|  |
| --- |
| UNCLASSIFIED |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | REFERENCES | ISSUE | DATE |
| logoIRT-StExupery |  |  |  |  |
|  |  |  |  |  |
| logoAirbus |  |  |  |  |
|  |  |  |  |  |
| logoONERA |  |  |  |  |
|  |  |  |  |  |
| logoRR |  |  |  |  |
|  |  |  |  |  |
| DASAVIA |  |  |  |  |
| logoSAFRAN |  |  |  |  |
| logoThales |  |  |  |  |

|  |
| --- |
|  |
| Streamlining Development Assurance  µXAV systems Process Definition |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

|  |
| --- |
|  |

|  |  |  |
| --- | --- | --- |
|  |  | Page : 1 / 46 |

|  |
| --- |
| This document is the intellectual property of RESSAC members It may not be used, reproduced, modified or disclosed without its authorization.  RESSAC members Proprietary Data. |

signatures

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | REFERENCES | APPROVERS |
|  |  | Fonction 1 |
|  |  |  |
|  |  | Name 1 |
|  |  | Fonction 2 |
|  |  |  |
|  |  | Name 2 |
|  |  | Fonction 3 |
|  |  |  |
|  |  | Name 3 |
|  |  | Fonction 4 |
| logoRR |  |  |
|  |  | Name 4 |
|  |  | Fonction 5 |
| DASAVIA |  |  |
|  |  | Name 5 |

Table of contentS

1. Document issues 5

2. PURPOSE AND SCOPE OF THE DOCUMENT 6

3. ORGANIZATION 6

4. LIFE CYCLE PROCESSES 6

4.1 Principles 6

4.2 Description 7

5. Process Activities 8

5.1 Planning process 8

5.1.1 µXAV systems process definition 8

5.1.2 Collective review (µXAV systems process definition verification) 8

5.2 Increment definition 9

5.2.1 Increment definition 9

5.3 µXAV systems level 10

5.3.1 µXAV systems specification 10

5.3.2 µXAV systems specification validation 11

5.3.3 µXAV systems functional decomposition ? 11

5.3.4 µXAV systems architecture 12

5.3.5 µXAV systems architecture verification? 12

5.3.6 µXAV systems integration and verification 13

5.4 System Level 14

5.4.1 System specification and architecture 14

5.4.2 System specification validation 15

5.4.3 Inter-System specification verification 15

5.4.4 System integration (implementation) and verification 16

5.4.5 Inter-System verification 16

5.5 Software items 17

5.5.1 SW item type 1: SCADE 17

5.5.2 SW item type 2: SPARK 18

5.5.3 SW item type 3: C and formal method 21

5.6 Hardware items 21

5.6.1 HW item type 1: method name 21

5.6.2 HW item type 2: method name 21

5.7 Equipment items 21

5.7.1 Equipment item type 1: method name 21

5.7.2 Equipment item type 2: method name 21

6. Environments 21

6.1 Tools 21

7. CONFIGURATION MANAGEMENT PROCESS 22

7.1 Configuration Management Environments 22

7.2 Change management principles 22

7.3 Life Cycle Data 22

8. Appendixes 23

8.1 Mission scenario template 23

# Document issues

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Issue** | **Author(s)** | **Updating purpose** |
| 10/01/2016 | First drafts | F.Pothon | Creation of the document based on several meeting, discussions, and contributions from RESSAC Partners |

# PURPOSE AND SCOPE OF THE DOCUMENT

The purpose of this plan is to describe the development and integral life-cycle processes for the µXAV/µXAV systems as developed in the scope of the RESSAC project

The RESSAC project will define Overarching Properties (OPs) and criteria for development assurance. In order to avoid any assumptions on the satisfaction of these OPs and criteria, the processes and data defined for the use case are identified independently of the definitions of Overarching Properties and criteria. Matching assessment between use case processes and RESSAC criteria will be performed in parallel in order to consolidate these criteria.

# ORGANIZATION

In the scope of the RESSAC project, the activities are shared between the project partners.

# LIFE CYCLE PROCESSES

## Principles

The product life cycle is incremental, developing progressively, a limited number of new functions/features. This incremental approach creates an iterative life cycle. Iteration consists of adding/modifying and verifying features of the µXAV systems.

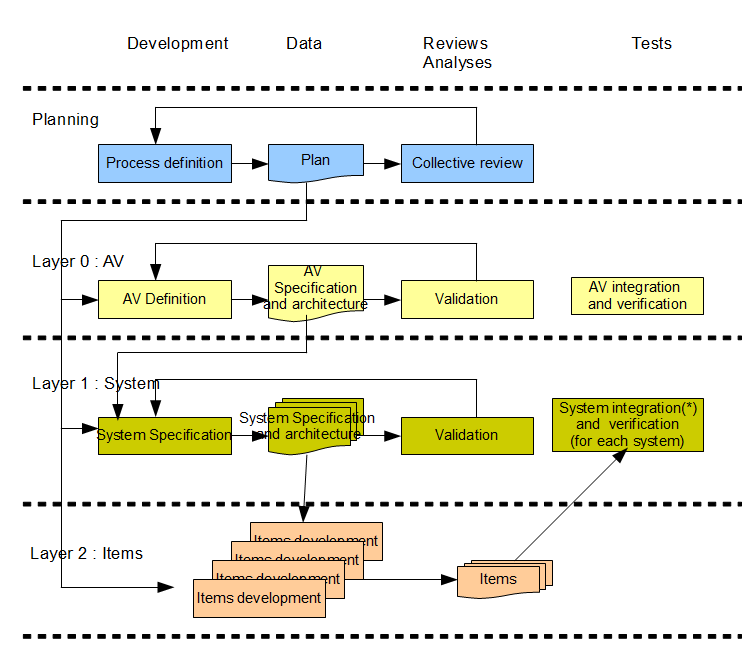
The complete development of the µXAV systems includes several abstraction layers:

* Layer 1: µXAV level. It consists in defining the µXAV specification and the µXAV external interfaces
* Layer 2: System level. This layer starts with the definition of the “system”, the specification of each system and the architecture of each system. This architecture includes the definition of each items of the system and their interfaces
* Layer 3: It is the item level. The development of each item is separately developed based on the system specification allocated to this item. Each item may use different methods and level of refinements (internal development tiers). Items are defined in the system architecture, and may be
  + A pure software item. Which means that the item is developed independently of the target hardware. HW/SW integration is performed at system level. The item is delivered for system integration in a form that will be compatible with the target hardware
  + One or several hardware components, complex or not. An electronic card is considered as a hardware item
  + “Equipment” where software is embedded in hardware.

The incremental approach may be applied at the 3 abstraction layers.

## Description

The following figure displays the activities to be performed on the 3 layers



(\*) System integration is the integration of items to implement the system

# Process Activities

## Planning process

This process consists of identifying the applicable life cycle and the activities to be performed. A first release of the µXAV systems process definition is provided at the beginning of the project. Then this definition may be updated iteratively as necessary. The µXAV systems process definition is accepted through a collective review by the RESSAC project partners.

### µXAV systems process definition

* Description:

The purpose of this activity is to identify the activities, environment, methods and responsibilities for the development, verification/validation of the µXAV systems.

* Methods:

The process definition is a textual document. It is developed through a set of collective workshops and discussion with various partners and experts. Each collective workshop is prepared through several exchanges and documented by a designated author.

* Environment:

No particular environment is necessary for this activity, only textual editor.

* Responsibilities: TBD
* Inputs: No specific inputs
* Outputs
  1. Process Definition Document

### Collective review (µXAV systems process definition verification)

* Description:

The purpose of this activity is to verify the applicability and correctness of the process definition.

* Methods:

This activity consists of a review performed during a face to face meeting. All attendees read collectively the document under review, and update it. Consensus is required to implement the changes.

The history section of the document records the date of the review, the main changes, and attendees

In case of remaining topics needing further investigations, discussions TBD

* Environment:

No particular environment is necessary for this activity, only textual editor.

* Responsibilities: TBD
* Inputs: No specific inputs
* Outputs:
  1. Updated Process Definition Document

## Increment definition

As stated in [§4.1 Principles](#_Principles), the development of µXAV systems is incremental. So the first iteration includes a subset only of desired behavior. After this first iteration identification of new functions are identified. These functions can be added at any layer.

### Increment definition

* Description:

The purpose of this activity is to identify the new functions to be added at µXAV systems and/or system layer and/or items level. Along with this identification, an impact analysis is conducted, identifying the change impact and the need for re-verification.

* Methods:

The increment definition is formalized in an Increment Definition File, which is a textual document. It is developed during a collective workshop.

* Transition criteria : The minimum criteria to launch the increment definition are
  1. End of last increment
* Environment:

No particular environment is necessary for this activity, only textual editor.

* Responsibilities: TBD
* Inputs
  1. Data produced during last increment
* Outputs
  1. Increment Definition File

## µXAV systems level

Three activities are performed at µXAV systems level: The µXAV systems specification, the µXAV systems specification validation and the µXAV systems integration (implementation) verification.

### µXAV systems specification

* Description:

The µXAV systems specification is mainly based on mission scenario description. The Mission Scenarios address the different operational modes and the possible degraded modes in case of failure, or abnormal environment conditions.

These scenarios are supplemented as necessary with additional requirements and constraints such as performance aspects. These additional requirements and constraints do not duplicate the scenarios but express characteristics and conditions that cannot be included in any scenarios.

The foreseeable conditions in which the µXAV systems will operate are identified. These conditions define the normal and abnormal inputs and conditions.

In parallel of mission scenario development, the µXAV systems external interfaces are defined.

Iterations between the different element of the µXAV systems specification (Mission Scenarios, additional requirements, external interfaces and foreseeable conditions) are performed to ensure the consistency of the complete specification.

* Methods:

The µXAV systems specification is a textual document.

Each scenario identifies step by step the conditions and inputs and observable properties. The combination of all mission scenarios should be representative of all operating conditions.

A scenario description uses the template defined in appendix A of this document.

This specification is used as an input for each system development. The Mission Scenarios may be directly used as inter-system integration verification cases, while additional requirements will be the purpose of additional validation activities.

* Transition criteria: The minimum criteria to launch the µXAV systems specification are:
  1. Applicable sections of Process Definition Document agreed
  2. Increment Definition File identified
* Environment:

No particular environment is necessary for this activity, only textual editor.

* Responsibilities: TBD
* Inputs

The µXAV systems specification is developed based on the knowledge and background of the RESSAC project patterns of the desired behaviour and of the foreseeable operating conditions.

* Outputs
  1. µXAV systems specification

### µXAV systems specification validation

* Description:

The µXAV systems specification is validated by the different stakeholders having the knowledge of the desired system behavior and foreseeable operating conditions.

* Methods:

The validation is performed through a proof-reading of the document.

* Transition criteria: The minimum criteria to launch the µXAV systems specification validation are:
  1. Applicable sections of Process Definition Document agreed
  2. µXAV systems specification under validation
* Environment:

No particular environment is necessary for this activity

* Responsibilities: TBD
* Inputs
  1. µXAV systems specification
* Outputs
  1. TBD

### µXAV systems functional decomposition ?

To be confirmed: Part of µXAV systems specification or another artefact?

* Description:
* Methods:

*NOTE: to discuss: use of SYSML for functional decomposition. It may be useful for consistency checks.*

* Transition criteria
* Environment:
* Responsibilities: TBD
* Inputs
* Outputs

### µXAV systems architecture

* Description:

The µXAV systems architecture is developed identifying the systems and their interfaces. Architectural mitigation introduced by safety analysis are addressed

* Methods:

SysML

*NOTE: to discuss: use of SYSML for architecture decomposition. It may be useful for consistency checks*

* Transition criteria
* Environment:

TBD

* Responsibilities: TBD
* Inputs
  1. µXAV systems specification
  2. Architectural mitigation provided by system analysis
  3. µXAV systems functional decomposition
* Outputs
  1. µXAV systems architecture

### µXAV systems architecture verification?

* Description:

A verification of consistency of the decomposition of the µXAV systems into items is performed with regards to the µXAV systems specification and functional decomposition

* Methods:
* Transition criteria
* Environment:
* Responsibilities: TBD
* Inputs
  1. µXAV systems specification
  2. Architectural mitigation provided by system analysis
  3. µXAV systems functional decomposition
* Outputs
  1. TBD

### µXAV systems integration and verification

* Description:

This activity consists of integrating the systems progressively and verifying the compliance of the µXAV systems to its specification.

* Methods:

After integration of the systems, this activity consists of piloting the µXAV systems, applying the mission scenarios of the µXAV systems specification.

Additional mission scenarios may be developed to checks further operating conditions combination as necessary, or performances.

* Transition criteria
* Environment: Integrated µXAV systems
* Responsibilities: TBD
* Inputs

µXAV systems specification

Integrated µXAV systems

* Outputs

TBD

## System Level

The following activities are performed for each µXAV system. It includes system specification, the system specification validation and the system integration (implementation) and related verification.

### System specification and architecture

* Description:

The activity may be conducted separately on each system. It consists of defining the expected behavior of the system. Then for each system the architecture is defined, identifying all of its items, and their interfaces.

Iterations between the different elements of the system specification (system requirements and architecture) are performed to ensure the consistency of the complete specification.

* Methods:

Modelica is used for system specification

System architecture: SysML

* Transition criteria ? µXAV systems specification exists and upper level functional decomposition is available with interfaces definition.
* Environment:

SysML

Modelica

* Responsibilities: TBD
* Inputs

µXAV systems specification

µXAV systems architecture

* Outputs

System specification (Modelica + SysML)

### System specification validation

* Description:

Each system specification is verified for correctness and consistency.

* Methods:

The verification is performed through modelica simulation. (To be completed)

* Transition criteria ?
* Environment:

Modelica simulation

+ SYSML consistency checks

* Responsibilities: TBD
* Inputs

System specification (Modelica model)

* Outputs

TBD

### Inter-System specification verification

* Description:

A verification of consistency of all system specification is performed. This verification is based on the µXAV systems specification and µXAV systems architecture. Its purpose is to detect the complete implementation of the mission scenario and compliance to the additional requirements. System requirements that do not participate to any µXAV systems specification items are identified and justified.

* Methods:

If systems architecture are modeled with SYSML, then architecture consistency can checked by SYSML checker

* Transition criteria ?
* Environment:

TBD

* Responsibilities: TBD
* Inputs

µXAV systems specification

µXAV systems architecture

All System specification

* Outputs

Verification results.

Consolidated architecture.

### System integration (implementation) and verification

* Description:

This activity consists of verifying the compliance of each system to its specification. This activity is performed separately on each system.

* Methods:

This verification is based on the use of modelica using an incremental approach. When an item is available, it is plugged into the models and replaces its specification. Then the simulation performed for system specification verification is re-run and compared to the simulation results.

What if the modelica model is not as accurate as the software model (e.g. SCADE)? This will lead to behavioral differences.

Quid if there are minor differences in value and time?

This activity is re-entered each time new items(s) are available.

* Transition criteria ?
* Environment: Integrated system
* Responsibilities: TBD
* Inputs

System specification

Items

* Outputs

TBD

### Inter-System verification

* Description:
* Methods:
* Transition criteria ?
* Environment: Integrated system
* Responsibilities: TBD
* Inputs
* Outputs

TBD

## Software items

Three technologies are identified for software items development (SCADE model, SPARK, and C language+formal method)

### SW item type 1: SCADE

#### Methods definition

* Description:

The purpose of this activity is to define guidelines to apply the chosen methods on the item.

* Methods:

write a development and verification plan

* Environment:

No particular environment is necessary for this activity, only textual editor.

* Responsibilities: TBD
* Inputs:

No specific inputs

* Outputs

SCADE guidelines

#### Item development

* Description:

The purpose of this activity is to develop the SCADE model to implement the system requirements allocated to the item

* Methods:

SCADE model design

* Environment:

SCADE editor and checker

SCADE KCG for code generation (can/should it be moved to system integration?)

* Responsibilities: TBD
* Inputs:

System specification allocated to the item

* Outputs

SCADE model

C or Ada source code and/or object code (to be discussed)

#### Item verification

* Description:

The purpose of this activity is to verify the compliance of the SCADE model to the system specification allocated to the item

* Methods:

Semantic checks

Model simulation

Model review (complementary to model simulation)

* Environment:

SCADE simulation

Semantic checks with KCG

* Responsibilities: TBD
* Inputs:

System specification

SCADE model

* Outputs

Simulation procedures and results

Model review results

### SW item type 2: SPARK

#### Methods definition

* Description:

The purpose of this activity is to define guidelines to apply the chosen methods on the item.

* Methods:

TBD

* Environment:

No particular environment is necessary for this activity, only textual editor.

* Responsibilities: TBD
* Inputs:

No specific inputs

* Outputs

SPARK guidelines (requirements and code)

Formal analysis description

#### Item requirement development

* Description:

The purpose of this activity is to develop the item requirements (in SPARK) to implement the system requirements allocated to the item

* Methods:

SPARK

* Environment:

TBD

* Responsibilities: TBD
* Inputs:

System specification allocated to the item

* Outputs

Item requirements

#### Item requirement verification

* Description:

The purpose of this activity is to verify the compliance of the item requirements to the system specification allocated to the item

* Methods:

Review?

* Environment:

TBD

* Responsibilities: TBD
* Inputs:

System specification

Item requirements

* Outputs

TBD

#### Item implementation

* Description:

The purpose of this activity is to implement the item requirements into source code, and to produce the executable object code

* Methods:

Ada

* Environment:

Gnat Ada environment

* Responsibilities: TBD
* Inputs:

Items requirements

* Outputs

Source code

Executable object code

#### Item verification

* Description:

The purpose of this activity is to verify the compliance of the item implementation to the item requirements

* Methods:

The activity consists of running GNATProve. The tool detects all cases where the contracts are not satisfied.

* Environment:

GnatProve

* Responsibilities: TBD
* Inputs:

Item requirements

Source code

* Outputs

Analysis report

### SW item type 3: C and formal method

## Hardware items

### HW item type 1 COTS AEH

We consider here below one particular type of Airborne Electronic Hardware (AEH) item, namely a Commercial Off-The Shelf ( COTS) Integrated Circuit (IC) such as COTS Controllers (not intended to execute Software) or COTS Microcontrollers (destined to execute Software), the latter being possibly Multi-Core, multi-Peripheral, and both are in general Complex AEH per ED-80/DO-254.

In the scope of RESSAC usecase we consider a Microcontroller from a well-known provider (e.g. Freescale) with the following characteristics:

* Multicore with at least two of them used by the application

#### Some communication peripherals (e.g. CAN bus driver) are used by the application General introduction

A COTS component is more or less a black box that can only be viewed as an interface between its inside structure & functions, and its outside conditions & surroundings. This is only when matching between the inside and the outside can be shown to fit correctly that we could say that such a black box would serve the expected purpose for the hardware design in which it is embedded.

The ED-80/DO-254 section 11.2 on COTS AEH, recognizes that a process-approach to design and development assurance, does not really apply to COTS as the necessary artifacts, i.e. design and process data are not available for review and only a limited amount of descriptive data is available for hardware and system design. This clearly suggests that COTS AEH really need an alternate or at least and adapted process, and possibly new approaches to what is currently available in general per ED-80/ DO-254 guidance and acceptable practices during a certification process.

COTS AEH can then be selected as good candidates to quickly test a properties-based assurance approach without the need to refer to the whole set of objectives, activities and data recommended by DO-254/ED-80 for developmental hardware items. The process described below consists in addressing the various aspects -generally considered during a design- and as instantiated for COTS.

#### Item Specification as Definition of the Intended Function (DIF)

The intended function of a COTS AEH Component can only be established on the basis of available device data (e.g. its datasheet). Of course assurance of a COTS component would be easier if design data is in addition available from the COTS supplier. However this is not generally the case.

A COTS AEH Item will only be able to perform its intended function when properly used within limitations and recommendations provided by the COTS Supplier via the datasheet or via other manuals, or as established by the AEH designer per in-house dispositions.

* Description:

The purpose of the activities that are accessible for a COTS and that can be incorporated in the assurance process is to capture all data available and intended to be used for the hardware design:

* Description of the COTS device characteristics, in terms of functions, functional blocks & functional modes. This is generally provided via the basic datasheet of the COTS. Key characteristics of the COTS that need to be captured in order to allow matching with upper level of DIF or upper layer (e.g. tier at Circuit Board Assembly (CBA) design.
* Identification of other COTS available data: user’s manual, installation or interface manuals and errata sheets, possibly design data available from the COTS supplier.
* Definition of the Usage Domain (UD) limitations, i.e. limitations in terms of used and/or unused and activated and/or deactivated COTS functions, and configuration settings, including default settings at power-up or reset. A preliminary H/W – H/W and H/W – S/W Interface Document may be established at this stage of the development.
* Methods:

Engineering analysis documented in Circuit Board Assembly (CBA) Requirements Specification and/or in CBA Hardware Design/Description Document (HDD), or any form.

* Environment:

No particular environment is necessary, only text editor and configuration management.

* Responsibilities:

The Hardware Engineering Manager, Hardware architect and designer

* Inputs:

COTS data from the COTS supplier. CBA Requirements and Design

* Outputs

Documented description of COTS Intended Function

#### Item Validation vs Desired System Behavior (DSB)

A COTS is not developed using a structured process per ED-80/DO-254 guidance, hence there is neither “Requirement capture” nor “Requirements validation” as addressed by ED-80/DO-254. However the COTS is expected to fit the next upper level of integration of either H/W, S/W or both.

To this end, a matching assessment should be documented between the requirements allocated to hardware at CBA level and the COTS characteristics and performance, including COTS limitations, functions, modes, configurations, interfaces, peripherals, etc.

* Description:

The purpose of this activity is to Validate that the selected COTS together with its identified limitations are commensurate with the requirements and design at CBA layer.

- Identification of the overall system and safety purpose expected from the COTS component (e.g. allocated safety and functional requirements, functional modes, etc.),

- Validation of limitations with respect to the as-captured system, safety & software (if any) allocated requirements.

- Assessment of H/W – H/W and H/W – S/W (if any) Interfaces as matching with the upper level of integration.

* Methods:

Validation Technical Review based on dedicated checklist

* Environment:

Text editor, review forms and configuration management

* Responsibilities:

The Hardware Engineering Manager, Hardware architect and designer

* Inputs:

CBA Design, COTS Datasheet and documented limitations, Checklists

* Outputs

Review report.

#### Item Safety for Proper and Safe functioning

Both the functional behavior, and potential dysfunctional behavior should be analyzed at the adequate level to show that safety objectives are met whatever the configuration settings of the COTS (if configurable), including for inadvertent alteration of those settings.

In addition, the safety impact of unused functions and both known errata and new ones should be analyzed in order to make sure that as-designed workarounds and mitigation remain compatible with the safety objectives and adequately cater for both functional and dysfunctional behaviors.

* Description:

The purpose of this activity is to ensure that the COTS once embedded into its surrounding hardware design, will be able to meet the safety requirements allocated from system to hardware.

- Analysis of functional behavior depending on configuration settings of the COTS component (if any) taking into account the potential impact of inadvertent “wake up” of unused function.

- Analysis of Errata workarounds or mitigation w.r.t. functional or safety impact and aggressive environmental conditions (HIRF, LIE, SEU).

- Analyses of dysfunctional behavior (FMEA/FMECA/FFPA, etc.), including for inadvertent alteration of settings, and feed back to system safety.

* Methods:

Architecture analyses. Errata Analysis, Failures analysis

* Environment:

Excel sheets, tools for failure analyses

* Responsibilities:

Safety or FMEA experts. Expert knowledgeable in the COTS device.

* Inputs:

Errata sheets. CBA detailed design. COTS Datasheet and any additional data

* Outputs

FMEA/FMECA at CBA level including COTS details to an adequate level.

#### Item implementation to Technically Suitable Performance

All characteristics and performance of the COTS component that are contributing functionally, environmentally or for any safety reason in the design of the encompassing system (e.g. Circuit Board Assembly, unit of equipment or system); must be considered as essential features.

In addition, the limitations of the COTS should be taken into account in the implementation and shown compatible with the COTS surrounding design

* Description:

The purpose of this activity is to implement the COTS within its surrounding CBA design and development while ensuring that suitable characteristics and performance could be featured.

* The defined limitations of the COTS component should be properly satisfied, possibly enforced by its surrounding CBA design,
* Considerations on all characteristics and performances that are critical to proper operation (e.g. signal quality & timings, power supply, temperature ratings, etc.)
* Assessment of any other safety mechanisms (monitoring, partitioning, protections, etc.,) when provided by the COTS to contributing to any kind of performance.
* Methods:

Hardware CBA design

* Environment:

Hardware Design workshop

* Responsibilities:

Hardware designers and implementers

* Inputs:

CBA Hardware Design Description, COTS Limitations, Datasheet and Errata sheets

* Outputs

CBA Detailed Drawings and Layout

#### Item verification within Operating and Environmental Conditions (OEC)

Functional verification and environmental qualification is generally performed at the level of the unit of Equipment (e.g. LRU), not at COTS device or even at CBA levels. Operating conditions mean interfaces, environmental conditions and robustness to normal & abnormal conditions.

This is commensurate with the fact that COTS can only be considered at the level of their interfaces and related behavior while verification of a COTS is always performed when integrated within its CBA environment.

* Description:

The purpose of this activity is to verify conformity of the COTS implementation within the CBA Implementation versus the CBA requirements and other COTS key characteristics.

* Verification of the CBA H/W against CBA including to requirements allocated to the COTS, and to H/W – S/W Interface (HSI).
* Qualification of the encompassing hardware (generally at the LRU level) to its operating and environmental conditions,
* Analysis of functional coverage and adequate completion of verifications together with transition to production, incl. Acceptance Test.
* Methods:

Verification Test benches

* Environment:

Test Lab

* Responsibilities:

Verification team

* Inputs:

Physical implementation

* Outputs

Tests and Analyses Procedures and Reports.

### HW item type 2 : High Level Synthesis approach for FPGA and SoC development

#### Introduction

HLS method (High Level Synthesis) will be used to develop and verify a FPGA.

Such method is based on a C, C++ or System C description of the FPGA functionality.

By using dedicated tools (as XilinX Vivavo suite) it will be possible to generate HDL code and then implement it into a FPGA.

Verification will be based on HLS description instead of HDL description, allowing a faster maturity loop and an easy means to evaluate constraints impact on design.

In such a way VHDL code will be only a design tier, necessary to go through some design or verification step, but not the focal point of the FPGA design as it is today.

Moreover HLS approach will allow an easier link to system model (if any) and then, will preserve the modeling continuum.

This approach can be combined with IP description and IP-XACT standard usage (System on Chip design and verification)..

HW/SW co-simulation can also be used in this environment.

#### Methods definition

* Description:

The purpose of this activity is to define guidelines to apply the chosen methods on the item.

* Methods:

TBD

* Environment:

No particular environment is necessary for this activity, only textual editor.

* Responsibilities: TBD
* Inputs:

Inputs from HLS solution provider (e.g. Xilinx), HLS users (e.g. feedbacks from DO254 User Group members, feedback from Ingequip teams )

* Outputs

HLS guideline for design and verification of a FPGA in an avionic DAL A certification context.

#### Item requirement capture

* Description:

The purpose of this activity is to develop the item requirements (in C/C++/System C) to implement the system requirements allocated to the item.

Remarks :

No natural language specification anticipated here. Only HLS description will be produced

* Methods:

Could be

* 1. Extracted by tool from system model (link to system to be defined) and addition of local constraints (technology, ressources …)
  2. Handmade description of item requirements and associated constraints
* Environment:

TBD

* Responsibilities: TBD
* Inputs:

System specification allocated to the item

* Outputs

Item requirements (in C, C++,System C) and constraints

#### Item requirement validation

* Description:

The purpose of this activity is to ensure the compliance of the item requirements to the system specification allocated to the item

* Methods:

Review

Model based verification

Formal proof or equivalence proof with system model

* Environment:

TBD

* Responsibilities: TBD
* Inputs:

System specification

Item requirements

* Outputs

Validation report based on activity sanctions.

#### Item implementation

* Description:

The purpose of this activity is to implement the item requirements into source code (VHDL/Verilog) to produce the executable object (bitstream) and the FPGA item (or part of the FPGA).

* Methods: Automatic generation of the successive artefacts/tiers)
* Environment: Automatic generation suite (e.g. Xilinx Vivado suite)
* Responsibilities: TBD
* Inputs: Items requirements
* Outputs

Intermediate tiers : VHDL/Verilog HDL code, VHDL/verilog netlist, bitstream

Final implementation : FPGA component loaded with bitstream

#### Item verification

* Description:

The purpose of this activity is to verify the compliance of the item implementation to the item requirements

* Methods:

Verification of tiers regarding HLS model

Correct by construction in case of qualified tool for some steps

TBC (see below a Xilinx presentation as an example) 

* Environment:

TBD

* Responsibilities: TBD
* Inputs:

Item requirements

Tiers

Final implementation

* Outputs

Analysis report

Tool qualification dossier

### HW item type 2: method name

## Equipment items

### Equipment item type 1: method name

### Equipment item type 2: method name

# Environments

## Tools

# CONFIGURATION MANAGEMENT PROCESS

## Configuration Management Environments

Github

## Change management principles

## Life Cycle Data

# Appendixes

## Mission scenario template