EUROPEAN MIDDLEWARE INITIATIVE

EUROPEAN

MIDDLEWARE INITIATIVE

TESTING POLICY

Document Version:	**3.0**
Date:	**22.08.2011**
Document Status:	**APPROVED**

This work is co-funded by the EC EMI project under the FP7 Collaborative Projects Grant Agreement Nr. INFSO-RI-261611.

Table of Contents

EMI Testing Policy

1. Introduction

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

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

2. Definitions

- **Product**: EMI software product as specified in DNA1.3.2 - Technical Development Plan [R1].
- **Release Task**: Savannah Release Task [R2] where the new version of the product is tracked.

3. 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 product.

Each section describes:

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

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

For more information on how to use these plugins, visit the ETICS Plugins web page [R13].

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.

3.2. Unit tests

Type of Test: Mandatory.

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: Since ETICS doesn't have a plugin to calculate unit test code coverage, there is no code coverage requirement.

Release Criteria: Unit tests must be run for any new product release. Report about which unit tests have been run and their result in your test report. If you already calculate the code coverage for your software product with any other tool, please include the code coverage % in the test report as well.

3.3. Deployment tests

Type of Test: Mandatory (see sections 3.3.1, 3.3.2. and 3.3.3 for more details).

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

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 product release.

3.3.1. Installation tests for EMI 1

EMI 1 supported package formats are tar.gz, src.tar.gz, rpm, src.rpm as stated in the Packaging policy [R3]. Installation tests must be performed in EMI 1 for at least rpm packages.

The installation tool used in EMI 1 for rpm packages is YUM.

Installation tests must specify the necessary YUM commands and YUM repositories needed to install the software product.

- The YUM commands must be in the form of:
  - `yum install/update metapackage_name`, for products that have an associated metapackage. Details on how to define metapackages can be found in the Packaging Policy [R3].
  - `yum install/update metapackage_name`, for the rest of the products, i.e. libraries or utilities.
- The YUM repositories needed to install the software product are:
  - ETICS YUM repository of the `emi_R_X_rc` project configuration, which is the project configuration containing all the new packages scheduled for this release plus the production versions of the packages that do not change.
  - Some of the following external repositories may be also needed, depending on the software product:
    - EPEL repository [R4]
    - SL5 OS repository [R5]
    - EGI trustanchors repository [R6]

3.3.2. Installation tests for EMI 2

This section will be completed in the upcoming months.

3.3.3. Configuration tests

The configuration tool depends on the specific product. Deployment tests must also define the configuration commands, configuration variables and/or configuration files, if any, needed to be able to configure the product. Depending on the product, one or more of the following have to be presented:

- A configuration command is the command that needs to be run to configure your software product. 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`.
- A configuration file or template is a file containing the minimal set of configuration items for the product to be up and running.

Configuration tests must be run after the corresponding installation test (See section 3.3.1 and 3.3.2) for both clean and upgrade installations.

3.4. System tests

System tests cover the majority of the tests for a new product version 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.

3.4.1. Basic functionality tests

Type of Test: Mandatory.

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

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

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

3.4.2. 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. A regression test must have an identifier, ideally corresponding to the associated RfC (optional).

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

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

3.4.3. Performance tests

Type of Test: Optional.

Definition: Performance tests are tests that aim at verifying the performance of a product, 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 product is able to meet those requirements.

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

3.4.4. Scalability tests

Type of Test: Optional.

Definition: Scalability tests are meant to verify that the product 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 product must be identified by the PT. The expected behaviour of the product with respect to the variation of those variables must be also identified. A corresponding set of tests must be defined to check that the product behaves as expected.

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

3.4.5. 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 product releases where the standard is adopted for he first time.

3.4.6. Inter-component tests

Type of Test: Mandatory.

Definition: Inter-component tests are meant to verify that a software product is able to operate with other EMI products. Inter-component tests must specify which external EMI products to the one under test are needed. Inter-component tests are written by PTs and executed in the EMI Testbed.

Required Coverage: Not clear yet.

Release Criteria: The execution of inter-component tests is triggered by the release manager and relies on the use of the EMI testbed [R12]. Intere-component tests are the final stage before going to Production and they are done on certified software products. The exact context and procedure to run and monitor the result of these tests still needs to be understood.

4. Test Plan

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

The following template describes the information that must be present: Test Plan template [R7].

QC is gathering all the EMI test plans under the QC Test Plan [R8] twiki. We encourage PTs to add their Test Plans in this twiki.

5. Test Report

The Test Report must contain the result of the tests specified in the Test Plan [R7] that have been executed on a product release. It should be written by the PTs.

The following template must be used: Test Report template [R9].

This report must be attached in the corresponding release task as explained in the Change Management Policy [R10].

6. Testing Process

PTs are responsible for testing their products.

Testing a new version of a product 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 product `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 product builds 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 EMI Release Checklist [R11] for more details.

The input for testing a product is the Test Plan [R7] of the product.

The output of the testing phase is the Test Report [R9]. This report must be attached in the corresponding release task, as explained in the Change Management Policy [R10].

7. Contacts

EMI SA2

8. Table of References

Reference	URL
R1	EMI Technical Development Plan https://twiki.cern.ch/twiki/pub/EMI/DeliverableDNA132
R2	EMI Release Tracker https://savannah.cern.ch/task/?group=emi-releases
R3	EMI Packaging Policy https://twiki.cern.ch/twiki/bin/view/EMI/EmiSa2PackagingPolicy
R4	EPEL repository http://fedoraproject.org/wiki/EPEL
R5	SL5 OS repository https://www.scientificlinux.org/distributions/5x/
R6	EGI trustanchors repository https://wiki.egi.eu/wiki/EGI_IGTF_Release
R7	EMI Test Plan Template https://twiki.cern.ch/twiki/bin/view/EMI/EMITestPlan
R8	EMI QC Test Plan twiki https://twiki.cern.ch/twiki/bin/view/EMI/QCTestPlan
R9	EMI Test Report Template https://twiki.cern.ch/twiki/pub/EMI/EMITestReport/EMI_Test_Report_template.txt
R10	EMI Change Management Policy https://twiki.cern.ch/twiki/bin/view/EMI/EmiSa2ChangePolicy
R11	EMI Release Checklist https://twiki.cern.ch/twiki/bin/view/EMI/EMIReleaseChecklist
R12	EMI testbed https://twiki.cern.ch/twiki/bin/view/EMI/TestBed
R13	ETICS Plugins https://tomtools.cern.ch/confluence/display/EMITOOLS/Plugins

9. Logbook

Details Hide Details

v3.0 (approved on 22.08.2011)

- *22.08.2011*: Added feedback from ARC: improved Deployment section to differentiate from EMI 1

and EMI 2 (which will also include Debian). Given more details on configuration tests.
- *05.08.2011*: Added link to ETICS plugins web page.
- *04.08.2011*:
    - Added feedback from developers.
    - Review and update all sections.
    - Update Test Plan and Test Report twikis.
- *02.08.2011*:
    - Added section numbers.
    - Changed `component` with `product` where applicable.
- *03.03.2011*: Feedback from the training:
    - Improve section on unit tests. not clear what it is requested.
    - Improve Test Report template to clarify that ETICS configuration name refers to the subystem.
- *02.03.2011*: Changed `Integration tests` with `Inter-component tests`.

v2.0 (approved on 28.02.2011)

- *22.02.2011*: Meeting with PEB. Feedback given on unit tests (should be mandatory), improve definition of integration tests, improve test report template, other general feedback.
- *21.02.2011*: 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.
- *14.02.2011*: 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.

v1.0 (approved on 03.12.2010)

- *01.12.2010*: Minor updates plus removed the section 'Middlewares documentation'.
- *07.09.2010*: Document updated according to feedback from meeting
- *07.09.2010*: SA2 meeting discussing the draft
- *02.09.2010*: First draft prepared

Appendix: EMI Test Plan Template

1. Introduction

This twiki page presents a template for the EMI Test Plan. PTs must use the template to make sure the following sections are present in the test plan for their EMI software products.

2. EMI Product Description and Version

The Test Plan must include a description of the EMI product. It could be a link to a service reference card, when it exists.

Sice the Test Plan is a document that will evolve and change with newer versions of the product, it must include the most recent product version for which the Test Plan is suitable. This means that the Test Plan contains all the necessary tests to test such version.

3. Unit tests

The Test Plan must describe the tools used to execute unit tests. Note that there is no need to describe individual tests, only give an overview of the strategy choosen to run unit tests. Unit tests must be automatically executed for any product release.

4. Deployment tests

The Test Plan must describe deployment tests. Deployment tests include installation and configuration tests.

Deployment tests must be done for at least rpm packages for EMI 1. For EMI 2, this document will be updated in the upcoming months.

4.1. Installation tests for EMI 1

Installation tests must define:

- YUM repositories needed to install the product version under testing.
- YUM command needed to test a clean installation, that is an installation of the product version under testing a clean machine.
- YUM command needed to test an upgrade installation, if applicable. That is an installation of the product version in production, using the EMI production repository, on a clean machine. And then an upgrade to the product version under testing.

4.2. Installation tests for EMI 2

This section will be updated in the upcoming months.

4.3. Configuration tests

Configuration tests must define the configuration commands and configuration files needed to configure the product under testing. The configuration test must be done for both clean and upgrade installations.

5. System tests

The Test Plan must describe system tests. The following sections describe some of the system tests to be included in the test plan.

5.1. Basic functionality tests

The Test Plan must describe basic functionality tests in the following way:

- For each functionality present in the product:
  - ♦ Description of the functionality to be tested (describing what it does and not how) must be included. If a test is available to verify and validate this functionality, this must be clearly stated as `implemented`, otherwise it should say `not implemented`.
  - ♦ In case the functionality is covered by a test:
    - ◊ Test for normal workflow: description of the test that proves that the functionality is present and works as expected. The following must be also included:
      - · Input needed for the test.
      - · Criteria for considering the test succesful (`PASS`) or failed (`FAIL`).
    - ◊ Test for error cases: description of the test that proves that the proper error messages are returned in anomalous scenarios.The following must be also included:
      - · Input needed for the test to trigger the anomalous scenario.
      - · Criteria for considering the test succesful (`PASS`) or failed (`FAIL`).

Implementing the missing functionality tests is an area for improvement that the PTs should consider in future versions of the test plan.

5.2. Regression tests

The Test Plan shall describe regression tests for software defects (`bugs`) tracked in RfCs relevant to the product.

Regression tests must contain at least the following information:

- Regression Test unique ID.
- If the Regression Test is not the RfC unique ID, then, include as well the RfC unique ID.
- Description of the test that will prove that the RfC has been properly implemented and that the problem it fixes is indeed not present any more in the product.
- Input needed for the test.
- Criteria for considering the test succesful (`PASS`) or failed (`FAIL`).

5.3. Performance and scalability tests

The Test Plan shall describe the strategy to implement load, stress, performance and scalability tests. If these tests are present, they should contain at least the following information:

- Description of the test.
- Description of the scenario under which the product will be tested. This should include parameters like time, load, stress conditions, etc.
- Input needed for the test. This should include scripts or any other tools used to recreate the special conditions needed to execute the test.
- Log files, memory usage, average response time or any other mean that proves the expected behaviour of the product.
- Criteria for considering the test succesful (`PASS`) or failed (`FAIL`).

5.4. Standard compliance and conformance tests

The Test Plan shall describe the strategy to implement standard compliance and conformance tests. If these tests are present, they should contain at least the following information:

- Description of the test.
- Description of the standard to be tested.
- Input needed for the test.
- Criteria for considering the test succesful (`PASS`) or failed (`FAIL`).

5.5. Inter-component tests

The Test Plan shall describe the strategy to implement inter-component tests. If these are present, they should contain at least the following information:

- Description of the test.
- EMI products involved in the test.
- Input needed for the test.
- Criteria for considering the test succesful (`PASS`) or failed (`FAIL`).

Appendix: EMI Test Report Template

1. Introduction

This twiki page presents a template for the Test Report produced after testing new releases of EMI software products.

2. Test Report Template

The following template must be used by PTs to report about the result of their testing. The use of a common template will simplify the work of the QC activity.

```
********************************
EMI Test Report Template
********************************

- Product:

- Release Task:

- ETICS Subsystem Configuration Name:

- VCS Tag:

- EMI Major Release:

- Platform:

- Author:

- Date:

- Test Report Template : v. 3.0

*************
Summary
*************

1. Deployment tests:
   1.1. Clean Installation - PASS/FAIL
   1.2. Upgrade Installation - PASS/FAIL/NA
2. Static Code Analysis - PASS/FAIL/NA
3. Unit Tests Execution - YES/NO
4. System tests:
  4.1. Functionality tests - PASS/FAIL
  4.2. Regression tests - PASS/FAIL/NA
  4.3. Standard Conformance tests - PASS/FAIL/NA
  4.4. Performance tests - PASS/FAIL/NA
  4.5. Scalability tests - PASS/FAIL/NA

REMARKS:

************************** Detailed Testing Report ************************************

1. Deployment log
************************

1.1. Clean Installation
----------------------------
- YUM Testing Repo file contents:
- YUM Install command:
- YUM log:
```

- Configuration log:

1.2. Upgrade Installation

- YUM Production Repo file contents:
- YUM Install command:
- YUM Testing Repo file contents:
- YUM Upgrade command:
- YUM log:
- Configuration log:

2. Static Code Analysis
****
- URL where static code analysis results can be accessed

3. Unit Tests
****
- URL pointing to the results of the Unit Tests.
- Code Coverage %, if available.

4. System tests
****
- URL where the tests/testsuite can be accessed:
- URL where the test results can be accessed:

OR

please, use the template below to include the test results in this document:

---------- System Test Summary ----------
For each Basic Functionality test, please include the following summary:
* Description of the test:
* Result: PASSED/FAILED:

For each Regression test, please include the following summary:
* Description of the test:
* Test Unique ID/RfC unique ID:
* Result: PASSED/FAILED:

For each of Performance and Scalability tests, please include the following summary:
* Description of the test:
* Description of the specific context:
* Result: PASSED/FAILED:

For each Standards Compliance/Conformance test, please include the following summary:
* Description of the test:
* Adopted Standard:
* Result: PASSED/FAILED:

4.1. Basic Functionality tests
****
For each test:
COPY & PASTE TEST OUTPUT.

4.2. Regression tests
****
For each test:
COPY & PASTE TEST OUTPUT.

4.3. Standard Conformance tests
****
For each test:
COPY & PASTE TEST OUTPUT.

4.4. Performance tests
****

```
For each test:
COPY & PASTE TEST OUTPUT.

4.5. Scalability tests
*************************
For each test:
COPY & PASTE TEST OUTPUT.

---------------End of System Test Summary ------------------
```

3. Test Report in detail

3.1. Product

The Test Report must specify the name of the tested product as specified in the DNA1.3.2 - Technical Development Plan.

3.2. Release Task

The Test Report must specify the unique id for the Release Task in the EMI release tracker where the new product version is being tracked.

3.3. EMI major release and Platform

The Test Report must specify the EMI major release and platform of the new product version.

3.4. Author

The Test Report must contain the name of the PT members who have performed the testing of the new product version.

3.5. Summary

In order to have a quick summary of the results of the tests, PTs must fill in the summary section of the template, where `PASS` means all the tests have passed; `FAIL` means that at least one test has failed; `NA` means the test is not applicable for the product.

Remarks about the result of the tests can be added in the summary.

3.6. Deployment log

3.6.1. Clean Installation

The Test Report must contain a test for a clean installation, that is an installation of the new product version including the change on a clean machine.

For EMI 1 don't forget to include:

- the testing yum repo file with the details of the yum repository that is used to do the installation.
- the yum command used to install the product.
- the yum output.
- configuration output (YAIM log or any other tool or set of commands run to configure the product).

For EMI 2, specific instructions will be added in this section in the upcoming months.

Verify that the packages included in the change have been correctly installed.

3.6.2. Upgrade installation

The Test Report must contain a test for an upgrade from production, if applicable. That is an installation of the product from the production repository on a clean machine, and then an upgrade to the new product version including the change.

For EMI 1 don't forget to include:

- the production yum repo file with the details of the yum repository that is used to do the installation.
- the yum command used to install the production version of the product.
- the testing yum repo file with the details of the yum repository that is used to do the upgrade.
- the yum command used to upgrade to the new product version.
- the yum output.
- configuration output (YAIM log or any other tool or set of commands run to configure the product)

For EMI 2, specific instructions will be added in this section in the upcoming months.

Verify that the packages included in the change have been correctly updated.

3.7. Static Code Analysis

If the product has executed Static Code Analysis plugins in ETICS, include a link to the plugin results. If you have used any other tool, you can optionally include a URL with the results as well.

3.8. Unit Tests

The Test Report must report on the results of Unit Tests. Include a URL to the results of the Unit Tests. You can optionally report about the unit test coverage in % for the product.

3.9. System tests

The Test Report must report on the results of the system tests defined in the Test Plan of the product.

Don't forget to include:

- URL of the tests/testsuites (with the revision number in case of tests maintained on a VCS)
- When tests are automated, a link to the tool-specific report, otherwise, please specify:
    - ♦ a short description of the tests run (1-2 lines should be enough)
    - ♦ the outcome of the tests, that is either PASSED or FAILED.
    - ♦ specific command to run the test and output of the test.

3.9.1. Basic functionality tests

The Test Report must contain the result of running the Basic Functionality tests described in the test plan of the product.

3.9.2. Regression tests

The Test Report must contain the result of running the existing tests associated to software defects (`bugs`) tracked in specific RfCs, as it should be specified in the test plan of the product.

Don't forget to include:

- The RfC unique ID for which the regression test is written.

3.9.3. Performance and scalability tests

The Test Report may contain the result of running performance and scalability tests, as defined in the test plan of the product.

Don't forget to include:

- A description of the context, user case or scenario for the tests. This should include the amount of time used to run the test, stress conditions, interactions with other services, etc.
- any relevant log, CPU usage statistics, etc that proves the good performance of the service.

3.9.4. Standard conformance tests

The Test Report may contain the result of running standard compliance/conformance tests, as defined in the test plan of the product.

Don't forget to include:

- a reference to the standard adopted.

4. Test Report Name

In order to automate the QC task, the name of the Test Report attached to the release task must contain the words `test` and `report`. The case is not sensitive, so both upper and lower case is allowed. Some valid examples are: `VOMS_test_report_1.1.txt` or `ReportTest_CREAM1.2.txt`.

5. Test Report Format

In order to automate the QC task, at least the summary section of the Test Report must be provided in txt format and must be attached to the release task with the name specified as described in the previous section 4. Test Report Name. The remaining part of the test report can be provided in any other format as an extra attachment if this is more convenient for the PT, but in that case, the name should not contain the words `test` nor `report`.

The summary section we are referring to is the one below:

```
*************
Summary
*************

1. Deployment tests:
   1.1. Clean Installation - PASS/FAIL
   1.2. Upgrade Installation - PASS/FAIL/NA
2. Static Code Analysis - PASS/FAIL/NA
3. Unit Tests Execution - YES/NO
4. System tests:
  4.1. Functionality tests - PASS/FAIL
  4.2. Regression tests - PASS/FAIL/NA
  4.3. Standard Conformance tests - PASS/FAIL/NA
  4.4. Performance tests - PASS/FAIL/NA
  4.5. Scalability tests - PASS/FAIL/NA

REMARKS:
```