idm@F4E UID / VERSION
3S94CT / 1.1

VERSION CREATED ON / STATUS
04 January 2017 / Approved

EXTERNAL REFERENCE

Supplier Document

D4 MARTe Verification and Validation Plan

Under the Specific Contract no. F4E-OFC-361-06, with subject "Fast plant controller prototype", this document is deliverable D4 MARTe Verification and Validation Plan.

The scope of this document applies to the software development activities of the MARTe project. It details the verification processes applied by GTD in the framework of this project to satisfy the software verification process ...

Approval Process			
	Name	*Action*	*Affiliation*
Author	**Herrero I.**	**04 January 2017:signed**	
Co-Authors			
Reviewers			
Approver	**Cabrita Neto A.**	**06-Jan-2017: approved**	**ITERD**
RO: Cabrita Neto Andre (F4E)			
Read Access	**LG: GTD team, LG: OFC-361-06-CCFE, AD: IDM_F4E, AD: F4E-A40_HEAD, AD: I-CODAC, AD: IDM IE-TS-CO-00 CODAC, GG: IAC, GG: IAS Audit on Document Management, project administrator, RO**		

Orig. Document MD5#: CBA8C841B8599365FF33F64167A7170F

Change Log			
D4 MARTe Verification and Validation Plan (3S94CT)			
Version	***Latest Status***	***Issue Date***	***Description of Change***
v0.0	In Work	02 June 2015	
v1.0	Approved	02 June 2015	First version of the document.
v1.1	Approved	04 January 2017	Coding unification among QA documents Add “FlexeLint for C/C++” as software used in V&V environment Update flags Update date format on for audit review report.

Identification of the document			
Document Reference	F4E-OFC-361-06 D4	Revision	1.1
F4E Reference	F4E_D_3S94CT	F4E TRO	André C. Neto
F4E Customer Reference	N/A		
Date	2017-01-04		
Supplier	GTD SISTEMAS DE INFORMACION		
Graded Quality level	Class 3 – Any safety related item (SR) or non safety related item (NSR) whose failure could result in MODERATE impact.		

MARTe Verification and Validation Plan

Fast plant controller prototype (F4E-OFC-361-06)

Summary

Under the Specific Contract no. F4E-OFC-361-06, with subject "Fast plant controller prototype", this document is deliverable D4 MARTe Verification and Validation Plan.

The scope of this document applies to the software development activities of the MARTe project. It details the verification processes applied by GTD in the framework of this project to satisfy the software verification process objectives: software testing, reviews, or traceability.

	Written by	Revised by	Approved by
Name	Ivan Herrero	Ivan Herrero	Javier Varas
Signature			
Date	2017-01-04	2017-01-04	2017-01-04

Contact person GTD: javier.varas@gtd.eu
Phone: +34 93 493 93 00

DISSEMINATION

Distributed to	Copies	Means
Project Team	1	e-mail
Fusion for Energy	1	Electronic

SUMMARY OF MODIFICATIONS

Edition	Date	Chapter	Modification	Author/s
1.0	02/06/2015	All	Document creation	IH
1.1	04/01/2017	Headers V&V Environment Compiler assumptions Annexes	Coding unification among QA documents Add “FlexeLint for C/C++” as software used in V&V environment Update flags Update date format on for audit review report.	IH

CONTENTS

CONTENTS 3

0 Introduction 5

0.1 Purpose and scope of the document 5

0.2 Relationship to other plans 5

0.3 Applicable and reference documents 5

0.4 Definitions, acronyms and abbreviation 6

1 Organization 7

2 Independence 7

3 Verification and validation methods 7

3.1 Software analysis 7

3.1.1 Requirements review process 7

3.1.2 Architecture and design review process 8

3.1.3 Code and documentation review process 8

3.2 Software testing 9

3.2.1 Unit test review 9

3.2.2 Integration tests review 10

3.2.3 Acceptance test review 10

4 Transition Criteria 11

4.1 States 11

4.1.1 Start 11

4.1.2 Requirements review 11

4.1.3 Architecture and design review 12

4.1.4 Code review 12

4.1.5 Unit test review 12

4.1.6 Integration test results review 12

4.1.7 Acceptance test results review 12

4.1.8 Bug found Stop validation 12

4.1.9 Error reported 12

4.1.10 Error fixed 12

4.1.11 End Product OK 12

5 Verification and validation environment 13

5.1 Software 13

5.1.1 Git and GitLab 13

5.1.2 Redmine 13

5.1.3 CppUnit / GoogleTest 13

5.1.4 Sonar 13

5.1.5 Jenkins 13
5.1.6 FlexeLint for C/C++ 13
5.2 Hardware 13
6 Compiler assumptions 14
7 Annexes 15
7.1 Audit review report 15
7.1.1 Report metadata properties 15
7.1.2 Report workflow properties 15
7.1.3 Report template 15

INDEX OF FIGURES

Figure 1. Relationship to other plans 5
Figure 2. Transition flow 11

INDEX OF TABLES

Table 1 Applicable documents 5
Table 2 Reference documents 5
Table 3 Definitions, acronyms and abbreviations 6
Table 4 Compiler assumptions 14
Table 5 Audit review report metadata 15
Table 6 Audit review report workflow 15

0 INTRODUCTION

0.1 Purpose and scope of the document

Under the Specific Contract no. F4E-OFC-361-06, with subject "Fast plant controller prototype", this document is deliverable D4 MARTe Verification and Validation Plan (VVP).

The scope of this document applies to the software development activities of the MARTe project. It details the verification processes applied by GTD and F4E in the framework of this project to satisfy the software verification process objectives: software testing, reviews, or traceability.

0.2 Relationship to other plans

The next figure shows the relationship between this plan and the other plans referenced in this document.

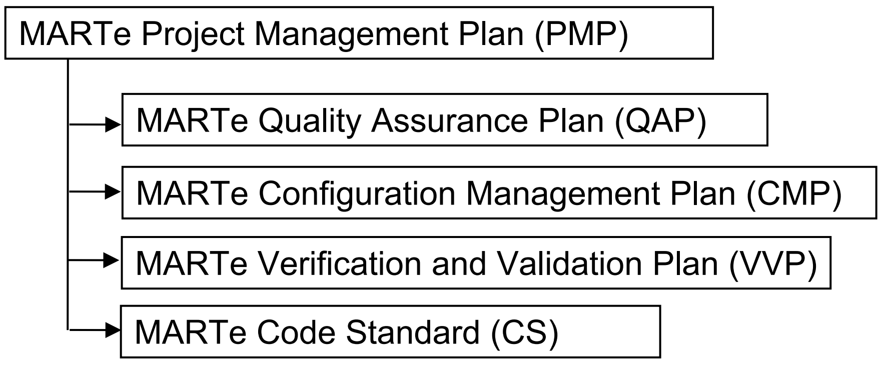
MARTe Project Management Plan (PMP)
MARTe Quality Assurance Plan (QAP)
MARTe Configuration Management Plan (CMP)
MARTe Verification and Validation Plan (VVP)
MARTe Code Standard (CS)

Figure 1. Relationship to other plans

Quality assurance, configuration management and verification/validation processes are known as integral processes. That is, they are present during the entire software lifecycle of MARTe.

0.3 Applicable and reference documents

Ref.	Code	IDM / Date	Ver.	Description
AD1	MARTe-P-PMP-01	F4E_D_2RBL9F	1.0	MARTe Project Management Plan

Table 1 Applicable documents

Ref.	Code	Date	Ver.	Description
RD1	---	----	---	---

Table 2 Reference documents

0.4 Definitions, acronyms and abbreviation

Term	Definition
AD	Applicable Document
API	Application Program Interface
ITER	International Thermonuclear Experimental Reactor
F4E	Fusion for Energy
IDM	ITER Document Management
MARTe	Multithreaded Application Real-Time executor
RD	Reference Document

Table 3 Definitions, acronyms and abbreviations

1 ORGANIZATION

The MARTe project manager is responsible for the verification process. He will guarantee that the verification and validation activities are executed accordingly to the software development process defined in [AD1]. In particular he will be responsible for checking that the transition criteria are observed.

The MARTe project manager is the sole responsible for appointing the team responsible for executing the verification activities detailed in this VVP: reviews, traceability and software testing. This team will be composed by MARTe team members and its composition may vary over time, provided that the criteria set on the following independence chapter is guaranteed.

2 INDEPENDENCE

The verification team shall not take part in the development process and shall be in charge of the verification process: this will ensure the satisfaction of the independence criteria between development and verification activities.

3 VERIFICATION AND VALIDATION METHODS

This chapter describes all the reviews and its associated logs, which have as objective checking that the software produced meets the system requirements.

All the logs created by the reviews will be put into an audit report, which will be created as an issue into the project's Redmine service. That issue will hold the report's metadata using the issue properties, while the content of the report itself will be put into the issue description following a specific template (see §7.1 Audit review report).

3.1 Software analysis

3.1.1 Requirements review process

This process aims to verify that all requirements have been correctly written and defined. The person responsible for this process will take special attention to new requirements, how they were written and whether they were correctly identified into the requirements document.

Once the review of requirements is made, the responsible will create a review log that explains the conformity and non-conformity of requirements. This review at least must describe the following information:

- Date of the review
- Person who did the review
- Version of requirements
- Result of review (pass/fail)
- List of non-conformity elements of requirements, if any exist.

This review log will be put as a chapter of the audit review report.

3.1.2 Architecture and design review process

This process aims to verify that the architecture and design follows are aligned with requirements and follow best practices. The responsible will pay special attention in possible redundant modules and global coherence of system. Deprecated modules shall be erased from the architecture document and design document.

Once the review of documents was made, the responsible will create a review log that explains the conformity and non-conformity of architecture document and design document. This review must describe at least the following information:

- Date of the review
- Person who did the review
- Version of architecture & design document
- Result of review (pass/fail)
- List of non-conformity elements of architecture and design, if any exist.

This review log will be put as a chapter of the audit review report.

3.1.3 Code and documentation review process

It is possible to focus this review on several key points. Automatic creation of code API documentation and a manual review of source code against good programming practices of software.

- Documentation

  Source code will have comments to produce API documentation automatically. It is necessary to manually review the API documentation automatically produced to verify that it is good enough as to develop new features with the MARTe library.

- Code review

  The use of good programming practices to make software can be highly beneficial and for the long term can facilitate software maintenance. It is for this reason that one should take extra steps to produce quality software. The code review is one of these actions.

  - Automated code review
    The objective is to check source code for compliance with a predefined set of rules and best practices. This test can be supported by tools in an automatic way for most of the rules.

  - Manual code review
    For those rules and best practices which cannot be analysed automatically by tools, a manual review is mandatory.

Once the code review is made, the responsible will create a review log that explains the conformity and non-conformity of software made by the team. This review must describe at least the following information:

- Date of the review
- Person who did the review
- Version of the source code

- Result of review (pass/fail)
- List of non-conformity elements of code, if any exist.

This review log will be put as a chapter of the audit review report.

3.2 Software testing

This chapter explains the testing methodology to validate the software made. In general, one can get support from test tools. However, although testing tools could perform test cases and then report the results into the reporting application, in this case only manual procedures will be used to test the software. This decision is justified because it is too expensive to use complex applications to manage a very short number of global tests.

The purpose of the testing is to validate that the software meets the functional requirements and that it works correctly. Moreover, these also help to ensure that the code is maintainable and properly documented. If the code created is not maintainable, errors will occur more often and repair costs are more expensive, up to the cost of having to redo part of the development.

3.2.1 Unit test review

In this phase, the responsible person has to review some quality facets of source code. This review is very important to assure the maintenance of source code. The responsible person will focus in the following aspects:

3.2.1.1 Coverage review

Coverage has two aspects to consider:

- **Static analysis**: The objective is to calculate how many public functions of the source code have unit tests defined (we assume black box unit testing). Code with a low percentage will be rejected.
- **Dynamic analysis**: The objective of this analysis is to calculate what percentage of code has been executed (we assume white/grey box unit testing). Code not executed shall be reviewed (this could be due to a poor developed unit testing or dead code). This test is supported by tools in an automatic way.

3.2.1.2 Functional tests review

The responsible person shall check the percentage of test successfully passed and the percentage of faulty tests. If the percentage is lower than a given level, all code will be rejected. If there are manual tests and automatic tests, the responsible person first will look for the automatic test percentage, before passing to the manual tests. The manual tests will only be executed if the automatic test percentage is deemed to be sufficient.

3.2.1.3 Review log

Once the review of each facet was made, the responsible will create a review log that explains the conformity and non-conformity of every type of tests. This review must describe at least the following information:

- Date of the review

- Person who did the review
- Version of unit tests.
- Result of coverage tests review
- Result of functional tests review
- Result of review (pass/fail)
- List of non-conformity elements and regressions, if any exist.

This review log will be put as a chapter of the audit review report.

3.2.2 Integration tests review

The integration tests will assure that all modules of source code work correctly without collisions between them.

Once integration tests analysis have been made, the responsible will create a review log that explains the conformity and non-conformity of tests. This review shall contain the following information:

- Date of the review
- Person who did the review
- Version of integration tests.
- Result of review (pass/fail)
- List of non-conformity elements and regressions, if any exist.

This review log will be put as a chapter of the audit review report.

3.2.3 Acceptance test review

This review is focused on acceptance tests. The acceptance test will focus in assuring that the software meets all requirements. A good set of functional and performance tests is necessary to assure this point. This is the last set of tests, before the release of software, so it must assure that all requirements are tested. As all other tests, the acceptance tests can be executed in a manual or automatic way. In all cases, results of tests will be registered into the reporting tool.

Once the acceptance test review was made, the responsible will create a review log that explains the conformity and non-conformity of acceptance tests. This review must describe at least the following information:

- Date of the review
- Person who did the review
- Version of acceptance tests.
- Result of review (pass/fail)
- List of non-conformity elements and regressions, if any exist.

This review log will be put as a chapter of the audit review report.

4 TRANSITION CRITERIA

In this diagram of state transition, it is possible to see all possible states of the verification and validation phases.

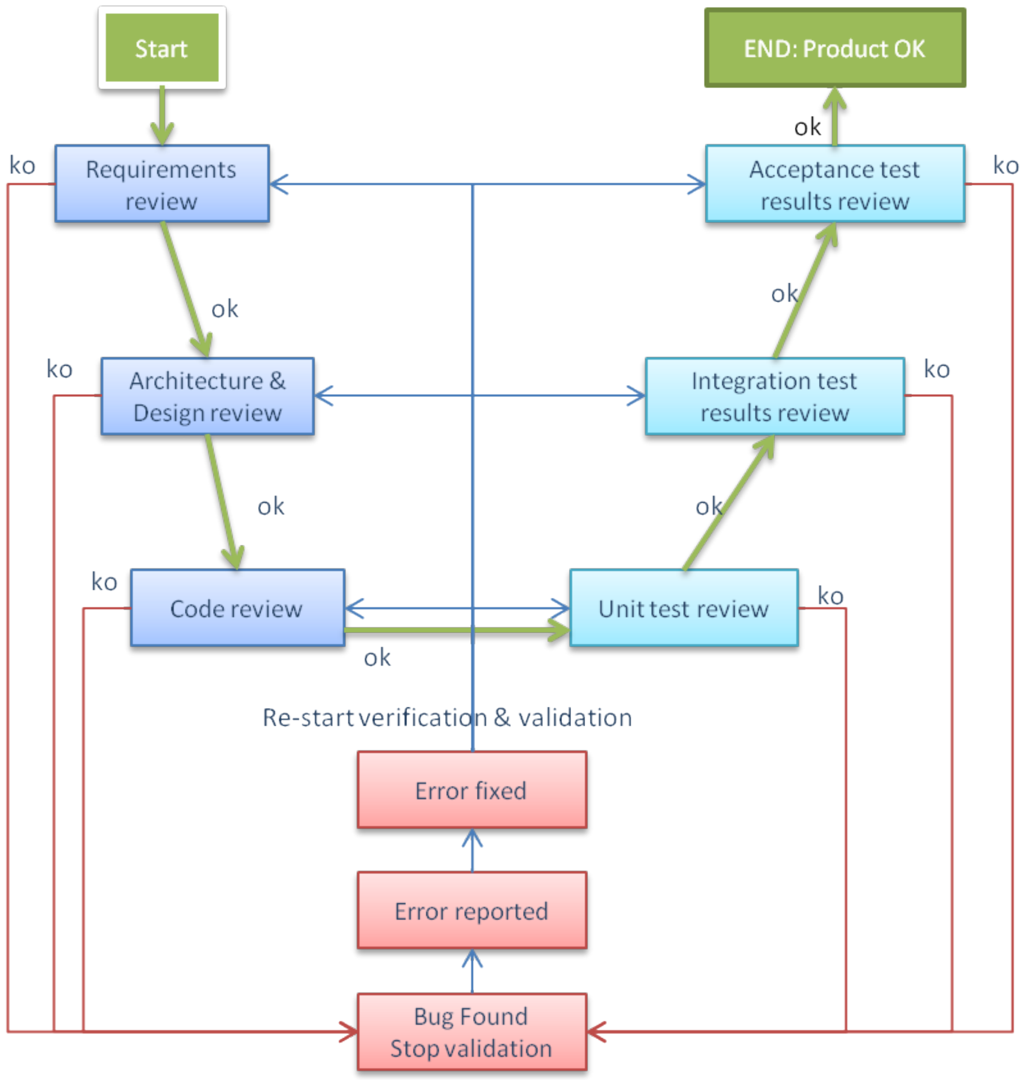
Start
END: Product OK
ok
ko
Requirements review
Acceptance test results review
ko
ok
ok
ko
Architecture & Design review
Integration test results review
ko
ok
ok
ko
Code review
Unit test review
ko
ok
Re-start verification & validation
Error fixed
Error reported
Bug Found Stop validation

Figure 2. Transition flow

4.1 States

4.1.1 Start

State that init the test process.

4.1.2 Requirements review

This is the first state after start state and shows the situation of process when the responsible person is reviewing all system requirements.

4.1.3 Architecture and design review

This is the next state after requirements review state, and shows the situation of process when the responsible person is reviewing the system architecture and the system design.

4.1.4 Code review

In this case, the responsible person is reviewing all the source code of libraries and their documentation.

4.1.5 Unit test review

This state shows the situation where all tests have been executed and the responsible person is reviewing all results.

4.1.6 Integration test results review

This state shows the situation where all integration tests and regression tests have been executed and the responsible person is reviewing all the results.

4.1.7 Acceptance test results review

This state shows the situation where all acceptance tests have been executed and the responsible person is reviewing all the results.

4.1.8 Bug found Stop validation

This state shows the situation where one or more bugs have been found, and should be reported and fixed. It should be noted that bug here is used in a wider sense, as requirement or architecture problem will also lead to this state.

4.1.9 Error reported

This state shows the situation where one or more bugs have been found and reported, and should be fixed.

4.1.10 Error fixed

This state shows the situation where one or more bugs have been found, reported and fixed, and the general process of verification and validation should be re-started.

4.1.11 End Product OK

The end state shows that the software is correct and can be released.

5 VERIFICATION AND VALIDATION ENVIRONMENT

5.1 Software

The following subsections describe the main components of the V&V environment.

5.1.1 Git and GitLab

GIT and GITLAB are the tools for source control version. They offer a large set of features that allow to efficiently and safely distribute source code. In particular they are widely used and distributed, besides being currently used to manage the sources of some of the most prestigious systems in the world.

5.1.2 Redmine

REDMINE is an open source tracking tool, with a large number of plug-in and characteristics making it easy the tracking of issues. In particular, it allows for direct links with tools of continuous integration.

5.1.3 CppUnit / GoogleTest

These are the libraries selected to provide support for test cases.

5.1.4 Sonar

SONAR with Cobertura and lcov is the selected toolset to manage coverage tests.

5.1.5 Jenkins

Jenkins is a continuous integration application to automate the build and the tests of software products. Jenkins has a large number of options, and has a large number of connectors to other applications, helping to automate procedures into a continuous integration job.

5.1.6 FlexeLint for C/C++

FlexeLint is a static code analysis tool which comes with MISRA C++ rules out-of-the-box. It is used for automating the review of the source code.

5.2 Hardware

The hardware needs to be as close as possible to real situations in order to test the libraries that will be executed. For the purpose of this development, MARTe will be compiled only for Windows and Linux (CentOS) environments. So, two execution environments will be needed: one for Windows and another for CentOS.

6 COMPILER ASSUMPTIONS

Compiler versions and compiler flags must be captured and applied systematically to build files.

	Linux	Windows
Compiler & Version	GCC v4.8.2	MSC (MS Visual Studio 2010)
Optimization flags		-W1 -O2 -Ob2
Other flags	-fPIC -frtti -Wall -std=c++98 -Werror -Wno-invalid-offsetof -DUSE_PTHREAD -pthread	-MD -GR

Table 4 Compiler assumptions

The flags are defined in the Makefiles of MARTe. The purpose of putting them here is to define them as a reference, in order to verify later that they are applied to makefiles.

7 ANNEXES

7.1 Audit review report

7.1.1 Report metadata properties

Property	Value	Comments
Tracker	Test	
Subject	QA Audit (vX.Y)	For each release, the version name follows this convention: X = major version and Y = minor version
Priority	Normal	
~~Private~~	~~No~~	

Table 5 Audit review report metadata

7.1.2 Report workflow properties

Property	Value	Comments
Assignee	<name>	Person in charge to coordinate all the quality reports.
Status	<status>	{New, In progress, Closed}
Start date	<date>	
Due date	<date>	
% Done	<ratio>	Ratio number of reports done of total reports.
Watchers	<list of names>	Should be the persons with role programmer.

Table 6 Audit review report workflow

7.1.3 Report template

```
h1. Requirements review


*Date of the review:* DD/MM/YYYY


*Person who did the review:* Full name


*Version of requirements:* X.Y
```

```
*Result of review:* [PASS|FAIL]


*List of non-conformities:* (*)

* Comment 1

* Comment 2

* ...

* Comment N


(*) Or write only "N/A" instead of a list if review passed without non-conformities.


h1. Architecture & design review


*Date of the review:* DD/MM/YYYY


*Person who did the review:* Full name


*Version of architecture & design document:* X.Y


*Result of review:* [PASS|FAIL]


*List of non-conformities:* (*)

* Comment 1

* Comment 2

* ...

* Comment N


(*) Or write only "N/A" instead of a list if review passed without non-conformities.


h1. Code and documentation review


*Date of the review:* DD/MM/YYYY


*Person who did the review:* Full name


*Version of source code:* X.Y


*Result of review:* [PASS|FAIL]


*List of non-conformities:* (*)

* Comment 1

* Comment 2
* ...
* Comment N

(*) Or write only "N/A" instead of a list if review passed without non-conformities.

h1. Unit test review

*Date of the review:* DD/MM/YYYY

*Person who did the review:* Full name

*Version of unit tests:* X.Y

*Result of coverage tests review:*

*Result of functional tests review:*

*Result of review:* [PASS|FAIL]

*List of non-conformities:* (*)
* Comment 1
* Comment 2
* ...
* Comment N

(*) Or write only "N/A" instead of a list if review passed without non-conformities.

h1. Integration test review

*Date of the review:* DD/MM/YYYY

*Person who did the review:* Full name

*Version of integration tests:* X.Y

*Result of review:* [PASS|FAIL]

*List of non-conformities:* (*)
* Comment 1
* Comment 2
* ...
* Comment N


(*) Or write only "N/A" instead of a list if review passed without non-conformities.


h1. Acceptance test review


*Date of the review:* DD/MM/YYYY


*Person who did the review:* Full name


*Version of acceptance tests:* X.Y


*Result of review:* [PASS|FAIL]


*List of non-conformities:* (*)
* Comment 1
* Comment 2
* ...
* Comment N


(*) Or write only "N/A" instead of a list if review passed without non-conformities.
```

Note: This template follows the formatting rules of https://redmine.gtd.es/help/wiki_syntax.html.