Flight SW Autocoding Life-cycle process – Model-in-the-Loop

D4.1

Document Code:	AUR-SEN-RP-0031
Document Version:	1.0
Document Date:	15/11/2021
Internal Reference:	DOC00205396

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101004291

Signature Control

Written	Checked	Approved Configuration Management	Approved Quality Assurance	Approved Project Management
J. Gómez Javier Gomez del Pulgar Vazquez Firmado digitalmente por: Javier Gomez del Pulgar Vazquez ND: CN = Javier Gomez del Pulgar Vazquez email = javier.gomezpulgar@aeroespacial.sener C = AD O = Sistemas de Vuelo OU = Sener Aeroespacial Fecha: 2021.12.02 08:12:40 +01'00'	A. Rodríguez Ana Isabel Rodriguez Firmado digitalmente por Ana Isabel Rodriguez Fecha: 2021.12.02 15:57:02 +01'00'	R. Talavera Raquel Talavera Firmado digitalmente por: Raquel Talavera ND: CN = Raquel Talavera email = raquel.talavera@aeroespacial.sener C = ES O = SENER Aeroespacial OU = CDM Fecha: 2021.12.02 16:06:15 +01'00'	A. López Alfonso López Firmado digitalmente por: Alfonso López ND: CN = Alfonso López email = alfonso.lopez@aeroespacial.sener C = ES O = SENER Aeroespacial S.A. OU = Quality Division Fecha: 2021.12.02 09:26:58 +01'00'	A. Rodríguez Ana Isabel Rodriguez Firmado digitalmente por Ana Isabel Rodriguez Fecha: 2021.12.02 15:57:17 +01'00'
Date and Signature	Date and Signature	Date and Signature	Date and Signature	Date and Signature

Signature not needed if electronically approved by route

Changes Record

Rev	Date	Author	Affected section	Changes
0.1	2021-10-25	J. Gómez	All	Initial draft version
1.0	2021-11-15	J. Gómez	All	Initial issue

Index

1. Introduction

1.1. Purpose

This document describes the Flight SW life-cycle for autocoding and the different processes and stages of a model-based process which cover the whole SW life-cycle from requirements to qualification.

The procedure is mainly focused for AOCS/GNC SW which has been selected as the primary use case of the project, but it can be adapted to other subsystems as well.

1.2. Scope

The Flight SW autocoding life-cycle process definition is the main core of the *WP4 Flight SW Autocoding Life-cycle Process Definition* of AURORA, as described in Annex 1 Part A of [AD1]. The document gathers the main process for the SW generation toolchain departing from the System requirements up to complete qualification, detailing it for the different stages of a typical software verification process

This document is an output of the T4.1 activity included in WP4. Future version of this deliverable will be provided at months M15, M18 and M21.

1.3. Document structure

The document has been structured as follows:

- Section 1: this introduction
- Section 2: Related documentation
- Section 3: Overview of the AURORA methodology
- Section 4: Flight SW Autocoding Life-Cycle Process
- Section 5: Model-in-the-loop stage
- Section 6: Software-in-the-loop stage
- Section 7:Processor-in-the-loop and Hardware-in-the-loop stage.

2. Related documentation

The following documents in the latest issue/revision from a part of this document.

2.1. Applicable documents

AD #	Title	Project Reference	Issue	Rev
[AD1]	AURORA Grant Agreement	GA number 101004291	-	-
[AD2]	AURORA Consortium Agreement (CA)	CA Nº 101004291 AURORA	-	-

Table 1 Applicable documents

2.2. Reference documents

RD #	Title	Reference	Issue	Rev
[RD1]	Space engineering Software	ECSS-E-ST-40	C	-
[RD2]	Space Software Product Assurance	ECSS-Q-ST-80	C	-
[RD3]	Software Engineering Handbook	ECSS-E-HB-40	A	-
[RD4]	Guidelines for the Automatic Code Generation for AOCS/GNC flight SW Handbook. Vol1 – General concepts	-	1	0
[RD5]	AOCS/GNC Modelling Guidelines	AUR-SAE-RP-0006	1	1
[RD6]	Guidelines for the Automatic Code Generation for AOCS/GNC flight SW Handbook. Vol2 – Mathworks specific guidelines	-	1	1

Table 2 Reference documents

2.3. Acronyms

Acronym	Description
AD	Applicable Document
ATB	Avionics Test Bench
COTS	Commercial Off The Shelf
EBd	Executive Board
ESE	Engineering Simulation Facility
FES	Functional Engineering Simulator
GA	Grant Agreement
GeA	General Assembly
HILF	Hardware-In-the-Loop Facility
HW	Hardware
MIL	Model in the Loop
N/A	Not Applicable or Available
PFM	Proto Flight Model
PIL	Processor in the Loop
RD	Reference Document
SDP	Software Development Plan
SIL	Software in the Loop
SRR	System Requirements Review
SVF	Software Verification Facility
SW	Software
TRB	Test Review Board
WP	Work Package

Table 3 Acronyms

2.4.Terms and definitions

N/A

3. Overview

The AURORA WP4 “Flight SW Autocoding Life-cycle Process Definition” [AD1] approaches the definition of a SW Autocoding Life-cycle Process, where Autocoded system refers to any Complex-Models systems that make a full use of MATLAB/Simulink for modelling the algorithms and behavior of the system. The most representative case of such a system in Space missions are the AOCS/GNC systems. In our approach Design and Development and running chained to verification activities and therefore improving the OBSW integration and validation program.

This approach is supported by:

- An early verification of the navigation models.
- Auto-generated source code software following an iterative process.
- Mission requirements Verification at GNC model level and component model.
- Integration phase when OBSW components implement standard interfaces (API).
- Aligned with Space standards and allowing as much as possible the automation of the process.
- Iterative execution of the WP taking inputs from the technology Demonstrator activity.

The Model-in-the-loop (MIL) , Software-In-the-Loop (SIL) and Processor-In-the-Loop (PIL) are key points of the incremental validation in order to verify the behavior of the GNC code in a representative environment and to identify computational resources required through code profiling.

The whole process is iterative. This means that it is applicable several times for each function/mode iteration. The functional iterations are defined e.g., for a subset of functions that can be easily validated independently. For example, an AOCS iteration is associated with an AOCS mode. In the following, the subsystem of choice is the AOCS, but could be any functional chain subsystem expressed with models having Autocode capability (e.g., thermal, power).

This activity enclosed the definition of following **In-the-loop** steps:

- Model-in-the-loop

  The models have to comply with Aurora modelling standards and guidelines, (QGen framework) and the model simulations demonstrate the feasibility of the preliminary design and the robustness of the selected solutions using Monte Carlo test campaigns. Being able to perform such tests during the preliminary stages of the development allows for efficient iterations at system level, giving valuable contributions for trade-offs that involve other subsystems.

- Software-In-the-Loop

  The auto-coding of the navigation model (QGen framework) will allow testing the Autocoded SW with respect to the algorithms already validated in a MIL environment.

- Component-In-the-Loop

  The SW as an OBSW component has to follow the AURORA standard API, therefore the SW is integrated into a wrapper that implements the API for getting the services provided by the algorithms of the model-based design GNC and reacting to its outputs (CBI component model). The TASTE/QGen tool suite is used to compile, link and execute the components software.

  The TASTE/QGen tool suite is used to compile, link and execute the software.

- Platform-In-the-Loop

  To validate the SW component running in the execution platform connected to an open-loop-environment, typically using an Avionics Test Bench (ATB) equipment.

The process is iterative, and any error or change is done at model-level only and implies to iterate previous In-the-loop steps.

4. Flight SW Autocoding Life-cycle Process

The space SW generation procedure has traditionally relayed on a linear approach based on manual coding of the SW functions, which departs from the requirements coming from the top-level system, which are derived into SW requirements. From them, a SW architecture is defined, and further requirement levels might be derived. Then, an implementation procedure follows, which is lately checked and verified at the different levels, from unit to integrated architecture, in different facilities. Moreover, the SW is checked for readiness, correctness, maintainability, trying to detect implementation errors beyond those that can be detected by test. This process is tedious and implies a big number of resources.

For AOCS/GNC, the main use case included in AURORA, this traditional process was composed of two parallel workflows with different stages:

- Matlab/Simulink Models:

  This workflow relies on the implementation of Simulink models to define the GNC algorithm for the SC. It consists of the following steps:

  - Definition of requirements, which is common to the other workflow. Departing from the system requirements some requirements are derived to the GNC algorithms.
  - Model prototyping, developing the basic GNC algorithms to cover the mission/system needs. This covers the preliminary design.
  - Model detailed design. This includes the refinement of the models and the formal verification campaign using a representative simulator. This stage finishes the model workflow.

- Manual SW implementation:
  - Definition of requirements, which is common to the other workflow. Departing from the system requirements some requirements are derived to the SW requirements.
  - From the algorithm implementation in the preliminary design phase, the SW requirements are refined to include compatibility with the outlined design.
  - Based on the SW requirements, the manual part of the SW not depending on the GNC algorithms is implemented. Once the preliminary design is over, a first GNC coding is performed and integrated and tested together with the other SW part.
  - After the detailed design phase, the SW is refined introducing some updates and the details coming from GNC algorithms. A SW validation campaign is performed in a representative simulation environment.
  - Then the generated SW is integrated within the system facilities and an extensive verification campaign is run (SIL, PIL, HIL).

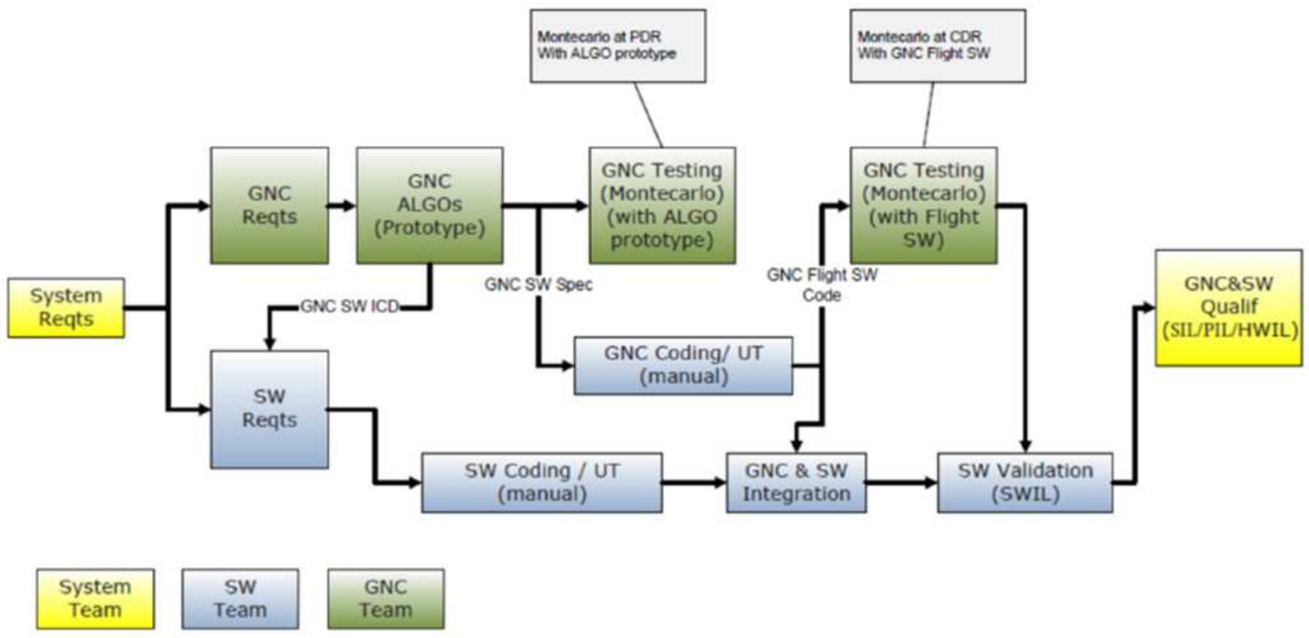

Figure 1 Traditional GNC SW development with manual coding (from [RD4])

This traditional workflow normally takes large implementation times, is prone to human errors which are difficult to track and debug and it is therefore more expensive and less reliable.

An alternative to use this manual based process, relies on autocoding techniques applied to models, in a model-based approach targeting a simplified and more reliable procedure, reducing the implementation times, the number of errors and increasing maintainability, readiness and comprehensiveness.

For AOCS/GNC the use of this model-based approach is the natural evolution of the abovementioned manual procedure, since the models have been already used in the past and can be used as baseline architecture and SW implementation, by using the appropriate autocoding conversion tool.

This document gathers the different processes and stages of this model-based process which cover the whole SW life-cycle from requirements to qualification.

The Table 4: Test facilities definition summarizes the main stages and facilities of AOCS/GNC validation.

Verification Stage	Facility	Comment
MIL	**FES** **Functional Engineering Simulator**	Model of the GNC algorithms implemented in a simulation framework (Matlab/Simulink)
SIL	**FES** **Functional Engineering Simulator**	Software produced from model is connected to a spacecraft simulator to demonstrate that software is still requirement compliant
PIL	**SW Test Bench**	SW is executed on a real OBC, which is connected to a Real Time Simulator (RTS). This stage is done to verify computing budget usage
SVF	**SW Validation Facility**	The AOCS/GNC software is executed with the whole on-board software into a model of the OBC

Verification Stage	Facility	Comment
HWIL	FUMO (Functional model) ATB (Avionics Test Bench) PFM (Proto Flight Model)	Final on-board software is run with some real avionics equipment with some spacecraft simulator, which closes the loop

Table 4: Test facilities definition

The complete software development cycle is presented in Figure 2: Autocoding vs Manual SW development cycle, where the different milestones and documents to be reported are listed at every milestone.

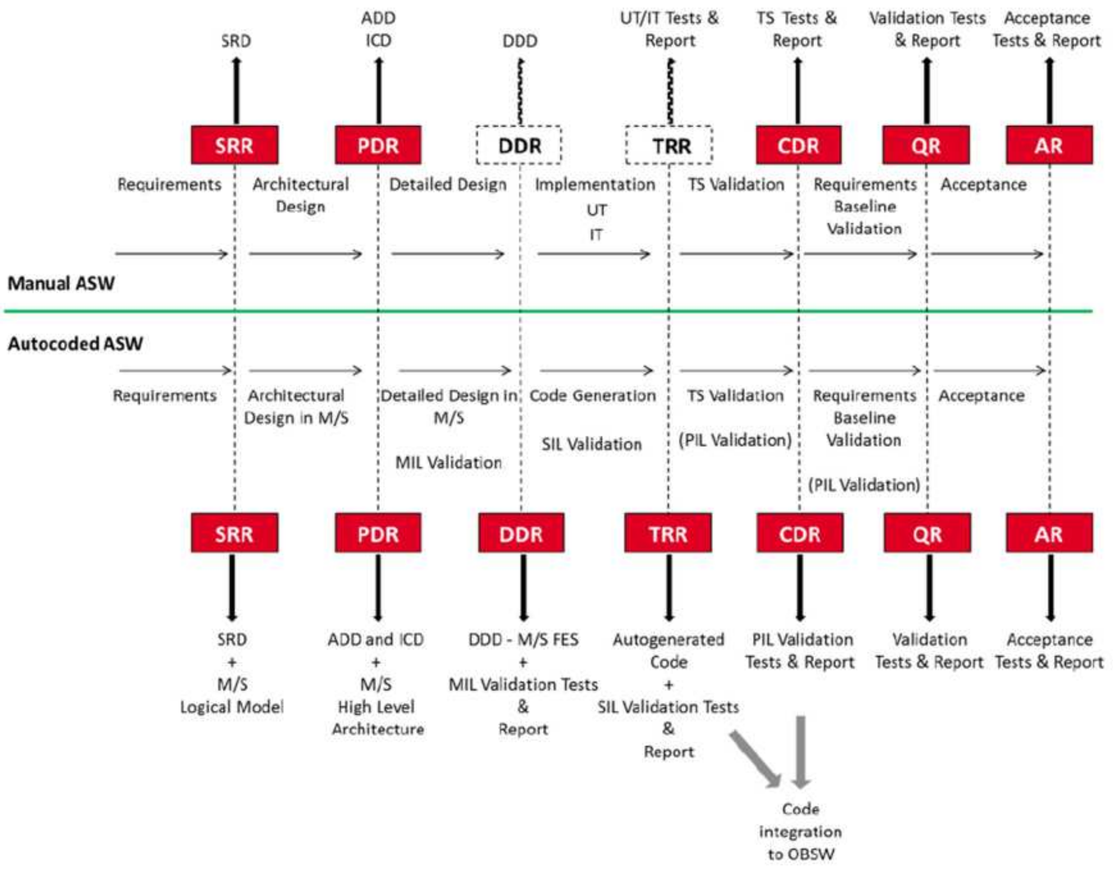

Figure 2: Autocoding vs Manual SW development cycle

5. Model-in-the-Loop Stage

This stage is focused on the generation of Matlab/Simulink models, compliant with requirements and early validated with tests that will result in the generation of an automatically generated code with QGen, representative of the original model.

This stage is quite similar with the traditional approach of manual code generation, however some difference in the process is observed due to the earlier availability of the AOCS/GNC SW. At the beginning of the process the following documents shall be prepared:

- ICD: joint work shared between the GNC and SW team in which the data flow and frequency required by the GNC specification is taken into account. Definition of the code generator settings are defined in this document. The GNC engineer is no longer blind to the software side of the process and shall have some insight on the final autogenerated code.
- Model Requirements Specification: document used to design and implement the GNC algorithms based on the GNC requirements specification.

Once that those documents are issued for the SRR milestone (System Requirements Review), the Model- in-the-Loop begins and the generation of the models can start. In this step, the GNC engineer is being supported by a Modelling Guideline Handbook, which gathers industry modelling standards that are recommended to follow for a later easy integration and model maintainability. For a generic Simulink guideline for autocoding model generation, please refer to [RD5] and [RD6].

For Aurora's scope, a custom set of guidelines was generated ([RD5]). These new guidelines are Euclid heritage and were modified to account for QGen limitations i.e., limitations in terms of Simulink block constraints for instance.

The resulting Simulink will apply the algorithms specified in the Model Requirements Specification. In parallel, models representative of the real word, such as DKE models, sensors or actuators shall be developed and ready for performance test.

These model algorithms are then subjected to testing in order to ensure compliance against mission requirements, to identify bugs and to ensure sufficient model coverage. Note that model coverage is not the same as code coverage. Nonetheless, typically, large model coverage implies large code coverage, something to be seek in later stages of software validation. Two different test scenarios are defined:

- Unitary Integration Test of the individual models
- Verification of the AOCS/GNC performance requirements on a validated FES with representative test cases. This campaign typically includes a full Monte Carlo campaign.

5.1. Unitary Integration Test

Testing starts at unitary level, where Unitary Integration Tests are defined by the GNC engineer. This UIT are developed to cover all the functionalities implemented in each function, to test boundaries and to verify requirements allocated to unitary level. These tests can be considered as the classical bottom-up approach in which a set of pre-defined inputs are fed to the model in open-loop simulations.

For each AOCS mode, the UIT campaign will start with the deeper models (leaf models), which are hierarchy tested in the first place. These leaf models are isolated from the rest of the models. Once that the model has been properly tested and its behavior has been properly assessed, the process continues with upper levels, aggregating the previously tested models. Following this procedure, if a top model test fails, it can be safe to assume that the lower models do correctly behave.

The typical procedure of generating the UIT is via test harness, in which the model to test is placed into a model reference block where inputs are fed, and outputs are collected for a final PASS/FAIL evaluation according to the test specification. I/O signals shall be collected for later verification campaigns (SIL/PIL) as those will be used as a confirmation that the autogenerated code behaves as models, that is, same inputs results in same outputs.

In this sense, the AOCS models are used as Technical Specification for the auto generated code as the software behavior is validated against model behavior. Library functions used in the model development shall also be unitary tested.

Inputs can be fed via Simulink Signal builder, which allows for easy change of inputs signals and creation of various signal groups. This method allows for an effective and easy execution of the unitary test with easy signal replacement without effectively changing the test harness.

Collection of the I/O can for each test case is typically done via the Signal Logging capabilities of Simulink, where data is automatically stored as a Simulink Dataset variable, although the user is free to choose the most suitable signal save option for their need.

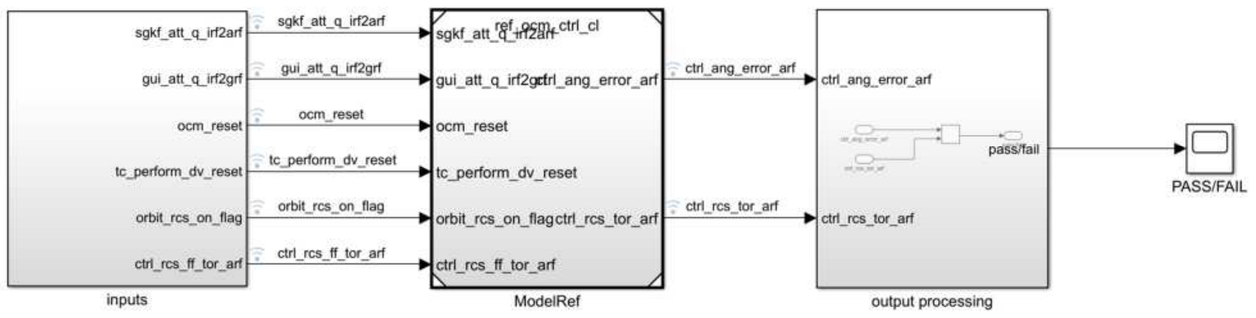

Figure 3: UIT Model harness

Steps

a) Generate UIT specification, defining what inputs and outputs are expected
b) Generate the test harness with the following components
    a. Input block
    b. Model reference block
    c. Output block with PASS/FAIL criteria
c) Run MIL UIT to validate correct test implementation
    a. In case of FAIL, review test case implementation and repeat the MIL execution test
d) Gather I/O signals for MIL-SIL comparison
e) Report results obtained in the corresponding section of the Test Report

In conclusion, UIT are open-loop test cases defined for an early verification of the GNC algorithms and requirements at unitary level with the addition of a preliminary model coverage. Once the model's behavior has been tested at unitary level, then, they can be included inside a simulation architecture for requirements verification.

5.2. Performance cases in FES

A Functional Engineering Simulator is a simulation environment whose purpose is the verification of the AOCS/GNC models. This simulator is in charge of managing the different test and mission scenarios specified, being also a direct support of the software development.

The main components of a FES architecture are:

- *Simulation Engine*: responsible of the definition of a simulation scenario, definition of the mission and the configuration of the models to be simulated. Parameters are configured and pre-processed to obtain simulation parameters, which are set in the mask parameters.
- *Simulation Core*: Simulink templates that is customized for each operation mode. Different Simulink libraries containing the GNC algorithms are present so that the template can replace the adequate models.
- *Monte Carlo Simulation*: functions that manage the configuration and control of Monte Carlo simulations, generating perturbed values of the model parameters and controlling the storage of the raw data
- *Post-processing*: functions to post-process the raw data obtained. This component typically generates representative plots and graphs needed for AOCS/GNC validation.
- *Failure injection*: component in charge of injecting failures in the simulation to check failure conditions or FDIR algorithms. Typical failure comprises of freeze signal, set a signal to a desired value or linear signal behaviour.

It is important to remark that the FES itself must be validated according to a Software Verification and Validation Plan, which complies with the ECSS-E-40 standard.

Unlike UIT, test cases are run in closed loop, including, not only the GNC models generated and unitary tested, but the real word representative models (DKE, sensors and actuators), which were previously validated to ensure good overall performance.

Test cases in a FES are no longer defined as a set of inputs, but as a timeline file that it is read by the simulation engine. This timeline defines the set of initial conditions and the operational timeline, which defines the set of commands to be followed. This timeline is typically defined as an external file, XML file for instance, however, this file is simulator dependent.

Test cases are defined in order to verify that the system is compliant with the requirements specified in the SRR. These tests may include single shots runs with simulator parameters adjusted for adequate testing or Monte Carlo simulations, with the perturbation of relevant parameters.

Once that the results of the test have been formally verified, reported and accepted in Test Reports, the PDR closes this stage.

5.3. Code Generation

Code generation will be further discussed in the next sections as this process belongs to the SIL campaign, nonetheless it is close related to the model development, so a brief insight is presented here.

After running the complete MIL campaign verification, the code generation process starts. Autocoding tools such as Simulink Coder toolbox or QGen can be used to translate the model architecture into C code software files, which can be embedded into a software testing facility for the SIL campaign.

The proposed approach here is that the autogenerated code **shall not** be manually modified at any level. In case of some bugs identified during the software verification process that require correction, the solution shall be applied to the model, being the autocode process regenerated. This is done to ensure that the models and the code generated from them are always align and the AOCS team and Software team can maintain their own process with no major differences. An assessment of the tests to be repeated is done to ensure that the modifications to the models do not imply fail tests.

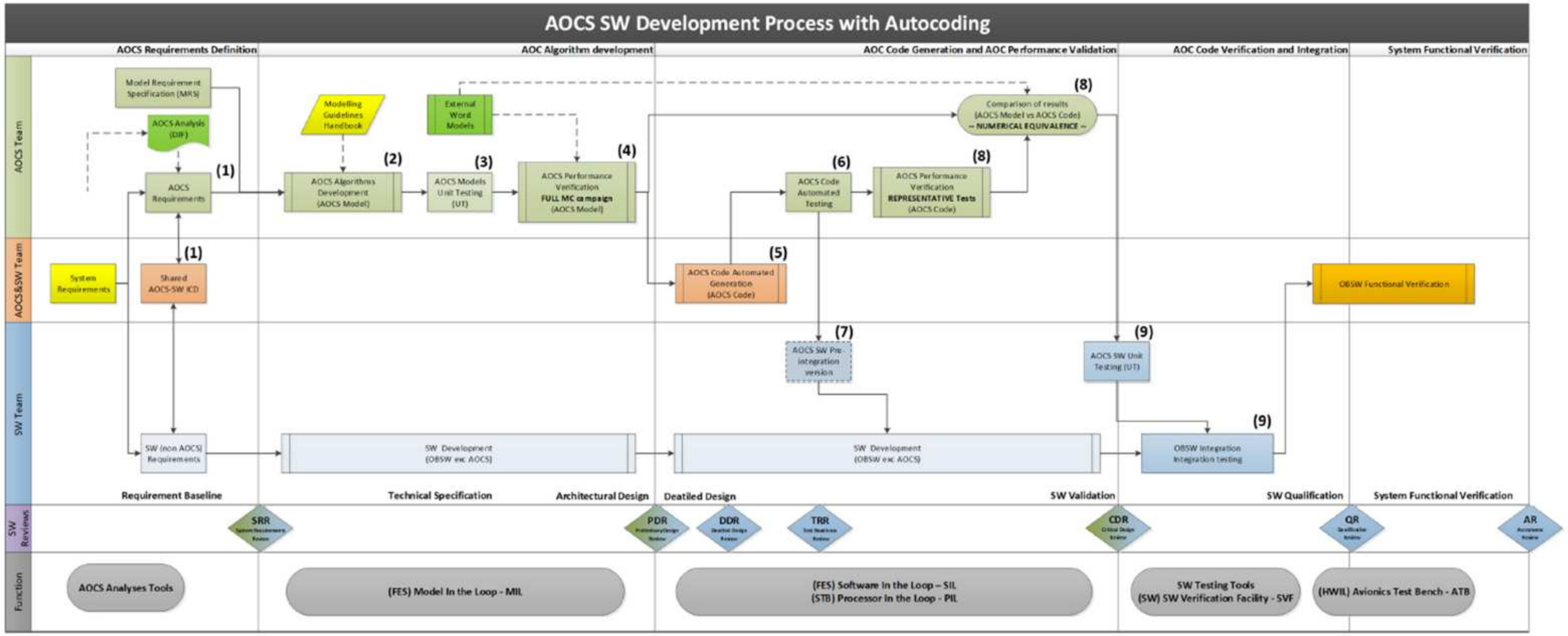
AOCS SW Development Process with Autocoding
AOCS Requirements Definition
AOC Algorithm development
AOC Code Generation and AOC Performance Validation
AOC Code Verification and Integration
System Functional Verification
AOCS Team
AOCS&SW Team
SW Team
SW Reviews
Function
Model Requirement Specification (MRS)
AOCS Analysis (DIF)
AOCS Requirements
(1)
Modelling Guidelines Handbook
AOCS Algorithms Development (AOCS Model)
(2)
External Word Models
AOCS Models Unit Testing (UT)
(3)
AOCS Performance Verification FULL MC campaign (AOCS Model)
(4)
Comparison of results (AOCS Model vs AOCS Code) – NUMERICAL EQUIVALENCE –
(8)
AOCS Code Automated Testing
(6)
AOCS Performance Verification REPRESENTATIVE Tests (AOCS Code)
(8)
System Requirements
Shared AOCS-SW ICD
(1)
AOCS Code Automated Generation (AOCS Code)
(5)
OBSW Functional Verification
AOCS SW Pre-integration version
(7)
AOCS SW Unit Testing (UT)
(9)
SW (non AOCS) Requirements
SW Development (OBSW exc AOCS)
SW Development (OBSW exc AOCS)
OBSW Integration Integration testing
(9)
Requirement Baseline
Technical Specification
Architectural Design
Deatiled Design
SW Validation
SW Qualification
System Functional Verification
SRR
PDR
DDR
TRR
CDR
QR
AR
AOCS Analyses Tools
(FES) Model In the Loop - MIL
(FES) Software In the Loop – SIL
(STB) Processor In the Loop - PIL
SW Testing Tools
(SW) SW Verification Facility - SVF
(HWIL) Avionics Test Bench - ATB

Figure 4 ESA proposed development life-cycle for AOCS/GNC SW (from [RD4])

6. Software-in-the-Loop Stage

This section is an output of the Task 4.2 Definition of the Software-In-the-Loop

To be written in future document issues.

7. Processor-in-the-Loop and Hardware-in-the-loop Stage

This section is an output of the Task 4.3 Definition of the Component-In-the-Loop Task 4.2 Definition of the Software-In-the-Loop

To be written in future document issues.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101004291