Table of Contents

EMI Testing Policy

Latest approved version of this policy
03.12.2010: EMI_SA2_CertificationAndTesting_v_1_0.pdf

Note that this twiki contains a newer version of the Testing Policy to be approved by PEB. The latest approved version is accesible in the PDF file.

Introduction

This document describes the EMI policy to be followed when testing a new version of an EMI software component. The following topics are covered by this policy:

- Tests to be performed.
- Test Plan.
- Test Report.
- Testing process.

Definitions

- **Component**: It refers to any of the EMI software components as listed here.
- **Component Release or CR**: New version of a component.
- **Testing**: action done by a PT on a component release in order to detect defects in the software. Testing is done by contrasting a computer program's expected results with its actual results for a given set of inputs.

Tests to be performed

The following sections explain in detail the type of tests, mandatory or optional, to be included in the test plan of any EMI software component.

Each section describes:

- Type of test: mandatory or optional.
- Definition of the type of test.
- Required coverage: how many tests must be defined for each type of test.
- Release Criteria: how many tests must be executed and passed for each component release.

Static Code Analysis

Type of Test: Mandatory if the necessary plugins are available in ETICS, otherwise this test is optional.

Definition: Static Code Analysis is the analysis of software by examining the code without executing the program. Automated tools are often used to carry out static code analysis. In the context of EMI, ETICS offers plugins for static code analysis. Currently the plugins available for static code analysis are:

- Java: FindBugs, PMD, Checkstyle.
- C/C++: None yet. CCCC plugin is under development.
- Python: None yet. Pylint plugin is under development.

Required Coverage: No requirement.

Release Criteria: Report about the results of the Static Code Analysis in yout test Report. If your code is written in a language not supported by the ETICS plugins, static code analysis is optional and can be also included in the Test Report.

Unit tests

Type of Test: Optional.

Definition: Unit tests are meant to test the correctness of specific sections of source code. Tools like CPPUnit, JUnit, PyUnit are generally used and they provide the necessary documentation to get started with unit tests.

Unit tests code coverage describes the degree to which the source code of a program has been tested. A high unit test code coverage not only improves the reliability of the software, but also helps to have software that will be easier to maintain.

Required Coverage: ETICS doesn't have a plugin to calculate unit test code coverage. For this reason, there is no code coverage requirement.

Release Criteria: You are welcome to report about unit test code coverage in your test report if you already calculate this for your software component with any other tool of your preference.

Deployment tests

Type of Test: Mandatory for **rpm** packages.

Definition: Deployment tests verify that the component can be properly installed and configured on all the supported platforms.

EMI components are distributed in several package formats according to the Packaging Policy. Deployment tests must be performed for at least rpm packages.

The installation tool for rpm packages supported by EMI is YUM.

Deployment tests must specify the necessary YUM commands and YUM repositories needed to install the software component.

- The YUM commands must be in the form of: `yum install/update metapackage_name`. Details on how to define metapackages can be found in the Packaging Policy.
- The YUM repositories needed to install the software component are:
  - ETICS YUM repository of the `emi_B_1_dev` project configuration, which is the project configuration containing all the new components scheduled for this release plus the production versions of the components that do not change.
  - Some of the following external repositories may be also needed, depending on the software component:
    - EPEL repository
    - SL5 OS repository
    - EGI trustanchors repository
    - Other repositories for external packages not included in EPEL like `fetch-crl`. (to confirm which repos for which packages from PEB).

The configuration tool depends on the specific component. Deployment tests must also define the configuration commands and configuration variables, if any, needed to be able to configure the component.

- A configuration command is the command that needs to be run to configure your software component. For instance, in case of using yaim: `yaim -c -s site-info.def -n metapackage_name`.
- A configuration variable is a variable that needs to be defined by the user. For instance, in case of using yaim: `BDII_HOST`.

Required Coverage: deployment tests must include both clean and upgrade installations and configurations.

Release Criteria: all deployment tests must be executed and succesfully passed for any component release.

System tests

System tests cover the majority of the tests for a component release. System tests considered in this document are:

- Basic functionality tests
- Regression tests
- Performance tests
- Sacalability tests
- Standard compliance/conformance tests

The broader is the scope of system testing in these areas, the better. We encourage PTs to keep on improving their test plans thoughout the EMI project lifetime by broadening and enlarging the scope of their tests as well as their level of automation.

Basic functionality tests

Type of Test: Mandatory.

Definition: Basic functionality tests aim at testing the core features of the component. Basic functionality tests must include the description of the functionality to be tested. When new functionality is added to the component, new tests must be written and added to the test plan of the component.

Required Coverage: All the core features of a component must have a corresponding basic functionality test.

Release Criteria: All basic functionality tests must be executed and succesfully passed for any component release.

Regression tests

Type of Test: Mandatory.

Definition: Regression tests are tests that are meant to verify specific software defects (`bugs`). A regression test must be associated to the RfC where the defect has been reported in the RfC tracker.

Required Coverage: All RfCs tracking a bug where the bug verification can be automatically implemented must have a corresponding functionality test.

Release Criteria: All available regression tests must be executed and succesfully passed for any component release.

Performance tests

Type of Test: Optional.

Definition: Performance tests are tests that aim at verifying the performance of a component, which in many cases involves the measurement of the response time for specific service requests. Performance tests should verify how well the service behaves with nominal workloads.

The execution of performance tests depends on the service under test, it may or may not be automated, and can involve the use of external tools. In some cases the execution of performance tests may require the establishment of a specific testbed, and may involve several sites and the coordination of SA2.6.

Required Coverage: a minimum set of performance requirements must be identified by the PT. A corresponding set of tests must be defined to check that the component is able to meet those requirements.

Release Criteria: If there are performance tests defined, they must be executed in every major component release where there are substantial functionality changes.

Scalability tests

Type of Test: Optional.

Definition: Scalability tests are meant to verify that the component behaves according to its specifications when varying one of the variables that can affect its performance. Load and stress tests are included.

The execution of scalability tests depends on the service under test, it may or may not be automated, and can involve the use of external tools. In some cases the execution of scalability tests may require the establishment of a specific testbed, and may involve sites and the coordination of SA2.6.

Required Coverage: a minimum set of variables related to the performace of the component must be identified by the PT. The expected behaviour of the component with respect to the variation of those variables must be also identified. A corresponding set of tests must be defined to check that the component behaves as expected.

Release Criteria: If there are scalability tests defined, they must be executed in every major component release where there are substantial functionality changes.

Standard compliance/conformance tests

Type of Test: Optional.

Definition: Standard compliance/conformance tests are meant to verify that a software conforms or complies to a specific standard. In the context of EMI, some examples are `Glue v.2.0` or `SMRv2`.

Required Coverage: Standard compliance/conformance tests must be defined for at least the basic functionality provided by the adoption of the standard.

Release Criteria: compliance/conformance tests must be executed only in those component releases where the standard is adopted for he first time.

Integration tests

Type of Test: Mandatory.

Definition: Integration tests are meant to verify that a software component is able to operate with other EMI components.

Required Coverage: Not clear yet.

Release Criteria: The execution of integration tests is triggered by the release manager and relies on the use of the EMI testbed. Integration tests are the final stage before going to Production and they are done on certified software components. The exact context and procedure to run and monitor the result of these tests still needs to be understood. A draft plan is available here: EmiIntegrationTests.

Test Plan

The Test Plan must describe the strategy that will be adopted for testing the software component. It should be written by the PTs.

The following template describes the information that must be present: Test Plan template.

QC is gathering all the EMI test plans under the QC Test Plan twiki. We encourage PTs to add their Test Plans in this twiki.

Test Report

The Test Report must contain the result of the tests specified in the Test Plan that have been executed on a component release. It should be written by the PTs.

The following template must be used: Test Report template

This report must be included in the corresponding component release item tracked in Savannah, as explained in the Change Management Policy.

Testing Process

PTs are responsible for testing their components.

Testing a new version of a component has to be **ALWAYS** done for each EMI major release and supported platform. Exceptions may apply for Emergency Releases that willl be discussed at the EMT. For example, imagine EMI-1 and EMI-2 are the two major EMI releases supported, and you have to do a minor release of component `emi-sherpa` which fixes some bugs present in both EMI-1 and EMI-2. Imagine that EMI-1 is supported in SL5 and EMI-2 is supported in SL6 and Debian 6. You will need to test:

- `emi-sherpa` for EMI-1 SL5
- `emi-sherpa` for EMI-2 SL6
- `emi-sherpa` for EMI-2 Debian 6

The pre-condition for starting the testing phase is that the component buils without any errors, passes all the unit-tests and has all the needed packages registered in the ETICS permanent repository. This is the repository where packages are stored once the corresponding ETICS configurations have been locked and built. See the Configuration and Integration Policy for more details.

The input for testing a component is the Test Plan of the component.

The output of the testing phase is the Test Report. This report must be included in the corresponding component release item tracked in Savannah, as explained in the Change Management Policy.

The testing process can be summarised in the following steps:

- Perform deployment tests
- Perform system tests
- Check and update RfCs
- Write the Test Report
- Upload the Test Report in the corresponding item in the EMI release tracker.

Contacts

- Gianni.Pucciani@cernNOSPAMPLEASE.ch
- Maria.Alandes.Pradillo@cernNOSPAMPLEASE.ch
- jozef.cernak@upjsNOSPAMPLEASE.sk

References

- R. Patton, Software Testing (second edition) SAMS Publishing 2006 ISBN:0-672-32798-8
- Software testing http://en.wikipedia.org/wiki/Software_testing
- 741968.pdf: IEEE Standards Description: 829-1998

Progress

- *2 September 2010*: First draft prepared
- *7 September 2010*: SA2 meeting discussing the draft
- *7 September 2010*: Document updated according to feedback from meeting
- *1 December 2010*: Minor updates plus removed the section 'Middlewares documentation'.
- *14 February 2011*: The following changes have been implemented:
  - Change `guidelines` with `policy`.
  - Change `release candidate` with `component release` to harmonise with Change Management Policy.
  - Change `bug` with `RfC` to harmonise with Change Management Policy.
  - Reorganise the name of the main sections:
    - `general testing guidelines` to `tests to be performed to certify an EMI component release`
    - `certification on a release candidate` to `component release certification process`
    - Added a section for Static Code Analysis
  - Reorganised `Tests to be performed to certify an EMI component release`
  - Reorganised `Software Verification and Validation Report` and `Software Verification and Validation Plan` after feedback collected from UNICORE. A new template is now included and both twikis have been modified to match with the new template. The relevant sections in this twiki have been updated as well to be aligned to the changes in the twikis.
- *21 February 2011*: The following changes have been implemented:
  - Use `testing` instead of `certification` since `certification` is going to be used with a different meaning within EMI.
  - Use `Test Plan` instead of `Software Verification and Validation Plan` and `Test Report` instead of `Software Verification and Validation Report`.
  - Applied feedback from ARC: Added whether tests are mandatory or optional, added section for definitions, added required coverage and release criteria. Reorganised Testing process to avoid duplication.

This topic: EMI > EmiSa2TestGuidelines
Topic revision: r1 - 2011-02-22 - MariaALANDESPRADILLO