Collaborative Large-scale Integrating Project

Open Platform for EvolutioNary Certification Of Safety-critical Systems

Industrial use cases: Description and business impact D1.2.b Avionics Use Case

Work Package:	WP1: Industrial Use Case Specification and Benchmark
Dissemination level:	CO
Status:	Final
Date:	02 July 2012
Responsible partner:	Cedric Chevrel – Thales avionics
Contact information:	cedric.chevrel@fr.thalesgroup.com

Contributors

Names	Organisation
Marc FUMEY, Cedric Chevrel	THALES Avionics

Document History

Version	Date	Remarks
V0.1	2012-03-14	Template: Contents of the use cases
V0.2	2012-04-15	First draft version including wide coverage, with some incomplete aspects.
V0.3	2012-04-26	First full version
V0.4	2012-06-08	Ready for WP review
V0.9	2012-06-22	Ready for PB review
V1.0	2012-06-28	Approved by PB

TABLE OF CONTENTS

Executive Summary **5**

1 Introduction **9**

2 System description **9**

2.1 Industrial use case study actors and environment 9
2.2 Industrial use case operational scenarios 10
2.3 Main functions provided by the system 11
2.4 Architecture of the system 11
2.5 General characteristics of the system 12

3 Development Lifecycle Activities **12**

3.1 Engineering and certification stakeholders 12
3.2 Activities executed by stakeholders 14

4 Engineering environment **15**

5 Description of the compositional approach **15**

5.1 Overall context 15
5.2 Use Case context 16

6 Summary of main argument for safety **17**

7 System lifetime events **17**

7.1 Planning 17
7.2 Development milestones 17
7.3 Production 18
7.4 Qualification trials in target environment or on final customer site 18
7.5 Certification, approval or award of qualification 18
7.6 Operation and maintenance 18

8 Relationship to conceptual and technical work packages and expected results **19**

8.1 General user interface: 19
8.2 Identification of elements to be managed using OPENCOSS platform 20
8.2.1 Common activities 20
8.2.2 Hardware design and verification activities 22
8.2.3 Software design and verification activities: 24

9 Conclusion **26**

List of Figures

Figure 1: Use-Case process environment 9
Figure 2: Use-Case environment and actors 10
Figure 3: Execution Platform & IMA Platform block diagram 11
Figure 4: certification steps 13
Figure 5: compositional approach 16
Figure 6: development milestones & reviews 18
Figure 7: Way of using OPENCOSS Framework in the use-case 19

Executive Summary

This document presents the avionic use case for the OPENCOSS study.

General context of Avionics Use-Case is a situation of reuse of product (Execution Platform, i.e. Computing Unit & Operating System) from one domain (Railways) to another domain (Avionic). The goal is to build the Qualification Dossier, based on elements provided with the reused parts.

Building this use case will lead to identification of all necessary data to be managed in OPENCOSS framework, and define their formats in order to be reusable between various partners.

General objective of use-case is the exchange of qualification data from Railways domain to Avionic domain. Presentation of use case includes general information permitting to understand general environment in which avionic IMA platforms are used.

REFERENCE DOCUMENTS

1. RTCA DO 178B - Software considerations in airborne systems andequipment certification
2. RTCA DO 254 – Design Assurance Guidance for Airborne Electronic Hardware
3. RTCA DO 297 - Integrated Modular Avionics (IMA) development Guidance and Certification Considerations
4. ARP 4761 Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment

GLOSSARY

Abbrev./terms		Definition
IMA	Integrated Modular Avionics	Represents avionic architectures in which resources (Processing, Network , Input Outputs, …) are shared between several applications.
	IMA platform	consists of the IMA modules and core software without any aircraft functions or applications installed.
	IMA platform architecture	refers to the means of structuring, connecting and combining IMA modules to support the requirements of the hosted applications and aircraft functions. Note: IMA platform architecture excludes applications.
	IMA Platform Architect	industrial actor in charge of defining IMA platform architecture. This actor is considered to be different than IMA Platform Integrators, Module Suppliers and Function Suppliers.
	Acceptance (Module Acceptance by Platform Architect)	acknowledgement by the Platform Architect that the module satisfies to all constraints defined by the IMA Platform Architect. This Acceptance is one criterion for incremental certification. In the scope of the use-case, this acceptance will assess that provided elements are relevant to be used for certification as expressed in plans.
	(IMA) Module	A component or collection of components that may be accepted by themselves or in the context of IMA. A module may also comprise other modules. A module may be software, hardware, or a combination of hardware and software, which provides resources to the IMA-hosted applications. Modules may be distributed across the

Abbrev./terms		Definition
		aircraft or may be co-located. 2 level of modules are identified: Execution Platform: object on which the use-case is focused, Computing Module: physical module that would be installed on aircraft. This level is not considered in the use-case.
	(IMA) Module Supplier	The Module Supplier develops and manufactures the module according to the module specification. In the use-case, Module Supplier designates the provider of the Execution Platform
	Resource	Any object (processor, memory, software, data, etc.) or component used by an IMA platform or application. A resource may be shared by multiple applications or dedicated to a specific application. A resource may be physical (a hardware device) or logical (a piece of information).
API	Application Programming Interface	The Application Programming Interface provides a standard interface between the application software and the platform (hardware and OS services), allowing independence between the applications and hardware.
	Partitioning	An architectural technique to provide the necessary separation and independence of functions or applications to ensure that only intended coupling occurs. The mechanisms for providing the protection in an IMA platform are specified to a required level of integrity.
	Partition	Resource segment, containing code and data, receiving part of application which can be loaded into the CPU memory of a MODULE. The partition is the CPU time and space segregation unit. The Operating System has absolute control over the use this partition makes of CPU resources (memory, time, other …) such that each partition is segregated from the other partitions on the MODULE.
	Aircraft Function	The capability of the aircraft that may be provided by the hardware and software of the systems on the aircraft. Functions include flight control, autopilot, braking, fuel management, flight instruments, etc. IMA has the potential to broaden the definition of avionics to include any aircraft function.
	Application	Software and/or application-specific hardware with a defined set of interfaces that, when integrated with a platform, performs a function. An Application may be composed of one or more executables installed in partitions located on one or several modules. Each executable may then perform part of a function.
	Function Supplier	Supplier of system application software function integrated in the Module (might be identical to System Supplier, when the whole system including the system application software function comes from the same

Abbrev./terms	Definition
	supplier).
ARINC-653	aeronautical standard defining API between an IMA platform and avionic applications.
Integrated Module	A module embedding the entire Configuration Table and all supported applications.
Platform (Module) Integrator	The Platform (Module) Integrator is responsible for the integration of the Integrated Module, i.e. providing guaranty that application can correctly be executed on modules. It includes the development of the global configuration parameters of the module. Note: the Platform Integrator does not consider functionality of applications.
Validation	this term in this document represents V&V activities relative to the specification and definition of a product.
Verification	this term in this document represents V&V activities relative to the integration and tests of a product.
Qualification Dossier	collection of all documents providing all data, artifacts and proofs in compliance with certification baseline. The Platform Qualification Dossier is established at IMA Platform level. This dossier is presented to certification by the IMA Platform Architect.

1. Introduction

General context of the Avionics Use Case is a situation of reuse of product from one domain (Railways) to another domain (Avionic). The goal is to build the Qualification Dossier, based on elements provided with the reused parts. The Qualification Dossier is then presented for certification.
It will be taken the example of Execution Platform (Computing Unit and Operating System) to build a scenario where complete Execution Platform or parts of it are provided by an industrial actor in a given domain (Railways) and installed in an architecture in another domain (Avionic).
The general way to proceed will be to identify data to be provided by the provider to permit building the qualification dossier of the Execution Platform in its final environment.
The present document aims to identify what the OPENCOSS framework should provide to improve certification process efficiency, reducing effort in building the platform qualification dossier.
The figure below represents the general process around the use-case.

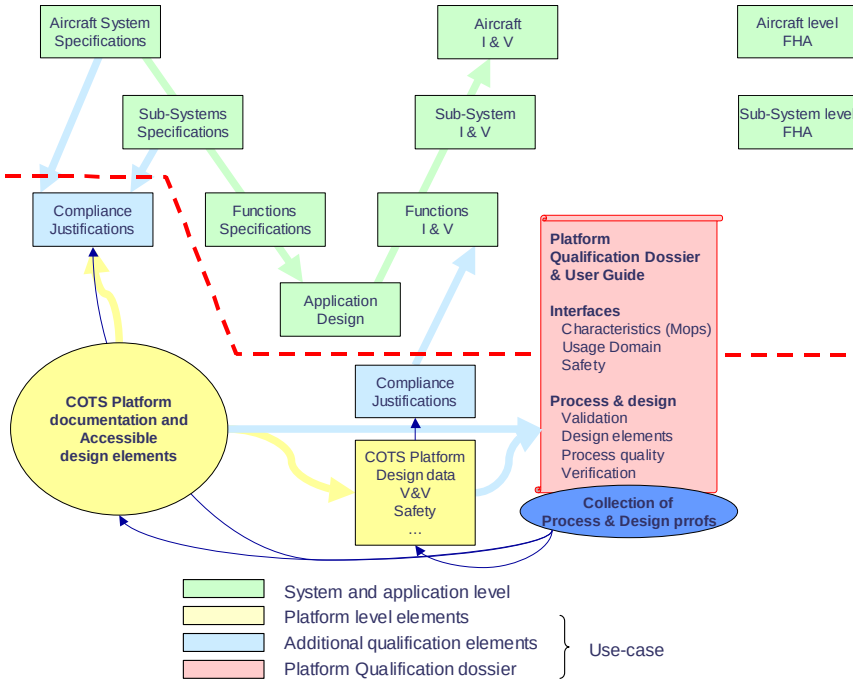

Figure 1: Use-Case process environment

2. System description

2.1 Industrial use case study actors and environment

The use-case proposed is relative to IMA general environment.
The Execution Platform is a common resource set used by several actors:

- It is defined and validated by the IMA Platform Architect
- It is provided by an Execution Platform Supplier, also called Module Supplier
- It is also used by Function Suppliers in charge of developing applications

Tools set is also delivered with the Execution Platform by the Module Supplier for the users (IMA Platform Architect and Function Suppliers), containing at minimum:

- OS configuration tools
- OS Simulation tools

Remark: the use-case will be focused on technical data to be provided for Hardware and OS. Tools will not be considered.

The main technical interfaces to be managed for the Execution Platform are:

- The API defining all services offered to applications
- The characteristics of Execution Platform and constraints for using it, necessary to establish Usage Domain of the IMA Platform

The main relation relative to the use-case is between IMA Platform Architect and Module Supplier: the IMA Platform Architect will formally accept the Execution Platform provided by Module Supplier.

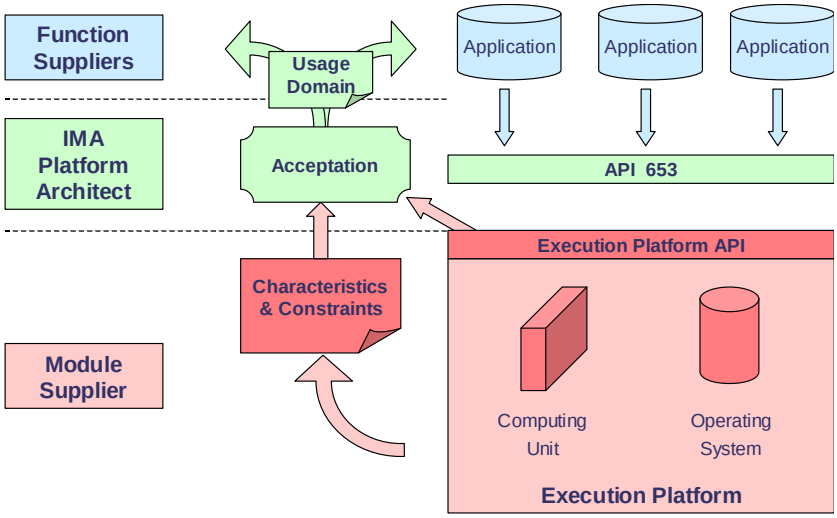

Figure 2: Use-Case environment and actors

2.2 Industrial use case operational scenarios

Use-Case operational scenario is the following:

IMA Platform Architect establishes general hypothesis and certification baseline:

- Sizing hypothesis (memory, processor throughput)
- Certification standards applicable (DO254, DO178B …)
- Functionality expected (API A653 …)

IMA Platform Architect fixes Execution Platform perimeter for Module Supplier:

- Hardware (Processing, IO, Mass Memory …)
- Software (OS, drivers, Platform System functions …)

Module Supplier provides Usage Domain (characteristics and usage constraints)

Module Supplier provides qualification material for certification demonstrations

IMA Platform Architect validates Module Supplier data and provides formal acceptation

2.3 Main functions provided by the system

Main resources and services provided by the Execution Platform are:

- Computing resource (processor)
- Operating System including:
    - OS heart providing:
- capability to manage real time scheduling
- capability to manage application processes
    - Drivers offering capability to access to IOs and network

Additional services (Platform System Applications) present on IMA Platform are not in the perimeter of the Execution Platform:

- Initialization and modes management
- Data-loading capability
- Monitoring and BITE capability

2.4 Architecture of the system

Typical IMA Platform architecture is the following:
Hardware level composed of:

- CPU board supporting avionic network interface
- IO boards connected to CPU via PCI internal bus

OS level composed of:

- OS kernel
- IO drivers addressing IO boards

IMA System level composed of system applications handling platform level services such as Data-Loading, BITE, Instrumentation

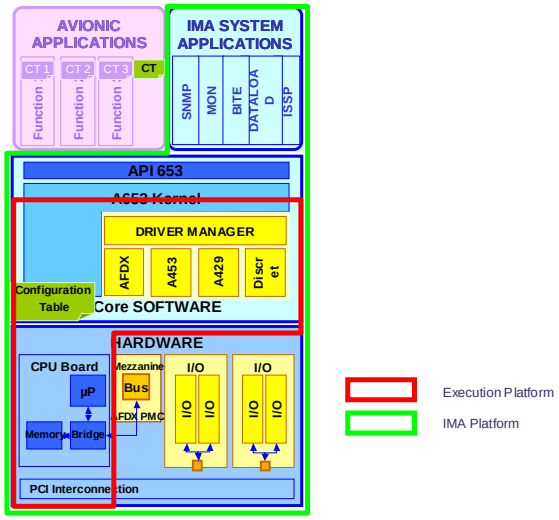

Figure 3: Execution Platform & IMA Platform block diagram

Avionic Applications are installed on the IMA Platform and use offered resources via A653 API

2.5 General characteristics of the system

Environmental conditions: defined by DO160 categories
Performances: capability to manage several avionic applications on one processing unit

Safety constraints:

- robust partitioning capable to support incremental certification process
- design insuring determinism: independence between parts of the Execution Platform (IO / IO, Network / IO, no demon processes in background, performances characterized in worst case, …)
- Undetected failure rate at 10-6 / h
- Loss rate at 10-5 / h

3. Development Lifecycle Activities

3.1 Engineering and certification stakeholders

General organization identified to come to certification is the following:

- Platform Architect will be in charge of all relations with certification authorities. Subsequently, the Platform Architect will build and present all elements and documents to the authorities in the scope of certification.
- Platform Architect will be supported by Module Supplier as far as possible, and use directly when possible the material provided by Module Supplier. Module Supplier may be asked to present the elements to the certification authorities under Platform Architect responsibility.
- Certification material (plans, documents, artifacts, summaries …) will be built or reused under Platform Architect responsibility.

There are generally 4 steps leading to final certification. These steps are covered by particular audits with following perimeter:

- Plans review validating methods employed
- Design review addressing specification and detail design
- Verification review addressing verification and artifacts
- Certification, assessing completeness and closure of all activities and actions

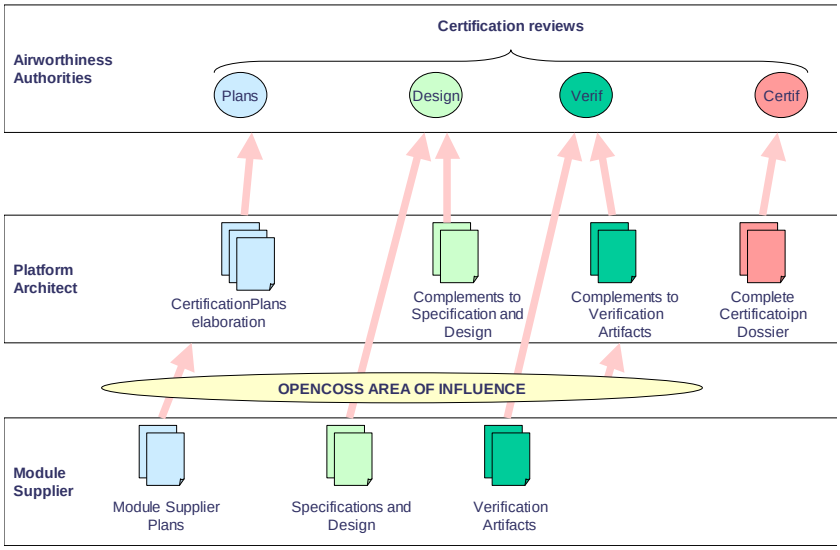

Figure 4: certification steps

Documents necessary for certification have been presented in the document D1.1. Paragraphs below provide description of the 4 certification review content, and associated documents presented for review for hardware and software domains.

Plans review content is detailed here after. Documents reviewed are the following:

- PHAC & PSAC (Plans for Hardware & Software Aspects for Certification). These plans are entry points for certification and identify:
  - activities and document provided for certification (artifacts),
  - specificities of organization (independencies between actors or teams),
  - methods and tools employed,
  - credits claimed from other projects or developments
  - particular technical aspects (novelties, specific techniques employed)

- Configuration Management, Quality Assurance plans providing
  - Methods and tools employed for configuration management
  - Process for evolution management
  - Quality process in place

- Validation and Verification Plans providing:
  - Reviews and criteria for requirements and test validation
  - Validation means and tools used
  - Verification means and tools used for integration and verification

- Standards defined to help and secure development covering
  - Plans
  - Specification and requirements
  - Design and coding
  - Tests

Design review content is detailed here after.

- Design Baseline identification, configuration management, evolution process and quality process are evaluated.

- Documents reviewed are the following:
  - Requirement document (including explanations, requirements, rationale, traceability information)
  - Design documents (high level/conception and detailed)
  - Code or drawings
  - Validation material (peer reviews, validation matrix)

- Sampling evaluation is done to verify that activities are conducted in conformance with plans and certification guidelines (DO178B & DO254). These sampling evaluate in particular:
  - Requirement correctness and completeness
  - Traceability to upper level requirements, and to design
  - Validation following criteria identified in plans and standards

Verification review content is detailed here after.

- Verification Baseline identification, configuration management, evolution process and quality process are evaluated.

- Documents reviewed are the following:
  - Verification test cases and procedures
  - Verification results

- Sampling evaluation are done to verify that activities are conducted in conformance with plans and certification guidelines (DO178B & DO254). These sampling evaluate in particular:
  - Test case and procedures correctness and completeness
  - Verification execution correctness and completeness
  - Verification following criteria identified in plans and standards
  - Traceability to requirements and coverage

Certification review content is detailed here after.

- Certification Baseline identification, configuration management, evolution process and quality process are evaluated.
- Recall of previous reviews, and evaluation of completion of all activities and pending actions.
- Documents reviewed are the following:
  - Accomplishment Summary providing or pointing all artifacts demonstrating compliance to plans and guidelines.
  - Configuration Index Document providing the complete identification of the baseline

3.2 Activities executed by stakeholders

Main activities of the various stakeholders are the following:
IMA Platform Architect:

- Defines the overall organization in which the Module Supplier will have to work
- Identifies Baseline (technical and certification)
- Provides development and certification plans, presenting the strategy of reuse.
- Defines outputs and level of involvement of Module Supplier (specifications, verification artifacts, complementary documents to provide …).
- Defines the platform specification, reusing and establishing the links with specification elements provided by Module Supplier
- Leads reviews with Module Supplier to validate outputs provided by Module Supplier
- Integrates the Execution Platform provided by Module Supplier in the IMA Platform
- Elaborates complementary elements for certification if necessary, as identified in plans

- Build the certification documents, and presents the IMA Platform (including Execution Platform) to certification authorities.

Module Supplier:

- Defines and provides development plans
- Presents these plans to Platform Architect to get agreement and acceptation
- Develops items if not already existing
- Provides all identified elements among which:
  - Safety elements: FMEA/FMES addressing failure modes computations, availability and undetected rate computation
  - Performances characterization in a worst case approach
  - User constraints leading to guaranty on functionality and performances
  - Particular analysis for determinism and partitioning
- Supports necessary reviews with Platform Architect to get agreement and acceptation
- Supports IMA Platform Architect for certification reviews

4. Engineering environment

OPENCOSS platform might be interfaced with other tools used in the overall development of the rest of the project. These tools are mainly:

- DOORS managing requirements and traceability
- CLEARCASE / CLEARQUEST for problem management and configuration management
- Other tools such as Word and Excel should also be taken into account.

Other tools may be considered, depending on the level of exchange between Module Supplier and IMA Platform Architect. Current tooling to be considered:

- Hardware mechanical: CATIA
- Hardware schematics: Various possible CAO tools. Should be considered through generic exchange standards.
- Software development: DIAB DATA compilers and linkers

5. Description of the compositional approach

5.1 Overall context

Overall context is presented in the figure below. Following elements are identified in an integrated platform. These elements should be considered separately in the compositional approach, because provided by various actors:

- The Execution Platform: object of the use-case, reused from Railways domain on an avionic architecture,
- Boards of computing module: provided by other suppliers, should not influence or be influenced by Execution Platform definition, except through identified interfaces
- Platform Applications are applications implementing transverse services for the platform (Data-loading, BITE …): provided by other suppliers, should not influence or be influenced by Execution Platform definition, except through identified interfaces (API & Usage Domain)
- Common Configuration: includes configuration parameters expressing the way the resource is organized and distributed to various applications. Provided by Platform Integrator, should not

influence or be influenced by Execution Platform definition, except through identified interfaces (Usage Domain)

- Applications: implement avionic functions. Provided by Function Suppliers, should not influence or be influenced by Execution Platform definition, except through identified interfaces (API & Usage Domain).

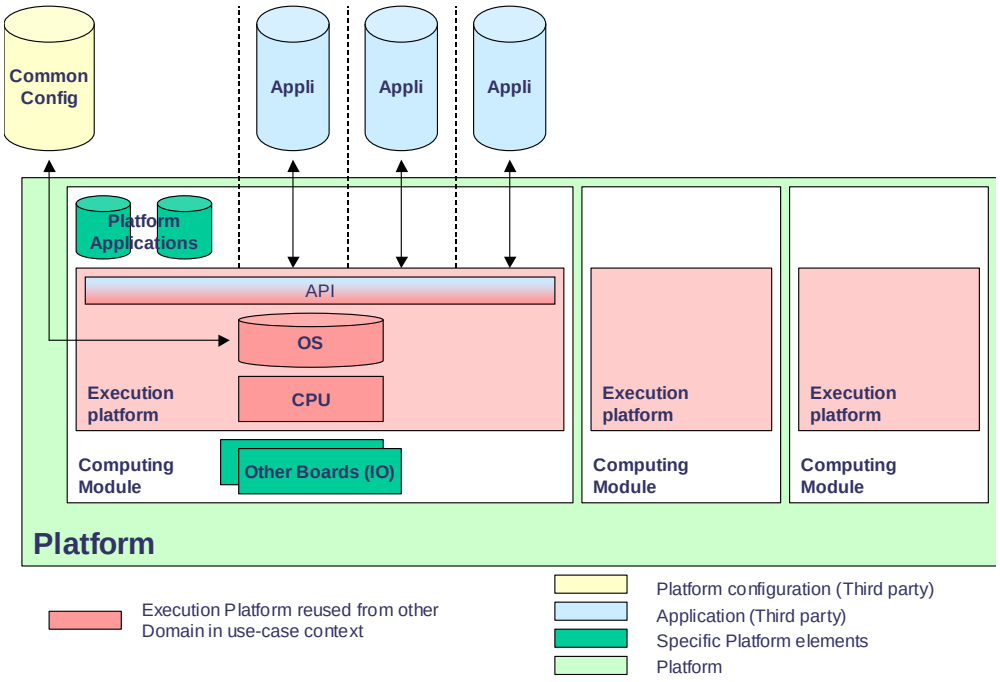

Figure 5: compositional approach

5.2 Use Case context

In the context of the OPENCOSS use-case, only part of the execution platform will be considered. This also represents a potential real industrial case where only part of platform is reused from another domain (typically OS heart).

As shown on Figure 3, several items are composing the execution platform:

- OS kernel
- Drivers
- Hardware items (boards)

All items (software or hardware) should be considered as independent, provided interfaces are defined:

- Hardware / Software interface
- Driver / OS interface

The proposed used-case focuses on data to be exchanged between stakeholders, mainly hardware and/or software data. Associated processes and data are described in § 8.2 below.

6. Summary of main argument for safety

The main arguments in the demonstrations due for certification are:

- Completeness and Correctness of specifications
- Segregation insurance between partitions
- Determinism of the design
- Completeness of verification artifacts
- Safety elements (undetected failure rates)

7. System lifetime events

7.1 Planning

The Avionic use-case is focused on technical data exchanged between domain relative to item exchanged. Planning considerations are not considered.

7.2 Development milestones

Development is mastered through development and acceptance reviews.
Development reviews constitute basic and classical development milestones. They are described in this chapter.
Acceptance reviews are more formal reviews and are qualification / certification oriented . These reviews are described in § 7.5 below.
All these reviews are presented on figure 6 below.

Development Reviews are the following:
KOR: Kick Off Review
Initialization of the project. It is verified that all hypothesis are valid (planning, engagements, responsibilities, teams identification …)
PDR: Preliminary Design Review
Validation of specifications, identify any risks and corresponding actions
CDR: Critical Design Review
Detailed design review. Validates that production (board, prototypes …) step can be launch without significant risk.
TRR: Test Readiness Review
Review before entering in integration steps that delivered elements to be integrated are sufficiently verified, and that integration & test means are available.
Certif: Certification review (IMA Platform level)
Assessment of completeness and correctness of all certification material.
There are typically 2 reviews: First Flight Review (FFAR), Certification review (CFAR)

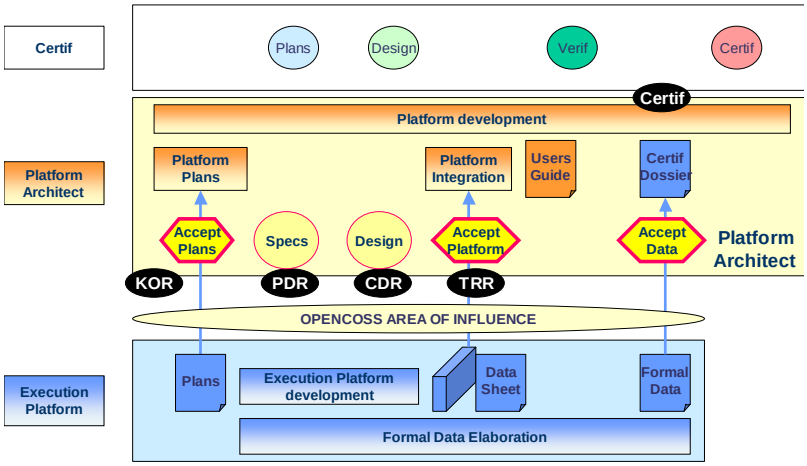

Figure 6: development milestones & reviews

7.3 Production

The Avionic use-case is focused on exchanges between domains for the type certification. Production considerations are not considered.

7.4 Qualification trials in target environment or on final customer site

Any data, artifact, analysis established for Execution Platform should be directly used for qualification. OPENCOSS here provides the opportunity to make these elements adequate for qualification in the avionic domain.
Nevertheless, it cannot be excluded that some qualification elements cannot be directly taken from Execution Platform (missing data, lack of representativity …) and used for qualification. In such cases, complementary activity or data will be established on the final IMA Platform.

7.5 Certification, approval or award of qualification

Acceptations: Independently of Development reviews, formal acceptances are performed:

- Plan Acceptation: validation that plans provided by supplier are compliant with expected criteria.
- Execution Platform Acceptation: based on status provided by supplier on characteristics, artifacts as expected in plans
- Formal Data Acceptation: based on final status on complete data package as expected in plans

7.6 Operation and maintenance

The Avionic use-case is focused on exchanges between domain to reach item qualification. Operation and maintenance considerations are not considered.

8. Relationship to conceptual and technical work packages and expected results

8.1 General user interface:

General vision of the way the OPENCOSS platform should be used is presented in the view below.

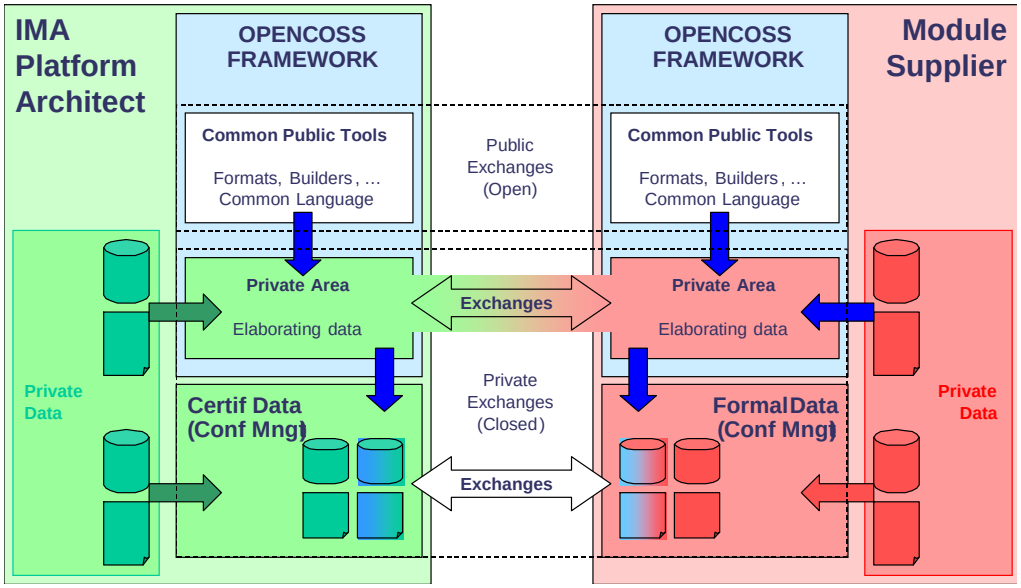

Figure 7: Way of using OPENCOSS Framework in the use-case

As a general principle, data presented to certification are considered as private and confidential.
They have then to be protected regarding access by any other third party than those identified and involved in a given project.
OPENCOSS platform should then be structured in separated common public area and separated protected private area.
This principle does not exclude exchanges managed through the OPENCOSS platform, but these exchanges should satisfy above expressed constraints.
2 levels of exchanges can be identified:

- Public exchanges: cannot in general concern technical data. These exchanges mainly correspond to providing formats, language elements, tools for a common tool box, …
- Private exchanges: exchanges between actors involved in a project. These exchanges can support technical certification data, provided confidentiality is insured. Considered data are the one elaborated in the activities identified in § 3.2.

Tool Box:
Outcome and benefit of using OPENCOSS platform in the proposed use-case is the identification of a collection of capabilities that can be called “Tool Box”.
This tool box should provide following capabilities (non limitative):

- Means to express complex logic argumentation linking various elements (generic common language and argumentation building tools)

- Means for any user of any domain (providing an item) to get access to a reduced set of necessary elements to provide for certification in another domain
- Means for any user of any domain (reusing an item) to identify usable data provided and insure an easy check of correctness or completeness.
- Means to link various elements such as specification items and verification elements, through advanced traceability means (simple links as well as complex argumentation provided by means above).

8.2 Identification of elements to be managed using OPENCOSS platform

This chapter provides the list of elements that are expected to be available under OPENCOSS platform. These elements are extracted from current certification plans (Software and Hardware), and presented in the logic order of the development process.
No format is presented, but content identified. The intent is to avoid elimination of elements which may not be fully compliant with current Thales avionics way to proceed. It is also taken into account that work will be necessary to complete and package evidences obtained from another domain.

Common principles exist behind software and hardware constraints defined in the DO178B and DO254 guidelines. These principles are summarized below:

- Design is requirement based. This leads to the identification of low level requirements accurate enough to realize the design, and perform a verification based on requirements, and independently from the design.
- Experience from previous projects and state of the art has to be fixed in standards followed in design phase.
- Requirements have to be validated using identified criteria to insure correctness and completeness as well as verification and maintenance.
- Independent verification from design has to be achieved to insure complete design coverage, and insure that design completely meets requirements.
- Proofs of achievement of above principles have to be provided for certification (evidences).
- Development process has to be mastered, through dedicated activity:
    - Configuration Management
    - Evolutions tracking
    - Quality Assurance

Process and data expected are presented below supposing a level A(highest) is identified for items. A possible outcome of the use case is the identification of a limitation in the DAL (Design Assurance Level) in the context explored, due to inexistent data or insufficient evidence.

8.2.1 Common activities

Project Management, Configuration Management and Quality Assurance

Development, Configuration Management process and Quality Assurance processes have to provide following evidences:

- Development plans established and followed
- Configuration Index Document established
- Independent assessment that activities are passed and followed identified criteria
- Reviews conducted under independent quality control

Expected elements coming through OPENCOSS:

- Configuration Index Document attached to hardware and software items providing
    - the list of documents applicable to the item
    - the history of previous versions

- the list of evolutions from last version
- Evolution history attached to hardware and software items including
  - Functional evolutions
  - Problem corrections
- Quality Assurance information attached to hardware and software items
  - Reports of reviews including action management and closure status
  - Hardware and Software item Configuration inspection reports

Safety assessment

Safety objectives are fixed from upper level (system level, product level) through system level preliminary safety analysis. These activities lead to identification of safety constraints allocated to Platform Hardware and Software:

- Failure conditions
  When specific failures are identified, constraint is provided to platform and hardware as follows:
  - Failure description (example "undetected erroneous data on specific output")
  - Failure rate tolerated (example "10-9 / hour")
  - Category (example "CAT" – for Catastrophic)
- DAL (Design Assurance Level)
  Each item of the platform has to get a DAL, from which relevant activities can be defined. Justification of DAL has to be provided, especially in case lower DAL is identified.
  By default, as platform are largely used, platform items DAL is the highest (A).
- FMEA/FMES has to be provided
  FMEA provides identification of failure modes of the platform, computation of the failure rate, following a validated method (MIL HDBK handbook).
  FMES provides analysis of effect of internal monitoring installed on platform (detection rates).
- SEU/MEU
  General objectives are generally identified relative to SEU/MEU effects.
  Means of elimination of SEU/MEU have to be set in place and characterized.
- Particular constraints
  Additional constraint such as partitioning is applicable to multi application platforms. Specific activity has to be addressed in development plans to sustain demonstration that partitioning is correctly handled.

Possible elements coming through OPENCOSS:

- FMEA/FMES of hardware items
  Typical format for FMEA/FMES is a table providing the list of identified failure modes, and for each of them:
  - Identifier including version id
  - Link to design (device or hardware block)
  - Failure rate and link to elementary data (failure mode data base)
  - Identification of monitoring
  - Detection rate of monitoring
  - Resulting failure rate
- SEU/MEU analysis of hardware items
- Partitioning/Segregation analysis for hardware and software items
  Such an analysis purpose is the identification of all interference channels that may exist between 2 segregated items (typically partitions). Analysis document should contain
  - Interference channel identification
  - Mechanism permitting to avoid interference
  - Associated constraints

  Associated validation evidence should be provided showing

- Completeness of analysis
- Correctness of analysis

Requirements capture:

The requirements capture process identifies and records the requirements, from the upper level specifications and design needs or constraints. This process is often iterative since additional derived requirements may become known during the next design processes.

Requirement capture process takes place for each item that is managed in configuration:

- Electronic board
- Complex Electronic devices
- Software items

Following attributes shall be defined for each requirement:

- Requirement identification, including individual version
- Requirement title
- Requirement text
- Derived requirement identification in case requirement is not directly deduced from upper requirement
- Rationale providing justification for this requirement
- Traceability to upper requirement
- Validation status

Possible elements coming through OPENCOSS:

- Hardware and software Item specification documents providing requirements including above defined attributes.

Validation activities:

Requirements shall be validated. Validation activity is based on several points of views:

- Traceability and completeness of the set of requirements,
- Accuracy and correctness of each requirement,
- Verifiability and maintainability of each requirement
- Justification that derived requirement do not alter safety objectives

Validation evidences have to be available for certification audits, then link has to be established for each requirement and validation element. Validation reports have to show completion of review for each requirement regarding validation criteria.

Possible elements coming through OPENCOSS:

- Validation elements defined above linked to requirements,
- Validation matrix, or elements permitting to build validation matrix based on requirements
  Validation evidences have to clearly show the validation criteria and the status for each requirement

8.2.2 Hardware design and verification activities

Hardware certification constraints are mainly expressed for complex devices (FPGA, ASICs, complex COTS). Two levels of hardware elements are then identified:

- Boards and simple devices for which simple evidences are requested
- Complex Devices for which complete set of evidences are requested

Board and simple devices activities:

Board and simple devices level activities are typically the following:

- Design

Translates the requirements into functional blocks, internal interfaces between blocks and inputs/outputs interfaces. It identifies constraints on software components.
Elements are gathered in a Hardware Design Document.

- Verification
  All requirements have to be covered by verification activity. Verification means (tests, analysis or inspection) has to be identified and justified. Complete coverage of requirements has to be shown.
  Verification procedures and tests have to be reviewed to establish correctness and coverage of requirement.

Possible elements coming through OPENCOSS:

- Board specification document
- Board design document and schematics
- Board Verification document
- Coverage matrix and coverage justification
  These evidences have to show:
  - Compliance to standards
  - Validation for each requirement
  - Test Coverage of each requirement

Complex devices design phases:

Complex devices design phases are typically the following:

- Conceptual design
  Translates the requirements into functional blocks, internal interfaces between blocks and inputs/outputs interfaces. It identifies constraints on software components.
  Elements are gathered in a Hardware Design Document (HDD).
- Detailed design
  requirements are translated, through the main functions identified during the conceptual design, into detailed functions and into a VHDL description of the device.
- Implementation
  produces hardware items, which implement the design and are compliant with their requirements.
- Production transition
  Production Readiness Review
  First Article Inspection
  Production Transfer Review
- Acceptance Test
  Verification of the assembly
  Verification of programming files using checksum

Possible elements coming through OPENCOSS:

- Hardware item Design Document
- Design files (VHDL)
- Binary files
- Result and status of reviews
  These evidences have to show on a requirement base:
  - Traceability between design and requirements
  - Verification of compliance to design standards

Complex devices verification phases

Verification ensures that the hardware component description and implementation meet its requirements. Verification independence has to be managed with design.

Complex devices verification activities are typically the following:

- Simulations
  Objective of simulation is the verification that the complete design is correct. It includes timing analysis taking into account routing.
- Design check
  Verification that standards are correctly and completely applied in the design.
- Physical verification
  This is the main verification phase, exercising the device in the real environment (on board).
  All requirements have to be covered by verification activity. Coverage of requirements has to be shown. Verification means are tests, analysis or inspection.
  Verification procedures and tests have to be reviewed to establish correctness and coverage of requirement.
- Robustness
  robustness tests have to be performed on the product in order to identify and check design margins.

Possible elements coming through OPENCOSS:

- Test procedures for simulation and physical test
- Review reports for validation of test procedures
- Simulation results
- Physical test results
- Analysis and inspection results
- Robustness test results
- Coverage matrix and coverage justification
  These evidences have to show:
  - Compliance to standards and verification criteria
  - Coverage of each requirement

8.2.3 Software design and verification activities:

Design phase:

The design phase is spitted in two parts: preliminary and detailed design.
The goal of preliminary design is to:

- identify architecture and software modules from functional requirements
- specify interfaces between software modules and real time constraints

The goal of detailed design is to describe the design to permit to enter in the coding phase using requirements only. The activities lead to:

- refine requirements and define Low Level Requirements for each software module
- specify software interfaces in detail
- establish traceability to upper requirements

The design is described in a Software Design Document providing:

- functional overview
- calling interface
- external services called
- external data used
- detail description of the Low Level Requirements

Possible elements coming through OPENCOSS:

- Software item Design Document providing

  - Architecture and interfaces description
  - Low Level requirements and Interface requirements
- Result and status of reviews
  These evidences have to show on a requirement base:
  - Traceability between design and requirements
  - Verification of compliance to design standards

Coding phase:

The goal of coding phase is to:

- Develop the source code
  Coding activity has to be done in accordance to design coding standards, defined and validated prior to coding phase.
- Compile the code to obtain object file
  Compilation condition (compilation options in particular) have to be identified and justified in standards to insure source code to object code traceability, necessary for level A.

Possible elements coming through OPENCOSS:

- Source code files
- Object code files
- Result and status of reviews
  These evidences have to show on a requirement base:
  - Traceability between code and design / requirements
  - Verification of compliance to design standards

Software verification phase:

The goal of this phase is to:

- Verify each Low Level requirement through Unitary tests
- Verify each High Level requirement through Functional Tests
- Demonstrate complete coverage of code
- Demonstrate complete flow control
  Coverage of all branching including combination of conditions
- Robustness tests
  robustness tests have to be performed on the product in order to identify and check design margins.

Possible elements coming through OPENCOSS:

- Test procedures for unit tests and functional tests
- Review reports for validation of test procedures
- Unit test and Functional test results
- Analysis and inspection results
- Robustness test results
- Coverage matrix and coverage justification
  These evidences have to show:
  - Compliance to standards and verification criteria
  - Coverage of each requirement

9. Conclusion

Avionic use case for the OPENCOSS study has been described in this document. General goal of this use-case is the exchange of qualification data between Railways and Avionic domains.
General information permitting to understand general environment in which avionic IMA platforms are used is provided.
Necessary data possibly managed in OPENCOSS framework, in order to be reusable by partners of Avionic domain has been identified.