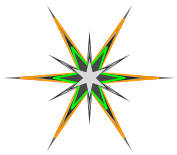


S T S A R C E S

Standards for Safety Related Complex Electronic Systems

Annex 3

Tools for Software fault avoidance

Task 1.2: Guide to evaluating software quality and safety requirements

Final Report of WP1.2

Philippe Charpentier

INRS

European Project STSARCES
Contract SMT 4CT97-2191

Contents

SUMMARY

This document is a complement to the document "Software Quality and Safety Requirements" which defines requirements applicable to embedded software. It is intended to guide evaluators in evaluating an embedded software with respect to the software quality and safety requirements that the designer of a system must satisfy.

This document present:

- the principles of evaluation for each requirement : verification recommendations or additional questions to be raised in order to proceed with the evaluation of software products with acceptance or refusal criteria;
- the principles of determining the software requirement levels.

All this information is meant to assist the analyst judge the capacity of the embedded software to satisfy each of the applicable requirements. The guidance sheets provided in this document help the evaluator to collect information and proof to allow an expert judgement on whether the software product is acceptable or not. It is a tool for the evaluator to determine, from different angles when necessary, the responses to each of the requirements.

1 INTRODUCTION

1.1 AIM OF THE DOCUMENT

This document is intended to guide evaluators in evaluating a software product with respect to the software quality and safety requirements that the designer of a system must satisfy.

It is a complement to the document[1] "Software Quality and Safety Requirements" which defines all the requirements applicable to the software to be assessed. It should be borne in mind that these requirements concern software forming part of a machinery control system ensuring safety functions. **Highly critical systems (aeronautics, nuclear, etc.) are excluded, as is application software** such as PLCs.

To achieve this aim, the guide is composed of two parts:

- a first part presenting the list of points to be checked - corresponding to the list of requirements detailed in the document[1] "Software Quality and Safety Requirements" - along with the phase during which it is preferable to evaluate these points.
- a second part corresponding to the annexes laying down :

  - the principles of evaluation for each requirement : verification recommendations or additional questions to be raised in order to proceed with the evaluation of software products with acceptance or refusal criteria;

  - laying down the principles of determining the software requirement levels;

  - the general practices regarding software evaluation : evaluation strategy, organisation, recommendations for the evaluator.

All this information is meant to assist the analyst judge the capacity of the software product to satisfy each of the requirements applicable to the software undergoing evaluation.

1.2 TARGET PUBLIC OF THE DOCUMENT

The present guide is meant for the analyst and is not intended for diffusion to the designers of the software to be evaluated.

It should be noted that software is not an isolated product and that the analyst analyses the entire system. This guide is limited to software evaluation with respect to the software quality and safety requirements, this software evaluation contributing to the analysis of the overall system.

1.3 MODE OF USE

The reader’s attention is drawn to the fact that this guide is a series of guidance sheets that help the evaluator collect information and proof to allow an expert judgement on whether the

[1] SOFTWARE QUALITY AND SAFETY REQUIREMENTS : STSARCES Project - WP 1.2 / Aspect 1 - INRS – Feb 2000.

software product is acceptable or not. They are not exhaustive, and have not been designed to be used as a verification check-list. They guide the expert in the evaluation approach, and throughout the audit the expert must remain attentive to any information the designer may communicate. This information may be useful in directing the evaluation in real time or later when assessing another aspect of the software product.

The starting point of the evaluation is the software requirement level, which sets the requirements that the software must satisfy. This guide is not modulated in function of this level as each requirement is taken separately with a view to answering the question : " Does the software product satisfy this requirement ? " The evaluator should refer to the document " Software Quality and Safety Requirements " to determine how applicable a requirement is to the software undergoing evaluation.

This guide is a tool for the evaluator to determine, from different angles when necessary, the responses to each of the requirements. To achieve this, the evaluator should have a basic knowledge of software quality or software development. This knowledge should have been acquired through experience of developing software in an environment imposing high quality constraints and/or through specific training. It is also desirable that the evaluator has undergone training in audit techniques.

1.4 OVERVIEW OF THE DOCUMENT

In addition to this introductory chapter, the document is composed of the following chapters which contain recommendations for the evaluation :

- chapter 2 : evaluation for quality and safety requirements
- appendix A : requirements evaluation

  - evaluation of the requirements relative to the software product,
  - evaluation of the requirements relative to the software development process,
  - evaluation of the requirements relative to software verification and validation.

- appendix B: general principles for determining the software requirement level

2 EVALUATION OF QUALITY AND SAFETY REQUIREMENTS

2.1 PRESENTATION

The following table represents a check-list of requirements to be verified by the analyst in order to ensure that the software product and the software process conform to the requirements laid down in the document "Software Quality and Safety Requirements". It is also gives the preferential phase during which it is preferable to check these requirements.

2.2 CHECK-LIST

				Level		Preferential phase
				1	2	
1 - Introduction :						
	1.3 - Instructions					
		Point 1.1	Request for deviation with regard to the requirements mentioned in this document.	O	O	***At any time in the development process***
2 - Requirements for software product :						
	2.2.1 - Interface with system architecture					
		Point 1.2	Determination of safety requirements on the basis of system safety analysis.	/	O	***Specification review***
		Point 1.3	List of constraints imposed on the software by the hardware architecture.	O	O	***Specification review***
	2.2.2 - Software Specification					
		Point 1.4	Checklist of the content of a given software specification.	O	O	***Specification review***
		Point 1.5	Specification of requirements in each mode.	O	O	***Specification review***
	2.2.3 - Software that can be parameterised by the user					
		Point 1.6	Specification of parameters and independence with regard to the software product.	R	R	***Specification and Design review***
		Point 1.7	Parameter protection mechanisms and fault tolerance mechanisms.	R	O	***Specification review at the earliest, final evaluation review***
	2.2.4 - Pre-existing Software					
		Point 1.8	Earliest possible indication of pre-existing software use.	O	O	***First review***
		Point 1.9	Configuration management for pre-existing software.	O	O	***Final review***
	2.2.5 - Software Design					
		Point 1.10	Check-list of content of a given software design description.	O	O	***Design review***

			Level		Preferential phase
			1	2	
		Point 1.11 Software architecture properties: modularity, interface between modules, function of modules resulting form specifications.	O	O	***Design review***
		Point 1.12 Separation of safety and non safety parts.	O	O	***Design review***
	2.2.6 - Development Languages				
		Point 1.13 Language function of the application and limited language subset.	R	O	***Design review***
	2.2.7 - Coding				
		Point 1.14 Requirements for the source code.	O	O	***Final review***
		Point 1.15 Rules for coding.	R	O	***Final review***
3 - Requirements for software development process :					
	3.1.2 - Software Lifecycle Requirements				
		Point 2.1 Formalised description of the lifecycle (example: in a Software Quality Plan)	O	O	***First review***
		Point 2.2 Description of phases lifecycle: input and output products, design and verification activities.	O	O	***First review***
	3.2.1 - Software Quality Assurance Requirements				
		Point 2.3 Documentation for the Software Quality Assurance Requirements, (example: in a SQP).	R	O	***First review***
	3.2.1 - Safety Supervision and Management Requirements				
		Point 2.4 Follow-up at each phase of the safety constraints.	R	R	***Evaluation review over the course of the project.***
	3.3.1 - Documentation Management Requirements				
		Point 2.5 List of the software documentation to be supplied at the beginning of a project, (example: in a SQP).	O	O	***First review, Final review***
		Point 2.6 Documentation management.	O	O	***First review***
		Point 2.7 Establishment of documents at each stage of the lifecycle and traceability.	R	O	***Intermediate reviews***
	3.4.1 - Configuration and archiving Management Requirements				
		Point 2.8 Configuration management procedures.	O	O	***Final review***
		Point 2.9 Versions of configuration articles and software version.	O	O	***Final review***
		Point 2.10 Configuration management audit.	O	O	***Final review***
		Point 2.11 Software version archiving procedures.	O	O	***Specification review at the earliest, final evaluation review***
	3.4.2 - Software modifications Management				
		Point 2.12 Management of software modifications.	O	O	***Final review***
		Point 2.13 Content of software modification files.	R	O	***Final review***
	3.5.1 - Development Tools Requirements				
		Point 2.14 No optimisation object code.	R	O	***Final review***
		Point 2.15 Validity of tests in cases of modification of compilation options, of the compiler, or the linker.	R	O	***Final review***

	Level		Preferential phase
	1	2	
Point 2.16 Identification of development tools (example in a SQP).	O	O	***Final review***
3.6.1 - External Sub-contracting Requirements			
Point 2.17 Respect of present requirements by sub-contractor.	O	O	***First review***
Point 2.18 Control of sub-contractors by the designer.	O	O	***Final review***
3.7.1 - Executable Code Production Requirements			
Point 2.19 Recording of compilation options in a Version Sheet.	O	O	***Final review***
3.7.2 - Software Installation and Exploitation Requirements			
Point 2.20 Recording of failures during the installation and the use of the software.	O	O	***After the final review***
4 - Requirements for Software Verification :			
4.2 - General Verification and Validation Requirements			
Point 3.1 Verification of the conformity to the requirements and of technical aspects subject to evaluation.	O	O	***All the reviews***
Point 3.2 Evaluation of a given software version.	O	O	***Final review***
4.3.1 - General Verification Requirements			
Point 3.3 Verification Report for each verification activity.	R	O	***All the reviews***
4.3.2 - Reviews Requirements			
Point 3.4 Specification review and specification verification activities.	R	O	***Specification review***
Point 3.5 Content of verification activities and design.	O	O	***Design review***
Point 3.6 Validation review enabling deliverance of the software qualification.	O	O	***Final review***
Point 3.7 Review documentation and follow-up to activities decided during reviews.	O	O	***Final review***
4.3.3 - Code Verification (source code and data) Requirements			
Point 3.8 Conformity to design documents and programming rules.	R	O	***Final review***
4.4.1 - General Validation Requirements			
Point 3.9 Test plan: strategy/ techniques/ verification tools and independent evaluation.	R	O	***First review***
Point 3.10 Non-regression tests for a new version.	O	O	***First review***
Point 3.11 Content of test procedures.	R	O	***First review***
Point 3.12 Possibility to re-perform tests at the analyst's request.	R	O	***Final review***
4.4.2 - Software Specifications Verification Requirements : Validation Tests			
Point 3.13 Validation test coverage and traceability matrix.	O	O	***Specification and final review***
Point 3.14 Checklist of validation report content.	O	O	***Specification and final review***
4.4.3 - Software Design Verification Requirements : Software Integration Tests			
Point 3.15 Integration test coverage and traceability matrix.	R	O	***Design and final review***
Point 3.16 Analysis of the impact of modification on the software (non-regression tests).	R	O	***Design and final review***
Point 3.17 Checklist of the content of the integration report.	R	O	***Design and final review***
4.4.4 - Detailed Module Design Verification Requirements : Module Tests			

			Level		Preferential phase
			1	2	
		Point 3.18 Unit test coverage and traceability matrix.	/	R	***Design and final review***
		Point 3.19 Checklist of the content of unit test reports.	/	R	***Design and final review***

3 APPENDIX A : REQUIREMENTS EVALUATION

3.1 INTRODUCTION

As in the document presenting the different requirements, the requirements and their verifications are presented in three sections:

- Section 2: requirements dealing with the software product :

  - Interface with system architecture
  - Software Specification
  - Software that can be parametrized by the user
  - Pre-existing Software
  - Software Design
  - Development Languages
  - Coding

- Section 3: requirements dealing with the software development process :

  - Development process
  - Organisation
  - Documentation
  - Configuration and Software Modifications Management
  - Tools
  - External Sub-contracting
  - Reproduction, delivery.

- Section 4: requirements dealing with software verification :

  - Software Verification :

    . reviews
    . code verification

  - Software Tests :

    . Validation Tests
    . Integration Tests
    . Module Tests

SUBJECT : CO-ORDINATION WITH THE ANALYST FOR SOFTWARE EVALUATION	**Requirement n° 1.1**

Requirement

Any deviations from the requirements presented in this document should be pointed out by the applicant to the analyst and should be approved by the latter.

Aim of the evaluation

The evaluation consists in determining whether the deviations communicated by the designer are acceptable or not.

Preferential evaluation phase and supports necessary for the evaluation

The applicant can submit a deviation at any time in the development process

Recommendations / techniques for the evaluation

The developer must justify any deviation.

Evaluation consists, on reception of the deviations, in analysing its importance and the justification given with respect to the corresponding requirement (s).

This means, for each deviation, taking the decision in accordance with the elements contained within the evaluation guidance sheet.

Specific consultation must take place with the designer for each deviation.

Acceptance / refusal criteria

Refusal:

- absence of justification.

The other criteria depend on the requirement in question, the extent (full or partial non respect of a requirement) and the category of requirement (eliminating or not).

3.2 REQUIREMENTS FOR SOFTWARE PRODUCT

3.2.1 Presentation

The software product requirements concern :

- Interface with system architecture
- Software Specification
- Software that can be parametrized by the user
- Pre-existing Software
- Software Design
- Development Languages
- Coding

3.2.2 Requirements to be evaluated relative to the software product

The corresponding requirement evaluation guidance sheets are presented below.

SUBJECT : INTERFACE WITH SYSTEM ARCHITECTURE	**Requirement n° 1.2**

Requirement

Software safety requirements as well as the determination of expected events should arise from safety analyses at system, functional and hardware level, etc.

Aim of the evaluation

Verify traceability between the safety requirements resulting from the system and the software safety requirements as expressed in the software specification document.

Preferential evaluation phase and supports necessary for the evaluation

The evaluation can be conducted just after the end of the software specification. It is based on the system specification documents (or contractual specification), on the software specification and the system safety analysis documents.

Recommendations / techniques for the evaluation

Verification is undertaken with the help of a traceability matrix, which brings the system safety requirements and the functional and non-functional requirements of the software together. Examples are:

- safety functions covering a safety requirement of the system
- fault tolerance functions
- periodic tests
- software function performances in order to meet system safety requirements
- V & V requirements with regard to software functions in order to check that the safety requirement of the system have been met

Expected events at the software level must be characterised from expected events at the system level and functional, hardware and software decomposition (example: fault tree analysis, analysis of the failure causes etc.).

Acceptance / refusal criteria

Refusal:

- Safety requirement at the system level not covered by a safety requirement in the software as expressed in the specification document.

SUBJECT : INTERFACE WITH SYSTEM ARCHITECTURE	Requirement n° 1.3

Requirement

The list of constraints imposed by hardware architecture on software should be defined and documented.

Consequences of any hardware/software interaction on the safety of the machine or system being monitored should be identified and evaluated by the designer, and taken into account in the software design.

Aim of the evaluation

The evaluation consists first in ensuring that the constraints are known (identified and documented) by the developer. It serves as the reference for the other requirements covering the development process (hardware/software interaction). The evaluation consists to obtain the confidence that the safety of the system is not downgraded by unforeseen interactions between the hardware and software.

Preferential evaluation phase and supports necessary for the evaluation

The evaluation of the list of constraints can be conducted just after the end of the software specification. It is based on the system specification documents (or contractual specification) and on the software specification.

For the consequences of any hardware/software interaction, several phases are possible :

- preliminary theoretical analysis at the system specification,
- analysis at the software design,
- software error analysis at the coding.

Recommendations / techniques for the evaluation

The documentation can take any form facilitating comprehension : text, dictionary of interfaces, interface diagrams, timing (input stimuli : data/events, system state, actions performed).

The justifications supplied can be :

- feedback from similar hardware,
- a theoretical analysis (feared events, possible causes, consequences),
- test results.

The evaluation is to be carried out at system level (software + hardware). The consequences of software errors must be in conformity with the class of equipment.

The following points are also to be evaluated :

- erroneous software command,
- failure of a sensor (stuck at a logic state, change of state, erroneous measurement).

Acceptance / refusal criteria

Refusal :

- absence of documentation on the interfaces,
- documentation incoherent or insufficient to allow evaluation of the hardware/software interfaces,
- absence of a system analysis of the consequences of a software or hardware error.

SUBJECT : SOFTWARE SPECIFICATIONS	Requirement n° 1.4

Requirement

Software specifications should take the following points into account:

a) safety functions with a quantitative description of the performance criteria (precision, exactness) and temporal constraints (response time), all with tolerances or margins when possible,
b) non safety functions with a quantitative description of the performance criteria (precision, exactness) and temporal constraints (response time), all with tolerances or margins when possible,
c) system configuration or architecture,
d) instructions relevant to hardware safety integrity (programmable electronic systems, sensors, actuators, etc.),
e) instructions relevant to software integrity and the safety of safety functions,
f) constraints related to memory capacity and system response time,
g) operator and equipment interfaces,
h) instructions for software self-monitoring and for hardware monitoring carried out by the software,
i) instructions that allow all the safety functions to be verified while the system is working (on-line testing).

Aim of the evaluation

This evaluation :

- verifies that the software specification is both complete and correctly defined,
- ensures that the specification is characterised by quantified magnitudes that can therefore be tested or verified,
- ensures that the system has mechanisms verifying that it remains capable of carrying out its task and that any fault is detected,
- ensures that the safety functions have indeed been verified while the system is operating.

Preferential evaluation phase and supports necessary for the evaluation

This evaluation can be carried out as from the end of software specification. It is based on the software specification and the safety analysis documents.

Recommendations / techniques for the evaluation

The evaluation is carried out by documentary analysis and appraisal of the specification documents.
The evaluator verifies that the information corresponding to each of the required headings is present in the specification documents.

If there is information missing, the evaluator can broaden the scope to earlier documents (preliminary design principally).

The objectives must be defined and able to be verified. The memory size and CPU load are provisional measurements made during the specification that can be refined over the course of the development.

Possible margins (free memory, available CPU) provide scope for unforeseen development and system behaviour modelling incidents. They can also be provisionally employed for future system changes.

These measurements on the final system (memory size, CPU load, performance) must be carried out by the designer.

The self monitoring facilities must be adapted to the operating constraints :

- continuous operation : is the system still capable of carrying out its task ?
- environment : can external influences (temperature, electromagnetic disturbances) perturb the integrity of the system ?

The following events must have been taken into account by the designer :

- sensor : stuck at fault, erroneous of state or measurements,
- actuator: stuck at fault, emission of erroneous commands.

Analysis of Software Failure Modes and of their Effects from a functional model resulting from the software specification can be carried out in order to check the non-application of expected events in the software when confronted with failures such as:

- sensors
- actuators
- support hardware for the software
- networks
- parameter setting
- common mode failures (libraries, global data, etc).

Acceptance / refusal criteria

Refusal :

- the safety objectives have not been defined or are not in conformity with the task required of the equipment.
- absence of quantitative description (precision, accuracy, response time).
- safety objectives and operating constraints not specified.
- self test facilities poorly adapted or inefficient.
- existence of safety functions not verified in operation without justification.

SUBJECT : SOFTWARE SPECIFICATIONS	**Requirement n° 1.5**

Requirement

Functional requirements should be specified for each functional mode. The transition from one mode to the other should be specified.

Aim of the evaluation

This evaluation ensures that all the possible operating modes have indeed been taken into consideration.

Preferential evaluation phase and supports necessary for the evaluation

This evaluation can be carried out as from the end of software specification, and is conducted primarily with the specification documents.

Recommendations / techniques for the evaluation

- analyse the justification for the absence of degraded mode : the system has a mode of safe stop, the operator knows that the system is out of operation.
- analyse the feedback and the errors detected during integration of the software product.

Acceptance / refusal criteria

Refusal :

- incompleteness of the definition of operating modes, making analysis of the system impossible,
- operating modes not specified without justification.

SUBJECT : SOFTWARE THAT CAN BE PARAMETRIZED BY THE USER	**Requirement n° 1.6**

Requirement

The parameters should be formally specified (type, relations, …) in the form of memory arrays. Moreover, the software and the parameters should be capable of independent evolutions.

Aim of the evaluation

This evaluation aims to check the completeness of the parameters specification and the separation between parameters and the software properly speaking.

Preferential evaluation phase and supports necessary for the evaluation

The evaluation can be conducted in two phases:

1) just after the end of the software specification. It is based on the software specification.

2) during production of the code.

Recommendations / techniques for the evaluation

- Check the completeness of the parameter specification (exhaustiveness of parameters, min and max, data type, ordering between the parameters, …).

- Determing, by examining the parameter specification document if a parameter elaboration strategy has been adopted and analyse if this will guarantee the independence of parameters in relation to the software.

- Check the independence of parameter data in relation to the code by static analysis or critical reading of the code.

Acceptance / refusal criteria

Refusal:

- incomplete nature of parameters specification,
- absence of effective separation between parameters and software.

SUBJECT : SOFTWARE THAT CAN BE PARAMETRIZED BY THE USER	**Requirement n° 1.7**

Requirement

Software specifications should define mechanisms that can be used to prevent the possibility of any parameters set by the user can affect the system safety. In so far as modifiable parameters are concerned, these mechanisms should provide protection against :

- **undefined or invalid initial values,**
- **values falling outside functional limits**,
- **data alteration.**

The definition of software parameters by users should be kept within the limits established by the system specifications approved by the analyst.

Aim of the evaluation

To ensure that the user, by modifying parameters, has no possibility of rendering the system unsafe, and that the parameter setting possibilities of the software correspond to the specifications and in turn to the verifications that will be conducted.

Preferential evaluation phase and supports necessary for the evaluation

End of specification review at the earliest, final evaluation review. Specification documents, design, and source code are necessary.

Recommendations / techniques for the evaluation

- analyse the mechanisms described in the specifications,
- verify the test strategy of these mechanisms (are these mechanisms indeed activated ? ; has software operation been tested in different parameter-setting configurations ?),
- verify the traceability between specification, design and code with respect to parameter setting, and analyse the coherence (type of data with parameter setting capability, maximum and minimum values),
- verify, if possible through tests, that it is not possible to exceed the limits set by the specification (modification of an unforeseen variable, entry of an out of limit value),
- analyse the tests carried out : do they indeed cover the parameter variation range and has the influence of all the parameters been studied ?

Acceptance / refusal criteria

Refusal :

- absence of protective mechanisms against the effects of parameter setting on system safety,
- insufficient validation of the operation of these mechanisms,
- inefficiency of the mechanisms proposed,

  incoherence between the specification and the parameter settings possible leading to non validated operating domains,
- possibility of modifications by the user going beyond the authorised limits.

SUBJECT : PRE-EXISTING SOFTWARE	Requirement n° 1.8

Requirement

The designer should indicate the use of pre-existent software to the analyst, and it is the designer's responsibility to demonstrate that pre-existent software has the same level as the present requirements.

Such a demonstration should be done:

- **either by using the same verification activities on the pre-existent software as on the rest of the software,**
- **or through practical experience where the pre-existent software has functioned on a similar system in a comparable executable environment (e.g. it is necessary to evaluate the consequences of a change of the compiler or of a different software architecture format).**

Aim of the evaluation

To ensure ahead of development that consultation takes place with the analyst to identify difficult points and to deal with them appropriately and that the pre-existing software product satisfies quality criteria equivalent to those of the software developed by the designer.

Preferential evaluation phase and supports necessary for the evaluation

First review or contacts with the designer before development or at as early as possible in the evaluation. All the documents relative to these pre-existing software products are necessary.

Recommendations / techniques for the evaluation

The analyst conducts interviews with the project manager to determine the origin of all the software products used. In the pre-existing software category, the following must be classified :

- software products purchased commercially (sequencer, real-time monitor, programme library, etc.),
- software products stemming from an earlier project (whether developed by another team, a subsidiary, a sub-contractor, etc.).

Conduct the evaluation of this type of software product in an identical way to the rest of the software : verify that all the documents exist (design, code, tests) and that the test level of this software is comparable with the rest of the software.

When difficulties arise in the demonstration (absence of certain documents, unknowns in certain phases like the unit test), it is possible to have recourse to the in-service experience on other projects. Attention should then be paid to the representativeness of this experience : does it indeed involve the same software product ? Is the execution environment comparable ?

Acceptance / refusal criteria

It is advantageous for the designer to establish as early as possible a consultation on the subject in order not to put back the evaluation through the late discovery of significant obstacles due to the absence of certain documents for these software products.

The absence of notifying the use of pre-existing software products is therefore a priori not grounds for refusal.

Refusal :

- absence of control of the behaviour of these software products (no experience, no tests for example).
- associated documentation insufficient, not allowing determination of the exact content (no design documents for example).

SUBJECT : PRE-EXISTING SOFTWARE	**Requirement n° 1.9**

Requirement

Pre-existent software should be identified using the same configuration management principles that were applied to the rest of the software.

Aim of the evaluation

To ensure that it is known how to identify changes of versions of pre-existing components integrated into the software.

Preferential evaluation phase and supports necessary for the evaluation

Final evaluation review. The evaluation is based on the documents associated with the pre-existing software, supplemented by source and link edition files.

Recommendations / techniques for the evaluation

- verify that the documents include an identification of the version and that this is coherent with the source used.
- verify the versions of modules, possibly by employing the link edition files.

Attention : this requirement does not impose that the pre-existing software identifiers are identical to the rest of the software. It is only required that it is known how to identify the versions. It should be ensured that all the necessary verifications (impact analysis, tests, etc.) have been carried out again when the version of the pre-existing software product is changed.

Acceptance / refusal criteria

Refusal :

- total absence of pre-existing software version identification making it impossible to control the content of the software to be evaluated,
- changes to versions of pre-existing software without impact analysis or minimum non-regression tests.

SUBJECT : SOFTWARE DESIGN	Requirement n° 1.10

Requirement

Description of the software design should include at the very least:

- **a description of the software architecture that defines the structure decided on to satisfy specifications,**
- **a description of inputs and outputs (e.g. in the form of an internal and external data dictionary), for all the modules making up the software architecture,**
- **sequencers and interruptions,**
- **global data,**
- **a description of each software module (inputs/outputs, algorithm, design particularities, etc.),**
- **libraries used,**
- **pre-existent software used.**

Aim of the evaluation

The evaluation must determine whether the design has been described completely and coherently, and in such a way as to satisfy the software specifications.

Preferential evaluation phase and supports necessary for the evaluation

At the end of the design phase. The evaluation is based on all the software specification and design documents.

Recommendations / techniques for the evaluation

- verify that one element of design documentation corresponds to each point laid down in the requirement.
- analyse the possible links between the operational modes and the periodic test (absence of common failures). Example: use of identical sub-routines that could lead to the periodic tests not detecting certain faults. The periodic test must contribute effectively to detecting faults.

Analysis of software Failure Modes and of their Effects from a functional model resulting from the design documents can be carried out to check the potentially dangerous expected software events when confronted with software module failures and in particular failures in the common modules (analysis of common mode failures)

Acceptance / refusal criteria

Refusal :

- software design does not exist or is very inadequate.

SUBJECT : SOFTWARE DESIGN	Requirement n° 1.11

Requirement

Software should be modular in order to facilitate its maintenance:

- **each module or group of modules should correspond, if possible, to a function in the specifications,**
- **interfaces between modules should be as simple as possible.**

Aim of the evaluation

The evaluation must ensure:

- the effective modularity of the software
- the links between the modules and the functions in the specification
- the limited number of exchanges of data between modules.

Preferential evaluation phase and supports necessary for the evaluation

At the end of the coding phase. The evaluation is based on the detailed design documents and the source code.

Recommendations / techniques for the evaluation

- analyse with the help of a traceability matrix which brings the modules, the specification functions, the distribution of functions and the completeness of the modules into correspondence.
- analyse the modularity of the software architecture : functional cohesion of the modules.
- analyse the modularity of the code :
  . coupling of the compilation units (by sub-programmes, by data),
  . size of the compilation units,
  . number of sub-routines per compilation unit.

The analyst can have recourse to the static analysis results supplied by the designer (if the latter has carried out this type of analysis).

Otherwise, the evaluator carries out sampling and manual analysis.

Acceptance / refusal criteria

Refusal :

- size of sub-routines too high (>150 lines without justification),
- excessive depth of in call graph (> 5 calls per module),
- traceability “function/ module” impossible.

SUBJECT : SOFTWARE DESIGN	**Requirement n° 1.12**

Requirement

Software should be designed to limit those parts associated with safety:

- **data/functional architecture: strict limitation of global variables, implementation of operators on state variables (visibility),**
- **control of the layout of arrays in memory (risk of array overflows).**

Aim of the evaluation

This evaluation ensures that the parts associated with safety have been limited.

Preferential evaluation phase and supports necessary for the evaluation

The evaluation can take place at the end of software design. It is based on the software specification and design documents, and on the safety analysis.

Recommendations / techniques for the evaluation

- analyse the functions provided by the system and the functions not linked to safety,
- analyse the possible interactions between the functions :

  . risk of CPU capture by a non safety related function,

  . risk of wiping out common data,

  . risk of modifying common data.
- study whether different development principles (documentation, tests, etc.) have been employed for the different parts of the software ?.
- at the final review, analyse the problems encountered during integration and validation to ensure the relevance of the solutions retained.

Acceptance / refusal criteria

Refusal :

- absence of partitioning between parts linked to safety and parts not linked to safety although different development principles have been employed.

SUBJECT : DEVELOPMENT LANGUAGES	Requirement n° 1.13

Requirement

The selected programming language should correspond to the characteristics of the application, and should be fully and clearly defined or at least limited by clearly defined characteristics.

Aim of the evaluation

This evaluation ensures the adequacy of the language with the application to be developed and the maturity of the language employed.

Preferential evaluation phase and supports necessary for the evaluation

First evaluation review. The programming and coding manuals of language are necessary.

Recommendations / techniques for the evaluation

The characteristics of the application encompassed by this evaluation are :

- its size,
- its type : industrial, scientific, management software, etc.,
- its real-time performance constraints.

The evaluation ensures that the language is adapted to its characteristics :

- capabilities of establishing data types: possibility of data description adapted to the complexity (structure, tables, index, etc.) and to the types represented (float, long integers etc.)
- existence of error processing mechanisms supplied,
- existence of real-time primitives adapted to needs (interruption management, task management, etc.).

This evaluation depends on the type of programming language :

- Assembler :
  - . verify that the instructions have been defined.
- high level language :
  - . verify that an international standard defining it exists (example : C ANSI),
  - . the case of "dialects" (extension or limitation of an internationally defined language) must be studied in detail.

Remark : by programming language is meant the language used by the designer to write the source code of the programme. It does not include any possible language placed at the disposal of users to set parameters or programme the system, such as is the case with a PLC. Besides, this latter category of system does not fall within the scope of the " Software Quality Requirements " document.

Acceptance / refusal criteria

Refusal : non documented and non validated macro-instruction based language.

SUBJECT : CODING	**Requirement n° 1.14**

Requirement

The source code should:

a) be readable, understandable, and subject to tests,
b) satisfy design specifications of the software module,
c) obey the coding manual instructions.

Aim of the evaluation

This evaluation ensures the quality of the code and its conformity with the design.

Preferential evaluation phase and supports necessary for the evaluation

End of coding. Design documents, code, and coding manual are necessary.

Recommendations / techniques for the evaluation

- analyse the relevance of source comments by sampling sources (that they are not too close to the code),
- analyse the conformity of the detailed design (when foreseen in the development process).
- analyse if the coding rules have been respected (in relation to the coding manual of the designer).

  If there is no manual, the evaluator refers to the coding rules proposed in the annex of the document " Software Quality and Safety Requirements " in order to evaluate the quality of the code.

  The absence of a detailed design at the end of development is not grounds for refusal (it can, for example, be included in the source code if this is foreseen in the development process).

Acceptance / refusal criteria

Refusal :

- numerous cases of incoherence between the detailed design and the code,
- non respect of the coding rules.

SUBJECT : CODING	Requirement n° 1.15

Requirement

The coding rules applicable to a given software product should be outlined in detail in a coding manual and used to develop the software. The coding manual should :

- **indicate what programming principles should be applied and prohibit any uncertain language aspects,**
- **describe rules for source code presentation and documentation,**
- **include conventions used in naming components, subroutines, variables and constants.**

Aim of the evaluation

This evaluation ensures :

- that a reference set of coding rules exists, is known to the developers, and is being applied.
- that the unsafe characteristics of the language have been identified in the coding manual and that they are not being used.
- that the rules of source code presentation and documentation are adequate and have indeed been respected to produce the source code.
- that the coding manual includes the conventions for naming, and that they are being respected to produce the source.

Preferential evaluation phase and supports necessary for the evaluation

The evaluation of the coding manual must take place in preference before the coding phase. The source code at the end of the coding phase is the support to verify that the coding manual is being applied.

Recommendations / techniques for the evaluation

On the basis of the coding rules of the annex of the document " Software Quality and Safety Requirements ", it is recommended to proceed as follows :

- have recourse to the results of the verification carried out by the designer (inspection reports, etc.), if they exist.
- see if the coding manual contains protection against unsafe characteristics of the language.
- compare the coding rules with the coding rules in the annex of the document " Software Quality and Safety Requirements ",
- verify the existence of source code presentation and documentation rules in the coding manual (structuring of IF ... THEN ... ELSE, comments, module identification, etc.).
- verify the existence of conventions (naming of modules, subroutines),
- by sampling on the source modules, choosing a representative sample. Main criteria : programmer, types of functions, language when several languages are used, and modifications carried out in the case of a new evaluation.
- verify respect of the coding manual rules by sampling on the source modules,

- the use of software tools can be justified on large-scale projects or in firms with numerous projects. The investment does however remain considerable, and manual verifications can suffice on the types of projects falling within the scope of this document.

The absence of written rules is not grounds for refusal:

- for a small team (1 to 2 people). The rules proposed in annex in the document " Software Quality and Safety Requirements " then apply by default and must be respected.

- if all the sources in fact respect a common presentation at the end of development. The designer must then foresee formalising this rule (quality plan for example) to ensure its future application.

Acceptance / refusal criteria

Refusal :

- absence of coding rules and use of unsafe language characteristics.
- non-respect of the coding manual on several modules.
- unjustified non respect of coding rules important for safety.

3.3 REQUIREMENTS FOR SOFTWARE DEVELOPMENT PROCESS

3.3.1 Presentation

The software developpement process requirements concern :

- Development Process
- Organisation
- Documentation
- Configuration and Software Modifications Management
- Tools
- External Sub-contracting
- Reproduction, delivery.

3.3.2 Requirements to be evaluated relative to the software development process

The corresponding requirement evaluation guidance sheets are presented below.

SUBJECT : SOFTWARE LIFECYCLE	**Requirement n° 2.1**

Requirement

The software development lifecycle should be specified and documented (e.g in a Software Quality Plan). The lifecycle should include all the technical activities and phases necessary and sufficient for software development.

Aim of the evaluation

To ensure that all the development activities have been foreseen, that the phases are of a realistic duration, and that the main documents have been laid down.

Preferential evaluation phase and supports necessary for the evaluation

First evaluation review. The evaluator employs all the planning elements available (planning, development plan or quality plan if it exists, etc.) and finishes off with interviews.

Recommendations / techniques for the evaluation

Verify the presence of all the phases (specification, design, coding, tests), ensuring that the test phases have not been minimised (not less than 30% of the total time).

If several evaluations have taken place over the course of development, ensure that activities are planned and that there is no reduction in the duration of phases in order to " stick to deadlines ".

Ensure the coherence of the software development with the hardware development (will the hardware be available for the validation) or the development of specific test facilities.

This requirement is of the utmost importance as it allows a structured development to be required.

Acceptance / refusal criteria

Refusal :

- no description of the development cycle during an evaluation at the start of development.

SUBJECT : SOFTWARE LIFECYCLE	**Requirement n° 2.2**

Requirement

Each phase of the lifecycle should be divided into its elementary tasks and should include a description of:

- **inputs (documents, standards etc.),**
- **outputs (documents produced, analytical reports, etc.),**
- **activities to be carried out,**
- **verifications to be performed (analyses, tests, etc.).**

Aim of the evaluation

To ensure, in detail, of the content of each phase of the development cycle.

Preferential evaluation phase and supports necessary for the evaluation

First evaluation review, the development planning document(s) should be used, completed with interviews.

Recommendations / techniques for the evaluation

Verify that each phase indeed has got documents at the output, and that the verifications required for the corresponding level have been foreseen.

If the evaluation has taken place in several stages, verify that the activities already carried out are in conformity with the descriptions.

Acceptance / refusal criteria

Refusal :

- no description of the development cycle during an evaluation at the start of development.

SUBJECT : SOFTWARE QUALITY ASSURANCE	**Requirement n° 2.3**

Requirement

The programme used to guarantee software quality should be well-documented (e.g in a Software Quality Plan) and include at least:

- **the organisation, the people who are responsible for quality assurance, development and tests, and the required independence,**
- **the quality assurance activities included in the software lifecycle (*examples of methods, reviews, inspections*),**
- **any documents produced (*reports, etc.).***

Aim of the evaluation

To ensure, a priori, that adequate arrangements have been described and that they have subsequently been respected.

Preferential evaluation phase and supports necessary for the evaluation

First evaluation phase to verify the arrangements made. All the other phases to check that they are being applied. The evaluator employs the quality plan or procedures if they exist, and the proof of activity (reports, etc.) as the basis.

Recommendations / techniques for the evaluation

Ensure that these arrangements are described in a project document (for example the Software Quality Plan) or that a project document makes reference to the procedures to be applied. Verify by means of both reports and interviews that they are being applied.

Lowest arrangements are acceptable when the project team is very limited (1 or 2 people).

Acceptance refusal criteria

Refusal :

- no arrangement has been laid down although the development team comprises more than two people.

SUBJECT : SAFETY SUPERVISION AND MANAGEMENT	**Requirement n° 2.4**

Requirement

Safety supervision should be a permanent activity while the software is being produced.

Aim of the evaluation

The evaluation must be limited to project control : have all the foreseen activities taken place ?

Preferential evaluation phase and supports necessary for the evaluation

Evaluation review over the course of the project.

Recommendations / techniques for the evaluation

The aim of this evaluation is neither to determine whether the project is profitable nor whether it has fallen behind the development schedule.

On the other hand, the evaluator must ensure that all the activities foreseen have been carried out and, in particular, the final phases of the life cycle (tests).

Acceptance / refusal criteria

Refusal :

- missing or shortened phase.

SUBJECT : DOCUMENTATION MANAGEMENT	**Requirement n° 2.5**

Requirement

The list of documents to be produced should be defined at the outset of the project (e.g in a Software Quality Plan)

Aim of the evaluation

To ensure that this list exists and it is complete.

Preferential evaluation phase and supports necessary for the evaluation

First evaluation review.

Recommendations / techniques for the evaluation

This list of documents serves as the guide to becoming aware of the project documentation during the evaluation.

It is desirable to obtain it at the start of the evaluation, for example in the Software Quality Plan, and to employ it as a support to identify documents that already actually exist.

This list will be the basis to verify :

- that all the foreseen documentation exist,
- the references of the documents,
- the revision indexes.

Acceptance / refusal criteria

Refusal :

- no list.

SUBJECT : DOCUMENTATION MANAGEMENT	**Requirement n° 2.6**

Requirement

Each document should at least:

- **be identified in a unique way (reference, version, revision index),**
- **be dated,**
- **carry a title that indicates the scope of its content and that sets the document in the context of the documentation as a whole. (specification, design, etc.),**
- **be written in the language mutually agreed by the applicant and the analyst.**

Furthermore, any subsequent changes to the documents should follow established guidelines (management of revision indices, etc.), and all documents should be available in their definitive version when the final software evaluation is undertaken by the analyst.

Aim of the evaluation

To ensure that :

- that the documents are subject to configuration management
- that all the documents likely to be consulted are in the agreed language(s)
- control of changes to the documents
- at the final evaluation, the real state of the version submitted for evaluation and, in particular, of all the documents.

Preferential evaluation phase and supports necessary for the evaluation

All the development phases and all the documents useful for the evaluation are concerned.

At the final evaluation review, all project documentation can be consulted.

Recommendations / techniques for the evaluation

Note the identification of each document presented :

- reference,
- version,
- date/revision index,
- title.

The status of the documents must be clearly established : rough, internal re-reading, applicable. This status can, for example, appear on the documents themselves, or a " master " list stating the version of applicable documents can be established.

Remark : there is no obligation to have a " paper " version of all the documents available systematically.

The analyst can accept all the languages that he is able to evaluate (analysis of the documentation and interviews).
Care should be taken with pre-existing software products, particularly those purchased

commercially, which could include documents in languages not falling within the initial agreement. It is necessary to translate a minimum of the documentation.

If the agreement on the languages employed is not respected, the translation is the responsibility of the designer.

This evaluation is conducted by surveying, whenever required, the documents presented to verify the other requirements.

Trace a document modification :

- date of the modification request,
- nature of the correction carried out,
- date of the document later than the modification request date.

Over the course of the evaluation, the evaluator must pay attention to all the elements presented, and ensure that they are for the latest version. This work is facilitated when the designer has developed a directory of documents or a file listing the documents for the version to be evaluated.

During the final evaluation, all the documents must be terminated and finalised. In the case of specific difficulties, a waiver is possible by issuing a decision " with reservations ", the reservations covering the document(s) to be sent to the analyst at a date set to finish the evaluation.

Acceptance / refusal criteria

Refusal :

- several sets of documents present on site : sets differing in content and not in their identification,
- documents with no identification or no version,
- the documents or proof of activity are in a language other than that (those) foreseen,
- non respect of the rules for changing documents,
- absence of several documents or documents in a state too far removed from a final version at the time of the final evaluation.

SUBJECT : DOCUMENTATION MANAGEMENT	Requirement n° 2.7

Requirement

The necessary documentation should be established at each phase of the lifecycle to facilitate verification and validation, and the software safety requirements should be traceable and capable of being verified at each stage of the process (traceability matrix for each definition document).

Aim of the evaluation

This evaluation ensures that the documentation foreseen has been established and allows for the checking of the traceability of the safety requirements in the different definition documents.

Preferential evaluation phase and supports necessary for the evaluation

Intermediate reviews. All the project documents can be consulted.

Recommendations / techniques for the evaluation

This evaluation can only take place when the analyst participates in evaluations over the course of the development.

The review reports allow determination of the documents actually available at each review.

All the documentation must exist and be verified by the designer before presentation for evaluation (except in the case of a specific agreement for the intermediate reviews : documents in provisional versions, etc.).

It is necessary to check the presence and the completeness of the traceability matrix in the documents established for each software development phase.

Acceptance / refusal criteria

Refusal:

- document absent not allowing for evaluation
- traceability of the non-demonstrated requirements at each software development phase.

SUBJECT : CONFIGURATION AND ARCHIVING MANAGEMENT	**Requirement n° 2.8**

Requirement

A procedure for configuration management and modifications management should be defined and documented. This procedure should, as a minimum, include the following items:

- **articles managed by the configuration, at least :**
  - **software specification,**
  - **preliminary and detailed software design,**
  - **source code modules,**
  - **plans, procedures and results of the validation tests.**
- **identification rules (of a source module, of a software version, etc.),**
- **treatment of modifications (recording of requests, etc.).**

For each article of configuration, it is necessary to be able to identify any changes that may have occurred and the versions of any associated elements.

Aim of the evaluation

This evaluation :

- ensures that a configuration management system has been established,
- ensures that the configuration reference system is adequate,
- contributes to estimating the control of the configuration management system.

Preferential evaluation phase and supports necessary for the evaluation

Final evaluation review. All articles managed in the configuration system and their history can be consulted. Configuration management procedure. Articles subject to configuration management.

Recommendations / techniques for the evaluation

The identification of the configuration management procedure (name, date, edition, etc.) and its scope of application must be noted.
Remark : how efficiently the procedure is being applied is evaluated through the other requirements.

It must be ensured that all these articles are indeed managed by the configuration management system :

- control of modifications since the preceding configuration,
- coherence of articles of the same reference system.

The specification, the design, the source and the test results must correspond to the same product state (identical version).

The supplier must be capable of tracing these changes for each of the articles either by means of the tool used or by using the manual management procedures foreseen.

Acceptance / refusal criteria

Refusal :

- absence of procedure,
- unrecorded request for modification (quoted by the developers but not recorded),
- configuration article missing with respect to the minimum required,
- incoherence between the articles subject to configuration management for the software version undergoing evaluation,
- identification not allowing the changes of version to be traced,
- modifications made without changing the identification after entry into the configuration management system.

SUBJECT : CONFIGURATION AND ARCHIVING MANAGEMENT	**Requirement n° 2.9**

Requirement

Software configuration management should allow a precise and unique software version identification to be obtained. Configuration management should associate all the articles (and their version) making up a software version .

Aim of the evaluation

This evaluation ensures the control of a software version.

Preferential evaluation phase and supports necessary for the evaluation

Final evaluation review. All the configuration articles can be consulted.

Recommendations / techniques for the evaluation

The version of the software is the identification known by the configuration management system. It can be different from the " brand " name of a customer version. In this case, the explicit link must be provided.

Remark : the evaluation report should always state the version of the software undergoing evaluation to avoid any ambiguity.

There in no obligation to use a tool. For a small project, it is possible to employ " minimalist " management :

- file identification and location rules (name of directory) ,
- " master " list providing the files falling within a configuration (date/version, etc.),
- manual links with the documentation.

Acceptance / refusal criteria

Refusal :

- no configuration management established, making accurate identification of the software version undergoing evaluation impossible.

SUBJECT : CONFIGURATION AND ARCHIVING MANAGEMENT	**Requirement n° 2.10**

Requirement

All articles in the software configuration should be covered by the configuration management procedure before being tested or being requested by the analyst for final software version evaluation.

Aim of the evaluation

This evaluation ensures that the developer has finalised the reference system before the start of the evaluation.

Preferential evaluation phase and supports necessary for the evaluation

Final evaluation review. All the project documents can be used.

Recommendations / techniques for the evaluation

Particular attention should be paid to these points if several presentations have been necessary:

- verify that the date of entering the configuration precedes the start of the evaluation.

- verify that the reference system presented is coherent (all the articles must correspond to the same software version).

Acceptance / refusal criteria

Refusal :

- article not managed by the configuration system although forming part of the minimum necessary for the evaluation.

SUBJECT : CONFIGURATION AND ARCHIVING MANAGEMENT	**Requirement n° 2.11**

Requirement

Procedures for the archiving of software and its associated data should be established (methods for storing backups and archives).

Aim of the evaluation

This evaluation ensures the control of back-ups and archives.

Preferential evaluation phase and supports necessary for the evaluation

Final evaluation review. All the configuration articles can be consulted.

Recommendations / techniques for the evaluation

- verify the existence and application of a back-up procedure. This procedure can have recourse to central facilities or much more basic facilities (simple floppy disk correctly identified and protected).
- it may be advantageous to verify the efficiency of the procedures employed to request restoration of the last version from the back-ups/archives.

Acceptance / refusal criteria

Refusal :

- absence of software product back-ups.

SUBJECT : SOFTWARE MODIFICATIONS MANAGEMENT	Requirement n° 2.12

Requirement

Any software modification is subject to the rules established for modification and configuration management, and requires that the development process be recommenced at the highest "upstream" point needed to take the modification into account.

Aim of the evaluation

This evaluation ensures that modifications are controlled.

Preferential evaluation phase and supports necessary for the evaluation

Final evaluation review. All requests for modifications and the project documents must be accessible.

Recommendations / techniques for the evaluation

It is advised to proceed by analysing the traceability of modifications. Take a few modifications and follow their treatment with a view to answering the following questions:

- has the modification procedure been applied ?
- have the modifications been carried out ?
- have the documents been updated ?
- have the earlier development phases been carried out again (unit tests, etc.) ?

In the case of a software product being presented several times for evaluation (failure of a previous evaluation or new evaluation following changes to a software product having previously been granted a favourable decision), particular attention should be paid by the evaluator to modifications stemming from the deviations noted during a previous evaluation.

Remark : with this in mind, all the evaluation reports have to be archived to be able to use them at a later date (reconstructing evaluation histories, list of observations, etc.).

Acceptance / refusal criteria

Refusal :

- earlier activities not carried out again (documents not updated, absence of non regression tests without justification, etc.).

SUBJECT : SOFTWARE MODIFICATIONS MANAGEMENT	Requirement n° 2.13

Requirement

The description of software modifications should include details of each modification made. This should include at least the following items for each modification:

- **the modification request,**
- **the report detailing the analysis of the impact of the software modification, the decisions made in this respect and their justification,**
- **the version of the software to be modified as well as the configuration articles and their version.**

Aim of the evaluation.

The aim of this evaluation is to ensure control of the modifications management process.

Preferential evaluation phase and supports necessary for the evaluation

Final evaluation review. Documents used : requests for modifications and associated analyses, all project documents.

Recommendations / techniques for the evaluation

proceed by tracing a modification :

- request for modification,
- impact analysis (documentation, code, tests, etc.),
- modification made (source/documentation), tests, link with the configuration management system.

In the case of a software product being presented several times for evaluation (failure of a previous evaluation or new evaluation following changes to a software product having previously been granted a favourable decision), particular attention should be paid by the evaluator to modifications stemming from deviations noted during a previous evaluation.

Remark : with this in mind, all the evaluation reports have to be archived in order to be able to use them at a later date (reconstructing evaluation histories, list of observations, etc.).

Acceptance / refusal criteria

Refusal :

- modifications carried out not in the modification request circuit,
- the modifications carried out since the preceding evaluation presentation were not traced (which implies carrying out the entire evaluation again).

SUBJECT : DEVELOPMENT TOOLS	**Requirement n° 2.14**

Requirement

Optimisation of object code performance options are forbidden.

Aim of the evaluation

To ensure the coherence between coding and the code generated and the non-introduction ~ during the generation of the executable code ~ of instructions contrary to safety or leading to dysfunction.

Preferential evaluation phase and supports necessary for the evaluation

Final evaluation review. The compilation / construction files of the executable code are necessary.

Recommendations / techniques for the evaluation

The evaluator verifies whether compilation options have been used and that optimisation of object code performances options have not been used.

Acceptance / refusal criteria

Refusal :

- use of optimisation of object code performances options.

SUBJECT : DEVELOPMENT TOOLS	**Requirement n° 2.15**

Requirement

If a new compiler or a new linker is used during the development procedure, the validity of the testing activities already performed should be analysed by the designer.

Aim of the evaluation

To ensure the validity of the verifications carried out on the software when the development tools have been modified.

Preferential evaluation phase and supports necessary for the evaluation

Final evaluation review. The versions of the tools and the utilisation options must be communicated by the designer.

Recommendations / techniques for the evaluation

The introduction of a new compiler (or linker) during test activities is always inadvisable and, in general, is a case of force major (correction of a bug by the constructor of the compiler).

To evaluate this requirement, the tools and options used are identified, and by analysing the command files (compilation, linkage) it is ensured that they have not been modified over the course of the development. Corroborate the information through interviews with the developers.

In the case of changes (tool or version), an impact analysis must be conducted if all the tests are not carried out again.

Acceptance / refusal criteria

Refusal :

- changing a software tool or utilisation option during development without a precise impact study.

SUBJECT : DEVELOPMENT TOOLS	**Requirement n° 2.16**

Requirement

Tools used during the development procedure (compiler, linker, tests, etc.) should be identified (name, reference, version, etc.) in the documentation associated with the software version (e.g. in the Version Sheet).

Aim of the evaluation

To ensure that the development environment has been defined.

Preferential evaluation phase and supports necessary for the evaluation

Final evaluation review.

Recommendations / techniques for the evaluation

- verify (for example in the Version Sheet) that the development tools have been referenced in the documentation associated with the software.
- then verify (for example in the Version Sheet) that the references indeed correspond to the tools used. If the tools have been modified over the course of the development, refer to requirement 2.15.

Acceptance / refusal criteria

Refusal :

- absence of references of the tools used,
- tool references incorrect.

SUBJECT : EXTERNAL SUB-CONTRACTING	**Requirement n° 2.17**

Requirement

In the event that any part (even partially) of the software development is subcontracted to a third party, the present requirements should also apply to the subcontractor. They may possibly be adapted to reflect the importance and nature of the subcontracted tasks.

Aim of the evaluation

To ensure that software products not directly developed by the designer respond to the same quality criteria.

Preferential evaluation phase and supports necessary for the evaluation

First evaluation review. The contract signed with the sub-contractor (only the technical part) is the principal supporting document.

Recommendations / techniques for the evaluation

- verify that the quality requirements are laid down in the contract signed with the sub-contractor, possibly only in part depending on the work sub-contracted.
- internal sub-contracting is not concerned by this requirement as it is assimilated with the designer from the point of view of the quality and safety principles applied.
- if there is no mention of the quality and safety requirements in the sub-contract (forgetting of the designer or late knowledge of the requirements for the contract), the evaluator should pay closer attention to the sub-contracted parts.
- over the course of the evaluation, the evaluator treats the sub-contracted parts no differently from the others, bearing in mind that the entire software product must satisfy the software quality requirements.

Refusal :

- requirements not respected for the sub-contracted software product (Cf. requirement criteria concerned to formulate the final decision).

SUBJECT : EXTERNAL SUB-CONTRACTING	Requirement n° 2.18

Requirement

The designer should ensure and demonstrate that the requirements have been respected by the subcontractor(s).

Aim of the evaluation

To ensure that designers have undertaken activities to verify the respect of the quality requirements by their external sub-contractors.

Preferential evaluation phase and supports necessary for the evaluation

End of evaluation review. The evaluation supports are made up of the reports of reviews, meetings and audits at the premises of the sub-contractor.

Recommendations / techniques for the evaluation

- on the basis of the reports presented, verify that the designer has ensured respect of the quality and safety requirements.
- when deviations come to light, also go into the way the designer ensures that they are resolved. Determine the unresolved deviations.
- carry out a number of verifications directly on the sub-contracted software product, modulating the effort in accordance with the extent of the verifications carried out by the designer, the problems observed by the designer, the complexity of the software, etc.

Remark : the absence of requirement respect checks by the designer does not, a priori, justify a negative decision on the sub-contracted software. The sub-contractor may well indeed have respected the requirements by applying satisfactory quality and safety principles. The objective encompasses the final software product, the requirements being a mean of minimising software faults.

In the case where the designer has carried out no verification on the sub-contracted software, the evaluator should pay very close attention to the sub-contracted software during the evaluation.

Acceptance / refusal criteria

Refusal :

- requirements not respected on the sub-contracted software product

SUBJECT : EXECUTABLE CODE PRODUCTION	Requirement n° 2.19

Requirement

Any option or change in the generation, during the software production should be recorded (e.g. in the Version Sheet) so that it is possible to say how and when the software was generated.

Aim of the evaluation

This evaluation ensures control of the software production environment and the relevance of all the verifications carried out.

Preferential evaluation phase and supports necessary for the evaluation

Final evaluation review.

Recommendations / techniques for the evaluation

Over the course of interviews, the following questions may be asked :

- have the tools used (compiler, assembler, link editor, etc.) to create a product been archived ?
- have the generation procedures (which files, which tools with which options) been archived ?
- has the software been recompiled several times over the course of the development depending on the test phase ? (example : use of an instrumentation-based code in unit tests). Verify the validity of the verifications carried out in this case.
- if the development tools are still available, the evaluator will have a better idea by requesting the version to be regenerated from the sources.

Acceptance / refusal criteria

Refusal :

- absence of the necessary development environmental control allowing it to be ensured that the software product verification was appropriate.

SUBJECT : SOFTWARE INSTALLATION AND EXPLOITATION	**Requirement n° 2.20**

Requirement

All failures linked to safety and dependability functions brought to the attention of the designer of the system should be recorded and analysed.

Aim of the evaluation

This evaluation ensures that the failures linked to safety functions brought to the attention of the designer are dealt with.

Preferential evaluation phase and supports necessary for the evaluation

Final evaluation review. The following documents can be used : anomaly reports, mail from users, customer claims, etc.

Recommendations / techniques for the evaluation

- identify, by looking through anomaly reports, mail from users, etc. the failures linked to safety functions.
- evaluate these possible failures with respect to :
    - corrective actions to the software product,
    - corrective actions of the quality system (quality system corrective action established to detect any other anomaly of the same nature),
    - feedback (enriching of a list of feared events, etc.).
- ensure that the anomaly has been corrected or that the lack of correction has been justified by means of an analysis.

Acceptance / refusal criteria

Refusal :

- unjustified existence of an uncorrected failure having an impact on system safety.

3.4 REQUIREMENTS FOR SOFTWARE VERIFICATION AND VALIDATION

3.4.1 Presentation

The verification and validation activities are intended to demonstrate that the software products stemming from a phase of a development cycle are in conformity both with the specifications established during the earlier phases and with the applicable rules and standards.

They are also intended to detect and deal with errors that may have been introduced over the course of the software development.

The software verification and validation requirements concern :

- Software Verification :

  . reviews
  . code verification

- Software Tests :

  . Validation Tests
  . Integration Tests
  . Module Tests.

3.4.2 Requirements to be evaluated relative to the software verification

The corresponding requirement evaluation guidance files are presented in the following section.

SUBJECT : GENERAL VERIFICATION AND VALIDATION REQUIREMENTS	**Requirement n° 3.1**

Requirement

The analyst should be able to carry out the evaluation of software conformity to the present requirements by conducting any audits or expertises deemed useful during the different software development phases.

All technical aspects of software lifecycle processes are subject to evaluation by the analyst.

The analyst must be allowed to consult all verification reports (tests, analyses, etc.) and all technical documents used during software development.

Aim of the evaluation

This requirement is intended to guarantee the analyst the possibility of conducting audits or expertises to verify the respect of the software quality and safety requirements and the access to all the technical information necessary for the evaluation.

The analyst can have recourse, during the evaluation, to consult any element of proof concerning the activity undertaken by the designer.

Preferential evaluation phase and supports necessary for the evaluation

All the development phases and activities are likely to be audited, expertised or evaluated.

The evaluation is carried out at the end of a phase (example : end of design, end of integration test, and so on) on the basis of all the reports available.

Recommendations / techniques for the evaluation

This requirement is to be employed when particular difficulties arise in verifying certain requirements (activities of the designer unclear for example) or when doubts exist on the reality of certain activities of the designer : in this case, go into more depth on one or several subjects by an audit (half to one day maximum).

It is recommended to group audit/ expertise subjects to minimise the number of intervention visits to the premises of the designer.

These audits can also be the opportunity to prevent evaluation failure by proposing corrective measures to the designer at an early enough stage in the development process. This activity must not, however, lead to a transfer of responsibility to the analyst.

The evaluation must be strictly limited to the technical aspects of the software product to be evaluated and only to the software. In particular, any considerations on the general organisation of the firm (ISO 9000 quality system, for example), and human or financial

aspects are out of bounds.

In the case of difficulties in relations with the designer, the principles of the independence, no competition, and objectivity of the analyst can reminded.

The evaluator should not hesitate to question the designer on existing elements in all their forms (computer files, folders, data sheets, etc.).

Hand-written documents are acceptable provided they are clear and identified according to the principles of configuration management. It is important to be able to determine unambiguously which software version the activity has taken place on. If there is any doubt, evaluators themselves can, by means of sampling, conduct certain verifications.

Difficulties can arise in the case of distribution of the software development between several industrialists, between subsidiaries of the same group, and between several establishments of the same firm. The case of external sub-contracting and the use of pre-existing commercially available software also create difficulties of access to documents.

A presentation of the industrial framework of the software development by the designer will ensure the evaluator access to all the documents.

The evaluation can, if required, be conducted in the premises of a sub-contractor or in a subsidiary having developed the software if this facilitates access to all the documents. Access via computer link (company network for example) to certain information is also acceptable.

The designer is not required to make paper copies of all the documents available, and a computer consultation is acceptable with the possibility of printed copy on request. In this case, the designer must definitely provide any assistance necessary to the evaluator for the electronic consultation.

Acceptance / refusal criteria

Refusal :

- the designer refuses the intervention of the analyst for the audit or expertise on the software development.
- restriction of access of the designer to purely technical information (on the grounds of industrial confidentiality). These restrictions bring he feasibility of the evaluation into question.
- absence of proof of verification on the final version of the software.
- the impossibility to access certain documents is grounds for stopping the evaluation.

SUBJECT : GENERAL VERIFICATION AND VALIDATION REQUIREMENTS	**Requirement n° 3.2**

Requirement

Evaluation of software conformity to the present requirements is performed for a specific, referenced software version. Any modification of previously evaluated software which has received a final opinion from the analyst should be pointed out to the latter in order that any additional evaluation activities can be carried out to update this opinion .

Aim of the evaluation

The analyst must ensure that he or she evaluates a precise and clearly identified version of the software.

Preferential evaluation phase and supports necessary for the evaluation

This requirement is primordial at the final evaluation. All the configuration management documents are concerned.

Recommendations / techniques for the evaluation

When the evaluation takes place in several stages, it should be ensured that the modifications carried out since do not bring previous conclusions into question. If they do, the evaluation must be gone through again, centring on the modifications made.

When a decision has already been formulated (the case of the evaluation of a new version of a product), the deviations between the two versions must be accurately identified otherwise the entire evaluation may have to be gone through again. The analyst can modulate the additional evaluation in accordance with the extent of the modification (example : a modification of a few observations or a few lines can be dealt with by sending justifying documents defined by the analyst with no new on-site evaluation).

Acceptance / refusal criteria

Refusal :

- impossible to identify the precise content of the software product (software modules with no version, etc.),
- modifications introduced into a software product that has already been evaluated and distributed to end users without informing the analyst.

SUBJECT : GENERAL VERIFICATION REQUIREMENTS	Requirement n° 3.3

Requirement

A verification report should be produced for each verification activity, and should identify and document all distortions (non-conformities) with respect to:

- **the corresponding specifications,**
- **rules or standards (design, coding),**
- **any quality assurance procedures that may exist.**

Aim of the evaluation

To verify that the verification activities carried out have indeed been formalised.

Preferential evaluation phase and supports necessary for the evaluation

Final evaluation review. All elements of proof of tests are necessary.

Recommendations / techniques for the evaluation

- verify the existence of records for all the verification activities
- these elements of proof can take many forms (hand-written files, folders/data sheets, computer files), as long as they provide the object of the verification, the results and the exact state of the software verified.

Acceptance / refusal criteria

Refusal :

- no proof that the foreseen verifications have been carried out.
- absence of the results of these verifications (correct or incorrect).

SUBJECT : REVIEWS REQUIREMENTS	Requirement n° 3.4

Requirement

An external specification review (with the analyst) should be held at the end of the software specification phase.

Activities involving analysis and software specification verification should:

- **verify the exhaustiveness and adequacy of the software specifications with respect to the system specifications,**
- **verify the traceability with respect to the system specifications.**

Aim of the evaluation

This evaluation ensures that this review has indeed taken place and that the objectives of the review have been achieved and that the software specifications have been verified.

Preferential evaluation phase and supports necessary for the evaluation

The evaluation can take place at the end of the specification phase. The basis of the evaluation is the review report and the system specification documents (or contractual specification) and software specifications are necessary.

Recommendations / techniques for the evaluation

- examination of the review report (versions of documents, participants, etc.).
- comparison of the date of the review with the project planning.
- examination by sampling of how the decisions of the review have been taken into account.
- analyse the results of the verifications carried out (reports, traceability matrix).
- verify by sampling :
  . the coherence between the system and software documents,
  . the traceability between the two documents.
- verify the inherent quality of the software specifications (completeness, internal coherence, etc.).

General comments on evaluating all reviews :

- the requirement covers the existence of a review. There is no requirement for a review to be carried out for each new version of the documents. The review must, however, cover a version that is sufficiently representative of the final version.
- grouping of several reviews (example : specification and design) is acceptable if the aim of each of the review has been achieved.

Acceptance / refusal criteria

Refusal :

- absence of a specification review.
- significant incoherence between the documents (system functions not foreseen in the software, etc.).
- significant incoherence in or incompleteness of the software specifications (no description of the hardware interfaces for example)
- absence of “software specification / hardware specification” traceability matrix.

SUBJECT : REVIEWS REQUIREMENTS	Requirement n° 3.5

Requirement

Analysis activities and software design verification should verify the conformity to specifications.

Aim of the evaluation

To ensure that verifications have been carried out by the designer and that there is a coherence between the design and specifications of the software.

Preferential evaluation phase and supports necessary for the evaluation

End of detailed design. The documents necessary are the specifications and the design, together with all the reports of the verifications carried out by the designer.

Recommendations / techniques for the evaluation

- verify the existence of proof of verifications and note the versions of the elements that the verifications covered (are they indeed for the last versions ? ; if not, have additional verifications been carried out after the modifications ? Check the dates of documents as an indication.
- by sampling, verify the coherence between documents : go in both directions (from the specifications and from the design). Take several different functions.
- the absence of proof of verification must push the evaluator to step up the sampling effort.

Acceptance / refusal criteria

Refusal:

- significant incoherence between the software design and specifications noted by the evaluator
- absence of “design / specification” traceability matrix.

SUBJECT : REVIEWS REQUIREMENTS	**Requirement n° 3.6**

Requirement

An external validation review (with the analyst) should be held at the end of the validation phase.

Aim of the evaluation

This evaluation ensures that this review has indeed taken place and that the objectives of the review have been achieved.

Preferential evaluation phase and supports necessary for the evaluation

Final evaluation review. The evaluation centres on the review report.

Recommendations / techniques for the evaluation

- examination of the review report.
- comparison of the date of the review and the project schedule.
- examination by sampling of how the decisions of the review have been taken into account.

(Cf. general remarks : requirement 3.4).

Acceptance / refusal criteria

Refusal :

- absence validation review.

SUBJECT : REVIEWS REQUIREMENTS	**Requirement n° 3.7**

Requirement

The result of each review should be documented and archived. It should include a list of all actions decided on in the review process, and the review conclusion (decision on whether or not to move on to the next activity). The activities defined in the review should be monitored and treated.

Aim of the evaluation

The evaluation ensures that the actions decided at the review have been recorded and can be monitored and that all the actions decided at a review have indeed been taken into account.

Preferential evaluation phase and supports necessary for the evaluation

Final evaluation review. All the review reports and all the project documents are necessary.

Recommendations / techniques for the evaluation

- verify the dates of the reviews with respect to the development plan.
- verify that each report accurately identifies :
  - . the actions to be undertaken,
  - . who is responsible for these actions,
  - . the deadlines for these actions.
- examination by sampling of how the decisions of the reviews on the final software version have been dealt with. Unresolved decisions must be justified.
- possibly have recourse to the monitoring facilities of the designer (recapitulative table, etc.).

Acceptance / refusal criteria

Refusal :

- unjustified absence of proof that reviews have been conducted.
- non respect of a major action decided at review.

SUBJECT : CODE VERIFICATION (SOURCE CODE AND DATA)	**Requirement n° 3.8**

Requirement

Code verification (static analysis) should ensure that the code conforms to :

- **the software design documents,**
- **coding rules.**

Aim of the evaluation

To ensure the coherence between the detailed design and the source code.

Preferential evaluation phase and supports necessary for the evaluation

Unit test phase or final evaluation review. Use the detailed design, source code and corresponding verification documents (review, inspection reports, etc.) to conduct the evaluation.

Recommendations / techniques for the evaluation

- verify the coherence between code and design by sampling,
- study the results of the verifications carried out by the designer if they exist, and ensure that the actions or points raised by the documents have been followed up and dealt with.

Remark : the design can be included in the source files. In this case, ensure that the detailed design and the source comments are clearly distinguishable.

Acceptance / refusal criteria

Refusal :

- multiple cases of incoherence between the detailed design and the code,
- absence of design documents.

SUBJECT : GENERAL VALIDATION REQUIREMENTS	Requirement n° 3.9

Requirement

The software verification strategy used at the different software development steps and the techniques and tools used for this verification should be described in a Test Plan before being used. This description should, as a minimum, include:

- **identification of the software and its safety-related components that will be submitted to validation procedure before use,**
- **organisation of the verification activities (integration, validation, etc.) and any interfaces with other development activities,**
- **independence of the verification (if applicable): the verification strategy should be developed and implemented, and the test results should be evaluated independently (by an individual, department, or organisation) in relation to the size of the development team,**
- **verification methods and tools used (types of tests, etc.),**
- **environment of the verification (test equipment, emulators, etc.),**
- **manner in which test results were verified,**
- **a traceability matrix demonstrating the correspondance between the tests to be undertaken and the objectives of the tests defined.**

Aim of the evaluation

To ensure that a verification strategy exists and that it is described in a project document. These descriptions allow the evaluator to verify at a later date that they are being respected.

To ensure that independence has been planned for the conducting of tests.

Preferential evaluation phase and supports necessary for the evaluation

First evaluation review, the strategy must be described at the start of the project. The evaluator employs the document(s) describing this strategy as the basis.

First evaluation phase for the arrangements foreseen. Final review to verify that the arrangements have been applied.

Recommendations / techniques for the evaluation

The assessor must verify that :

- a strategy has been specified ;
- the strategy really exists : the tests are indeed structured in sub-sets with clear objectives ;
- it is coherent with the pre defined objectives and the desired test coverage;
- it is well integrated into the general verification strategy ;
- verifications on models are only complementary ;
- the parts of the software subject to evaluation are indeed validated before being put into service or delivered to end users ;
- all the verification phases required have been foreseen ;
- the scheduling of stages is coherent (for example, if the test of a function relies on data provided by another function or on the behaviour of another function, this other function

must be tested first) ;
- the responsibilities for verification have been specified (previously when developing the tests or simply recorded in the test reports) ;
- as a minimum, they respect requirement of independance ;
- the facilities have been specified and traced ;
- they are coherent with the envisaged strategy and methods, and that the whole (strategy / method / means) represents a feasible testing approach ;
- the tests are identified ;
- the tracing system between the objectives and the tests has been established (matrix) ;
- the foreseen tests conform to the envisaged strategy and indeed cover the objectives of the specified tests ; for this assessment, he should have recourse to the specified objectives and to the documents serving as the reference, in general being the software specifications ;
- verify that the verifications will be formalised, as will the results, and that these results will be verified manually (the case of test tools with automatic result generation facilities).

At the first evaluation review, the evaluator requests the designer to complete the arrangements if they are inadequate. At the final stage, the evaluation will be based on the results of the verifications (do they exist for all the phases ? are they correct ?, etc.).

Verify that the verification strategy foresees this independence of the verification, and obtain explanations of its practical application (who carries out design/coding, who tests ? ...).

Proof that this independence is being applied must be provided (example : name or initials of the members of the team in the source listing document sand in the test procedures, etc.).

When the team is reduced to only one person, the application of independence is more delicate, because of requesting an external intervention in the project. The absence of independence in this case can be tolerated : verify thoroughly that the development has indeed been carried out by only one person (no sub-contractor, etc.).

Acceptance / refusal criteria

Refusal :

- no verification planning or description of the means of verification.
- independence not applied although the team is not reduced to only one person.

SUBJECT : GENERAL VALIDATION REQUIREMENTS	Requirement n° 3.10

Requirement

Verification of a new software version should include non-regression tests.

Aim of the evaluation

To ensure that non-regression tests exist and that the results of their execution are correct when several versions of a software product have been developed.

Preferential evaluation phase and supports necessary for the evaluation

First evaluation review of a new version or final evaluation review. The test documents (unit, integration, validation) are the basis of the evaluation, supplemented by documents allowing identification of the modifications.

Recommendations / techniques for the evaluation

For the unit tests, non-regression consists in carrying out all the module test cases again (intervention on a source module can introduce an error at any point in the module).

For the other test phases (integration, validation), the important point is more the determination than the existence of non-regression tests. : does an impact study exist ? on what basis have the non -regression tests been selected ?, etc.

It should be borne in mind that the requirement does not impose carrying out all the existing tests, but the designer must be able to justify the sub-set done again with respect to the extent of the modifications carried out.

Acceptance / refusal criteria

This requirement does not apply to an initial development of the version that must include the entire range of tests.

Refusal :

- the non-regression validation tests do not cover all the functions modified or affected by the modifications carried out.

SUBJECT : GENERAL VALIDATION REQUIREMENTS	**Requirement n° 3.11**

Requirement

Directives for drawing up test procedures should include :

- **a description of the input data to be used (*value*),**
- **a description of the expected output (*value),***
- **criteria on which test results will be judged acceptable (*tolerance*).**

Aim of the evaluation

To ensure that the written arrangements have been planned in such a way that the tests are correctly formalised.

Preferential evaluation phase and supports necessary for the evaluation

First evaluation review. The documents planning the test activities are necessary (test plan or quality plan, etc.).

Recommendations / techniques for the evaluation

Verify that the instructions for drafting a test procedure require the description of the inputs, outputs and the results acceptance criteria. If the tests are already in progress, the evaluator will get a better idea by examining examples for each of the test phases.

Acceptance / refusal criteria

Refusal :

- no description of inputs and outputs has been foreseen,
- the test results acceptance criteria have not been described for aspects of the performance or complex algorithmic calculations.

SUBJECT : GENERAL VALIDATION REQUIREMENTS	Requirement n° 3.12

Requirement

The tests formalised in reports should be able to be carried out again (e.g., in the presence of the analyst).

Aim of the evaluation

To ensure that the tests presented by the designer both exist and supply the expected results.

Preferential evaluation phase and supports necessary for the evaluation

End of a test phase or final evaluation review. The test facilities must still be available. The evaluation has recourse to test procedures and results.

Recommendations / techniques for the evaluation

It is preferable that the evaluator warns the designer in advance (before the evaluation or at the start of the evaluation) of his or her intention to conduct certain tests again so that the argument of delays in setting up the test facilities is not put forward.

The evaluator takes samples (a few tests for each stage : modular, integration, validation) and verifies the results obtained.

Indirectly, the evaluator can verify the principles of configuration management applied by the designer (can the software version and the corresponding tests be found again ?, have all the files necessary for the test indeed been archived ?, etc.).

If the test facilities are no longer available, manually verify a number of tests.

Acceptance / refusal criteria

The total impossibility to conduct tests again is not grounds for evaluation refusal (for example, the facilities no longer exist), but must draw the attention of the evaluator to the reality of all the elements presented and not exclude a manual verification of a few test results.

SUBJECT : SOFTWARE SPECIFICATIONS VERIFICATION	**Requirement n° 3.13**

Requirement

The test coverage should be made explicit in a traceability matrix and respect the following requirements:

- **each element of the specification, including safety mechanisms, should be covered by a validation test,**
- **it should be possible to verify the real-time behaviour of the software in any operational mode.**

Furthermore, the validation should be carried out in conditions representative of the operational conditions of the system.

Aim of the evaluation

To ensure that the test coverage sufficient, that means the validation tests allow to verify :

- that all the functions foreseen in the software specifications behave as expected,
- that the validation carried out is meaningful to the real operating conditions of the system,
- that the constraints necessary for verification by tests have indeed been taken into account when designing the architecture of the software.

Preferential evaluation phase and supports necessary for the evaluation

This requirement can be dealt with as soon as the designer has set a validation strategy.

Recommendations / techniques for the evaluation

- analyse how the designer has ensured validation of all the functions (existence of a "validation tests / specification elements" traceability matrix).
- verify by sampling if the functions have indeed been validated (consider degraded modes and specific operating modes).

The representativeness of the validation is influenced by :

- the hardware: is it identical to the target system ? If not, have the differences been evaluated ?
- the software execution conditions : is the execution frequency identical ? are the input events alike ? are the interruptions identical ?
- the software : is it indeed the final version ? has the software been recompiled with different options ?
- the use : are the conditions of use those of the final system (behaviour of users, etc.) ? Particular attention should be paid to hardware tests not carried out on account of their destructive nature for the hardware (hardware fault tests by fault injection). On the other hand, the simulation of system failures is not always instumentable.

Moreover, the people in charge of the verification should:

- analyse the possible operating modes and verify that each of the modes will be capable of being tested (except if it leads to hardware destruction).
- verify that the foreseen test environment (software tools, simulator, etc.) has characteristics compatible with the foreseen software performance.
- at the final review, ensure that all the modes have indeed been verified as well as the transitions between modes if such transitions exist, that the foreseen test environment is indeed available, and that its characteristics are compatible with the verifications to be carried out.

Acceptance / refusal criteria

Refusal :

- absence of a “validation test/ specification element” traceability matrix.
- safety-related functions not validated.
- validation conditions unrepresentative of the final use of the system that could lead to unsafe behaviour in the real environment.
- existence of operating modes that cannot be verified, except if this involves degraded modes that cannot be checked without destroying the hardware.
- existence of transitions between non-tested modes.

SUBJECT : SOFTWARE SPECIFICATIONS VERIFICATION	Requirement n° 3.14

Requirement

Validation results should be recorded in a validation report that should cover at least the following points:

- **the versions of software and system that were validated,**
- **a description of the validation tests performed (inputs, outputs, testing procedures),**
- **the tools and equipments used to validate or evaluate the results,**
- **the results showing whether each validation test was a success or failure,**
- **a validation assessment: identified non-conformities, impact on safety, decision as to whether or not to accept the validation.**

A validation report should be made available for each delivered software version and should correspond to the final version of each delivered software product.

Aim of the evaluation

To verify the content of the validation report and, in particular, the description of the results and to ensure that the validation has indeed covered the final version of the software delivered to users of the system.

Preferential evaluation phase and supports necessary for the evaluation

End of validation phase or final evaluation review. Both the validation report and the source code are necessary.

Recommendations / techniques for the evaluation

- verify the content of the validation report :
  . has the software version been correctly identified ? ; is the software version covered by the validation known exactly ? ; have there been any modifications to the version during the validation and, if so, how has the validation been completed ?
  . has each test carried out in the validation been fully described ? The evaluator proceeds by conducting certain tests to ensure the thorough description of the inputs/outputs/results, the action to be taken, and the means of testing (version of tools used, etc.).
  . has the appraisal of each test been clearly stated ? ; have the problems detected been well identified (either by mention of a deviation or by reference to an anomaly file) ? ; how have problems encountered in validation been followed up ? ; do uncorrected anomalies still exist ?
- verify the coherence of versions between validation report and source software. Complete the evaluation by verifying the dates of the documents and the dates of the last modification of the source files.
- also have recourse to the documents formalising the modifications to verify that modifications were not introduced subsequent to the validation.

- examine the validation reports of the different versions if the last validation is partial (the case of modifications introduced) and then verify that all the functions have been validated in their final version.

Acceptance / refusal criteria

Refusal :

- absence of validation results allowing both the smooth running and scope of the validation carried out to be corroborated.
- functions not validated in the final version delivered to users,
- modifications introduced after the validation without non-regression verification.

SUBJECT : SOFTWARE DESIGN VERIFICATION	Requirement n° 3.15

Requirement

Software integration tests should be able to verify:

- **correct sequencing of the software execution,**
- **exchange of data between modules,**
- **respect of the performance criteria,**
- **non-alteration of global data.**

The test coverage should be given explicitly in a traceability matrix demonstrating the correspondence between the tests to be undertaken and the objectives of the tests defined.

Aim of the evaluation

To verify the content of the software integration tests.

Preferential evaluation phase and supports necessary for the evaluation

End of the integration phase or final evaluation review. The supports required for the analysis are the preliminary design documents and the integration test report.

Recommendations / techniques for the evaluation

- software sequencing : ensure that the module is correctly called and verified, that processing is carried out on coherent data (no intermediate acquisitions which could mean that all the modules do not process the same data), and that the outputs are achieved after processing and are coherent with respect to each other.
- verify that all the modules are called by means of the tests, and that the order and type of calling parameters has been verified (by rereading for example).
- verify that performance tests exist, these tests possibly being carried out during validation on the target system. Verify how the tests on the accuracy of an algorithm employing several modules are carried out.
- for the global variables, study how data loss is prevented (multiple access to the same data).
- check or carry out an “integration tests / coverage requirements” traceability matrix (e.g. of the following types: functionality of each module, interfaces between modules, performances, module input and output limits.

Acceptance / refusal criteria

Refusal :

- significant shortcomings in the content of the integration tests
- non-existence or impossibility of easily constructing a traceability matrix for tests undertaken.

SUBJECT : SOFTWARE DESIGN VERIFICATION	Requirement n° 3.16

Requirement

Any modification of the software during its integration should be analysed to identify the impact on the relevant modules and to ascertain whether certain verifications should be repeated.

Aim of the evaluation

To ensure the effective representatives of the integration tests.

Preferential evaluation phase and supports necessary for the evaluation

Final evaluation review. The supports required are the integration documents.

Recommendations / techniques for the evaluation

- analyse the integration reports and verify the dates and the versions of the documents to ensure that the integration covers the final version of the software,
- if modifications were introduced during the integration or at the end of integration, verify that an analysis has been carried out to identify any possible tests to be redone. If this analysis has not been formalised (which may likely be the case), carry out verifications by sampling : on a number of modifications, analyse the impact of the modification and the integration tests that should have been redone: verify that they have been.

Acceptance / refusal criteria

Refusal :

- modifications introduced subsequent to integration without redoing the integration activities or providing justification of no impact on integration.

SUBJECT : SOFTWARE DESIGN VERIFICATION	Requirement n° 3.17

Requirement

Integration test results should be recorded in a software integration test report, which should, as a minimum, contain the following points:

- **the version of the integrated software,**
- **a description of the tests performed (inputs, outputs, procedures),**
- **the integration tests results and their evaluation.**

Aim of the evaluation

To ensure that the integration tests have been conducted and that they are appropriate.

Preferential evaluation phase and supports necessary for the evaluation

End of the integration phase or final evaluation review. The evaluation is conducted primarily on the basis of the integration test reports.

Recommendations / techniques for the evaluation

- verify that the tests indeed correspond to the last version of the software ; if not, ensure that a minimum of integration tests are carried out again in accordance with the impact of each modification.

Acceptance / refusal criteria

Refusal :

- unjustified absence of certain integration tests.

SUBJECT : DETAILLED DESIGN VERIFICATION	Requirement n° 3.18

Requirement

Each software module should be submitted to a series of tests to verify, using input data, that the module fulfils the functions specified at the detailed design stage.

The test coverage should be given explicitly in a traceability matrix that demonstrates the correspondence between the tests to be undertaken and the objectives of the tests defined.

Aim of the evaluation

To ensure that the unit tests exist, and that they are appropriate to verify the functions laid down in the detailed design of each software module.

Preferential evaluation phase and supports necessary for the evaluation

End of the unit test phase or final evaluation review. Detailed design, code and documents related to the unit tests are necessary.

Recommendations / techniques for the evaluation

- for each module, verify that a unit test or a verification corresponding to the objective of a unit test exists;
- verify that all the functions of the detailed design are activated by the applied input data (logic functions, algorithmic calculations, etc.). Make use of the comments associated with the test procedures (if they exist);
- verify that the inputs applied indeed activate the foreseen function;
- check or carry out a "unitary test/ cover requirements" traceability matrix(e.g. of the following type: function of each module, execution path, module input and output limits.

Use these evaluations as the opportunity to carry out a parallel evaluation of the coherence between the detailed design and the code and the traceability between them .

Acceptance / refusal criteria

Refusal :

- modules not verified singly without justification,
- functions of the detailed design not activated by the unit tests without justification.
- non-existence or impossibility of easily constructing a traceability matrix for tests undertaken.

SUBJECT : DETAILLED DESIGN VERIFICATION	Requirement n° 3.19

Requirement

Module test results should be recorded in a report that contains at least the following points:

- **the version of the module tested,**
- **the input data used,**
- **expected and observed results,**
- **an evaluation of the results (positive or otherwise).**

Aim of the evaluation

To verify the level of formalisation of the module tests and the results of these tests.

Preferential evaluation phase and supports necessary for the evaluation

End of unit test phase or final evaluation review. The unit test documents are necessary.

Recommendations / techniques for the evaluation

The verifications are carried out by sampling (choose different functions, and possibly different developers if there are at least two involved in the project).

- verify that the tests have been conducted on the final version of the source module (by sampling).
- verify the presence of the results obtained and expected, and the conformity between them.
- examine how cases are dealt with when the results obtained are incorrect.

Acceptance / refusal criteria

Refusal :

- modules not tested in their final version,
- absence of results or unjustified incorrect results.

4 APPENDIX B : GENERAL PRINCIPLES FOR DETERMINING THE SOFTWARE REQUIREMENT LEVEL

4.1 PRESENTATION

This chapter presents the general principles for determining the software requirement level (1, 2) in function of the classification (with respect to the EN-954 and IEC 61508 Standards) of the safety related parts of the control system.

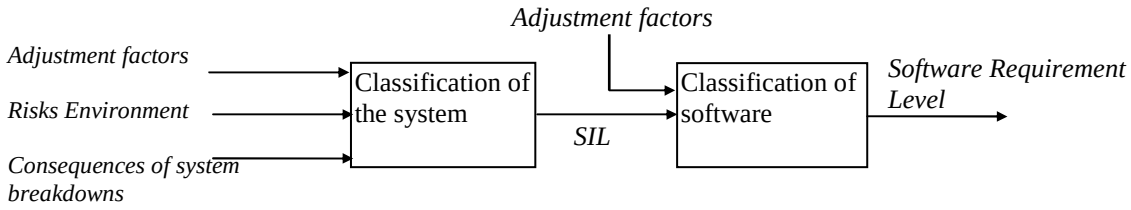
An error in the specification, design or coding can cause the failure of a system. The level therefore determines the degree of rigour required for the software development to avoid faults where the software could be the cause.

Determining the level and establishing a relationship between this level, the safety integrity level of the system and the category dependent on the area of application, the type of system, the damage it can cause to its environment (human in particular), the people it is intended for, and the structure of the system itself (architecture for example) : a specific evaluation of each system is necessary.

The current state of the art regarding software does not provide clear rules. A few guidelines for the software products that this document applies to are, however, provided to help determine the requirement level to be employed.

4.2 GENERAL PRINCIPLES FOR DETERMINING THE REQUIREMENT LEVEL

The process of classification to set the software requirement level is made up of two stages :



- **classification of system** : as the structure of the system and both the operational and environmental conditions have been defined, this involves identifying the types of dangers of this system (in all its operating modes) as well as the failures or erroneous use of the system and their consequences. This classification must take into account adjustment factors such as the architecture of the system, hardware redundancies or possible restrictions of use. The SIL or category is allocated to the system on the basis of the highest risk of the system.

- **software classification** : as the software product intended to ensure the (or certain) system functions has been defined, this involves determining the software requirement level to be set in accordance with the classification of the system.

As for the system aspects, certain system architecture or software design decisions should be taken into consideration to determine whether they affect the software requirement level retained.

In practice, the software requirement level is equivalent to the SIL system, unless acceptable justification has been given allowing for the reduction of the software

requirement level in relation to the system (hardware architecture or use of a particular piece of software, etc.).

For example, software with multiple diverse versions (or N-versions programming), a design technique that consists in creating two or several software components ensuring the same functions in a way that can prevent certain sources of common errors (introduction of heterogeneity through programming by different people, use of different languages, etc.) allows limitation of error impact or fault detection.

No precise rule exists, however, to deduce from this a reduction in the software requirement level, and a case by case analysis, often delicate, is necessary.

4.3 CLASSIFICATION OF SYSTEMS : CURRENT STATE OF STANDARDISATION

As indicated in the presentation paragraph, system classification is dealt with differently depending on the industrial sector and the standardisation authority. As an example, for the machinery, two types of classification can be applied: the EN-954-1 and IEC 61508.

These draft standards, in their current state of definition, do not highlight the immediate relationship between system classification. This absence of a relationship is based on two observations :

- the bases of evaluation are different : presence of faults and system behaviour in the presence of a fault for the EN-954 Standard, probability of a dangerous failure occurring in the case of IEC 61508,
- the target covered by these two documents is not the same. The EN-954 Standard is dedicated for all technologies, whereas CEI 61508 focused on E / E / PE systems.

4.4 THE CASE OF MACHINERY

Given the preceding observations, and the need for a case by case analysis of systems to set the software integrity level, the following information is only provided as an indication.

Account taken of the machinery context, the definition of the software quality requirements has directly targeted systems with important safety constraints but which are less critical than those of certain on-board avionics systems or control-command automation systems of nuclear power stations. This hypothesis has, in many cases, led to moderating the requirements either with respect to the IEC 61508 Standard or with respect to the state of the art in the other industrial sectors for more critical systems (level A or B of DO-178B for example)

It should also be noted that the graduation in requirement levels for the software does not stem from precise rules, but result from current trends in the graduation of the importance of different aspects of a software development agreed in several industrial sectors.

Provided that the system in question does not include specific arrangements (system architecture or software design level) **aimed at lowering the software requirement level**, the following relationship between categories and software requirement levels for the software could be proposed :

Categories of EN 954-1	**Requirement levels for the software**
2	*1*
3	*2*
4	*2*

It should be noted that the development of a software product at a given level does not imply that a failure rate has been allocated to it. The safety analysis cannot therefore use reliability rates based on the requirement level, as can be done for hardware failure rates.