

IVVNN-INT-F002-UNCLASS-021403



**INTRODUCTION
TO THE
DEVELOPMENT OF METHODOLOGIES FOR
INDEPENDENT VERIFICATION AND VALIDATION
OF NEURAL NETWORKS**

IVVNN-INT-F002-UNCLASS-021403

Research Grant NAG5-12069

February 14, 2003

**Prepared for:
National Aeronautics and Space Administration
Goddard Space Flight Center
Greenbelt, MD 20771**

**Attention:
Mr. Harold D. Coleman, Grants Officer
Mr. Nelson H. Keeler, Director, NASA IV&V**

**Prepared by:
Institute for Scientific Research, Inc.
320 Adams Street
P.O. Box 2720
Fairmont, WV 26555-2720
<http://www.isr.us>**

This Report includes data that shall not be disclosed outside the Government and shall not be duplicated, used, or disclosed in whole or in part for any purpose other than to evaluate this Report. This restriction does not limit the right of the Government to use information contained in this Report if it is proprietary data contained herein, if obtained from another source without restriction. The data subject to this restriction are contained in all sheets of this Report. The proprietary data contained herein, if disclosed to the public, would affect ISR's competitive position in obtaining business; therefore, it is considered to be exempt from public release under the Freedom of Information Act (5 USC §552, as amended), paragraph (b)(4).

IVVNN-INT-F001-UNCLASS-021403

Approval Page

Approved By:	_____	_____
	Ken McGill NASA Contracting Officer Technical Representative	Date
Approved By:	<u>Christina Moats</u>	<u>3/9/2003</u>
	Christina Moats NASA IV&V Project Manager	Date
Approved By:	<u>Spiro Skias</u>	<u>2-14-03</u>
	Spiro Skias ISR Project Manager	Date
Prepared By:	<u>Brian J. Taylor</u>	<u>2-14-3</u>
	Brian J. Taylor ISR Principal Investigator	Date
Prepared By:	<u>Marjorie Darrah</u>	<u>2-14-2003</u>
	Marjorie Darrah ISR Principal Investigator	Date
Recorded By:	<u>Karen Tucker</u>	<u>2-14-03</u>
	Karen Tucker ISR Document Control	Date

Use or disclosure of data contained on this sheet is subject to the restriction on the title page of this Report.

Use or disclosure of data contained on this sheet is subject to the restriction on the title page of this Report.

EXECUTIVE SUMMARY

The use of artificial neural networks (ANNs) within the NASA applications is expected to increase over the next few decades. Currently, there are over 20 NASA funded activities that use neural network (NN) technology. High criticality software applications of NNs will require a rigorous verification and validation (V&V) process. No overall standard exists that addresses a comprehensive V&V process specifically for NNs. Although techniques do exist that apply to the V&V of NNs, many are still underdeveloped or not sufficiently tested.

To prepare for this future need, the NASA Independent Verification & Validation (IV&V) Facility has funded an initiative for the independent verification and validation of neural networks (IVVNN), whose goal is to develop a new software assurance methodology specifically for NNs. This document, prepared by the Institute for Scientific Research, Inc. (ISR) for the NASA Goddard Space Flight Center (GSFC), represents a preliminary step toward the development of this methodology.

Pilot certification based on human factors analysis (PC-HFA) was examined to consider what aspects, abstract concepts or specific processes might be applicable or adaptable to the IVVNN. The findings presented here are based on the examination of the major issues, design factors, and operational factors that underpin the pilot certification process. These factors provide a scientific foundation; their analysis may lead to insights and approaches that would apply to the task of IVVNN.

The final methodology will incorporate state-of-the-art practices from top researchers in the field of V&V of NNs along with the experiences and knowledge from the ISR based upon its work on the Intelligent Flight Control Systems (IFCS) project. The IFCS project is a collaborative effort among the NASA Dryden Flight Research Center (DFRC), the NASA Ames Research Center (ARC), Boeing Phantom Works, West Virginia University (WVU) and the ISR. It will address each life cycle process and will be designed so that IV&V personnel and contractors may apply this methodology at various stages of a project.

Rather than developing a new IV&V standard that would have little validity and applicability to software in general, this methodology will be written with the *IEEE Standard for Software Verification and Validation, IEEE Std. 1012-1998* (IEEE 1012) as a base document with the intent of incorporating it as a supplemental procedure. PC-HFA and current and developing neural network V&V methods will be examined to determine which activities may be mapped to the IEEE 1012.

TABLE OF CONTENTS

1.0 INTRODUCTION.....1

2.0 OVERVIEW OF METHODOLOGY.....3

2.1 The Problem 3

2.2 Project Goals 4

2.3 Accepted Standards 5

 2.3.1 The IEEE 1012..... 5

 2.3.2 Modifications of the Standards 5

3.0 HUMAN FACTORS ANALYSIS AND PILOT CERTIFICATION.....6

3.1 Introduction..... 6

 3.1.1 Justification 6

 3.1.2 Systems Definition..... 6

3.2 The Human-in-the-Loop 8

 3.2.1 Pilot Certification..... 8

 3.2.2 Human Factors Analysis..... 9

 3.2.3 Accident Analysis 10

 3.2.4 Human Error Theory..... 10

3.3 HFACS Defined..... 11

 3.3.1 Tier 1: Unsafe Acts 12

 3.3.2 Tier 2: Preconditions for Adverse Events..... 13

 3.3.3 Tier 3: Unsafe Supervision 14

 3.3.4 Tier 4: Organizational Influences 14

3.4 HFACS Tier 1 Analyzed..... 14

 3.4.1 NN Violations 15

 3.4.1.1 NN Routine Violations 15

 3.4.1.2 NN Exceptional Violations 16

 3.4.2 NN Errors..... 17

 3.4.2.1 NN Skill-Based Errors 17

 3.4.2.2 NN Perceptual Errors 19

 3.4.2.3 NN Decision Errors..... 20

4.0 INTEGRATION INTO ACTIVITIES AND PROCESSES OF V&V 22

4.1 Process: Development..... 22

 4.1.1 Formal Methods 22

 4.1.1.1 The IEEE 1012 and Formal Methods 23

 4.1.1.2 PC-HFA and Formal Methods 24

 4.1.1.3 Formal Methods Tool Example 24

 4.1.2 Visualization 26

 4.1.2.1 The IEEE 1012 and Visualization..... 26

4.1.2.2 PC-HFA and Visualization 28

4.1.2.3 Visualization Tool Examples 28

4.1.3 Testing..... 30

4.1.3.1 The IEEE 1012 and Testing 31

4.1.3.2 PC-HFA and Testing..... 31

4.1.3.3 Testing Examples..... 31

4.2 Process: Operation..... 33

4.2.1 The IEEE 1012 and Run-Time Monitoring 35

4.2.2 PC-HFA and Run-Time Monitoring..... 35

4.2.3 Run-Time Monitoring Examples 35

5.0 CONCLUSION 36

6.0 CITATIONS 38

APPENDIX A - ACRONYMS 40

LIST OF FIGURES

FIGURE 4-1. A 2D REPRESENTATIVE OF THE RESPONSE AREA OF AN LRU 25
FIGURE 4-2. A 3D RIDGE REPRESENTATIVE OF THE RESPONSE AREA OF AN LRU 25
FIGURE 4-3. A THREE-LAYER NN CONVERTED INTO SIMULINK BLOCKS. 29
FIGURE 4-4. SIMULINK MODEL THAT INCLUDES THE NN 29
FIGURE 4-5. WVU F-15 SIMULATION 30
FIGURE 4-6. ORDER OF TESTING ACTIVITIES 30

1.0 INTRODUCTION

The use of artificial neural networks (ANNs), or more simply neural networks (NNs), within the NASA applications is expected to increase over the next few decades. Currently, there are over 20 NASA-funded activities that use NN technology. Most of these projects utilize NNs as a classifier to help analyze data. For example, the NASA Jet Propulsion Laboratory (JPL) is doing multivariate statistical and NN data analysis for biosignature detection in extraterrestrial samples¹, while the NASA Ames Research Center (ARC) is using NNs to understand the history of El Nino through the analysis of patterns in tree rings². The NASA JPL has also partnered with Ford Motor Company to utilize NN technology to diagnose misfiring under the hoods of Ford automobiles³.

Most NNs are being applied in low criticality software applications where a human can determine the validity of the network's results. However, at least four NASA centers are involved in the development of highly safety-critical applications that make use of NNs. The NASA Dryden Flight Research Center (DFRC) and the NASA ARC are developing intelligent flight control systems that contain NNs. The NASA JPL is investigating the use of NNs for environmental monitoring and control of the International Space Station and future research stations in Earth orbit⁴. The NASA Glenn Research Center (GRC) is developing an NN-based sensor fault detection, isolation, and accommodation (SFDIA) technology. SFDIA can be used in various systems that require continuous health-condition monitoring to achieve high productivity and avoid unnecessary shutdown⁵. The NASA GRC is also investigating NN genetic algorithm techniques for aircraft engine performance diagnostics⁶. The aforementioned applications will require a rigorous verification and validation (V&V) process for the NNs, which may be the determining factor in system-critical decisions.

As the facility responsible for improving software safety, reliability, and quality of programs and missions, the NASA Independent Verification & Validation (IV&V) Facility will be increasingly challenged to certify and evaluate software systems that contain NN technologies. Currently, the NASA IV&V Facility has not been commissioned to work on a project involving NN software; however, the independent verification and validation of neural networks (IVVNN) software will be of vital importance in the future as applications for NNs become more feasible and prevalent. Some of the most promising applications of this artificially intelligent technology are in safety-critical situations where NNs can process data and react much faster than a human.

This document, prepared by the Institute for Scientific Research, Inc. (ISR) for the NASA Goddard Space Flight Center (GSFC), is a preliminary outline of a methodology that can be used by the NASA IV&V practitioners as part of an overall process to verify and validate NN software.

1 <http://www.astrobiology.com/asc2000/abstract.html?ascid=190>

2 <http://earthobservatory.nasa.gov/newsroom/nasanews/2002/2002120910972.html>

3 <http://www.qadas.com/qadas/nasa/nasa-hm-1345.html>

4 <http://mishkin.jpl.nasa.gov/environmental.html>

5 <http://technology.grc.nasa.gov/tech/tops/ic/top3-00003.pdf>

6 <http://gltrs.grc.nasa.gov/citations/all/tm-2001-211088.html>

This methodology will incorporate state-of-the-art practices from top researchers in the field of V&V of NNs along with the experiences and knowledge from the ISR based upon its work on the Intelligent Flight Control Systems (IFCS) project. The IFCS project is a collaborative effort among the NASA DFRC, the NASA ARC, Boeing Phantom Works, West Virginia University (WVU) and the ISR. Refinement of the methodology will come from application of the IVVNN methodology on the IFCS project.

Several documents have been written for NASA that have assisted the ISR's efforts toward the goal of a comprehensive methodology for the IVVNN. The NASA DFRC and the NASA ARC produced one such document entitled "Verification and Validation of Neural Networks for Aerospace Systems," which focused on the V&V of pre-trained neural networks (PTNN) and does not provide a comprehensive discussion for V&V of on-line learning neural networks (OLNN) [Mackall 2002].

Another document that provided direction is the "Software Verification and Validation Plan (SVVP) for the Airborne Research Test System (ARTS) II Intelligent Flight Control Program," which was prepared by the ISR for the NASA DFRC and outlined some of the V&V processes that were performed to support the IFCS project [ISR 2000].

Researchers at the NASA ARC and WVU prepared a draft document for the IFCS project entitled "Validation and Verification Process Guide for Software and Neural Nets." Although never completed, this document was meant to identify required V&V activities and recommend techniques for their accomplishment. The SVVP for the IFCS project considered some of the early versions of this NASA ARC document.

The "V&V of Advanced Systems at NASA," prepared at the NASA ARC for Northrop Grumman Corporation, has provided insight into problems concerning the IVVNN, especially in the area of formal methods [Nelson 2002].

Other industry V&V efforts exist that may lend background information for the IVVNN research. "Verification, Validation and Evaluation of Expert Systems: A Federal Highway Administration (FHWA) Handbook" describes one such effort. Although this document does not specifically address NN software, some generalizations may transfer to this research.

The aforementioned documents, along with many articles, technical reports, and presentations, will be distilled into a methodology that can be applied by an IV&V practitioner when faced with the task of verifying and validating a system containing NNs.

This document provides an overview that defines the problem, states the project goals, examines the constraints of the problem, and discusses current V&V techniques and IV&V practices. Rather than create an entirely new standard, the preliminary document will discuss how the proposed methodology will complement or augment current practices. As a highly regarded industry standard, the *IEEE Standard for Software Verification and Validation, IEEE Std. 1012-1998* (IEEE 1012) will provide a framework upon which to build.

There will be a discussion of pilot certification based on human factors analysis (PC-HFA) beginning with an introduction to complex systems and then a discussion of human-in-the-loop design. This will be followed by an examination of accident analysis and human error theory to gain insight into the development of a comprehensive methodology for the IVVNN and what parallels can be drawn from the rigorous certification of a human pilot. The underlying motivation for pilot certification activities will be examined to determine how the

assessment of a pilot takes into account the adaptive and non-deterministic nature of the human that is being certified.

The final section of this document will look at the integration of NN-specific methods into current IV&V practices based on the observations from the HFA using the IEEE 1012 as a backdrop. This will involve augmentation or adaptation of the V&V activities in the concept, requirements, design, implementation, test, installation, and operation processes. There will be a discussion of how formal methods, visualization, advanced testing, and run-time or operational monitoring can be used to enhance current IV&V practices when evaluating NN software.

2.0 OVERVIEW OF METHODOLOGY

Software quality is often defined differently from project to project. For simpler systems, such as software used in calculators and watches, software quality may be defined to include usability, efficiency, and functionality. For complex systems, such as aircraft control technologies and medical diagnostic equipment, the software quality definition is expanded to include dependability, reliability, accuracy and, perhaps, understandability.

While developers should strive for software quality on all projects, regardless of size or importance, it becomes paramount for safety-critical systems in which human life can be endangered or extensive financial losses can occur. Software quality is not something that can be appended to a software project; it must be introduced at the initial phase and carried through faithfully until the system is complete. Traditional programming techniques have adequate provisions for software quality and several standards and guides have been written to provide software organizations the means by which to ensure quality. The same cannot be said of non-traditional techniques, like NNs, which create a problem for software engineers seeking to ensure quality in such systems.

2.1 The Problem

The design and implementation of NNs differs from that of traditional software. Instead of being programmed, a learning algorithm teaches an NN a mathematical function by making use of statistical and signal processing techniques. This development of the code is inspired by biological NNs, such as the human mind, that learn, adapt, and re-program based upon new inputs introduced over time.

An NN's greatest strength, adaptability, also creates its greatest challenge: how to ensure that its judgment and decisions are sound. This software must be scrutinized to ensure it will perform as expected in every situation. NNs may encounter unforeseen situations in the field resulting in unpredictable responses. Addressing this challenge will require new methods or extensions of existing methods.

The NN's response over time may not be predictable; the reasons the NN arrived at its knowledge may not be explained easily. Because of the non-deterministic results of an NN's adaptation, it is often considered a black box. One of the few methods used in the V&V of an NN is rigorous testing with data that is similar to that used during the network's training.

In more complex, safety- and mission-critical systems, the standard NN training-testing approach is insufficient to provide a reliable method for certification. Verifying correct operation of NNs within NASA projects, such as autonomous mission control agents and

adaptive flight controllers, or within nuclear engineering applications, such as safety assessors and reactor controllers, requires a more rigorous approach.

This V&V challenge is further compounded by adaptive NN systems that modify themselves, or learn, during operation. These systems continue to evolve after deployment, for better or for worse. Traditional software assurance methods fail to account for systems that change while in use.

2.2 Project Goals

Having surveyed the literature concerning current methods and tools available for V&V of NNs, the ISR has identified and will address two key issues:

- No overall standard exists that addresses IV&V techniques specifically for NNs.
- Current IV&V techniques for NNs are still immature and are not sufficiently developed.

As the second stage of research, the ISR will research and identify the components that will comprise a unique methodology effective for the IVVNN. This preliminary document represents an effort to address the above two concerns in the form of a standard. It will serve as an initial assessment that will look at the methodology from a top-level viewpoint and supply a description of the possible layout of the final methodology for the IVVNN.

The methodology will consider two types of NNs:

- *Fixed, Non-Adaptive Neural Networks*: Sometimes referred to as a PTNN, an NN that has undergone training and then becomes set. The internal structure of the network remains unchanged during operation. After training is complete, all weights, connections, and node configurations remain the same, and the network reduces to a repeatable function.
- *Dynamic, Adaptive Neural Networks*: Sometimes referred to as an OLNN, an NN that is never fixed, so the system continues to develop throughout its life. An OLNN is continuously adapting to current data, changing its internal structure of neurons and weights. OLNNs are employed in situations where a system learns while in use.

Background research for this methodology will include three areas of research. First, research will be performed in quantifying human factors used in certifying pilots that may bring insight into the development of a methodology for IV&V of these adaptive and non-deterministic systems. Second, existing V&V techniques, such as formal methods and automated testing, will be studied in detail and possibly extended to be included in a unique methodology effective for the IVVNN. Finally, current IV&V practices will be examined so that the new methodology does not replace existing practices but complements them.

The final methodology will address each life cycle process and will be designed so that IV&V personnel and contractors may implement it during any phase of the project. An emphasis will be placed on the requirements and testing processes over the others due to their accessibility to IV&V practitioners when engaging in new projects. As indicated, this methodology will focus on NNs and is not intended to account for all non-deterministic systems. However, it is expected that, once developed, some of the approaches may apply to a broader domain of software.

This methodology will offer supplemental activities, including NN specific tools, towards the goal of enhancing existing standards that the IV&V practitioner can apply.

2.3 Accepted Standards

Part of the NASA IV&V Facility's mission is to "Increase software safety and quality, reduce software costs, and improve delivery time through the early detection and resolution of errors, by utilizing and applying empirically based software engineering best practices."⁷ In the case of traditional software and programming techniques, the NASA IV&V Facility relies upon a well-accepted industry standard to complete its mission, the IEEE 1012. However, this standard fails to address the uniqueness and special circumstances related to NNs.

As noted above, certain sections of the IEEE 1012 may be more applicable when considering IV&V participation in other NASA software projects. Since the NASA IV&V Facility and family of subcontractors may not participate at the start of a software project, they will have greater involvement within the requirements and testing processes.

2.3.1 The IEEE 1012

Current NASA IV&V practices are modeled after the IEEE 1012, which was developed with the intent to address all software life cycle processes including acquisition, supply, development, operation, and maintenance [IEEE 1998].

The IEEE 1012 was created to provide a list of activities for software engineering that, when applied to a software system, would help developers add quality to the project. The standard offers a classification of software integrity levels that a software engineer can use to decide the level of V&V required for the software throughout each life cycle process. In this manner, a minimum set of V&V activities can be defined, allowing a V&V practitioner to adjust tasks as needed to meet certification criteria for each level.

As an example of the tasks for a software stage, consider the IEEE 1012 suggested activities for the requirements process. Suggested activities include performing a traceability analysis, software requirements evaluation, interface analysis, criticality analysis, system V&V test plan generation and verification, acceptance V&V test plan generation and verification, configuration management assessment, hazard analysis, and risk analysis. The standard also provides a brief description of expected results from an activity and identifies required inputs and outputs to successfully complete the activity.

As a standard for traditional software, the IEEE 1012 is accepted as a guideline and is widely used through the government, academia, and commercial industries. It was written in a manner that maps the activities and tasks of the IEEE 1012 to another software industry standard, ISO/IEC 12207, which heavily influenced the creation of ISO 9001-2000, the top-level standard used within the NASA IV&V Facility.

2.3.2 Modifications of the Standards

Rather than developing a totally new standard that would have little validity and applicability to software in general, this IVVNN methodology will be written using the IEEE 1012 as a base document. All additions and suggestions to account for NN software will be developed

⁷ http://www.ivv.nasa.gov/mis_vis/index.shtml

to supplement the IEEE 1012. The study of PC-HFA, as well as the research on current and developing neural network V&V methods, will be used to determine which supplemental activities to attach to the IEEE 1012.

Since there appears to be a growing trend towards the use of NNs within intelligent flight control systems, special attention will be given to HFA with regard to pilot certification; however, the choice to study pilot certification will not limit this methodology to flight control systems. If we consider pilots to be the equivalent of a biological NN, then the study of the training, testing, analysis, and subsequent certification of these NNs may translate to ANNs. This methodology will consider this approach from a high-level abstract viewpoint so that the end results will be more general.

3.0 HUMAN FACTORS ANALYSIS AND PILOT CERTIFICATION

This section considers what aspects of the pilot certification process might be directly applicable, or at least adapted and extrapolated, to the IVVNN based systems, specifically to their use in air-flight controllers.

3.1 Introduction

The findings discussed in this section are based upon an examination of major principles and major contributing design and operational factors that provide the scientific foundations that underpin the pilot certification process. This examination is intended to lead to new insights and approaches that will prove applicable to the task of IVVNN systems.

3.1.1 Justification

Humans and NNs have much in common, both structurally and architecturally. The historical development and success of ANN systems has depended heavily upon efforts to model and mimic the neural systems of living things. Thus, the certification of humans may well provide insight into how NN systems may be verified and validated.

In this study, the motivation for considering pilot certification is also influenced by another line of reasoning and analysis. In particular, the pilot and the flight controller are focused on successfully controlling the operation and flight of aircraft. In addition to sharing a common architectural basis, they also share another defining aspect, operating in the same problem domain.

The following issues should be considered:

- Nature and defining characteristics of this problem domain
- Necessity of the a human-in-the-loop to an aircraft's successful operation
- Problems incurred by having a human-in-the-loop

3.1.2 Systems Definition

Some of the defining characteristics of complex systems may include such attributes as (1) adaptive, (2) autonomous, and (3) non-deterministic. The formal definition and significance of each of these attributes follows.

- *Adaptive* refers to the capacity or suitability for, or the tendency toward change, modification, etc.⁸ In the process of its functioning and executing, the system may adapt not only by modifying current behaviors, but possibly even by adopting new behaviors.
- *Autonomous* refers to being free from external control and constraint in action and judgment, independent in mind or judgment, self-directed.⁸ While the system may be initialized externally with specific goals, assignments, and constraints, the system still retains considerable freedom in how it will respond to events and accomplish its assignments, even possibly to adapting itself in some permanent sense.
- *Non-deterministic* refers to an algorithm or process having the property that a computation or execution may have or may produce multiple plausible results.⁹ The system engineer cannot, with absolute certainty, specify what the outcome will be. The space of possible outcomes may include not only acceptable outcomes, but also those that could be judged as undesirable.

Complex systems possessing the above attributes (adaptive, autonomous, and non-deterministic) are developed because of the many potential benefits and functionalities that these attributes can enable. The success of many useful systems depends upon the presence of these attributes.

Such desirable attributes also represent a system engineer's conundrum. These attributes bring with them a risk of uncertainty and, ultimately, failure. The set of all possible system states and outcomes may well include some that are not acceptable, if not decidedly dangerous. The conundrum thus becomes one of capitalizing on all the many desirable features and capabilities while eliminating, or at least minimizing, the undesirable possibilities and failures.

In the case of complex systems, in which unconditional performance and success cannot be absolutely guaranteed, the system engineer must design for failure. The pursuit of such design paradigms leads to the following considerations:

- *Fault-tolerance* refers to building to detect and recover from failure.⁹
- *Graceful degradation* refers to when performing less than optimally is better, or preferable, to the complete total cessation of that function.¹⁰

The introduction of the human-in-the-loop certainly offers one means to imbue a system with the aforementioned attributes, which all humans innately possess in varying degrees. The presence of the human-in-the-loop hopefully can ensure that only desirable outcomes are realized. The human is expected to provide, or at least contribute to, such failure-recovery mechanisms as fault-tolerance and graceful degradation. This achievement would represent the best of all worlds.

8 <http://www.Dictionary.com>

9 On-line Computing Dictionary

10 Telecom Glossary 2000

3.2 The Human-in-the-Loop

Although the introduction of the human-in-the-loop may provide the solution to many complex system design requirements, it also introduces yet another class of problems, specifically human error. Humans, by their very nature, make mistakes. Various studies have implicated human error in a variety of occupational accidents [O'Hare 1994; Wiegmann 1999; Yacavone 1993]. In particular, 70% to 80% of those occupational accidents occurred in civil and military aviation.

The human-in-the-loop also creates yet another unique limitation in how other system components can be designed and integrated to address various systems and operational requirements. The systems engineer may have considerable opportunity and latitude in addressing shortcomings of existing or proposed system designs, often with order-of-magnitude improvements in non-human system components.

Where the human-in-the-loop is concerned, however, the system engineer must work within the fairly fixed physical, physiological, and psychological capabilities and limitations of the human being. Some researchers and developers would go so far as to say, "The MACHINE must be designed to match the characteristics of the MAN! Not the other way around [SOSU 2002]!"

Yet another layer of complexity is introduced when the system under consideration is also classified as a safety-critical system because human lives may be at risk.¹¹ The prevention of undesirable outcomes, especially those failures that could result in harm to humans, becomes the top priority even at the expense of accepting less desirable solutions and performance levels. The seriousness of this situation is reflected in the set of processes, procedures, and activities that constitute pilot certification.

This brings the discussion to the consideration of how the human-in-the-loop design problem can be addressed. Among the special tools, techniques, procedures, and sources of knowledge available to the systems engineer are: (1) the pilot certification process; (2) HFA; (3) accident analysis; and (4) human error theory. These are discussed in the following sections.

3.2.1 Pilot Certification

The purpose of the pilot certification process is to verify and validate that the incorporation of the pilot (the human-in-the-loop) meets the system design requirements, including those introduced by the presence of the human. The purpose of pilot certification, as officially stated by the Federal Aviation Administration (FAA) and the Department of Transportation, is "... to enhance the ability of pilots to meet the evolving demands of the National Airspace System and operate safely and effectively in this environment [DOT 1997]."

This description of pilot certification emphasizes that the *demands* on pilots are *evolving* and include two key components, safety and effectiveness.

The pilot certification process, like the system operations that it supports, is quite complex and could be decomposed and analyzed from various perspectives. For the purpose of this study, the pilot certification process is viewed in terms of how it addresses *performance*

¹¹ <http://www.VirtualLibrary.com>

focus (normal or standard operation of the aircraft), and *safety focus* (continued fault-tolerant, possibly degraded, operation of the aircraft under abnormal conditions).

Pilot certification focuses on ensuring that the pilot is thoroughly prepared to properly perform the known aspects of aeronautics, as well as reasonably prepared for handling the unknown. Consequently, system design concepts such as fault-tolerance and graceful degradation are fundamental necessities to pilot certification.

The principal challenge to the systems engineer is to leverage all available resources, including the pilot certification process, to preemptively eliminate, wherever possible, the consequences of human and systems failures. This challenge arises in all systems that involve a human-in-the-loop and that, consequently, must address human error.

3.2.2 Human Factors Analysis

The human-in-the-loop creates another unique limitation to how other system components can be designed to address various systems and operational requirements. The systems engineer must work within the fairly fixed physical, physiological, mental, and psychological limitations of the human being.

The minimax challenge (maximizing system performance while minimizing the possibility of failure) provides the motivation for the scientific discipline of HFA and related disciplines, such as accident analysis (the study of the circumstance and causality of accidents).

Historically, World War II provided the impetus for the emergence of human factors in aviation. During that time, a mismatch between the abilities of military personnel and the complexity of machines resulted in huge losses, both in financial and human terms.

Simply stated, HFA can be defined as the study of people in their working and living environments [SOSU 2002]. HFA is concerned with the relationships between people and machines, people and their environment, and people and other people. The objectives of HFA are to optimize the effectiveness of the system with respect to safety and efficiency, and optimize the well being of the individual. HFA is multi-disciplinary, drawing upon such disciplines as psychology, physiology, engineering, ergonomics, medicine, and sociology.

The principles that guide HFA are applied both proactively and reactively. In a proactive sense, system engineers, equipment designers, and training personnel utilize human factors knowledge (methodologies and results) to improve a system's performance and usefulness. From a reactive perspective, accident or incident investigations involve the application of this same knowledge to understand the causes of errors made by pilots, controllers, designers, and management.

The major goal of HFA is to develop appropriate preemption or prevention of human-related failures where possible, and to mitigate, or reduce the severity of a failure. In particular, HFA and its related disciplines, such as accident analysis, provide the human component of the technical foundation for the planning, designing, execution, and evaluation of any certification process [Wells 1997].

One could adopt a process-focused viewpoint of HFA, systems design, and certification as constituting a continuous feedback loop. Such a view reflects the interrelated roles that these processes share in systems development, operation, and maintenance.

3.2.3 Accident Analysis

Fortunately, comprehensive theories have been developed to characterize system accidents and failures, especially those that are human-related. Safety-critical system domains, such as aeronautics and nuclear propulsion, have provided fertile areas to study. In fact, the aerospace industry has been the largest contributor, whose results have been leveraged by other industries [Ford 1999].

The research literature has well established that accidents or mishaps are generally not attributable to a single cause, or in most instances, to a single individual or person [Weigmann 2001]. Rather, accidents usually are determined to be the result of a myriad of interrelated causes and circumstances, only the last of which culminated in the failure.

As a first attempt to understanding the cause-effect interrelationships among a mesh of failure events, the individual failure events may be classified as active or latent.

- *Active failures* are the actions or inactions of operators, system components, etc., that are believed ultimately to have caused an accident.
- *Latent failures* are other errors committed by individuals or elsewhere in the supervisory chain of command and system operation that affect the sequence of events that characterize an accident.

Although a mesh composed of active and latent failures may be adequate to describe the chronology of an accident, it does not adequately capture deeper interrelationships that are represented by the various links connecting those failure events. Previously mentioned system attributes (adaptive, autonomous, and non-deterministic), together with the innate capabilities and shortcomings (error proneness) of the human must be juxtaposed with the occurrence of an accident and its chronology mesh of active and latent failure to truly explain its cause(s) and to develop reasonable preemption or remediation approaches for that type of accident for the future.

3.2.4 Human Error Theory

In support of the juxtaposition of human nature with complex system design and operation, accident analysis practitioners and human factors analysts have developed comprehensive theoretical frameworks of human error from which to organize and explain an accident's mesh of failure events from a human perspective [Geller 2000]. One early approach to this problem is provided by Frank Bird's *Domino Theory*, which promoted the idea that, like dominos stacked in sequence, mishaps are the end result of a series of errors made throughout the chain of command [Bird 1974].

Building upon Frank Bird's efforts, James Reason developed his generalization of the Domino Theory, the *Swiss Cheese* model [Reason 1990]. His theory, originally developed for the nuclear industry, identified a taxonomy of multiple levels at which active and latent failures could occur and interact within complex operations.

The Swiss Cheese model is particularly useful because it provides a framework to address latent failures within the causal sequence of events. One naturally asks how all these holes, the active and latent failures, are related, and are they too numerous to define.

Fortunately, further analysis has revealed that many mishaps are not unique from their predecessors, but have very similar causes. Consequently, the characterization of these system failures, or holes, does provide an adequate structure for identifying their roles in mishaps, or better yet, for detecting their presence and correcting them before a mishap occurs.

Expanding upon Reason's efforts, the United States military developed a comprehensive framework, the Human Factors Analysis and Classification System (HFACS), from which to identify and analyze the holes related to aeronautical accidents [Naval Safety Center 1996]. While developed for the military flight environment, the HFACS framework has since been applied to commercial aviation, as well as in other safety critical problem domains, such as the nuclear and medical industries. One example is the Marine Facilities Division of the California State Lands Commission [Gutierrez 2002].

Other efforts similar to HFACS have since been developed. For example, EUROCONTROL (European Organization for the Safety of Air Navigation) has developed its own accident analysis framework for air traffic management (ATM). The Human Error in ATM Technique (HERA) is intended not only for the retrospective analysis of airspace incidents, but also as a prospective diagnosis tool during ATM system development [Isaac 2001].

Both of these systems, HFACS and HERA, treat the individual operator as an element in a larger safety critical system. Conceptually, both techniques analyze error events by considering the relationships between elements in the system. Both techniques examine individual errors and the situational and organizational factors surrounding the event. The differences in the HERA and HFACS frameworks are in their levels of detail and their focus of concepts.

While the newer HERA offers a more detailed taxonomy, the HFACS provides a sufficiently developed framework from which to consider how the principles and tools that constitute and support pilot certification may be adapted to the V&V of NN based systems. Another reason to consider the HFACS is one of practicality. The HFACS was developed for, and is now used by, the United States military and the FAA.

3.3 HFACS Defined

This section examines the higher-tier components and relationships that constitute the HFACS framework. The HFACS framework describes a taxonomy consisting of four first-tier levels of failure, which are: (1) Unsafe Acts; (2) Preconditions for Unsafe Acts; (3) Unsafe Supervision; and (4) Organizational Influences.

The four tiers are presented in this section in reverse chronological, as well as reverse causal order. The framework begins with treatment of immediate failures, termed unsafe acts, then reasons backward to understand the chronology and causality of those circumstances and events that led to those unsafe acts. Greater detail is given for tiers most related to the unsafe act, chronologically and causally.

As originally formulated, the HFACS framework focused on the roles of humans in the causation of accidents and failures; however, many insights gleaned from this framework are readily applicable to other system components. Researchers who share this viewpoint have contributed their extensions of the HFACS framework as they adapted the HFACS to their particular analysis requirements [Gutierrez 2002].

3.3.1 Tier 1: Unsafe Acts

Unsafe acts are operator actions or inactions that occur immediately to, and often trigger, an adverse event, previously termed an active failure. Unsafe acts are further classified into two categories, violations and errors. The differentiation between violations and errors is based on whether the action is currently considered acceptable, or legal, behavior.

- *Violations*, in contrast to errors, are willful deviations of accepted regulations, whether or not they actually result in the occurrence of a failure in the given instance. The violator is willing to accept the risk and consequences of a failure. Similarly, violations are further divided into the following sub-types:
 - *Routine* violations, whether recurring or habitual, are instances of breaking regulations that are part of a behavior pattern. Such behaviors are often recognized but condoned by management and, while known to be unsafe and could potentially result in failures, do not often result in immediate failure.
 - *Exceptional* violations are not typical of an individual and are not condoned by management. These isolated offenses may or may not involve malice, the intention to cause harm or failure.
- *Errors* are legal mental and physical activities that fail to achieve their intended outcome and occur within currently accepted governing regulations. Errors are further decomposed into the following sub-types:
 - *Skill-based* errors occur during execution of a familiar procedure that normally requires little or no conscious thought. They often result from a lapse in memory, such as forgetting a step, or a loss of focus or attention, such as distraction from another task.
 - *Perceptual* errors are misinterpretations of what is seen, heard, or otherwise received through the senses. They generally occur when sensory input is degraded or unusual.
 - *Decision* errors represent conscious, goal-intended behavior that proceeds as designed, yet proves inadequate or inappropriate for that situation. These errors tend to occur when a familiar situation is not recognized or is misdiagnosed, or when an unfamiliar situation occurs, and generally result in the application of an unsuccessful procedure.

In addition to the human-focused classifications of unsafe acts, the HFACS framework has been broadened to consider non-human aspects of the system, such as structural damage and mechanical failure, as additional sources of unsafe occurrences [Gutierrez 2002]. These refer to how damaged or otherwise malfunctioning equipment, structures, and workspaces can be precursors of an adverse event. HFACS further classifies these events depending on criteria such as where they occur and whether the damage is structural or functional.

3.3.2 Tier 2: Preconditions for Adverse Events

Most events have continuity with a history of prior and currently existing conditions, circumstances, and activities. The prior events provide the context for the event that is occurring now. The HFACS framework attempts to identify and classify aspects of that historical context, which constitute other active or latent failures that could be causal to the current event.

The presence of these failure events, at this and later tiers, may be viewed as weaknesses in the system's defenses and as lost opportunities for preventing a given incident. An understanding of these sources of failure provides a foundation for the development of preemption strategies for preventing future unsafe acts.

Preconditions for adverse events describe a context framework that includes existing and possibly previous conditions and practices of operators, as well as the currently prevailing state of the workspace, systems, and environment. The HFACS framework has divided this historical context into the following top-level categories of preconditions and their corresponding subtypes.

- *Substandard conditions of operators* are states or characteristics of operators around the time of an incident that pre-dispose the individual to error. Substandard conditions are further divided into the following:
 - *Adverse mental states* are physical and mental conditions (for example, loss of situational awareness, complacency, misplaced motivation, effects of sleep loss) that negatively affect performance.
 - *Adverse physiological states* include such conditions as physical fatigue, illness, and medication effects that are known to influence performance.
 - *Physical/mental limitations* are sensory, motor or cognitive limits that result in not seeing, not hearing, not understanding, or not acting quickly enough to safely complete an action or procedure.
- *Substandard practices of operators* are failures of individuals or groups to adequately prepare for and communicate during work activities. Substandard practices are further divided into the following:
 - *Crew resource management* issues are instances of poor crew coordination, communication, or direct supervision that result in unsafe behavior.
 - *Personal readiness* includes instances either of poor judgment in maintaining readiness for work, such as using time for adequate rest or violation of any existing work readiness rules.
- *Substandard work interfaces* are instances of design or maintenance of equipment and workspaces that are inadequate for safe activities and include problems with design and maintenance of structures.

- *Adverse environmental conditions* are factors, such as weather, that interfere with perception, communication, and actions necessary to safe operations. Environmental conditions include those that are controllable, such as neatness and cleanliness, and those that are uncontrollable, such as the weather.

3.3.3 Tier 3: Unsafe Supervision

Unsafe supervision refers to supervisory practices or decisions that are inadequate to ensure safe and secure functioning of an operation. These practices or decisions are often removed from the time or place of the incident and include the following subtypes:

- *Inadequate supervision* refers to instances when supervisors fail to provide adequate training opportunities, guidance, leadership or motivation.
- *Planned inappropriate operations* are approved operations or activities carried out in haste that often result from the pressure of production outweighing the need for protection.
- *Failure to correct a known problem* includes deficiencies among individuals, equipment, training, or procedures that are known to a supervisor but that are allowed to continue unabated.
- *Supervisory violations* are instances of willful disregard of the rules, or a failure to enforce rules and regulations.

3.3.4 Tier 4: Organizational Influences

Organizational influences are often omitted in incident inquiry programs, but can directly affect both supervisory and operator practices, as well as the physical and cultural environment of an operation. Types of organizational influences include the following:

- *Resource management* issues include the sufficiency of human, equipment, and monetary resources supplied an operation to safely and effectively operate.
- *Organizational climate* refers to instances when the work atmosphere is substandard for conducting a safe and effective operation. The willingness to report errors, clarity about acceptable and unacceptable behavior, and flexibility to respond to incidents and learn from mistakes are all examples of a good organizational climate.
- *Organizational process* refers to the adequacy of policies that guide everyday operations, such as operating procedures or incentive systems that strain safe operation.

3.4 HFACS Tier 1 Analyzed

In this section, Tier 1 of the HFACS framework is explained, applied to understanding the pilot certification process, and ultimately extrapolated to provide new insight and guidance into how the V&V of NNs could be improved. This treatment of Tier 1 establishes the approach by which the other tiers will be analyzed during the continued execution of the ISR's project with NASA IV&V.

Analysis of the HFACS framework provides a sound scientific basis from which to approach the enhancement of other non-human components of complex systems that exhibit the same

human-like behaviors (adaptive, autonomous, and non-deterministic) that have been previously discussed. In particular, this document will consider their application to the V&V of NNs. The ideas presented here are to be confirmed and adjusted as necessary as the project progresses.

3.4.1 NN Violations

Tier 1 bifurcation in the HFACS unsafe act taxonomy is based upon failure due to error versus failure due to violation. The first question to be raised concerns when and how the concept of violations might arise in the failure taxonomy in an NN. Violations are determined by rules and regulations, which are externally-imposed constraints, independent of what an NN system is capable of learning technically.

For some problems, OLNN systems may be unencumbered by external rules and regulations; therefore, there would be no NN violations. An example of this situation would be the application of OLNN technology to general data mining tasks where supposedly *a priori* illegal patterns do not exist.

In the case of the OLNN developed for the IFCS project by the ISR, external rules and regulations certainly do exist. Part of the pilot certification process involves appraising the pilot's knowledge and understanding of these rules and regulations. The pilot's certification is subject to review and possible revocation for violations.

In its consideration of how and why humans break rules, the HFACS identified two violation types: routine and exceptional. An OLNN could learn to break the rules, routinely or otherwise.

3.4.1.1 NN Routine Violations

Technically, an OLNN could be capable of learning to fly an aircraft in maneuvers that are judged NOT acceptable for reasons that are outside of the learning space of the OLNN, and therefore, are not recognizable from within. Over time, the OLNN, unconstrained by rules and regulations, will learn to fly the aircraft in otherwise unsafe ways.

An explanation of how this situation can occur is quite simple. The components of most systems typically are over-specified, over-designed, and over-engineered rather than being merely adequate to meet individual system specifications. Otherwise, systems could become quite brittle at their specification boundaries. Under normal circumstances, the actual system should be able, technically, to perform better than the specified system.

Consequently, the OLNN, which is otherwise unconstrained in its learning to improve the actual system's performance, may push the boundaries of that actual system, which indeed can outperform the specified system. At some point, the OLNN, through routine learning, could be flying the aircraft in this over-compensated regime that subsumes the pre-specified flight envelope. This would be in violation of the specified system's capabilities. To the extent that those specifications are backed by rules and regulations, the OLNN has just learned to routinely commit a violation.

This introduces an interesting dichotomy regarding the NNs embedded in the ISR developed intelligent flight control system. This system is expected to control the aircraft properly under normal conditions, both externally with respect to the flight environment, and internally with respect to the aircraft's subsystems. The intelligent flight control system is

also expected to adapt and learn to control the aircraft's flight under less than optimal circumstance, even to the point of possible failure of various aircraft components and subsystems. Thus, the OLNN is learning to fly the aircraft routinely in an otherwise unsafe mode.

3.4.1.2 NN Exceptional Violations

As important as the consideration of how a person or system might learn inappropriate behavior, is the analysis of why the behavior is learned. Analyzing Tiers 2 through 4 of the HFACS framework may shed additional light on the question of why.

Consider the situation of a human who breaks the rules for a higher purpose. In such a circumstance, the human may find himself to be operating in an abnormal situation that is not adequately addressed by the current rules; rules that were not visionary enough to have contemplated such circumstance as the current situation. Perhaps a *rule taxonomy* is required that differentiates what is conditionally or arbitrarily illegal (reflecting current technical or management limitations) versus what is absolutely illegal (reflecting violations of the laws of science).

Another significant aspect of pilot certification involves preparing the pilot for those abnormal situations, including those where following the rules may not work. The training regime may place a pilot, by means of a simulator or hypothetically on a written exam, in an unsafe situation for the specific purpose of certifying the pilot's ability to adequately respond to such circumstances.

Sometimes, two wrongs do make a right, at least in the sense that the latter somehow compensates for the former in a fault-tolerant, error-correcting, graceful degradation sense. From this perspective, the latter action is illegal unless it is the only means of correcting a prior error that could lead to worse consequences.

To the extent that some abnormal situations are more likely to occur and could be more catastrophic than others, the *a priori* preemptive analysis of, preparation for, training and certification for, etc., can result in that situation being normal. In other words, a system is exposed to abnormal situations. This is an example of risk mitigation, which attempts to reduce the likelihood of an accident in an environment where accidents are indeed a distinct possibility. Similarly, the NNs of the intelligent flight control system are expected to learn to perform under such known-to-be unsafe conditions. The risk cannot be totally eliminated, but reasonable efforts should significantly reduce the likelihood of failure.

In the case of control applications, such as the OLNN developed by the ISR, constraining rules and regulations do exist regarding what is acceptable to do and therefore, learn. The handling of the rules and regulations can be addressed in several ways, each with its own issues.

- They are represented within the OLNN, so that the OLNN is self-regulating.
- They are captured internally; consequently, the V&V process must handle the correctness of this embedded rule-regulation set.
- They are applied by monitoring the learning of the OLNN, enabling the anticipation of a violation and determining whether recovery is possible.

3.4.2 NN Errors

In addition to violations, the HFACS framework identified three general types of errors, those legal activities that fail to achieve their intended outcome: (1) skill-based, (2) perceptual, and (3) decision. The assessment of an NN involves consideration of all three, because they are interrelated. For example, aspects such as distraction contribute to all three, but with differing manifestations and consequences.

3.4.2.1 NN Skill-Based Errors

Generally, an NN is trained to perform skill-based functions that involve the execution of a familiar procedure that normally requires little or no conscious thought. The skill-based functions may be quite complex, involving a variety of sub-skills.

The human becomes proficient at skill-based functions through practice, which involves the repetition of the skill-based task until it ceases to require conscious thought to be executed correctly. As part of the pilot certification process, the pilot is exposed to sufficient practice through simulations and/or actual flying for such tasks to become skills of that pilot.

The NN, likewise, requires appropriate training and evaluation for the skill-based task it is to perform. The training set must be sufficiently encompassing, including any abnormal situations of the operation space where one expects the skill to be used.

From the human's perspective, skill-based errors generally result from a lapse in memory, such as forgetting or otherwise omitting a step, or from loss of focus or attention, such as distraction from another task or external circumstance. Similar conditions exist when the NN is performing skill-based tasks, but the manifestations and consequences are different.

Such problems, as the memory error problem and its solution, are already well understood in computer science. Solutions include the development of error-correcting memory at the hardware level, runtime software-based detection and prevention of buffer overflows, etc. Such considerations are not unique to the NN environment and will not be given particular treatment in this document.

A distracting situation is generally due to the occurrence of an unexpected or unanticipated event, like the proverbial "from out of the blue" event. *Distractions* are events that could interfere, if noticed, with performing the task at hand, but for which ignoring them poses no undesirable consequences.

The human can be conditioned through training and practice to quickly recognize an oft-occurring distraction and dismiss it. With sufficient repetition, the recognize/ignore process can become an unconscious skill. Similarly, an NN can be conditioned to ignore an event just as they are trained to recognize that event. The NN is exposed to the task at hand, along with examples of that distraction, and is trained to produce the same results as when that distraction was not present. The solution to the distraction problem for an NN seems simple and apparent, but the problem is more complicated than it first appears.

A person can be conditioned to ignore a distraction, with or without conscious consent. This may seem to solve one problem, but there is the risk of introducing another problem. Someone may *a priori* characterize a given event class as being distractive, and go so far as to be conditioned and skilled in ignoring it. An undesired consequence is the possibility that some future occurrence is ignored by habit that should not have been ignored.

In the 1980's, a flight crew was practicing the procedures for landing a C-5 super cargo military aircraft on a runway. The crew would go through all the steps leading up to landing the aircraft, except for actually putting the plane down physically on the runway. Just before making physical contact with the runway, the crew would pull up for another go-around and continue practicing¹².

Since they did not plan to actually land the aircraft, they did not activate the landing gear; a violation which generated a warning alert. That alert became quite annoying and a distraction, at least under the given circumstance. So, they disabled the alert, which was yet another violation.

The magnitude of their cumulative failure became apparent only when they finally did land the aircraft. They had not re-armed the alert and there was no procedure for re-arming an alert that was not supposed to be off in the first place. They also failed to activate the landing gear, doing just as they had practiced. The consequence of this series of errors and violations was that the crew landed the aircraft on its belly!

One solution to such unintentional conditioning involves bringing the otherwise ignored event to the conscious level for confirmation that it indeed can be ignored. A tool or mental crutch commonly used for this purpose is the *checklist*. Critical steps and milestones, in what otherwise could become a routine skill that one might perform subconsciously, are explicitly called out for conscious note.

The general approach of explicit subtask decomposition, recognition, and conscious-level checklists presupposes that the skill-based task being implemented by the NN indeed lends itself to such decomposition. It also assumes that the external checklist manager can recognize when the NN has achieved a given subtask. This last concept is supportive of black-box V&V of an NN that, due to its design, lacks such internal monitoring and reporting features or other built-in testing features to support such analysis.

The checklist serves several purposes. During training, self-feedback that the total task is being learned correctly is provided. During the normal execution of a learned, skill-based task, the checklist confirms that specific subtasks are correctly addressed. To the extent that some sets of subtasks are sequentially related, the checklist provides one means of overseeing stepping through a process. In particular, the concept of the checklist can support real-time V&V procedures.

In the event of a distraction, the likelihood of an unnoticed error entering the process would be reduced because the distraction would be for a much shorter period of time since the checklist would assist in regaining conscious focus on the task at hand. Finally, this explicit elevation of the skill-based task to the conscious level provides an opportunity for the real-time re-evaluation of how well the task is proceeding and if it should be continued, modified, aborted, superseded, etc. Used after the fact, a recorded checklist provides an audit trail to support post facto analysis for purposes of certification, accident analysis, improvement of designs and procedures, etc.

12 Smith, James. Personal account of an incident that happened while employed by Lockheed-Martin.

3.4.2.2 NN Perceptual Errors

Perceptual errors are misinterpretations of what is seen, heard, or otherwise received through the senses. They generally occur when sensory input is degraded (incorrect or unusual) or when the actual input is correct, but misinterpreted by the receiver.

As previously noted, distractions are events that could interfere with performing the task at hand, but for which otherwise ignoring them poses no undesirable consequences. For practical purposes, distractions can be treated as system noise. On the other hand, not all confusing situations are so benign.

In the case of the NN, perhaps an alternative strategy to the human's simply ignoring distractions would be more appropriate to the previously presented one of simply training to ignore distractions. Specifically, the NN could be trained to still respond correctly to events involving distractions as before, but also to report, as a separate status output, the recognition of detected distractions that are otherwise ignored.

From a certification or V&V perspective, when speaking of NNs, knowing what the NN ignores can be as important as knowing what it recognizes. The concept of being conditioned to ignore distractions that can cause errors has been previously described. A strategy for handling such distractions is to recognize and so note them, but not to modify the normal behavior of the NN. The status recognition may be based on information already present in the NN, or it could require additional input to make the determination of status. Thus, the NN can improve on the way humans handle distractions.

The NN does consciously what the human does subconsciously, namely the act of recognizing and ignoring distractions. This status recognition of distractions available to the NN could be made available to other systems, or even to the pilot. It also represents the beginning of an NN developing self-awareness, which is when the NN is aware of where it is procedurally. This information could be useful in comparing what the NN perceives itself to be doing to what the outside world perceives. Checklists could also be implemented external to the NN by another monitoring process that notes such milestones.

Events that are incorrectly treated as distractions by the NN could have serious consequences. The larger problem to be solved, before one can appropriately handle a distraction, is to determine whether the event classified as a distraction is indeed a distraction or a significant event. This consideration leads to the discussion of confusing situations.

A *confusing situation* is generally due to mixed signals including inconsistent inputs, conflicting requirements, and complex events. Three general cases can be considered: (1) misinterpretation due to incorrect internal processing of correct information; (2) misinterpretation due to external conditions where information and inputs are incorrect; and (3) both internal and external sources of misunderstanding exist simultaneously.

Confusion can be a result of a lack of experience, such as a system that has not previously been exposed to the given combination of what appears to be mixed signals. One or more events, which in a separate context could be clearly recognized and handled, are juxtaposed, but otherwise do not interfere with each other. This scenario could be considered as being one step beyond the distraction case where one meaningful task is embedded in a sea of noise. This scenario now consists of two or more meaningful tasks to be performed. They should be technically doable with sufficient preparation through training and practice.

The previous observation regarding distractions applies here. If a pilot is expected to operate in distracting and confusing situations, then the pilot's training and experience should include a repertoire of distracting and confusing situations. Likewise, the NN training and evaluation sets should be sufficiently rich in confusing inputs. In fact, *over-training* might be a consideration. For instance, with simulator training the pilot may be exposed to situations that are beyond what is considered likely or even realizable in a real-life setting. This approach could be viewed as a form of stress testing, analogous to the use of a treadmill as part of the evaluation of the general health and endurance of the human. Such practice is, in fact, part of the pilot certification process.

Similarly, stressful training events are *apropos* to the NN training and evaluation regime. This view of systems development corresponds to the previously mentioned certification philosophy that any system component should not only meet the specifications to which it is designed and built, but also exceed them beyond a reasonable margin of error.

3.4.2.3 NN Decision Errors

Decision errors represent conscious, goal-intended behavior that proceeds as designed, yet proves inadequate or inappropriate for that situation. They tend to occur when a familiar situation is not recognized, is misdiagnosed, or when an unfamiliar situation occurs and generally result in the application of an unsuccessful procedure.

The more complicated scenario involves confusing situations in which the confusion exists because the total set of inputs, while correctly received, form an inconsistent or conflicting view. The tasks may be performable if taken individually, and the information sources may be plausible if taken individually; however, taken together they are neither performable nor plausible. In the milder case, their juxtaposition presents an incomplete situation where the whole is greater than the sum of the parts with sub-themes formed by conditionally combining subsets of the information in specific but possibly novel ways. The set of possible themes may be ambiguous and extendible to multiple plausible interpretations. Which interpretations constitute a better understanding of the situation and a better choice of action needs to be determined.

Several general methods or approaches have been developed in an effort to attack this class of problem. The scenario mentioned above is an example of the *data fusion* problem.

“*Data fusion* is the seamless integration of data from disparate sources. The data have been integrated across data collection "platforms" and geographic boundaries, and blended thematically, so that the differences in resolution and coverage, treatment of a theme, character and artifacts of data collection methods are eliminated. At present, this is a desirable but unattainable goal [Hastings 1997].”

On the other hand, the identification of those sub-themes is the domain of methods such as data mining, which is the analysis of data using tools that look for trends or anomalies without the knowledge of the meaning of the data.⁹ Simply stated, the data fusion process could be viewed as *looking at trees and seeing a forest (an ecosystem)*; while the data mining process could be viewed as *looking for a needle, the correct needle, in a haystack*.

Humans are capable of performing both of these complex tasks once they are able to look at the body of information from the correct perspective. To determine and facilitate this perspective for various tasks has been one of the major thrusts of HFA. Before being able to

see a pattern in the information, the human needs an appropriate background on which to draw. This background includes having the proper formal training, practice, and experience, which are all part of the pilot certification process. Without that appropriate background, the human's performance would be similar to the definition of data mining, which includes the phrase "without knowledge of the meaning of the data."

NNs have been employed extensively for both data fusion [Whittington 1990] and data mining [Lu 1996] applications. The IFCS project uses NNs in this capacity. At issue is how to certify NNs to have this appropriate background (proper formal training, practice, and experience). The observation of the NN's correct performance, as part of their training, indicates that certain specific knowledge has been acquired. This knowledge must correctly generalize to other situations.

Much effort of the pilot certification process is given to ensuring that the human possesses this background. Similarly, the certification of the NN must ascertain whether the NN possesses this background. The knowledge in the NN is in a compiled form and embedded in the weights, links, and structure of the NN. The NN must be taken as a whole. It cannot be asked a specific "do you know how to ..." question. It is presented with a compiled experience, a training set or a real-life set, to which it responds completely. Decisions are, by definition, conscious efforts. Consequently, this conscious aspect must be characterized for the NN.

Rather than only considering whether the NN possesses the appropriate background, a more comprehensive effort is to determine what knowledge the NN possesses, background or otherwise. Then, IV&V experts could better judge the background knowledge for correctness and completeness. Furthermore, new knowledge, which was not previously stated explicitly in human understandable terms, might possibly be gleaned from such an effort.

To determine what specific knowledge an NN possesses, various efforts have been explored to capture the underlying knowledge in some human-readable, recognizable, and understandable format. These efforts include methods, such as rule extraction and decision-tree extraction, which lend themselves to visualization methods that assist the human in literally seeing potential relationships between the nuggets of knowledge from rules and decision tree branches [Boz 1995, 2002].

Much effort has been made to address the expert system/NN conundrum. NNs have powerful knowledge acquisition and extraction capabilities by being able to gather knowledge from available examples. However, NNs lack an explanation capability. In contrast to the NN, the weakest aspect of expert systems is knowledge acquisition, while an explanation capability is one of their strongest aspects. Some form of knowledge extraction will be critical to understanding how and why an NN makes certain decisions. Consider the scenario where it is doing the correct action for the wrong reason. The success of the current situation could lead the NN to apply that same knowledge inappropriately in another situation that may appear to be similar but is, in fact, different in significant and knowledge-rich ways.

4.0 INTEGRATION INTO ACTIVITIES AND PROCESSES OF V&V

The HFACS was developed for error analysis, looking at a specific incident and going backwards across time and effort to find the causes for the problem. A certification process, like a methodology for the IVVNN, would work from the opposite direction. It would start with ensuring good organizational practices and policies, consist of high quality software engineering procedures, employ mechanisms in the design and development of the system to ensure software quality, and finally test and analyze the system checking for possible existing errors before certifying the system for usage.

A consideration in a new methodology for the IVVNN starts with looking for parallels in Tier 4. This might be as simple as noting the use of existing standards such as the ISO 9000 and the IEEE 1012. The development of the methodology for the IVVNN, like the existing standards, would serve the purpose of providing proper organizational influences.

The next three tiers begin to steer the direction of the methodology development. Tier 3, which looks at supervisory influences, indicates better methods and techniques should be developed to ensure that those who work with NNs understand what the NNs are doing, can communicate this to others on the project, and ensure the NNs behave in a desired manner (including proper adaptation, correct knowledge storage, and acceptable outputs).

Tier 2 examines ways in which problems could be detected in the system before becoming failures. One way to look at this could be testing, but another more interesting aspect, would be that of a system oracle which could identify anomalous behavior and transition the system into a fail-safe or non-failure scenario. This kind of technology is akin to fault-tolerance concepts that are added into the system during development.

Tier 1 observes simple detection of problems, as they exist in the system. This points towards the most common aspect to V&V practices, which is testing.

When considering how to unite an existing standard with the lessons learned from HFACS, sections in the IEEE 1012 that are most impacted by NN technology must be identified. The IEEE 1012 is composed of several processes, including management, acquisition, supply, development, operation, and maintenance. Of these, development and operation are more critical toward the creation of a methodology for the IVVNN because the other processes require no special activities to account for NNs.

4.1 Process: Development

The development process of the IEEE 1012 defines the activities of concept, requirements, design, implementation, testing, and installation and checkout. The activities conducted within Tiers 1 and 3 point to tasks that should be done within the development cycle, and therefore heavily influence this section. The following three techniques (formal methods, visualization, and testing) look at ways to improve NN development, requirements, testing, and ultimately, understanding.

4.1.1 Formal Methods

The term, *formal methods*, refers to the use of techniques from formal logic and discrete mathematics in the specification, design and construction of computer systems and software. The purpose of applying formal methods is to make V&V of the software more objective by supplementing the traditional testing methods. The more rigorous the formal method, the

more effort and skill required to apply it, and the more assurance the method will provide. The formal methods of rule extraction, rule initialization, and rule insertion (refinement), as they pertain to NN software, appear to be promising techniques that can be used by the IV&V practitioner.

Rule extraction is the process of developing English-like syntax that describes the behavior of an NN. These techniques can convert the NN structure to a prepositional *if-then* format that offers the possibility of requirements traceability to a system that is not explicitly designed. The rules can also undergo design team review and analysis to detect improper NN behaviors or missing knowledge.

Through rule extraction, a system analyst might be able to ascertain novel-learning behaviors not previously recognized. By translating these features into a comprehensible English sentence, the analyst can gain not only a better understanding of the NN's construction, but perhaps the input domain as well.

The same techniques used to map rules from the NN in rule extraction can also be used in two additional ways: rule initialization or rule insertion.

Rule initialization is the process of giving the adaptive NN some pre-system knowledge, possibly through early training or configuration. A system developer may have improved confidence if the starting condition of the NN is known, which may lead to a constrained path of adaptation.

Rule insertion is the method of moving symbolic rules back into an NN, forcing the NN's knowledge to incorporate some rule modifications or additional rules. An adaptive NN could benefit from this scheme if the system developer wanted to exert a condition onto the NN or reinforce conditions in the NN. Examples of this might include restricting the NN to a region of the input space or instructing it to deliberately forget some data it has already seen.

Rule extraction from NNs may have greater utility for PTNNs than for dynamic NNs. PTNNs proceed through the steps of training and testing until they reach an acceptable error threshold and, only then, are used within a system. The knowledge of the domain is considered embedded inside the weights and connections of the NN. If the NN is no longer encouraged to adapt, an IV&V practitioner could then apply rule extraction to obtain a reverse requirement generation on this knowledge. These rules could then be compared against the original set of requirements and would provide information for review of the correctness of the function the NN is approximating. At a minimum, extraction of these rules would provide some sense of confidence that the NNs will behave as intended.

With a dynamic NN, it may be that symbolic rule extraction would be required at intermediate stages of its learning. At some intermediate points, symbolic rules would need to be extracted and passed through an oracle or system monitor to confirm that the NN was still *correct*. It may be that the benefits for dynamic NNs lie with rule insertion and rule initialization.

4.1.1.1 The IEEE 1012 and Formal Methods

Documenting the design features of an NN related to its performance may be an awkward and difficult task, especially in domains where the system designers are unsure of how to describe the intended results of the NN. This would lead to overly simplistic requirements

levied upon the NN and would not add great benefit to the goals of traceability analysis and test case generation.

One hope for rule extraction would be that it is capable of generating testable statements from an NN. This would allow system designers to compare the rules/knowledge of the NN against the intended system requirements and give an NN tester insight into appropriate test selections for verification.

4.1.1.2 PC-HFA and Formal Methods

Rule extraction appears well suited to address two of the concerns within Tier 1 of the HFACS framework: skill-based and decision-based errors.

Rule insertion and rule initialization may be useful for mitigating skill-based, knowledge acquisition errors, and by preparing an NN with some initial starting point, the acquisition time for skill-based knowledge would be reducible.

Rule extraction could be used as a means of decision-based error detection, as it would provide a white-box testing approach to the inner knowledge of the NN and allow for system developers to judge if the NN knowledge was correct and complete. Rule extraction, as mentioned previously, could benefit from linking with visualization techniques to lead towards better human understanding of the knowledge content of the NNs. A system developer could find that an NN has resolved rules describing an input domain that had not been previously anticipated.

4.1.1.3 Formal Methods Tool Example

RULEX is a tool that can extract symbolic *if-then* rules from analyzing the underlying structure of a specific kind of back-propagation NN. The goal is that these symbolic rules provide an insight into the decision making process of the NN which leads to understanding [Andrews 1995].

RULEX works as a decompositional rule extraction tool, analyzing the NN neuron-by-neuron to construct the symbolic rules. These symbolic rules take the form of:

IF Condition 1 AND Condition 2 AND Condition 3 AND ... THEN TRUE

The RULEX tool is designed for the constrained error back-propagation (CEBP) NN (a specific implementation of a multiplayer perceptron), but it may be generalized to other types of NNs. It may even be applicable to self-organizing maps (SOMs) like the dynamic cell structure (DCS) NN used within the IFCS project.

The neurons of the CEBP are sigmoid-based locally responsive units (LRUs), each representing a disjoint segment of the input training space. The LRUs are explained as being composed of a set of ridges, one ridge for each dimension of the input. The two-dimensional representation of a ridge for an LRU is seen in Figure 4-1. An example of an LRU activation region across two dimensions in a three-dimensional representation is shown in Figure 4-2. CEBP NNs are capable of handling multi-dimensional data as the LRU regions are created through the superposition of a ridge for each dimension of the input.

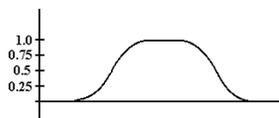


Figure 4-1. A 2D Representative of the Response Area of an LRU

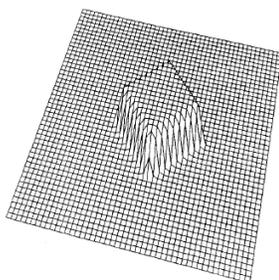


Figure 4-2. A 3D Ridge Representative of the Response Area of an LRU

The activation of the LRU only occurs if the value presented to the unit falls within the active range of the ridge. In this manner, these LRUs represent a region, and if the input stimulus falls within the region, the LRU activates.

Since all of the LRU ridges must be active to have the LRU active, the propositional *if-then* rules can be extracted from the LRUs. As an example:

IF RIDGE₁ is Active
AND RIDGE₂ is Active
AND RIDGE_N is Active
THEN Input Pattern is in the Target Class

This collection of *if-then* rules represents the description of the LRU, or that region of the input space. An example might be an NN that is trained to learn how to differentiate between automobiles. If an automobile is yellow, has six tires and is longer than it is wide, the NN might recognize it as a school bus. There would be three dimensions (color, tire count, and shape), and if an input into the NN activated those specific ridges (yellow, six, and longer than wide), then the input pattern would be classified as a school bus. This would be a rule for the NN and a way to investigate what knowledge it has retained.

The DCS NN in the IFCS project is comparable to a CEBP NN. With the DCS NN, similar groups of aircraft stability and control derivatives enter the NN and are clustered into groups. This similar data might be thought of as school buses, which appear comparable to each other and therefore fall into the same groupings.

Unlike the CEBP NN, where the ridges may be known classifications of automobiles, there are no pre-known conditions for the IFCS project and the DCS NN. Further, the DCS NN makes use of continuous data, not discrete values like *yellow* or *blue*. Through the use of distance metrics and neuron connection strengths, the DCS NN decides on its own how to best cluster the data into regions. The extraction of rules would then be English-like descriptions for what the DCS thought was a valid region of data. System developers could

look at these regions, and the values contained in them, and make assessments as to the validity or correctness for those regions.

The RULEX tool may be modified to be applied to non-CEBP architectures like the SOM; however, if the tool cannot be modified, a new or existing tool would need to be developed or identified that could be applied.

4.1.2 Visualization

Visualization is important for the V&V of NNs. Designers, end-users and IV&V practitioners need to understand the performance of the NN system: how it operates and arrives at its decisions. One tool that can be useful in meeting the goal of fully understanding the performance of the NN is visualization. Some forms of visualization involve transforming data into visual forms that can be more easily understood. Humans can comprehend vast quantities of data that have been visualized due to their highly developed visual pattern-recognition abilities.

There are a number of visualization techniques for understanding the learning and decision-making processes of NNs, as well as visual methods for testing the completed system containing an NN as a component [Craven 1992; Wejchert 1990; Munro 1991; Pratt 1993; Simmons 1997; Perhinschi 2002]. Additionally, visualization software can provide an interactive mechanism that enables the user to adjust parameters and quickly see the effects of the changes.

Visualization techniques may assist in understanding changes to the system that have occurred during training or in detecting errors and anomalies. Additionally, high fidelity simulations that include visualization can provide invaluable feedback for system integration testing. At this integration level in the testing process, the entire system can be evaluated and the input and output from the NN component can be analyzed. For example, current flight simulators can provide visual cues through a 3D visualization of the aircraft augmented by graphical representations of component analysis that look specifically at the NN component [Perhinschi 2002].

Visualization tools and techniques sometimes used in the V&V of traditional software (code coverage or structure examination) may prove even more important for NN software. Tools and techniques that assist understanding through visual representations may greatly aid an IV&V practitioner toward understanding the software they are called on to certify.

Visualization can aid in both developing and understanding projects involving NNs. Personnel involved in such a project may have little or no knowledge of the workings of an NN. Through the use of visualization technologies, such as the NN toolbox or even simple neuron models, the communication gap can be decreased or removed to allow all team members to attain some understanding of the project. Visual techniques can assist in discussions between project managers and system developers for requirement clarifications or between developers about testing results.

4.1.2.1 The IEEE 1012 and Visualization

Many visual techniques can assist V&V in the Development Process for NN software. Development V&V activities, such as concept (selecting architecture), requirements (defining functional and performance requirements), design (designing for software

component), implementation (transforming design into executable representations), and testing (software testing), may be addressed through the use of visualization tools.

Two-dimensional diagrams, three-dimensional plots, or even multi-screen displays of parameters could be used to visually compare structures and adaptation of the NNs. These activities could be used as a concept V&V activity to validate the constraints or limitations of the proposed NN architecture.

An example from the IFCS project serves to illustrate how visualization has been used as a tool to gain understanding of concepts and even help with writing requirements. Two types of NNs make up the components of the first generation intelligent flight control scheme for the IFCS project: a PTNN, which is composed of 34 separate multilayer perceptrons, and an OLNN, which is an SOM named DCS.

An iterative requirement process may be needed when requirements begin at a fairly high level and proceed to more refined statements. For instance, the IFCS project's initial requirements for the NNs used were aimed at metrics describing learning rates and acceptable errors over time. The next set of requirements was generated nearly three years later through reverse-engineering the previous software code. These requirements added detail beyond simple measuring metrics and began to include a discussion of acceptable inputs, acceptable outputs, detailed description of input processing, input scaling, and fault detection.

When the NNs were reverse-engineered, the first step was to document requirements based on observations of the code so that, when the system was re-implemented, it would be consistent with the previous implementation. The side effects of trying to re-implement the NNs were that the new requirements were actually better than the ones previously developed to describe how the PTNN worked, behaved, and was trained.

The DCS had to be reverse-engineered from two different sets of code, one in MATLAB and one version written in C. This process became very confusing until diagrams of SOMs were introduced. With the knowledge of the structure of the SOM, how the nodes evolved over time, and what connections meant, a model of the DCS was built in MATLAB. The DCS structure was then plotted across time and these plots were assembled into a movie. The movie was used at an early project-wide meeting to give the project team, who had limited knowledge of NNs, a basic understanding of how the DCS works and adapts. This movie proved to be an excellent tool to promote understanding and help the project gain support. It was very useful in explaining to the participants, especially managers, the workings of the DCS and how it would evolve over time. Now the project was at a point where the technical people had more understanding about the DCS (SOMs in general) and how it worked, and this in turn led the group to develop better requirements for the project.

Another process activity where visualization proves useful is testing. NNs are often tested as a black box; however, there are many visual techniques that would allow white box testing of NN software by the developers or IV&V practitioners. The capabilities of MATLAB's Neural Network Toolbox demonstrate some of these techniques that give visual examination to the internal workings of the NNs learning process [Mathworks 1998]. For the IFCS project, several plotting scripts were developed in MATLAB to look at the various results from the DCS (both in simulation and from a C version) to determine if it was working correctly. These scripts are still being used by the IFCS project today.

Visualization can aid in other aspects of the V&V process for NNs, including the design process, training set examination, examination of weights and biases during training, examination of structure after training, and analysis of operation during testing.

4.1.2.2 PC-HFA and Visualization

Visualization could help find routine violations that may have developed in the NN during training. Visual techniques can enable a PTNN trainer to examine the training data for omissions or outliers. The PTNN may have actually learned a violation of the expected operation and this could be determined by examining a visual representation of what has been learned.

For an OLNN, visualization could show an NN's proclivity towards a certain direction of adaptation. If the direction of adaptation is incorrect, then the designer can remedy the situation before the OLNN is deployed. An OLNN may, over time, begin to exhibit learning patterns that are considered violations. Visual tools and techniques can be useful in examining these patterns so the NN can be redesigned to prevent future occurrences of such violations.

For NNs, Tier 3 may correspond to the ability of adapting to data without proper guidance or control by system designers. Visualization may play a role in improving a system designer's supervision of NN adaptation.

Typically, training an NN is an automated routine: collect training data, process training data, set up an automated function for training, check for errors, modify the NN to some prior chosen method, and repeat. The designer can leave the system unattended and return when it is *done* learning. However, this may be inadequate supervision because the developer may not be sure of what the network learned. A visual interface could improve the supervision of the learning and lead to increased confidence in the system.

4.1.2.3 Visualization Tool Examples

MATLAB Simulink and Neural Network Toolbox provide comprehensive support for many proven NN paradigms, as well as a graphical interface that allows design and management of NNs.¹³ The toolbox simplifies the creation of customized functions and NNs. It has a graphical user interface for creating, training, and simulating NNs and has visualization functions for viewing performance.

One feature of MATLAB Simulink is the automatic generation of NN simulation blocks. In Figure 4-3 below, a three-layer NN has been converted into Simulink blocks indicating its structure. This tool can be used in the design activity to achieve a detailed design for the software component.

13 <http://www.Mathworks.com>

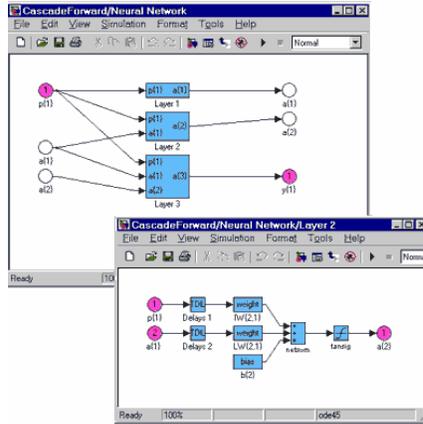


Figure 4-3. A Three-Layer NN Converted into Simulink Blocks.

Another visualization capability offered by this tool is the ability to model control system applications. NNs have been applied to the identification and control of nonlinear systems. The Neural Network Toolbox includes descriptions, demonstrations, and Simulink blocks for popular control applications: model predictive control, feedback linearization, and model reference adaptive control.

For testing activities, a Simulink model that includes the NN control block and plant model could be used. The example below shows a model for predictive control of a continuous stirred tank reactor (CSTR). In Figure 4-4, the upper left window shows the CSTR plant model that includes an NN block. The other windows allow one to visualize validation data (top right), to manage the NN control block (lower left), and the plant identification (lower right). These visualization features of Simulink could enhance the integration testing activities.

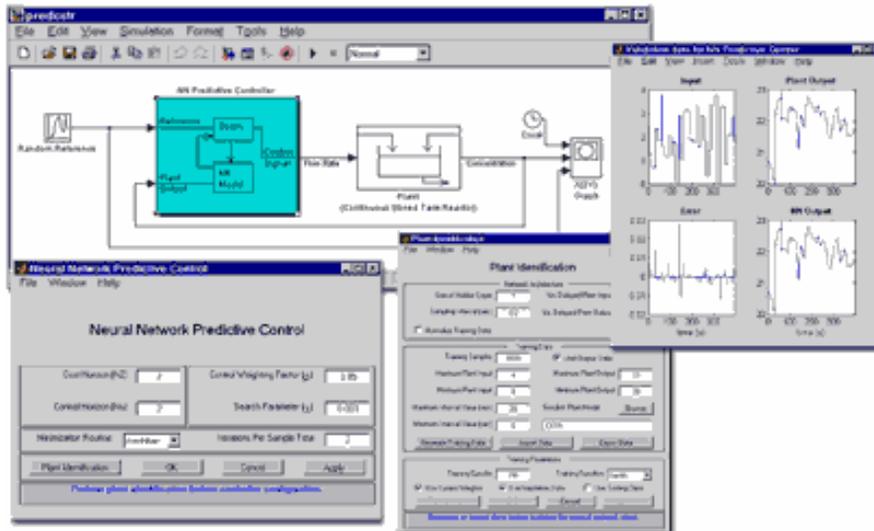


Figure 4-4. Simulink Model that Includes the NN

Other low-fidelity visual tools could include off-the-shelf graphical packages or original software developed for a specific purpose. The NN developer or IV&V practitioner could use graphical packages to analyze the training data or create specialized tools specific to the

individual situation for visualization of various aspects of the NN learning or operation. Greg Limes, a NASA Ames subcontractor working on the IFCS project, developed one such tool to watch the DCS adapt and train¹⁴.

The WVU F-15 Simulator, shown in Figure 4-5, provides a 3D representation of an aircraft and offers different viewing points, external and internal, to the vehicle [Perhinschi 2002]. It presents the traditional pilot instrumentation overlaid on the flying aircraft. Real-time MATLAB plots are generated during the flight and are displayed on the screen or stored on the hard drive for later analysis. The plots are user-selected and show various values including sensor data, error tracking of the research components, and pilot input.

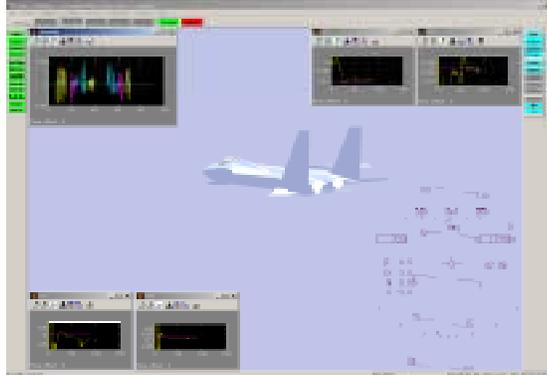


Figure 4-5. WVU F-15 Simulation

4.1.3 Testing

Testing is often seen as the central focal point for performing V&V activities and ensuring good software quality. Although many software engineers understand that testing alone cannot reliably produce successful software projects, they do recognize the importance of testing a system in the overall effort of V&V. Through testing, software engineers can verify the correctness, quality, functionality, performance, reliability, dependability, security, and usability of a system.

Testing activities include informal acceptance testing, component testing, formal acceptance testing, integration testing, and system testing. These activities should follow the order as shown in Figure 4-6.



Figure 4-6. Order of Testing Activities

Acceptance testing of a system validates that the software meets requirements and verifies that the NN is performing correctly for the problem domain. NN requirements would be comprised of testable NN features, such as acceptable error rates, timing issues and, perhaps, adaptability constraints including convergence and stability criteria.

¹⁴ Limes, Greg. Personal interaction with Brian Taylor on IFCS Project, 2001.

Because of the way NNs are developed, acceptance testing could be considered as both informal and formal. System developers will train PTNNs upon sets of training data and then test the PTNN with a smaller amount of testing data until the NN operates within specified acceptable limits. This testing comprises NN development, and can be thought of as an informal version of acceptance testing that continues as the NN develops.

Component or unit testing is the testing of individual software units or components to verify correct operation. Component testing for an NN may include testing of activation functions, as well as testing code that affects NN growth and shaping, adjusts the weights of a neuron, computes distance metrics, and is used for data interpolation or data scaling.

Formal acceptance testing would then follow component testing and would likely be performed by the system testers who were not necessarily involved during the informal acceptance testing.

Integration testing proves that, as software components are integrated to operate with one another, each component continues to work as it did before. Software engineers would need to investigate the NN interfaces for systems that integrate with other software components. This process would include input and output data units, range checking on input and output values (especially in regards to scaling), time responses, frequency of responses, and perhaps other data qualities like smoothness and trending.

System testing verifies that a completely integrated system meets all system requirements and extends beyond the testing of requirements levied upon the NN. At this stage, system testing would probably be best accomplished through a high fidelity level simulation of the entire system with special emphasis on analyzing the NN input and output data.

4.1.3.1 The IEEE 1012 and Testing

The V&V activities related to testing as defined within the IEEE 1012 include acceptance V&V test procedure generation and verification, integration V&V test execution and verification, system V&V test execution and verification, acceptance V&V test execution and verification. In addition to these tasks, there is also testing performed within the implementation activity under the IEEE 1012. This testing includes V&V test case generation and verification, V&V test procedure generation and verification, and component V&V test execution and verification.

These testing activities can be expanded to account for the additional constraints and needs of NN systems. Section 4.1.3.3 outlines some of the possible testing tasks which can be directed towards NNs and integrate well with the IEEE 1012.

4.1.3.2 PC-HFA and Testing

Tier 1 of the HFACS framework points to several considerations of human error and accident analysis that can be applied in the testing of NNs, specifically in the detection of errors in an NN system. Section 3.4 discusses ways in which skill-based, decision-based, and perceptual errors can affect an NN. Specific tests can be selected or developed that ensure these three types of errors will not be coded into the NN.

4.1.3.3 Testing Examples

System analysts have access to the internal structure that includes the NN's weights, connections, biases, activation functions, and neuron construction in PTNNs. This allows for

white-box testing including reliability analysis and traditional software testing practices, such as pathwise testing and other code coverage techniques. The original training data should be available and could be subject to analysis and testing itself. However, software engineers would be unlikely to test an OLNN as well as a PTNN. Some OLNN testing techniques include robustness analysis, simulation, and tests to investigate the underlying algorithms that construct the OLNN. Additionally, the OLNN can undergo interface testing, which includes the insertion of input data with various ranges to test NN reaction. OLNNs may also benefit from simulation and testing within different implementations to ascertain that the NN has been correctly implemented.

Improvements in Acceptance Testing

Issues to be addressed regarding acceptance testing (informal or formal) include:

- Appropriate NN selection
- Ability of NN to handle expected data range limits found in the input domain, and how range limits can be used to protect it
- Knowledge acquisition of the NN
- Timely NN convergence and stability

One NN testing practice that may facilitate acceptance testing would be reliability assessment. Some problem domains use a failure rate as a metric to indicate system reliability. It is not uncommon to see failure rates along the order of 10^{-5} or even 10^{-9} indicating that an acceptable rate of failure is once every 100,000 or 1,000,000,000 uses.

The difficulty with reliability assessment is that the traditional method for developing NNs leaves a smaller testing set than the data set used during training. The available testing data size may be too small to conduct a reliability assessment. To address this concern in an early stage of the IFCS project, an automated test data generator was developed.

The generator operates by clustering an existing small set of test trajectories, creating predictive linear or non-linear models that approximate these trajectories and then perturbing these models to generate statistically related, yet wholly new, trajectories that can be used for additional testing of the NN scheme.

Improvements in Component Testing

Issues to be addressed regarding component testing include:

- NN architecture implementation
- NN activation function implementation

One NN component testing practice, which has been used by members within the IFCS project, has been a method called *Gold Star*. When working with a *Gold Star* scheme, all members of a project work together to develop a single instance that implements a minimum of the requirements for the software project. In the case of NNs, this is essentially a single implementation of a chosen architecture that will ultimately be developed for the target system.

Subsequent versions of the solution include greater requirements coverage, with new versions compared for correctness against the *Gold Star*. For example, in the IFCS project, the PTNN was (1) implemented in Matrix-X, (2) auto-coded into Ada, (3) developed in

MATLAB and in Simulink, (4) programmed in a C implementation, which ran primarily on Unix platforms, and (5) implemented onto the target platform within C and designed to operate on a VxWorks embedded system.

Each time, newer versions of the NN relied upon the original Ada implementation for proof of correct operation. Having different implementations provided easily accessible tools for the different testers. The multiple versions also served as a form of cross-validation.

Improvements in Integration Testing

Issues to be addressed regarding integration testing include:

- Input measurement units consistency
- Output measurement units consistency
- Scaling factors verification
- NN sensitivity analysis to perturbations in the integrated system

One method used for integration testing within the IFCS project was the use of increasing system fidelities for testing. The first level comprised of an individual NN experiment implemented in MATLAB and Simulink. The second level was a combination of the PTNN, OLNN, and research components, all operating in C.

The next level of fidelity integrated the research experiments, which included the PTNN and OLNN, with the flight controller and a good-quality but low-sophistication F-15 simulation. The work was accomplished through WVU and implemented entirely within Simulink. This level included the usage of a 3D visualization package to provide visual feedback to system users who could control the aircraft through joystick inputs.

The most significant fidelity level involved a hardware-in-the-loop simulation (HILS) at a Boeing facility. Here, the target system was installed onto actual hardware and interfaced with a fairly sophisticated mock F-15 that included computers, actuators, and other equipment which would be found on the target F-15. A software environment worked alongside the HILS setup and allowed F-15 pilots to sit in a test cockpit and fly the aircraft against a 3D generated simulation.

4.2 Process: Operation

Within the operation process of the IEEE 1012 is defined a single activity, the operation activity. This activity covers the operation of the software system and how it is used. The IEEE 1012 does not have a perfect fit to account for autonomous systems, especially adaptive ones; however, Tier 2 does give some indication of what could be extended within the standard to account for NNs. One extension would be adding in system monitors and methods to assess the NN as it operates, continuously checking for indications of improper operation.

Online adaptation of NNs creates unique problems for V&V. Testing at any particular point in time *proves* the system only for that moment. The next data input, whether valid or not, has the potential to alter the behavior of the system as it adapts to accommodate the new information. Constant or periodic testing can detect system anomalies before a catastrophic event can occur.

The run-time monitoring program examines collected system information either to detect violation of system constraints or to manage resources during operations. The monitor can be designed to assess any number of NN characteristics. Examples include the following:

- Error metrics
- Output ranges and output value trending
- Number of training cycles to reach a point of stability
- Weight changes over time
- Node count over time

Ideally, the monitor would check the NN after each adaptation iteration, including learning/training steps, growth steps, and edge modification steps. Learning/training steps would be those procedures that modify the internal structure of the NN as it changes to new inputs, such as neuron weight modifications and neuron connection modifications. Growth steps are procedures that add (or remove) neurons to accommodate error reduction or eliminate old and obsolete neurons. Edge modification steps include those procedures that remove weak connections between neurons or add new connections as the neuron size increases.

One benefit of run-time monitoring is that, generally, it requires little incremental effort over traditional testing. It can locate difficult-to-find errors that testers might not find or envision. However, run-time monitoring could add overhead to program execution and may be prone to find false positives. Since run-time monitoring generally observes one execution, certain paths may not be covered in a specific run and some errors may be missed. Another drawback to run-time monitoring is that these schemes should be envisioned early in the project and developed concurrently with the NN systems. Currently, we believe at least four run-time monitoring techniques show good potential for use with NNs: safety monitors, Lyapunov stability, data sniffing, and built-in-testing (BIT).

Safety Monitors

A safety monitor looks at system characteristics or system data to detect anomalies without the need to have any specific knowledge of the NN. Safety monitors are generally application dependent.

Lyapunov Stability

Lyapunov stability is rooted in control systems theory and uses a continuously computed equation for stability analysis of linear and nonlinear systems, both time-invariant and time varying. It can provide insight into a system's behavior without solving the system's mathematical model. Viewed as a generalized energy method, it is used to determine if a system is stable, unstable, or marginally stable.

Data Sniffing

Data sniffing is used to assess data as it enters and exits an NN to determine its particular effects on the NN. For an OLNN, it could look to see if the data would cause instability to the learning or somehow corrupt previous knowledge. For a PTNN, it could be used to detect data values that were not originally used to train the PTNN, thus indicating the PTNN results carry less confidence.

Built-in-Testing

Built-in-testing is a scheme that is often used with traditional software. BIT allows for an NN to internally check itself for errors or other conditions that it may want to accommodate. The accommodation could be through signaling to the rest of the system that an error is occurring, taking a corrective action, or continuing operation, but in a degraded state that is flagged for later analysis.

4.2.1 The IEEE 1012 and Run-Time Monitoring

As currently described by the IEEE 1012, “The Operation V&V activity is the use of the software by the end user in an operational environment. The Operation V&V activity addresses operational testing, system operation, and user support. The objectives of V&V are to evaluate new constraints in the system, assess proposed changes and their impact on the software, and evaluate operating procedures for correctness and usability [IEEE 1998].”

Regarding NNs, the definition of operation V&V may need to be extended or modified to include continuing or on-going system assessment to check for dynamic system correctness and operation. This system assessment could then be conducted manually by a system engineer, or automatically by another routine or program, such as a run-time monitor.

4.2.2 PC-HFA and Run-Time Monitoring

Run-time and operational monitoring attempt to address the problems indicated in Tier 2 of the HFACS. One of the thoughts posed in Tier 2 discusses improving the weaknesses in a system’s defenses and capturing opportunities for preventing a given incident. For traditional software, such a system may be comprised of fault-tolerant software components, or systems that gracefully degrade, as discussed in Section 4.0.

Run-time monitors would attempt to prevent a system from failing before a fault occurs by catching improper data or improper behavior called substandard conditions. The act of catching a possible failure scenario significantly contributes towards a preemption strategy for preventing a future unsafe act.

4.2.3 Run-Time Monitoring Examples

The IFCS project developed two forms of run-time monitoring. One monitor was developed for the PTNN; while another was developed to accommodate safety assurance of the OLNN.

PTNN Safety Monitor

When in use, a PTNN can be thought of as a table of data with a sophisticated data lookup capability. It will not adapt, so the contents will not change. The domain knowledge has already been inserted and exists within the PTNN; it is only used to extract this data during operation. From a safety standpoint, system engineers may want an extra layer of protection to prevent the PTNN from generating anomalous data which testing was not able to uncover.

For the IFCS project, the PTNN was implemented within a *Class B* system, a designation of the NASA DFRC that is comparable to Level-3 criticality in the IEEE 1012 describing software integrity levels. The data from this PTNN system fed the flight controller that resided in a *Class A* system, corresponding to Level-4 in the IEEE 1012. Because the designers of the *Class A* system wanted an extra layer of protection to check the PTNN data

for correctness before it was allowed to be used in the flight controller, a safety monitor was developed.

The safety monitor was able to generate acceptability ranges with which to check the *Class B* PTNN data. Should any PTNN output exceed these defined ranges, the system would handle the error by transition away from an enhanced mode and back to a conventional operation.

OLNN Safety Monitor

The OLNN component to the IFCS project presented a different problem. Since it was going to adapt during flight, its expected outputs were not known prior to deployment. The PTNN safety monitor solution was not viable for the OLNN. To resolve this issue, the OLNN safety monitor designers looked at how to trust the OLNN data at later points in the system.

In the IFCS project's flight control scheme, the OLNN output is combined with the PTNN output and fed into other systems that compute aircraft gains used by the flight controller. While there would be no way to discern if the OLNN outputs were valid, there is a way to box the aircraft gains into valid ranges. Should an OLNN output be invalid, it would subsequently cause an invalid gain calculation that would, in turn, be identified through detected violations of the valid gain ranges.

With this scenario, the OLNN safety monitor was not directly applied to the OLNN outputs, but instead used upon later computed values in the systems that used these OLNN values.

5.0 CONCLUSION

This stage of the methodology development brings together three core ideas: the study of PC-HFA, implementation within the IEEE 1012, and four V&V of NN techniques (formal methods, visualization, testing, and run-time monitoring). Some of these techniques have prior use in the IFCS project and some show strong promise for the IFCS project in the future. While this process will be open to new developments, at this stage, research for this methodology will focus on V&V technologies and concepts discussed in this document.

The next phase of the methodology development will look into refining and analyzing of the four different V&V techniques. Plans for formal methods include the development of a new rule-extraction tool that can operate on SOMs. For visualization, this may include developing individual visualization tools that can be used on assorted NN architectures during the design (training) and testing stages. Development of visualization techniques will certainly involve enhancing the WVU F-15 simulation to investigate features specific to the needs of IV&V within NN simulations. The trajectory generator which falls under testing methods requires further maturity before it can be considered a practical tool. There would also be a benefit from documenting the IFCS project's experiences in regards to NN testing to look for useful NN analysis steps, even though the IFCS project represented a *Class B* or the IEEE 1012's Level-3 NN software criticality level. Further work for run-time monitoring techniques will look at Lyapunov stability and documentation of the steps considered in the creation of the IFCS project's safety monitors.

The relationships between PC-HFA and these four V&V techniques require some further discussion. This would include the raising of questions, such as which of these four techniques are more important, and should identify questions which an IV&V practitioner will need to consider as certification processes are performed. Some of these questions are already outlined in this document (such as the questions raised in Section 3.4 and 4.1.3).

There will also need to be communication between the ISR researchers and different NASA IV&V personnel to determine if there are V&V standards in place in addition to the IEEE 1012. While this methodology is planned to be a *hook* into the IEEE 1012, or perhaps a supplemental standard, any other practices commonly used by NASA IV&V contractors and staff should be considered for possible inclusion, or aid in the development of the methodology.

In view of the IEEE 1012, the final version of the methodology will take into consideration each process, activity, and task that the IEEE 1012 uses for traditional software. PC-HFA will assist in evaluating each task defined within the IEEE 1012 to determine how NN technology would influence that task. PC-HFA will also identify additional tasks and techniques that the IEEE 1012 lacks, which need to be accounted for within the IVVNN.

6.0 CITATIONS

- Andrews, R., and S.Geva. 1995. RULEX & CEBP networks as the basis for a rule refinement system. In *Hybrid Problems, Hybrid Solutions*, ed. John Hallam. IOS Press.1-12
- Bird, Jr. Frank E. 1974. *Management Guide to Loss Control*. Atlanta: Institute Press.
- Boz, Olcay. 1995. Knowledge Integration and Rule Extraction in Neural Networks. Ph.D. proposal Lehigh University.
- Boz, Olcay. 2002. Extracting Decision Trees From Trained Neural Networks. Paper presented at the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 2002.
- Craven, Mark W., and Jude W. Shavlik. 1992. Visualizing learning and computation in artificial neural networks. *International Journal on Artificial Intelligence Tools* 1(3):399-425.
- Department of Transportation (DOT). 1997. Pilot, Flight Instructor, Ground Instructor, and Pilot School Certification Rules. Federal Aviation Administration. RIN 2120-AE71.
- Ford, C., T. Jack, V. Crisp, and R. Sandusky. 1999. Aviation accident causal analysis. *Advances in Aviation Safety Conference Proceedings*, (P-343). Warrendale, PA: Society of Automotive Engineers Inc.
- Geller, E. 2000. Behavioral safety analysis: A necessary precursor to corrective action. *Professional Safety*, 29-32.
- Gutierrez, Maria. 2002. HFACS Reports, Quarterly Newsletter of Marine Facility Incidents using The Human Factors Analysis & Classification System.
- Hastings, David. 1997. *Data Fusion, What is it?* [online]. NOAA National Data Centers, NGDC. [cited 30 January 2003]. Available from World Wide Web: (<http://www.ngdc.noaa.gov/seg/tools/gis/fusion.shtml>)
- Institute for Scientific Research, Inc. (ISR). 2000. *Software Verification and Validation Plan for the Airborne Research Test System II Intelligent Flight Control Program*. IFC-SVVP-F001-UNCLASS-120100.
- Institute of Electrical and Electronics Engineering, Inc (IEEE). Software Engineering Standards Committee. 1998. *IEEE Standard for Software Verification and Validation*. New York, NY.(IEEE 1012-1998).
- Isaac, Anne, and Julia Pounds. 2001. Development of an FAA-EUROCONTROL Technique for the Analysis of Human Error in ATM. Paper presented at 4th USA / Europe Air Traffic Management R&D Seminar, 3-7 December, Santa Fe, NM.
- Lu, Hongiun, Rudy Setiono, and Huan Liu. 1996. Effective Data Mining Using Neural Nets. *IEEE Transactions on Knowledge and Data Engineering*, 6 November, 8.
- Mackall, Dale, S. Nelson, and J. Schumman. 2002. *Verification & Validation of Neural Networks for Aerospace Systems*. NASA Ames Research Center.
- Mathworks, Inc. 1998. *Neural Network User's Guide (version 3)*.

- Munro, P. 1991. Visualization of 2-D hidden unit space. Technical Report LIS035/IS91003, School of Library and Information Science, University of Pittsburgh, Pittsburgh, PA.
- Naval Safety Center. 1996. Quality Management Board Charter - Reducing Human Error in Naval Air Operations.
- Nelson, S., and C. Pecheur. 2002. *V&V of Advanced Systems at NASA for Northrop Grumman Corp.* Produced for the Space Launch Initiative 2nd Generation RLV TA-5 IVHM Project. NASA Ames Research Center.
- O'Hare, D., M. Wiggins, R. Batt, and D. Morrison. 1994. Cognitive failure analysis for aircraft accident investigation. *Ergonomics*, 37, 1855-69.
- Perhinschi, M. G., G. Campa, M. R. Napolitano, M. Lando, L. Massotti, and M.L. Fravolini. 2002. Modeling and simulation of a fault tolerant flight control system. Submitted to *International Journal of Modelling and Simulation* in April 2002.
- Pratt, L. Y., and S. Nicodemus. 1993. Case studies in the use of a hyperplane animator for neural network research. In *Proceedings of the IEEE International Conference on Neural Networks, IEEE World Congress on Computational Intelligence*, 1:78-83.
- Reason, James. 1990. *Human Error*. New York: Cambridge University Press.
- Simmons, Reid and Gregory Whelan. 1997. Visualization tools for validating software of autonomous spacecraft. In *Proceedings of the 4th International Symposium on Artificial Intelligence, Robotics, and Automation for Space (I-SAIRAS)*, Tokyo, Japan.
- Southeastern Oklahoma State University (SOSU). Aviation Sciences Institute. 2002. Course notes for AVIA 4643 - Physiology: Human Factors & The SHELL Model. Fundamentals of Safety Engineering and Human Factors.
- Wejchert, J., and G. Tesauro. 1990. Neural network visualization. In *Advances in Neural Information Processing Systems*, Vol. 2, 465-472. San Mateo, CA.: Morgan Kaufmann.
- Wells, Alexander T. 1997. Human Factors in Aviation Safety. Chapter 5 in *Commercial Aviation Safety*. Edited by A.E. Jackson. McGraw-Hill.
- Whittington, G. and C. T. Spracklen. 1990. The Application of a Neural Network Model to Sensor Data Fusion. In *Proc. SPIE—The International Society for Optical Engineering*, January 1990.
- Wiegmann, D., and S. Shappell. 1999. Human error and crew resource management failures in Naval aviation mishaps: A review of U.S. Naval Safety Center data, 1990-96. *Aviation, Space, and Environmental Medicine*, 70: 1147-51.
- Wiegmann, Douglas A., and Scott A. Shappell. 2001. Human Factors Analysis and Classification System (HFACS): A Human Error Approach to Accident Investigation. *Aviation, Space, and Environment Medicine*, 72:1006-16.
- Yacavone, D. W. 1993. Mishap trends and cause factors in Naval aviation: A review of Naval Safety Center data, 1986-90. *Aviation, Space and Environmental Medicine*, 64, 392-5.

APPENDIX A - ACRONYMS

ARC	Ames Research Center
ARTS	Airborne Research Test System
ATM	Air Traffic Management
BIT	Built-in-Testing
CEBP	Constrained Error Back-Propagation
CSTR	Continuous Stirred Tank Reactor
DCS	Dynamic Cell Structure
DFRC	Dryden Flight Research Center
FAA	Federal Aviation Administration
GRC	Glenn Research Center
HERA	Human Error in ATM
HFA	Human Factors Analysis
HFACS	Human Factors Analysis and Classification System
HILS	Hardware-in-the-Loop Simulation
IEEE 1012	IEEE Standard for Software Verification and Validation, IEEE Std. 1012-1998
IFCS	Intelligent Flight Control Systems
ISR	Institute for Scientific Research, Inc.
IV&V	Independent Verification & Validation
IVVNN	Independent Verification & Validation of Neural Networks
JPL	Jet Propulsion Laboratory
LRU	Locally Responsive Unit
NN	Neural Network
OLNN	On-line Learning Neural Network
PC-HFA	Pilot Certification Based on Human Factors Analysis
PTNN	Pre-Trained Neural Network
SFDIA	Sensor Fault Detection, Isolation, and Accommodation
SOM	Self-Organizing Map
V&V	Verification and Validation
WVU	West Virginia University