Dependable Real-time Infrastructure for Safety-critical Computer

Project number: 869945

Project acronym: De-RISC

http://www.derisc-project.eu/

D3.2	**RISC-V XtratuM verification report**		
Work Package	WP3	**Lead Beneficiary**	FEN
Type	Report	**Dissemination level**	Public
Due Date	31/03/2021	**Version**	1.1

Brief description

This document describes the verification activities carried out for the XtratuM hypervisor, which are in line with those described in the ECCS and the results of the verification process.

European Commission

Document control page

Written by

Name	**Beneficiary**	**Date**
Vicente Nicolau	FEN	25/03/2021

Reviewed by

Name	**Beneficiary**	**Date**
Jimmy le Rhun	TRT	29/03/2021

Approved by

Name	**Beneficiary**	**Date**
Paco Gómez	FEN	30/03/2021

Change log

Version	Date	Name	Beneficiary	Comments
0.1	15/02/2021	Vicente Nicolau	FEN	Initial document. SVerP defined.
0.2	02/03/2021	Vicente Nicolau	FEN	Preparations of the SVR part.
0.3	11/03/2021	Antonio García-Vilanova	FEN	Report on the verification activities.
1.0	26/03/2021	Vicente Nicolau	FEN	Final version of the document.
1.1	29/03/2021	Jimmy le Rhun	TRT	Document reviewed. Minor changes applied.

Disclaimer

The De-RISC Consortium comprises the following partners:

#	Partner legal name	Short name	Acronym	Country
1	FENT INNOVATIVE SOFTWARE SOLUTIONS SL	fentISS	FEN	Spain
2	BARCELONA SUPERCOMPUTING CENTER - CENTRO NACIONAL DE SUPERCOMPUTACIÓN	BSC	BSC	Spain
3	THALES SA	THALES	TRT	France
4	COBHAM GAISLER AB	COBHAM GAISLER	CG	Sweden

Copyright - This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement EIC-FTI No 869945. Unless officially marked PUBLIC, this document and its contents remain the property of the beneficiaries of the De-RISC Consortium and may not be distributed or reproduced without the express written approval of the project Consortium.

3 / 49

Table of contents

Index of tables

Index of illustrations

1. Introduction

Software verification process allows to confirm that adequate specifications and inputs exist for any activity, and that the outputs of the activities are correct and consistent with the specifications and input. The implementation of this process requires a Software Verification Plan (SVerP) where is identified the how and when the verification activities are performed. SVerP describes the organization aspects and defines the approaches and methodologies to execute the software verification activities under the European Cooperation for Space Standardization (ECSS) standards. The evidences of the execution of the verification process are gathered in the Software Verification Report (SVR), whose purpose is to present the results of all the software verification activities that have to be executed along the software development life cycle according to the SVerP.

Both ECSS documents (SVerP and SVR) are a constituent of the Design Justification File (DJF) and are called from the standards clause §5.8.2 of ECSS-E-ST-40C [EE402009] and clause §6.2.6 of ECSS-Q-ST-80C [EE802017].

This document is structured as follows:

- Chapter 4 describes the verification activities to be performed (SVerP);
- Chapter 5 report the results of execution of the verification activities (SVR).

2. Applicable and reference documents

2.1. Applicable documents

Reference	Document information
[DoA2019]	De-RISC Consortium: GRANT AGREEMENT NUMBER 869945 — De-RISC, 2019.
[EE402009]	ECSS: Software general requirements, ECSS-E-ST-40C, Available at http://www.ecss.nl, 2009.
[EE802017]	ECSS: Software product assurance, ECSS-Q-ST-80C, Available at http://www.ecss.nl, 2017.
[EE802017]	ECSS: Software product assurance, ECSS-Q-ST-80C, Available at http://www.ecss.nl, 2017.

Table 1: Applicable documents.

2.2. Reference documents

Reference	Document information
[D1.1]	De-RISC Consortium: Interim Platform and domain requirement specification and definition. Deliverable 1.1 of the De-RISC project. Grant Agreement EIC-FTI No 869945, 2019.
[D1.2]	De-RISC Consortium: Final platform and domain requirement specification and definition. Deliverable 1.2 of the De-RISC project. Grant Agreement EIC-FTI No 869945, 2020.
[D4.1]	De-RISC Consortium: Validation strategy definition. Deliverable 4.1 of the De-RISC project. Grant Agreement EIC-FTI No 869945, 2021.

Table 2: Reference documents.

3. Terms, definitions and acronyms

3.1. Terms and definitions

Term	Definition
Description of Action	Also known as “Technical Annex” or “Annex I to the Grant Agreement.” It is an annex to the contract (aka “Grant Agreement”) signed between the Project Consortium and the European Commission describing the technical content of the work to be carried out by the beneficiaries of the funding under the European Union's Horizon 2020 Programme.
Horizon 2020	It is the biggest EU Research and Innovation Programme ever with nearly €80 billion of funding available over 7 years (2014 to 2020) – in addition to the private investment that this money will attract. The purpose of Horizon 2020 is to foster the growth of breakthrough technologies, inventions and advanced developments by the promotion of scientific ideas from the laboratories to the market.
RISC-V	Pronounced "risk-five". It is an open-source hardware instruction set architecture based on established reduced instruction set computer principles.

Table 3: Terms and definitions

3.2. Acronyms

Acronym	Definition
AR	Acceptance Review
AppSW	Application Software

BSC	Barcelona Supercomputing Center
CDR	Critical Design Review
CI	Configuration Item
CG	Cobham Gaisler
CPU	Central Processing Unit
DJF	Design Justification File
DoA	Description of Action
EC	European Commission
ECSS	European Cooperation for Space Standardization
EU	European Union
ESA	European Space Agency
FDIR	Failure Detection, Isolation and Recovery
FEN	fentISS
FPGA	Field Programmable Gate Array
GA	Grant Agreement
GPP	General Purpose Processor
H2020	Horizon 2020
HM	Health Monitor
HW	Hardware
IRQ	Interrupt Request
ISA	Instruction Set Architecture
LTCF	LithOS Configuration File
MAF	Major Frame
MCU	Micro-Controller Unit
MPSoC	Multi-Processor (Multi-Core) SoC
NoC	Networks-on-Chip
ORR	Operational Readiness Review
PA	Product Assurance
PCB	Printed Circuit Board
PDR	Preliminary Design Review
PLIC	Platform-Level Interrupt Controller Specification
PRR	Preliminary Requirement Review
PUS	Packet Utilization Standard
PVEE	Partition virtual execution environment
QR	Qualification Review
RB	Requirements Baseline

RID	Review Item Discrepancy
RISC	Reduced Instruction Set Computer
RISC-V	Reduced Instruction Set Computer Five
RTOS	Real Time Operating System
SDD	Software Detailed Design
SDIL	Software Design and Implementation Leader
SoC	System-on-Chip
SDIL	Software Design and Implementation Leader
SITP	Software Integration Test Plan
SITR	Software Integration Test Report
SUTP	Software Unit Test Plan
SUTR	Software Unit Test Report
SValP	Software Validation Plan
SValR	Software Validation Report
SVS	Software Validation Specification
SVE	Software Verification Engineer
SVL	Software Verification Leader
SVerP	Software Verification Plan
SVR	Software Verification Report
SVT	Software Verification Team
SUM	Software User Manual
SSS	Software System Specification
SDR	System Design Review
SRR	System Requirements Review
SW	Software
TC	Telecommand
TM	Telemetry
TRL	Technology Readiness Levels
TRR	Test Readiness Review
TRT	Thales Research & Technology
WP	Work Package
WCET	Worst-Case Execution Time
XCF	XNG Configuration File
XML	Extensible Markup Language
XNG	XtratuM NextGeneration

Table 4: Acronyms.

4. Software Verification Plan

4.1. Software verification process overview

4.1.1. General

This section describes the approach used to implement the verification process through the software life-cycle, the verification effort, and the level of required independence for the verification tasks.

The software verification process is executed in parallel to the software development process life-cycle. Different verification activities must be completed at different phases of the project determined generally by a set of review milestones defined in the ECSS, typically System Requirements Review (SRR), Preliminary Design Review (PDR), Critical Design Review (CDR), Qualification Review (QR), Acceptance Review (AR) and Operational Readiness Review (ORR). Other intermediate reviews may also be included, such as System Design Review (SDR), Preliminary Requirement Review (PRR) or Test Readiness Review (TRR).

In the scope of the De-RISC project, no ECSS milestones have been defined as the aim is to ease the future ECSS qualification of the platform. Instead, the reviews will be synchronized with the project review, project milestones and deliverable submissions.

4.1.2. Organization

The software verification is an activity included in the WP3, whose lead beneficiary is FEN. There is not a defined task for this activity, but the activity is performed during the whole development life cycle, from the requirements to the acceptance tests, if any.

The verification activities will be performed by the software Validation and Verification (V&V) Team of FEN. The roles involved in the verification process are as follows:

- ***Software Verification Leader*** **(SVL)**: Person in charge of leading the verification activities;
- ***Software Verification Engineer*** **(SVE)**: People in charge of performing the verification of the project.

NOTE: The *Software Verification Team* (SVT) is composed by the SVL and the SVE, and they must not be involved in other technical activities of the software development process.

The SVL is in direct contact with the *Software Design and Implementation Leader* (SDIL) of FEN.

During verification activities when an issue is detected or in uncertainty situations in the interpretation or use of the CIs under verification, a preliminary reporting process is addressed from the SVL to the SDIL. This preliminary reporting is addressed through the Review Item Discrepancy (RID) management procedure. The RID procedure is described in the appendix A. All the RIDs must be closed before the end of the project (that is, solved or justified). Detailed information about problem reporting and resolution can be found in the section 4.2.

The results of the verification activities are reported in the Software Verification Report (SVR) located in section 5.

4.1.3. Schedule

The schedule of the software verification activities matches the duration of WP3 and WP4 (refer to [DoA2019, Annex 1, Part B, §3.1]). Therefore, verification activities start at M4 (when [D1.1] is available) and ends at M30. Although this document was aimed to be completed in M18, is not possible to have the verification of the software validation activities as they have not started yet (in fact, [D4.1] describes the strategy of the validation without including the implementation, whose due date is also M18). This way, the current document will report the verification activities carried out until M18, and the rest of the software verification activities will be reported in an update of this document in M30.

4.1.4. Resource summary

The resources listed in the table 5 are required for performing the verification activities.

Type	Resource
Staff	Role – Software Verification Leader (SVL)
Staff	Role – Software Verification Engineer (SVE)
Hardware	Xilinx Kintex UltraScale FPGA KCU105
Hardware	x86_64 PC compatible
Software	Linux Ubuntu
Software	Vivado Design Suite
Software	GRMON3
Software	GCC toolchain (RISC-V cross compiler)
Software	GNU binutils (RISC-V cross compiler)
Software	GCC toolchain
Software	GNU binutils
Software	GNU make
Software	PDF reader
Software	Python
Software	VCS tool (Subversion)
Software	Text editor (LibreOffice Writer)
Software	Spreadsheet application (LibreOffice Calc)
Software	Serial terminal (Minicom/Picocom/Cutecom)

Table 5: Resource summary.

4.1.5. Responsibilities

There are several skills and responsibilities for the SVL and for the SVE roles. In particular, for software verification, the responsibilities for the SVL are:

- Establish the software verification strategy and plan;
- Monitor and implement the verification plan and activities;
- Interface with PC and SDIL.

On the other hand, the responsibilities for the SVE are:

- Carry out the verification activities;
- Report the verification findings in the section 5.

4.1.6. Identification of risk and level of independence

The risk management is performed at project level where the PC is the main responsible for risk identification, monitoring and mitigation, with the collaboration of the work packages lead beneficiaries.

The verification activities can contribute to mitigate the risks in other technical phases. However, this risk evaluation is performed on each progress meeting.

As mentioned in section 4.1.2, the personnel involved on verification activities does not participate directly in the technical activities of the software development and it can be considered enough isolation to perform the verification activities.

4.1.7. Tools, techniques and methods

The software tools used on the verification activities are listed in the table 5 of this document.

The verification techniques to be used in the verification process are: inspection, cross-referencing and traceability analysis.

The usage and the phases where the tools and techniques are used is described on each verification activity in the section 4.3.

4.1.8. Personnel requirements

The personnel performing the software verification activities must be trained in:

- UML;
- C99 programming language;
- Knowledge on the RISC-V architecture;
- Knowledge on the NOEL-V processor family;
- Knowledge on the Vivado Design Suite;
- Assembly language for RISC-V;
- Python language;
- ECSS standard (ECSS-E-ST-40C [EE402009] and ECSS-Q-ST-80C [EE802017]), focusing on the requirements related to the verification process;
- Use of VCS.

4.2. Control procedures for verification process

This chapter presents the mechanisms to be used when problems or discrepancies are found during the software verification process of the CIs.

4.2.1. Problem reporting and resolution

When a problem is detected during a verification activity, a RID (Review Item Discrepancy) is generated. The goal of the RID procedure is to record identified problems, questions and solutions arising from the examination of the documentation and the detection of problems in the software. A self-explanatory template list of RIDs in Appendix A will keep track of the status of such discrepancies and documents the process to solve the raised issues, consolidate the findings and provide recommendations for their closure.

4.2.2. Deviation and waiver policy

Any deviation found during the verification activities will be reported as RID and reviewed in future project iterations. So far, the redaction of a Software Development Plan (SDP) and a Software Configuration Management Plan (SCMP) is out of the scope of this project, but they will be added in further revisions leading to a full ECSS qualification. These documents will define the deviation and waiver policy.

4.2.3. Control procedures

The control procedures allow to check and track the verification activities by means of a systematic process that control the actions performed in each task. The following control procedures have been identified:

- Management of CIs to be verified for each formal review.

4.2.3.1. Verification process

Description

The verification process starts when the CIs are available and delivered by the SDIL, and this process ends with the generation of the SVR integrated in the section 5. The review activities may generate RIDs for the CI being verified. These RIDs are managed according to the RID processing procedure self explained in the Appendix A.

When

- For starting each verification activity: check inputs;
- After finishing each verification activity: generation RIDs and evidences.

Inputs

- The CIs required as input in each verification activity according to the verification plan.

Outputs

- Generation of RIDs;
- Gathering of evidences;
- Generate the SVR.

Steps

1. The SDIL delivers the CI required as input in each verification activity;
2. If the CIs have not been modified from the initial delivery before starting the first verification activity, then the SVL accepts the inputs. Otherwise, if the delivery of some CIs have been updated during verification process, then the SVL can decide if accept or reject the CI modified;
3. Execute the actions proposed in the verification activities. Once the verification activity starts, the CIs are versioned and these cannot be modified;
4. The SVE generates RIDs and evidences. These RIDs are processed according to the RID processing procedure self described in the Appendix A;
5. The verification team generates the SVR in the section 5.

4.3. Verification activities

4.3.1. General

This section describes the verification activities to be performed in the frame of a ECSS-driven project, identifying the execution of the activities and tasks in the software development life-cycle. The verification activities can be grouped based on software development phases where are developed as follows:

1. Software Requirements Specification:
   a. Technical specification verification activity.
2. Software design development:
   a. Architectural design verification activity;
   b. Detailed design verification activity;

   c. Software User Manual verification activity.

3. Software code implementation:
   a. Source code verification activity;
   b. Unit testing verification activity;
   c. Integration testing verification activity;
   d. Technical budgets verification activity.
4. Software validation:
   a. Activity of verification of software validation with respect to TS.

There are some verification activities considered in the ECSS standard that are not included as a verification activities in this plan, these are:

- ✗ **Verification of requirements baseline**: The verification activity related to the Software System Specification (SSS) is not included in this plan due to the Requirements Baseline (RB) are not foreseen to be present in the full ECSS qualification of the software.
- ✗ **Verification of software documentation**: This verification activity is typically delegated to the Product Assurance (PA) team as part of the Quality Assurance activities.
- ✗ **Schedulability analysis for real-time software**: This activity is not applicable within this project due to the final application (i.e. user application) is outside of the context of this qualification. The schedulability analysis must be performed using the whole system, considering the final application and which is not within the scope of this project.
- ✗ **Technical budgets management**: This activity is not completely performed because the margins and the specific requirement of budgets must be imposed by the final application, which is not considered in the scope of this qualification. However, the product being qualified provides performance measures of the services and operations supplied, which will be helpful in the elaboration of schedulability analysis and technical budgets analysis.
- ✗ **Verification of software validation with respect to RB**: This verification activity is not considered due to validation with respect to RB is not contemplated in the software development process of the product as there is no RB. This activity must be performed in the integration with a final application.

Additionally, automatic code generation is not considered in this project. Therefore, specific verification activities for automatically generated code are not required.

The verification process (XNG-VE) in the project is broken down into different verification activities (XNG-VE-Ann) and each verification activity is composed of tasks. All the verification elements (process, activities and tasks) are unequivocally identified. The table 6 shows in detail how the verification elements are identified.

Element	Identifier	Description
Verification process	*XNG-VE*	Verification process identifier within the project
Verification activity	*XNG-VE-Ann*	Where *XNG-VE* identifies the verification process, *A* identifies the element as a verification activity and *nn* identifies the number of the activity. *nn* is a number left padded with zeros until 2 digits are reached.
Verification task	*XNG-VE-Ann-Tx*	Where *XNG-VE-Ann* identifies the number of activity and *Tx* identifies the number of task within the activity in the format *T1*, *T2*, *T3*...

Table 6: Nomenclature for the verification elements.

The table 7 shows the verification activities grouped by software development phases highlighting the deadline where the outputs of the verification activities must be delivered. Note that some activities are defined but not planned in the scope of the De-RISC project. They are defined for the sake of completeness of the verification plan to ease the future full qualification. Those activities will be reported as out of the scope of the project in the SVR (section 5). Other activities will be reported in the next update of this document in M30 due to the activities have not started yet at the time of delivering the current document.

Although the table 7 shows the general plan to execute the verification activities, these activities could be activated from some other proceeding that requires to execute some activity during different point in the software life-cycle, for example, due to the resolution of a SPR.

Identifier	Verification activity	M06	M18	M30
	Software Requirement Specification			
XNG-VE-A01	Technical specification verification	X		
	Software design development			
XNG-VE-A02	Architectural design verification	-	-	-
XNG-VE-A03	Detailed design verification	-	-	-
XNG-VE-A04	Software User Manual verification		X	X
	Software code implementation			
XNG-VE-A05	Source code verification		X	X
XNG-VE-A06	Software unit testing verification			X
XNG-VE-A07	Software integration testing verification			X
XNG-VE-A08	Verification of technical measures values			X
	Software validation			
XNG-VE-A09	Verification of validation testing wrt TS			X

Table 7: Verification activities timeline.

The verification activities are described in the section 4.3.2 using the format presented in the table 8.

Activity identifier	<XNG-VE-Ann>
Title	<Verification activity name>
Development phase	<Software development phase as described in the table 7>
Start	<Restriction to start the activity. Based on main reviews or intermediate generation of deliverable>
End	<Restriction to finish the activity. Based on main reviews>
Objectives	<Objectives of the activity>
Techniques and tools	<Verification techniques, tools and facilities utilized to accomplish the verification activity>
Input	<Inputs required to achieve the verification activity>
Output	<Intermediate and final outputs resulting of the execution of the verification activity>
Tasks	<The activity is breakdown into tasks. Each task describes how they are performed (actions) and the methodology (techniques) to be used. New tables will be used to describe each task (see table 9)>

Table 8: Verification activity description format.

Task identifier	<XNG-VE-Ann-Tx>
Title	<Verification task name>
Techniques	<Methodology to be used to perform the task>
Actions	<Steps to perform the task>

Table 9: Verification task description format.

4.3.2. Software process verification

This section lists the verification activities to be executed in the verification process and how they are performed, identifying the required inputs, the outputs generated and methodologies and tools used to accomplish the activity. The details of the activities and its tasks are presented following the format presented in the tables 8 and 9. The activities list in this section are based on those presented in the table 7.

4.3.2.1. Technical specification verification

Activity identifier	XNG-VE-A01
Title	Technical specification verification
Development phase	Software Requirement Specification
Start	When the TS ([D1.1] and [D1.2]) documents are delivered
End	In PDR according to the ECSS (M06 for De-RISC scheduling)
Objectives	Verify that the representation performed of the software system specification in the software requirements specification is complete, consistent, correct, feasible, testable and maintainable.
Techniques and tools	- PDF reader
	- LibreOffice Calc
Input	- D1.1 (SRS)
	- D1.2 (SRS)
Output	Section 5.1.2
Tasks	XNG-VE-A01-T1.
	XNG-VE-A01-T2.
	XNG-VE-A01-T3.
	XNG-VE-A01-T4.
	XNG-VE-A01-T5.
	XNG-VE-A01-T6.

Table 10: XNG-VE-A01: Technical specification verification.

Task identifier	XNG-VE-A01-T1
Title	Verify the software requirement consistency and traceability.
Techniques	Inspection and traceability analysis.
Actions	1. Check that the software system requirements are traceable to software requirement specification.
	2. Check that the software requirement specification are traceable to software system requirements.
	3. Check that the software requirements that are not traced to the system requirements are justified.

Table 11: XNG-VE-A01-T1.

Task identifier	XNG-VE-A01-T2
Title	Verify the interface requirements.
Techniques	Inspection.
Actions	1. Check that the interface requirements specification represents consistently the characteristics of the system interfaces expected for the software.
	2. Check that the interface requirements specification represents consistently the interfaces with the hardware.
	3. Ensure that the interfaces specification of the software and the software with the hardware has a consistent level of detail, for example: data and control flows, data usage and format, and performance.

Table 12: XNG-VE-A01-T2.

Task identifier	XNG-VE-A01-T3
Title	Verify the testability of the software requirements specification.
Techniques	Inspection.
Actions	1. Ensure that the software requirements are testable in some way.
	2. Check that the validation criteria and method proposed for the requirements specification are objectives.

Table 13: XNG-VE-A01-T3.

Task identifier	XNG-VE-A01-T4
Title	Verify the feasibility and maintainability of the software requirements specification.
Techniques	Inspection and cross-reference.
Actions	1. Check that the software requirements specification are feasible to be used in the software design.
	2. Ensure that the software requirements are feasible to be maintained in a correct way. Changes in the software requirements are correctly traced and updated with a reduced impact.
	3. Ensure that the engineering model used to manage the software requirements contains information consistent with the documentation generated from a point of view of maintainability.

Table 14: XNG-VE-A01-T4.

Task identifier	XNG-VE-A01-T5
Title	Verify the correctness of the software requirements specification.
Techniques	Inspection.
Actions	1. Ensure that the software requirements are specified in a uniform manner and style.
	2. Ensure that the software requirements has a consistent structure and they are free from ambiguous terms.
	3. Check that the software requirements related to safety and criticality are considered in a correct way.
	4. Check that hardware environment and implementation constrains are correctly identified.

Table 15: XNG-VE-A01-T5.

Task identifier	XNG-VE-A01-T6
Title	Verify the logical model.
Techniques	Inspection and cross-reference.
Actions	1. Check that the behaviour described in the logical model is consistent with the purpose of the software described in the software requirements and system requirements. 2. Ensure that the artifacts described by the logical model are specified in a uniform manner and style. 3. Ensure the correspondence between the software requirements and the artifacts described by the logical model. 4. Check that the software services expected are completely defined and the behaviour is specified as a function or procedure. 5. Check that the content of the artifacts are correctly enumerated and the data and control flows have a consistent level of detail. 6. Ensure that the engineering model where the engineering diagrams are maintained contains information consistent with the documentation generated and the changes in the diagrams can be traced. 7. Ensure that the engineering diagrams are referenced and correctly documented. 8. Check that the software design constrains are correctly identified.

Table 16: XNG-VE-A01-T6.

4.3.2.2. Software architectural design verification

Activity identifier	XNG-VE-A02
Title	Architectural design verification
Development phase	Software design development
Start	When the SDD document is delivered
End	In PDR according to the ECSS
Objectives	Verify that the architectural design is complete, consistent, correct, feasible and maintainable
Techniques and tools	- PDF reader
	- LibreOffice Calc
Input	- D1.2 (SRS)
	- SDD
Output	Section 5.1.3
Tasks	XNG-VE-A02-T1.
	XNG-VE-A02-T2.
	XNG-VE-A02-T3.
	XNG-VE-A02-T4.
	XNG-VE-A02-T5.

Table 17: XNG-VE-A02.

Task identifier	XNG-VE-A02-T1
Title	Verify the software components consistency and traceability.
Techniques	Inspection and traceability analysis.
Actions	1. Check that the software components are traceable to software requirement specification.
	2. Check that the software requirement specification are traceable to the software components.
	3. Check that the software components that are not traced to the software requirement are justified.

Table 18: XNG-VE-A02-T1.

Task identifier	XNG-VE-A02-T2
Title	Verify the interface and architecture of the design.
Techniques	Inspection and cross-reference.
Actions	1. Check that the interfaces and architecture of the design represent consistently the characteristics presented in the software requirements specification. 2. Check that the interfaces of the software components represent consistently the interfaces with the hardware. 3. Ensure that the software components has an internal consistency and high level of detail.

Table 19: XNG-VE-A02-T2.

Task identifier	XNG-VE-A02-T3
Title	Verify the feasibility and maintainability of the architectural design.
Techniques	Inspection.
Actions	1. Check that using the architectural design is feasible to produce the detailed design. 2. Ensure that the architectural design and interfaces are feasible to be maintained in a correct way. Changes in the software requirements are correctly traced and updated with a reduced impact in the software components and models. 3. Ensure that the engineering model used to manage the architectural design and interfaces contains information consistent with the documentation generated from a point of view of maintainability.

Table 20: XNG-VE-A02-T3.

Task identifier	XNG-VE-A02-T4
Title	Verify the correctness of the design.
Techniques	Inspection.
Actions	1. Ensure that the architecture and components of the design are presented in a uniform manner and style. 2. Check that the architectural design and interfaces are correct respect to safety and critical requirements. 3. Check that the design specify a correct sequence of events, including inputs, outputs, interfaces defining a logical flow. 4. Ensure that the software dynamic is correctly represented with enough level of detail.

Table 21: XNG-VE-A02-T4.

Task identifier	XNG-VE-A02-T5
Title	Verify the completeness of the architectural design.
Techniques	Inspection and cross-reference.

Actions	1. Check that the software behaviour is completely represented in the behaviour view of the architectural design. 2. Check the design model describes and covers completely the software dynamic justifying the real-time choices. 3. Check that in the architectural design the hierarchical breakdown from high level components to terminal ones is provided. 4. Ensure that exists a temporal characterization between external and internal interfaces, i.e., check that characteristics related to determinism, response time, etc. are considered.

Table 22: XNG-VE-A02-T5.

4.3.2.3. Software detailed design verification

Activity identifier	XNG-VE-A03
Name	Detailed design verification
Development phase	Software design development
Start	When the SDD document with the detailed design is delivered after PDR
End	In CDR according to the ECSS
Objectives	Verify that the detailed design is complete, consistent, correct, testable and maintainable.
Techniques and tools	- PDF reader - LibreOffice Calc
Input	- D1.2 (SRS) - SDD
Output	Section 5.1.4
Tasks	XNG-VE-A03-T1. XNG-VE-A03-T2. XNG-VE-A03-T3. XNG-VE-A03-T4. XNG-VE-A03-T5. XNG-VE-A03-T6.

Table 23: XNG-VE-A03.

Task identifier	XNG-VE-A03-T1
Title	Verify the software units traceability.
Techniques	Inspection and traceability analysis.
Actions	1. Check that the software units are traceable to software components.
	2. Check that the software components are traceable to the software units.
	3. Check that the software units are traceable to software requirements specification.
	4. Check that the software requirements specification are traceable to software units.
	5. Check that the software units and software requirements that are not traced to the software requirements and software units respectively are justified.

Table 24: XNG-VE-A03-T1.

Task identifier	XNG-VE-A03-T2
Title	Verify the interfaces of the detailed design.
Techniques	Inspection and cross-reference.
Actions	1. Check that the interfaces of the software units represent consistently the characteristics presented in the architecture.
	2. Check that the interfaces of the software units considers the interfaces with the hardware.
	3. Ensure that the software units has an internal consistency and high level of detail.

Table 25: XNG-VE-A03-T2.

Task identifier	XNG-VE-A03-T3
Title	Verify the testability of the software units.
Techniques	Inspection.
Actions	1. Check that the software units are testable ensuring that can be commandable and observable to fulfill the testing requirements.
	2. Check that the detailed design includes temporal and computational invariant properties.
	3. Check that the software units can be tested through faults injection.

Table 26: XNG-VE-A03-T3.

Task identifier	XNG-VE-A03-T4
Title	Verify the feasibility and maintainability of the detailed design.
Techniques	Inspection and cross-reference.
Actions	1. Check that an internal consistency exits between software components and software units. 2. Ensure that the detailed design is feasible to be maintained in a correct way. Changes in the software requirements specification are correctly traced and updated with a reduced impact in the software units. 3. Ensure that the engineering model used to manage the software units and interfaces contains information consistent with the documentation generated.

Table 27: XNG-VE-A03-T4.

Task identifier	XNG-VE-A03-T5
Title	Verify the correctness of the detailed design.
Techniques	Inspection.
Actions	1. Ensure that the software units and detailed design are presented in a uniform manner and style. 2. Check that the software units and interfaces are correct with respect to safety and critical requirements. 3. Check that the detailed design specify a correct sequence of events, including inputs, outputs, interfaces defining a logical flow. 4. Ensure that the software dynamic is correctly represented with enough level of detail.

Table 28: XNG-VE-A03-T5.

Task identifier	XNG-VE-A03-T6
Title	Verify the completeness of the detailed design.
Techniques	Inspection and cross-reference.
Actions	1. Check that the behaviour design of the software units is completely represented in the detailed design. 2. Check the design model describes and covers completely the software dynamic justifying the real-time choices. 3. Check that the hierarchical breakdown from high level components to terminal ones is provided. 4. Ensure that a temporal characterization exists between external and internal interfaces, i.e., check that characteristics related to determinism, response time, etc. are considered.

Table 29: XNG-VE-A03-T6.

4.3.2.4. Software user manual verification

Activity identifier	XNG-VE-A04
Title	Software user manual verification
Development phase	Software design development
Start	When the Software User Manual (SUM) document is delivered after PDR.
End	In CDR according to the ECSS (M30 for De-RISC scheduling)
Objectives	Verify that the User Manual is adequate, complete, consistent and maintainable.
Techniques and tools	- PDF reader - LibreOffice Calc
Input	- SUM - D1.2 (SRS) - SDD
Output	Section 5.1.5
Tasks	XNG-VE-A04-T1. XNG-VE-A04-T2. XNG-VE-A04-T3. XNG-VE-A04-T4. XNG-VE-A04-T5.

Table 30: XNG-VE-A04.

Task identifier	XNG-VE-A04-T1
Title	Verify the readability of the User Manual.
Techniques	Inspection.
Actions	1. Check that the user manuals has a clear, unified and consistent structure. 2. Check that the target software user is identified and that the content is according to that audience. 3. Check that the user manuals have all the required elements for its understanding (i.e. conventions used, acronyms, terms, etc.).

Table 31: XNG-VE-A04-T1.

Task identifier	XNG-VE-A04-T2
Title	Verify the completeness of the User Manual.
Techniques	Inspection.
Actions	1. Check that the User Manual describes all the functionalities implemented in the software. 2. Ensure that the operations performed by the software are described with enough detail level. 3. Check that the user interfaces of the software are included in the user manuals. 4. Ensure that performance measures are included in the User Manual.

Table 32: XNG-VE-A04-T2.

Task identifier	XNG-VE-A04-T3
Title	Verify the correctness of the User Manuals.
Techniques	Inspection and cross-reference.
Actions	1. Check that the information provided by the User Manual is according to the documents SRS and SDD. 2. Check that the restrictions and limitations of the software are correctly documented. 3. Check that the hardware-independent user manual does not contain references to specific features related to the hardware. 4. Check that the user manuals, both the hardware dependent and independent, do not contain features and restrictions duplicated without justification that can lead to contradictions.

Table 33: XNG-VE-A04-T3.

Task identifier	XNG-VE-A04-T4
Title	Verify the dependability and safety aspects are considered in the User Manual.
Techniques	Inspection.
Actions	1. Check that the User Manuals describes how to deal with fault detection, isolation and recovery according to the restrictions described in the SRS and SDD. 2. Ensure that the User Manual describes how the software behaves against hardware faults.

Table 34: XNG-VE-A04-T4.

Task identifier	XNG-VE-A04-T5
Title	Verify the feasibility and maintainability of the User Manual.
Techniques	Inspection and cross-reference.
Actions	1. Check that using the User Manual is feasible to use the software.
	2. Ensure that the User Manual is feasible to be maintained in a correct way. Changes in the SRS and SDD are correctly traced and updated with a reduced impact in the User Manual.
	3. Ensure that the engineering model used to generate models, diagrams or artifacts used in the User Manual contain information consistent with the documentation generated.

Table 35: XNG-VE-A04-T5.

4.3.2.5. Source code verification

Activity identifier	XNG-VE-A05
Title	Source code verification
Development phase	Software code implementation
Start	When the source code, software validation report, software unit and integration reports are delivered after PDR.
End	In CDR according to the ECSS (M30 for De-RISC scheduling)
Objectives	Verify that the source code is consistent, correct, feasible, testable and maintainable.
Techniques and tools	- PDF reader
	- LibreOffice Calc
	- LDRA
	- Python
Input	- SUM
	- D1.2 (SRS)
	- SDD
	- Software Unit Tests Report (SUTR)
	- Software Integration Tests Report (SITR)
	- Software Validation Report (SValR)
	- Software Source Code
Output	Section 5.1.6
Tasks	XNG-VE-A05-T1.
	XNG-VE-A05-T2.
	XNG-VE-A05-T3.
	XNG-VE-A05-T4.

Table 36: XNG-VE-A05.

Task identifier	XNG-VE-A05-T1
Title	Verify the software code consistency and traceability.
Techniques	Inspection and traceability analysis.
Actions	1. Check that the source code is traceable to the software units and components presented in the design and to the software requirements in the SRS.
	2. Check that the source code not traced to the units is justified.
	3. Check that the source code implemented is consistent with the representation of the software units and components.
	4. Check that the external and internal interfaces provided by the software code represents consistently the interfaces described in the requirements and design.

Table 37: XNG-VE-A05-T1.

Task identifier	XNG-VE-A05-T2
Title	Verify the correctness and robustness of the software code.
Techniques	Inspection and analysis of the results obtained from an external tool to check coding standards compliant.
Actions	1. Ensure that the software code is structured and organized in a uniform manner and style.
	2. Check that deactivated code cannot be activated or that its accidental activation cannot harm the operation of the system.
	3. Check that the source code follows the coding style proposed in the project.
	4. Check that the analysis of coding standards of the source code is correct. This check must be performed with the assistance of an external tool comparing the results obtained with the results presented in software development.
	5. Check that the software code implements correctly the methods proposed to cover the safety and critical requirements.
	6. Check that the code follows correctly events sequences considering interfaces, inputs, outputs and control flows presented in the design and requirements.
	7. Ensure that the code implements mechanisms against hardware faults and that the implementation describes the behave presented in the design for the error handling.
	8. Ensure that the source code includes protection mechanisms to numerical errors, division by zero, pointers, resources sharing, etc.

Table 38: XNG-VE-A05-T2.

Task identifier	XNG-VE-A05-T3
Title	Verify the feasibility and maintainability of the software code.
Techniques	Inspection and cross-reference.
Actions	1. Ensure that the software code is feasible to be maintained in a correct way. Changes in the software requirements and design can be updated within the source code with a reduced effort.
	2. Ensure that changes in the software code are correctly maintained and traced.

Table 39: XNG-VE-A05-T3.

Task identifier	XNG-VE-A05-T4
Title	Verify the testability and coverage of the source code.
Techniques	Inspection and analysis of the results obtained of coverage.
Actions	1. Ensure that the software code is testable using the validation criteria and method proposed in the software requirements.
	2. Check that analyzing the results of the execution of unit, integration and validation tests, the coverage achieved of the source code statement is 100%.
	3. Check that analyzing the results of the execution of unit, integration and validation tests, the coverage achieved of the source code decision is 100%.
	4. Check that if the coverage is not achieved through the test execution, the behave of the non covered code is analyzed by inspection or review of design.

Table 40: XNG-VE-A05-T4.

4.3.2.6. Software unit testing verification

Activity identifier	XNG-VE-A06
Title	Software unit testing verification
Development phase	Software code implementation
Start	When the unit test report is delivered.
End	In CDR according to the ECSS (M30 for De-RISC scheduling)
Objectives	Verify that the software unit testing is complete, consistent, correct, feasible and maintainable.
Techniques and tools	- PDF reader
	- LibreOffice Calc
Input	- D1.2 (SRS)
	- SDD
	- Software Unit Test Plan (SUTP)
	- Software Unit Test Report (SUTR)
	- Software Source Code
	- Software Unit Test Suite

Output	Section 5.1.7
Tasks	XNG-VE-A06-T1. XNG-VE-A06-T2. XNG-VE-A06-T3. XNG-VE-A06-T4.

Table 41: XNG-VE-A06.

Task identifier	XNG-VE-A06-T1
Title	Verify the software unit testing consistency and traceability.
Techniques	Inspection and traceability analysis.
Actions	1. Check that the unit tests are traceable to the detailed software design. 2. Check that the unit tests are traceable to the software requirements. 3. Check that the unit tests are traceable to the software code. 4. Check that exits an internal consistency of the unit tests specification and design with the detail design and software requirements.

Table 42: XNG-VE-A06-T1.

Task identifier	XNG-VE-A06-T2
Title	Verify the correctness and completeness of the software unit.
Techniques	Inspection.
Actions	1. Check that the unit activities proposed in the plan have been performed and completed in the unit testing. 2. Check that the unit testing has been performed according to the unit strategy proposed in the plan: types of tests, stubs, level of coverage, boundaries, etc. 3. Ensure that the source code of the unit tests is structured and organised in a uniform manner and style. 4. Check that the source code of the unit tests follows the coding style proposed in the project. 5. Check that the source code of the unit tests is according to the specification presented in the plan.

Table 43: XNG-VE-A06-T2.

Task identifier	XNG-VE-A06-T3
Title	Verify the unit test report.
Techniques	Inspection and cross-reference.
Actions	1. Check that the test report includes the results for all unit tests proposed in the specification.
	2. Check that the coverage is presented conforming to the unit plan, reaching the coverage levels proposed.
	3. Check that abnormal termination conditions and unfinished testing are reported and documented.
	4. Check that the results obtained are in line with what is expected.

Table 44: XNG-VE-A06-T3.

Task identifier	XNG-VE-A06-T4
Title	Verify the feasibility and maintainability of the unit testing.
Techniques	Inspection.
Actions	1. Ensure that the unit tests are feasible to be maintained in a correct way.
	Changes in the software units can be correctly traced and updated with a reduced impact in the test specification and unit tests.
	2. Test results, test logs, test cases and test documentation are maintained under configuration control and changes control.

Table 45: XNG-VE-A06-T4.

4.3.2.7. Software integration testing verification

Activity identifier	XNG-VE-A07
Title	Software integration testing verification
Development phase	Software code implementation
Start	When the integration test report is delivered.
End	In CDR according to the ECSS (M30 for De-RISC scheduling)
Objectives	Verify that the software integration testing is complete, consistent, correct, feasible and maintainable.
Techniques and tools	- PDF reader
	- LibreOffice Calc
Input	- SDD
	- Software Integration Test Plan (SITP)
	- Software Integration Test Report (SITR)
	- Software Source Code
	- Software Integration Test Suite
Output	Section 5.1.8
Tasks	XNG-VE-A07-T1.
	XNG-VE-A07-T2.
	XNG-VE-A07-T3.
	XNG-VE-A07-T4.

Table 46: XNG-VE-A07.

Task identifier	XNG-VE-A07-T1
Title	Verify the software integration testing consistency and traceability.
Techniques	Inspection and traceability analysis.
Actions	1. Check that the software integration test specification is traceable to the software design.
	2. Check that the strategy to address the integration testing is coherent with the software design and with the goals of coverage.
	3. Check that exits consistency between the strategy integration testing and the software components and units.

Table 47: XNG-VE-A07-T1.

Task identifier	XNG-VE-A07-T2
Title	Verify the correctness and completeness of the software integration.
Techniques	Inspection.

Actions	1. Check that the integration activities proposed have been performed and completed in the integration testing. 2. Check that the software integration has been performed according to the integration strategy proposed in the plan: types of tests, relation of components, level of coverage, etc. 3. Check that the integration test specification covers the interface testing goals. 4. Ensure that the source code of the integration tests is structured and organised in a uniform manner and style. 5. Check that the source code of the integration tests follow the coding style proposed in the project. 6. Check that the source code of the integration tests is according to the specification presented in the plan.

Table 48: XNG-VE-A07-T2.

Task identifier	XNG-VE-A07-T3
Title	Verify the integration test report.
Techniques	Inspection and cross-reference.
Actions	1. Check that the test report includes the results for all integration tests proposed in the specification. 2. Check that the coverage is presented conforming to the integration plan, reaching the coverage levels proposed. 3. Check that the results obtained are in line with what is expected and abnormal behaviours are documented.

Table 49: XNG-VE-A07-T3.

Task identifier	XNG-VE-A07-T4
Title	Verify the feasibility and maintainability of the integration testing.
Techniques	Inspection.
Actions	1. Ensure that the software integration are feasible to be maintained in a correct way. Changes in the software components or units can be correctly traced and updated with a reduced impact in the test specification and integration tests. 2. Test results, test logs, test cases and test documentation are maintained under configuration control and changes control.

Table 50: XNG-VE-A07-T4.

4.3.2.8. Verification of technical measures values

Activity identifier	XNG-VE-A08
Title	Verification of technical measures values
Development phase	Software code implementation
Start	When the source code, software validation report, software unit and integration reports are delivered.
End	In AR according to the ECSS (M30 for De-RISC scheduling)
Objectives	Verify the correctness and completeness of the most relevant budgets values of the software.
Techniques and tools	- PDF reader
	- LibreOffice Calc
Input	- D1.2 (SRS)
	- SDD
	- SUTR
	- SITR
	- SValP
	- Software Source Code
Output	Section 5.2
Tasks	XNG-VE-A08-T1.
	XNG-VE-A08-T2.

Table 51: XNG-VE-A08.

Task identifier	XNG-VE-A08-T1
Title	Verify the completeness of the technical performance measures.
Techniques	Inspection and cross-reference.
Actions	1. Check that the most relevant technical performance measures are presented. These measures must contain aspects related to worst case execution time (WCET) of the main services, memory footprints, worst case response time (WCRT) of the main operations, etc.

Table 52: XNG-VE-A08-T1.

Task identifier	XNG-VE-A08-T2
Title	Verify the correctness of the technical performance measures.
Techniques	Inspection.
Actions	1. Check that the technical performance measures obtained are according to temporal characteristics related to determinism and response times considered in the software requirements and design.
	2. Check that the result of the technical performance measures are consistent after analysing by inspection the software code and the results of the unit and integration testing.

Table 53: XNG-VE-A08-T2.

4.3.2.9. Verification of validation testing with respect to TS

Activity identifier	XNG-VE-A09
Title	Verification of validation testing with respect to TS
Development phase	Software validation
Start	When the software validation report is delivered.
End	In CDR according to the ECSS (M30 for De-RISC scheduling)
Objectives	Verify that the software unit testing is complete, consistent, correct, feasible and testable.
Techniques and tools	- PDF reader
	- LibreOffice Calc
Input	- D1.2 (SRS)
	- SDD
	- Software Validation Plan (SValP)
	- Software Validation Specification (SVS)
	- Validation test suite
	- SValR
	- Operational software
Output	Section 5.1.9
Tasks	XNG-VE-A09-T1.
	XNG-VE-A09-T2.
	XNG-VE-A09-T3.
	XNG-VE-A09-T4.

Table 54: XNG-VE-A09.

Task identifier	XNG-VE-A09-T1
Title	Verify the validation testing consistency and traceability.
Techniques	Inspection and traceability analysis.
Actions	1. Check that the software requirements are traceable to the validation test design specifications. 2. Check that the validation test design specification are traceable to the software requirements. 3. Check that exits consistency between the software requirements and software validation specification (test designs). 4. Check that the test designs are traceable to the test cases. 5. Check that the test cases are traceable to the test designs. 6. Check that consistency between the test designs and the software test cases.

Table 55: XNG-VE-A09-T1.

Task identifier	XNG-VE-A09-T2
Title	Verify the correctness and completeness of the software validation.
Techniques	Inspection.
Actions	1. Check that the validation activities proposed in the plan have been performed and completed in the validation. 2. Check that the validation has been performed according to the validation strategy proposed in the plan: types of tests, level of coverage, etc. 3. Ensure that the source code of the validation tests is structured and organised in a uniform manner and style. 4. Check that the source code of the validation tests follows the coding style proposed in the project. 5. Check that the source code of the validation tests is according to the validation test cases specification.

Table 56: XNG-VE-A09-T2.

Task identifier	XNG-VE-A09-T3
Title	Verify the validation test report.
Techniques	Inspection and cross-reference.
Actions	1. Check that the test report includes the results for all validation tests proposed in the validation specification.
	2. Check that the coverage is presented conforming to the validation plan.
	3. Check that if the coverage is not achieved through the test execution, the behave of the non covered code is analysed by inspection.
	4. Check that abnormal termination conditions and unfinished testing are reported and documented.
	5. Check that the results obtained are in line with what is expected.

Table 57: XNG-VE-A09-T3.

Task identifier	XNG-VE-A09-T4
Title	Verify the feasibility and maintainability of the software validation.
Techniques	Inspection.
Actions	1. Ensure that the validation tests are feasible to be maintained in a correct way. Changes in the software requirements can be correctly traced and updated with a reduced impact in the validation specification and validation test suite.
	2. Test results, test logs, test cases and test documentation are maintained under configuration control and changes control.

Table 58: XNG-VE-A09-T4.

4.3.3. Software quality requirement verification

The software quality requirements verification are integrated in several verification activities of the verification plan (refer to section 4.3.2). Each of these activities defines one or more tasks that cover the verification of software quality requirements. The inputs, outputs, methodology and tools are specified in each verification activity and task.

5. Software Verification Report

This chapter reports the status of the verification activities described in the chapter 4.

5.1. Verification activities reporting and monitoring

5.1.1. General

This section includes all the results of the verification activities performed throughout the life cycle of the software.

Each verification task is divided in actions, or tasks, that are verified individually. Each subsection presents the results for each of the tasks, gathered in a table that contains the columns listed in the table Table 59.

Column name	**Contents**
Action	Sub-task being verified.
Description	Description of the verification actions performed.
Result	Result of the verification.

Table 59: Template to report a verification task.

The result of a verification task shall correspond to one of the following:

- *Pass*, when there are no RIDs raised or there are some RID classified as minor.
- *Fail*, when there are some RID classified as major.
- *N/A*, when the sub-task is not applicable.

The complete list of RIDs raised during the verification activities reported in the context of this report can be found in the Appendix B.

5.1.2. Technical specification verification

Action	XNG-VE-A01-T1.
Description	Refer to XNG-VE-A01-T1.
Result	N/A: Traceability to RB not available in the scope of the project.

Table 60: verification task report.

Action	XNG-VE-A01-T2.
Description	Refer to XNG-VE-A01-T2.
Result	N/A: Interface requirements not available in the scope of the project.

Table 61: XNG-VE-A01-T2. verification task report.

Action	XNG-VE-A01-T3.
Description	Refer to XNG-VE-A01-T3.
Result	Pass

Table 62: XNG-VE-A01-T3. verification task report.

Action	XNG-VE-A01-T4.
Description	Refer to XNG-VE-A01-T4.
Result	Pass

Table 63: XNG-VE-A01-T4. verification task report.

Action	XNG-VE-A01-T5.
Description	Refer to XNG-VE-A01-T5.
Result	Pass

Table 64: XNG-VE-A01-T5. verification task report.

Action	XNG-VE-A01-T6.
Description	Refer to XNG-VE-A01-T6.
Result	N/A: There is no logical model available in the scope of the project.

Table 65: XNG-VE-A01-T6. verification task report.

5.1.3. Software architectural design verification

According to the section 4.3.1 and the table 7, this activity is out of the scope of De-RISC as it is not planned to deliver documentation related to the architectural design in the frame of the project.

5.1.4. Software detailed design verification

According to the section 4.3.1 and the table 7, this activity is out of the scope of De-RISC as it is not planned to deliver documentation related to the detailed design in the frame of the project.

5.1.5. Software user manual verification

According to the table 7, a preliminary verification of the software user manual has been carried out in M18. An update of this activity will be performed in M30.

Action	XNG-VE-A04-T1.
Description	Refer to XNG-VE-A04-T1.
Result	Pass

Table 66: XNG-VE-A04-T1. verification task report.

Action	XNG-VE-A04-T2.
Description	Refer to XNG-VE-A04-T2.
Result	Pass

Table 67: XNG-VE-A04-T2. verification task report.

Action	XNG-VE-A04-T3.
Description	Refer to XNG-VE-A04-T3.
Result	Pass

Table 68: XNG-VE-A04-T3. verification task report.

Action	XNG-VE-A04-T4.
Description	Refer to XNG-VE-A04-T4.
Result	Pass

Table 69: XNG-VE-A04-T4. verification task report.

Action	XNG-VE-A04-T5.
Description	Refer to XNG-VE-A04-T5.
Result	Pass

Table 70: XNG-VE-A04-T5. verification task report.

5.1.6. Source code verification

According to the table 7, a preliminary verification of the source code has been carried out in M18. An update of this activity will be performed in M30.

Action	XNG-VE-A05-T1.
Description	Refer to XNG-VE-A05-T1.
Result	N/A: software units and components are not available in the scope of the project (there not exist any SDD).

Table 71: XNG-VE-A05-T1. verification task report.

Action	XNG-VE-A05-T2.
Description	Refer to XNG-VE-A05-T2.
Result	Pass

Table 72: XNG-VE-A05-T2. verification task report.

Action	XNG-VE-A05-T3.
Description	Refer to XNG-VE-A05-T3.
Result	Pass

Table 73: XNG-VE-A05-T3. verification task report.

Action	XNG-VE-A05-T4.
Description	Refer to XNG-VE-A05-T4.
Result	N/A: coverage will be available after the software validation test campaign.

Table 74: XNG-VE-A05-T4. verification task report.

5.1.7. Software unit testing verification

According to the section 4.3.1 and the table 7, this activity will be reported in the next update of this document in M30 due to the unit testing has not started yet at the time of delivering the current document.

5.1.8. Software integration testing verification

According to the section 4.3.1 and the table 7, this activity will be reported in the next update of this document in M30 due to the integration testing has not started yet at the time of delivering the current document.

5.1.9. Verification of validation testing with respect to TS

According to the section 4.3.1 and the table 7, this activity will be reported in the next update of this document in M30 due to the validation testing has not started yet at the time of delivering the current document.

5.2. Margin and technical budget status

5.2.1. Technical budgets and margins computation

As in the framework of the qualification there is no final user application, this activity cannot be performed because the margins and the specific requirement of budgets must be imposed by the final application.

However, the product being qualified provides performance measures of the services and operations supplied, which will be helpful in the elaboration of the schedulability and the technical budget analyses. Theses figures could be provided in M30.

5.2.2. Software budget

For the same reason as previously stated, providing performance measures of the services and operations supplied by the software will be helpful in the elaboration of the software budget. These figures could be provided in M30.

5.2.3. Schedulability simulation and analyses

Schedulability simulation and analyses are not applicable to this software.

5.3. Numerical accuracy analysis

Numerical accuracy analysis is not applicable to this software.

A. Template for the RIDs

The technical appendix A is available in the folder *De-RISC_D3.2_XVR_Appendix_A*. Inside, the file *Verification_RIDs_template.ods* is available and self-contains the RID procedure.

B. List of RIDs

The complete list of RIDs raised during the verification activities reported in this document is available as a technical appendix B in the folder *De-RISC_D3.2_XVR_Appendix_B*. The file is titled *Verification_RIDs_report.ods*. Those RIDs marked as closed have been agreed, and the recommendations provided have been incorporated to the reviewed CIs.