Orbital CS
Carretera de Artica 29, 3ª Planta
31013 ARTICA (Navarra)
Tel. +34 848 47 31 81

FEASIBILITY ANALYSIS OF INNOVATIVE PRACTICES IN VIRTUAL TESTING METHODS FOR AIRCRAFT CERTIFICATION

Project: FAVIT

Grant Agreement nº: 864475

D01.a. Gap Analysis

DUE DATE OF DELIVERABLE: 23/10/2020

ACTUAL SUBMISSION DATE: 23/10/2020

START DATE OF THE PROJECT: 01/10/2019

PROJECT DURATION: 30 months

DISSEMINATION LEVEL: PU (Public)

This project has received funding from the Clean Sky 2 Joint Undertaking (JU) under grant agreement No 864475.
The JU receives support from the European Union's Horizon 2020 research and innovation programme and the Clean Sky 2 JU members other than the Union.

Document Change Log

Iss./Rev.	Date	Section / Page	Change Description
D01.a_v1	16/10/2020	All	Version submitted

This project has received funding from the Clean Sky 2 Joint Undertaking (JU) under grant agreement No 864475.
The JU receives support from the European Union's Horizon 2020 research and innovation programme and the Clean Sky 2 JU members other than the Union.

Contents

LIST OF TABLES

EXECUTIVE SUMMARY

FAVIT's main objective is to deliver a set of knowledge-based proposals for the improvement of aerospace standards and guidelines for the system suppliers and aircraft manufacturers. FAVIT will analyse the current aerospace standards and guidelines to identify how the design and verification processes can be enhanced to accelerate the processes using the state-of-the-art technologies based in virtual testing.

The purpose of this deliverable is to present an analysis of the existing standards used in the certification of airborne software systems identifying any gaps in those standards that would otherwise allow the use of Virtual Testing Technologies (VTTs) to better contribute towards the certification of airborne software systems.

This document does not seek to replace target platform testing as a means of final software verification. Rather it will aim to suggest why, according to the current standards, VTTs are not being used to fully certify such software; and how those standards might also be enhanced to more readily facilitate the use of VTTs in the certification of an airborne software system.

The motivation behind this activity is to reduce the need for large-duration, exhaustive testing on test rigs that are fully representative of the final aircraft avionics system. Such test rigs are sometimes known as the "aircraft zero" (AC/0). They are expensive to assemble and maintain and are typically in high demand by various partners of the overall aircraft project.

VTTs can provide a cheaper and more accessible testing environment that can be used throughout the software development process to perform the bulk of functional testing, error finding and debugging. The ultimate goal being that a final re-run of tests on an AC/0 rig would serve more to confirm that the software operates correctly rather than to reveal and debug any possible errors. Thus alleviating the demand on AC/0 which should have cost and time benefits for the project as a whole.

The major conclusion of this analysis is that some formal documentation is needed that deals specifically with different test environments (different VTT methods).

This would include a guideline for how to use them in certifiable product development processes and a means of categorizing such test environments so that they can be more appropriately applied to different assurance levels of the product under test.

Another conclusion is that the existing standards have their fair share of ambiguities when it comes to what test environment can currently be used in the gathering of certification evidences. This may well explain why AC/0 type testing is still so heavily relied upon in the industry.

REFERENCED DOCUMENTS

[SAE-ARP 4754]	*Certification Considerations for Highly-Integrated or Complex Aircraft Systems, November 1996*
[RTCA DO-254]	*Design Assurance Guidelines for Airborne Electronic Hardware, April 2009*
[RTCA DO-178/C]	*Software Considerations in Airborne Systems and Equipment Certification, December 2011*
[D100.3.6.1_a]	*Virtual Testing Methods Definition, December 2017*
[D100.3.2.1_e]	*Documentation to the Technology Readiness Level 5 Demonstration of the Software Core Environment for MISSION Framework, March 2018*

1 INTRODUCTION

Section ***2*** of this document shall first present an overview of the following:

- Airborne software systems
  Defining terms and concepts that shall be referenced throughout the subsequent contents.
- Summary of Certification Standards
  Identifying their scopes and how they are typically applied within the industry
- Virtual Testing Technologies
  Definitions of the different types of VTTs identifying a typical application of VTTs applied to a product lifecycle

Section ***3*** shall be a more in-depth analysis of the certification standards to identify points of interest with respect to VTTs and where there are possible gaps in the standards that might be addressed to better promote the use of VTTs. Naturally, the majority of these document analyses will be concentrated on the testing aspects therein.

Finally section ***4*** shall summarize the findings of this analysis and provide some recommendations on how the current standards may be enhanced to better facilitate the use of VTTs.

Throughout this document, references to other sections shall take the form:

- ***Section x.y.z*** – Meaning a reference to the external document under discussion
- ***Section x.y.z*** – Meaning a reference to this document

2 OVERVIEW

2.1 AIRBORNE SOFTWARE SYSTEMS

Airborne software systems typically contain one or more computing platforms known as Line-Replaceable Units (LRUs). Each LRU can be a hardware-only device or can contain one or more software programs that execute to control or monitor various avionics systems or provide information to the pilot or flight recording/fault management subsystems. Each LRU or group of related LRUs can be thought of as a subsystem.

In modern, large scale aircraft, there would be many different LRUs. Some of which would be executing more than one independent software program. All such subsystems would be interconnected to each other and various hardware devices via one or more physical networks. These networks could be serial-based (e.g. RS422), Ethernet-based (e.g. AFDX) or other networks more specific to the connecting equipment (e.g. MIL-STD 1553B).

Such multi-LRU systems would typically be a collaboration of several industrial partners, each of which would provide hardware, software or both. Each of these providers is usually responsible for the certification of the subsystem that they are providing. One of these partners (usually the aircraft manufacturer) would act as an overall aircraft integrator and be responsible for testing the entire avionics system as a whole on an aircraft zero (AC/0) test rig and the real aircraft.

2.1.1 Testing

This document is focusing on the testing activities of a product lifecycle. By way of an example, the typical testing process for a whole system and its constituent subsystems would be performed in various phases such as:

Phase	Test Activity	Example	Test environment	Responsible Party
Phases 1 to 4 would be repeated for each subsystem				
1	Individual subsystem engineering testing	Landing gear control unit OR flight management system	Target LRU and supporting, simulated environment	Subsystem provider
2	Multiple inter-dependent subsystem engineering testing	Landing gear control unit AND Flight management system	Multiple LRUs and supporting simulated environment OR **AC/0**	Subsystem providers
3	Individual subsystem formal testing	Landing gear control unit OR flight management system	**AC/0**	Subsystem provider
4	Multiple, inter-dependent subsystem formal testing	Landing gear control unit AND Flight management system	**AC/0**	Subsystem provider
5	Aircraft system engineering testing	All subsystems	**AC/0**	Aircraft integrator
6	Aircraft system formal testing	All subsystems	**AC/0**	Aircraft integrator
7	Aircraft system ground testing	All subsystems	Real aircraft, on ground	Aircraft integrator
8	Aircraft system flight testing	All subsystems	Real aircraft, in flight	Aircraft integrator

Table 1 – System/Subsystem Testing Phases

In the above example it can be seen just how important an **AC/0** test environment is. Such an environment can be expensive to assemble and maintain and is typically only found within the aircraft supplier's installations, so it is also not readily accessible by all partners of a project.

Phases **4** and **6** must be performed on a test rig that contains the real target hardware (i.e. aircraft zero). However, phases **1**, **2**, **3** and **5** could be tested in a virtual testing environment. Hence the objective of this analysis.

Phases **7** and **8** are the final testing activities before an aircraft can enter into service. Before performing these tests all subsystems will have been tested and verified as being functionally correct and robust such that these phases are beyond the scope of this document.

In this document, the term “validation” of system or software requirements means that those requirements are checked for correctness, completeness and consistency (with each other).

The term “verification” of system or software requirements means the process of testing a system or subsystem to verify that it is functionally correct (according to its corresponding requirements) and robust.

2.1.2 Integrated Modular Avionics (IMA)

In recent years, IMA systems have become increasingly common-place. Such systems adhere to internationally accepted standards like ARINC-653 and ASAAC STANAG-4626.

In IMA systems, each LRU is considered to be a module. Each module could be hardware only or could host one or more software partitions. Each partition uses an established API to interact with other partitions and other LRUs via dedicated communication channels. The communication channels and the software partitions for all LRUs (hence the whole system) are scheduled to operate in dedicated, pre-determined execution windows.

A partition may also act as a “signal concentrator” to access and re-distribute hardware signals. For example, an IMA system may have a dedicated partition to handle all serial I/O data. If another partition needs to use this data, it would perform its own serial I/O via that “serial partition” and not directly with the hardware.

Each software partition comprises one or more executable processes and if a partition fails it would have no impact on the execution of other partitions of the same LRU.

It is common practice that a single CSCI has a direct mapping to one software partition but it could be distributed across several partitions. For the purposes of this document a CSCI shall be considered as being implemented as a single IMA software partition. This document shall also be focusing on using VTTs in the context of an IMA system.

One point of interest for this document is that the application software for an IMA partition could and *should* be designed and implemented so that it depends only on the software API (e.g. ARINC 653) for the IMA system. This means that the actual application source code can easily be ported to a host PC environment for functional testing, so long as the API and the associated behaviour of an IMA OS can also be hosted on that PC (*see section* ***Software In-The Loop (SIL)***).

2.2 CERTIFICATION STANDARDS

Several standards documents exist within the aerospace industry. This document shall concentrate on the following widely-used standards.

The focus of this document is on the analyses of these standards rather than presenting their contents. ***[D100.3.2.1_e]*** discusses these standards in slightly more detail but it is assumed that the reader has access to these documents.

2.2.1 SAE ARP-4754

"Certification Considerations for Highly-Integrated or Complex Aircraft Systems."
The document defines "highly-integrated" systems as being those that contribute to multiple aircraft-level functions. "Complex" systems are defined as those whose safety cannot be shown by testing alone (analysis tools would also be needed).

It is a top-level guide to the development lifecycle for the certification of whole aircraft avionics systems. It delegates more specific details of the development process for the certification of hardware systems and software systems to the documents ***[RTCA DO-254]*** and ***[RTCA DO-178/C]*** respectively.

The document introduces the concept of Item Development Assurance Levels (IDALs). These identify the criticality of a failure of a software or hardware function. Level A being the most catastrophic and level E having no safety impact.
The consequence being that functions with a higher IDAL must be subjected to more rigorous testing and certification criteria.

The document would typically be used by systems engineering as input to the system development process.

2.2.2 RTCA DO-254

"Design Assurance Guidelines for Airborne Electronic Hardware."
The document considers aspects of electronic hardware development for the certification of airborne systems.

It starts from a system design process that allocates system functionality to hardware components and assigns associated Design Assurance Levels (DALs) to those components (as defined by the IDALs in ***[SAE ARP-4754])***.

The document is typically used during the hardware development process to define project standards/processes and aid in the specification of hardware requirements, design, and test cases.

2.2.3 RTCA DO-178/C

"Software Considerations in Airborne Systems and Equipment Certification"

The document considers aspects of software development for the certification of airborne systems.

It presents 3 top-level lifecycle processes: System, Hardware, and Software. Only the software process is elaborated upon in great detail in the document.

The system process provides inputs to the hardware and software processes.

The software process also takes inputs from the hardware process and provides feedback to both the system and hardware process in an iterative development manner.

The document does not necessarily focus on a single CSCI but in practice each partner of a project would typically apply ***[RTCA DO-178/C]*** independently to a single CSCI rather than the system as a whole. The document considers all phases of the software development lifecycle, only some of which shall be discussed in this document.

The IDALs defined in ***[SAE ARP-4754]*** are re-used here under the name of “software levels”. These are commonly referred to in the industry as DALs.

This document is typically used during the software development process to define project standards/processes and aid in the specification of software requirements, design, and test cases.

2.3 VIRTUAL TESTING TECHNOLOGIES

There are 4 methodologies of VTT which are briefly presented below. *(For a more detailed explanation of VTTs see **[D100.3.2.1_e]**).*

The basic idea of all VTTs is to simulate or model the real aircraft environment in various levels of detail and complexity depending on the system/subsystem that is to be tested in that environment.

Some VTTs would include real LRUs and others would have no real hardware involved.

The major benefits of VTTs are accessibility and cost savings. While the developers of a particular subsystem may not have access to the necessary LRUs or a complete AC/0 test rig, (indeed such a test rig may not even exist in the earlier stages of a project) they can simulate those LRUs which interact with their own subsystem in order to test and debug that subsystem.

The LRU that would host a developers CSCI(s) can even be simulated if the hardware is not readily available. This means that VTTs enable cost savings by frontloading test activities to earlier development stages, thus finding potential defects sooner without consuming valuable time on the AC/0 test rig.

In the wider application of FAVIT to other projects, it is decided that system-level virtual testing is to be mapped to the system process of ***[SAE-ARP 4754A]***. Hardware and software level virtual testing are to be mapped to the development processes of ***[RTCA DO-254]*** and ***[RTCA DO-178/C]*** respectively.

For the testing of software CSCIs, an IMA-based application has particularly great potential for testing by VTTs because the core functionality of the software is isolated from any hardware dependencies by means of an abstraction layer (e.g. ARINC-653 APEX layer).

For the purposes of this document any particular VTT described below can be thought of as a type of test environment.

2.3.1 Model In-The Loop (MIL)

This involves the use of COTS tools that can model both a simulated environment (inputs, outputs, timings), called a *plant* and control algorithms that implement functional requirements of the LRUs under test (*controllers*). Usually the plant and controllers run on the same host PC.

One advantage of MIL virtual testing is the early verification of requirements that specify any control algorithms. It is also commonplace that the tool used to model the controllers is used to generate source code from the models that will be re-used in SIL testing or the real LRU.

2.3.2 Software In-The Loop (SIL)

Automatically-generated source code from MIL testing or even manually developed source code ***** can be used in a SIL virtual test environment.
All software would execute on host PCs so there would be no LRUs in the environment but it could well be possible to have at least some parts of the avionics network replicated as per AC/0 (e.g. an AFDX network can be used to connect several PCs).

A SIL test environment could also simulate a LRU's operating system. For IMA-based systems this could be used to test inter-partition dependencies and the allocation of the operating systems resources to each partition.

Since a PC is used for SIL testing, it is most probable that any CSCIs under test would be compiled to execute on a CPU which is not the same as that used in the real LRU for that CSCI. This brings into question the value of complete functional testing which would have to be repeated using a different binary image. It would of course be useful for informal, development testing. However, it could be argued that the testing of various inter-CSCI and inter-LRU interfaces has more value in a SIL environment.

Due to the nature of a hardware product, it is unlikely that SIL testing would be very useful as part of the hardware verification process. It may, however, be a useful tool for simulating external components of a system or possibly using a software program to simulate some aspects of the hardware product's behaviour as part of the requirements capture process.

**** Such source code can ultimately be part of the real CSCI so long as it is hardware independent***

2.3.3 Virtual Processor In-The-Loop (VPIL)

When the real hardware is not available a host PC can run a CPU simulator within which a CSCI can execute. This simulation would include hardware models that mimic certain characteristics of the target hardware's CPU and resources.
Such simulators are commonly provided with a source code compiler tool and are typically used by a software developer to gain confidence in the functional correctness of the software before testing in a Hardware In-The-Loop (HIL) environment.

VPIL virtual testing is typically used for debugging or informal testing of CSCIs as part of the software development process.

Since host PCs are used for both SIL and VPIL testing methods, it would be simple to introduce a VPIL component into an existing SIL environment.

2.3.4 Hardware In-The-Loop (HIL)

The VTT method includes the actual LRUs that are to be tested for a subsystem. All other LRUs which communicate with the subsystem are simulated in some manner. This could be as simple as a host PC sending and receiving data streams to one of the LRUs under test or there could be a more complex software simulation of the behaviour of another LRU or the real-world environment (e.g. a flight simulator to stimulate various phases of flight and aircraft movement). Additional equipment can be added to the test environment when testing specific hardware characteristics of the LRU (e.g. power outputs, signal frequencies).

HIL is commonly used by the developer of a CSCI to perform high-level requirements functional testing. An LRU can also be used to perform low-level requirements testing and code coverage analysis if integrated with a suitable testing tool.

A debug version of a CSCI can also be loaded onto a real LRU and together with a COTS debugger, the source code can be debugged whilst running on the real hardware.

At system or subsystem level, HIL virtual testing as it is currently used corresponds to testing phases **1** and **2** in ***Table 1 – System/Subsystem Testing Phases***. At a wider system level, HIL can also be used to test aspects of an aircraft communication network if more than one LRU is included in the VTT,

2.3.5 Summary

For the purposes of this document, it is useful to identify at what phase of a development lifecycle the different VTTs could be used and why. The following table summarises this:

Phase	Description	Example	VTT Environment	Motivation
1	High-Level Requirements elaboration	Prototyping of algorithms and state machines. Done in parallel with	MIL	Facilitate the specification of the requirements

		requirements elaboration.		
2	High-Level Requirements validation	Testing of HLRs and elaboration of HLR test cases	MIL	Validate that the HLRs are correct before the design phase begins
3	Software Design elaboration	Prototyping of software code and associated debugging	SIL VPIL	Verify auto-generated code from MIL tools and/or provide feedback to the design
4	Software Design verification (informal)	Integration testing of the software components and associated debugging	SIL HIL	To check the software architecture and Low-Level Requirements implementation and internal interfaces of the CSCI
5	Subsystem Integration Testing	Integration testing of a CSCI/LRU with other components of the same subsystem	SIL HIL	To check external interfaces of the CSCI/LRU with simulations of other components of the subsystem
6	High-Level Requirements verification (informal)	Performing High-Level Test Cases on software CSCIs	SIL HIL	Informal verification of the software to find and eliminate errors
7	Low-Level Requirements verification (informal)	Performing Low-Level Tests on software CSCIs	VPIL HIL	Informal verification of LLRs and source code coverage analysis using COTS tools with the software running on a simulated processor or LRU to find and eliminate errors

Table 2 - VTTs and Development Lifecycle

As can be seen in the above table, the current application of VTTs does not include any *formal* testing activities. That is to say, none of the test results would be used to compile evidence for certification of the product, which is the typical approach taken to certification at present.

3 ANALYSES OF CERTIFICATION STANDARDS

Given the scope of this document the bulk of the analysis (in some cases all the analysis) of the current standards will be concentrated on testing or verification activities. Presented below are the analyses of the three standards documents one-by-one.

3.1 SAE ARP-4754

3.1.1 Requirements Validation

Section 7 of the document is concerned with the validation of requirements. In broad terms it involves checking or testing of requirements to confirm that they are correct. This is not the same as verification which tests that the implementation of those requirements is correct.

It can be the validation of system, hardware or software requirements at differing stages of the development lifecycle.

Of particular interest to this document is section ***7.6.1 Validation Methods***. Two of the recommended methods are *Modelling* and *Test* and for IDALs A, B and C they are amongst those methods recommended to be used as part of the certification activities (see ***[SAE ARP-4754]:*** ***TABLE 6 - Requirements Validation Methods and Data***).

The modelling validation method is quite clearly applicable to MIL virtual testing, however the document does not go into any detail about this validation method whatsoever. Nor does it discuss how any outputs of using this verification method may feed into the rest of the development lifecycle (e.g. auto-generated source code from the requirements models).

ARP-4754: GAP001

More detail about the modelling validation method would be useful.

The description of the testing validation method does have slightly more detail and mentions such practices as simulations and prototyping whilst ensuring that those methods and the associated environment is sufficiently representative of the real system (see ***4.2.1***). The term "sufficiently representative", and how this could be demonstrated, however, is not clearly defined.

3.1.2 Implementation Verification

This is covered in section 8 of the document. It does state that the level of verification needed is proportional to the IDAL of the system or item under test. Something that will be seen to be echoed in the sub document for software certification.

Under the ***Verification Methods*** section (***8.4***) modelling is mentioned as part of the ***Analysis*** methodology (section ***8.4.2***). As for the validation section, no outputs of model-based testing are discussed but it does (somewhat ambiguously) mention that modelling may be used for "other purposes".

ARP-4754: GAP002

Elaborate upon the "other purposes" for which model-based testing may be used and discuss how such things as auto-generated code can feed into the lifecycle.

Section 8.4.3 is concerned with testing as a verification method. The section does not make any mention of such concepts as "target platform" or any test environment in particular. It states that testing is done on an "item" or a "system" which can only be assumed to mean the real target environment.

However, there is one mention to modelling:

> *Tests are performed on all or part of the physical system or item **or an appropriate validated model** using procedures documented in sufficient detail so that a second party could reproduce the test results.*

This can be inferred to mean that MIL virtual testing is an acceptable means of verification, but why no mention of SIL, VPIL or HIL VTTs?

ARP-4754: GAP003

No mention of other virtual testing methods, only the use of models.

In this section there is also a table of verification activities against IDALs. ***TABLE 7 - Verification Methods and Data*** states that for levels A and B testing must be done plus at least one other method (e.g. review).

3.2 RTCA DO-254

Section 1.7 introduces the idea of using alternative processes or methods to provide hardware design assurance. Of potential interest to this document are the alternative processes (i.e. a verification process using VTTs).

The section does not mention specific lifecycle activities but it does state that such alternative methods or processes must be justified and demonstrated to be valid. This would take much of the certification responsibility of the product away from the document and into the hands of the user of the document.

In ***section 4.1*** the planning process activities are described. Amongst these is activity ***6.*** Which describes the design environment. Briefly mentioned is the task of verifying the hardware item but this section seems not to be really concerned with a test environment as such.

RTCA DO-254: GAP001

A test environment could be chosen and defined as part of the planning process.

As part of the overall hardware design process, ***sections 5.1*** and ***5.2*** elaborate the requirements capture and conceptual design processes respectively. Missing from both of these sections is the use of modelling to help elaborate either requirements or design concepts. As will be seen in the following subsection, MIL virtual testing can be used to help elaborate the software requirements of a CSCI at an early stage in the lifecycle. Why not use the same VTT method here?

RTCA DO-254: GAP002

No mention of modelling to facilitate requirements or design elaboration.

Requirements validation and verification is covered by ***section 6*** of the document and more specifically, ***section 6.3.1*** discusses the methods to be used for this activity.

No mention of particular test environments is made at all in this section. However the following paragraph is of interest:

> *When it is not feasible to verify specific requirements by exercising the hardware item in its intended operational environment, other verification means should be provided, and justified.*

This would seem to suggest that a VTT environment could be used for verification activities but all of the responsibility for justifying it (and therefore using those test results for certification) falls to the user of the document. In short, there are no guidelines for test environments.

RTCA DO-254: GAP003

Some discussion about test environments and specifically "non-intended operational environments" would be useful here.

Aside from testing, there is also analysis as a necessary verification activity. ***Section 6.3.2*** covers this activity and includes the following text:

> ***Simulation*** *is an important design analysis tool both for visualization of circuit operation and for higher level functional operation.* ***Simulation*** *can be used to analyze the impact of production variations in hardware parameters that would be difficult to do using other verification means and thus build confidence in reduction of design errors affecting safety due to these variations. Since the results depend on the* ***models*** *and scenarios employed, simulation results alone cannot be used for the purpose of certification credit without supporting evidence of their validity.*

The use of simulation in the analysis activity would seem to lend itself to MIL testing with possible SIL components providing auxiliary functions in the test environment. The last sentence, however, seems to imply that any models which are used are *themselves* considered to be simulations and not necessarily representative of the real item under test.

This is a different concept to the use of modelling for software items in which the model is an exact specification of the HLRs and can even be used to auto-generate the actual code.

RTCA DO-254: GAP004

What if models were used to generate HDL for the item?

Analysis by simulation would seem to require particular test environments to be precisely specified. This is a similar observation as ***RTCA DO-254: GAP001***, above.

3.3 RTCA DO-178/C

As previously stated, the focus of this document with respect to RTCA DO-178/C is largely concerned with the testing aspects of that document. However, there are also some points of interest regarding the software integration process which are presented first.

3.3.1 Software Integration Process

Section 5.4 describes this process.

This section contains a glaring contradiction. In ***5.4.1*** it states:

The objective of the integration process is:

a. *The Executable Object Code and its associated Parameter Data Item Files, if any, are produced and **loaded into the target hardware** for hardware/software integration.*

However, the following subsection (***5.4.2***) describes the integration process activities, amongst which are:

*b. Software integration should be performed **on a host computer, a target computer emulator**, or the target computer.*

*c. The software should be loaded **into the target computer** for hardware/software integration.*

So on the one hand, software integration is all about using the target computer, but on the other hand an emulator or host computer can be used. How can emulated target environments or host computers be used when the test objective is for the real target computer?

DO-178/C: GAP001

Why does test activity* b *allow non-target computer platforms which seems to contradict the objective?

3.3.2 Testing

Section 6.4 of the document is concerned with Software Testing.

It highlights 5 ***objectives*** of Software Testing:

a. The Executable Object Code complies with the high-level requirements.

b. The Executable Object Code is robust with the high-level requirements.

c. The Executable Object Code complies with the low-level requirements.

d. The Executable Object Code is robust with the low-level requirements.

e. The Executable Object Code is compatible with the target computer

Points **a.** to **d.** are concerned with testing the functional compliance and robustness of the software which is typically done by high-level testing (through scripts or manual procedures) and low-level testing (via 3rd party testing tools). Only point **e.** is concerned with the actual target hardware.

This would suggest that only point **e.** would need to be executed on the target computer platform. However, the term "*compatible*" is not further defined here and it is open to interpretation as to how to demonstrate compatibility with the target computer.

Section 6.3 also uses the term *compatible* when reviewing High-Level Requirements (HLRs), architecture and Low-Level Requirements (LLRs) to ensure that no requirements conflicts exist with the features of the target environment. It is unclear if this use of the word is also what is meant by point **e**.

DO-178/C: GAP002

Need a more detailed definition of "compatible"? Does it mean that all possible tests have been run on the target, or just a subset specific to that target? Or something else?

DO-178/C also states that there are 3 types of ***testing activities****:*

- **Hardware/software integration testing:**
  To verify correct operation of the software in the target computer environment.
- **Software integration testing:**
  To verify the interrelationships between software requirements and components and to verify the implementation of the software requirements and software components within the software architecture.
- **Low-level testing:**
  To verify the implementation of low-level requirements.

The term "*target computer environment*" would presumably mean the real LRU when considering a single CSCI, but it could be extended to mean *all target computers* in the environment in which *all CSCIs reside*. In other words, AC/0.

DO-178/C: GAP003

When testing a single LRU, does "**target computer environment*****" mean only that LRU or <u>all</u> the real LRUs that communicate with the LRU under test?***

As to the aims of the ***Hardware/Software Integration*** testing activity, it depends on the meaning of "*correct operation of the software*". This could be inferred as being all of the previous objectives **a.** through **e.** or does it just mean that the software does not fail or cause other systems to fail? Thus allowing objectives **a.** through **d.** to have been tested in a different environment?

DO-178/C: GAP004

What is meant by the term "*correct operation of the software*"? How to demonstrate this?

The ***Software Integration*** testing activity does not mention any hardware constraints. It seems that the testing of inter-component software interfaces, logical sequencing of events and any timing dependencies between components would be the main purpose of this testing activity. For a single CSCI with external inputs from other components, these could be provided by simulations or modelling.

Low-Level Testing also makes no mention of the hardware so any of the virtual testing methods could be used.

It is interesting to note that these testing objectives and activities are not directly associated with different DALs of the software under test (see ***3.3.2.2*** below).

3.3.2.1 Test Environment

Section 6.4.1 describes how the test environment should be. It is worth repeating this entire section here because of its direct relevance to the use of VTTs:

> More than one test environment may be needed to satisfy the objectives for software testing. A preferred test environment includes the software loaded into the target computer and tested in an environment that closely resembles the behaviour of the target computer environment.
>
> Note: *In many cases, the requirements-based coverage and structural coverage necessary can be achieved only with more precise control and monitoring of the test inputs and code execution than generally possible in a fully integrated environment. Such testing may need to be performed on a small software component that is functionally isolated from other software components.*
>
> Certification credit may be given for testing done using a target computer emulator or a host computer simulator. Activities related to the test environment include:
>
> a. Selected tests should be performed in the integrated target computer environment, since some errors are only detected in this environment.

The first paragraph states that more than one test environment may be necessary. This implies that not all testing must be done on the target hardware or even that not all testing *can be done* on the target hardware.

Furthermore, the definition of a “preferred testing environment” could be interpreted as either a HIL virtual testing setup or an AC/0 test rig.

The note about an isolated software component could be applied to SIL or HIL virtual testing methods because no mention of target hardware is made.

The final paragraph explicitly concedes that certification evidence may be presented from tests that are *not performed on target hardware*. Specifically, in VPIL and SIL virtual testing environments. However, it offers no guidance on how to do this. It also acknowledges that some tests can only be performed on the target hardware.

Given these statements, it is not apparent why the typical practice in the industry is to run *all tests* on the target hardware, and subsequently present these test results, and only these results, as the certification evidence. These statements could also be interpreted as being contrary to the above-mentioned high-level testing activity “*correct operation of the software in the target computer environment*”.

DO-178/C: GAP005

Cohesion between the test objectives / test activities and the testing environment is missing.

3.3.2.2 Software Levels

Levels A through E assign the safety impact of a failure in a software system. DO-178/C contains tables to describe how the verification of a CSCI should be performed according to the software level assigned to it.

There are several annexed tables in the document, each of which is dedicated to the outputs of a particular development process. Of interest to this document are the tables:

- ***A-3: Verification of Outputs of Software Requirements Process*** (HLRs)
- ***A-4: Verification of Outputs of Software Design Process*** (LLRs)
- ***A-6: Testing of Outputs of Software Integration Process***
  (executable object code, parameter data files, compiling, linking and loading data)

Each of these tables contain the previously-mentioned objective **e** ("*Executable object code is compatible with the target computer"*). For tables ***A-3*** and ***A-4***, this objective is only mandatory for software levels A and B, for table ***A-6*** it is mandatory for levels A through D.

So it can be inferred that for software levels A and B the target computer must be used at least for some HLR and LLR testing but for levels C and D these testing activities could possibly be performed using only VTTs. Although this is not explicitly stated.

Software integration testing must be done on the target computer for levels A through D, according to table ***A-6***, although, somewhat problematically, the software integration process section itself mentions other test environments (see ***3.3.1***).

The tables have other test or verification objectives but none of them mention the test environment.

What is notable by its absence is the omission of any *non-target* computer environments in these tables. Therefore, there is no provision for tailoring the test environment according to software levels for particular test objectives.

DO-178/C: GAP006

No consideration of test environments for software levels.

4 CONCLUSIONS AND RECOMMENDATIONS

4.1 SUMMARY OF IDENTIFIED GAPS

The following tables present summaries of any gaps found during the analyses of the documents.

Reference	Page	Brief
ARP-4754: GAP001 ***More detail about the modelling validation method would be useful.*** The description of the testing validation method does have slightly more detail and mentions such practices as simulations and prototyping whilst ensuring that those methods and the associated environment is sufficiently representative of the real system (see *4.2.1*). The term "sufficiently representative", and how this could be demonstrated, however, is not clearly defined.	19	Need more detail about modelling validation.

4.1.1 Implementation Verification

This is covered in section 8 of the document. It does state that the level of verification needed is proportional to the IDAL of the system or item under test. Something that will be seen to be echoed in the sub document for software certification.

Under the ***Verification Methods*** section (***8.4***) modelling is mentioned as part of the ***Analysis*** methodology (section ***8.4.2***). As for the validation section, no outputs of model-based testing are discussed but it does (somewhat ambiguously) mention that modelling may be used for "other purposes".

ARP-4754: GAP002

Elaborate upon the "other purposes" for which model-based testing may be used and discuss how such things as auto-generated code can

feed into the lifecycle. ***Section 8.4.3*** is concerned with testing as a verification method. The section does not make any mention of such concepts as “target platform” or any test environment in particular. It states that testing is done on an ”item” or a “system” which can only be assumed to mean the real target environment. However, there is one mention to modelling: *Tests are performed on all or part of the physical system or item* ***or an appropriate validated model*** *using procedures documented in sufficient detail so that a second party could reproduce the test results.* This can be inferred to mean that MIL virtual testing is an acceptable means of verification, but why no mention of SIL, VPIL or HIL VTTs? **ARP-4754: GAP003** ***No mention of other virtual testing***		

methods, only the use of models. In this section there is also a table of verification activities against IDALs. ***TABLE 7 - Verification Methods and Data*** states that for levels A and B testing must be done plus at least one other method (e.g. review).		
ARP-4754: GAP002 ***Elaborate upon the "other purposes" for which model-based testing may be used and discuss how such things as auto-generated code can feed into the lifecycle.*** ***Section 8.4.3*** is concerned with testing as a verification method. The section does not make any mention of such concepts as "target platform" or any test environment in particular. It states that testing is done on an "item" or a "system" which can only be assumed to mean the real target environment.	20	Model-based testing "other purposes" needs elaboration.

However, there is one mention to modelling: *Tests are performed on all or part of the physical system or item* ***or an appropriate validated model*** *using procedures documented in sufficient detail so that a second party could reproduce the test results.* This can be inferred to mean that MIL virtual testing is an acceptable means of verification, but why no mention of SIL, VPIL or HIL VTTs? **ARP-4754: GAP003** ***No mention of other virtual testing methods, only the use of models.*** In this section there is also a table of verification activities against IDALs. ***TABLE 7 - Verification Methods and Data*** states that for levels A and B testing must be done plus at least one other method (e.g. review).		
ARP-4754: GAP003 ***No mention of other virtual testing***	20	Only MIL VTT mentioned testing verification method.

methods, only the use of models. In this section there is also a table of verification activities against IDALs. ***TABLE 7 - Verification Methods and Data*** states that for levels A and B testing must be done plus at least one other method (e.g. review).		

Table 3 - SAE ARP-4754 Identified Gaps

Reference	Page	Brief
RTCA DO-254: GAP001 ***A test environment could be chosen and defined as part of the planning process.*** As part of the overall hardware design process, ***sections 5.1*** and ***5.2*** elaborate the requirements capture and conceptual design processes respectively. Missing from both of these sections is the use of modelling to help elaborate either requirements or design	21	Test environments and the planning process.

concepts. As will be seen in the following subsection, MIL virtual testing can be used to help elaborate the software requirements of a CSCI at an early stage in the lifecycle. Why not use the same VTT method here? **RTCA DO-254: GAP002** ***No mention of modelling to facilitate requirements or design elaboration.*** Requirements validation and verification is covered by ***section 6*** of the document and more specifically, ***section 6.3.1*** discusses the methods to be used for this activity. No mention of particular test environments is made at all in this section. However the following paragraph is of interest: *When it is not feasible to verify specific requirements by exercising the hardware*		

item in its intended operational environment, other verification means should be provided, and justified. This would seem to suggest that a VTT environment could be used for verification activities but all of the responsibility for justifying it (and therefore using those test results for certification) falls to the user of the document. In short, there are no guidelines for test environments. **RTCA DO-254: GAP003** ***Some discussion about test environments and specifically "non-intended operational environments" would be useful here.*** Aside from testing, there is also analysis as a necessary verification activity. ***Section 6.3.2*** covers this activity and includes the following text: ***Simulation*** *is an important design analysis tool both for visualization of circuit operation and for higher level functional*		

operation. ***Simulation*** *can be used to analyze the impact of production variations in hardware parameters that would be difficult to do using other verification means and thus build confidence in reduction of design errors affecting safety due to these variations. Since the results depend on the* ***models*** *and scenarios employed, simulation results alone cannot be used for the purpose of certification credit without supporting evidence of their validity.* The use of simulation in the analysis activity would seem to lend itself to MIL testing with possible SIL components providing auxiliary functions in the test environment. The last sentence, however, seems to imply that any models which are used are *themselves* considered to be simulations and not necessarily representative of the real item under test. This is a different concept to the use of modelling for software items in which the model is an exact specification of the HLRs and can even be used to auto-generate the actual code. **RTCA DO-254: GAP004**		

What if models were used to generate HDL for the item? Analysis by simulation would seem to require particular test environments to be precisely specified. This is a similar observation as ***RTCA DO-254: GAP001***, above.		
RTCA DO-254: GAP002 ***No mention of modelling to facilitate requirements or design elaboration.*** Requirements validation and verification is covered by ***section 6*** of the document and more specifically, ***section 6.3.1*** discusses the methods to be used for this activity. No mention of particular test environments is made at all in this section. However the following paragraph is of interest: *When it is not feasible to verify specific requirements by exercising the hardware item in its intended*	21	Use of modelling in requirements and design processes.

operational environment, other verification means should be provided, and justified. This would seem to suggest that a VTT environment could be used for verification activities but all of the responsibility for justifying it (and therefore using those test results for certification) falls to the user of the document. In short, there are no guidelines for test environments. **RTCA DO-254: GAP003** ***Some discussion about test environments and specifically "non-intended operational environments" would be useful here.*** Aside from testing, there is also analysis as a necessary verification activity. ***Section 6.3.2*** covers this activity and includes the following text: ***Simulation*** *is an important design analysis tool both for visualization of circuit operation and for higher level functional operation.* ***Simulation***		

can be used to analyze the impact of production variations in hardware parameters that would be difficult to do using other verification means and thus build confidence in reduction of design errors affecting safety due to these variations. Since the results depend on the ***models*** *and scenarios employed, simulation results alone cannot be used for the purpose of certification credit without supporting evidence of their validity.* The use of simulation in the analysis activity would seem to lend itself to MIL testing with possible SIL components providing auxiliary functions in the test environment. The last sentence, however, seems to imply that any models which are used are *themselves* considered to be simulations and not necessarily representative of the real item under test. This is a different concept to the use of modelling for software items in which the model is an exact specification of the HLRs and can even be used to auto-generate the actual code. **RTCA DO-254: GAP004** ***What if models were used to generate HDL for the item?***		

Analysis by simulation would seem to require particular test environments to be precisely specified. This is a similar observation as ***RTCA DO-254: GAP001***, above.		
RTCA DO-254: GAP003 ***Some discussion about test environments and specifically "non-intended operational environments" would be useful here.*** Aside from testing, there is also analysis as a necessary verification activity. ***Section 6.3.2*** covers this activity and includes the following text: ***Simulation*** *is an important design analysis tool both for visualization of circuit operation and for higher level functional operation.* ***Simulation*** *can be used to analyze the impact of production variations in hardware parameters that would be difficult to do using other verification means and thus build confidence in reduction of design errors affecting safety due to these variations. Since the results depend on the* ***models*** *and scenarios employed, simulation results alone*	22	Non-intended operational environments.

cannot be used for the purpose of certification credit without supporting evidence of their validity. The use of simulation in the analysis activity would seem to lend itself to MIL testing with possible SIL components providing auxiliary functions in the test environment. The last sentence, however, seems to imply that any models which are used are *themselves* considered to be simulations and not necessarily representative of the real item under test. This is a different concept to the use of modelling for software items in which the model is an exact specification of the HLRs and can even be used to auto-generate the actual code. **RTCA DO-254: GAP004** ***What if models were used to generate HDL for the item?*** Analysis by simulation would seem to require particular test environments to be precisely specified. This is a similar observation as ***RTCA DO-254: GAP001***, above.		
RTCA DO-254: GAP004	22	Models used for HDL generation.

Table 4 – RTCA DO-254 Identified Gaps

Reference	Page	Brief
DO-178/C: GAP001	23	Software integration process contradiction.

DO-178/C: GAP002	24	Ambiguous term “*compatible*”.
DO-178/C: GAP003	24	Ambiguous term “target computer environment”.
DO-178/C: GAP004	24	Ambiguous term “correct operation of the software”.
DO-178/C: GAP005	26	Cohesion of testing and test environment is missing.
DO-178/C: GAP006	26	Test environments and software levels.

Table 5 – RTCA DO-178/C Identified Gaps

4.2 RECOMMENDED ENHANCEMENTS TO STANDARDS

By implication, the above “gap” tables will serve as recommended points to be addressed for the documents which could result in subsequent document enhancements. The following subsections present more descriptive recommendations.

4.2.1 SAE ARP-4754

The document quite openly delegates responsibility of product certification to hardware and software certification guidelines documents. This is understandable because it is intended to be a guide to recommended practices that can then be applied to both hardware and software guides.

It states that testing via prototyping, simulation or other means is an acceptable way to validate requirements, but stipulates that:

> *Care should be exercised to ensure any simulation is sufficiently representative of the actual system, its interfaces, and the installation environment.*

This might have been a good point at which to introduce the idea of another document that is responsible for the certification of test environments (if it existed). This idea is discussed further in the following sections.

During the verification section of the document there is also a notable lack of discussion about test environments. This sentiment will be repeated for the subsequent hardware and software certification guidelines documents under analysis

There is lip-service paid to MIL virtual testing but no real discussion of how to apply it.

A comprehensive section on different test environments and their use would be a good addition to the document.

4.2.2 RTCA DO-254

Any discussion of different test environments (and therefore possible VTTs) is all but absent from the document.

It could therefore be argued that the test environment is considered to be irrelevant for hardware certification evidence gathering. The assumption being that the only environment that matters is the real environment in which the LRU was designed to operate.

This is at odds, however, with the description of simulation testing for Analysis verification activities. Stating that some things "can only be tested by simulation".

So there is some confusion here about the environment(s) in which verification must be performed. As with ***[SAE ARP-4754]*** (and ***[RTCA DO-178/C]*** below) a greater consideration of test environments would be useful in the document.

There is also virtually no mention of modelling (and therefore no relevance to MIL testing) in the document, but these are covered by the gap table above.

4.2.3 RTCA DO-178/C

The biggest area of improvement here would be to eliminate the ambiguities (see ***Table 5 – RTCA DO-178****/C Identified Gaps*). It is perhaps because of these ambiguities that the current, typical certification practices present evidence from target platform testing above all other forms of testing.

Regarding the test environment description of the document (see ***3.3.2.1***), the section does not seem to be in-line with the previous sections that detail the testing objectives and the test activities.

If the testing section of the document was more oriented towards different testing environments and then define test objectives and test activities as applied to those environments, it could possibly clarify how a user of the document could apply different VTTs for gathering certification evidence. As it stands, the test environment section appears to be more of an afterthought and can cause confusion.

The software integration process section also mentions different test environments. This seems to be misleading and perhaps it should concentrate solely on the target hardware test environment.

There is no consideration of software levels as applied to different test environments. There could be some classification of different test environments by how closely they resemble the real target environment. Such “test environment levels” could then be applied to the testing of CSCIs of differing software levels (see the ***Summary*** section below).

4.2.4 Summary

There exist guidelines for the certification of tools, but not specifically for the “certification” of test environments. Guidelines on how to demonstrate how closely a test environment represents the target environment would be a useful addition to the suite of documents.

A test environment could also be assigned a “level” similar to those used in the three analysed documents depending on a set of criteria. Level A being an AC/0 test rig, for example. Such categorised environments could be matched to hardware and software levels for the validation and verification processes in ***[RTCA DO-254]*** and ***[RTCA DO-178/C]***.

Apart from the suggested enhancements to those documents analysed (see above), a recommendation from this analysis as a whole would be to create a new document for the classification of test environments and their use in the certification process with particular attention to virtual testing methodologies. If not a new document then perhaps a dedicated chapter could be added to ***[SAE ARP-4754]***.

5 ACRONYMS & ABBREVIATIONS

AC/0	Aircraft Zero (testing rig)
AFDX	Avionics Full-Duplex Switched Ethernet
API	Application Programming Interface
ARINC	Aeronautical Radio, Incorporated
ARP	Aerospace Recommended Practice
ASAAC	Allied Standards Avionics Architecture Council
CSCI	Computer Software Configuration Item
DAL	Design Assurance Level
HDL	Hardware Description Language
HIL	Hardware In-The-Loop virtual testing methodology
HLR	High-Level Requirement
IDAL	Item Development Assurance Level
I/O	Input/Output
IMA	Integrated Modular Avionics
LLR	Low-Level Requirement
LRU	Line-Replaceable Unit
MIL	Model In-The-Loop virtual testing methodology
RTCA DO	Radio Technical Commission for Aeronautics DOcument
SAE	SAE International (formerly Society for Automotive Engineers)
SIL	Software In-The-Loop virtual testing methodology
STANAG	Denotes NATO Standardization Agreement
TOR	Tool Operational Requirement
VPIL	Virtual Processor In-The-Loop virtual testing methodology
VTT(s)	Virtual Testing Technology(ies)

End of Document.