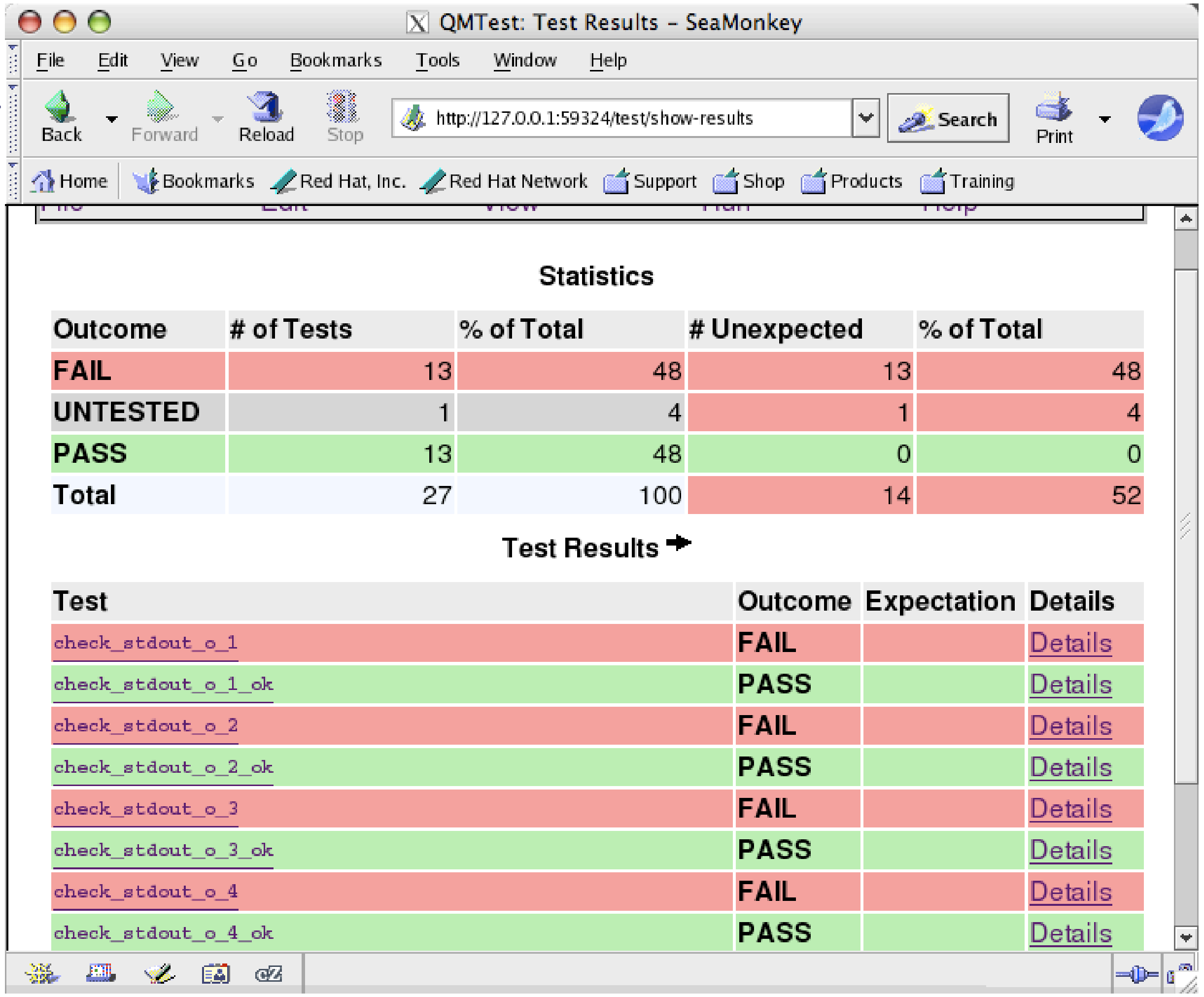
- Fill the fields in the Web interface
- Or you may want to generate the test-case config file automatically.


QMTest: New Test my_new_test – SeaMonkey
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop http://127.0.0. Search Print
Home Bookmarks Red Hat, Inc. Red Hat Network Support Shop
Done

Exit Code The expected exit code.	0
Standard Output tag The stdout will be compared on those	MyTAG
Excluded lines in the standard Output The stdout will be compared on those	pppppp ----> I don't want to compare this line #### This line must be excluded

QMTest: test-domain extension example(4)

- Before use, perform some unit-tests with your new test-class (PASS and FAIL tests).


QMTest: Test Results - SeaMonkey
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://127.0.0.1:59324/test/show-results
Search Print
Home Bookmarks Red Hat, Inc. Red Hat Network Support Shop Products Training

Statistics

Outcome	# of Tests	% of Total	# Unexpected	% of Total
FAIL	13	48	13	48
UNTESTED	1	4	1	4
PASS	13	48	0	0
Total	27	100	14	52

Test Results

Test	Outcome	Expectation	Details
check_stdout_o_1	FAIL		Details
check_stdout_o_1_ok	PASS		Details
check_stdout_o_2	FAIL		Details
check_stdout_o_2_ok	PASS		Details
check_stdout_o_3	FAIL		Details
check_stdout_o_3_ok	PASS		Details
check_stdout_o_4	FAIL		Details
check_stdout_o_4_ok	PASS		Details

Conclusions

- QMTest is an open-source, general-purpose, cross-platform sw-testing tool written in Python.
- QMTest helps in the creation/organization of the tests, execution, display of results (domain-independent) and gluing other test frameworks (domain-dependent).
- QMTest is very customizable and has a readable code which makes easy the user extensions.
- An example of a test-domain extension was presented.
  - The example tries to show how QMTest can be extended by users in a clean way (registering the new test classes) against the central QMTest distributions.
  - In this particular example we focussed into the test approach in which a log file is compared with a reference file only in those tagged lines.
  - Users can make their own extension and store/develop it in their project/s
- Feedback and suggestions are very welcome.