Table of Contents

Certification and Testing Guidelines

Current approved version of these guidelines
link

Introduction

This document provides guidelines on the following topics:

- general testing guidelines: which kind of tests to perform and when,
- how to write a Software Verification and Validation Plan,
- how to perform certification on a release candidate,
- how to write a Software Verification and Validation Report.

To clarify the terminology used, certification is the action done by a Product Team on a release candidate in order to "certify" that the software has been verified (the software has been correctly built) and validated (the right software has been built). Verification and validation involve several kind of testing that will be explained in this document.

General testing guidelines

We will use the following classification for different type of tests:

- Unit tests
- Deployment tests
- Functionality tests
- Regression tests
- Performance tests
- Scalability tests

Apart from unit tests, we strongly encourage PTs to have a clear distinction between developers and testers. When this is not possible, due to limited manpower, developers of one component should act as testers for different components.

The first level of testing occurs when a subsystem in ETICS is being built. The build process should trigger the execution of ETICS plugins for static code analysis. Currently the list of plugins available for static code analysis are:

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

The execution of these plugins should provide a quick and early feedback on the quality of the code committed on the VCS.

Unit tests

Unit tests are meant to test the correctness of individual units or a group of related units. The definition of unit can vary with the programming language used, but in general the method is quite standard. Tools like CPPUnit, JUnit, PyUnit are generally used and they provide the necessary documentation to get started with unit tests using your favorite programming language.

Within EMI the goal of increasing unit testing is recognized. A high unit tests code coverage not only improves the reliability of the software, but also helps to have software that will be easier to maintain. Unit test code coverage should be measured. The target value can be different for different components, but in general:

- unit tests code coverage of [0,25%] should be improved,
- when reaching coverage around 80%, the decision to continue developing unit tests should depend on an ad-hoc analysis of the component under test.

Metrics for unit tests code coverage should be collected and periodically reported. The SA2.3 activity is in charge of defining the metrics and their format. It should be possible to run unit tests within Etics, and have code coverage metric reports. However, currently there is no plugin in ETICS to measure unti tests code coverage.

When unit tests are performed and their coverage measured, the results should be reported in the Software Verification and Validation Report when a release candidate is being certified.

Deployment tests

Deployment tests verify that the component can be properly installed and configured on all the supported platforms. The deployment (installation + configuration) method can vary; the test should follow as close as possible the method used by the end users to deploy the software. At the moment there is no recommended tool for this kind of tests.

Deployment tests should be done always on a release candidate, and their execution should be reported in the Software Verification and Validation Report together with the output of the commands executed.

Functionality tests

In this context functionality tests cover the majority of the tests done on a component to verify it's correct (according to specifications) behavior. System tests and Integration tests (defined here as tests that involve the interaction among different components) are considered part of functionality tests.

Functionality tests should be performed according to the Software Verification and Validation Plan of the given component. Depending on the component, it may or may not include interaction with other components, developed by the same or a different product team. When the interaction with an external component is needed, the testbed provided by SA2 with reference services can be used. Information on how to join and use the testbed are available here: https://twiki.cern.ch/twiki/bin/view/EMI/TestBed.

When the component is released with a CLI or API, these interfaces have to be properly tested: all the commands/methods have to be tested, with expected and erroneous input data.

Product teams should aim at automating as much as possible the functionality tests. The execution of the tests should be reported in the Software Verification and Validation Report, and it should include at least a short description of the test performed and the outcome (PASSED or FAILED). All the tests performed on the release candidate should pass.

Functionality tests should be performed always on a release candidates; exceptions can be done for the release of urgent bug fixes and special occasions agreed within the EMT.

Regression tests

In this context regression tests are tests that are meant to verify specific bug fixes. A regression tests can be associated to a bug (RfC) reported in the bug (defect) tracker. A Product Team should aim at providing regression tests whenever the bug fix can be automatically (with a script) verified.

The execution of regression tests should be automated, and may or may not be part of the test suite used for unit and functionality tests. When regression tests are distinguished, their execution should be highlighted in the Software Verification and Validation Report.

Regression tests should be performed always on a release candidate; exceptions can be done for the release of urgent bug fixes and special occasions agreed within the EMT.

Standard Compliance/Conformance tests

This type of test is meant to verify that a software comply/conform to a specific standard (e.g Glue2, SMRv2, Posix). Requirements for standards are different for different components. When a component has to follow a standard, compliance/conformance tests have to be run, and their execution explicitly mentioned in the Software Verification and Validation Report.

Performance tests

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.

The details of how to execute performance tests on a component and their success/fails criteria should be described in the Software Verification and Validation Plan of the given component.

Performance tests should be done only when important changes in the component are being released and can be targeted to the study of specific parts of the component.. In general, every major release should include in the certification report a section for the performance tests, and the results should be compared with the ones of the previous release.

Scalability (load and stress) tests

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.

The details of how to perform scalability tests on a component and their success/fails criteria should be described in the Software Verification and Validation Plan of the given component.

Scalability tests should be done only when important changes in the component are being released and can be targeted to the study of specific parts of the component. In general, every major release should include in the

certification report a section for the scalability tests, and the results should be compared with the ones of the previous release.

Software Verification and Validation Plan

The Software Verification and Validation Plan (i.e test plan) document should describe the strategy that will be adopted in testing each software component.

Author: Product teams and PTB.

It should contain details on at least the following type of tests: Deployment tests, Functionality tests, Regression tests. It is desirable however that it includes also details on how to do Performance and Scalability tests. The main purpose of this document is to provide test developers with clear instructions on what has to be tested and how.

The following template can be used as a starting point: Software Verification and Validation Plan template.

For the moment, no recommendations are present concerning the format; pdf, doc and twikis are all accepted.

Certification of a Release Candidate

When preparing a release candidate of a component, each product team is responsible for performing a certification to verify and validate their software. This phase should produce a report (explained in the next section) to give to the release manager together with the software packages.

In general, the certification involves two main steps:

- deploying the component,
- testing the component.

This has to be done on each supported platform.

A pre-condition for starting the certification phase is that the software has be correctly built and unit-tested.

The deployment involves the installation of the software through a set of packages (rpm or deb), and the configuration of the software. The deployment has to be done both on a clean (with only the OS installed) machine and as an upgrade on a machine where the current production version is installed. After the installation, the presence of the correct packages has to be verified.

Testing the component means running a set of predefined tests (according to the Software Verification and Validation Plan for the given component) on the just deployed instances (clean and upgrade). Besides running the tests, the "certifier" has also to check that each bug fix released with this release is properly working. When the software release includes new features, these features have to be properly tested and the test suite updated accordingly.

When testing a component involves the use of another component from a different product team, the EMI SA2 testbed can be used. This testbed should make production version of each component available for testing. The available of a larger testbed for special testing activity can be organized, ad-hoc, by SA2.

Software Verification and Validation Report

The Software Verification and Validation Report should contain the result of the tests specified in the Software Verification and Validation Plan that have been executed on a release candidate. This document should be presented when a release candidate is given to the release manager.

Author: Product teams

It should contain at least the following information:

- Change Id of the change being certified (could change depending on release guidelines)
- Name of the component (could change depending on release guidelines)
- Author(s) and E-mail contacts
- Outcome: certified/rejected/in progress
- Details on clean installations and upgrade from production
- Test report containing:
    - short description of the tests being executed
    - outcome of the tests, namely PASSED or FAILED
    - regression tests executed and their outcome
- When needed, report on Performance and Scalability tests

The following template can be used as a starting point: Software Verification and Validation Report template

For the moment, no recommendations are present concerning the format; pdf, doc and twikis are all accepted.

Contacts

- Gianni.Pucciani@cernSPAMNOT.ch
- jozef.cernak@upjsSPAMNOT.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'.

This topic: EMI > EmiSa2CertTestGuidelines
Topic revision: r17 - 01-Dec-2010 - 16:12:27 - GianniPucciani