NASA/CR–2013-218010

Final Report – Regulatory Considerations for Adaptive Systems

Chris Wilkinson
Honeywell International Inc., Columbia, Maryland

Jonathan Lynch
Honeywell International Inc., Albuquerque, New Mexico

Raj Bharadwaj
Honeywell International Inc., Golden Valley, Minnesota

June 2013

NASA STI Program . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA scientific and technical information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI program provides access to the NASA Aeronautics and Space Database and its public interface, the NASA Technical Report Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA Programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers, but having less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include organizing and publishing research results, distributing specialized research announcements and feeds, providing information desk and personal search support, and enabling data exchange services.

For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at *http://www.sti.nasa.gov*

- E-mail your question to help@sti.nasa.gov

- Fax your question to the NASA STI Information Desk at 443-757-5803

- Phone the NASA STI Information Desk at 443-757-5802

- Write to:
  STI Information Desk
  NASA Center for AeroSpace Information
  7115 Standard Drive
  Hanover, MD 21076-1320

NASA/CR–2013-218010

Final Report – Regulatory Considerations for Adaptive Systems

Chris Wilkinson
Honeywell International Inc., Columbia, Maryland

Jonathan Lynch
Honeywell International Inc., Albuquerque, New Mexico

Raj Bharadwaj
Honeywell International Inc., Golden Valley, Minnesota

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

Prepared for Langley Research Center
under Contract NNL12AB32T

June 2013

Acknowledgments

This work was performed under NASA Flight Critical Systems Research Contract NNL06AA05B. We would like to thank Kelly Hayhurst at NASA for her support and encouragement

Available from:

NASA Center for AeroSpace Information
7115 Standard Drive
Hanover, MD 21076-1320
443-757-5802

Table of Contents

1 Introduction 1

2 Current Certification Guidance and Standards 2

2.1 ARP-4754A Guidelines for Development of Civil Aircraft and Systems 3

2.1.1 Recent Changes to ARP-4754 5

2.2 ARP-4761 Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment 8

2.3 Software Design Assurance 8

2.3.1 Recent Changes to DO-178B 10

3 Adaptive System Types and Architectures 12

4 Adaptive System Certification 14

4.1 Concerns on the Feasibility of Applying DO-178B to Software Design Assurance of Adaptive Systems 14

4.2 Suggested Approach to Software Design Assurance of Adaptive Systems 14

4.3 System Level Approaches to the Certification of Adaptive Systems 16

4.4 Defining the System Level Characteristics of an Adaptive System 16

5 Aspects of Tool Qualification for Adaptive Systems 19

6 Recommendations 19

6.1 Recommendations for Adaptive System Safety and Functional Requirements Derivation and Validation 20

6.2 Recommendations for Adaptive System Requirements Verification 20

7 Future Adaptive Systems Certification Research Needs 21

8 Conclusions 22

9 Annex A 23

10 References 37

Table of Figures

Figure 1 – Certification Process Flow and Applicable Standards 3
Figure 2 – ARP-4754 – ARP-4754A Sections Mapping 5
Figure 3 – In/Out Mapping of ARP-4754 and ARP-4754A 6
Figure 4 – Adaptive System Taxonomy 12
Figure 5 – Exemplar Flight Control Architecture 14

Table of Tables

Table 1 - ARP-4754A Invocation 4
Table 2 - DO-178B Invocation 9
Table 3 - System Safety Objectives for Adaptive Systems (developed with exemplar adaptive flight control system in mind) 17
Table 4 – DO-178B/C Objectives Applied to Adaptive Systems (developed with exemplar adaptive flight control system in mind) 23

Glossary

AC	Advisory Circular (FAA)
AS	Adaptive System
ASA	Aircraft Safety Assessment
ATM	Air Traffic Management
CFR	Code of Federal Regulations
COTS	Commercial Off The Shelf
CRI	Certification Review Item (EASA)
CRI	Certification Requirement Item (EASA)
CS	Certification Specification (EASA)
DAL	Design Assurance Level
DER	Designated Engineering Representative
EASA	European Aviation Safety Agency
EOC	Executable Object Code
EuroCAE	European Organization for Civil Aviation Equipment
FC	Failure Condition
FDAL	Function Development Assurance Level
FFS	Functional Failure Sets
FM	Formal Methods
HLR	High Level Requirements
IDAL	Item Development Assurance Level
IP	Issue Paper (FAA)
LLR	Low Level Requirements
MBD	Model Based Design
MRAC	Model Reference Adaptive Control
OOT&RT	Object-Oriented Technology and Related Techniques
PASA	Preliminary Aircraft Safety Assessment
PRA	Probabilistic Risk Assessment
RBT	Requirements Based Test
SMT	Satisfiability Modulo Theory
STC	Supplementary Type Certificate
TC	Type Certificate
TQL	Tool Qualification Level
TSO	Technical Standards Order
V&V	Verification and Validation
VHM	Vehicle Health Management
WCET	Worst Case Execution Time

1 Introduction

In the *Decadal Survey of Civil Aeronautics: Foundation for the Future* [1] the National Research Council has identified *intelligent and adaptive systems* as one of the five common threads for the "51 high-priority R&T challenges". The report has explicitly identified adaptive systems (AS) technologies to be the key enabler for intelligent flight controls, advanced guidance and adaptive air traffic management (ATM) systems and for health management techniques to extend life and improve maintenance.

Adaptive flight and engine controls systems have been researched for decades. The main attraction for adaptive systems is to detect, anticipate and prevent failures and reconfigure the aircraft displays/controls/engine operations in response. Adaptive systems can detect the degradation in performance due to failure or damage. An effective way of adaptation is through intelligent utilization of dynamic system identification and learning systems. System identification and learning is used to update the on-board aircraft process model and the control laws to adapt the system behavior and so continue to effectively control the aircraft. This increases time on wing, improve readiness and reduce operating costs.

Another push for use of adaptive systems is coming from the growth in US air transport system capacity. The US air transportation system is expected to grow to double or triple the current size through the 2025 time frame; there is concern that the current ATM systems are not flexible enough to handle that growth. To meet the need for increased capacity and efficiency while maintaining operational safety, new technologies, processes and innovations must be implemented. These innovations provide for reduced separation and support transition from traditional, rules-based operations to performance-based ones. In light of the changes anticipated in NextGen ATM there is a renewed push on adaptive systems research. The adaptive guidance and control systems also enable the safe operation of both piloted and unmanned aircraft under various environmental disturbances. They minimize the impact of uncertainty (due to weather) and failure conditions.

Finally adaptive systems are being proposed for management of human machine interactions on aircraft and ATM systems to mitigate the safety incidents due to failures at the human machine interface. In this case the emphasis is on the system behavior that adapts to the current context (tasks, user state, system configuration, environmental states, and so on). Also, some methods of assessing context may utilize a learning methodology. So for instance, using electroencephalogram (EEG) data to feed a system to assess the current cognitive state of a user may require training data so the system can learn what high and low workload looks like.

There are two fundamental questions that need to be addressed for all the adaptive systems namely; when to adapt and how to adapt? Answering these questions when performing verification of the adaptive systems behavior can lead an unbounded increase of verification cases. The real challenge is to show that the system behavior is bounded, correct and accurate under all inputs. The new capabilities that adaptive systems offer can only be realized if an acceptable methodology can be found for certifying them within the context of the current Code of Federal Regulations (CFRs), associated advisory materials and industry standards. This work will provide suggestions for advanced design and safety assurance methodologies and techniques and the associated guidance that would address the current roadblocks. The techniques suggested are likely to be applicable to conventional systems as well and may provide ways in which the

DO-178B [2] design assurance burden can be reduced by focusing on the safety related requirements and providing more automated methods for development of verification evidence.

Adaptive systems are perceived to be particularly difficult to certify because by definition they change their software defined parameters whilst in operation in response to the experienced time varying operating environment. Complying with the objectives and requirements of DO-178B is apparently problematic since DO-178B (and its successor DO-178C [3]) were conceived with an implicit assumption of time-invariant software parameters. One objective of this program was to show whether an adaptive system could be certified within the current regulatory framework, or if not, suggest changes to the framework that would allow for certification whilst preserving the essential features of safety in the final product.

We first review the current certification framework and the recent changes it has undergone and then describe our suggested approach to design and certification methodology and framework that could result in successful progression through this framework.

Much of what follows is applicable to DO-178C also and in fact we will show that the new supplements added to DO-178C are in fact essential components of a successful certification of an adaptive system.

2 Current Certification Guidance and Standards

We first discuss the current certification framework since we would like to determine whether or not adaptive systems could be certified within it as is, if some changes would be needed and if so, what those changes are recommended to be.

This framework is built around CFR Title 14 (hereafter 14CFR) requirements. Figure 1 shows the general process flow and the applicable de-facto standards of current certification practice. These standards relate to1) system development, 2) safety assessment and 3) design assurance of system hardware and software. Details of all activities and deliverables to be fully compliant are not shown in interest of focusing on the key steps; these can be found within the documents referenced. The scope here is to give an overview and not a full descriptive narrative. We do not discuss DO-254 [4] since this is not within the scope of the present program and is in any case limited in application to programmable devices and not to hardware generally [5, 6]. Similarly, SAE ARP-5150 [7] provides guidance for assessing ongoing safety during commercial operations but we do not discuss this aspect here.

In each case there is a direct equivalence between US and European editions of these documents. These are denoted by SAE/RTCA document numbers and corresponding European (published by EuroCAE) documents numbers. We refer only to the US editions here for brevity.

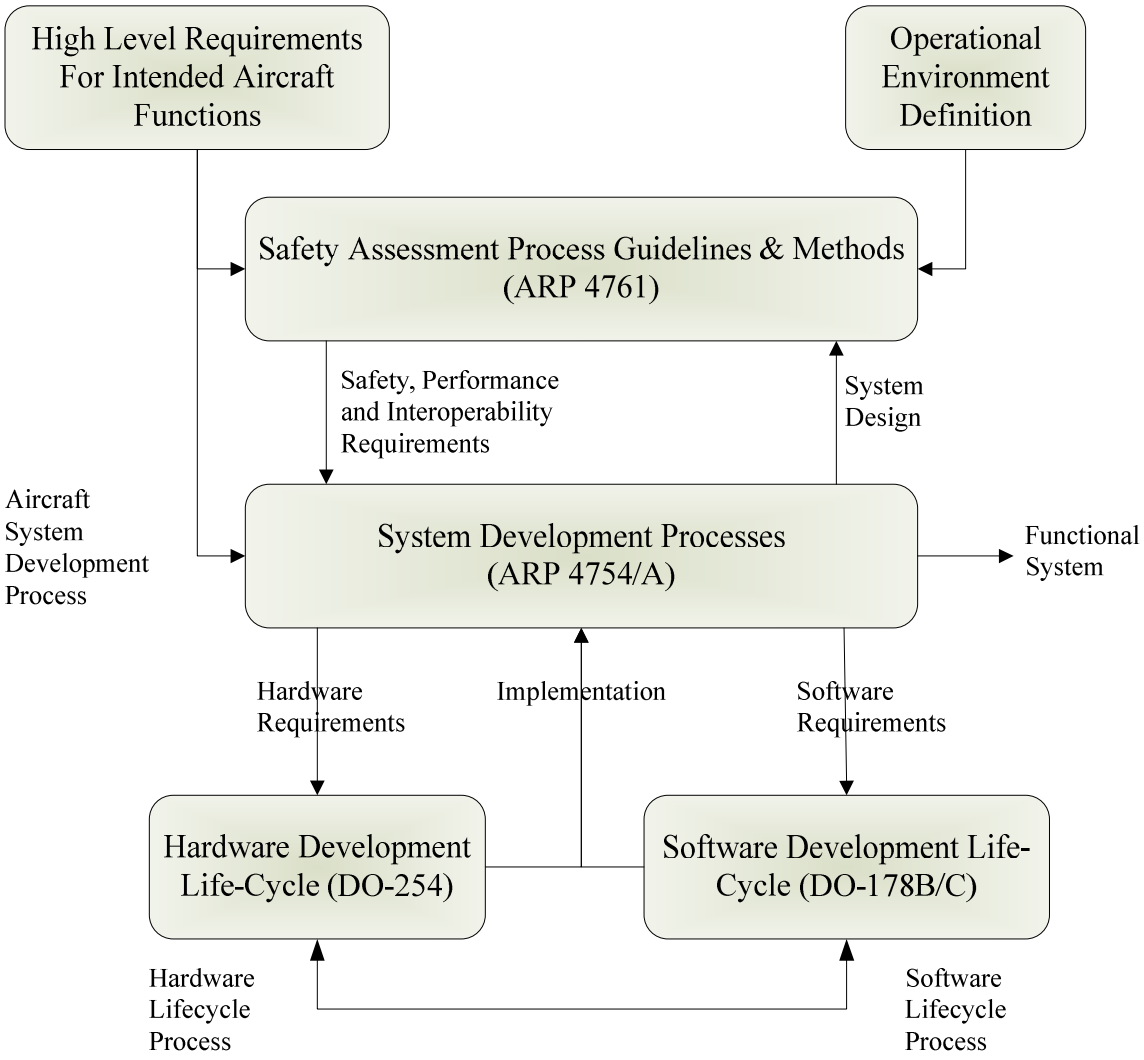

Figure 1 – Certification Process Flow and Applicable Standards

2.1 ARP-4754A Guidelines for Development of Civil Aircraft and Systems

The original version of ARP-4754 [8] has been in use for 15 years. The development of ARP-4754A started in 2003 by the SAE S-18 Committee and was published in 2010. It now has a new title "Guidelines for development of civil aircraft and systems" (o*ld title: Certification considerations for highly-integrated or complex aircraft systems*). ARP-4754A discusses the certification aspects of systems installed on aircraft, taking into account the overall aircraft operating environment and functions. To quote from ARP-4754A;

"This document discusses the development of aircraft systems taking into account the overall aircraft operating environment and functions. This includes validation of requirements and verification of the design implementation for certification and product assurance. It provides practices for showing compliance with the regulations and serves to assist a company in developing and meeting its own internal standards by considering the guidelines herein.

The guidelines in this document were developed in the context of Title 14 Code of Federal Regulations (14CFR) Part 25 and European Aviation Safety Agency (EASA) Certification Specification (CS) CS-25. It may be applicable to other regulations, such as Parts 23, 27, 29, 33, and 35 (CS-23, CS-27, CS-29, CS-E, CS-P).

This document addresses the development cycle for aircraft and systems that implement aircraft functions. It does not include specific coverage of detailed software or electronic hardware development, safety assessment processes, in-service safety activities, aircraft structural development nor does it address the development of the Master Minimum Equipment List (MMEL) or Configuration Deviation List (CDL). More detailed coverage of the software aspects of development are found in RTCA document DO-178B, "Software Considerations in Airborne Systems and Equipment Certification" and its EUROCAE counterpart, ED-12B. Coverage of electronic hardware aspects of development are found in RTCA document DO-254/EUROCAE ED-80, "Design Assurance Guidance for Airborne Electronic Hardware". Design guidance and certification considerations for integrated modular avionics are found in appropriate RTCA/EUROCAE document DO-297/ED-124. Methodologies for safety assessment processes are outlined in SAE document ARP4761, "Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment". Details for in-service safety assessment are found in ARP5150, "Safety Assessment of Transport Airplanes In Commercial Service" and ARP5151 Safety Assessment of General Aviation Airplanes and Rotorcraft In Commercial Service." Post-certification activities (modification to certificated product) are covered in section 6 of this document. The regulations and processes used to develop and approve the MMEL vary throughout the world. Guidance for the development of the MMEL should be sought from the local airworthiness authority."

Table 1 below shows the means by which ARP-4754A may be invoked for a particular certification project. ARP-4754A guidance may also be applicable to aircraft equipment certified to other 14CFR Parts such as Parts 23, 27, 29 and 33 so for brevity in what follows we discuss this in reference to Part 25. In this table, we use the term 'invocation/policy' to mean that the referenced document is recognized by the regulator as an acceptable means of compliance with the applicable 14CFR Part.

The issuance of an advisory circular (AC) is a regulatory policy declaration that an applicant's compliance thereto is one, but not the only, acceptable means of showing compliance to the referenced Part of 14CFR. Compliance is recommended but is not mandatory. Compliance to ARP-4754A may also be required by the certification authority through the issue of a project specific Issue Paper (FAA, IP) or Certification Review Item (EASA, CRI). Final regulatory approval of all systems is assumed to be accomplished through or within a Technical Standards Order (TSO), Type Certificate (TC) or Supplementary Type Certificate (STC) certification project.

Table 1 - ARP-4754A Invocation

Reference	Description	Applicability	Invocation
ARP-4754A	Guidelines for Development of Civil Aircraft and Systems	Aircraft systems and equipment	AC 20-174 [9], IP, CRI
AC 25.1309-1A [10]	Describes various acceptable means for showing compliance with the requirements of 14 CFR section 25.1309(b), (c), and (d)	Applies to any system on which compliance with any of those requirements is based. Section 25.1309(b) and (d) specifies required safety levels in qualitative terms, and requires that a safety assessment be made	Policy

2.1.1 Recent Changes to ARP-4754

ARP-4754A has had substantial updates relative to its predecessor. We summarize the major changes below. This summary is based on unpublished presentations given at an S-18 WG-63 sub-committee meeting in Miami, January 2011.

- (+) Reinforces Development aspects (not only for Certification).
- (-) Highly-integrated or complex systems notion lost (in the title) because ambiguous.

The guidelines are primarily directed toward systems that support aircraft level function. Typically, these systems involve significant interactions with other systems in a larger integrated environment. The contents are recommended practices and should not be construed to be regulatory requirement. It is recognized that alternative methods to the processes described or referenced may be available to an applicant desiring to obtain certification. A conceptual mapping of the old and new sections is shown below.

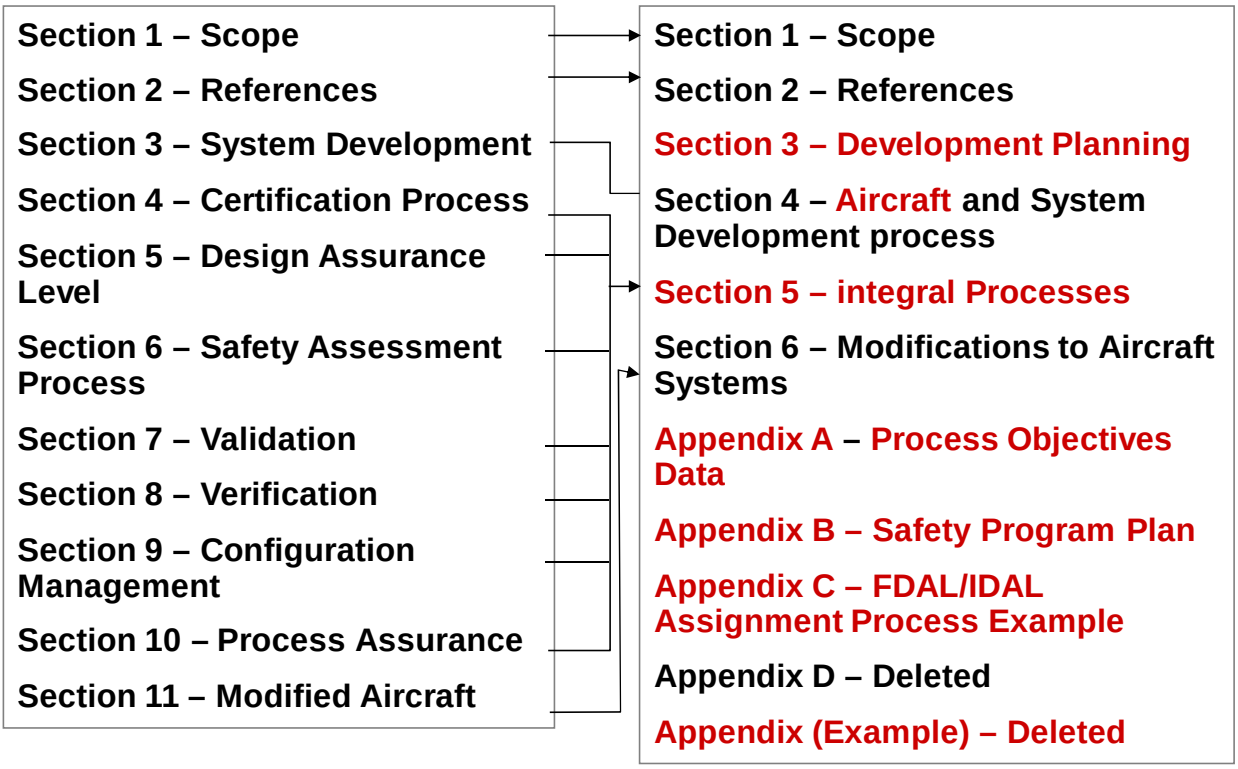

Figure 2 – ARP-4754 – ARP-4754A Sections Mapping

The major changes and new content can be summarized as:

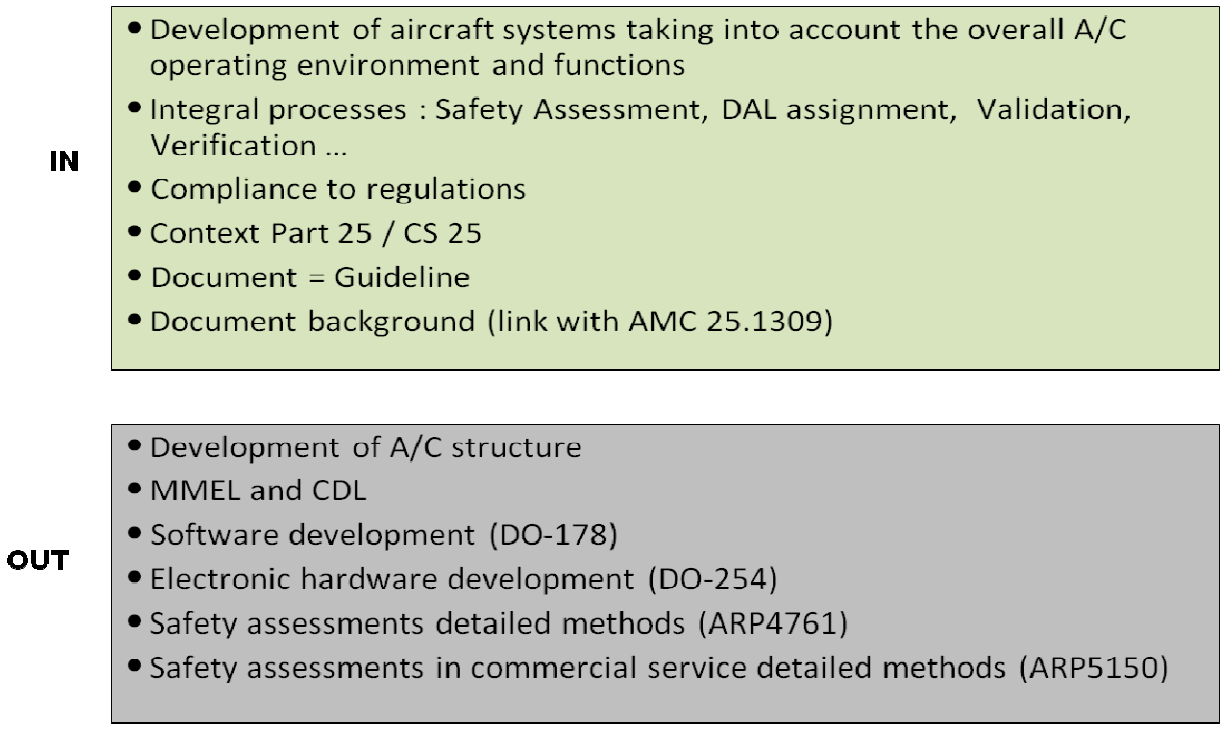

Figure 3 – In/Out Mapping of ARP-4754 and ARP-4754A

Section 1 - Scope

- A paragraph has been deleted with reference to the precedence of this document in the event of conflict between the text of this document and the text of DO-178B
- All the information concerning highly integrated or complex systems has been deleted

Section 2 – References

- Applicable documents -> Relationship between American/European standards
- Definitions
- Abbreviations and acronyms

Section 3 - Development Planning

- Life cycle process checkpoints and reviews
- Maturity expectations
- Transition Criteria (i.e. life cycle process checkpoints and reviews, which are aligned with program phases and gates)
- Management of deviations from plans

Section 4 - Aircraft and system development process

- Identification of Aircraft-Level Functions, Function Requirements and Function Interfaces
- Relationship between Requirement Levels, Function Development Assurance Level (FDAL) and Item Development Assurance Level (IDAL)
- The objectives for accomplishment of FDAL and IDAL (i.e. ARP4754A Appendix A, DO-254/ED-80 and DO-178B/ED-12)

Section 5.1 – Safety Assessment

- Safety Case / Safety Synthesis
- Safety Program Plan
- Preliminary Aircraft Safety Assessment (PASA)
- Aircraft Safety Assessment (ASA)

Section 5.2 – Development Assurance Level Assignment

- DAL assignment based on failure condition (FC) severity classification and independence attributes (no more based on type of architectures)
- Two different DAL: FDAL that apply to function requirement development and IDAL that apply to item (Hardware/Software) development
- Concept of Functional Failure Sets (FFS)
- New Table 5-2 "Development Assurance Level Assignment to Members of a Functional Failure Set" with two assignment options
- FDAL Assignment Taking Credit for External Events

Section 5.3 - Requirements Capture

- Re-use of Existing Certified Systems and Items. The requirements to which the system or Item was certified should be validated according to the new application and modified as necessary
- Deriving Safety-related Requirements from the Safety Analyses

Section 5.4 - Requirements Validation

- Definition of correctness and completeness improved
- Validation Rigor improved with the concept of independence in the validation process
- The application of independence in the validation process is dependent upon the development assurance level
- The validation plan should include a description of the validation activities to which independence is applied
- Independent review of requirement data and supporting rationale
- The reviews should be documented including the review participants and their roles

Section 5.5 - Implementation Verification

- Identification of key **verification activities** and sequence of any dependent activities
- Identification of the roles and responsibilities associated with conducting the verification activities and a description of independence between design and verification activities

Section 5.6 - Configuration management

- Two System Control Categories (see ARP-4754A Table 5-6**)**

Section 5.7 – Process Assurance

- The process assurance activities described are not intended to imply or impose specific organizational structures or responsibilities. However, process assurance should have a level of independence from the development process

Section 5.8 – Certification Process

- Certification Planning: there may be a single certification plan for the project or a top-level plan for the aircraft and a set of related plans for each of the aircraft systems
- Early coordination and approval of the plan is strongly encouraged

Section 6 - Modification to Aircraft or Systems

- Aviation Authority requirements and regulations categorize aircraft modifications into either "minor or major" changes
- When a modification is proposed to an item, system or aircraft an initial impact analysis should be performed and should include an evaluation of the impact of the modification on the original safety assessments
- The modification impact analysis should be confirmed or updated once verification activities have been completed. Results of these analyses should be reflected in:
    - The appropriate certification documentation
    - The verification activities needed to assure that no adverse effects are introduced during the modification process
    - The modification summary in which the impact of the implemented modifications is confirmed

Appendices

- Appendix A – Process Objectives Data
- Table A-1: Process Objectives, Outputs & System Control Category by function development assurance level (note: the scope and detail of the life cycle data varies depending on the FDAL assigned)
- Appendix B – Safety Program Plan
- Appendix C – FDAL/IDAL assignment example
- Appendix D – deleted
- Previous guidelines in this appendix have been superseded by the material found in section 5.2 of ARP-4754A

2.2 ARP-4761 Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment

The major guidance for civil airborne systems and equipment safety assessment is SAE ARP-4761 [11]. This is commonly accepted by certification authorities and industry as an acceptable, but not the only, means of showing compliance to AC 25.1309. However it is not formally referenced or recognized in an issued AC. ARP-4761 describes guidelines and a variety of example probabilistic risk assessment (PRA) methods and techniques for performing the safety assessment of civil aircraft systems and equipment. SAE S-18 is currently updating this document with an expected release in 2014.

2.3 Software Design Assurance

The primary software design assurance guidance document is DO-178B. The table below shows how it is invoked by the current regulatory framework.

Table 2 - DO-178B Invocation

Reference	Description	Applicability	Invocation
DO-178B	Software Considerations in Airborne Systems and Equipment Certification	Provides guidance for the production of software for airborne systems and equipment that performs its intended function with a level of confidence in safety that complies with airworthiness requirements	TSO, AC 20-115B
Order 8110.49, Change 1	Software Approval Guidelines	This order guides Aircraft Certification Service (AIR) field offices and Designated Engineering Representatives (DER) on how to apply RTCA/DO-178B, “Software Considerations in Airborne Systems and Equipment Certification,” for approving software used in airborne computers.	Policy
AC 20-115B	Calls attention to RTCA DO-178B, “Software Considerations in Airborne Systems and Equipment Certification”	Calls attention to RTCA DO-178B, “Software Considerations in Airborne Systems and Equipment Certification,” issued December 1992. It discusses how the document may be applied with FAA technical standard order (TSO), authorizations, type certification (TC), or supplemental type certification authorization (STC).	Policy

2.3.1 Recent Changes to DO-178B

DO-178B has been revised to DO-178C by the joint RTCA SC-205/ EUROCAE WG-71 special committees. The following paragraphs summarize the major changes and rationale. DO-178B did not provide any specific guidance for the newer software technologies listed below. There are now 4 new supplements to provide this. DO-178C is not presently invoked by the certification authorities.

2.3.1.1 Core Document

In general the new revision makes only minor changes to the main body, largely to clarify some language which had caused confusion in the past. The changes to Section 6.0 were very limited. So there should be very little impact to the way verification is performed today. However, there are some subtle changes that could impact projects that had interpreted DO-178B differently than intended. In general, the language was improved and made more explicit. There were also a few objectives added to explicitly call out what were sometimes called "hidden objectives" in DO-178B. The prime example is Section 6.4.4.2b of DO-178B that states that analysis was needed to compare source code to object code. In the end only three objectives were added in Tables A-3 through A-7.

2.3.1.2 Model-Based Development (MBD) and Verification Supplement to DO-178C and DO-278A

The goal of the MBD supplement RTCA DO-331 [12] is to provide guidance in an area where DO-178B is somewhat awkward to apply. The purpose of this supplement is (quoting from DO-331);

"This supplement contains modifications and additions to DO-178C objectives, activities, explanatory text, and software life cycle data that should be addressed when model-based development and verification are used as part of the software life cycle. This includes the artifacts that would be expressed using models and the verification evidence that could be derived from them. Therefore, this supplement also applies to the models developed in the system process that define software requirements or software architecture"

Models may represent high-level requirements (specification model) or equivalently behavior or may express the internal architecture, data structures and flow control (design model) of a software function. Models may be loosely defined as any computer representation of a function. Underlying all models is a set of mathematical equations, logical expressions etc. describing the functional, logical and temporal relationships of inputs and outputs. These relationships may be encoded in a tool language (e.g. a Simulink or similar block) which is used by the tool (the simulation environment, e.g. an interpreter) to execute under planned test conditions to demonstrate the model behavior.

This supplement treats the verification of models in the same way as DO-178C treats the verification of software. Models must be shown to be compliant with the stated model requirements by a combination of analysis, test and review, i.e. by satisfaction of the various DO-178C objectives.

Executable object code can be produced automatically by some tools. When code is produced this way, it does not alleviate the necessity of demonstrating that the executable object code satisfies the usual and corresponding high and low-level requirements of DO-178C. The certification credit attainable from the use of MBD and automated code generators is dependent on the desired software assurance level and the tool qualification level (TQL) per DO-178C.

2.3.1.3 Object-Oriented Technology and Related Techniques (OOT&RT) Supplement to DO-178C and DO-278A

The basis for the OOT&RT supplement DO-332 [13] was the guidance previously provided in the OOTiA (Object-Oriented Technology in Aviation) handbook [14]. Verification of software written with OOT&RT methods is intended to be performed in the same way as any other type of software per DO-178C. The supplement also addresses some verification challenges unique to OO, such as dynamic memory allocation. Quoting DO-332, the stated purpose is;

"The purpose of this supplement is to provide guidance for the production of software using OOT&RT for systems and equipment that performs its intended function with a level of confidence in safety that complies with airworthiness requirements. This supplement contains modifications and additions to DO-178C objectives, activities, explanatory text, and software life cycle data that should be addressed when OOT&RT are used as part of the software life cycle."

2.3.1.4 Formal Methods (FM) Supplement to DO-178C and DO-278A

The formal methods supplement DO-333 [15] introduces the concept of "formal analysis." The basic concept is that if FM is applied in a development program, then some credit can be taken toward some of the DO-178C verification objectives by performing formal analysis. Quoting DO-333, the stated purpose is;

"This supplement identifies the modifications and additions to DO-178C objectives, activities, explanatory text, and software life cycle data that should be addressed when formal methods are used as part of the software life cycle. This includes the artifacts that would be expressed using some formal notation and the verification evidence that could be derived from them."

As is well known testing has fundamental limitations of coverage. The theory is that by using formal proofs of behavior, we are able to verify the system more thoroughly than using traditional testing means. FM can prove that system functional and safety properties are satisfied under all conditions; criteria that would be infeasible or very difficult to prove through testing alone. Testing cannot be completely eliminated, but it could be reduced.

FM utilizes formal models that are operated on by a formal analysis tool. The tool must be assessed according to the DO-178C tool qualification supplement DO-330 [16]. FM are strongly linked to the MBD supplement described above with the emphasis being on mathematical models that have a precisely defined syntax and semantics e.g. mathematical notations such as logic, sets, differential equations or ADA and C subsets. Formal models need to be developed according to those guidelines.

2.3.1.5 Software Tool Qualification Considerations

DO-178C retains the tool types of DO-178B and adds one additional type. These tool types are now defined in terms of 3 criteria. The carried over tool types are those that could inject an error (development tools, Criteria 1) and those that could fail to detect an error (verification tools, Criteria 3). The additional type (Criteria 2) is a subset of verification tools. The criteria are now defined as below.

- Criteria 1: A tool whose output is part of the airborne software and thus could insert an error
- Criteria 2: A tool that automates verification process(es) and thus could fail to detect an error, and whose output is used to justify the elimination or reduction of:
    - Verification process(es) other than that automated by the tool, or

- Development process(es) that could have an impact on the airborne software
- Criteria 3: A tool that, within the scope of its intended use, could fail to detect an error

In addition a new table is given in DO-178C that relates the software assurance level, the tool type (i.e. the above criteria) and a TQL designator, TQL1-5 where TQL1 is the most rigorous level applying to Criteria 1 tools used for DAL-A software. DO-330 defines tool qualification objectives and provides guidance on the activities necessary to satisfy the objectives according to the TQL designation. Broadly these are the same objectives and activities that are defined for operational software in DO-178C.

3 Adaptive System Types and Architectures

Adaptive systems are a broad category of systems with many application domains and a variety of adaptation methods. We have categorized them in the taxonomy of Figure 4 below.

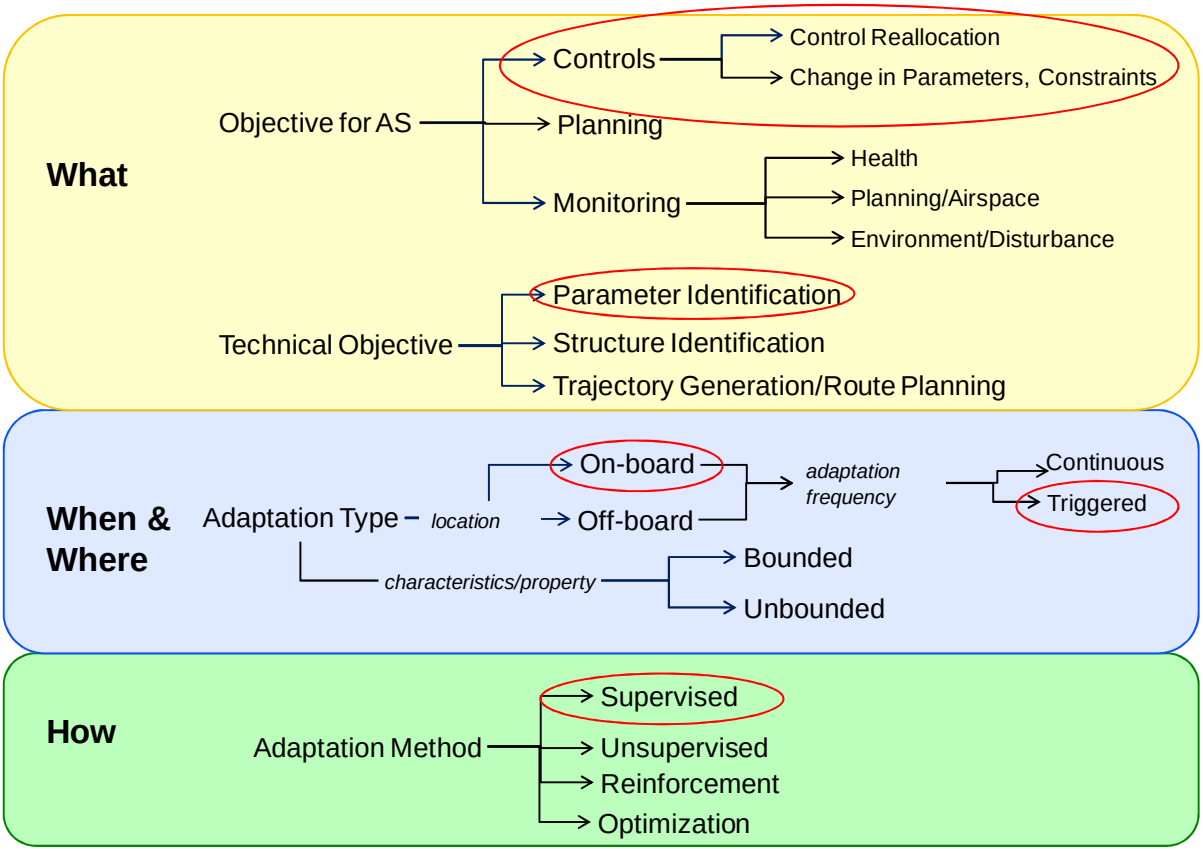

Figure 4 – Adaptive System Taxonomy

An AS typically incorporates a reference model of the desired response and a learning mechanism that adjusts the AS control parameters in response to measured changes in the external environment. The control objective is to adjust parameters so that the actual performance matches the desired. The learning mechanism may take several forms e.g. neural or radial basis function network, reflection/autonomic (dynamic code assignment or self-modifying code) and genetic update (random mutation and fitness selection). More details of these and references are provided in a predecessor NASA study [17] which we incorporate by reference.

Because of the wide variety of AS, we have focused our efforts on a particular adaptive system type. We defined a specific set of AS features and constraints to be more specific in our conclusions and recommendations. These are indicated by the red circles above. We selected a controls type of application (e.g. a flight controller that continuously monitors the environment (parameter identification), is on-board the aircraft (i.e. parameter updates are calculated locally not uplinked from a ground facility) and uses supervised learning to perform the parameter update. This choice was guided by two considerations;

First, adaptive control has been in use since at least the 1960s and has been the subject of much research and several military applications. Some examples are given below.

- NASA/USAF F-111 Mission Adaptive Wing [18]
- Boeing is using adaptive control for production JDAM
- NASA has been using the L1 adaptive control for research with an unmanned model of a small scale commercial aircraft [19]
- Rockwell Collins (formerly Athena) has demonstrated the Automatic Supervisory Adaptive Control (ASAC) on an unmanned small scale F/A-18 [20]
- Honeywell/AFRL Integrated Adaptive Guidance & Control For Future Responsive Access To Space Technology (FAST) [21]

We are not aware of the use of adaptive control in any commercial aircraft.

Secondly, we had tentatively concluded that to assure the safety of an airborne AS it would 1) be necessary to impose some system level features and constraints and 2) be of a type that could be feasibly and accurately represented by a computer model of a type amenable to automated analysis. We will discuss these aspects later in this report. These self-imposed requirements excluded the genetic algorithm and reflection/autonomic types of learning since they appeared to present extreme modeling difficulty. We did not consider controllers of the gain scheduled type since the adaptation is limited to a few pre-determined states and can therefore be verified using the standard methods of DO-178B/C. In a similar vein we did not consider an AS that is pre-trained offline and thereafter remains in a fixed configuration.

We therefore constructed a representative adaptive system architecture exemplar that generalizes the above examples and meets our imposed requirements to use in our analysis of verifiability per DO-178B/C. A block diagram is shown in Figure 5. This is a triggered system that uses an expected response model to update its operating parameters. The updated values are used when triggered by a signal, e.g. by pilot command or vehicle health management (VHM) system acting as an observer of various related aircraft components such as sensors, hydraulics, actuators, control surfaces etc., or the occurrence of a failure or off-nominal condition. It otherwise remains in a fixed/static configuration.

We set 2 major safety requirements; 1) that parameter adjustment be constrained to pre-determined ranges that guarantee stability and controllability (e.g. in the Lyapunov criteria sense) and 2) that the control error signal converges asymptotically to the neighborhood of zero, within an arbitrarily small bound, in finite time. These latter seem to be essential features of a certifiable flight control system whether adaptive or not.

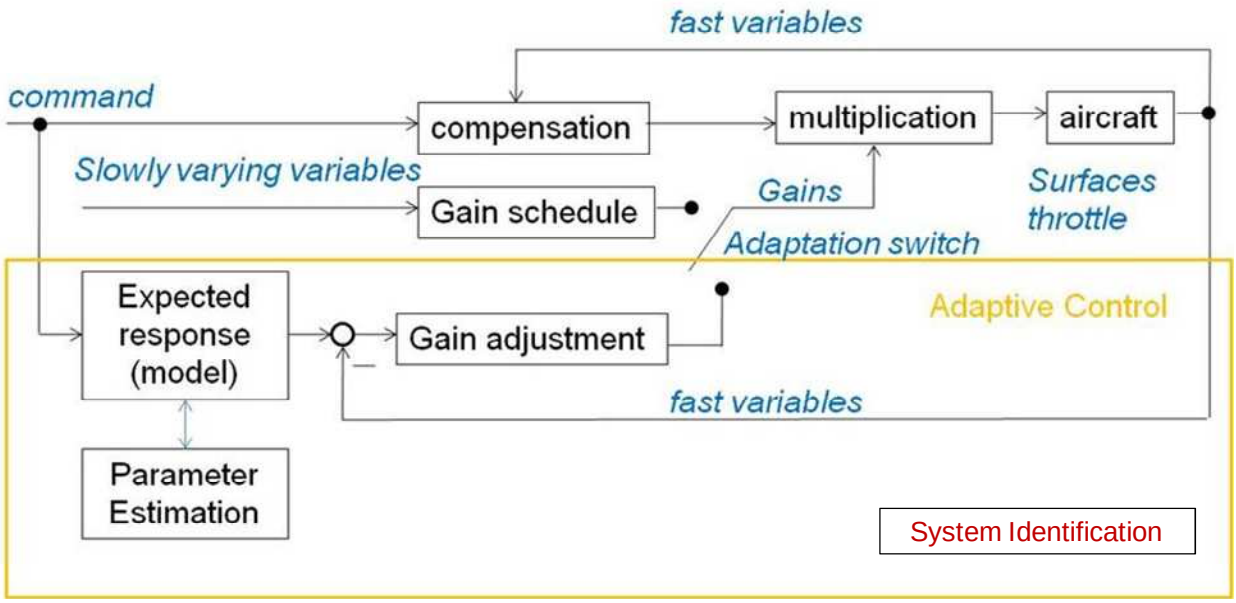

Figure 5 – Exemplar Flight Control Architecture

4 Adaptive System Certification

We now discuss adaptive system certification in the context of the selected adaptive system controller example and the current certification framework described above.

4.1 Concerns on the Feasibility of Applying DO-178B to Software Design Assurance of Adaptive Systems

All adaptive systems embody a learning sub-system of some type and some embedded model of the desired system performance. The learning sub-system changes the system operating parameters (e.g. control gain) in response to measurements taken on the external environment. The objective is that the actual performance closely matches the desired performance represented in the model. Non-adaptive (fixed configuration) systems assume the external environment remains constant. The external environment includes the sensors providing the inputs and actuators operating control surfaces. There may be several parameters that adjust value over time through the learning function. These will, in general, be real-valued variables. This means that an adaptive system has an infinite set of possible parameter values even if the allowable range is constrained. This immediately leads to difficulties since it is infeasible to show by test (as DO-178B is conventionally interpreted) that in the implemented system, all system requirements are satisfied under all possible parameter values. Secondly it is difficult to establish what the expected output from a test should be since the exact system state has evolved through learning, is unobservable and therefore unknown. These difficulties are explained at length in Jacklin [22, 23] and Schumann [24]. Our objective in this work therefore is to find ways in which these difficulties can be overcome.

4.2 Suggested Approach to Software Design Assurance of Adaptive Systems

In other works (e.g. [22]), the problem has been stated in terms of ‘what methods could be applied to comply with the assurance standards for software of DO-178B?’ and ‘what changes or

additions to DO-178B would be necessary to permit compliance?'. We suggest that the answers to these questions can best be arrived at by considering the methods by which we can write validated, verifiable HLR for adaptive systems. With this approach, the software assurance problem becomes more tractable since DO-178B/C defines a process to verify that operational code meets the previously stated and assumed to be correct and complete requirements as provided by the systems and safety development processes.

We suggest that DO-178B alone cannot provide adequate software design assurance but that DO-178C and supplements offers a promising avenue to accomplish adequate software design assurance if they are preceded by rigorous system design activities that generate validated and verifiable design constraints and requirements. This is largely because DO-178C and supplements offers a well defined methodology to partially shift the software design assurance burden from test to analysis. We have therefore considered the means and methods by which an applicant can derive and validate a complete and consistent set of verifiable adaptive system requirements expressed in formal or mathematical terms with well defined syntax and semantics that are amenable to modern analysis methods capable of providing a high level of design assurance.

We begin by asserting the following principles;

- Software design assurance alone is inadequate to assure the safe application of adaptive systems
- System safety objectives must be defined and captured. These form the basis of an unambiguous safety case
- The adaptive system must, by design, exhibit certain functional and safety properties to ensure an acceptable level of safety. These properties need to be established and captured as part of the system requirements capture process
- System level validation of the system properties is necessary to assure safe operation

We suggest that some of the newer techniques incorporated in DO-178C and the associated supplements augmented by system level considerations offer a possible means to overcome the difficulties of AS software design assurance. More specifically, we suggest that;

- More reliance be placed on verification by analysis or simulation rather than test
- Multi-layered verification methods involving a judicious combination of test, analysis and simulation of models should be used
- MBD techniques to capture system behavior in an analyzable form should be used
- FM analysis techniques should be used because the learned state space is too rich for test alone to provide adequate assurance and to predict expected test results
- Improved system safety analysis techniques should be used to derive system safety properties and that those properties be expressed in mathematical notations that are amenable to verification by FM

In addition, to make the verification effort manageable, techniques analogous to 'equivalency classes' to subdivide the learned state space and to 'structural coverage' analysis to measure the verification completeness of the learned state space are needed to complete the approach. Currently these are not known.

4.3 System Level Approaches to the Certification of Adaptive Systems

In view of the difficulties of showing software design assurance of AS by following only the processes of DO-178C we conclude that to assure the safe use of adaptive systems, additional prior work must be accomplished at the system level with the imposition of certain constraints on the architecture and permitted adaptation. The constraints result in the construction of desired system safety properties and requirements which, when verified, assure that the AS provides an acceptable level of safety. The system high level requirements (HLR) are thus established as part of the system design and safety processes of ARP-4754A and ARP-4761 through the construction of system functional and safety requirements. These properties must be written such that;

- The system-level HLR fully express the necessary system properties
- They are verifiable by one of the means identified in DO-178C and supplements

The system properties are then verifiable by formal or other methods, so that objective proof of conformance can be established. In our analysis of problematic DO-178C objectives, we make use of the system properties that exist as a consequence of satisfying the AS safety objectives.

The definition of requirement given in ARP-4744A implies that requirements are valid only if there are means by which they can be verified. Therefore requirements generation must be cognizant of the expected verification methods. Derived requirements, which develop throughout the development phase, should be fed back to the system and safety processes for validation. This is necessarily an iterative process because there are no known stopping criteria that can reliably determine completeness.

4.4 Defining the System Level Characteristics of an Adaptive System

In order to concretize our analysis, we have utilized the example architecture diagram of our target (Section 3) to define some of its salient system-level characteristics in terms of the system-level design and safety objectives that we consider essential to enable compliance with the objectives of DO-178B/C and other airworthiness standards.

These characteristics are inputs to the requirements generation process to be followed by the implementation activity. In our analysis of challenging DO-178B/C objectives, we make use of the system properties that exist as a consequence of satisfying these adaptive system safety objectives. The first step in our process is to define the "system safety objectives" that must be satisfied as part of systems requirements, design and verification and validation (V&V) processes. These are shown in Table 3 below.

Table 3 - System Safety Objectives for Adaptive Systems (developed with exemplar adaptive flight control system in mind)

	AS System Safety Objectives	Activities and Techniques for Satisfying the Objective
1	Assure adaptive algorithm stability & convergence (To be satisfied during development)	**Activities** • The following activities apply to both the adaptive system and the closed loop system: • Develop system level requirements to ensure stability & convergence • Define stability & convergence assumptions (e.g. linear plant dynamics, continuous time implementation) • Define stability & convergence constraints (e.g. operating condition (input space) limitations, learned state space limitations, maximum convergence time) • Define computational resource usage constraints • Define engagement/disengagement criteria with transient suppression • Define runtime monitors for detection of: ▪ Violation of assumptions or constraints ▪ Loss of stability or convergence ▪ Excessive usage of computational resources • Develop system level requirements that specify recovery behavior in the event of monitor alarm • Validate system requirements • Validate assumptions and constraints **Techniques** • Use of analytical models in combination with formal methods and automated tools to: ▪ Specify mathematically rigorous system requirements and design ▪ Develop proofs of stability & convergence (e.g., Lyapunov stability proof) ▪ Validate system requirements ▪ Generate expected results for requirements based testing ▪ Determine optimal adaption gains to balance stability vs. convergence time ▪ Perform automated synthesis of real-time monitors (Runtime Verification) • Use adaptive controller designs with proven stability and convergence properties (e.g. L-1 adaptive control)

2	Assure adaptive algorithm stability & convergence are satisfied (To be satisfied during real-time operation consistent with Rushby [25, 26])	**Activities** • Development activities for this system objective are covered in row 1 above **Techniques** • Confidence tool (confidence measure of NN weight convergence) • Envelop tool to predict and avoid regions of instability • Real time range limiter on learning state space • Real time range limiter on input space • Real time stability/convergence monitor with recovery logic if: ▪ Stability/convergence is observed to have failed ▪ Stability/convergence cannot be assured due to observed violation of assumptions or constraints ▪ Computational resource margins are observed to be violated
3	Assure adaptive algorithm actively controls only when appropriate	**Activities** • Development activities for this system objective are covered in row 1 above **Techniques** • Use of engage/disengage mechanisms: • VHM • Confidence/Envelop tools
4	Assure no adverse safety impact due to transients when AS is engaged/disengaged	**Activities** • Development activities for this system objective are covered in row 1 above **Techniques** • Fader functionality for smooth transition of conventional to adaptive control • Allow adaptation learning prior to AS trigger to eliminate the need for forcing function excitation to enable adaptation learning.
5	Assure adaptive algorithm does not adapt to noise or drift away from good solution when lacking useful command/response dynamics	**Activities** • Development activities for this system objective are covered in row 1 above **Techniques** • Use dead band on adaptive system inputs such that learning is allowed only when useful command/response dynamics are available • Use Bayesian reasoning to update the learning parameters only in the presence of sufficient excitation

The next step in our process is to work through all the objectives of DO-178B/C and, based on the principles described and the table above, assign methods and techniques that would provide satisfaction of those objectives. We followed the following methodology.

- List DO-178B/C objectives that are difficult to meet for the example AS we chose
- Understand what the DO-178B/C objective is asking for and why it's more difficult to meet for this example adaptive system versus a non-adaptive system
- List the adaptive system system-level functional and safety objectives. These objectives will feed the requirements creation process of an implementation
- List the methods that could be used to provide the evidence necessary for the adaptive system system-level objective

These steps have been followed for our example adaptive system and the results partially tabulated in Table 4 of Annex A. This table is presently incomplete. There remain some objectives for which we were unable to conclude whether there were special difficulties meeting DO-178C and therefore it remains for further work to complete this step. Note that in this table, we have merged objectives so that they do not appear in the table in numerical order.

In our table development, we have emphasized the system-level use of mathematical models and MBD to describe (i.e. specify) the complete system behavior in a form suitable for analysis. We have also emphasized the system and software level use of FM (and other FM-like techniques) to enable proofs of system safety and performance properties and to explore the full state space of the adaptive system within feasible simulation times - at least within the veracity of the mathematical description and the models.

Lastly, in order to bound the analysis problem further, we suggest that the use of equivalence classes is a possible means for classifying real numbered parameter and I/O variables into a finite and possibly small number of subsets. Establishing equivalence classes requires detailed knowledge of the application domain and target architecture.

5 Aspects of Tool Qualification for Adaptive Systems

Our recommended approach to software verification of adaptive systems utilizes MBD and FM tools. We do not envisage that there is any need to amend the guidance provided in the tool qualification section of DO-178C or of the tool qualification supplement DO-330.

6 Recommendations

The summary recommendation is that for the safe use and successful certification of adaptive systems a three step strategy of 1) a safety assessment to create a structured, objective safety case followed by 2) system design activities to create the corresponding safety requirements and 3) software design assurance using the latest standards. These steps must be supported by mathematically based formal or similar methods and model based design techniques. This approach is fully consistent with the current regulatory and standards framework. We caution that not all the necessary analysis and modeling tools are presently available and therefore further work is required before such an approach can be applied in a practical application.

We break this down into a number of more detailed recommendations. These are classified as validation and verification recommendations. Although we have focused on one particular

adaptive system type, these recommendations are likely to be applicable to a wider variety of systems.

6.1 Recommendations for Adaptive System Safety and Functional Requirements Derivation and Validation

We recommend the following steps for the creation and validation of system functional and safety requirements.

- Derive system safety objectives that are adaptive system application domain specific by the construction a structured, objective, evidence based safety case. The system level properties that need to exist essentially form the basis of a safety case. Certification of adaptive systems depends on both a system safety case (i.e., formal safety claims, arguments & evidence) in addition to system and software design assurance
- Derive system level properties that satisfy the safety objectives to assure an acceptable level of safety. Such properties will drive constraints on the system design
- Derive system requirements from system safety properties and objectives. Safety properties should be specified for;
    - When adaptation may be engaged (triggering)
    - Allowable learned state-space, implying that each learned parameter value be constrained to a known and verifiable range
    - Detection and fall-back if they exceed the allowable range, i.e. response when constraints are violated
- Embed system level properties and requirements in computer models suitable for automated analysis with qualified tools that can be further decomposed and passed down to the verification processes
- We recommend the use of ARP-4754A and DO-178C and supplements
- We recommend that ARP-4761 be updated to include a include a structured, evidence based safety case methodology

New regulatory policy instruments are needed to invoke DO-178C and an updated ARP-4761.

6.2 Recommendations for Adaptive System Requirements Verification

The verification process of the system level functional and safety requirements and the resulting derived requirements can be summarized by the following steps.

- We recommend that MBD techniques incorporating mathematical models with a well defined syntax and semantics should be used. This provides well defined input to subsequent analysis tools. The mathematical model should express requirements for safety properties e.g. controllability, overshoot, stability, convergence in our example AS
- The system behavior should be represented by discrete-time mathematical models if the implementation will be a digital system
- FM (and other FM-like techniques) or similar methods should be used to verify requirements (aka ‘behavior’). These methods are needed because;
    - Learned state space is too rich to adequately test, or for test to provide adequate coverage assurance
    - They allow construction of verification test cases and predict expected test results
    - They can provide proof of system safety and performance properties

    - Allow to explore the full state space within feasible simulation times
- The use of DO-178C and supplements is necessary. DO-178B is inadequate to provide sufficient software design assurance
- A multi-layered verification methodology will be necessary, involving all of the available techniques i.e. test, inspection and analysis (simulation)
- Need to ensure the system/safety properties (that form the basis of the safety case) remain intact during software requirements development and implementation. This implies traceability up and down the requirements and verification hierarchy
- The certification process will need to place increased reliance on some compliant but non-traditional means of compliance with certain DO-178C objectives, i.e. more reliance on verification by analysis, simulation and formal proofs of correctness rather than test
    - Accept analysis and inspection based verification results for partial certification credit
    - Use of outputs from the system processes
    - Use of system analytical models as software requirements
    - Use of system analytical models to perform software verification

7 Future Adaptive Systems Certification Research Needs

We have identified some gaps in methods and techniques that appear to be necessary to be able to perform the steps identified above.

- A new technique is needed, analogous to conventional equivalency classes, to classify the learned state spaces into a finite (and possibly small) number of equivalence regions or ranges to make verification manageable
- A new technique is needed, analogous to structural coverage analysis, to adapt the current notion of structural coverage to measure coverage completeness of the learned state space
- Further work is needed to complete the table in Annex A. In particular, it is still necessary to determine whether additional V&V methods and activities or system level constraints are needed to meet the DO-178C objectives as yet unaddressed
- We suggest that there be a study of the mapping of available MBD and FM tools to the AS application domain to identify capability and qualification gaps. In particular, there are presently capability gaps in showing in-target object code conformance to HLR and low level requirements (LLR) and in showing worst case execution time (WCET) objectives are met
- ARP-4761 presently provides only limited and incomplete guidance on the construction of structured, evidence based safety cases. We suggest that an update to this standard is needed
- Because of the specialized nature of FM and MBD techniques, these abilities are not well diffused into the developer community. We recommend that a more formalized process map be developed along with supporting user guides
- We recommend that the methodology we have outlined be demonstrated on an actual AS application and implementation. The application should be well defined and have the supporting mathematical models and code available that are readily translatable into FM/MBD constructs and be amenable to all levels of verification up to and including flight test, e.g. the L1 controller

8 Conclusions

We view the software design assurance problem for adaptive systems to be principally one of how to develop correct and complete requirements that define the necessary system functional and safety properties for safe use. These properties need to be established primarily by analysis. We do not think that certification of an adaptive system can be accomplished using a software design assurance methodology that is based principally on test since the test difficulty is insuperable unless guided by analysis. A set of system safety properties must first be specified and then design requirements and constraints must be imposed at the system level so that the safety properties are first assured by design and then passed down to the software design assurance process (DO-178C and supplements) for verification to show that they are implemented correctly. The verification of requirements can be accomplished by the use of FM and MBD system and software design and verification techniques as are currently envisaged by DO-178C and supplements. The methods we suggest are within the scope of the current 14CFR regulatory framework and no major change need be contemplated. An update to ARP-4761 to include a structured, evidence based safety case methodology and inclusion of this within the current framework is suggested. The principle compliance methodology changes we suggest are 1) attaching more emphasis to the system and safety development processes through the construction of a structured, evidence based safety case and 2) placing more reliance on system and software analysis using FM and MDB or similar techniques and less on test for gaining certification credit.

9 Annex A

Table 4 – DO-178B/C Objectives Applied to Adaptive Systems (developed with exemplar adaptive flight control system in mind)

DO-178B/C Objective/ Section	Objective Description	What is the Intent of 178B/C Objective(s)? [Ref: primarily DO-248C]	Why is the 178B/C Objective Difficult to Satisfy (for Chosen AS Example)?	Activities & Techniques Used To Satisfy 178B/C Objective (may involve use of system level properties)
A-2	**Software Development Process**	• **Systematic requirements and design decomposition** • **Complete capture of software behavior**		
A-2.1, A-2.4	High-level and low-level requirements are developed.	• Capture of high-level and low-level requirements	• Difficulty ensuring that system level stability and convergence properties are retained as the requirements are decomposed • The learned state space varies based on operating environment history. This can increase the difficulty in decomposing requirements to the next lower level that define complete software behavior (e.g., Sys > HLR > LLR > Source) with no unintended functionality • Difficulty assessing impact of derived software requirements on AS system safety (joint with systems and safety).	**Activities** • Decompose system level requirements to develop software requirements and code such that: ▪ System safety properties are retained through implementation ▪ System safety assumptions and constraints enforceable by software are satisfied by software requirements and implementation • System defined runtime monitors are satisfied by software requirements and implementation • Generate high-credibility evidence of compliance with next higher level of requirements **Techniques** • Reuse system level analytical models as software requirements (i.e., take software credit for system process outputs)
A-2.2, A-2.5	Derived high-level and low-level requirements are defined.	• Capture of all derived requirements • Ensure safety analysis is not compromised by improper implementation of safety-related requirements or introduction of new behavior not envisioned by the safety analysis		
A-2.6	Source Code is developed.	• Develop source code		

				• Apply software-level formal method techniques (e.g., model checking, compositional verification, static analysis, program synthesis, runtime analysis) to ensure: ▪ System level stability & convergence properties are retained in the software requirements and implementation ▪ System level assumptions & constraints allocated to software are properly decomposed and implemented
A-2.7	Executable Object Code is produced and integrated in the target computer.		• No more difficult	
A-3, A-4, A-5, A-6, A-7	**Software Verification Process**	• **Apply layers of verification** • **Ensure detection and removal of errors early in the development processes**		
A-3	**Verification of Software Requirements Process**	• **Ensure correct, consistent high-level requirements** • **Ensure full implementation of system requirements (completeness)**		
A-4	**Verification of Software Design Process**	• **Ensure correct, consistent low-level requirements** • **Ensure full**		

		implementation of HLR (completeness)		
A-5	**Verification of Software Coding Process**	• **Ensure correct, consistent source code** • **Ensure full implementation of LLR (completeness)**		
Compliance, Compatibility A-3.1, A-4.1, A-4.8, A-5.1, A-5.2, A-5.8	• Software high-level requirements comply with system requirements • Low-level requirements comply with high-level requirements • Software architecture is compatible with high-level requirements • Source Code complies with low-level requirements	• Ensure functional, performance and safety-related systems requirements are satisfied • Ensure derived HLR are justified and correctly defined • Ensure HLR are satisfied • Ensure derived LLR requirements are justified and correctly defined • Ensure software architecture does not conflict with HLRs • Ensure the Source Code is accurate and complete with respect to LLRs	• Difficulty verifying (via reviews) that system level stability and convergence properties remain intact through decomposition and implementation. • Difficulty verifying (via reviews) that system & safety requirements are decomposed into HLR & LLR correctly and implemented in source code correctly • Difficulty verifying (via reviews) that HLR, LLR and source code capture the complete evolving AS software behavior	**<u>Activities</u>** • Verify that the software requirements and code: ▪ Exhibit system safety properties ▪ Satisfy system safety assumptions and constraints allocated to software ▪ Satisfy runtime monitor next higher level requirements • Generate high-credibility evidence of compliance with next higher level of requirements **<u>Techniques</u>** • Apply software-level formal method techniques (e.g., model checking, compositional verification, static analysis, program synthesis, runtime analysis) to ensure: ▪ System level stability & convergence properties are retained in the software requirements and implementation ▪ System level assumptions & constraints allocated to software are properly decomposed and implemented

	• Source Code complies with software architecture • Parameter Data Item File is correct and complete.	• Ensure no undocumented functionality • Ensure Source Code matches architecture data flow and control flow • Ensure HLR are satisfied with respect to parameter data item file (e.g., database)		
Accuracy, Consistency A-3.2, A-4.2, A-4.9, A-5.6	• High-level requirements are accurate and consistent. • Low-level requirements are accurate and consistent. • Software architecture is consistent. • Source Code is accurate and consistent.	• HLRs are accurate, unambiguous, sufficiently detailed, consistent • LLRs are accurate, unambiguous, sufficiently detailed, consistent • Ensure correct relationship exists between the components of the software architecture • Ensure Source Code is correct and consistent with respect to stack usage, fixed point arithmetic overflow and resolution, resource	• Difficulty verifying (via reviews) accuracy & consistency attributes • AS learned state space makes it more difficult to determine and verify worst case critical computer resource usage and margins (memory, throughput, WCET etc).	**Activities** • Verify that the software requirements and code: ▪ Are accurate & consistent with respect to system safety properties ▪ Are accurate & consistent with respect to system safety assumptions and constraints allocated to software ▪ Properly implement system defined computational resource constraints and monitors **Techniques** • Apply software-level formal method techniques (e.g., model checking, compositional verification, static analysis, program synthesis, runtime analysis).

		contention, worst-case execution timing, exception handling, use of uninitialized variables or constants, unused variables or constants, and data corruption due to task or interrupt conflicts.		
Compatibility with Target A-3.3, A-4.3, A-4.10	• High-level requirements are compatible with target computer. • Low-level requirements are compatible with target computer. • Software architecture is compatible with target computer.	• Ensure compatibility with hardware (e.g., resource utilization)	• AS learned state space makes it more difficult to determine and verify worst case critical computer resource usage and margins (memory, throughput/WCET, etc).	**Activities** • Verify that the software requirements and code properly implement system defined computational resource constraints and runtime monitors **Techniques** • Apply software-level formal method techniques (e.g., model checking, compositional verification, static analysis, program synthesis, runtime analysis).
Verifiability A-3.4, A-4.4, A-4.11, A-5.3	• High-level requirements are verifiable. • Low-level requirements are verifiable. • Software architecture is	• Ensure HLRs can be verified • Ensure LLRs can be verified	• Difficulty assessing if HLR, LLR, source code can be verified (by test) for all AS configurations. • Difficulty verifying HLR, LLR, source code produce deterministic (predicable) behavior	**Activities** • Identify test and non-test verification techniques for requirements, architecture and source code. **Techniques** • Select appropriate software-level formal method techniques (e.g., model checking, compositional verification, static analysis, program synthesis,

	verifiable. • Source Code is verifiable.	• Ensure architecture can be verified • Ensure architecture is deterministic and predicable • Ensure source code can be verified		runtime analysis). **Notes:** • 1) Software aspects of certification of AS will likely require a greater reliance on verification by analysis or simulation for objectives related to verification by test. • 2) Our chosen AS is deterministic given fully defined initial conditions (including learned state space values).
Conformance to Standards A-3.5, A-4.5, A-4.12, A-5.4	• High-level requirements conform to standards. • Low-level requirements conform to standards. • Software architecture conforms to standards. • Source Code conforms to standards.	• Ensure HLRs are consistent with HLR standards • Ensure LLRs are consistent with design standards • Ensure architecture is consistent with design standards • Ensure Source Code is consistent with coding standards	• Might not be more difficult for AS	
Traceability A-3.6, A-4.6, A-5.5	• High-level requirements are traceable to system	<u>System requirements trace to HLR:</u> • Ensure HLRs fulfill system requirements	• Difficulty verifying that trace demonstrates complete requirements decomposition and implementation all	**<u>Activities</u>** • Verify that the software requirements and code: ▪ Exhibit system safety properties

	requirements. • Low-level requirements are traceable to high-level requirements. • Source Code is traceable to low-level requirements.	• Ensure all the system requirements (including safety requirements) allocated to software are incorporated in the HLRs. <u>HLR trace to system requirements:</u> • Identification of functionality not explicitly required by system requirements • Ensure that derived HLRs are captured, justified and fed back to safety process <u>HLR trace to LLR:</u> • Ensure LLRs fulfill HLRs <u>LLR trace to HLR:</u> • Identification of functionality not explicitly required by HLRs • Ensure that derived LLRs are captured, justified and fed back to safety process <u>LLR trace to Source Code:</u> • Ensure Source Code fulfill LLRs <u>Source code trace to LLR:</u> • Expose any source code functionality (intended or	intended functionality (i.e. complete behavior of AS) • Difficulty verifying that the trace demonstrates absence of unintended functionality	▪ Satisfy system safety assumptions and constraints allocated to software ▪ Satisfy runtime monitor next higher level requirements **<u>Techniques</u>** • Apply software-level formal method techniques (e.g., model checking, compositional verification, static analysis, program synthesis, runtime analysis) to ensure: ▪ System level stability & convergence properties are retained in the software requirements and implementation ▪ System level assumptions & constraints allocated to software are properly decomposed and implemented

		unintended) that is unsupported by the LLRs • Ensure unintended functionality is removed		
Algorithm Accuracy A-3.7, A-4.7	• Algorithms are accurate for HLR • Algorithms are accurate for LLR	• Ensure accuracy and behavior of HLR algorithms • Ensure accuracy and behavior of LLR algorithms	• Difficulty verifying (via reviews) that system level stability and convergence properties are retained through requirements decomposition and implementation.	**Activities** • Verify that the software requirements and code: ▪ Have accurate algorithms with respect to system safety properties ▪ Have accurate algorithms with respect to system safety assumptions and constraints allocated to software ▪ Have accurate algorithms with respect to runtime monitors **Techniques** • Apply software-level formal method techniques (e.g., model checking, compositional verification, static analysis, program synthesis, runtime analysis).
Partitioning Integrity A-4.13	Software partitioning integrity is confirmed.	• Ensure partitioning breaches are prevented or isolated	• Possibly no more difficult for AS	
Completeness, Correctness A-5.7	Output of software integration process is complete and correct.	• Ensure results of the integration process are complete and correct	• Possibly no more difficult for AS	
A-5.9	(Verification coverage. Combined with A-7.3, A-7.4)			

A-6	**Software Testing Process**	• **Ensure executable object code satisfies HLR and LLR** • **Ensure executable object code is robust**		
A-6.1, A-6.3	• Executable Object Code (EOC) complies with high-level requirements. • EOC complies with low-level requirements.	• Ensure EOC satisfies HLRs for normal range inputs • Ensure EOC satisfies LLRs for normal range inputs	• Difficulty developing normal range test cases for all possible input space and learned state space • Difficulty developing adequate set of robustness test cases to expose unintended functionality • Difficulty assuring software dynamic stability and convergence by test for all AS learned states • Verification by test is likely inadequate to show EOC is correct for all possible AS behavior	**Note:** • Software aspects of certification of AS will likely require a greater reliance on verification by analysis or simulation for objectives related to verification by test. • Certification process will likely need to allow for the use of multi-layered verification methods. **Activities** • Verify that the EOC: ▪ Exhibits system safety properties ▪ Satisfies system safety assumptions and constraints allocated to software ▪ Satisfies runtime monitor requirements • Generate high-credibility evidence of EOC compliance **Techniques** • Apply software-level formal method techniques (e.g., model checking, compositional verification, static analysis, program synthesis, runtime analysis) to ensure: ▪ System level stability & convergence properties are retained in the software
A-6.2, A-6.4	• Executable Object Code is robust with high-level requirements. • Executable Object Code is robust with low-level requirements.	• Ensure EOC is robust such that it can continue to operate correctly despite abnormal inputs and conditions • Ensure failure detection and recovery capabilities are effective and robust in mitigating hazards		

				requirements and implementation ▪ System level assumptions & constraints allocated to software are properly decomposed and implemented • Apply formal method techniques to develop normal/robust test cases and expected results for input space and learned state space. • Monte Carlo simulations • Need an analytical method for establishing "equivalence classes" for learned state space. • Continuity-based equivalency class partitioning
A-6.5	Executable Object Code is compatible with target computer.	• Ensure compatibility with hardware (e.g., resource utilization) • Ensure detection of target-related errors or compiler target-specific errors	• Learned state space makes it more difficult to test worst case resource utilization.	**<u>Activities</u>** • Verify that the software requirements and code properly implement system defined computational resource constraints and runtime monitors **<u>Techniques</u>** • Apply software-level formal method techniques (e.g., model checking, compositional verification, static analysis, program synthesis, runtime analysis).
A-7	**Verification of Verification Process**	• **Ensure thorough testing of the EOC** • **Ensure completeness of HLR and LLR testing requirements based test (RBT) coverage**		

		• **Ensure completeness of HLRs and LLRs (Structural coverage)** • **Ensure unintended functionality is exposed (Structural coverage)**		
A-7.1	Test procedures are correct.	• Ensure test cases were accurately developed into test procedures and expected results	• Difficulty predicting correct expected results covering the input space, learning state space and dynamic states (e.g., converging, converged).	**Activities** • Develop test/analysis/simulation cases into test/analysis/simulation procedures **Techniques** • Apply formal method techniques to develop normal/robust test cases and expected results for input space and learned state space.
A-7.2	Test results are correct and discrepancies explained.	• Ensure test results are correct and that discrepancies between actual and expected results are explained	• Possibly no more difficult for AS	
A-7.3, A-7.4, A-5.9	Test coverage of high-level requirements is achieved. Test coverage of low-level requirements is achieved. Verification of Parameter Data	• Ensure completeness of HLR test cases • Ensure completeness of LLR test cases • Ensure completeness of verification with respect to Parameter Data Item File (e.g., database)	• Difficulty developing normal range test cases for all AS behavior • Difficulty developing adequate set of robustness test cases to expose unintended functionality • Difficulty assuring software dynamic stability and convergence by test for	**Note:** • Software aspects of certification of AS will likely require a greater reliance on verification by analysis or simulation for objectives related to verification by test. • Certification process will likely need to allow for the use of multi-layered verification methods. Test coverage trace may need to be expanded to test, analysis, and simulation coverage trace.

	Item File is achieved.	elements	learned state space and input space.	**Activities** • Ensure complete test/analysis/simulation coverage trace **Techniques** • Augment testing with software-level formal method techniques (e.g., model checking, compositional verification, static analysis, program synthesis, runtime analysis) to ensure: ▪ System level stability & convergence properties are retained in the software requirements and implementation ▪ System level assumptions & constraints allocated to software are properly decomposed and implemented • Monte Carlo simulations
A-7.5	Test coverage of software structure (modified condition/decision) is achieved.	• Ensure completeness of HLRs and LLRs • Ensure unintended functionality is exposed • Ensure unreachable code is exposed • Ensure that the compiler does not inject functionality that was not specified in the source code • Ensure requirements are	• Structural coverage analysis insufficient to measure completeness of test/analysis/simulation of all possible input space and learned state space.	**Activities** • Ensure complete decision/statement coverage **Techniques** • Static analysis tools. Rely on FM proofs to verify all program path executions • Would still not be adequate for MC/DC or decisions unrelated to branching. ▪ Need an analytical method to measure verification coverage completeness of learned state space.
A-7.6	Test coverage of software structure (decision coverage) is achieved.			
A-7.7	Test coverage of software structure (statement coverage) is achieved.			

		sufficiently detailed (similar decision structure as the code)		
A-7.8	Test coverage of software structure (data coupling and control coupling) is achieved.	• Ensure test coverage with respect to the software architecture (specifically the data flow between software components and the control of software component execution) • Ensure a sufficient amount of hardware/software integration testing and/or software integration testing to verify that the software architecture is correctly implemented with respect to the requirements		
A-7.9	Verification of additional code that cannot be traced to Source Code is achieved.	• Ensure that the EOC is evaluated for any functionality added by the compiler • Ensure that compiler added functionality has no safety impact		
12.2	Tool Qualification	• Ensure tool provides confidence at least equivalent to that of the process(es) eliminated,	• Difficulty with tool qualification of development tools (TQL-1 through TQL-4) for the auto generation of	

		reduced or automated	adaptive source or object code. • Difficulty with tool qualification of verification tools (TQL-4 or TQL-5) intended to simulate all operating conditions or invoke complete (evolving) software behavior.	

10 References

[1] Kaminski, Paul, "Decadal Survey of Civil Aeronautics: Foundation for the Future," National Academies Press: Steering Committee for the Decadal Survey of Civil Aeronautics, National Research Council, 2006.

[2] RTCA, "DO-178B - Software Considerations In Airborne Systems And Equipment Certification," RTCA, Washington DC DO-178B, December 1, 1992.

[3] RTCA, "DO-178C - Software Considerations In Airborne Systems And Equipment Certification," RTCA, Washington DC DO-178C, December 13, 2011.

[4] RTCA, "DO-254 - Design Assurance Guidance For Airborne Electronic Hardware," RTCA, Washington DC DO-254, April 19, 2000.

[5] FAA, "Advisory Circular: Document RTCA DO-254, Design Assurance, Guidance For Airborne Electronic Hardware," FAA, Washington DC AC 20-152, June 30, 2005.

[6] FAA, "Final Policy Statement on Applying Advisory Circular 20-152, “RTCA, Inc., Document RTCA/DO-254, Design Assurance Guidance for Airborne Electronic Hardware,” to Title 14 Code of Federal Regulations, Part 23 Aircraft; PS-ACE100-2005-50001," Federal Aviation Authority, Washington DC, January 26, 2007.

[7] SAE, "Safety Assessment of Transport Airplanes in Commercial Service," SAE, Warrandale, PA ARP-5150, November, 2003.

[8] SAE, "Guidelines for Development of Civil Aircraft and Systems," SAE, Warrendale, PA ARP-4754A, December 2010.

[9] FAA, "Advisory Circular: Development of Civil Aircraft and Systems," FAA, Washington DC AC 20-174, September 30, 2011.

[10] FAA, "Advisory Circular: System Design and Analysis," FAA, Washington DC AC 25.1309-1A, June 21, 1988.

[11] SAE, "Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment," SAE, Warrandale, PA ARP-4761, December 1996.

[12] RTCA, "DO-331 - Model-Based Development and Verification Supplement to DO-178C and DO-278A," RTCA, Washington DC DO-331, December 13, 2011.

[13] RTCA, "DO-332 - Object-Oriented Technology and Related Techniques Supplement to DO-178C and DO-278A," RTCA, Washington DC DO-332, December 13, 2011.

[14] FAA, "Handbook for Object Oriented Technology in Aviation (OOTiA)," Federal Aviation Authority, Washington DC, October 26, 2004.

[15] RTCA, "DO-333 - Formal Methods Supplement to DO-178C and DO-278A," RTCA, Washington DC DO-333, December 13, 2011.

[16] RTCA, "DO-330 - Software Tool Qualification Considerations," RTCA, Washington DC DO-330, December 13, 2011.

[17] Woodham, Kurt, "Verification of Adaptive Systems Phase 1 Final Report," NASA Langley Research Center, Hampton, VA DOT/FAA/AR-xx/xx, February 10, 2012.

[18] Boeing Advanced Systems, "AFTI (Advanced Fighter Technology Integration)/F-111 Mission Adaptive Wing Briefing to Industry," AFTI/F-111 Mission Adaptive Wing Program Office,, Dayton, OH AFWAL-TR-88-308Z, October 1988.

[19] Xargay, Enric, Hovakimyan, Naira, Dobrokhodov, Vladimir, Kaminer, Isaac, Gregory, Irene M., and Cao, Chengyu, "L1 adaptive flight control system: Flight evaluation and technology transition," presented at the AIAA Infotech at Aerospace 2010, April 20, 2010 - April 22, 2010, Atlanta, GA, United States.

[20] Rockwell Collins (2008), (Accessed: 11/27/2012). *Rockwell Collins successfully controls and lands wing-damaged UAV*. Available: http://www.rockwellcollins.com/~/media//Files/Unsecure/News%20Archive/FY08/20080610%20Rockwell%20Collins%20successfully%20controls%20and%20lands%20wing-damaged%20UAV.pdf

[21] Busch, D., Bharadwaj, R., Enns, D., Kim, K., Pratt, S., Vechart, A., and Oppenheimer , M., "FAST Effector Health Management and Integrated Adaptive Guidance and Control," presented at the Integrated Systems Health Management (ISHM) Conference 2011, Boston, MA, July 19-21, 2011.

[22] Jacklin, Stephen A., "Closing the certification gaps in adaptive flight control software," presented at the AIAA Guidance, Navigation and Control Conference and Exhibit, August 18, 2008 - August 21, 2008, Honolulu, HI, United States.

[23] Jacklin, Stephen A., Lowry, Michael R., Schumann, Johann M., Gupta, Pramod P., Bosworth, John T., Zavala, Eddie, Kelly, John W., Hayhurst, Kelly J., Belcastro, Celeste M., and Belcastro, Christine M., "Verification, validation, and certification challenges for adaptive flight-critical control system software," presented at the Collection of Technical Papers - AIAA Guidance, Navigation, and Control Conference, August 16, 2004 - August 19, 2004, Providence, RI, United States.

[24] Schumann, Johann, Gupta, Pramod, and Jacklin, Stephen, "Toward verification and validation of adaptive aircraft controllers," presented at the 2005 IEEE Aerospace Conference, March 5, 2005 - March 12, 2005, Big Sky, MT, United States.

[25] Rushby, John, "Runtime Certification," presented at the Runtime Verification: 8th International Workshop, RV 2008 Selected Papers, Budapest, Hungary, March 30, 2008.

[26] Rushby, John, "How Do we Certify for the Unexpected?," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, ed: American Institute of Aeronautics and Astronautics, 2008.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
01- 06 - 2013	Contractor Report	

4. TITLE AND SUBTITLE

Final Report - Regulatory Considerations for Adaptive Systems

5a. CONTRACT NUMBER

NNL06AA05B

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Wilkinson, Chris; Lynch, Jonathan; Bharadwaj, Raj

5d. PROJECT NUMBER

5e. TASK NUMBER

NNL12AB32T

5f. WORK UNIT NUMBER

534723.02.02.07.10

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

NASA Langley Research Center
Hampton, Virginia 23681

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSOR/MONITOR'S ACRONYM(S)

NASA

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

NASA/CR-2013-218010

12. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Category 03
Availability: NASA CASI (443) 757-5802

13. SUPPLEMENTARY NOTES

Langley Technical Monitor: Kelly J. Hayhurst

14. ABSTRACT

This report documents the findings of a preliminary research study into new approaches to the software design assurance of adaptive systems. We suggest a methodology to overcome the software validation and verification difficulties posed by the underlying assumption of non-adaptive software in the requirements-based-testing verification methods in RTCA/DO-178B and C. An analysis of the relevant RTCA/DO-178B and C objectives is presented showing the reasons for the difficulties that arise in showing satisfaction of the objectives and suggested additional means by which they could be satisfied. We suggest that the software design assurance problem for adaptive systems is principally one of developing correct and complete high level requirements and system level constraints that define the necessary system functional and safety properties to assure the safe use of adaptive systems. We show how analytical techniques such as model based design, mathematical modeling and formal or formal-like methods can be used to both validate the high level functional and safety requirements, establish necessary constraints and provide the verification evidence for the satisfaction of requirements and constraints that supplements conventional testing. Finally the report identifies the follow-on research topics needed to implement this methodology.

15. SUBJECT TERMS

Adaptive systems; Assurance; Formal methods; Model based development; Verification

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			STI Help Desk (email: help@sti.nasa.gov)
					19b. TELEPHONE NUMBER *(Include area code)*
U	U	U	UU	46	(443) 757-5802