DOT/FAA/AR-xx/xx

Verification of Adaptive Systems

Air Traffic Organization
NextGen & Operations Planning
Office of Research and
Technology Development
Washington, DC 20591

DISCLAIMER
This draft document is being made available as a "Limited Release" document by the FAA Software and Digital Systems (SDS) Program and does not constitute FAA policy or guidance. This document is being distributed by permission by the Contracting Officer's Representative (COR). The research information in this document represents only the viewpoint of its subject matter expert authors.

The FAA is concerned that its research is not released to the public before full editorial review is completed. However, a Limited Release distribution does allow exchange of research knowledge in a way that will benefit the parties receiving the documentation and, at the same time, not damage perceptions about the quality of FAA research.

NOTICE

This document is disseminated under the sponsorship of the U.S. Department of Transportation in the interest of information exchange. The United States Government assumes no liability for the contents or use thereof. The United States Government does not endorse products or manufacturers. Trade or manufacturer's names appear herein solely because they are considered essential to the objective of this report. This document does not constitute FAA certification policy. Consult your local FAA aircraft certification office as to its use.

DRAFT

Technical Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
DOT/FAA/AR-xx/xx		

4. Title and Subtitle	5. Report Date
VERIFICATION OF ADAPTIVE SYSTEMS –FINAL REPORT	May 2013
	6. Performing Organization Code
7. Author(s) Chris Wilkinson, Jonathan Lynch, and Raj Bharadwaj (Honeywell International Inc.) Kurt Woodham (NASA Langley Research Center)	8. Performing Organization Report No.
9. Performing Organization Name and Address NASA Langley Research Center Mail Stop 130 Hampton, VA 23681-2199 Honeywell International Inc. 1985 Douglas Drive N Golden Valley, MN 55422-3935	10. Work Unit No. (TRAIS)
	11. Contract or Grant No. IA1-1073, DFACT-10-X-00008
12. Sponsoring Agency Name and Address	13. Type of Report and Period Covered September, 2010 – December 2012
	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

Adaptive software, which has the ability to change behavior at run-time in response to changes in the operational environment, system configuration, resource availability or other factors, has been an active research topic, but with limited use to date in the aviation domain. Progress in adaptive flight control systems and plans for using adaptive systems in the Next Generation Air Transportation System (NextGen), however, are compelling examination of requirements for verification of these systems for commercial applications. This report documents the findings of a two-phase research study of software assurance requirements for adaptive systems, especially from the perspective of RTCA/DO-178Band C. Phase 1 of the study was conducted by NASA Langley Research Center, and Phase 2 was conducted by Honeywell International Inc.

17. Key Words	18. Distribution Statement
Adaptive Systems, Neural Networks, Machine Learning, Verification, Assurance, Model Based Development, Formal Methods	This document is available to the U.S. public through the National Technical Information Service (NTIS), Springfield, Virginia 22161.

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified		

Form DOT F 1700.7 (8-72) Reproduction of completed page authorized

DRAFT

ACKNOWLEDGEMENTS

Work reported in this document was supported under Interagency Agreement IA1-1073, DTFACT-10-X-00008 between the FAA and NASA Langley Research Center. The work was divided into two parts, Phase 1 performed by NASA Langley Research Center, and Phase 2 performed by Honeywell International, Inc., under subcontract to NASA Langley. Completion of this research project in total would not have been possible without the support of Charles Kilgore, the FAA COTR, and Barbara Lingberg, FAA sponsor for the FAA's Software and Digital Systems Research Program.

DRAFT

TABLE OF CONTENTS

		Page
EXECUTIVE SUMMARY		viii
1.	INTRODUCTION	1
1.1	Overview of Phase 1 Activities	2
1.2	Overview of Phase 2 Activities	3
2.	TERMINOLOGY	4
3.	UNDERSTANDING ADAPTIVE APPROACHES	5
3.1	The Feedback Process	6
3.2	Life-Cycle Context	8
3.3	Learning Method	9
3.4	Role of adaptation	9
4.	USE OF ADAPTIVE ALGORITHMS AND THEIR ROLE IN ADAPTIVE SYSTEMS' EVALUATION	11
4.1	Neural Networks	11
4.1.1	Fundamental Concepts in Neural Networks	11
4.1.2	Example Application of a Neural Network	14
4.1.3	Observations on Neural Networks	16
4.2	Genetic Algorithms	17
4.3	Reflection/Autonomic Computing	17
4.4	Supporting Technologies	18
5.	ADAPTIVE CONTROLS AND THEIR USE IN ADAPTIVE SYSTEMS	18
5.1	Control System Overview	18
5.1.1	Controllability/Observability	19
5.1.2	Stability	19
5.1.3	Robustness	20
5.1.4	Robustness of Non-Adaptive Control Systems	20
5.1.5	Robustness of Adaptive Control	21
5.1.6	irworthiness Terminology Relevant to Stability/Robustness	22
5.2	Adaptive Control System Architectures	22
5.2.1	Model Reference Adaptive Control (MRAC)	22
5.2.2	Model Identification Adaptive Control (MIAC)	23
5.2.3	Model-Free Adaptive Control	24
5.2.4	Adaptive Augmentation	25

DRAFT

5.2.5 L1 Adaptive Control 25

6. INITIAL IDENTIFICATION OF SAFETY ISSUES 26

6.1 Impact of Life Cycle Context on Safety 26
6.2 Impact of the Role of Adaptation on Safety 27

6.2.1 Monitoring 27
6.2.2 Parameter Estimation for Performance/Efficiency 27
6.2.3 Parameter Estimation for Robustness 28
6.2.4 Control Reconfiguration 28
6.2.5 Planning 28

6.3 Safety Concerns for Adaptive Algorithms 29

7. INITIAL PHASE 2 ACTIVITIES 30

8. CURRENT CERTIFICATION GUIDANCE AND STANDARDS 33

8.1 ARP-4754A Guidelines for Development of Civil Aircraft and Systems 34

8.1.1. Recent Changes to ARP-4754 36

8.2 ARP-4761 Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment 40
8.3 Software Design Assurance 40

9. ADAPTIVE SYSTEM CERTIFICATION 42

9.1 Concerns on the Feasibility of Applying DO-178B to Software Design Assurance of Adaptive Systems 42
9.2 Suggested Approach to Software Design Assurance of Adaptive Systems 42
9.3 System Level Approaches to the Certification of Adaptive Systems 43
9.4 Defining the System Level Characteristics of an Adaptive System 44

10. ASPECTS OF TOOL QUALIFICATION FOR ADAPTIVE SYSTEMS 48

11. RECOMMENDATIONS 48

11.1 Recommendations for Adaptive System Safety and Functional Requirements Derivation and Validation 48
11.2 Recommendations for Adaptive System Requirements Verification 49

12. FUTURE ADAPTIVE SYSTEMS CERTIFICATION RESEARCH NEEDS 50

13. SUMMARY 51

14. REFERENCES 52

DRAFT

APPENDIX A--TERMINOLOGY A-1

APPENDIX B--LITERATURE SEARCH RESULTS B-1

APPENDIX C-- DO-178B/C OBJECTIVES APPLIED TO ADAPTIVE SYSTEMS C-1

DRAFT

LIST OF FIGURES

Figure		Page
1	Elements of the Feedback Process	7
2	Model of a Neuron	12
3	Neural Network Topology	12
4	Gradient Descent Learning Rules	14
5	Comparison of Fuel Measurement System Approaches	15
6	Closed Loop Control System	19
7	Control System with Gain Scheduling	21
8	Model Reference Adaptive Control	23
9	Model Identification Adaptive Control	24
10	Model-Free Adaptive Control	24
11	Adaptive Augmentation	25
12	Adaptive System Taxonomy	31
13	Exemplar Flight Control Architecture	32
14	Certification Process Flow and Applicable Standards	34
15	ARP-4754 – ARP-4754A Sections Mapping	37
16	In/Out Mapping of ARP-4754 and ARP-4754A	37

DRAFT

LIST OF TABLES

Table		Page
1	Role of Statistical Techniques and Representative Applications	10
2	Table 2. ARP-4754A Invocation	36
3	DO-178B Invocation	41
4	System Safety Objectives for Adaptive Systems	45

DRAFT

LIST OF ACRONYMS AND ABBREVIATIONS

AC	Advisory Circular (FAA)
AI	Artificial Intelligence
ANN	Artificial Neural Network
AS	Adaptive System
ASA	Aircraft Safety Assessment
ATC	Air Traffic Control
ATM	Air Traffic Management
CFR	Code of Federal Regulations
COTS	Commercial Off The Shelf
CPU	Central Processing Unit
CRI	Certification Review Item or Certification Requirement Item
CS	Certification Specification
DAL	Design Assurance Level
DER	Designated Engineering Representative
EASA	European Aviation Safety Agency
EOC	Executable Object Code
EuroCAE	European Organization for Civil Aviation Equipment
FAA	Federal Aviation Administration
FC	Failure Condition
FDAL	Function Development Assurance Level
FFS	Functional Failure Sets
FHA	Functional Hazard Assessment
FM	Formal Methods
HLR	High Level Requirements
IDAL	Item Development Assurance Level
IP	Issue Paper
JAA	Joint Aviation Authority
LaRC	Langley Research Center
LLR	Low Level Requirements
MBD	Model Based Design
MRAC	Model Reference Adaptive Control
MIAC	Model Identification Adaptive Control
NAS	National Airspace System
NASA	National Aeronautics and Space Administration
NextGen	Next Generation Air Transportation System
NN	Neural Network
OOT&RT	Object-Oriented Technology and Related Techniques
PASA	Preliminary Aircraft Safety Assessment
PRA	Probabilistic Risk Assessment
PSSA	Preliminary System Safety Assessment
RBT	Requirements Based Test
SESAR	Single European Sky ATM Research
SMT	Satisfiability Modulo Theory

DRAFT

SSA	System Safety Assessment
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

DRAFT

EXECUTIVE SUMMARY

This report documents the Phase 1 and Phase 2 accomplishments on the Verification of Adaptive Systems research task, under Interagency Agreement IA1-1073, DTFACT-10-X-00008 between the FAA and NASA Langley Research Center. This research study addresses verification and safety assurance issues for the use of adaptive systems such as those planned for the Next Generation Air Transportation System (NextGen) and the National Airspace System (NAS) air traffic control (ATC) system. In this research task, an adaptive system was defined as a system that changes behavior based on an active feedback process in the presence of changes in the system or its environment. Requirements for system safety assurance are based on the general concept that correct behavior of a system can be specified, predicted, and verified prior to operation. Consequently, any proposed use of adaptive systems that violate that concept raises issues that must be evaluated. The goal of this research was to conduct a preliminary examination of what is necessary to provide sufficient assurance that an adaptive system is safely used in an aircraft product from a software perspective.

The research in both phases of this effort targeted understanding the applicability of existing software assurance requirements, especially those in RTCA/DO-178B "Software Considerations in Airborne Systems and Equipment Certification," and the recently released update to DO-178C with its corresponding supplements, for adaptive systems. Work for Phase 1 was performed by NASA Langley Research Center, and the Phase 2 work was performed by Honeywell International, Inc., under subcontract to NASA Langley.

The Phase 1 effort focused on understanding the state of the art in machine learning and the mechanisms that could cause an aircraft system to adapt, or change behavior, in response to change in its environment. Understanding the mechanisms used for adapting is essential to understanding the impact on software assurance. Much of the work in the initial phase consisted of gathering information on the broad field of machine learning, how machine learning is used to enable a system to adapt, especially with respect to feedback processes, and where machine learning is being used in various domains, with particular interest in controls applications. Research for Phase 1 is reported in terms of five areas: 1) terminology; 2) understanding adaptive approaches; 3) the use of adaptive algorithms and their role in adaptive systems' evaluation; 4) adaptive controls and their use in adaptive systems; and 5) initial identification of safety issues.

In Phase 2, the disparate information on different type of adaptive systems developed under Phase 1 was condensed into a useful taxonomy of adaptive systems. As evident from the taxonomy, the wide range of factors relevant to adaption makes it clear that the applicability of the DO-178C objectives likely will differ depending on the type of adaptive system. Hence, determining the applicability of the current software assurance process is extremely difficult for the general case (that is, for adaptive systems in general), but possible for a specific adaptive system. Consequently, the Honeywell team examined an exemplar adaptive system and evaluated how the output of that controller can be predicted and verified in compliance with system safety and assurance standards. A significant product of this evaluation is a table, provided in Appendix C, that describes the impact of the exemplar adaptive system on each DO-

DRAFT

178C objective. In addition, a number of system-level objectives were identified that may be necessary to ensure adequate verification of an adaptive system is possible. The importance of considering adaptive systems starting at the system level is discussed, along with recommendations for follow-on work in adaptive system safety and verification requirements.

DRAFT

1. INTRODUCTION

In the *Decadal Survey of Civil Aeronautics: Foundation for the Future* [1] the National Research Council identified *intelligent and adaptive systems* as one of the five common threads for the "51 high-priority R&T challenges". Generally speaking, adaptive systems are defined as those that have the ability to change behavior in response to changes in their operational environment, system configuration, resource availability or other factors. Adaptive systems have been used effectively in a number of application domains, from industrial plant control to missile guidance, but not civil aviation. That is expected to change. The decadal survey explicitly identified adaptive systems (AS) technologies to be the key enablers 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. These systems are attractive for several reasons. There are adaptive systems that have the ability to detect, anticipate, and prevent failures and reconfigure various aircraft systems (e.g., displays or controls) in response; some that simply improve or optimize performance in a changing operational environment; and others that can detect the degradation in performance due to failure or damage. Another push for use of adaptive systems is coming from the growth in national air transport system capacity. The NextGen Integrated Work Plan [2], for example, describes "net-enabled adaptive control of ground, airborne and satellite weather observation sensors in real time" as an enabling capability to meet needs for improved weather observations. Adaptive systems are also 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).

The use of advanced computational techniques, such as those that underlie adaptive systems, is not a new topic in the aviation domain. In 1994, the FAA published a chapter in their Digital Systems Validation Handbook titled "Artificial Intelligence with Applications for Aircraft" [3]. Artificial intelligence (AI) is a broad and rapidly expanding field of technology "devoted to computer programs that will mimic the product of human problem solving, perception, and thought" [3]. The handbook chapter provided an overview of AI technology, focusing on expert systems, and identified potential certification issues for aviation systems that would use those technologies. At that time, expert systems were intended to automate already known procedures, and serve as assistants or advisors, instead of primary decision tools. Today, expert systems are safely used in that capacity in aviation applications.

Adaptive systems, however, have succeeded expert systems as the next AI technology for aviation applications. Adaptive technologies, such as neural networks, can be introduced into the design of a system to achieve a goal such as enhancing performance or efficiency, maintaining desirable behavioral traits, such as robustness, or responding to changes in the system or its environment. Research supported by Eurocontrol investigated a neural network-based system for automatic recognition and diagnosis of safety-critical, non-nominal events" in air traffic management (ATM) for improving safety monitoring for the Single European Sky ATM

DRAFT

Research (SESAR) initiative [4].

Adaptive systems "learn" as they execute, thus exhibiting behavior that can be less predictable than traditional avionics systems. Because requirements for system safety assurance are based on the concept that correct behavior of a system can be specified, predicted, and verified, any use of adaptive systems in civil applications poses challenges in assuring safety by means of traditional safety assurance methods and procedures. This includes understanding the impact of adaptation on system requirements and design, as well as software implementation and verification, since adaptation is ultimately realized through software.

The primary aim of the Verification of Adaptive Systems task was to develop an understanding of the ramifications of adaptive systems on software assurance. The task also aimed, to the extent possible, to develop a rational and practical approach for the assurance of flight software that uses adaptive techniques, potentially including approaches targeted at the system level. This report documents the results of the two phases of research activity to accomplish those aims.

1.1 OVERVIEW OF PHASE 1 ACTIVITIES

Work on Phase 1 was performed by NASA Langley Research Center. Phase 1 research focused on developing an understanding of the state-of-the-art in adaptive systems technology, especially machine learning, and how adaptive technology is used or proposed to be used in aviation applications, including controls. The following four objectives were defined for Phase 1:

Objective 1: Provide definitions of terminology associated with verifying adaptive systems in a safety-critical airborne environment, e.g., adaptive system, neural net, adaptive software, artificial intelligence, and deterministic.

Objective 2: Describe contrasting characteristics of adaptive systems and deterministic systems including relative benefits, strengths, and weaknesses.

Objective 3: Investigate the differences between an adaptive approach to system development and a deterministic approach, and their effects on system and software verification.

Objective 4: Identify safety issues when an adaptive, non-deterministic system approach is used and propose mitigation techniques to address these in a safety-critical airborne environment.

The intent of Phase 1 was to lay the groundwork necessary to identify the differences between conventional and adaptive systems from both a requirements and design perspective, and subsequently identify any unique software safety considerations that would not be addressed using existing assurance processes, especially DO-178B [5]. As such, much of the Phase 1 effort involved gathering information about machine learning and current uses of adaptive systems in industry, and trying to develop a cogent set of terminology associated with use of machine learning in aviation applications.

DRAFT

Sections 2 through 6 of this report document the results of the Phase 1 effort. Section 2 provides an overview of terminology issues for adaptive systems. Terms and definitions relevant to adaptive systems are listed in Appendix A. Section 3 describes fundamental aspects of adaptive approaches, including strengths and weaknesses, with special emphasis on feedback processes. In section 4, adaptive algorithms are discussed, including neural networks, genetic algorithms, and reflective programming. Different approaches to adaptive control are presented in Section 5. Then, section 6 provides an initial assessment of safety issues for adaptive systems. Section 7 contains a summary of the Phase 1 work in preparation for Phase 2. The results of the literature search are provided as a bibliography in Appendix B.

1.2 OVERVIEW OF PHASE 2 ACTIVITIES

Work in Phase 2 was performed by Honeywell International Inc. The Honeywell team started with the foundational work in Phase 1, then focused Phase 2 activities on determining the extent to which existing guidance in DO-178B/C[1] and associated supplements can provide the basis for assurance of adaptive systems, where additional or alternate objectives and activities might be necessary, and recommendations for additional research. Phase 2 objectives were to:

Objective 5: Maximize the current use of RTCA/DO-178B/C. Where aspects of adaptive systems cannot be approved using DO-178B/C, provide recommendations for alternate methods to be considered, including viability of these methods using current technology or as areas where additional research may be necessary.

Objective 6: Make recommendations for the safe use of adaptive systems, especially, being planned for the potential future implementation of adaptive systems for the NextGen and NAS ATC.

Objective 7: Provide effective outputs that can be used by the FAA for the development of FAA policy, guidance, and training.

Sections 7 through 12 of this report document the results of Honeywell's support for Phase 2. Section 7 provides a transition from the Phase 1 activities to Phase 2, including a helpful taxonomy of adaptive systems, based on the different adaptive system types and architectures. Section 8 provides an overview of current certification guidance and standards, including those for system safety and design assurance. In section 9, concerns about the feasibility of applying DO-178B/C to adaptive systems are enumerated and demonstrated by analysis of the objectives against a particular adaptive system. The analysis shows the implications of adaptation on activities and objectives at the system level. Next, section 10 touches on aspects of tool qualification for adaptive systems. Section 11 contains the recommendations for adaptive system safety, and section 12 identifies some continuing research needs.

[1] During the course of this task, the RTCA/DO-178B document was updated to DO-178C [6], and 4 supplementary documents (covering Object Oriented Technology, Model-Based Development, Formal Methods, and Tool Qualification) were approved by the RTCA. The Phase 2 activities considered the changes to those documents in the course of their research. The term DO-178B/C is intended to indicate that consideration.

The report concludes with a brief summary of the research effort.

2. TERMINOLOGY

As with many technologies, especially those with application in many diverse domains, there often lacks a standard vernacular used consistently across those domains. This is certainly the case with adaptive systems. Work on this task started with an effort to define terminology associated with adaptive systems, including terms such as adaptive system, neural net, adaptive software, artificial intelligence, and deterministic. It did not take long to realize that the terms often used in discussing adaptive systems in the aviation domain do not reflect the current state of the art in machine learning. Consequently, the simple task of defining terminology evolved into an effort to explore and understand the burgeoning world of machine learning and its applicability to aviation applications.

Throughout this research effort, a list of relevant terms and definitions were compiled while reviewing source material obtained through a broad literature search. The literature search for this effort culminated in over 206 references, including textbooks, conference proceedings, journal publications, standards and guidelines, industry papers, and academic sites, as well as other on-line resources. Appendix A contains a list of terms and definitions, and the results of the literature search are listed in a bibliography in Appendix B.

A few terms and definitions are worth special mention to provide the context for the remainder of this report. The first term is adaptive system. For this study, an adaptive system is one whose behavior changes in response to an active feedback process to achieve a goal in the presence of changes in the system environment. That environment might be the computational environment, including the components of the computing platform, such as middleware, or might be the physical or external system in which the computer program operates. For airborne systems, changes to the environment might include change to the physical structure of the aircraft or change in weather.

Behavior change, in response to an adaptive technique such as a neural network, is not intended to imply behavior change only when the system is operating in service. Behavior changes might be explored during the design phase of a system, where an adaptive algorithm is used to help define the behavior of a system. In this case, at the end of the system design phase, the design can be fixed such that the feedback mechanism is no longer needed and the behavior does not adapt in real time operation of that software. In other cases, the behavior change from the feedback of an adaptive system may take place on a continuous basis in real-time operation (referred to as a fully adaptive system), or only periodically (referred to as a partially adaptive system).

The definition of adaptive system used in this report does not include the term "deterministic" or its complement "non-deterministic", either of which is often used to attempt to distinguish between a system that adapts and one that does not. In Appendix A, a deterministic system is defined as one in which no randomness is involved in the development of future states of the system. Given the same input, the future state of a deterministic system can be precisely

determined or predicted from knowledge of an initial system state and a predictable sequence of intermediate states. In theory, determinism precludes the existence of randomness that influences the intermediate states. A system influenced by random processes would be non-deterministic, as the future state of the system cannot be uniquely predicted from a defined sequence of intermediate states.

In practice, all physical systems have random processes or uncertainties at some level. For instance, there is likely to be noise in system electronics or in sensor data used within the system. Being predictable does not necessarily imply that a deterministic system is free from any random processes or other uncertainties; rather, the implication is that these factors do not cause appreciable variations in the observed behavior of the system. Depending on the internal system processes (such as filters applied to the sensor data), these effects may not be detectable at the system interface, or their impact on the system behavior may be negligible. For example, consider the case of multithreaded programs [7].

> Non-determinism, inherent in threaded applications, causes significant challenges for parallel programmers by hindering their ability to create parallel applications with repeatable results….
>
> Application developers rely heavily on the fact that given the same input, a program will produce the same output. Sequential programs, by construction, typically provide this desirable property of deterministic execution. However, in shared memory multithreaded programs, deterministic behavior is not inherent. When executed, such applications can experience one of many possible interleavings of memory accesses to shared data. As a result, multithreaded programs will often execute non-deterministically following different internal states that can sometimes lead to different outputs.

In this case, threaded programs can be considered deterministic if the uncertainties associated with concurrency issues are properly considered in the software design. While the exact central processing unit (CPU) cycle occupied by a particular floating point operation may vary dramatically frame to frame, the important factor for deterministic behavior will likely be that the entire computation is complete before the frame deadline.

For adaptive systems, the output may be deterministic or non-deterministic, depending on where and how the feedback process is used. Because the terms deterministic and non-deterministic can cause more confusion than add clarity when discussing adaptive systems, those terms are not used in the remainder of this report. The issues of concern with respect to software assurance are related to having sufficient understanding, predictability, and verification of the effect of the adaptation; hence, this report focuses on understanding that process.

3. UNDERSTANDING ADAPTIVE APPROACHES

"Adaptive systems can be characterized by how aggressively they attempt to adapt to the unexpected. In the simplest case, adaptive systems behave like a complex curve-fitting or

pattern-matching mechanism. In more complex cases, they are designed to learn continuously from their environments." [8] There are at least four factors important to understanding adaptive approaches from this perspective, and understanding the issues relevant to assurance of systems using these approaches. These factors include:

- the feedback process
- the system life cycle context where adaptation is actively used
- how learning takes place
- the role of the adaptive approach

3.1 THE FEEDBACK PROCESS

The working definition for adaptation includes an active feedback process to modify behavior towards a goal. The notion of active feedback provides a primary distinction between what is truly an adaptive process and a process whose response is preconfigured. For instance, an on-board process may select a set of parameters through a table lookup function based on airspeed and ambient temperature. The behavior of the supported process changes as new parameters are pulled from the table and used in calculations; however, the change in behavior is driven only by the variables used in selecting parameters that were preconfigured through a priori analysis.

An adaptive system, on the other hand, essentially performs in-situ design iterations by monitoring the performance of the system against performance goals, and making incremental changes in its parameters until a reasonably optimal configuration is achieved. These updates are generated based on the actual response of the system and not solely on design assumptions and anticipated environmental conditions made in a priori analysis. In this case the active feedback process can respond to unanticipated events or conditions, resulting is systems that may be more efficient or robust than their non-adaptive counterparts.

Given the importance of an active feedback process in identifying a system as adaptive, it is helpful to take a more extensive look at feedback loops. In [9], four elements of feedback processes are defined explicitly, as shown in Figure 1.

Discussing active feedback in terms of these four elements provides a way to contrast between adaptive systems and non-adaptive systems. Non-adaptive systems have pre-determined actions that are based on a priori analysis of the assumed system characteristic operating in an assumed environment. Consequently, the “decide” process for a non-adaptive system would select a pre-defined response based on the conditions determined by “collect” and “analyze.” Conversely, an adaptive process actively refines parameters, and then evaluates their influence on performance through the subsequent act/collect/analyze processes. That is, adaptive systems include active refinement of parameters through the feedback process, whereas non-adaptive systems use either fixed parameters or parameters that are passively selected from a pre-configured set.

DRAFT

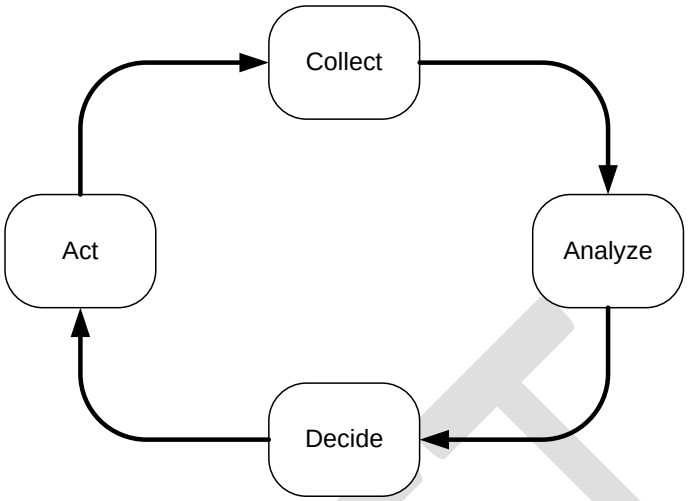

Figure 1. Elements of the Feedback Process

Certain strengths and weaknesses of adaptive systems as compared with conventional systems can be identified from this perspective. The primary strength of adaptive systems lies in their ability to refine their behavior based on actual in-situ experience, rather than assumed system or environmental properties. For conventional systems, design assumptions are typically conservative in order to assure that the system is robust against uncertainties. This in turn will often lead to sub-optimal performance as the system design is a compromise over a range of conditions, rather than optimized for a specific operating point. The conventional tradeoff between robustness and performance is partially eliminated, then, through the use of a system that adapts in operation, because it can be both robust to changes in the system or its environment and optimized for performance in its current conditions.

While this offers attractive benefits, it does not come without some weaknesses that are cause for concern. Convergence of the active feedback process must be evaluated carefully. Of particular concern is whether the active feedback process can find a globally optimal solution (that is, find the best solution out of all possible candidates), within a time-span that does not compromise system safety. This often becomes a trade-off between time-to-convergence and achieving a globally-optimal solution, and this trade-off has been the focus of a substantial amount of adaptive systems research.

Related to this concern and also relevant to the active feedback path is the concern that the analysis and decision elements of the feedback process are focusing on the correct features within the feedback signal. For instance, it is possible for these elements to focus the adaptation on noise rather than on the underlying signal that represents the true dynamics of the system, without sufficient attention to mitigations to prevent this.

Finally, a strength of a conventional approach is that the critical performance parameters of the system can be predicted (notwithstanding the accuracy of those predictions) for any given system configuration and environmental condition, whereas the performance of an adaptive system can only be predicted if the learning history of the system is known as well.

DRAFT

3.2 LIFE-CYCLE CONTEXT

Identifying where the adaptation occurs in the system life-cycle is an important aspect of adaptive systems. Adaptive systems can be divided into those that use adaptive approaches for development only, typically to mature a design, and those whose active feedback system is active in real-time in operation.

One benefit of using an adaptive approach during the design phase is that a detailed understanding of the underlying physical process may not be necessary for the adaptive approach to converge on an acceptable design solution. For instance, if data is available to describe the intended input/output behavior of a complex interaction, it may be feasible to train a neural network to replicate this behavior to a specified accuracy. In this case, a mathematical model of the underlying process, based on principles of physics, thermodynamics, or other disciplines, does not need to be derived. This provides a major strength of using an adaptive approach to design, as it provides a means for developing representations of complex systems for which derivation of a mathematical model by hand may be prohibitively difficult or impossible.

By the same token, a weakness of using an adaptive approach to development may stem from this ability to circumvent the need for a detailed understanding of an underlying process: the outcome of the underlying process is now encoded in the input and output data used to train the adaptive process, and the structure of the algorithm that is trained on this data bears little to no resemblance to the actual process being modeled. This leads to a lack of traceability between the resulting algorithmic design and the underlying process being modeled. Additionally, it underscores the necessity that the intended behavior be fully captured within the data used to train the adaptive system as part of development.

3.3 LEARNING METHOD

Many of the adaptive approaches identified in this study fall under the auspices of machine learning techniques. Machine learning is a branch of AI concerned with the development of algorithms that allow computers to evolve behaviors based on observing data and making statistical inferences about that data. A major focus of machine learning research is to automatically recognize (learn or be trained to recognize) complex patterns and make intelligent decisions based on data. As the computational power of computers has increased, so has the ability of algorithms to evaluate large amounts of data to automatically recognize complex patterns, to distinguish between exemplars based on their different patterns, and to make predictions. Many advances in adaptive controls algorithms, for example, are based on the ability to rapidly process data representing the aircraft's environment and make similarly rapid changes in parameters intended to maintain stability when that environment changes unexpectedly.

DRAFT

Learning is accomplished through inductive inference, that is, making predictions based on observations. These predictions are based on observing data that represents incomplete information about some statistical phenomenon and generalizing it to rules and making predictions on missing attributes or future data. [10] The process of learning from observations is often characterized by 3 different types of learning: supervised, unsupervised, and reinforcement learning.

- Supervised learning (or learning by example) is done with pre-determined sets of data representing input and response relations. The machine learning algorithm generates a model in terms of a mathematical function of the relationship between input data and response based on the pre-determined training. This model can be used to predict the response to input that was not included in the training data. Training data typically consists of examples in the form of an input value(s) paired with a desired output.

- Supervised learning can also occur in an operational context if a model is provided that can describe the desired system response. Comparing the desired against the actual response provides a means for measuring the performance of the adaptive system. Using this performance measure in an active feedback provides a way for the adaptive system to perform supervised learning while in operations.

- Unsupervised learning applies statistical techniques to identify non-obvious relationships within the data. In this approach, the learning algorithm is not given any target information to guide it nor is the algorithm given feedback based on previous decisions/actions. In a sense, unsupervised learning can be thought of as finding patterns in the data above and beyond what would be considered pure unstructured noise.

- Reinforcement learning involves mapping situations to actions, so as to maximize a cumulative reward obtained through a number of intermediate actions. Reinforcement learning is often called “learning from experience” or “learning through interaction”. The learner is not told which actions to take, as in supervised learning, but instead must discover which intermediate actions yield the best results. In the most interesting and challenging cases, actions may affect not only the immediate reward but also the next situation and, through that, all subsequent rewards. These two characteristics: trial-and-error search, and delayed reward, are often referred to as “exploration” and “exploitation” and are the two most important distinguishing features of reinforcement learning.

In general, of the three types of learning methods, supervised learning poses the fewest challenges with respect to verification.

3.4 ROLE OF ADAPTATION

Machine learning algorithms that enable adaptation are statistics-based algorithms, designed to identify complex patterns from data and make decisions based on data. The statistical task generally has one of four purposes. These can roughly be associated with particular types of application domains, as shown in Table 1.

Table 1. Role of Statistical Techniques and Representative Applications

Role	Definition	Applications
Regression	Identifies mathematical relationships among variables	• Adaptive Controls • Continuous Models • Parameter Estimation
Classification	Classifies input data into one of a number of discrete classes	• Monitoring • Situational Awareness
Clustering	Identifies similarities between features of the data and groups similar data items into clusters with similar features.	• Monitoring • Situational Awareness • Data Post-Processing
Optimization	The selection of a "best" element from some set of available alternatives	• Planning • Path/Trajectory Analysis

Regression: Many application domains, such as controls or sensor processing, require an accurate mathematical relationship of the physical system, its environment, or some process within the system. When the exact relationship is unknown, a conservative representation is often used due to uncertainties or approximations that are necessary to reduce the complexity of the mathematical relationship to a reasonable level. In other instances, the process, system, or environment is too complex for a mathematical expression to be derived from physical principles associated with dynamics, thermodynamics or other related disciplines. Some adaptive techniques such as neural networks provide a way for an accurate mathematical relationship to be generated from heuristic data: representative input/output data is processed using statistical methods to converge on a mathematical representation of the data. The resulting algorithm can then be fixed if it is developed exclusively as part of the design process, or the adaptive process can continue to refine the representation in operations. Continuous models developed using regression analysis are often used in adaptive controls and parameter estimation applications. Typically this involves the use of supervised learning.

Classification: Classification is closely associated with regression; however, the statistical methods used for classification seek to place each input data item into one of a finite set of possible outcomes. Statistical methods used for classification produce discrete models that bin the results, whereas regression analysis produces continuous models. Some statistical methods used for classification generate new classes if a suitable match is not found for a given input. In other approaches, the set of possible outcomes is fixed in advance.

Classification can be used for situational awareness where inputs are classified according to features relevant to the monitor. Classification can also be used for pattern recognition such as image processing, where a best match is selected from an image database on which the adaptive approach is trained. Classification typically involves supervised learning, but can also include unsupervised learning for instances where the adaptive system is allowed to generate new classifications.

Clustering: Clustering analysis assigns a set of objects into groups, called clusters, so that the

objects in the same cluster are more similar, in some sense or another, to each other than to those in other clusters. It is a statistical technique that is particularly powerful in identifying relationships in the data that are not otherwise obvious through conventional methods of data analysis. Clustering is often used as a data post-processing technique to identify trends or indicators. Some forms of clustering are applicable to in-situ real-time health monitoring environments where they may be able to identify precursors to adverse conditions. Clustering generally uses unsupervised learning.

Optimization: Optimization is a statistical method that seeks to identify the best solution to a given goal while taking into account constraints on the solution. These solutions typically involve a number of intermediate actions, and the optimization method uses a balance between exploration of possible intermediate actions and exploitation of intermediate actions that contribute to the optimal overall solution. Optimization typically uses reinforcement learning in developing solutions, and is particularly well-suited for applications such as path planning and trajectory analysis.

4. USE OF ADAPTIVE ALGORITHMS AND THEIR ROLE IN ADAPTIVE SYSTEMS' EVALUATION

This section discusses some of the more common approaches or types of algorithms used to implement the statistical analyses used for an adaptive system.

4.1 NEURAL NETWORKS

Of the myriad approaches used to enable a system to adapt, neural networks are likely the most prevalent and well known. A neural network, or artificial neural network (ANN), is a mathematical model, inspired by the structure of biological neural networks that processes information through an interconnected group of nodes called neurons. In practical terms, neural networks are non-linear statistical analysis or decision making tools. They can be used to model complex relationships between inputs and outputs or to find patterns in data. Neural networks can be constructed using supervised, unsupervised, or reinforcement learning.

4.1.1 Fundamental Concepts in Neural Networks

All ANNs shares some common terminology and features; however the wide variety of specialized configurations and features preclude an exhaustive coverage of all ANN terminology in this paper. This section cites glossary definitions mostly from [11, 12] as representative of definitions consistent with all sources reviewed as part of the literature search.

A neuron is a basic computation element of the neural network [12], consisting of a weighted sum of input connections, passed through an activation function (also called a threshold function), that is typically nonlinear, such as a hyperbolic tangent, or another sigmoid (S-shaped) function, as shown in Figure 2 [13].

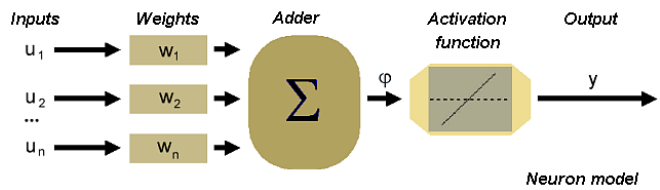

Figure 2. Model of a Neuron

The adaptive feedback process for ANNs refines each weight, defined as the "numerical values attached to specific neuron inputs to indicate significance." [12]

The topology of the ANN, which is the manner and organization in which the neurons are connected together [11], varies significantly for different types of ANNs; however, a common form contains an input layer of neurons (f_1 through f_3 in the following diagram, taken from [14]), one or more hidden layers (f_4 and f_5), and an output layer (f_6):

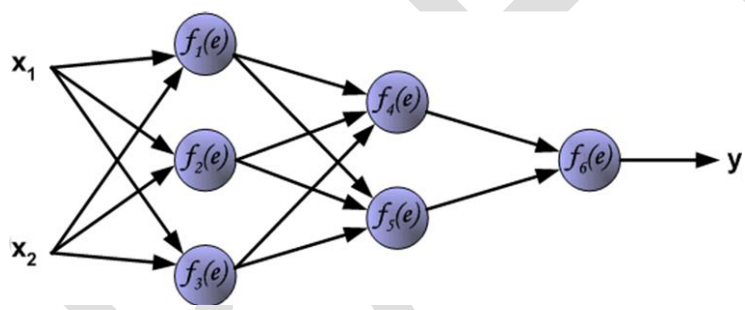

Figure 3. Neural Network Topology

Note that the number of neurons in the input layer corresponds to the number of inputs to the system. While linear systems require the inputs to be linearly independent, this restriction is not placed on neural networks, which makes them attractive for systems with data sources that may be related. Similarly the number of outputs is not restricted to one.

The topology represented above is a feedforward network, defined as "a network in which signal paths can never return to the same signal node." [15] This is in contrast to a feedback network, defined as "a network in which signal paths can return to the same signal node" [15], which would be represented in the above diagram by right-to-left signal arrows between neurons. This implies a temporal dependence of the network on its previous states; whereas a feedforward network has no such state dependency and is a function only of its current inputs. Feedforward networks are prevalent in the literature discussing the application of ANNs to safety-critical systems, with the above topology usually identified as a multi-layered perceptron.

Many applications of this topology include only one hidden layer, with the number of neurons in the layer driven by the intrinsic characteristics of the phenomenon the ANN is intended to approximate. Initial observations about the phenomenon are provided to the ANN through

training using training data, sometimes referred to as the training set.

- Training Data: Input and desired output data pairs used to train the system. This data allows the neural network to learn, by adjusting certain parameters, the appropriate response to predefined stimuli. [11, 12]

Note that data pairs does not imply single input/single output training data; rather the pair would consist of an input vector and an associated output vector. A useful delineation between training and learning is:

- Learning: The modification of a neural network's behavior in response to its environment [12].

- Training: Learning using pre-operation knowledge that defines the appropriate response to predefined stimuli.

The primary distinction for ANNs between training and learning in general is the context: training encompasses learning that occurs outside of an operational context in a controlled environment (such as during system development). Learning in general may also occur during operation. This distinction will serve as a primary discriminator in the following discussion of the use of ANNs in a safety-critical system: concerns regarding ANNs that are trained during development and then fixed (no additional learning occurs during operations) vary significantly from concerns regarding the use of ANNs that continue to learn during operations (dynamic).

In feedforward networks, the common approach to learning is through the use of a back-propagation scheme. In simple terms, once the network has processed the input from the training data, the network looks at the difference between the network's response and the expected response, then the weights associated with the nodes of the network are adjusted by working backwards through the network.

While there are a substantial number of variations, back-propagation typically involves a gradient descent learning rule that influences the adjustment of the network weights in the direction of the steepest error gradient, as illustrated in Figure 4.

DRAFT

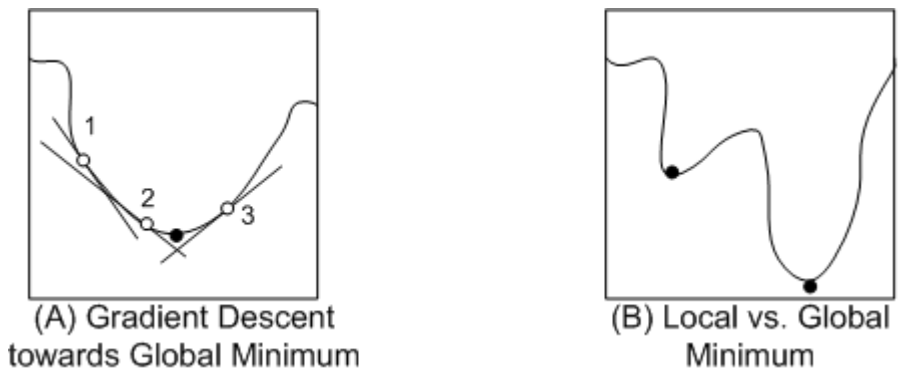

Figure 4. Gradient Descent Learning Rules

In diagram A in Figure 4, starting at point 1, the gradient descent rule implies that the weights are adjusted such that the response of the system is influenced towards point 2. At the subsequent application of the same training data, the gradient at point 2 will influence the response in the same direction, resulting in an overshoot of the global minimum shown in the figure as a solid dot. The gradient descent at point 3 would then influence the solution back towards the global minimum, with the magnitude of the gradient influencing the step size. This repeats until the process converges on the global minimum error within an acceptable tolerance. Diagram B in Figure 4 illustrates a common concern that learning converges on a local minimum, resulting in significantly higher training error that what would be achieved at the global minimum. Sophisticated back-propagation schemes have been developed to improve the speed of convergence on a global minimum error and avoid convergence to a local minimum error.

4.1.2 Example Application of a Neural Network

Worked performed by Goodrich and Barron Associates in applying a neural network to an in-flight fuel measurement system provides a good example of the use of a neural network in an airborne application [16, 17].

The conventional approach to a fuel measurement system involves the development of a multi-dimensional lookup table that maps fuel sensor signals to fuel levels. Due to the shape of the fuel tanks and nonlinearities of the sensors, it is not analytically feasible to develop a mathematical expression for this nonlinear estimation problem. Instead, nonlinear data is generated by a complex high-fidelity model of the fuel system and tabularized into a form that can be indexed by a table lookup software function.

The table resulting from this approach requires a very large memory footprint. Furthermore, this approach requires each sensor input to be treated as an independent degree of freedom (a separate table dimension) when in actuality significant coupling may exist between sensors that can be exploited to generate more accurate measurements. These two reasons motivated research into using a neural network to represent the mapping between sensor data and fuel measurements.

A comparison between the conventional approach and the neural network approach is shown in Figure 5.

DRAFT

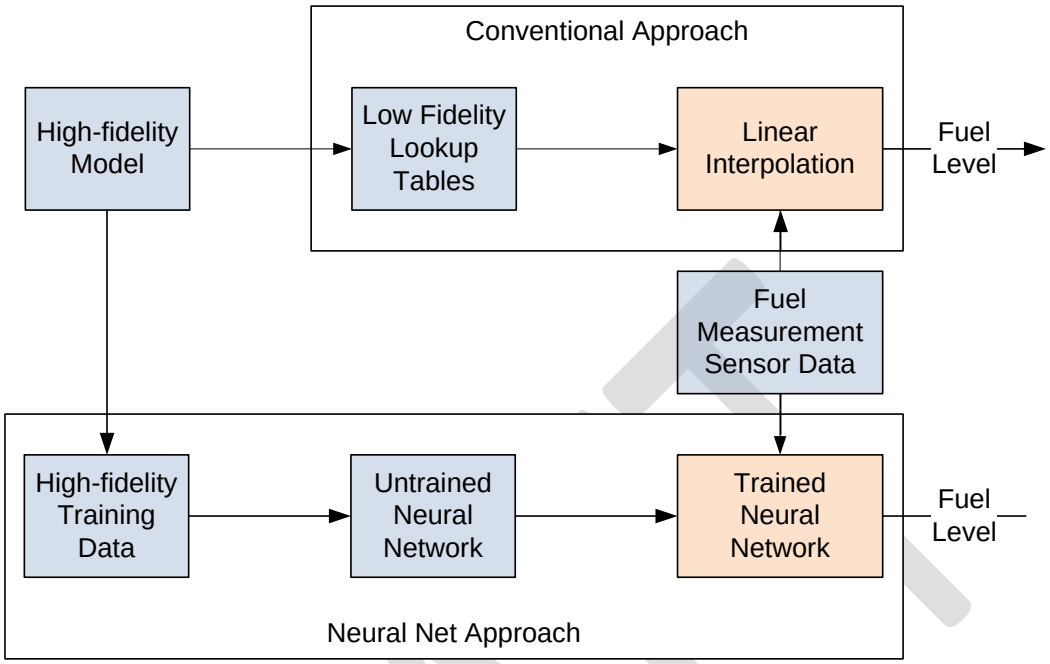

Figure 5. Comparison of Fuel Measurement System Approaches

The neural network approach used the data from the same high-fidelity model used to develop the table data to generate training data for a feed-forward neural network. Goodrich and Barron Associates developed techniques to monitor the back-propagation training of the neural network against verification data to assure that the training converged on a solution that matched the required behavior to an acceptable accuracy. Additionally, analytical methods were developed to guarantee that the nonlinearities in the resulting network were bounded to an acceptable level. Once trained, the neural network was fixed, and the resulting static network algorithm represented the algorithm that would be implemented in software.

The following describes the four factors discussed in section 3 for this example.

1. Feedback Process: The active feedback path is used during network training in a controlled environment using data from the high-fidelity model. The back-propagation process refines network weights based on the difference between the output defined in the training data and the actual network output, known as network error. Network error continues to be fed back to the back-propagation process through subsequent iterations until the network error is reduced to an acceptable threshold across all inputs, at which point the feedback is disabled and the network configuration fixed.

2. Life-Cycle Context: The neural network training occurs during development only and is fixed for operation.

3. Learning Method: Supervised training is used to train the network with data generated from the high-fidelity model.

4. Role of Adaptation: The statistical analysis method is regression analysis between the input (sensors) and output (fuel measurement). This is a nonlinear, continuous model of the process.

The neural network approach compares to the conventional table-lookup approach for fuel measurement as follows:

- Training through the back-propagation resulted in a compact algorithmic representation of the fuel management system when no analytical solution could be derived through conventional means. This resulted in a representation that required far less computational resources than the conventional approach.

- Specific analysis was necessary to verify that all nonlinearities within the network were suitably bounded and would not result in unintended network behavior.

- Because the network was fixed for operations, the concerns regarding its use in operations are much the same as would be expressed for a conventional approach (assuming that the nonlinear nature of the algorithm has been thoroughly reviewed).

- The use of supervised training provided the means to quantify the network error and establish that the network error met an acceptable tolerance across the input space. Verification data, independent of the training data, was used to verify that network was not over-trained. Note that the use of supervised training requires verification that all intended functionality is completely expressed within the training data.

4.1.3 Observations on Neural Networks

A strong appeal of neural networks is that they can be trained to represent nonlinear relationships. Nonlinearity is also where many of the concerns lie with the use of neural networks. Discontinuities, regions of the data that are highly nonlinear, and the presence of a large number of inflections may indicate that (a) training may be difficult, and (b) the ANN topology may be inordinately complex. Accordingly, verification that the nonlinearities in the network are well behaved and bounded is a critical verification task. Additionally, if a neural network is adaptive in operation, specific consideration must be given to the integrity of the convergence of the training and the stability of the network in general. Finally, because the training is encoded in the network through the refinement of the neuron weights, the relationship between the network structure and the process that the network represents is often difficult or impossible to ascertain analytically. In other words, traceability between the network and the requirements addressed by the network often cannot be evaluated by inspection: it is only through demonstrating that the network behavior is representative of the specified behavior that the traceability can be established.

4.2 GENETIC ALGORITHMS

A genetic algorithm is a stochastic search method that mimics the process of natural selection and the resulting changes in genetic composition. Whereas most stochastic search methods operate on a single solution to the problem at hand, genetic algorithms operate on a population of solutions. Genetic algorithms are often used to generate useful solutions to optimization and search problems. To use a genetic algorithm, one must encode solutions to a problem in a structure that can be stored in the computer. This object is called a genome (or chromosome). The genetic algorithm creates a population of genomes and then applies evolutionary operations to the individuals in the population to generate new individuals. It uses various selection criteria so that it picks the best individuals for further evolution.

Genetic programming is a specialization of genetic algorithms where each individual genome is a computer program. It is used to optimize a population of computer programs according to a fitness landscape determined by a program's ability to perform a given computational task. Genetic programming evolves a population of computer programs. That is, generation by generation, genetic programming stochastically transforms populations of programs into new, hopefully better, populations of programs.

A generic algorithm for genetic algorithms is as follows:

1) Randomly create an initial population of genomes representing a valid solution
2) REPEAT
    a. Analyze the population to ascertain their fitness
    b. Select one or two from the population with a probability based on fitness to participate in genetic operations
    c. Create new individuals by applying genetic operations with specified probabilities
3) UNTIL an acceptable solution is found or some other stopping condition is met
4) RETURN the selected member of the population

Genetic programming, like nature, is a random process, and it can never guarantee results. As such, the likelihood that this particular approach for an adaptive system would be considered for use in a civil airborne application is extremely small.

4.3 REFLECTION/AUTONOMIC COMPUTING

In computer science, reflection is the process by which a computer program can observe and modify its own structure and behavior at runtime. Reflective/Autonomic Computing is computation carried out on data representing the state of an executing system’s hardware, software components and their interactions. This is typically accomplished by dynamically assigning program code at runtime. Self-adaptive reflective middleware employs reflective computation in order to adapt its behavior in response to evolving conditions such as system performance. For instance, a single node might alter the thread schedule for high-demand services, a distributed computing system might replicate services that are under high-demand to multiple nodes, or a particular protocol may be selected based on monitoring network traffic.

DRAFT

4.4 SUPPORTING TECHNOLOGIES

Two additional technologies, fuzzy logic and expert systems, are mentioned here to supplement the algorithms described above. These technologies are not adaptive, that is, they do not have a feedback loop.

Fuzzy Logic: Fuzzy logic is a form of many-valued logic derived from fuzzy set theory to deal with reasoning that is fluid or approximate rather than fixed and exact. Fuzzy logic is a superset of conventional (Boolean) logic that has been extended to handle the concept of partial truth, where the truth value may range between completely true and completely false. Fuzzy logic was developed for complex control systems in which mathematical models were difficult or impossible to create such as in highly non-linear systems. It has been evaluated for use in a variety of control applications including flight controls [3, 18]. Fuzzy logic can be combined with a neural network to produce an adaptive fuzzy control system.

Expert Systems: An expert system is a computer-based system design to emulate the problem solving behavior of a human expert [3]. An expert system usually is comprised of a rule base and an inference engine that cooperate to simulate the reasoning process that a human expert pursues in analyzing a problem and arriving at a conclusion. An expert system can be made adaptive if an adaptive technology is used to refine the inference weights based on feedback from the user.

5. ADAPTIVE CONTROLS AND THEIR USE IN ADAPTIVE SYSTEMS

This section provides an overview of adaptive controls, separate from the other adaptive technologies, because of the distinct use of feedback for control systems. A summary of concepts and terms relevant to control systems is provided first, followed by a discussion of conventional and adaptive controls. Then, representative adaptive control system architectures are discussed to illustrate the variety of adaptive control approaches.

The field of adaptive controls is extremely wide, and this section provides only a short overview of a few representative approaches. Note that other adaptive mechanisms, such as neural networks, may be used to provide the adaptivity within the control system. Accordingly, this section focuses on the general concept of adaptive controls, rather than the specific form of the adaptive mechanisms used within the control system.

5.1 CONTROL SYSTEM OVERVIEW

A control system is the combination of, at a minimum, a controller connected to the physical system being controlled. A closed loop system, shown in Figure 6, is formed by connections between an externally-provided command ("Cmd" in Figure 6), the controller and the actuator(s) of the physical system, and the feedback from the sensor(s) of the physical system to the controller. The controller feedback provides measured aspects of the physical response, such as rates or positions, depending on the type of sensors used in the feedback path.

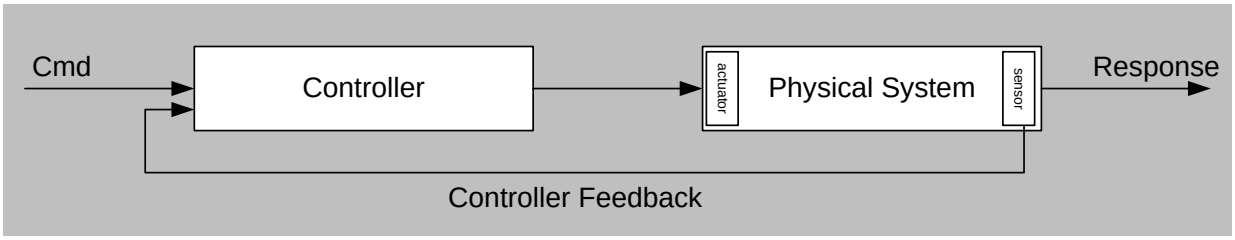

Figure 6. Closed Loop Control System

5.1.1 Controllability/Observability

Controllability and observability are two important properties associated with the physical system and the interfaces between the physical system and the controller. While both properties can be expressed mathematically, the fundamental concepts behind them, described in [19] are as follows:

- Controllability is related to the possibility of placing the physical system into a particular state by using an appropriate control signal. If a state is not controllable, then no controller signal will ever be able to control the state. If a state is not controllable but its dynamics are stable, then the state is termed stabilizable.

- Observability is related to the possibility of observing, through output measurements, the state of a system. If a state is not observable, the controller will never be able to determine the behavior of an unobservable state and hence cannot use that state to stabilize the system.

Note that both definitions use the concept of state. State is often expressed mathematically, but the concept of state can be thought of informally as a snapshot of the dynamic (time-varying) properties of the system under discussion.

Controllability and observability are important not only from a conventional design perspective, but also in the way in which they limit what can be accomplished through the use of adaptive control techniques if the characteristics of the physical system change during operation. For example, a state may become uncontrollable due to an actuator failure or a state may become unobservable due to a sensor failure. Either case may significantly restrict the feasibility or effectiveness of an adaptive control technique.

5.1.2 Stability

The definitions for controllability and observability discuss their impact on the ability of a controller to stabilize the system. Stability also has various mathematical definitions, which for linear systems (systems for which the output is proportional to the input) are typically expressed in terms of gain and phase margins in the context of frequency-domain analysis. Frequency-

DRAFT

domain analysis evaluates the control system behavior based on the frequency content of an input signal. Conversely, time-domain analysis focuses on the actual dynamic response of the system.

5.1.3 Robustness

The robustness of a control system is defined as the quality of maintaining designed controller stability and performance properties in the presence of uncertainties in the system or unmodeled disturbances. Gain and phase margins provide, for linear systems, a standard approach for ensuring robustness. Robustness can be evaluated in frequency domain analysis by confirming that positive stability margins remain for small variations in dynamics or modeling assumptions. This analysis covers both independent and coupled variations. Robustness analysis is then extended into time-domain analysis where additional effects are included, such as system nonlinearities whose stability impacts were approximated through linearization in frequency domain analysis.

5.1.4 Robustness of Non-Adaptive Control Systems

For a non-adaptive control system, controller parameters such as gains and filter coefficients are determined during design and then fixed for operations. As a result, the controller design is based on a priori knowledge and assumptions regarding the dynamics of the physical system operating in its environment. Performance and robustness of the control system may degrade if the actual dynamics of the physical system, or the actual environment, do not match what was assumed while developing the controller.

Non-adaptive control design methods result in a fixed controller for a design point defined by a specific physical system operating in a specific environment. Some methods, such as robust control theory, account for uncertainties and disturbances within the physical system as part of the design methodology. These methods still result in a fixed controller, albeit one with more robust characteristics than what might result from more classical design methods. Additional analysis can assess the robustness of the control system by varying the properties of the physical system or the environment away from this design point.

For many application domains, such as aircraft flight controls, multiple design points can be used to maintain robustness across the range of operations. Often the same controller architecture can be used across the design points, with only variations in controller parameters such as gains and filter coefficients required to tune the controller for each point. For these situations, a gain scheduling approach, shown in Figure 7, can be used to select the correct set of parameters based on look-up tables generated during design, as depicted in the following diagram. Note that in Figure 7 and subsequent figures, the shaded area represents the structure of the conventional control system shown on the previous page. This does not imply that the controller in the shaded area remains unchanged from the one shown in the initial diagram; instead, it is intended to show the evolution of various control system approaches away from the conventional baseline. In this instance, the structure of the controller is fixed, but the parameter values used within controller are passed in from the gain schedule tables. Note also that, for the sake of simplicity, the sensor/actuator dynamics are assumed to be part of the physical system.

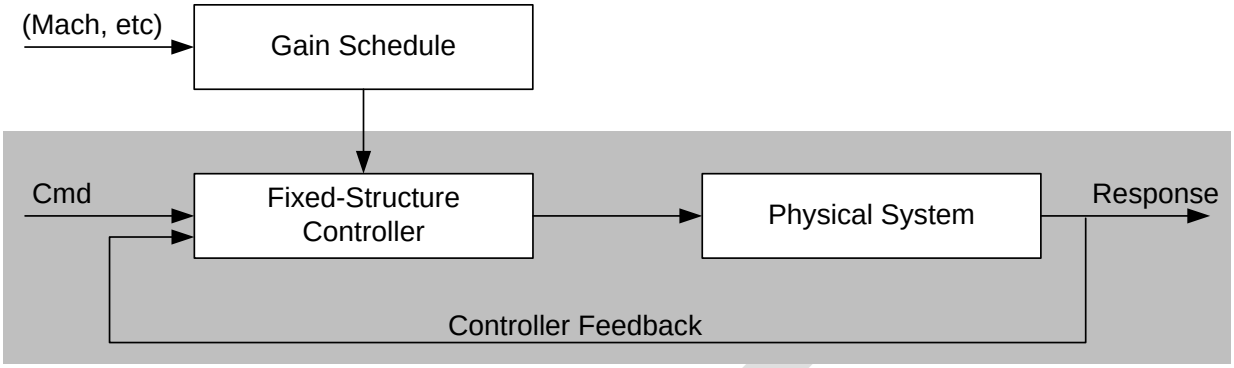

Figure 7. Control System with Gain Scheduling

5.1.5 Robustness of Adaptive Control

This discussion provides a primary delineation between what is considered, for the purposes of this study, to be adaptive and what is considered to be non-adaptive:

- An adaptive approach uses real-time, in situ feedback of the control system performance to refine controller parameters.
- A non-adaptive approach uses a priori analysis to design a controller for a specific operating point.

Note that the non-adaptive approach is not refining parameters based on performance; it is merely indexing parameters that have been configured through a priori design analysis. Based on this discussion, adaptive controls may provide increased robustness by providing the means to refine controller parameters in situ based on the actual response of the physical system, rather than assumed response used for design of a fixed controller. Additionally, adaptive approaches may be able to accommodate unanticipated changes in the dynamics of the controlled physical system or its environment that are outside of the operational space assumed for a non-adaptive approach. Conversely, adaptive controls may provide no benefit if the dynamics of the system are well-known across the operational range to support the controller design, and in some instances may be less robust than a non-adaptive approach, particularly if the time to refine parameters is large.

It should be noted here that adaptive control design methods can also be used to design a fixed (non-adaptive) controller at a design point. The adaptive approach converges on a solution through training conducted as part of the controller design process, after which the structure of the controller and the controller parameters are fixed. Additional analysis is still necessary to evaluate the robustness of the controller away from the design point. Because there are often nonlinearities in the controller developed using an adaptive approach, robustness analysis techniques may be different from the linear systems analysis techniques used for conventional design methods.

DRAFT

5.1.6 Airworthiness Terminology Relevant to Stability/Robustness

The Code of Federal Regulations (CFR) and associated Advisory Circulars relevant to control system characteristics do not discuss control system characteristics in terms of frequency domain gain/phase margins. Instead the emphasis is on characteristics that can be demonstrated in time-domain analysis and flight tests. Examples of CFR pertaining to stability and robustness include:

- §25.171 (Stability – General): longitudinally, directionally, and laterally stable across operational range defined in §25.173 through §25.177, and suitable stability and control feel for any normal operating condition necessary for safe operations.

- §25.173 (Stability – Static longitudinal): returns to trim speed within a defined percentage after perturbation away from trim for operational range defined in §25.175.

- §25.181 (Dynamic Stability): short period oscillations, other than lateral-directional, must be heavily damped with primary controls across operational speed range; and lateral-directional oscillations must be positively damped with controls free and controllable with primary controls without requiring exceptional pilot skill.

(Note that the above descriptions are summaries of the CFR text). These illustrate how the regulations address stability in terms of settling to trim conditions or damping of oscillations, and robustness in terms of characteristics maintained over an operational range. Damping can be expressed mathematically, but generally pertains to how quickly a dynamic system settles to a commanded steady state condition or returns to a prior steady state condition after a perturbation such as a wind gust. Positive damping indicates a reduction in magnitude to a negligible level within a small number of oscillations, and heavily-damped typically refers to the same reduction within one period of the oscillation.

5.2 ADAPTIVE CONTROL SYSTEM ARCHITECTURES

While non-adaptive control systems use fixed controller parameters, or parameters that are selected from a fixed lookup table, the primary characteristic of adaptive control systems is that they provide a feedback mechanism for modifying their parameters to improve in-situ control system performance, ideally without sacrificing robustness. In other words, the controller adapts based on the actual response of the physical system operating in its actual environment to improve its performance. To accomplish this, an adaptive control system includes a feedback mechanism that is often independent from the controller feedback path. This feedback mechanism collects data regarding the performance of the control system, analyzes the data, determines modifications to the control system parameters, and updates the parameters for use in the next execution cycle. The following subsections describe some of the most common adaptive control system approaches.

5.2.1 Model Reference Adaptive Control (MRAC)

In Model Reference Adaptive Control (MRAC), shown in Figure 8, a reference model describes the desired response of the physical system operating in its environment, given the input

command to the controller. If the controller is perfectly tuned, the actual response of the system would essentially be identical to the response of the reference model. However, because of assumptions or approximations made during design, there is likely to be some variation in the actual response of the system away from the ideal response described by the reference model. Accordingly, the MRAC approach refines the controller parameters with the goal of driving the response of the system to match that of the reference model. In the figure, the adaptive feedback loop is shown in the diagram (and subsequent diagrams) is shown in red.

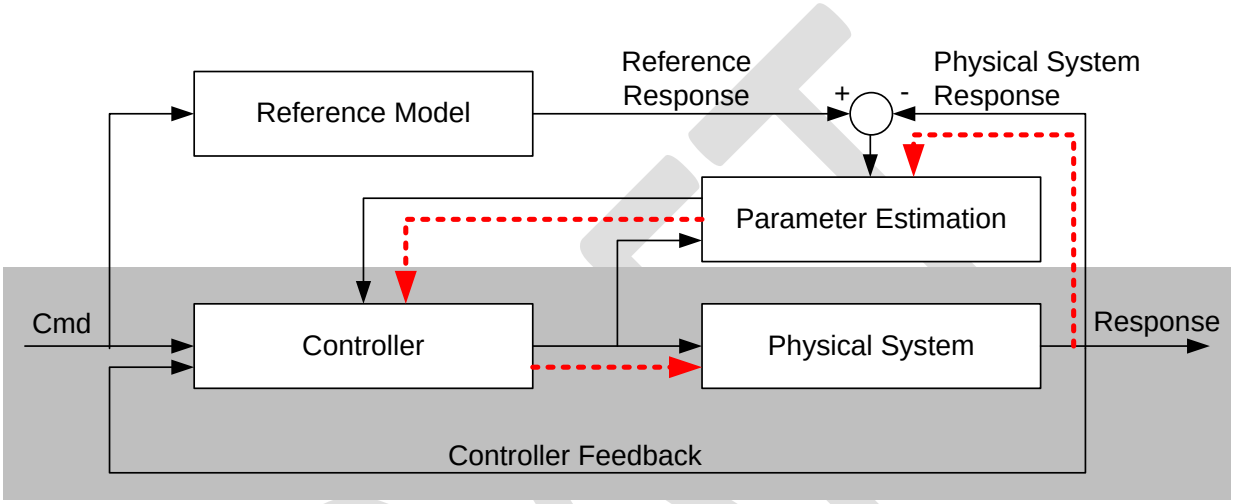

Figure 8. Model Reference Adaptive Control

MRAC is an example of direct adaptive control, where the controller parameters are directly calculated by the adaptive process.

5.2.2 Model Identification Adaptive Control (MIAC)

In Model Identification Adaptive Control (MIAC), shown in Figure 9, a predefined reference model is not provided to the parameter estimation. Instead, a dynamic reference model is built using system identification techniques. As the system is characterized, the parameter estimation uses this representation of the system to refine the controller parameters. MIAC is advantageous in applications where a predefined reference model is not feasible, but is limited to applications where the important characteristics of the system can be identified based on the observed response of the system.

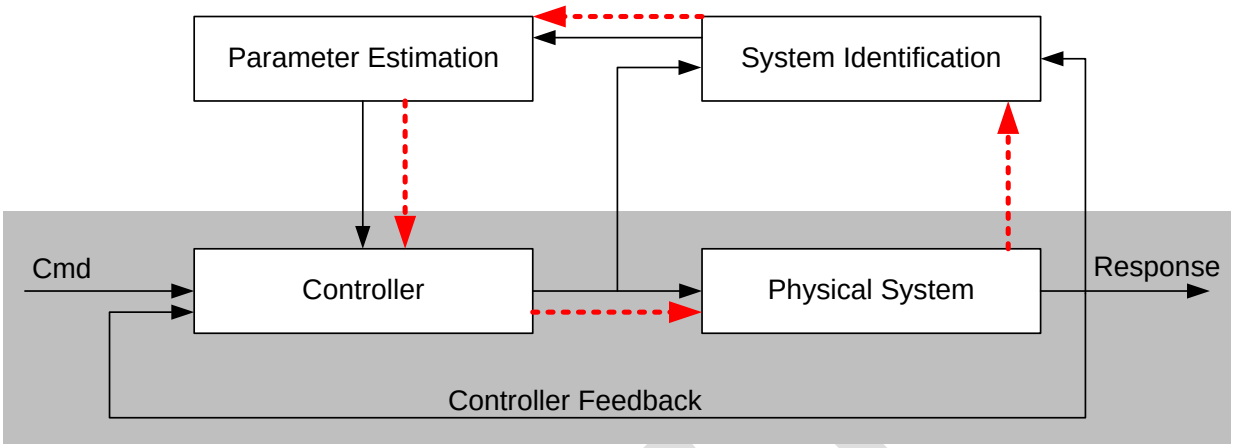

Figure 9. Model Identification Adaptive Control

MIAC is an example of indirect adaptive control, where the adaptive mechanism provides estimated system parameters to a separate process that calculates the controller parameters. For instance, the adaptive system identification process may refine estimates of aerodynamic coefficients, which are subsequently used in the parameter estimation process to calculate controller gains and filter coefficients.

5.2.3 Model-Free Adaptive Control

In Model-Free Adaptive Control, shown in Figure 10, the adaptive controller observes the response of the system and performs direct refinement of the control parameters. The model-free approach essentially collapses all of the functions associated with system identification, parameter estimation, and control into a single process. In many applications, little a priori knowledge of the system is needed to implement a model-free approach, other than a reasonable estimate of the system time lag (i.e., the interval of time between an input to the system and the response to that input). Disadvantages of the model-free approach are controller complexity and a lack of visibility into the process to support placement of process monitors.

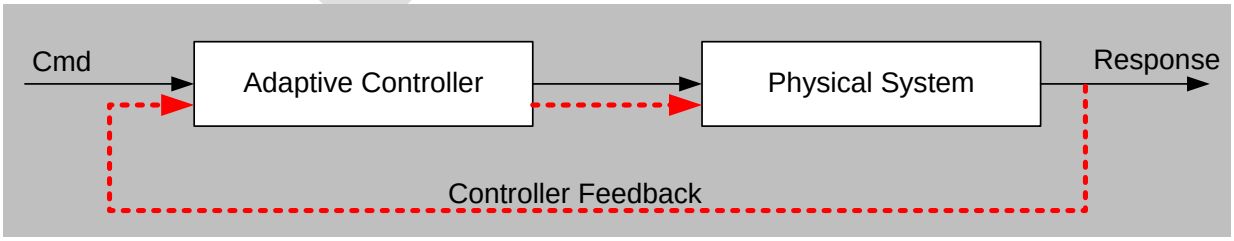

Figure 10. Model-Free Adaptive Control

5.2.4 Adaptive Augmentation

Adaptive augmentation, shown in Figure 11, provides a way to improve the robustness of a system that uses a fixed (non-adaptive) controller. A separate controller (labeled Adaptive Augmentation in Figure 11) provides a supplemental control signal designed to cause the physical system to track the response of the reference model. This augmentation essentially compensates for assumptions and approximations in the reference model used to design the non-adaptive controller.

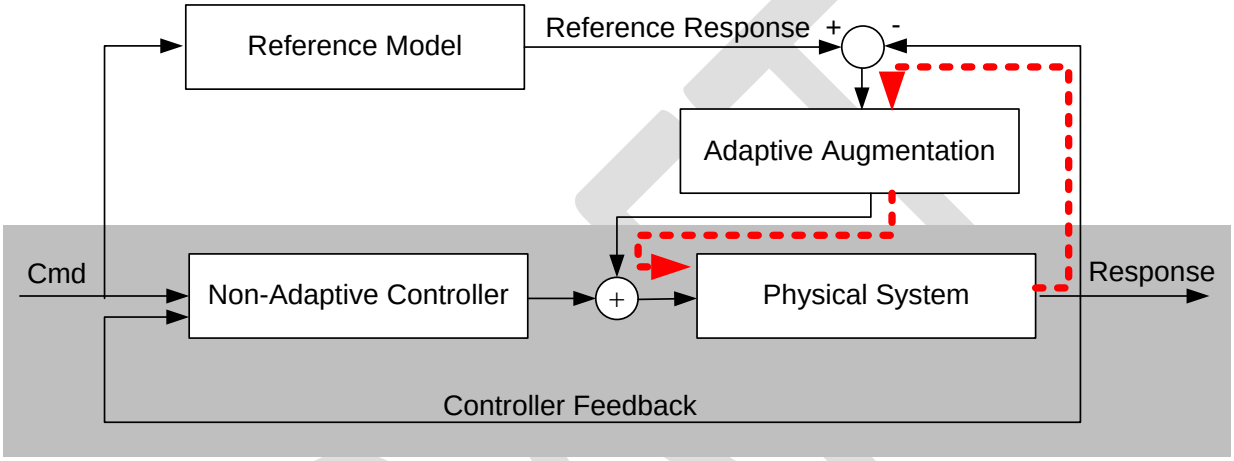

Figure 11. Adaptive Augmentation

5.2.5 L1 Adaptive Control

The intent of all adaptive control approaches is to improve robustness by adaptively refining controller parameters through a feedback process. There are two primary concerns for all of these approaches:

- The adaptation is always stable and transient responses are not unsafe.
- The time to adaptation is sufficiently short compared to other important system time-spans. For instance in adaptive flight controls, adaptation should be sufficiently fast to avoid adverse interactions with pilot inputs.

For many adaptive control approaches, the robustness of the solution is coupled with the time to adaptation. A design that has a high adaptation gain (i.e., is tuned to adapt quickly) can have unacceptable transient response during adaptation and poor stability. Conversely increasing the stability of the adaptive approach is often achieved by a lower adaptation gain, which requires a longer time period to converge on the refined control parameters.

The L1 adaptive control approach [20] provides a solution to these two concerns by providing guaranteed stability and fast adaptation. This is accomplished by decoupling robustness and the time to adaptation in the L1 design methodology. Through this decoupling, robustness

guarantees can be defined independently from the influence of adaptation time, and the adaption time can be made arbitrarily small (depending on hardware limits) without adversely impacting robustness. Because L1 is a controller design approach, it does not have a specific system topology.

6. INITIAL IDENTIFICATION OF SAFETY ISSUES

Understanding the safety issues associated with adaptive system is essential to developing an appropriate approach for software aspects of them. There are safety issues associated with the various factors described in Section 2 that influence an adaptive system, including the life cycle context where adaptation is actively used, the learning method employed, and the role that the adaptation plays in the system function. This section discusses safety issues identified during Phase 1, with respect to each of these factors.

6.1 IMPACT OF LIFE CYCLE CONTEXT ON SAFETY

A primary distinction with respect to safety is between adaptive systems that are adaptive only for development, and those that are intended to remain adaptive in operation. Because the design associated with adaptive systems used for development only are fixed, the safety concerns for these are similar to what would be appropriate for any other fixed design. However, there are some characteristics of the adaptive algorithm that should be addressed from a safety perspective.

Using the fuel measurement system example, the neural network was trained using data provided by a high-fidelity model relating sensor readings to fuel level. The trained network shows good agreement with the training data and verification data. Although the network was then fixed, it still contained nonlinearities that needed to be evaluated analytically to show that they would not present a threat to the safe operation of the aircraft by providing spurious fuel measurement readings into the system. An analytical approach was devised that showed mathematically that nonlinearities between test points were bounded and well behaved, demonstrating that network nonlinearities would not result in spurious fuel measurement.

Based on this example, the following safety considerations were identified for the use of an adaptive approach where the system is fixed in operation.

- The integrity of the training data should be substantiated. In particular, the training data should provide a complete and correct representation of the intended behavior of the adaptive system.

- Verification data that is independent of the training data should be provided to demonstrate that the network has converged to an acceptable solution.

- Any nonlinear characteristics in the resulting system should be shown not to adversely impact system safety. This argument should address the entire operational input space of the adaptive system, not just specific points covered by training or verification data.

- The means for continuing to update the solution should be shown to be disabled so that the system cannot inadvertently become adaptive in operations.

- Other safety concerns that may be associated with the use of similar conventional systems for the specific application area should be addressed.

For applications that adapt in operation, the safety considerations are predicated on the type of application and the safety concerns associated with the specific adaptive approach. These safety considerations are addressed in the following section.

6.2 IMPACT OF THE ROLE OF ADAPTATION ON SAFETY

Adaptation could be used in various types of applications that would be relevant to NextGen or to airborne systems. Additional types may be identified as this analysis progresses, or the definition of additional sub-types may prove beneficial in the future. For each type of application, the role that the adaptation would play is addressed, and safety considerations are listed.

6.2.1 Monitoring

Adaptive approaches that use statistical analysis for classification and clustering can be used for monitoring applications, such as vehicle or system health monitoring. If used in a batch environment, a monitoring technique can review a pre-recorded data set to look for anomalies and trends that may otherwise not be evident from inspection. If used in a real-time context, these techniques can assess current data in light of the data history to produce insight for situational awareness or vehicle health.

Safety considerations: Use of adaptive approaches in system monitoring is typically used in an off-line setting. Any insight gained from these analyses can be subject to verification before an action, typically manual, is taken. The use of these techniques in a real-time setting is a current research topic. Certainly the risk of either false-positive or false-negative indications resulting from these techniques would need to be carefully evaluated before any use could be proposed for a safety-critical environment, particularly if the indication triggered an automated response.

6.2.2 Parameter Estimation for Performance/Efficiency

An adaptive approach can be used to refine estimates of a parameter or set of parameters that are related to performance. For conventional systems, performance models are often deliberately conservative, and on-line estimation of related parameters may provide benefit in terms of efficiency or performance. Regression analysis is often used for parameter estimation.

Safety Considerations: Parameter estimation may represent minimal safety concerns if certain considerations can be addressed. The primary consideration is the establishment of a safe range from which the adaptive approach can select a parameter value. Excursions outside of this bound would be an indication that the adaptive approach is non-convergent, and a monitor would provide the means for reverting to a conventional approach, such as a fixed value that represents

the mid-point of the allowable parameter range. Analysis would be necessary to show that the discontinuous jump from the previous out-of-range value to the midpoint value does not introduce an unsafe transient.

6.2.3 Parameter Estimation for Robustness

The previous discussion addresses parameter estimation to improve performance and/or efficiency. Parameter estimation can also be used in a controls application to improve robustness of the controller. For cases such as indirect adaptive control, control parameters may be calculated from other system parameters such as airspeed. In this situation, the control law itself is not adaptive, but parameters provided to it, from which it calculates its control parameters, are determined using an adaptive approach. The use of indirect adaptive control is likely to result in a more robust control law for the current operational conditions, because adaptive estimation is likely to provide more accurate estimations of parameter values than what could be provided through conventional means (such as a priori analysis or a table lookup value).

Safety Considerations: The safety implications for parameter estimations that are used in indirect adaptive control applications may be greater than those associated with estimation used for performance/efficiency improvements. Errors in estimation, even if the estimates are within the tolerance boundaries, now imply a reduction in control system robustness, rather than merely a loss of performance or efficiency. Additionally, a discontinuous jump when the fall-back is invoked may result in an unsafe control system transient.

6.2.4 Control Reconfiguration

Parameter estimation for direct adaptive control generates refined values for the control system parameters directly, based on the response of the closed-loop system. Direct adaptive controls may provide the greatest benefit in terms of robustness, and may provide the only means to retain stability in situations where changes in the system or the operational environment deviate substantially from those assumed when the control law was developed.

Safety Considerations: It may be infeasible to establish parameter estimate bounds as well as a baseline for reversion if the direct adaptive approach fails. Accordingly, there may not be means for providing a back-up control system that can be invoked if the adaptive approach becomes unstable. The viability of an adaptive control system may then rest completely on the strength of the analytical evidence that can be established prior to deployment. Certain adaptive control techniques, such as L-1 adaptive control, provide guarantees regarding convergence and other characteristics related to the robustness. Consequently, safety considerations associated with adaptive control reconfiguration is largely tied to the type of control architecture used and the specific adaptive technology used within that architecture.

6.2.5 Planning

Statistical analysis for optimization combined with reinforcement learning may be applicable to applications involving planning or trajectory analysis. Similar to the discussion for monitoring, it

DRAFT

is envisioned that optimization using reinforcement learning could be used in a non-real time setting to generate plans or optimized trajectories that could then be verified before acted upon. It is possible, however, that these techniques could also be used in a real-time environment to perform in-situ trajectory revisions.

Safety considerations: Planning that uses an adaptive approach is typically performed in an off-line setting. Results from these adaptive techniques can be subject to verification before an action (typically manual) is taken. The use of trajectory optimization or other planning in a real-time context would necessitate the use of monitors to assure that no safety or operational constraints are violated by the results, and that a suitable fall-back is provided in case the adaptive approach is non-convergent.

6.3 SAFETY CONCERNS FOR ADAPTIVE ALGORITHMS

Safety concerns can differ depending on the type of adaptive algorithm or technique used. This analysis is ongoing, and when complete, can be overlaid on the safety considerations associated with specific applications to develop a composite picture that includes safety considerations for a particular adaptive technology used for a particular application.

Neural Networks: Safety considerations for neural networks are split between static (non adaptive in operation) and dynamic neural networks, including dynamically growing networks, where additional neurons may be inserted into the network during operations.

- In static neural networks, unsafe non-linear network response can occur during operation in a region of the input space that was not covered by verification.

- Dynamic neural networks have safety concerns including:

    - Feedback data that could contain a substantial amount of noise that adversely impacts convergence of the adaptation.

    - Neural network schemes that dynamically insert additional neurons into the network. A failure condition may result if this growth is unbounded to the point that it impacts convergence and/or computational resources because there are too many nodes to process in the time required.

    - Adaptation that fails to converge quickly enough to support continued safe operations.

Genetic Algorithms: As discussed section 4.2, genetic algorithms involve stochastic generation of populations of candidate solutions that compete in an evolutionary process. The benefit of the technique is to propose novel candidate solutions, and so would likely be applicable primarily to a development environment. Candidate solutions proposed through the use of genetic algorithms would be subject to verification to assure that they do not present safety concerns. As it is not certain that solutions provided by genetic algorithms are linear, their verification may be similar

to what would be performed on static neural networks or other nonlinear algorithms.

Reflection/Autonomic Computing: This technology can adjust the structure of the computer program at runtime, and therefore the safety considerations associated with the technology can extend to reallocation of computational resources. Relevant safety considerations include:

- Safety impacts due to the reallocation of resources
- Changes in the operating system schedule that may affect hard real-time deadlines
- Resources allocated to increase performance that could adversely affect components

Adaptive Controls: Section 5 discussed a number of representative adaptive control architectures. In some instances a specific adaptive technology, such as a neural network, provides the adaptivity (such as estimating an indirect parameter for model reference adaptive control). In other instances, the technology itself defines the architecture, as is the case with L-1 adaptive control.

Safety considerations for adaptive control depend not only on the type of control architecture used, but also on the characteristics of the system under control. Generally, systems with slowly varying system characteristics may provide less concern for the application of adaptive controls than a system with rapidly varying or discontinuous dynamics. Additionally, issues presented in Section 5 regarding controllability and observability are issues for any adaptive control approach.

In summary, safety considerations for adaptive controllers result from the combined contributions of the specific adaptive control technology, the architecture in which it is used, and the characteristics of the system to which the adaptive control scheme is applied.

7. INITIAL PHASE 2 ACTIVITIES

While the Phase 1 effort was aimed at identifying foundational aspects essential to assessing the assurance needs for adaptive systems, Phase 2 worked to condense that information to understand what specific requirements, especially at the software level, are needed to ensure safe use of adaptive systems. Because the work under Phase 2 was not done by those who worked on Phase 1, initial Phase 2 activities started with a review of the Phase 1 results and final report [21] and independent examination of the characteristics and architecture of several basic types of adaptive systems.

Figure 12 provides a useful taxonomy that captures the essential attributes of adaptive systems.

An adaptive system typically incorporates a reference model of the desired response and a learning mechanism that adjusts the 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, such as

neural network, reflection/autonomic (dynamic code assignment or self-modifying code) and genetic update (random mutation and fitness selection) as described in section 4.

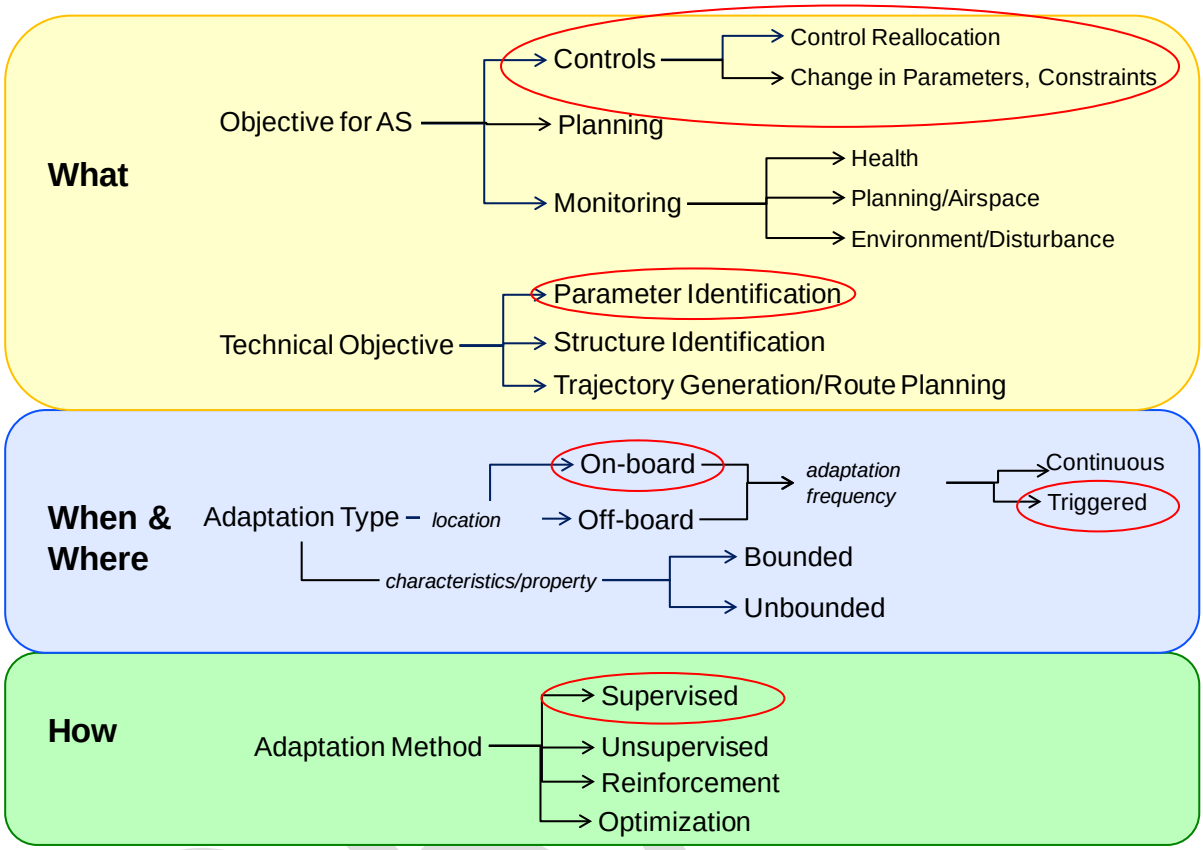

Figure 12. Adaptive System Taxonomy

As Figure 12 shows, there is tremendous diversity among different types of adaptive systems, making generic assessment of software assurance requirements infeasible. Consequently, much of the Phase 2 efforts targeted a particular adaptive system type. To that end, a specific set of features and constraints were defined so that more specific assessment of the impact of that adaptive system could be made. As per the taxonomy above, a controls application (e.g., a flight controller that continuously monitors the environment (parameter identification) was selected, on-board the aircraft (i.e., parameter updates are calculated locally not uplinked from a ground facility), that 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 [22]

- 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 [23]

- Rockwell Collins (formerly Athena) has demonstrated the Automatic Supervisory Adaptive Control (ASAC) on an unmanned small scale F/A-18 [24]

- Honeywell/AFRL Integrated Adaptive Guidance & Control For Future Responsive Access To Space Technology (FAST) [25]

These types of adaptive controllers have not been used in any commercial aircraft.

Secondly, the research team tentatively concluded that to assure the safety of an airborne adaptive system, 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. These aspects are discussed 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. Controllers of the gain scheduled type were not considered since the adaptation is limited to a few pre-determined states and can therefore be verified using the standard methods of DO-178B/C. Similarly, an adaptive system that is pre-trained offline and thereafter remains in a fixed configuration was not considered.

To focus the analysis, a representative adaptive system architecture exemplar was constructed that generalizes the above examples and meets the imposed requirements to use in analysis of verifiability per DO-178B/C. A block diagram is shown in Figure 13.

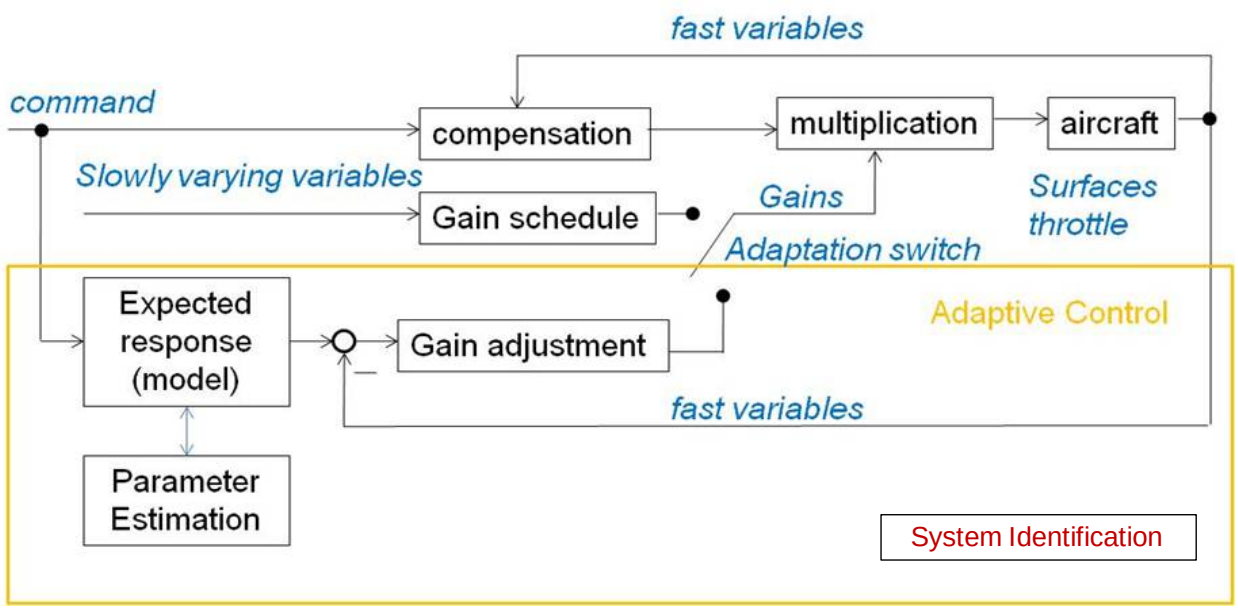

Figure 13. Exemplar Flight Control Architecture

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.
At the outset, two major safety requirements were considered for this controller: 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 seem to be essential features of a certifiable flight control system whether adaptive or not, although they may be challenging requirements, depending on the adaptive algorithm.

To determine the applicability of the DO-178B/C objectives to this adaptive controller, it is helpful to review the current certification framework and the recent changes it has undergone. The next section describes the certification methodology and framework in the current Code of Federal Regulations (CFRs), associated advisory materials and industry standards.

8. CURRENT CERTIFICATION GUIDANCE AND STANDARDS

Discussing the current certification framework is helpful in determining 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 14 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 the interest of focusing on the key steps; these can be found within the documents referenced. The aim here is to give an overview and not a full descriptive narrative. DO-254 [26] is not discussed since this is not within the scope of the original task and is, in any case, limited in application to programmable devices and not to hardware generally [27, 28]. Similarly, SAE ARP-5150 [29] 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) document numbers. The US editions are used here for brevity.

DRAFT

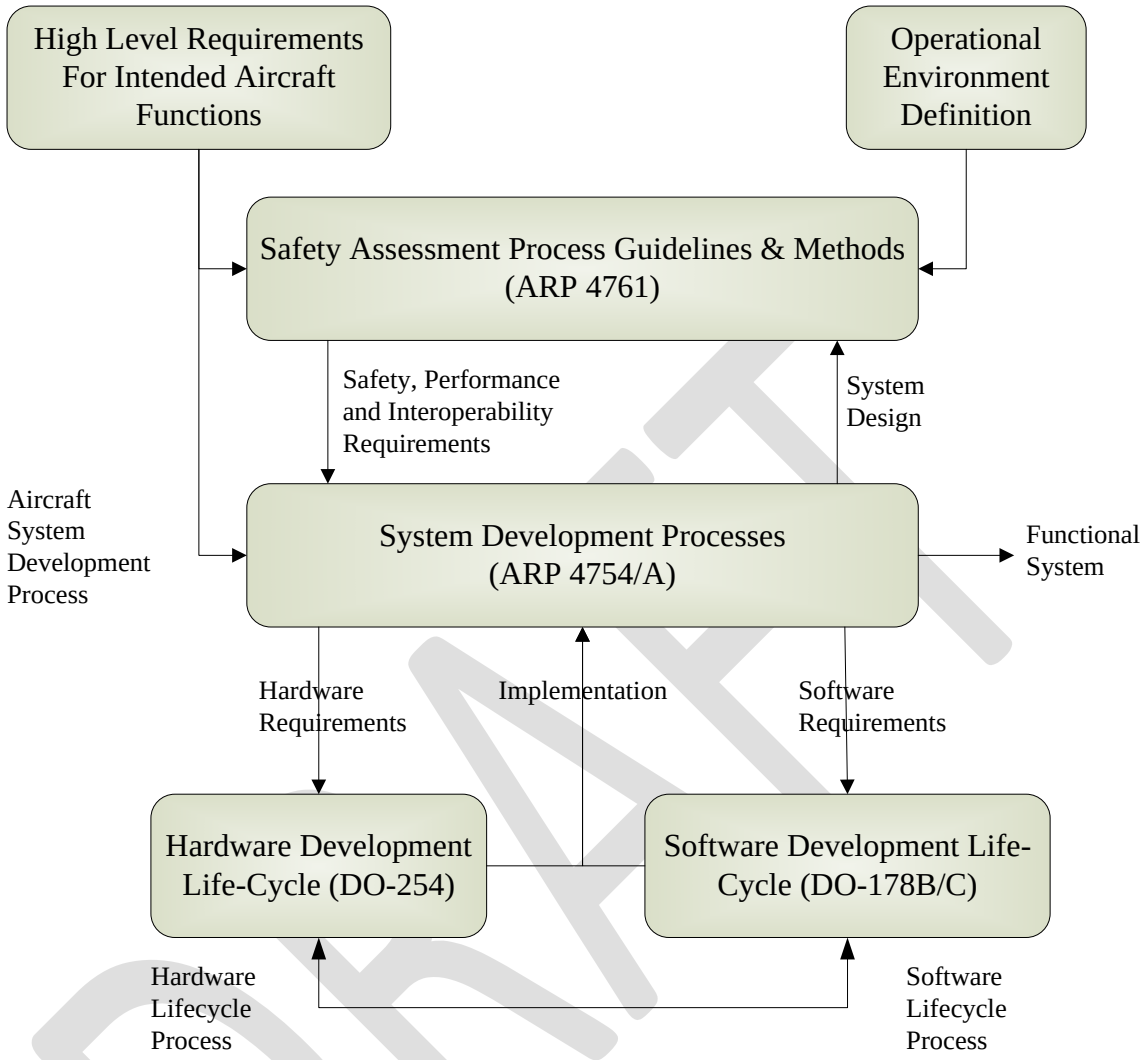

Figure 14. Certification Process Flow and Applicable Standards

8.1 ARP-4754A GUIDELINES FOR DEVELOPMENT OF CIVIL AIRCRAFT AND SYSTEMS

The original version of ARP-4754 [30] 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" (old 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 2 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, the term 'invocation/policy' means 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. AC 20-174 identifies ARP-4754A as an acceptable method for establishing a development assurance process. 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 2. ARP-4754A Invocation

Reference	Description	Applicability	Invocation
ARP-4754A	Guidelines for Development of Civil Aircraft and Systems	Aircraft systems and equipment	AC 20-174 [31], IP, CRI
AC 25.1309-1A [32]	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

DRAFT

8.1.1. Recent Changes to ARP-4754

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

The title of the document changed from "Certification Considerations for Highly-integrated or Complex Aircraft Systems" for ARP-4754, to "Guidelines for Development of Civil Aircraft and Systems." This change has both positive and negative aspects.

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

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 in Figure 15.

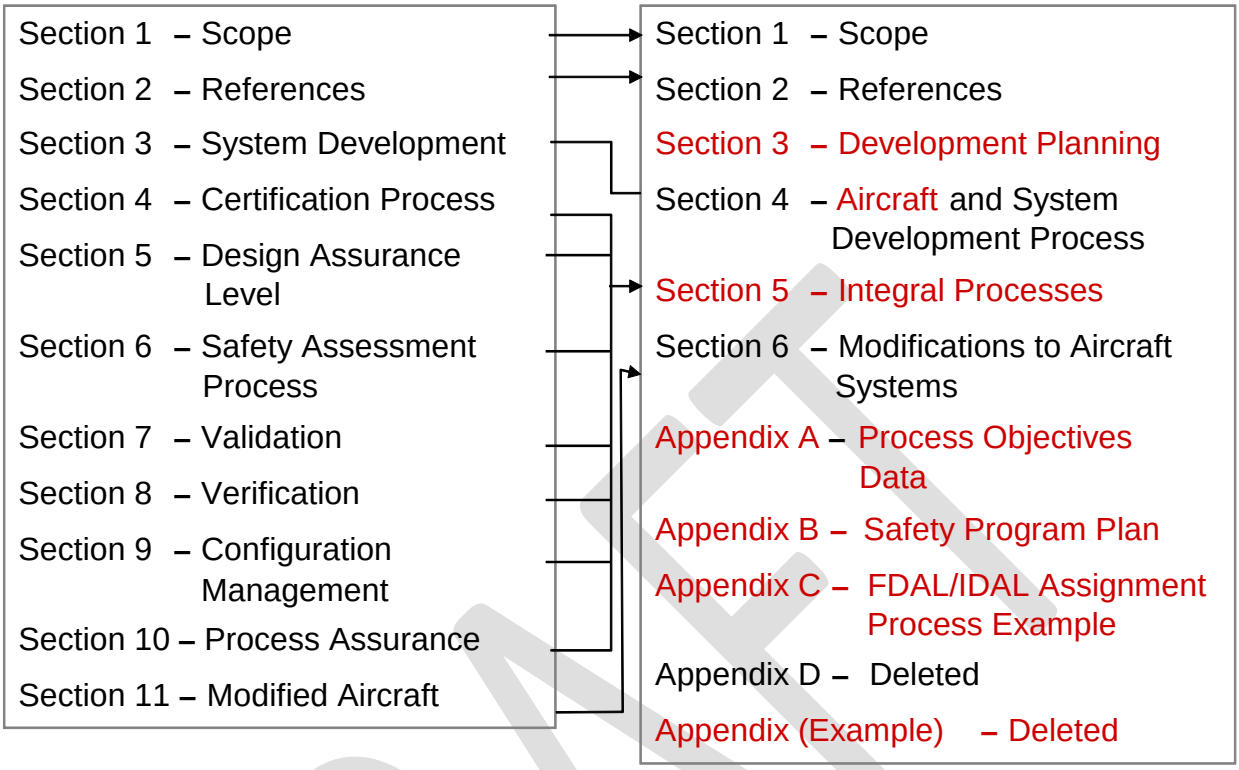

Figure 15. ARP-4754 – ARP-4754A Sections Mapping

The major changes and new content can be summarized as shown in Figure 16:

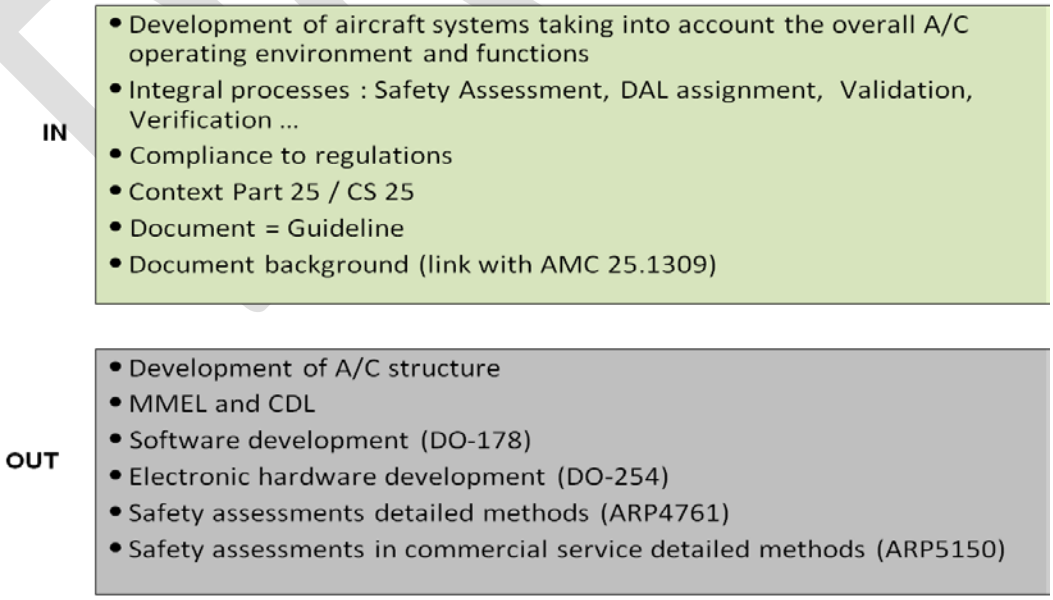

Figure 16. In/Out Mapping of ARP-4754 and ARP-4754A

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

DRAFT

- 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.

DRAFT

- 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.

8.2 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 [33]. 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.

8.3 SOFTWARE DESIGN ASSURANCE

The primary software design assurance guidance document is DO-178B. Table 3 below shows how it is invoked by the current regulatory framework.

Table 3. 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

DRAFT

9. ADAPTIVE SYSTEM CERTIFICATION

This section discusses adaptive system certification in the context of the selected adaptive system controller example and the current certification framework described above.

9.1 CONCERNS ON THE FEASIBILITY OF APPLYING DO-178B TO SOFTWARE DESIGN ASSURANCE OF ADAPTIVE SYSTEMS

All adaptive systems embody a learning subsystem of some type and some embedded model of the desired system performance. The learning subsystem 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 review, test, or conventional analysis 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 [34, 35] and Schumann [36]. One objective in this work therefore is to find ways in which these difficulties can be overcome.

9.2 SUGGESTED APPROACH TO SOFTWARE DESIGN ASSURANCE OF ADAPTIVE SYSTEMS

In other works (e.g., [34]), 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?’ Answers to these questions can best be arrived at by considering the methods by which validated, verifiable high level requirements (HLR) for adaptive systems can be written. 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.

A premise of this work is 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 offer a well defined methodology to partially shift the software design assurance burden from test to analysis. This research 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.

The following principles are asserted:

- 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.

This research suggests 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 software design assurance for adaptive systems. More specifically, this research suggests 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.

9.3 SYSTEM LEVEL APPROACHES TO THE CERTIFICATION OF ADAPTIVE SYSTEMS

In view of the difficulties of showing software design assurance of an adaptive system by following only the processes of DO-178C, this research concludes 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 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 the analysis of problematic DO-178C objectives, this research makes use of the system properties that exist as a consequence of satisfying the safety objectives of the adaptive system.

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.

9.4 DEFINING THE SYSTEM LEVEL CHARACTERISTICS OF AN ADAPTIVE SYSTEM

In order to concretize the analysis, the example architecture diagram of the target (Figure 13) was analyzed 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 the analysis of challenging DO-178B/C objectives, the system properties that exist as a consequence of satisfying these adaptive system safety objectives were used. The first step in the 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 4 below.

DRAFT

Table 4. System Safety Objectives for Adaptive Systems (developed with exemplar adaptive flight control system in mind)

	System Safety Objectives for Adaptive Systems	**Activities and Techniques for Satisfying the Objective**
1	Assure adaptive algorithm stability & convergence (To be satisfied during development)	**<u>Activities</u>** • 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 **<u>Techniques</u>** • 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)

DRAFT

2	Assure adaptive algorithm stability & convergence are satisfied (To be satisfied during real-time operation consistent with Rushby [37, 38])	**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 the adaptive system trigger to eliminate the need for forcing function excitation to enable adaptation learning.

DRAFT

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 the 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. The following methodology was followed.

- List DO-178B/C objectives that are difficult to meet for the example adaptive system chosen
- 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 Appendix C. Due to resource constraints, this table is incomplete. There remain some objectives for which the research team was 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, objectives were merged so that they do not appear in the table in numerical order.

In development of the table, the system-level use of mathematical models and MBD was emphasized to describe (i.e., specify) the complete system behavior in a form suitable for analysis. The system and software level use of FM (and other FM-like techniques) was also emphasized 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.

DRAFT

Lastly, in order to bound the analysis problem further, the use of equivalence classes is suggested as 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.

10. ASPECTS OF TOOL QUALIFICATION FOR ADAPTIVE SYSTEMS

The recommended approach to software verification of adaptive systems utilizes MBD and FM tools. No need was identified to amend the guidance provided in the tool qualification section of DO-178C or of the tool qualification supplement DO-330.

11. 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. One caution is 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.

This summary recommendation is broken down into a number of more detailed recommendations below. These are classified as validation and verification recommendations. Although one particular adaptive system type was the focus of the research, these recommendations are likely to be applicable to a wider variety of systems.

11.1 RECOMMENDATIONS FOR ADAPTIVE SYSTEM SAFETY AND FUNCTIONAL REQUIREMENTS DERIVATION AND VALIDATION

The following steps are recommended 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 and 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.

- Recommend the use of ARP-4754A and DO-178C and supplements.

- Update of ARP-4761 should include a structured, evidence based safety case methodology.

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

11.2 RECOMMENDATIONS FOR ADAPTIVE SYSTEM REQUIREMENTS VERIFICATION

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

- MBD techniques incorporating mathematical models with a well defined syntax and semantics are recommended to 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 the example adaptive system.

- 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

12. FUTURE ADAPTIVE SYSTEMS CERTIFICATION RESEARCH NEEDS

The following list describes some gaps identified 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 Appendix C. 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.

- A study of the mapping of available MBD and FM tools to the adaptive system application domain is suggested 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. Extending that guidance is suggested.

DRAFT

- Because of the specialized nature of FM and MBD techniques, these abilities are not well diffused into the developer community. A more formalized process map should be developed along with supporting user guides.

- The research team recommends that the methodology outlined be demonstrated on an actual adaptive system 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. This could include demonstrating whether formal methods can be used to demonstrate that neural net is stable under all normal and abnormal input conditions.

13. SUMMARY

Adaptive systems are used in myriad domains for a wide range of purposes. This range extends from passive monitoring or advisory systems, to estimating one or more parameters to improve performance or efficiency, and in the extreme, to complete reconfiguration of flight control laws in military systems in response to unanticipated system failures or environmental conditions. For adaptive systems to be used in civil applications for NextGen or in ATC systems in the NAS, those systems will need to comply with regulatory requirements just as any other system would. Determining what the design assurance requirements should be for software aspects of those systems was the main focus of the Verification of Adaptive Systems task.

The Phase 1 effort was aimed at understanding the range of adaptive systems and their factors that impact verification. Progress was made in developing a more comprehensive understanding of machine learning in general, how machine learning is used to enable a system to adapt, especially with respect to feedback processes, and where machine learning is being used in various domains, with particular interest in controls applications. Phase 2 significantly extended that knowledge to determine what requirements, especially at the system level, are needed for ensure safe use of adaptive systems.

The Phase 2 work concluded that the software design assurance problem for adaptive systems is 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. Certification of an adaptive system likely cannot 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 suggested are within the scope of the current 14CFR regulatory framework and no major change need be contemplated. Including a structured, evidence based safety case methodology in the update to ARP-4761 is suggested, and inclusion

of this within the current framework is suggested. The principle compliance methodology changes suggested 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.

14. REFERENCES

1. Paul Kaminski, "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. JPDO: NextGen Integrated Work Plan Version 1.0. September 30, 2008, available at http://www.jpdo.gov/newsarticle.asp?id=103, date last visited 8 November 2011.
3. L Harrison, P. Saunders, and J. Janowitz, "Artificial Intelligence with Applications for Aircraft", DOT/FAA/CT-88/10, Chapter 20, FAA Handbook-Volume II, Digital Systems Validation, July 1994.
4. S. H., Stroeve, A. Ypma, J. Spanjers, and P. W. Hoogers, "Neural Network-based Recognition and Diagnosis of Safety-critical Events", Technical Report NLR-CR-2004-501, December 2004.
5. RTCA, Inc., "Software Considerations in Airborne Systems and Equipment Certification", RTCA/DO-178B, 1992.
6. RTCA, "DO-178C - Software Considerations In Airborne Systems And Equipment Certification," RTCA, Washington DC DO-178C, December 13, 2011.
7. Marek Olszewski, Jason Ansel, Saman Amarasinghe, "Kendo: Efficient Deterministic Multithreading in Software", The International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '09), Washington DC, March 2009.
8. G. Frank McCormick, "Adaptive Controls", CSI Document 04-232-1245, Rev. 1.0, Appendix E, October 2004.
9. Yuriy Brun, Giovanna Di Marzo Serugendo, Cristina Gacek, Holger Giese, Holger Kienle, Marin Litoiu, Hausi Muller, Mauro Pezze, and Mary Shaw, "Engineering Self-Adaptive Systems through Feedback Loops", Self-Adaptive Systems, LNCS 5525, pp. 48–70, 2009.
10. Wikipedia, article on "Inductive Inference", available at http://en.wikipedia.org/wiki/Inductive_inference, date last visited 26 November 2011.
11. Zeshan Kurd, "Artificial Neural Networks in Safety Critical Applications", PhD Thesis, Department of Computer Science, University of York, Submitted September 2005.
12. Brian Taylor, Marjorie Darrah, Laura Pullum, Rebecca Giorcelli, Kareem Ammar, and Stacy Nelson, "Guidance for the Independent Verification and Validation of Neural Networks", Technical Report, Institute for Scientific Research, 2005.

DRAFT

13. "Introduction to Neural Networks", available at http://galaxy.agh.edu.pl/~vlsi/AI/intro/, date last visited 26 November 2011.

14. “Principles of training multi-layer neural network using backpropagation”, available at http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html, date last visited 28 November 2011.

15. Dale Mackall, Stacy Nelson, and Johann Schumann, "Verification and Validation of Neural Networks for Aerospace Systems", NASA/CR-2002-211409, June 2002.

16. R. Zakrzewski, "Verification of Performance of a Neural Network Estimator" Proceedings of the 2002 International Joint Conference on Neural Networks, IEEE 0-7803-7278-6/02, May 2002.

17. Jason Hull, David Ward, and Radoslaw R. Zakrzewski, "Verification and Validation of Neural Networks for Safety-Critical Applications", Proceedings of the American Control Conference, Anchorage, AK, May 2002.

18. D. B. Beringer, "Applying Performance-Controlled Systems, Fuzzy Logic, and Fly-By-Wire Controls to General Aviation", DOT/FAA/AM-02/7, May 2002.

19. Wikipedia, article on "Control Theory", available at http://en.wikipedia.org/wiki/Control_theory, date last visited 26 November 2011.

20. C. Cao, and N. Hovakimyan, "Design and Analysis of a Novel L1 Adaptive Control Architecture with Guaranteed Transient Performance", IEEE Transactions on Automatic Control, vol.53, No.2, pp. 586-591, 2008.

21. Kurt Woodham, "Verification of Adaptive Systems, Phase 1 Final Report", 10 February 2012 (unpublished).

22. 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.

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

24. 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

25. D. Busch, R. Bharadwaj, D. Enns, K. Kim, S. Pratt, A. Vechart, and M. Oppenheimer, "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.

DRAFT

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

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

28. 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.

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

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

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

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

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

34. Stephen A. Jacklin, "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.

35. Stephen A. Jacklin, Michael R Lowry, Johann M. Schumann, Pramod P. Gupta, John T. Bosworth, Eddie Zavala, John W. Kelly, Kelly J. Hayhurst, Celeste M. Belcastro, and Christine M. Belcastro, "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 16-19, 2004, Providence, RI.

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

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

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

39. Karl Astrom and Bjorn Wittenmark, Adaptive Control, Second Edition, Prentice Hall, December 1994.

DRAFT

APPENDIX A--TERMINOLOGY

The following list represents a working set of relevant terms and definitions from Phase 1, subject to further revision during Phase 2.

- Activation Function (also, Threshold Function): defines the output of node in a neural network given an input or set of inputs; typically a nonlinear curve such as a hyperbolic tangent, or another sigmoid ("S-shaped") function.

- Active Feedback: Continuously measuring the response of a system to changes in the environment and providing the measurement to the process that produces the stimulus to the system, such as a controller.

- Adapt: To change behavior using an active feedback process to achieve a goal in the presence of changes in the system or its environment.

- Adaptive Controller: A controller with adjustable parameters and a mechanism for adjusting the parameters. The controller becomes nonlinear because of the parameter adjustment mechanism [39].

- Adaptive Control System: An adaptive control system is formed using an adaptive controller to control a physical system that provides the feedback required to support the controller's parameter adjustment mechanism.

- Adaptive System: A system that changes behavior based on an active feedback process to achieve a goal in the presence of changes in the system or its environment.

- Artificial Intelligence: The branch of computer science associated with emulating aspects of human problem solving, perception, and thought.

- Back-propagation: A learning method used in neural networks that uses the derivative of the activation function together with the error formed from the difference between the response and intended response (during training), or between the response and some representation of the intended response generated by a goal function during operations.

- Classification: A method for statistical data analysis that groups input data into one of a number of discrete classes.

- Clustering: A method for statistical data analysis that identifies similarities between features of the data, and groups similar data items into clusters with similar features.

- Controllability: An important property of a control system, related to the ability of an external input (usually a control signal) to move the internal state of a system from any initial state to any other final state in a finite time interval.

DRAFT

- Controller: A computer algorithm that processes input commands and feedback from the dynamic system, and generates commands to the actuators of the dynamic system. The parameters of the controller can be (a) fixed for conventional controllers, (b) provided by an external table-lookup process for gain-scheduled controllers, (c) provided by an external adaptive algorithm, or (d) refined internally by the adaptive controller.

- Deterministic System: A system in which no randomness is involved in the development of future states of the system. Given the same input, a deterministic system will always produce the same output from a given initial state. This contrasts with stochastic or random systems in which future states are not determined from previous ones.

- Direct Adaptive Control: Controller parameters are calculated directly by an adaptive approach.

- Environment: The context in which a system operates.

- Genetic Programming: an evolutionary algorithm methodology inspired by biological evolution to find computer programs that perform a user-defined task.

- Indirect Adaptive Control: Controller parameters are calculated by a process that is separate from the adaptive approach. For instance, an adaptive algorithm may refine estimates of an aerodynamic coefficient that is used subsequently in a fixed (non-adaptive) algorithm to calculate controller parameters.

- Machine learning: a branch of artificial intelligence concerned with the design and development of algorithms that allow computers to evolve behaviors. A major focus of machine learning research is to automatically learn (be trained to) to recognize complex patterns and make intelligent decisions based on data. Statistical techniques are typically employed to achieve these means.

- Middleware: Software that is structured to reside between the operating system and the application, mediating the interaction between them by providing a standard collection of components and services from which to build distributed systems.

- Optimization: the selection of a “best” element from some set of available alternatives.

- Neural Networks: (also Artificial Neural Network: ANN) A network of neurons that are connected according to a prescribed topology. Note that there is no standard structure for neural network topology. However, a common structure consists of an input layer of neurons, connected to one or more intermediate layers (called “hidden layers”), which are connected to an output layer.

- Neuron: a basic computation element of the neural network [10], consisting of a weighted

sum of inputs passed through an activation function.

- Non-deterministic System: A system in which random processes influence the development of future states of the system. If a non-deterministic system is given some initial inputs, the system may produce a different state for each run.

- Observability: Related to the possibility of observing the state of a system through output measurements. In control theory, observability is a measure for how well internal states of a system can be inferred by knowledge of its external outputs.

- Reference Model: a model that describes the desired response of the physical system operating in its environment.

- Regression: a method of statistical analysis that identifies mathematical relationships among variables.

- Reinforcement Learning: Learning in which the system learns by interaction with the environment in order to optimize an accumulated reward. Reinforcement learning occurs by making intermediate decisions to explore previously unselected actions (exploration) or selecting actions based on experience (exploitation).

- Robustness: The quality of maintaining designed controller stability and performance properties in the presence of uncertainties in the system or unmodeled disturbances.

- Self-adaptive reflective middleware: Middleware that employs reflective computation in order to adapt its behavior in response to evolving conditions such as system performance.

- Stability: A system is stable if every bounded (limited) input produces a bounded output. Stability is quantified by gain and phase margins for frequency domain analysis, and by positive damping for time-domain analysis.

- Stochastic Search Method: a search method that makes use of random numbers and is able to find good solutions within reasonable time without guaranteeing the optimum.

- Supervised Learning: Learning that is accomplished with pre-determined sets of data that represent input and response relations. Supervised learning is often called “training,” with the pre-determined data set called “training data.” Supervised learning can occur in an operational setting if a model provides the intended input/response relationship, given the same input provided to the system.

- System: A collection of hardware and software components organized to accomplish a specific function or set of functions. [1]

- Time to adaptation: The time scale required for an adaptive system to respond to changes in the environment and modify its parameters. Note that for adaptive control systems, time to adaptation can define a transient period during which the system may experience reduced stability or instability.

- Training: Learning using pre-operation knowledge that defines the appropriate response to predefined stimuli.

- Training Data: Data used to provide the neural network with pre-operation knowledge. This data allows the neural network to learn, by adjusting certain parameters, the appropriate response to predefined stimuli. [10]

- Unsupervised Learning: Learning that applies statistical techniques to identify non-obvious relationships within the data.

DRAFT

APPENDIX B--LITERATURE SEARCH RESULTS

The literature search for this effort has culminated in 206 references, including textbooks, conference proceedings, journal publications, standards and guidelines, industry papers, and academic sites, as well as other on-line resources. These references were documented in a wiki created specific for this task, and are listed below.

1. Ulf Ahlstrom and Ferne Friedman-Berg, "Subjective Workload Ratings and Eye Movement Activity Measures", DOT/FAA/CT-05/32, 2005.
2. Robert Alexander, Martin Hall-May, Tim Kelly and John McDermid, "Safety Cases for Advanced Control Software: Final Report", University of York, June 2007.
3. Robert Alexander, Tim Kelly, Zeshan Kurd, and John McDermid, "Safety Cases for Advanced Control Software: Safety Case Patterns", Department of Computer Science, University of York, October 2007.
4. Robert Alexander, Martin Hall-May, and Tim Kelly, "Certification of Autonomous Systems", 2nd SEAS DTC Technical Conference, Edinburgh, UK, 2007.
5. R. Alexander, M Hall-May, T Kelly, "Certification of Autonomous Systems under UK Military Safety Standards", Proceedings of the 25th International System Safety Conference (ISSC '07), August 2007.
6. Swantje Amelsberg, "Pilot model for wake vortex encounter simulations for take-off and departure", Presented at the Models and Methods for Wake Vortex Encounter Simulations Workshop, Technical University of Berlin, Germany, June 2010.
7. Brian D. O. Anderson, "Failures of Adaptive Control Theory and Their Resolution", Communications in Information and Systems, Vol. 5, No. 1, pp. 1-20, 2005.
8. Dave Anderson and George McNeill, "Artificial Neural Networks Technology, A DACS State-of-the-Art Report", Prepared for: Rome Laboratory, RL/C3C Griffiss AFB, NY, August 20 1992.
9. Jesper Andersson, Rogerio de Lemos, Sam Malek, and Danny Weyns, "Towards a Classification of Self-Adaptive Software Systems", Computer, Vol. 5525, 2009.
10. Robert Andrews, Joachim Diederich and Alan B. Tickle, "Survey and critique of techniques for extracting rules from trained artificial neural networks", Knowledge-Based Systems, Vol. 8, Issue 6, pp. 373-389, December 1995.
11. Karl Astrom and Bjorn Wittenmark, Adaptive Control, Second Edition, Prentice Hall, December 1994.
12. Algirdas Avizienis, Jean-Claude Laprie, and Brian Randell, "Fundamental Concepts of Computer System Dependability", IARP/IEEE-RAS Workshop on Robot Dependability: Technological Callende of Dependable Robots in Human Environments, Seoul, Korea, May 21-22, 2001.
13. Algirdas Avizienis, Jean-Claude Laprie, Brian Randell, and Carl Landwehr, "Basic Concepts and Taxonomy of Dependable and Secure Computing", IEEE Transactions on Dependable and Secure Computing, Vol. 1, No. 1, January-March 2004.

DRAFT

14. Mario Barbacci, Mark H. Klein, Thomas A. Longstaff, and Charles B. Weinstock, "Quality Attributes", Technical Report, CMU/SEI-95-TR-021, ESC-TR-95-02, December 1995.

15. Mario R. Barbacci, "Analyzing Quality Attributes", News at SEI, available at http://www.sei.cmu.edu/library/abstracts/news-at-sei/architectmar99.cfm, March 1, 1999.

16. H. Basher and J. S. Neal, "Autonomous Control of Nuclear Power Plants", Technical Report ORNL/TM-2003/252, Nuclear Science and Technology Division, 2003.

17. Ellen J. Bass, Samuel T. Ernst-Fortin, Ronald L. Small, and James Hogans, Jr., "Architecture and Development Environment of a Knowledge-Based Monitor That Facilitate Incremental Knowledge-Base Development", IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans, Vol. 34, No. 4, July 2004.

18. D. F. Bedford, G. Morgan, and J. Austin, "Requirements for a Standard Certifying the use of Artificial Neural Networks in Safety Critical Applications", Proceedings of the International Conference on Artificial Neural Networks, 1996.

19. Alireza R. Behbahani, "Advanced, Adaptive, Modular, Distributed, Generic Universal FADEC Framework for Intelligent Propulsion Control Systems", Technical Report AFRL-RZ-WP-TP-2008-2044, Air Force Research Laboratory, Wright-Patterson Air Force Base, OH, September 2007.

20. Yoshua Bengio, "Learning Deep Architectures for AI", Foundations and Trends in Machine Learning, Vol. 2, Number 1, 2009.

21. Yoshua Bengio and Yann LeCun, "Scaling Learning Algorithms towards AI", Large-Scale Kernel Machines", MIT Press, 2007.

22. Dennis B. Beringer, "Applying Performance-Controlled Systems, Fuzzy Logic, and Fly-By-Wire Controls to General Aviation", Final Report, Civil Aerospace Medical Institute, FAA, Oklahoma City, OK, May 2002.

23. C. M. Bishop, "Novelty detection and neural network validation", IEE Proceeding of Vision, Image and Signal Processing, Vol. 141, No. 4, pp. 217-222, August 1994.

24. M. Bodson and J. E. Groszkiewicz, "Multivariable Adaptive Algorithms for Reconfigurable Flight Control", IEEE Transactions on Control Systems Technology, Vol. 5, Issue: 2, pp. 217 – 229, March 1997.

25. Yuriy Brun, Giovanna Di Marzo Serugendo, Cristina Gacek, Holger Giese, Holger Kienle, Marin Litoiu, Hausi Muller, Mauro Pezze, and Mary Shaw, "Engineering Self-Adaptive Systems through Feedback Loops", Self-Adaptive Systems, LNCS 5525, pp. 48–70, 2009.

26. James Buffington, "Validation and Verification of Intelligent and Adaptive Control Systems (VVIACS)", Technical Report AFRL-VA-WP-TP-2003-327, Sep 2003.

27. John J. Burken, Peggy Williams-Hayes, John T. Kaneshige and Susan J. Stachowiak, "Adaptive Control Using Neural Network Augmentation for a Modified F-15 Aircraft", 14th Mediterranean Conference on Control and Automation, Ancona, Italy, 28-30 June 2006.

DRAFT

28. John J. Burken, Curtis E. Hanson, James A. Lee, and John T. Kaneshige, "Flight Test Comparison of Different Adaptive Augmentations of Fault Tolerant Control Laws for a Modified F-15 Aircraft", AIAA AIAA Infotech@Aerospace Conference, April 2009.

29. Keith J Burnham and Olivier C L Haas (editors), "Intelligent and Adaptive Systems in Medicine", Taylor & Francis, 2008.

30. Massimo Buscema, "The General Philosophy of Artificial Adaptive Systems", Fuzzy Information Processing Society, NAFIPS 200 Annual Meeting of the North American, May 2008.

31. Anthony J. Calise, "Neural Network Based Adaptive Control of Uncertain and Unknown Nonlinear Systems", Technical Report for Air Force Office of Scientific Research, 2003.

32. C. Cao, and N. Hovakimyan, "Design and Analysis of a Novel L1 Adaptive Control Architecture with Guaranteed Transient Performance", IEEE Transactions on Automatic Control, vol.53, No. 2, pp. 586-591, 2008.

33. Chengyu Cao and Naira Hovakimyan, "Design and Analysis of a Novel L1 Adaptive Controller, Part I: Control Signal and Asymptotic Stability", Proceedings of the 2006 American Control Conference, Minneapolis, Minnesota, June 14-16, 2006.

34. Chengyu Cao and Naira Hovakimyan, "Design and Analysis of a Novel L1 Adaptive Controller, Part II: Guaranteed Transient Performance", Proceedings of the 2006 American Control Conference, Minneapolis, Minnesota, June 14-16, 2006.

35. Chengyu Cao and Naira Hovakimyan, "Stability Margins of Adaptive Control Architecture", IEEE Transactions on Automatic Control, Vol. 55, No. 2, February 2010.

36. Chengyu Cao, Vijay V. Patel, C. Konda Reddy, Naira Hovakimyan, Eugene Lavretsky, and Kevin Wise, "Are Phase and Time-delay Margins Always Adversely Affected by High-Gain?", AIAA Guidance, Navigation, and Control Conference and Exhibit, Keystone, Colorado, AIAA 2006-6347, August 2006.

37. Rich Caruana and Alexandru Niculescu-Mizil, "An Empirical Comparison of Supervised Learning Algorithms", Proceedings of the 23rd International Conference on Machine Learning, Pittsburgh, PA, 2006.

38. Rajeev Chandramohan, James E. Steck, Kamran Rokhsaz, and Silvia Ferrari, "Adaptive Critic Flight Control For a General Aviation Aircraft: Simulations For The Beech Bonanza Fly-By-Wire Test Bed", AIAA Infotech@Aerospace Conference and Exhibit, Rohnert Park, California, May 2007.

39. Betty H.C. Cheng, Rogerio de Lemos, Holger Giese, Paola Inverardi, and Jeff Magee, "Software Engineering for Self-Adaptive Systems: A Research Road Map", Self-Adaptive Systems, LNCS 5525, pp. 1–26, 2009.

40. George Cheng, "Model-Free Adaptive Control: Impact on the Automation Industry", Control Engineering – Europe Magazine, 24 January 2010.

41. Wing Ho Cheung, "Neural Network Aided Aviation Fuel Consumption Modeling", Thesis: Master of Science in Civil Engineering, Virginia Tech, Blacksburg, VA, August 1997.

DRAFT

42. David W. Clarke, "Self Tuning Control", The Control Handbook, Chapter 53, CRC Press, 1996.

43. Vittorio Cortellessa, Bojan Cukic, Diego Del Gobbo, Ali Mili, Marcello Napolitano, Mark Shereshevsky, and Harjinder Sandhu, "Certifying Adaptive Flight Control Software", Proceedings of ISACC 2000: The Software Risk Management Conference, Reston, VA, September 2000.

44. V. W. Crum, J. M. Buffington, G. S. Tallant, B. Krogh, C. Plaisted, R. Prasanth, P. Bose, and T. Johnson, "Validation and Verification of Intelligent and Adaptive Control Systems", Aerospace Conference, Big Sky, Montana, March 2004.

45. Ivica Crnkovic and Magnus Larsson, "Classification of quality attributes for predictability in component-based systems", IEEE Conference on Dependable Systems and Networks, Workshop on Architecting Dependable Systems, Florence, Italy, June 2004.

46. Bojan Cukic, "The Need for Verification and Validation Techniques for Adaptive Control System", 5th International Symposium on Autonomous Decentralized Systems, Dallas, TX, 2001.

47. Dennis Culley and Sanjay Garg, "Distributed Control Architecture for Gas Turbine Engine", NATO report on More Intelligent Gas Turbine Engines, RTO Technical Report TR-AVT-128, April 2009.

48. Marjorie Darrah, Brian Taylor, and Spiro Skias, "Rule Extraction From Dynamic Cell Structure Neural Networks Used in a Safety Critical Application", Proceedings of FLAIRS Conference, 2004.

49. Xiong Deng, Moustafa Ghanem, and Yike Guo, "Real-time Data Mining Methodology and a Supporting Framework", Third International Conference on Network and System Security, Gold Coast, Queensland, Australia, October 2009.

50. Noel Duerksen, "Fuzzy Logic Decoupled Longitudinal Control for General Aviation Airplanes", NASA Contractor Report 2011639, December 1996.

51. Guy A. Dumont and Mihai Huzmezan, "Concepts, Methods and Techniques in Adaptive Control", Proceedings of the 2002 American Control Conference, Vol.2, pp. 1137-1150, Anchorage, AK, May 2002.

52. Zachary T. Dydek, Anuradha M. Annaswamy, and Eugene Lavretsky, "Adaptive Control and the NASA X-15-3 Flight Revisited", IEEE Control Systems Magazine, June 2010.

53. H. Engström and A. Kaplan, "Adaptive Process Control in Laser Robotic Welding", 9th Conference on Laser Materials Processing in the Nordic Countries, pp. 251-258, August 2003.

54. "Artificial Intelligence with Applications for Aircraft: An Overview", FAA Update Edition 21, p. 8, Fall 1996.

55. FAA Office of Commercial Space Transportation, "FY-2005 Research and Development Accomplishments", FAA Report HQ-026505, p. 4, October 2005.

56. FAA NextGen Integration and Implementation Office, "NextGen Implementation Plan

DRAFT

2010", March 2010.

57. FAA NextGen Integration and Implementation Office, "NextGen Implementation Plan 2011", March 2011.

58. FAA, "Lessons Learned from Transport Airplane Accidents: TACA International Airlines 737 near New Orleans", available at http://accidents-ll.faa.gov/ll_main.cfm?TabID=1&LLID=40&LLTypeID=2, date last visited 28 November 2011.

59. FAA Research, Engineering and Development Advisory Committee (REDAC), “Report of the Weather-ATM Integration Working Group”, October 2007.

60. Gang Feng and Rogelio Lozano, "Adaptive Control Systems", Elsevier Newnes, 1999.

61. Roman D. Fresnedo, "Statistics and the Verification Validation & Testing of Adaptive Systems", International Joint Conference on Neural Networks, Montreal Canada, August 5, 2005.

62. Mohamed Medhat Gaber, Arkady Zaslavsky, and Shonali Krishnaswamy, "Mining Data Streams: A Review", ACM: Special Interest Group on Management of Data, Vol. 34, No. 2, June 2005.

63. Bill Gough and Don Matovich, "Predictive-Active Temperature Control of Molten Glass", The Glass Industry, Vol. 82, No. 4, pp. 33-37, 2001.

64. Bill Gough, Sava Kovac, Jim LeBlanc, and Barry Deobald, "Advanced Predictive Adaptive Control of Steam Header Pressure, Saveall Consistency, and Reel Brightness in a TMP Newsprint Mill", undated white paper, available at http://www.isa.org/~pupid/Gough_etal_Adv_Pred_Adapt_Ctrl_Stm_Hdr_Press_Saveall_Consistency_&_Reel_Bright_TMP_Newsprint_Mill.pdf, date last visited 27 November 2011.

65. B. Gough, D. Meade, G. England, and S. Kovac, "Model-based predictive adaptive control of pulp and paper mill processes", Technical Association of the Pulp and Paper Industry (TAPPI) Papermakers & Paper Industry Management Association (PIMA) International Leadership Conference, Jacksonville, Florida, March 2007.

66. Irene M. Gregory, Chengyu Cao, Enric Xargay, Naira Hovakimyan, and Xiaotian Zou, "L1 Adaptive Control Design for NASA AirSTAR Flight Test Vehicle", AIAA Guidance, Navigation, and Control Conference, August 2009.

67. Kurt D. Guenther, "SMART-T Project Overview", NASA Office of Safety and Mission Assurance (OSMA) Software Assurance Symposium (SAS), July 19, 2004.

68. Pramod Gupta and Johann Schumann, "A Tool for Verification and Validation of Neural Network Based Adaptive Controllers for High Assurance Systems", Proceedings of the IEEE High Assurance Software Engineering (HASE) Conference, Tampa, FL, March 2004.

69. P. Gupta, K.A. Loparo, D. Mackall, J. Schumann, and F. R. Soares, "Verification and Validation Methodology of Real-time Adaptive Neural Networks for Aerospace Applications", International Conference on Computational Intelligence for Modeling, Control, and Automation, 2004.

DRAFT

70. Jose Antonio Martin H., Javier de Lope, and Dario Maravall, "Adaptation, anticipation and rationality in natural and artificial systems: computational paradigms mimicking nature", Natural Computing, Vol. 8, No. 4, pp. 757-775, August 2008.

71. Jacob J. Hageman, Mark S. Smith, and Susan Stachowiak, "Integration of Online Parameter Identification and Neural Network for In-Flight Adaptive Control", NASA/TM-2003-212028, October 2003.

72. L Harrison, P. Saunders, and J. Janowitz, "Artificial Intelligence with Applications for Aircraft", DOT/FAA/CT-88/10, Chapter 20, FAA Handbook-Volume II, Digital Systems Validation, July 1994.

73. M. Tarafdar Haque, and A.M. Kashtiban, "Application of Neural Networks in Power Systems; A Review", World Academy of Science, Engineering and Technology, Vol. 6, 2005.

74. Tomohisa Hayakawa, "Direct Adaptive Control for Nonlinear Uncertain Dynamical Systems", PhD Dissertation, Georgia Institute of Technology, November 2003.

75. David Homan, "AFRL: Flight Critical Systems Software Certification Initiative", Presentation to SAE Aerospace Control and Guidance Systems Committee, Meeting 95, Salt Lake City, UT, 2 March 1995.

76. J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities", Proceedings of the National Academy of Sciences of the USA, Vol. 79, no. 8, pp. 2554-2558, April 1982.

77. Tomas Hrycej, "Neural-Network-Based Car Drive Train Control", IEEE 42nd Vehicular Technology Conference, Vol. 2, pp. 1042-1045, 10-13 May 1992.

78. E. Huh, “Certification of Real-Time Performance for Dynamic, Distributed Real Time Systems”, Ph.D. Dissertation, Department of Electrical Engineering, Ohio University, 2002.

79. Jason Hull, David Ward, and Radoslaw R. Zakrzewski, "Verification and Validation of Neural Networks for Safety-Critical Applications", Proceedings of the American Control Conference, Anchorage, AK, May 2002.

80. Mihai Huzmezan, Bill Gough, Sava Kovac, Long Le, Gary Roberts, "A New Generation of Adaptive Model Based Predictive Controllers Applied in Batch Reactor Temperature Control", undated white paper.

81. Moshe Idan, Matthew Johnson, Anthony J. Calise, and John Kaneshige, "Intelligent Aerodynamic/Propulsion Flight Control for Flight Safety: A Nonlinear Adaptive Approach", 2001 American Control Conference, Arlington, Virginia, June 2001.

82. Moshe Idan, Matthew Johnson, and Anthony J. Calise, "A Hierarchical Approach to Adaptive Control for Improved Flight Safety", AIAA Journal on Guidance, Control and Dynamics, July 2001.

83. International Electrotechnical Commission, "Functional safety of electrical/electronic/ programmable electronic safety-related systems", IEC-61508, 2010.

84. Petros A. Ioannou and Jing Sun, "Robust Adaptive Control", Prentice Hall, Inc, 1996.

DRAFT

85. Joint Planning and Development Office, “NextGen Integrated Work Plan”, Version 1.0, September 2008.

86. Joint Planning and Development Office, “Security Annex, Concept of Operations for the Next Generation Air Transportation System”, Version 2.0, June 2007.

87. Joint Planning and Development Office, “NextGen Weather Plan”, Version 2.0, October 2010.

88. Joint Planning and Development Office, “Concept of Operations for the Next Generation Air Transportation System”, Version 3.2, September 2010.

89. Stephen Jacklin, Michael Lowry, Johann Schumann, Pramod Gupta, John Bosworth, Eddie Zavala, John Kelly, Kelly Hayhurst, Celeste Belcastro, and Christine Belcastro, "Verification, Validation and Certification Challenges for Adaptive Flight Control Systems Software," Proceedings of AIAA Guidance, Navigation, and Control Conference and Exhibit, Providence, RI, Aug. 16-19 2004.

90. Stephen Jacklin, "Closing the Certification Gaps in Adaptive Flight Control Software", AIAA 2008 Guidance, Navigation, and Control Conference, Hawaii, 18-21 August 2008.

91. Stephen Jacklin, Johann Schumann, Pramod Gupta, Michael Richard, Kurt Guenther, and Fola Soares, "Development of Advanced Verification and Validation Procedures and Tools for the Certification of Learning Systems in Aerospace Applications", AIAA,Infotech@Aerospace Conference, 2005.

92. Daniel Jackson, Martyn Thomas, and Lynette I. Millett (Editors), "Software for Dependable Systems: Sufficient Evidence?", Committee on Certifiably Dependable Software Systems, National Research Council, The National Academies Press, 2007.

93. Jan Jantzen, "A Tutorial on Adaptive Fuzzy Control", EUNITE, 2002.

94. L. Jaw, W. T. Tsai, D. Homan, and K. Keller, "Verification of Flight Software with Karnough Map-based Checking", Aerospace Conference, Big Sky, MT, March 2007.

95. Link C. Jaw and Sanjay Garg, "Propulsion Control Technology Development in the United States - A Historical Perspective", NASA/TM-2005-213978, October 2005.

96. Eric N. Johnson and Suresh K. Kannan, "Adaptive Flight Control for an Autonomous Unmanned Helicopter", AIAA Guidance, Navigation, and Control Conference and Exhibit, Monterey, California, 5-8 August 2002.

97. Eric N. Johnson, Anthony J. Calise, and Hesham A. El-Shirbiny, and Rolf T. Rysdyk, "Feedback Linearization with Neural Network Augmentation Applied to X-33 Attitude Control", Proceedings of the AIAA Guidance, Navigation, and Control Conference, AIAA-2000-4157, 2000.

98. Jih-Gau Juang, Bo-Shian Lin and Kuo-Chih Chin, "Intelligent Fuzzy Systems for Aircraft Landing Control", Lecture Notes in Computer Science, Vol. 3613, pp. 851-860, 2005.

99. Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore, "Reinforcement Learning: A Survey", Journal of Artificial Intelligence Research, 1996.

DRAFT

100. David A. Karr, Robert A. Vivona, David A. Roscoe, Stephen M. DePascale, and Maria Consiglio, "Experimental Performance of a Genetic Algorithm for Airborne Strategic Conflict Resolution", Eighth USA/Europe Air Traffic Management Research and Development Seminar, 2009.

101. Howard Kaufman, Gurbux Alag, Paul Berry, and Samir Kotob, "Digital Adaptive Flight Controller Development", NASA-CR-141041, November 1974.

102. David Kim and Maciej Marciniak, "A Methodology to Predict the Empennage In-Flight Loads of a General Aviation Aircraft Using Backpropagation Neural Networks", DOT/FAA/AR-00/50, February 2001.

103. David Kim and Laure Pechaud, "Improved Methodology for the Prediction of the Empennage Maneuver In-Flight Loads of a General Aviation Aircraft Using Neural Networks", DOT/FAA/AR-01/80, December 2001.

104. Ron Kohavi and Foster Provost (Editors), "Special Issue on Applications of Machine Learning and the Knowledge Discovery Process", Machine Learning, 30, 271-27, 1998.

105. Tuevo Kohonen, "Self-Organized Formation of Topologically Correct Feature Maps", Biological Cybernetics, Springer-Verlag, 1982.

106. Z. Kurd, J. Austin, and T. P. Kelly, "Developing Artificial Neural Networks for Safety Critical Applications", Proceedings of Eunite'2003 - European Symposium on Intelligent Technologies, Hybrid Systems and their implementation on Smart Adaptive Systems, Oulu, Finland, July 2003.

107. Zeshan Kurd, Tim Kelly, and Jim Austin, "Developing artificial neural networks for safety critical systems", Neural Computing & Application, Vol. 16, No. 1, pp. 11–19, March 2006.

108. Zeshan Kurd, "Artificial Neural Networks in Safety Critical Applications", PhD Thesis, Department of Computer Science, University of York, Submitted September 2005.

109. Zeshan Kurd and Tim Kelly, "Safety Lifecycle for Developing Safety Critical Artificial Neural Networks", Lecture Notes in Computer Science, Vol. 2788, pp. 77-91, 2003.

110. Anthony A. Lambregts, "Automatic Flight Control Systems, An Interactive Video Teletraining and Self-Study Course", FAA IVT Training Course, January 27, 1999.

111. A. Lambregts, "Automatic Flight Controls, Concepts and Methods", white paper available at https://faaco.faa.gov/attachments/KNVLPAP6.pdf, date last visited on 27 November 2011.

112. Anthony A. Lambregts, "Vertical Flight Path and Speed Control Autopilot Design Using Total Energy Principles", Technical report, AIAA paper 83-2239, 1983.

113. Gregory V. Larchev, Stefan F. Campbell, and John T. Kaneshige, "Projection Operator: A Step Towards Certification of Adaptive Controllers", AIAA Infotech@Aerospace Conference, Atlanta, GA, 20-22 April 2010.

114. Eugene Lavretsky and Naira Hovakimyan, "Adaptive Compensation of Control Dependent Modeling Uncertainties using Time-Scale Separation", Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference, Seville, Spain, December 12-15, 2005.

DRAFT

115. Sunro Lee and Robert M. O'Keefe, Developing a Strategy for Expert System Verification and Validation", IEEE Transactions on Systems, Man and Cybernetics, Vol. 24, No. 4, April 1994.

116. F. Lewis, K. Liu, and A. Yesildirek, "Neural Net Robot Controller with Guaranteed Tracking Performance", IEEE Transactions on Neural Networks, Vol. 6, No. 3, pp. 703-715, May 1995.

117. Dale Lillfors, Chris Basset, Ray Akkerman, and Sava Kovac, "Blandin Paper Co: Blandin Reduces Variability in Pulp Brightness with Adaptive Controller", Pulp & Paper, January 2003.

118. P. J. G. Lisboa, "Neural Networks: Current Applications", Chapman & Hall, 1992.

119. Paulo J. G. Lisboa, "Industrial use of safety-related artificial neural networks", Contract Research Report 327 for the Health and Safety Executive, 2001.

120. Yan Liu, Bojan Cukic,and Srikanth Gururajan, "Validating neural network-based online adaptive systems: a case study", Software Quality Journal, Vol. 15, pp. 309-326, 2007.

121. Yan Liu, Bojan Cukic, Edgar Fuller, Sampath Yerramalla, and Srikanth Gururajan, "Monitoring techniques for an online neuro-adaptive controller", The Journal of Systems and Software, Vol. 79, pp. 1527-1540, 2006.

122. James T. Luxhøj and Jiangfeng Cheng, "Neural Network Modeling of Aviation Safety Field Studies", Proceedings of the 7th Annual Industrial Engineering Research Conference, Banff, Alberta, Canada, May 9-10, 1998.

123. David Lyon, Bill Gough, and Malcolm Cameron, "Implementation of an Innovative Self-Tuning Adaptive Controller for Complex Industrial Processes", undated white paper available at http://www.andritzautomation.com/documents/implementation-avpforindustry.pdf, date last visited 28 November 2011.

124. Dale Mackall, Stacy Nelson, and Johann Schumann, "Verification and Validation of Neural Networks for Aerospace Systems", NASA/CR-2002-211409, June 2002.

125. G. Frank McCormick, "Adaptive Controls", CSI Document 04-232-1245, Rev. 1.0, Appendix E, October 2004.

126. Warren S. McCulloch and Walter Pitts, "A Logical Calculus of the Ideas Immanent In Nervous Activity", Bulletin of Mathematical Biophysics, Vol. 5, 1943.

127. Michael B. McFarland and Anthony J. Calise, "Neural-adaptive nonlinear autopilot design for an agile anti-air missile", AIAA, Guidance, Navigation and Control Conference, San Diego, CA, July 29-31, 1996.

128. Philip. K. McKinley, R. E. Kurt Stirewalt, Betty H. C. Cheng, Laura K. Dillon and Sandeep Kulkarni, "RAPIDware: Component-Based Development of Adaptive and Dependable Middleware", Technical report, Michigan State University, August 2005.

129. Ali Mili, GuangJie Jiang, Bojan Cukic, Yan Liu, and Rahma Ben Ayed, "Towards the Verification and Validation of Online Learning Systems: General Framework and Applications", Proceedings of the 37th Hawaii International Conference on System Sciences, Big Island, Hawaii, January 2004.

DRAFT

130. Wang Mingmei, Cheng Qiming, Cheng Yinman, and Wang Yingfei, "Model-Free Adaptive Control Method for Nuclear Steam Generator Water Level", 2010 International Conference on Intelligent Computation Technology and Automation, Changsha, Hunan, China, May 2010.

131. Tom M. Mitchell, "The Discipline of Machine Learning", Technical Report CMU-ML-06-108, Carnegie Mellon University School of Computer Science, Machine Learning Department, July 2006.

132. Edoardo Mosca, "Optimal, Predictive, and Adaptive Control", Prentice Hall, 1995.

133. Rainer Muller, Hans-Hubert Hemberger, and Karlheinz Baier, "Engine Control using Neural Networks: A New Method in Engine Management Systems", Meccanica, Vol. 32, No. 5, pp. 423–430, 1997.

134. Jim Murtha, "Applications of fuzzy logic in operational meteorology", Scientific Services and Professional Development Newsletter, Canadian Forces Weather Service, pp. 42-54, 1995.

135. Nils J. Nilsson, "Introduction to Machine Learning - Draft of Incomplete Notes" Classroom notes, available at: http://ai.stanford.edu/~nilsson/mlbook.html, date last visited 28 November 2011.

136. Nhan T. Nguyen and Stephen A. Jacklin, "Neural Net Adaptive Flight Control Stability, Verification and Validation Challenges, and Future Research", Applications of Neural Networks in High Assurance Systems, pp. 77-107, Springer-Verlag, Berlin, 2010.

137. Nhan Nguyen and Kalmanje Krishnakumar, "A Hybrid Intelligent Flight Control with Adaptive Learning Parameter Estimation", AIAA Infotech@Aerospace 2007 Conference and Exhibit, Rohnert Park, California, AIAA 2007-2841, 7-10 May 2007.

138. P.Y. Oh, "Improved Model Reference Adaptive Control of Electro-hydraulic Servo Systems Using The Euler Operator", IEEE International Conference on Robotics and Automation (ICRA), Albuquerque, New Mexico, pp. 1626-1631, 1997.

139. Robert M. O'Keefe and Daniel E. O'Leary, "Expert system verification and validation: a survey and tutorial", Artificial Intelligence Review, Vol. 7, pp. 3–42, 1993.

140. Marek Olszewski, Jason Ansel, and Saman Amarasinghe, "Kendo: Efficient Deterministic Multithreading in Software", Architectural Support for Programming Languages and Operating Systems (ASPLOS ’09), Washington, DC, March 7-11, 2009.

141. Charles Pecheur and Willem Visser, "RIACS Workshop on the Verification and Validation of Autonomous and Adaptive Systems", NASA/TM-2001-210927, August 2001.

142. Radu-Emil Precup and Stefan Preitl, "Stability and Sensitivity Analysis of Fuzzy Control Systems - Mechatronics Applications", Acta Polytechnica Hungarica, Vol. 3, No. 1, 2006.

143. Danil V. Prokhorov, "Toyota Prius HEV Neurocontrol and Diagnostics", Neural Networks, Vol. 21, pp. 458–465, Special Issue 2008.

144. Laura L. Pullum, Marjorie A. Darrah, and Brian J. Taylor, "Independent Verification

DRAFT

and Validation of Neural Networks - Developing Practitioner Assistance", DoD Software Tech News, Vol. 7, Number 2, July 2004.

145. R. L. Ribler, J. S. Vetter, H. Simitci, and D. A. Reed, "Autopilot: Adaptive Control of Distributed Applications", Seventh IEEE International Symposium on High Performance Distributed Computing (HPDC-7 '98), Chicago, IL, 28-21 July 1009.

146. Michael Richard, "Certification of Adaptive Flight Control Software", Powerpoint presentation given at 2005 FAA National Software and Complex Electronic Hardware Standardization Conference, Norfolk, Virginia, July 26-28, 2005.

147. David Robertson and John Fox, "Industrial use of safety-related expert systems", Contract Research Report 296/2000, Prepared by the Division of Informatics, University of Edinburgh and the Advanced Computation Laboratory, Imperial Cancer Research Fund for the Health and Safety Executive, 2000.

148. David M Rodvold, "A Software Development Process Model for Artificial Neural Networks in Critical Applications", International Joint Conference on Neural Networks (IJCNN '99), pp. 3317-3322, 1999.

149. David M. Rodvold, "Validation and Regulation of Medical Neural Networks", Molecular Urology. Vol. 5, Issue 4, pp. 141-145, July 8, 2004.

150. RTCA, Inc., "Software Considerations in Airborne Systems and Equipment Certification", RTCA/DO-178B, 1992.

151. John Rushby, "A Safety-Case Approach for Certifying Adaptive Systems", AIAA Infotech@Aerospace Conference, Seattle, WA, April 2009.

152. John Rushby, "How Do We Certify For The Unexpected?", AIAA Guidance, Navigation and Control Conference and Exhibit, Honolulu, Hawaii, AIAA 2008-6799, 18 - 21 August 2008.

153. John Rushby, "Runtime Certification", The Eighth Workshop on Runtime Verification (RV2008), Budapest, Hungary, April 2008.

154. John Rushby, "Critical System Properties: Survey and Taxonomy" Reliability Engineering and System Safety, Vol. 43, No. 2, pp. 189-219, 1994.

155. John Rushby, "Quality Measures and Assurance for AI Software", Technical Report CSL-88-7R, September 1988.

156. SAE International, "Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment", ARP 4761, 1996.

157. SAE International, "Certification Considerations for Highly-Integrated or Complex Aircraft Systems", ARP 4754, 1996.

158. V. Santhanam, "Can Adaptive Flight Control Software be Certified to DO-178B Level A?", 2005 FAA National Software and Complex Electronic Hardware Standardization Conference, Norfolk, Virginia, July 26-28, 2005.

159. Shankar Sastry and Marc Bodson, "Adaptive Control: Stability, Convergence, and Robustness", Prentice-Hall, 1994.

160. A. Schulte, "Cognitive Automation for Tactical Mission Management: Concept and

DRAFT

Prototype Evaluation in Flight Simulator Trials", Cognition, Technology & Work, Vol. 4, pp. 146–159, 2002.

161. Johann Schumann, Pramod Gupta, and Stacy Nelson, "On Verification & Validation of Neural Network Based Controllers", Proceedings of International Conference on Engineering Applications of Neural Networks (EANN 2003), Vol. 3, pp. 401–47, 2003.

162. Johann Schumann, Stacy Nelson "Toward V&V of Neural Network Based Controllers", Proceedings of the first Workshop on Self-Healing Systems, Charleston, SC, pp. 67-72, 2002.

163. Johann Schumann and Yan Liu, (Eds.), "Applications of Neural Networks in High Assurance Systems", Studies in Computational Intelligence, Vol. 268, 2010.

164. Johann Schumann and Pramod Gupta, "Bayesian Verification and Validation Tools for Adaptive Systems", Powerpoint presentation available at http://sarpresults.ivv.nasa.gov/ViewResearch/54.jsp, 2006.

165. Frederick T. Sheldon, and Ali Mili, "Characterization of Software Quality Assurance Methods: Five Methods for Verification of Learning Systems", Workshop on Verification, Validation, and Testing of Learning Systems in conjunction with the 8th Annual Conference on Neural Information Processing Systems Conference, Whistler, British Columbia, Canada, December 16-18, 2004.

166. Huan-Jyh Shyur, James T. Luxhoj, and Trefor P. Williams, "Using Neural Networks to Predict Component Inspection Requirements for Aging Aircraft", Computers and Industrial Engineering, Vol. 30, No. 2, pp. 257-267, 1996.

167. Huan-Jyh Shyur, James T. Luxhøj, and Trefor P. Williams, "Use of Neural Networks for Aviation Safety Risk Assessment", Proceedings of the FAA-NASA Symposium on the Continued Airworthiness of Aircraft Structures, Atlanta, Georgia, August 28-30, 1997.

168. Fola Soares, John Burken, and Tshilidzi Marwala, "Neural Network Applications in Advanced Aircraft Flight Control System, a Hybrid System, a Flight Test Demonstration", Lecture Notes in Computer Science, 2006, Vol. 4234, pp. 684-691, 2006.

169. Fola Soares and John Burken, "A Flight Test Demonstration of On-line Neural Network Applications in Advanced Aircraft Flight Control System", International Conference on Computational Intelligence for Modeling Control and Automation and International Conference on Intelligent Agents Web Technologies and International Commerce (CIMCA'06), Sydney, Australia, November 28-December 01, 2006.

170. Fola Soares, Kenneth A. Loparo, John Burken, Stephen Jacklin, and Pramod Gupta, "Verification and Validation of Real-time Adaptive Neural Networks using ANCT Tools and Methodologies", Infotech@Aerospace Conference, Arlington, Virginia, AIAA 2005-6997, 26-29 September 2005.

171. Don Soloway and Pam Haley, "Pilots Rate Augmented Generalized Predictive Control for Reconfiguration", AIAA Space 2000 Conference and Exposition, AIAA-2000-5328, September 19-21, 2000.

172. James E. Steck, Kamran Rokhsaz, U. J. Pesonen, Sam Bruner, and Noel Duerksen,

DRAFT

"Simulation and Flight Test Assessment of Safety Benefits and Certification Aspects of Advanced Flight Control Systems", DOT/FAA/AR-03/51, August 2003.

173. Walter A. Storm, Mike Whalen, Darren Cofer, and Bruce Krogh, "Certification Techniques for Advanced Flight Critical Systems Final Report", Final Report for AFRL/RBCC, Wright-Patterson AFB, OH, (Note: Restricted Distribution).

174. S. H., Stroeve, A. Ypma, J. Spanjers, and P. W. Hoogers, "Neural Network-based Recognition and Diagnosis of Safety-critical Events", Technical Report NLR-CR-2004-501, December 2004.

175. Greg S. Tallant, James M. Buffington, and Bruce Krogh, "AFRL: Validation and Verification of Intelligent and Adaptive Control Systems (VVIACS) - Final Report", Technical Report AFRL-VA-WP-TR-2006-3169, July 2006.

176. Gregory S. Tallant, Prasanta Bose, James M. Buffington, Vincent W. Crum, Richard A. Hull, Tim Johnson, Bruce Krogh, and Ravi Prasanth, "Validation & Verification of Intelligent and Adaptive Control Systems", IEEE Aerospace Conference, paper #1487, Final Version, March 2004.

177. Brian Taylor, Marjorie Darrah, Christina Moats, "Verification and Validation of Neural Networks: a Sampling of Research in Progress", Intelligent Computing: Theory and Applications, Proceedings of 17th Annual International Symposium on Aerospace/Defense Sensing, Simulation, and Controls (AeroSense 2003), Orlando, FL, April 21-25, 2003.

178. Brian Taylor, Marjorie Darrah, Laura Pullum, Rebecca Giorcelli, Kareem Ammar, and Stacy Nelson, "Guidance for the Independent Verification and Validation of Neural Networks", Technical Report, Institute for Scientific Research, 2005.

179. Brian Taylor (Ed.), "Methods and Procedures for the Verification and Validation of Artificial Neural Networks", Springer, 2006.

180. UK Ministry of Defence, "Safety Management Requirements for Defence Systems", Defence Standard 00-56, Issue 4, 01 June 2007.

181. US Department of Defense, "Standard Practice for System Safety", MIL-STD-882D, 10 February 2000.

182. Vance VanDoren, "Model Free Adaptive Control", Control Engineering: Europe, 02 February 2001.

183. Robert A. Vivona, David A. Karr, and David A. Roscoe, "Pattern-Based Genetic Algorithm for Airborne Conflict Resolution", AIAA Guidance, Navigation and Control Conference, AIAA-2006-6060, August 2006.

184. Jiang Wang, Vijay V. Patel, Chengyu Cao, Naira Hovakimyan, and Eugene Lavretsky, "Verifiable L1 Adaptive Controller for Aerial Refueling", AIAA Guidance, Navigation and Control Conference and Exhibit, Hilton Head, South Carolina, 20-23 August 2007.

185. Li-Xin Wang and Jerry M. Mendel, "Fuzzy Adaptive Filters, with Application to Nonlinear Channel Equalization", IEEE Transactions on Fuzzy Systems, Vol. 1, No. 3, August 1993.

DRAFT

186. David G. Ward, Jason R. Hull, Bob Zeliff, Radek Zakrzewski, "Verification and Validation of Neural Networks for Flight Critical Applications", unpublished Powerpoint presentation given to NASA Langley Research Center, October 3, 2002.

187. David West, "Neural Network Credit Scoring Models", Computers and Operations Research, Vol. 27, No. 11-12, pp. 1131-1152, October 2000.

188. B. C. Williams, M. D. Ingham, S. H. Chung, and P. H. Elliott, "Model-Based Programming of Intelligent Embedded Systems and Robotic Space Explorers", Proceedings of the IEEE, Vol. 91, Issue: 10, pp: 212-237, January 2003.

189. Bruce Wilson, Bill Gough, and John Kay, "Adaptive Control of Sulfur Recovery Units", undated white paper, available at http://www.andritzautomation.com/documents/ adaptive_control_sulphur_recovery_units.pdf, date last visited 27 November 2011.

190. Kevin A. Wise and Eugene Lavretsky, "Robust Adaptive Control for the Joint Direct Attack Munition (JDAM)", The Impact of Control Technology, T. Samad and A.M. Annaswamy (eds.), 2011.

191. Kevin A. Wise, Eugene Lavretsky, Naira Hovakimyan, Chengyu Cao, and Jiang Wang, "Verifiable Adaptive Flight Control: UCAV and Aerial Refueling", AIAA Guidance, Navigation and Control Conference and Exhibit, Honolulu, Hawaii, AIAA 2008-6658, 18-21 August 2008.

192. Kevin A. Wise, "Adaptive Flight Control of a Sensor Guided MK-82 JDAM", SAE Meeting, Williamsburg, VA, 12 October 2006.

193. Bo K. Wong, Thomas A. Bodnovich, and Yakup Selvid, "Neural Network Applications in Business: A review and analysis of the literature (1988–1995)", Decision Support Systems, Vol. 19, Issue 4, pp. 301-320, April 1997.

194. Paul R. Work, "Special Session on Certification of Dynamic and Adaptive Systems", IEEE International Parallel and Distributed Processing Symposium, pp.164, 2007.

195. Peggy Wright, "Software Assurance for Adaptive Neural Networks In Aerospace Applications" Powerpoint presentation given at the 2005 FAA National Software and Complex Electronic Hardware Standardization Conference, Norfolk, Virginia, July 26-28, 2005.

196. Enric Xargay and Naira Hovakimyan, "Flight Verification and Validation of an L1 All-Adaptive Flight Control System on the NASA AirSTAR Flight Test Vehicle", Safe & Secure Systems & Software Symposium, June 15, 2010.

197. Enric Xargay, Naira Hovakimyant, Vladimir Dobrokhodov, Isaac Kaminer, Irene M. Gregory, and Chengyu Cao, "L1 Adaptive Flight Control System: Flight Evaluation and Technology Transition", AIAA Infotech@Aerospace Conference, Atlanta, GA, 20-22 April 2010.

198. Sampath Yerramalla, Edgar Fuller, and Bojan Cukic, "A Validation Approach for Neural Network-based Online Adaptive Systems", Software--Practice and Experience 2006; Vol. 36, Issue 11-12, pp. 1209-1225, August 2006.

199. Sampath Yerramalla, Bojan Cukic, and Edgar Fuller, "Lyapunov Stability Analysis of the Quantization Error for DCS Neural Networks", Proceedings of the 2003International

DRAFT

Joint Conference on Neural Networks, Portland, Oregon, July 2003.

200. Sampath Yerramalla, Edgar Fuller, Martin Mladenovski, and Bojan Cukic, and, "Lyapunov Analysis of Neural Network Stability in an Adaptive Flight Control System", Proceedings of the 6th International Conference on Self-stabilizing Systems, 2003.

201. Radoslaw R. Zakrzewski, "Verification of a Trained Neural Network Accuracy", In Proceedings of the 2001 International Joint Conference on Neural Networks, pp. 1657-1662, 2001.

202. Radoslaw R. Zakrzewski, "Verification of Performance of a Neural Network Estimator", Proceedings of the 2002 International Joint Conference on Neural Networks, Honolulu, HI, pp. 2632-2637, May 12-17, 2002.

203. Radoslaw R. Zakrzewski, "Fuel Volume Measurement in Aircraft Using Neural Networks", Proceedings of the International Joint Conference on Neural Networks, Washington DC, July 15-19, 2001.

204. Ji Zhang, Heather J. Goldsby, and Betty H.C. Cheng, "Modular verification of dynamically adaptive systems" Proceedings of the 8th ACM international conference on Aspect-oriented Software Development, Charlottesville, VA, pp. 161-172, 2009.

205. Y. Zhang, I.-L. Yen, F.B. Bastani, A.T. Tai, and S. Chau, "Optimal Adaptive System Health Monitoring and Diagnosis For Resource Constrained Cyber-Physical Systems", 20th International Symposium on Software Reliability Engineering, Mysuru, Karnataka, India, 16-19 November 2009.

206. Yi Zhou and Meng Joo Er, "Reinforcement Learning in Generating Fuzzy Systems", Theory and Novel Applications of Machine Learning, pp. 376, February 2009.

DRAFT

DRAFT

APPENDIX C-- DO-178B/C OBJECTIVES APPLIED TO ADAPTIVE SYSTEMS

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).	**<u>Activities</u>** • 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
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		

DRAFT

				<u>Techniques</u>
				• Reuse system level analytical models as software requirements (i.e., take software credit for system process outputs)
				• 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**	• **Ensure correct, consistent high-level requirements** • **Ensure full implementation**		

DRAFT

	Process	**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	• 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	• 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:

DRAFT

	requirements • Source Code complies with low-level requirements • Source Code complies with software architecture • Parameter Data Item File is correct and complete.	• Ensure the Source Code is accurate and complete with respect to LLRs • 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)		▪ 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
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	• HLRs are accurate, unambiguous, sufficiently detailed, consistent • LLRs are accurate, unambiguous, sufficiently detailed, consistent • Ensure correct relationship	• 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).	**<u>Activities</u>** • 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

DRAFT

	consistent. • Source Code is accurate and consistent.	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 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.		**Techniques** • Apply software-level formal method techniques (e.g., model checking, compositional verification, static analysis, program synthesis, runtime analysis).
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	• 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).

DRAFT

	compatible with target computer.			
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 verifiable. • Source Code is verifiable.	• Ensure HLRs can be verified • Ensure LLRs can be verified • Ensure architecture can be verified • Ensure architecture is deterministic and predicable • Ensure source code 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, 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	• High-level requirements conform to standards.	• Ensure HLRs are consistent with HLR standards	• Might not be more difficult for AS	

DRAFT

A-3.5, A-4.5, A-4.12, A-5.4	• Low-level requirements conform to standards. • Software architecture conforms to standards. • Source Code conforms to standards.	• Ensure LLRs are consistent with design standards • Ensure architecture is consistent with design standards • Ensure Source Code is consistent with coding standards		
Traceability A-3.6, A-4.6, A-5.5	• High-level requirements are traceable to system requirements.	<u>System requirements trace to HLR:</u> • Ensure HLRs fulfill system requirements • Ensure all the system requirements (including safety requirements) allocated to software are incorporated in the HLRs. <u>HLR trace to system</u>	• Difficulty verifying that trace demonstrates complete requirements decomposition and implementation all intended functionality (i.e. complete behavior of AS) • Difficulty verifying that the trace demonstrates absence of unintended functionality	**<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

DRAFT

	• Low-level requirements are traceable to high-level requirements.	requirements: • Identification of functionality not explicitly required by system requirements • Ensure that derived HLRs are captured, justified and fed back to safety process HLR trace to LLR: • Ensure LLRs fulfill HLRs LLR trace to HLR: • Identification of functionality not explicitly required by HLRs • Ensure that derived LLRs are captured, justified and fed back to safety process LLR trace to Source Code: • Ensure Source Code fulfill LLRs Source code trace to LLR: • Expose any source code functionality (intended or unintended) that is unsupported by the LLRs		**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 requirements and implementation ▪ System level assumptions & constraints allocated to software are properly decomposed and implemented

DRAFT

	• Source Code is traceable to low-level requirements.	• 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.	**<u>Activities</u>** • 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 **<u>Techniques</u>** • Apply software-level formal method techniques (e.g., model checking, compositional verification, static analysis, program synthesis, runtime analysis).

DRAFT

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	• Ensure EOC satisfies HLRs 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	**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

DRAFT

	requirements.	• Ensure EOC satisfies LLRs for normal range inputs	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	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 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
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 th can continue to operate cor despite abnormal inputs and conditions • Ensure failure detection and recovery capabilities are effe and robust in mitigating haz		

DRAFT

				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-r errors or compiler target-sp 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 t EOC** • **Ensure completeness of HLR LLR testing requirements ba test (RBT) coverage** • **Ensure completeness of HLR LLRs (Structural coverage)**		

DRAFT

		• **Ensure unintended function exposed (Structural coverag**		
A-7.1	Test procedures are correct.	• Ensure test cases were accu developed into test procedu 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 corre that discrepancies between and expected results are exp	• 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.	• Ensure completeness of HLR cases • Ensure completeness of LLR cases • Ensure completeness of	• 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 learned	**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.

DRAFT

	Verification of Parameter Data Item File is achieved.	verification with respect to Parameter Data Item File (e. database) elements	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 HLR LLRs • Ensure unintended function exposed • Ensure unreachable code is exposed • Ensure that the compiler do	• 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
A-7.6	Test coverage of software structure (decision coverage)			

DRAFT

	is achieved.	inject functionality that was specified in the source code		• 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-7.7	Test coverage of software structure (statement coverage) is achieved.	• Ensure requirements are sufficiently detailed (similar decision structure as the coc		
A-7.8	Test coverage of software structure (data coupling and control coupling) is achieved.	• Ensure test coverage with re to the software architecture (specifically the data flow be software components and th control of software compon execution) • Ensure a sufficient amount c hardware/software integrat testing and/or software inte testing to verify that the sof architecture is correctly implemented with respect t requirements		
A-7.9	Verification of additional code that cannot be traced to Source Code is achieved.	• Ensure that the EOC is evalu for any functionality added b compiler • Ensure that compiler added functionality has no safety ir		
12.2	Tool Qualification	• Ensure tool provides confide least equivalent to that of th process(es) eliminated, redu	• Difficulty with tool qualification of development tools (TQL-1 through TQL-4) for the auto	

DRAFT

		automated	generation of 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.	

DRAFT