Supported by
National Aeronautics and Space Administration
Graduate Student Researchers Grant NGT4-52401

KNOWLEDGE-BASED AIRCRAFT AUTOMATION

MANAGERS GUIDE ON THE USE OF ARTIFICIAL INTELLIGENCE FOR AIRCRAFT AUTOMATION and VERIFICATION AND VALIDATION APPROACH FOR A NEURAL-BASED FLIGHT CONTROLLER

Ron Broderick

12 May 1997

Engineering and Production Management
Loyola Marymount University

Executive Summary

The ultimate goal of this report was to integrate the powerful tools of artificial intelligence into the traditional process of software development. To maintain the US aerospace competitive advantage, traditional aerospace and software engineers need to more easily incorporate the technology of artificial intelligence into the advanced aerospace systems being designed today. The future goal was to transition artificial intelligence from an emerging technology to a standard technology that is considered early in the life cycle process to develop state-of-the-art aircraft automation systems.

First it provided a matrix that
ous artificial
guidance to
velopment of
ally evaluate

ence into the
uncertainty
sting aircraft
intelligence
face, system
aircraft flight
tand artificial

networks as
ftware quality
ural network
e applied to
result was a
at use neural

LOYOLA MARYMOUNT
UNIVERSITY

Engineering and Production Management

NASA Dryden Flight Research Center
P.O. Box 273
Edwards, CA 93523-0273
Mail Station: D-1044

July 18, 1997

SUBJECT: Final Close Out of NASA Dryden Grant NGT4-52401

Dear Mr. Steve Yee,

On behalf of Loyola Marymount University, we would like to thank you for the opportunity of performing research under NASA Contract NGT4-52401, "Knowledge-Based Aircraft Automation" for the NASA Graduate Researchers Program. Enclosed are five copies of the final report titled, "Managers Guide on the Use of Artificial Intelligence for Aircraft Automation and Verification and Validation Approach for a Neural-Based Flight Controller."

Please forward a copy to Dr. Kajal Gupta, the NASA Technical Officer, who we thank for his excellent support of the research effort. Please forward a copy to Mr. Leonard Voelker, the NASA Technical Sponsor, who encouraged the research and set-up of the

Acknowledgments

This study of Knowledge-Based Aircraft Automation was supported by a NASA Dryden Graduate Student Researchers Grant NGT4-52401 to Loyola Marymount University, Engineering and Production Management Program, Department of Mechanical Engineering, College of Science and Engineering.

I am grateful to the faculty of Loyola Marymount University, who encouraged me to pursue the grant, and who shared so generously of themselves, their knowledge, and enthusiasm, for completely this study especially:

Dr. Bohdan Oppenheim, Department of Mechanical Engineering
Dr. Stephanie August, Department of Electrical Engineering &Computer Science
Dr. Mel Mendelson, Department of Engineering & Production Management

I am thankful to my wife, Andreina, whose love, support, encouragement and airplane cookies kept me afloat.

I am grateful to my close friend, Leonard Cash, for his wisdom, humor, clarity and editorial guidance.

I am thankful to Dr. Birute Vileisis from the Loyola Academic Grants Office for her administrative assistance.

I am appreciative to Roberta Rubin from the Loyola Learning Center for her keen eye in editing and for her continuous encourage.

I am grateful to my NASA technical sponsor Mr. Leonard Voelker, from Dryden Flight Research Center for his encouragement, for the meeting he set-up for my presentation and for allowing me to experience some of the exciting engineering taking place at Dryden. I am grateful to Dr. K. Gupta, for his generous managerial support. I am grateful to Mr. John Bozworth for setting up my second presentation.

I am grateful to Dr. Charles Jorgensen from NASA Ames Research Center for allowing me to visit his Intelligent Flight Control Simulation Lab. and for the assistance in writing the paper titled "Verification and Validation Approach for a Neural-based Flight Controller" to be presented at the Artificial Neural Network in Engineering Conference at St. Louis, Missouri on November 9-12, 1997.

Table of Contents

Table of Figures

1.0 INTRODUCTION

Artificial intelligence (AI) is currently one of the most challenging topics in the field of aircraft automation, because of the safety-critical nature of these systems and the newness of their application. The problem is the discipline of artificial intelligence has not been applied by traditional engineers in aerospace and mechanical industries. It has been mastered primarily by a small community of applied mathematicians and computer scientists who, typically, have little or no knowledge of aircraft technology. For this reason, the powerful advances in the AI field have found little application in engineering practice. The goal of this report was to integrate the power of AI into the traditional process of software development. The report effectively transitions AI from an emerging technology to a technology that was routinely considered by engineers for the development of aircraft automation systems. The approach was to develop 1) a relationship matrix that identified aircraft automation tasks that were good candidates for AI implementation, and 2) a formal evaluation process for neural networks.

This report addresses two timely and critical topics in this field: 1) guidance to managers in the use of Artificial Intelligence in aircraft automation, and 2) the verification and validation of neural networks. Advances in electronics and computer technology have had a profound effect on modern aircraft. The evolution of artificial intelligence and computer technology provides the ability to develop more advanced, sophisticated and sturdy cockpit systems. Piloting is a classical expert behavior and the complexities of aircraft systems will provide challenges for the systems developer for years to come.

Section 2.0 of this report is intended to overcome the lack of understanding of the use of AI in the area of aircraft automation. To remedy this deficiency, a matrix was developed which identified typical aircraft automation topics conducive to various AI methods. The purpose of this matrix was to provide top-level guidance to managers contemplating the possible use of AI in modern avionics systems. The section started with an explanation of various AI methods that have been successfully utilized in aircraft automation. Next, an explanation of the characteristics of aircraft automation tasks that are conducive to AI is presented. The matrix is listed in section 2.2

Section 3.0 deals with a much narrower problem, the use of an AI method called neural networks which is applied to a neural-based aircraft flight controller. This particular AI method has been selected for two reasons. First, neural networks was selected to demonstrate an example of the power of AI in aircraft automation with some technical depth, and to provide educational benefit to the report author. Second, this application of AI was of strong interest to the project sponsor, NASA Dryden Flight Research Center, who needed a method for evaluating the neural-based flight controller before flight test. This section presents a verification and validation (V&V) method for the evaluation of a neural-based flight controller developed at NASA Ames Research Center and scheduled for flight demonstration on the F-15 "ACTIVE" aircraft at NASA Dryden Flight Research Center.

In the field of Computer Science, software has been traditionally evaluated, verified and validated using standard quality assurance methods. As neural networks gain wider

acceptance and are successfully used in the implementation of safety critical systems, it will become possible to progress from using standard guality assurance methods to an approach that will be suitable for neural networks. Since such networks are created using a very different paradigm from conventional software, a new framework for V&V was necessary. This section begins with a discussion of the functionality of traditional and neural-based flight controllers. Next, the NASA standard V&V approach for software was described. With this background, the report proceeds to expand the standard NASA approach to incorporate the evaluation of neural networks.

Because of the multi disciplinary nature (aerospace engineering, software engineering, computer science, knowledge engineering) of this report, a glossary of technical terms and an index were provided. Advances in the understanding of AI in aircraft automation are critical at this time. In general, both AI and neural networks are being considered for safety critical systems. The increased utilization of these technologies requires a guided, systemic application that was addressed in this study.

2.0 AI Methods Matrix Vs Aircraft Applications

There is a kaleidoscope of international research and development activities related to knowledge-based aircraft automation. Topics range from man-machine interface to situation assessment, automatic flight planning, and flight control. Much work has been accomplished in the development of knowledge-based aircraft systems, but the results are still not mature enough for routine use. There are several reasons knowledge-based aircraft systems are desirable. First, the availability of technology itself tends to drive the design of aircraft automation. Second, since over half of all aviation accidents are caused by human error (Nagel,1988), concern for safety and the belief that automation can reduce human error lead to new automated systems. Third, fuel economy can be enhanced through automated navigation. Fourth, automation offers the potential for reducing crew workload and stress. Finally, the special requirements of military missions, such as the need for extreme flexibility -- flying at the lowest possible altitude (300-500 feet) at fast speeds (400-600 knots), the complexity of sensor and weapon control tasks, and the importance of timing, demand the use of knowledge-based systems to support military pilots.

2.1 The Matrix

The Matrix shown in Figure 1 presents aircraft applications as rows and appropriate artificial intelligence methodologies used in the development of aircraft automation systems as columns.

2.1.1 Matrix Columns: AI Methods Used In Aircraft Automation

The discipline of artificial intelligence is a large, ever evolving area of knowledge. There is no clear separation between the various methods. Some methods overlap and others are applied in combination, as hybrid methods. All this tends to cause confusion among the uninitiated, that is, people that are not computer scientists or mathematicians. The organization of the various AI methods presented here is not standard. However, it is somewhat simplified since it is intended to facilitate an understanding of the AI discipline and assist aerospace managers in selecting the most suitable AI method(s) for a given application.

The various AI methodologies are organized into six categories as shown in Figure 1:

1. Logical Methods
2. Object Representation-based Methods
3. Distributed Methods
4. Uncertainty Management Methods
5. Temporal Methods
6. Neural Methods

Aircraft Applications vs. AI Methods Matrix

	Artificial Intelligence Methods	Logical Methods	Object Representat-ion-based Methods	Distributed Methods	Uncertainty Management Methods	Temporal Methods	Neural Methods
Aircraft Applications	Referenced Authors						
Pilot Vehicle Interface (PVI)							
(PVI) Pilot Task Management	1) Funk 2) Shelnutt 3) Krobusek 4) Onken	● ● ●	●	●		●	
(PVI) Mission Management	1) Wilber 2) Amalberti 3) Chin	● ●	●	●	●		
(PVI) Cockpit Displays	1) Shelnutt			●			
(PVI) Pilot Intent Pilot Error	1) Mitchell 2) Onken 3) Chin 4) Rouse	● ● ● ●	●	●			
(PVI) Pilot Dialogue	1) Onken		●				
Situation Assessment	1) Chin 2) Onken	● ●	●	●	●	● ●	
Automatic Flight Planng	1) Onken	●	●				
System Status Diagnosis	1) Ball	●	●	●	●		
Aircraft Flight Control	1) Jorgenson 2) Totah 3) Jorgenson & Schley 4) Steck 5) Painter 6) Werbos						● ● ● ● ● ●

Figure 1 - Aircraft Applications Vs AI Methods Matrix

The Logical Methods are based on formal logic and are used to represent and reason about procedural knowledge. Object Representation-based Methods are used to represent and reason about goals, plans, scripts, cases, and events and capture knowledge about such items. Distributed Methods are used to coordinate multiple reasoning options. Uncertainty Management Methods are used to deal with uncertainties in non-deterministic conditions. Temporal Methods are used to reason about time-dependent events or activities. Neural Methods are based upon the reasoning of the brain's biological processing of information and decision making. Each methodology is described in Section 2.2.

The dots in the boxes in Figure 1 identify where an AI method has been successfully implemented in an aircraft application. The names of the authors of various references that describe the given implementation are listed in the second column of Figure 1. The numbers listed before the names refer to the short explanations of each author's work, in subsections 2.3. For example, the referenced author, "Funk", which is row " PVI Pilot Task Management"which means that this author's referenced article is about the subject of Pilot Vehicle Interface, Pilot Task Management. The dot adjacent to Funk is in the column that represents Logical Methods, which means the implementation of that task was developed using a Logical Method. To obtain additional information about this implementation, go to subsection 2.3.1 Pilot Task Management and item 1 will provide a short summary of Funk's paper. In addition, Funk is listed in the reference material in case a copy of his paper is desired.

2.1.2 Matrix Rows: Aircraft Automation Applications Conductive To AI

With AI, it has become possible to implement such methods in a number of technologies, including the field of aircraft automation. The aircraft automation applications are conveniently grouped into the following five categories:

1. Pilot-Vehicle Interface, with it's subtasks of Pilot Task Management, Mission Management, Cockpit Displays, Pilot Intent, and Pilot Dialogue
2. Situation Assessment
3. Automatic Flight Planning
4. System Status and Diagnosis
5. Aircraft Flight Control

The rows in Figure 1 contain these categories.

The applications that are candidates for artificial intelligence are those that require the system to:

- infer a useful answer when the required input information is incomplete
- reason and form conclusions even when the real world does not conform to rigid assumptions required by analytical algorithms
- act like humans and can cope with ambiguous, vague and uncertain environments

Before AI, such tasks relied entirely on human intelligence. Automation, if any, was based on analytical algorithms. An analytical algorithm is basically a recipe. If the recipe is followed, predictable results will be achieved. Analytical algorithms exhibit excellent performance characteristics for tasks requiring a significant amount of computations. Future aircraft automation systems will be based on a combination of artificial intelligence and analytical algorithms. Artificial intelligence deals with rules of thumb and incomplete information. When the aircraft does not conform to rigid assumptions required by analytical algorithms, artificial intelligence systems can generate a result. Like humans, artificial intelligence systems have an "intuitive" ability to cope with ambiguous, vague and uncertain environments. They make deductive decisions in novel situations and generalize from past experiences.

2.2 Artificial Intelligence Methods

This section provides a brief description of the field of artificial intelligence. AI is described as a set of tools that a manager can utilize to build aircraft automation systems. The introduction describes the basic AI agent-based architecture. Next, the AI software portion of the architecture is presented. The remainder of this section is devoted to the description of the six methodologies that constitute AI software, as explained in 2.1. The basic structure of an AI system in terms of agent-based architecture is shown in Figure 2.

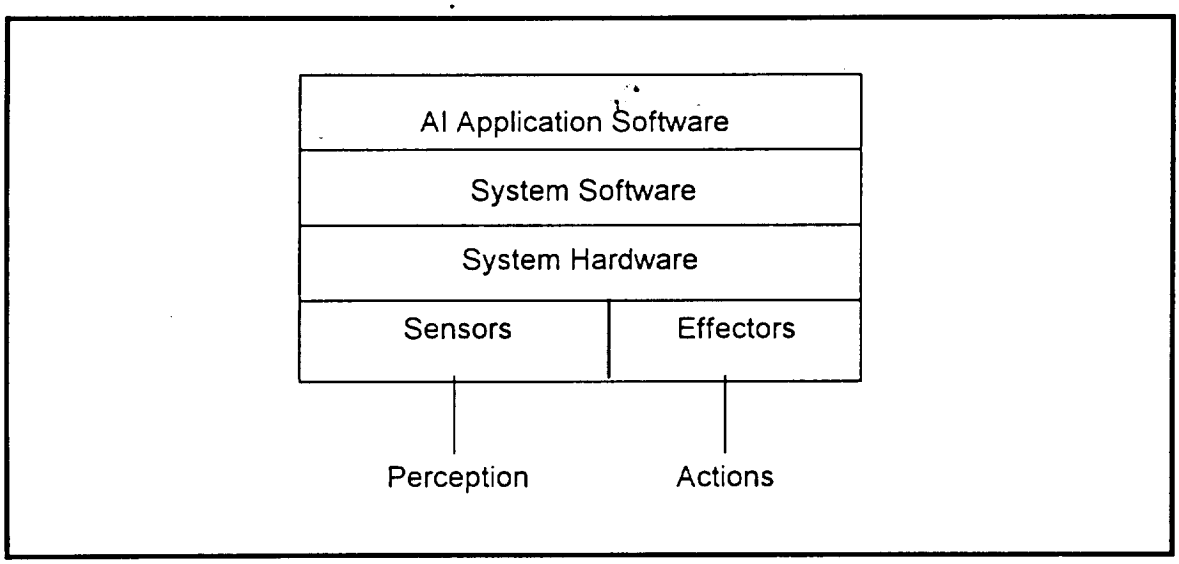

Figure 2 - System with AI Agent-based Architecture

The typical cockpit AI system has a set of sensors to receive input and a set of effectors to allow output in the form of actions. The system is composed of hardware, system software and AI application software. A simpler example of an agent is a PC: where the input is the keyboard, the output a set of commands shown on the monitor, and the system hardware and software are the PC hardware and operating system.

The AI software system architecture is pictured in Figure 3.

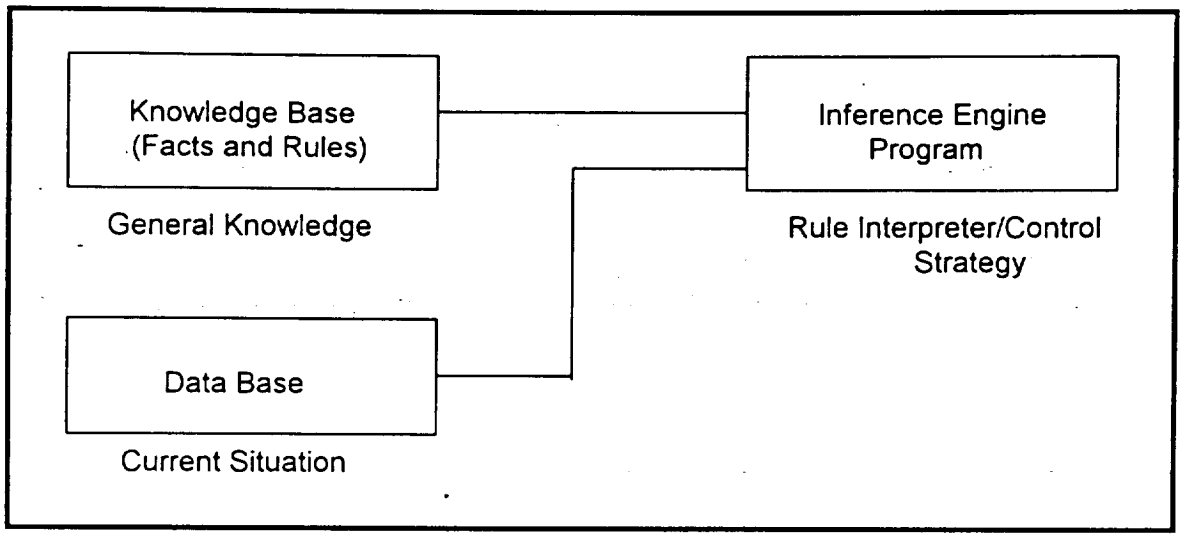

Figure 3 - AI Application Software

This architecture is different from conventional algorithmic programs in that the knowledge is separated from how it is used. Two other fundamental differences between conventional software and AI software are: 1) AI software uses highly domain-specific knowledge, and 2) the knowledge employed are heuristic rather than algorithmic in nature (Gonzalez, 1993, page 22). The architecture is composed of three separate entities: the knowledge base, the database, and the inference engine program. The knowledge base contains the problem domain knowledge that is explicitly represented as a collection of facts, rules, and relationships about the problem domain objects and concepts. The database contains assertions about the problem currently under consideration. The inference engine component represents the control of operations that continuously update the database allowing it to draw conclusions, and establishes the sequence in which different rules in the knowledge base are brought to bear on the problem.

In the development of a knowledge-based system, the two areas of development are the knowledge base and the inference engine. The knowledge base contains all the relevant, domain-specific, problem-solving knowledge. The choice of AI methods to represent this knowledge is derived from the nature and format of the application domain knowledge. The inference engine is the interpreter of the knowledge stored in the knowledge base. It consists of a set of operators and controls. The choice of a reasoning method is based on the knowledge representation and the application domain. Depending on the reasoning method selected, the inference engine can have the ability to make inferences, that is, derive new facts from the knowledge base and add these facts to the application knowledge base.

Paramount in the development of a knowledge-based system is the development shell. Development shells support knowledge representation and reasoning and assist in the structuring, debugging, modifying and expanding of the knowledge base. Depending on the application, development shells can be developed for a specific project or can be acquired commercially.

The six methods used to implement the knowledge base and the inference engine are the topic of the following subsections.

2.2.1 Logical Methods

Logical methods consist of the following: logic-based systems, rule-based systems, and search. Logic-based systems, rule-based systems have logic as their theoretical foundation. Basic search methods have always been used in AI to solve problems by exploring the problems domain-specific problem space.

Logic-Based Systems

In logic-based systems, logic is utilized for knowledge representation, as well as, reasoning and inference. Logic originated with the ancient Greeks as an accepted set of rules of reasoning. The nature of logic-based systems is algorithmic. Predicate logic includes propositional logic. Propositional logic is composed of simple statements that have a value of true or false. Simple statement are combined through the use of propositional connectives (e.g., and, or) to form complex statements which have a value of either true or false. Predicate logic, is more useful since it has the ability to express relationships between objects. These objects can be people, concepts or other objects. Predicate logic can also represent actions or an action relationship between objects.

An example of a simple propositional logic statement is:

Loyola is a university

This is a simple statement having a value of true or false.

An example of a complex propositional logic statement is:

(Loyola is a university) and (Loyola is in Westchester)

This is the conjunction of two simple statements, joined using the 'and' connective.

In the predicate logic, more complex logic can be expressed using predicates. For example:

controls(pilot, f-15)

expresses the predicate 'controls' to represent the action relationship between two terms; the pilot performing the act of controlling an F-15 aircraft.

Predicate logic allows one to have variables that represent objects that at the moment may be unknown. Using the above example, the variable X could represent any aircraft as follows:

controls(pilot, X)

expresses the predicate 'controls' to represent the action relationship between two terms; the pilot performing the act of controlling something to be determined in the future.

Additionally, variables can be qualified by two quantifiers. The universal quantifier, $\forall$ X, states that "for all X, it is true that....". For example:

$$\forall X, [pilot(X) \rightarrow controls (X, f\text{-}15)]$$

states that for all X, it is true that the set of pilots represented by X are able to control an F-15 aircraft.

The existential quantifier, $\exists$, states that "there exists an X, such that....". For example:

$$\exists X, [pilot(X) \rightarrow controls (X, f\text{-}15)]$$

states that there exists an X, who is a pilot who can control the F-15 aircraft.

Reasoning involves thinking coherently and logically, and making inferences from known facts.

For example, if larger(X,Y), means Y is larger than X. Then

larger(747, 767)

and

larger(767, F-15) are true,

then through deductive reasoning, it can reason to determine that:

larger(747, F-15) is true.

Deduction is the most widely used reasoning method since it guarantees correct results if the knowledge-base axioms are correct. Given correct minor premises supported by correct evidence, deduction can deduce a provable major premise.

Abduction is defined as a method where the major premise is evident, but the minor premise is not and therefore the conclusion is only probable. Abduction does not guarantee correct results since it tries to explain things. It is a good method to use for diagnosis.

Induction is a demonstration in which the general validity of a premise is inferred from observing the validity of minor premises and stating if the minor premises are correct the major premise is correct. Its reasoning from particular facts or individual cases to a general conclusion. Induction does not guarantee proof, but it forms the basis of scientific discovery.

Making inferences entails the derivation of new facts from a set of facts. 'Modus ponens' states:

If the statements p and ($p \rightarrow q$) are known to be true, then we can infer that q is true.

The 'modus tollens' rule of inference states:

If ($p \rightarrow q$) is known to be true, and q is false, then p is false.

$$\begin{array}{l} p \rightarrow q \\ \underline{\sim q} \\ \sim p \end{array}$$

Rule-based Systems

Rule-based systems can be utilized for knowledge representation, reasoning, and inference. Rules in such systems have a natural ability to represent heuristic knowledge. IF-THEN rules are a natural format used by experts to express problem-solving knowledge in many types of domains. Rule-based systems tend to be easy to implement and understand. Rules can capture knowledge such as that in the following example about an aborted landing (Painter, 1994):

IF (flight segment = "Final Approach") AND (altitude << 50 feet over ground)
THEN new flight segment : "Landing"
IF (flight segment = "Final Approach") AND (recognized crew intent = "Missed Approach") new flight segment: "Missed Approach"

The construction of a rule-based system that constitutes a pilot-aiding system is a formidable task and needs be divided into three subtasks, namely 1) knowledge acquisition, 2) knowledge base organization, and 3) maintenance (Chin, 1993). Knowledge acquisition involves understanding the pilot's thinking processes, understanding the pilot's language (the English language is very ambiguous), and the translation of this information into the production rules. Maintenance is extremely critical since errors or inconsistencies in the rules of a knowledge base can be very damaging.

Primary reasoning mechanisms used for rule-based systems are 1) forward chaining (data driven), 2) backward chaining (goal driven), and 3) pattern matching.

Backward chaining starts with a hypothesis that requires proving, and from there, searches for the evidence that is needed to support the hypothesis. If this evidence exists then it can be concluded that the original hypothesis is also true. Thus, backward chaining is effectively working from the hypothesis back to the evidence. *Forward chaining* takes as inputs to the system all the available evidence and from this information lets the system deduce which hypothesis is true.

Pattern matching is the process of determining whether two objects have a 'similar" structure. The similarity metric used can be simple, such as requiring that the two be identical or more complex requiring that the two meet certain specific conditions.

Search

Search is a process of methodically exploring a problem space. In general, search methods are well suited for AI applications. The selection of a search methodology is based on what is most effective for a particular problem domain. Basic search methods that may be effective for aircraft automation include depth-first, breadth-first, best-first, hill climbing and branch and bound optimal search.

Depth-first search is a blind search since it always starts expansion of nodes in a single direction to the deepest level of the tree. Only when the search hits a dead end does the search go back and expand nodes at shallower levels. While depth-first search is easy to implement, it is inefficient because it commits systematically exploring in a single direction at a time before examining adjacent nodes. Breadth-first method, also a blind search, investigates all the nodes at a given level of the tree and if an answer is not found, proceeds to the next lower level.

The hill-climbing search method was developed from the technique of depth-first search with the addition of local measurements. The outcome of each direction of search is compared against an evaluation function at each node, base on knowledge built into the tree. The evaluation function, or heuristic, determines the next appropriate step taken. Hill-climbing technique can greatly improve search efficiency in certain domains.

Best-first search reasons forward from the "best" node, no matter where it is in the partially developed search tree. Best-first search requires additional knowledge at the node level. In the best-first search, an estimate is provided at each node of how far the node is from the solution. With this information, the shortest path is selected. The total path length to find a solution using this technique tend to be closer to the optimal path length than those used by pure depth-first or breadth-first search.

The branch and bound search, another directed search, searches forward from the least-cost partial path through the search tree. The technique is used during a search where there will be many dead-end paths in the search tree space. This system extends the shortest path by one level, creating as many new incomplete paths as there are branches, and these paths are added to the set from which the shortest is again chosen. This process is repeated until the solution is found along the optimal path.

2.2.2. Object Representation-based Methods

As good as logic-based systems are they are lacking in their ability to show "causal relationships" and associations among objects.

Object representation-based methods have graphical structures in the form of a network. The nodes of the network represent fact, objects or concepts, while the arcs of the network show relationships or associations among the nodes. This section describes the following representation and reasoning systems:

1. Associative network-based systems
2. Frame-based systems
3. Object-based systems
4. Model-based systems
5. Qualitative systems
6. Case-based systems.

Associative Network-Based Systems

Associative (semantic) networks provide an ability to represent structured knowledge about physical or conceptual objects more easily than rules. This knowledge-based representation was originally developed in the late 1960's by M. R. Quillian to support the work on natural language processing. The networks provide a way of representing facts and relationships in the network. Network programming functions provide an ability to reason. Associative networks are labeled, directed graph, which symbolizes the association between various concepts. Semantic relations such as: "is-a", "part-of", "connected-to", and "number-of" all show types of relationships between nodes.

Frame/Object-Based Systems

Frame/object-based systems are closely related to associative networks. The concept of frames was developed by M. Minsky (1975) and is used to represent concepts and the relations between concepts. Objects are similar to frames, except they are more universal since they support any type of general computational need and provide encapsulation and polymorphism. Frame/object based representations account for a system's ability to deal with new situations, either object or actions, which are encountered each day, by using existing knowledge of previous events, concepts, and situations. Frame/objects represent knowledge rather than rules. Frame/objects are the preferred knowledge representation scheme used in model-based and case-based reasoning.

Frames support arbitrary levels of nesting for any given slot. The slots in frames correspond to the attributes of an entity. On the graph below, there are different types of objects, each with their own set of properties represented in frames. Apart from using the property inheritance of the 'is a' and 'inst' links, much of the power of frame/object nets stems from the ability to attach procedures to frame slots. Frames are most useful in representing the properties of objects, individuals, and events. Dynamic properties can also be captured in the form of action frames and state change frames. An example of aircraft pilot aiding using frames is Figure 4 (Chin, 1993), below.

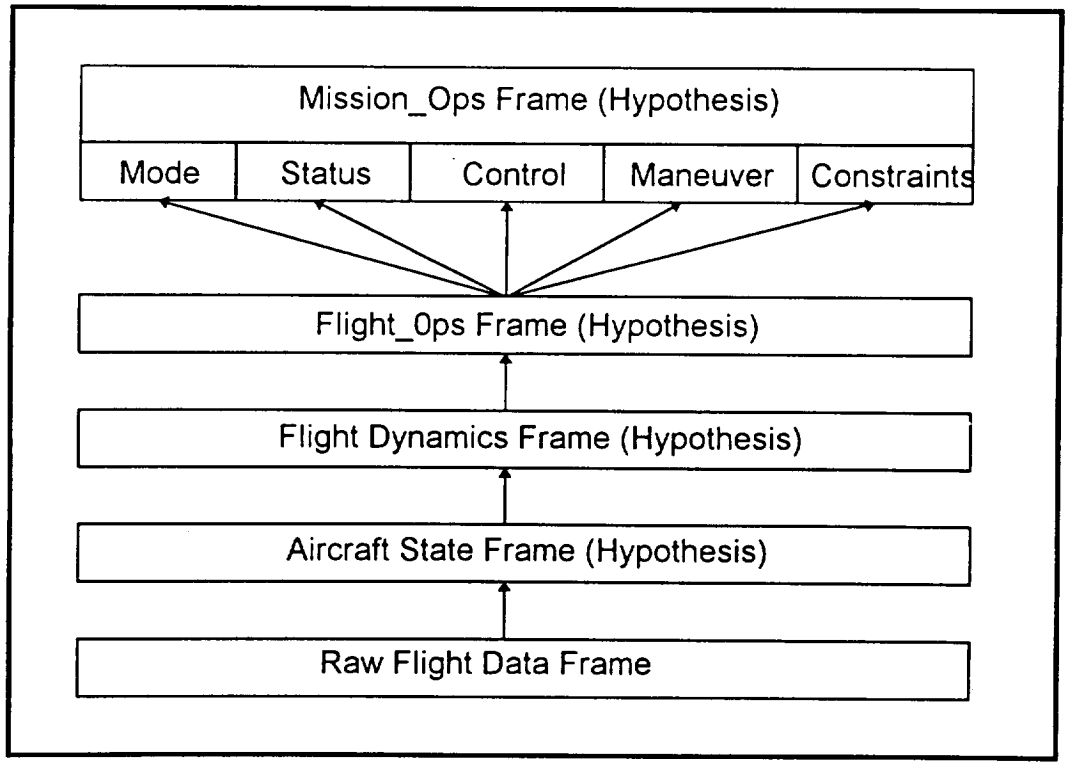

Figure 4 - An Inference Tree for Aircraft Pilot Aiding

Figure 4 shows the use of frames. The lowest level of information is the raw flight data that is derived from vehicle. Sensor information is converted the correct units and format for processing. The slots of the Aircraft State Frame are updated with the new sensor data from the Raw Flight Data Frame. Aircraft State Frame slots that contain procedures use the updated Raw Flight Data Frame data as input to their procedures.

Model/Qualitative/Case-based Systems

These three reasoning systems use a computer model n place of expert knowledge. These systems use frames, objects, and semantic network representations. Diagnostic systems have been developed using model-based reasoning.

Model-based reasoning represents physical systems by their structure and functionality. Qualitative reasoning is a reasoning system that qualitatively simulates the behavior of a physical system. Both eliminate the need for knowledge elicitation from the experts.

A case-based reasoning system consists of a library of historical cases, a means of using the key elements of the present problem to find and retrieve the most similar cases from the library, and a means for making modifications to the proposed solution when the case on which the solution is based is not identical with the current problem.

The ability to reason about the similarities of the current problem with the historical cases is called classification case-based reasoning. In classification, the system establishes whether or not a new case should be treated like an existing case. In this category, the system should come up with the pros and cons of why a new case should or should not be treated like an existing case.

2.2.3 Distributed Methods

Distributed methods are a relatively new concept used to coordinate multiple reasoning components.

Blackboard Systems

Blackboard architectures provide a structure where several knowledge sources can have there own particular knowledge representation and reasoning method. The knowledge sources do not communicate with each other directly. The knowledge sources can only see the blackboard, which constitutes a global knowledge base. Once a knowledge source gets the activation permission from the blackboards control unit, it takes the input information from the blackboard, performs its own reasoning process, and returns the result to the blackboard.

Opportunistic Reasoning

An opportunistic reasoning system uses both backward and forward chaining, depending on the nature of the data and the degree of goal-orientation of the user. In this reasoning method, the backward and forward mechanisms, as well as the search mechanism, are included as a part of the knowledge base, and the developer selects the circumstances suitable for applying each mechanism.

2.2.4 Uncertainty Management Methods

In reality, human knowledge is for the most part inexact and uncertain. Facts and rules in knowledge-based systems can contain various shades of vagueness, imprecision, and errors. Various methods can accommodate this uncertainty. Bayesian and Fuzzy Logic methods are described.

Bayesian

All the knowledge representation methods that have been considered so far have been based on the assumption that the knowledge being represented is exact and certain. This is often not the case, we frequently need to reason with inexact or incomplete information. The simplest probabilistic reasoning is based on Bayes' Theorem. Suppose we have evidence (e), and a hypothesis (h_i). The probability of concluding the hypothesis (h_i), given the evidence (e), has the form of a conditional probability:

$$p(h_i | e),$$

defined by the ratio

$$p(h_i | e) = p(h_i \cap e) / p(e).$$

Having the following equalities for conditional probabilities:

$$p(h_i \cap e) = p(h_i | e) p(e) = p(e | h_i) p(h_i),$$

where $p(h_i \cap e)$ is the probability that h_i and e occur together. From this, we get the Bayes formula:

$$p(h_i | e) = p(e | h_i) p(h_i)/ p(e).$$

Assume that the system has a number of rules that have e in their antecedent, with different consequence $h_1, h_2,...h_n$. Then, the probability of the evidence e is the sum of the intersections of e with all possible hypotheses, as:

$$p(e) = \Sigma p(e \cap h_j) = \Sigma p(e | h_j) p(hj).$$

We can now replace the denominator of Bayes formula with either of the two above and determine the sum of the intersections. In Bayesian probability, $p(h_i)$ is called the prior probability, and $p(h_i | e)$ is the posterior probability. The prior is the belief in the truth of the hypothesis. The posterior is the revised belief, after observing the evidence of e as a fact in a case.

Using IF-THEN rules to represent knowledge, the rule would be shown with a degree of uncertainty as follows:

Rule:
IF E is true
THEN H can be concluded with probably p

With an aircraft automation example:

Rule:

IF (the light indicates the landing gear are down) is true
THEN (aircraft is in landing mode) p(0.95)

Assume there are two hypotheses: h_1 = the aircraft is in landing mode and h_2 = the aircraft is not in landing mode. The evidence that is used in both hypotheses is e = the light indicates the landing gear are down. The belief net for this example is shown in Figure 5. The prior probabilities are $p(h_1) = 0.10$ and $p(h_2) = 0.90$.

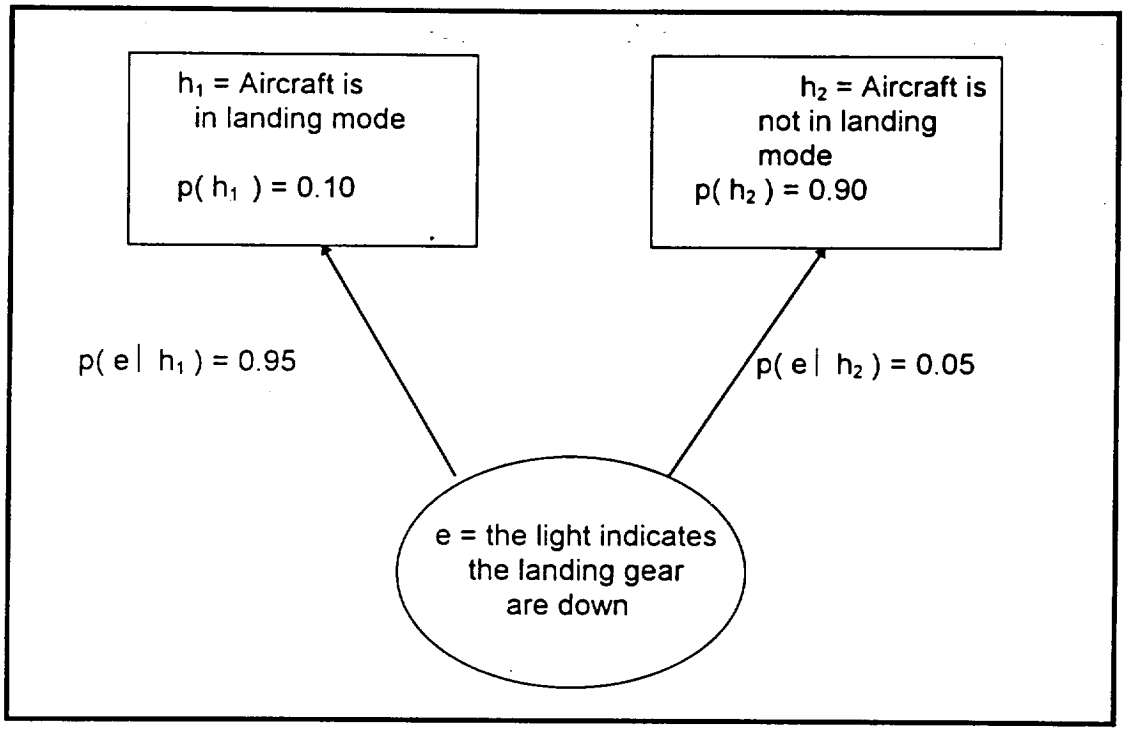

Figure 5 - Using Bayesian Approach

Applying the Bayes formula the posterior probabilities are:

$$p(h_1 \mid e) = \frac{p(e \mid h_1)\, p(h_1)}{p(e \mid h_1)\, p(h_1) + p(e \mid h_2)\, p(h_2)}$$

$$= \frac{(.95)(.10)}{(.95)(.10) + (.05)(.90)} = .68$$

$$p(h_2 \mid e) = .32$$

Fuzzy Logic

Fuzzy logic systems provide a mathematical tool for dealing with uncertainty and imprecision. This is done by using words in computing and reasoning as a way of implementing general concepts. Fuzzy systems store banks of fuzzy associations or common sense rules. Fuzzy logic systems reason with parallel associative inference. When given input, a fuzzy logic system fire each fuzzy rule in parallel, but to different degrees, to infer a conclusion or output. Neural and fuzzy logic systems naturally combine to evolve adaptive systems with sensory and cognitive components (Kosko, 1992). A block diagram of a fuzzy controller is shown in Figure 6.

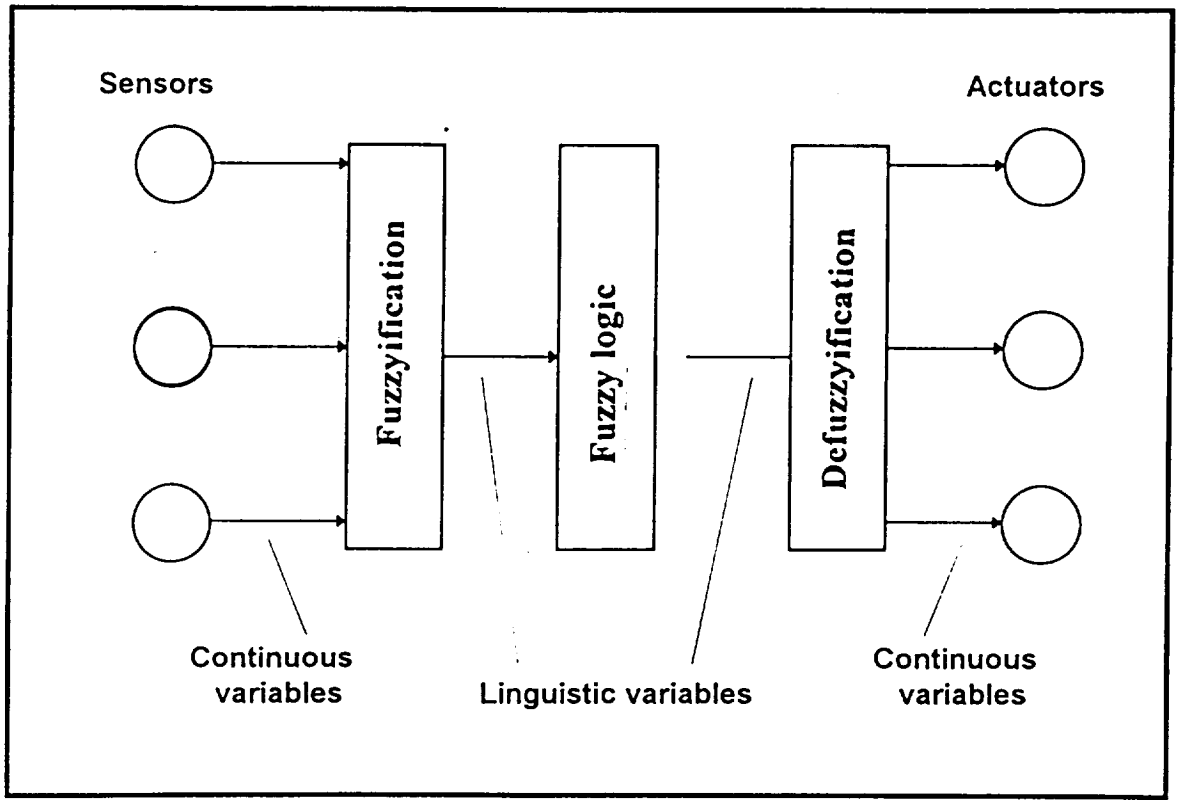

Figure 6 - A Fuzzy Controller

All sensor signals are converted to linguistic variables in a process called fuzzyification. This can be viewed as a quantization procedure. The quantified linguistic variable is then input to the fuzzy control law logic. The fuzzy control law logic, which can be a mixture of algorithms and/or rules, generates a control signal in the form of a linguistic variable. The linguistic variable is then mapped into a real number by an operation called defuzzyification.

2.2.5 Temporal Methods

Temporal methods deal with ability to reason about the time relationships between events. Time relationships are critical in aircraft automation especially in military and emergency context.

Temporal Reasoning

Many expert systems reason qualitatively about problems, but most of their reason is static analysis and involves a limited analysis of dynamic aspects of the problems because they lack a well-developed notion of time. In the early 1980s, an interval-based approach was developed (Allen, 1984) which provides a scheme of representing and manipulating time that is generally accepted basis of all temporal reasoning models,. In his approach Allen defines all actions or events as intervals of time having a nonzero duration. An interval is a segment of the time line bounded by two real numbers, (t_1, t_2) where $t_1 < t_2$. Time point t_1 is called the starting point of the interval and t_2 its end point. Allen also developed a method for describing relationships between two intervals.

2.2.6 Neurocomputing

Neurocomputing systems represent a very different computational paradigm to those previously discussed. The success and failure of the previously described AI methods is dependent on the ability to form some compact description of a task. As described previously, this may take the form of rules, for instance, which allow the manipulation of information in a governed way. However, there are many applications, such as speech recognition, adaptive system control, and the visual interpretation of images, where attempts to find compact descriptions have only had limited success (Werbos). The rationale behind the development of neurocomputing is to approach AI processing in a radically different and new way.

The fact that the human brain appears to handle speech recognition, adaptive system control, and the visual interpretation of images, with ease has led to a re-evaluation of the processing of these tasks. Neurocomputing systems estimate non-deterministic input-output functions. Training data determines the values of system internal weights. Unlike statistical estimators, neurocomputing systems estimate a function without a mathematical model of how outputs depend on inputs. Neurocomputing systems learn from numerical training data that represent input-output sets.

Neural Networks

A simple definition of a neural network is provided by Dr. Robert Hecht-Nielsen the inventor of one of the first neurocomputers, as: "...a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs"(Caudill, 1989). Neural networks are typically organized in layers. Layers are made up of a number of interconnected "nodes" which contain an "activation function". Training data or actual data are presented to the network via the "input layer", which communicates to one or more "hidden layers" where the actual processing is done via a system of weighted "connections". The hidden layers then link to an "output layer" where the answer is output as shown in the Figure 7.

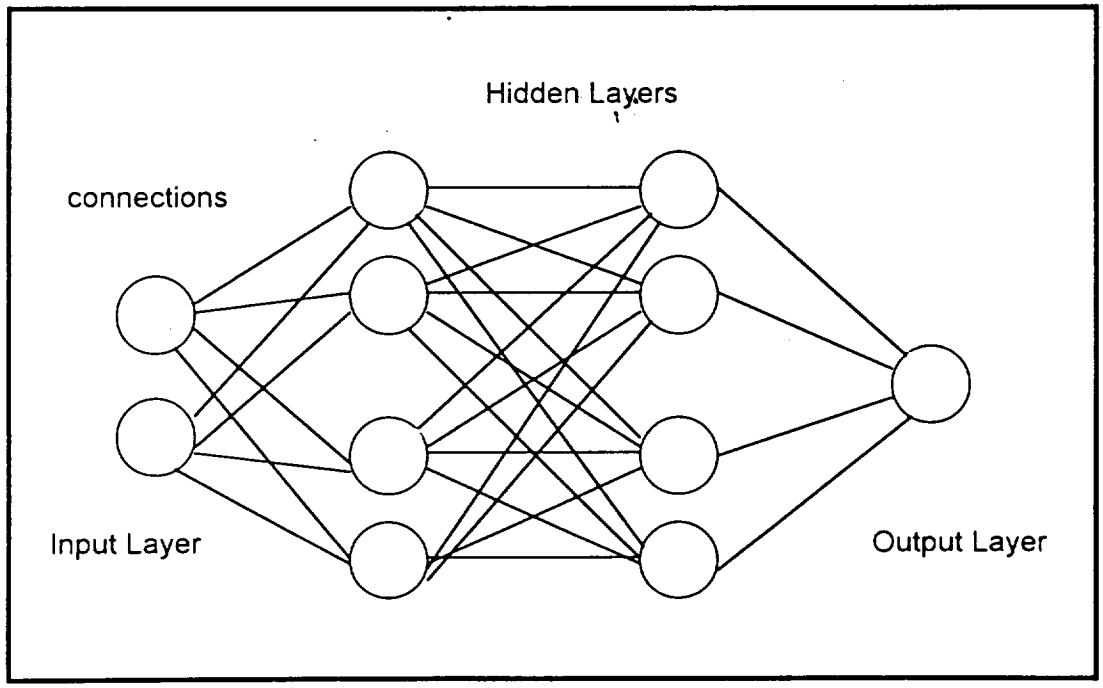

Figure 7- Basic Neural Network Architecture

Neural networks can be programmed or trained to store, recognize, and associatively retrieve patterns or database entries, to control ill-defined problems, or, in general, to estimate sampled functions when the form of the function is not known. The principal characteristics of neural networks are: 1) topology or mapping, which describes the number and characteristics of processing elements, the organization of the network into layers and the connections between layers, 2) learning, which illustrates how information is stored in the network and the training procedures, and 3) recall, which describes the method of retrieving the stored information."(Noor & Jorgenson, 1996). There are two types of network structure based on the connection pattern

(architecture). Neural nets can be grouped into feed-forward and recurrent (or feedback) networks. In a feed-forward network, links are unidirectional and there are no cycles. In a recurrent network, the links can form arbitrary topologies.

Neural networks used for control systems should be specifically designed to exploit learning behavior. A learning control system is one that has the ability to improve its future performance, based on experiential information it has gained in the past, through closed-loop interactions with the plant(aircraft) and its environment. Improving its performance indicates an autonomous quality. To improve its future performance, the system must operate in the context of an objective function.

2.3 Aircraft Automation Applications

This Section describes the aircraft automation tasks listed in the first column of the matrix.

2.3.1 Pilot-Vehicle Interface

The Pilot-Vehicle Interface (PVI) should be responsible for all communication between the aircraft and the pilot. This should include but is not limited to reading pilot's intention during his use of the throttle, stick, manual inputs to a touch screen or any switches, levelers, and valves in the cockpit, as well as pilot dialogue with the cockpit computer by voice. It should interpret and understand pilot actions in the context of pilot activities and mission events. The AI system could then respond by generating displays and setting control devices, feed the pilot as much information as he needs and is able to perceive at the given moment. The system also should detect pilot's errors, and adaptively aid him in the execution of his tasks to optimize his situation awareness and allow him to focus attention on important/critical events. The system could provide a display that meets the pilot's expectations in the specific situation and could be sensitive to the pilot's personal preferences and techniques.

The following paragraphs describe the various aspects of the pilot-vehicle interface.

Pilot Task Management

Pilot Task Management should be based on adaptive aiding and information management. Adaptive aiding optimizes information flow to match the given pilot's abilities to understand and process information and tasks, and provides assistance if appropriate. The system decides whether to provide the given information by voice, alarm or display, depending upon how busy the pilot is at the given moment, and his individual skills. The following are examples of pilot task management system available in the referenced literature. The paragraph numbers below match those the second column of Figure 1.

1. Task Support System (TSS), is a prototype avionics system designed to improve the information flow from the cockpit to the pilot and to reduce his manual and mental workloads. The TSS uses a rule-based representation and forward chaining reasoning, as explained in paragraph 2.2.1 (Funk, 1992).

2. Boeing's PVI Manager focuses on the integrated management of cockpit displays and low-level control tasks using a blackboard architecture, as explained in paragraph 2.2.3, (Shelnutt, 1989).

3. Texas Instruments developed a PVI in which the pilot can select from several types of operational pilot-PVI relationships or degrees of automation and autonomy. This system is a rule-based system, as explained in paragraph 2.2.1 (Krobusek, 1989).

4. In the Cockpit Assistant System (CASSY), object-based representation and reasoning are used to implement task management along with temporal reasoning as explained in paragraphs 2.2.2 and 2.2.3 respectively. The knowledge base started with production rules which were transformed into object-based representation constructs (Onken, 1995).

Mission Management

Mission management begins with on-ground mission preparation. Before flight, the pilot is required to input into the flight computer mission navigational parameters, pilot desired route waypoints and the characteristics of each waypoint. The mission management software then computes the pilot desired route and a number of alternative routes. This software also analyzes and presents to the pilot differences between the pilot-desired route and computer-recommended routes. Route differences can include improve flight performance, weather and hazard avoidance, decrease military risk, and decrease fuel consumption. Examples of mission management are:

1. The Pilot's Associate program is an intelligent co-pilot system that demonstrated the potential of integrating artificial intelligence into avionics systems. In the Pilot's Associate, the Mission Management function is divided into a Mission Planner module and a Tactical Planner module. The Mission Planner module monitors progress of the mission, evaluates the impact of route deviations, and replans the mission when changes occur. The Mission Planner module also generates paths to and from target areas that will minimize fuel consumption and risk. The Tactical Planner module sorts and prioritizes threats and targets, selects appropriate countermeasures, aids weapons deployment, and coordinates actions between aircraft flying in formation. It analyzes the current and predicted situations and provides the pilot with short-term offensive and defensive tactical options. Knowledge for both planner modules consists of specific constraints, actions, rules and deductions. Rule-based representation and reasoning is utilized, as explained in paragraph 2.2.1 (Chin, 1992).

2. Boeing's On-Board Mission Management System uses rule-based and heuristic methods to help pilots manage their missions. Detailed explanation of such methods is provided in 2.2.1. The system manages waypoints or mission navigation points, selects appropriate checklists to execute the current mode of flight, monitors aircraft systems health to evaluate current aircraft functional capabilities and assists the flight crew in enroute replanning if require by an aircraft system anomaly, bad weather or direction from Air Traffic Control (Wilber, G. F., 1989 & Wilber, R., 1989).

3. In the French study of cognitive modeling, a computer program that models human reasoning, a competence model, (a model of human reasoning sufficient to perform the task being investigated) was developed for mission management which used

over 100 goal-oriented procedures using declarative representations, as explained in paragraph 2.2.2. The goal-oriented procedures were either tactical procedures using mission/task completion knowledge or mission management procedures using knowledge about systems management (Amalberti, 1992).

Displays

The computer configures system displays based on the interpretation of past and present pilot actions and mission events. Displays could be a function of pilot expectations in specific situation and could be sensitive to the pilot's personal preferences.

Example of implemented system: [put also above]

1. Boeing's PVI Manager focuses on the integrated management of cockpit displays and using a blackboard architecture as explained in paragraph 2.2.3 (Shelnutt, 1989).

Pilot Intent/Error

Past implementations of automated systems that understand pilot's intentions (listed below) indicate that cockpit crew behavior should be modeled for normative and individual behavior. The normative model can be based on general aviation guidelines (e.g., landing, taxing procedures). For individual behavior modeling, knowledge about individual crewmembers can be learned by the on-line computer equipped with artificial intelligence software. The computer interprets pilot's actions and checks them for consistency with his mission goals, displaying warnings and alarms if any inconsistency is found. The following papers provide guidance on the implementation of such systems:

1. Research on an automated system that infers pilot's intentions, called OFMspert, was conducted at Georgia Tech, for potential use on commercial aircraft. A goal of the research was to understand how pilots select automatic modes to control aircraft operations. The system uses the blackboard model of problem solving, in which pilot functions, subfunctions, and tasks relevant to the current operating situation (such as take-off, cruise, communication with air traffic control, etc.)are posted to a blackboard data structure. Pilot functions, subfunctions and tasks are the actions the pilot takes necessary to accomplish the current overall flight mission and the sequence of individual steps required to accomplish any portion of the mission. The intent of the pilot is determined by the system based on various heuristic search methods. The search methods are describe in paragraph 2.2.1 and the blackboard methods are explained as a distributed method in section 2.2.3 (Mitchell, 1994).

2. In the German cockpit system, CASSY, the Pilot Intent and Error Recognition Module, identifies discrepancies as the difference between the model of appropriate pilot actions for the given flight plan and the actual actions. There are three possible reasons for such discrepancies: 1) pilot error -- the pilot deviates from the objectively correct expected behavior as derived by CASSY, 2) temporary discrepancy of pilot intent which is not acted upon by CASSY awaiting additional information, and 3) computer error which requires a restart of the computer system either by the pilot or CASSY (Onken, 1995).

3. For the Pilot's Associate, a model of the pilot's intentions is used to follow pilot's actions and reasoning at several levels of abstraction. The model of pilot's action is based on his cognitive model that includes his expectations and intentions. Since a cognitive model is a model of the pilots reasoning, the model includes the pilot's expectations and intentions for a variety of situations. For example, if the aircraft mode landing, then displays about the landing procedures are provided the pilot. The computer system provides the pilot with a feedback about his actions. The AI methods used are rule-based both representation and reasoning, which are described in paragraph 2.2.1 (Chin, 1992).

4. The inputs to the Pilot's Associate module are the numerous switch activation, stick movements, and engine status readings that constitute the operations of the aircraft. From these inputs, it is the job of the intent module to infer an understanding of the pilot's intent. The key to the system architecture is the plan-

goal graph, which describes the elements used to link the actions or intentions of the pilot with a particular mission goal. IF-THEN rules are used to represent knowledge in the graph, as explained in paragraph 2.2.1 (Rouse, 1990).

Pilot Dialogue

The interface between the computer co-pilot and the crew can be achieved through speech recognition, speech synthesis, and color graphic display. Normally, lengthy time periods are necessary for the pilot to look down and input information into the computer. Therefore, speech recognition should be chosen as the desirable input device. For the same reason, a speech synthesizer should be chosen as the desired output device. The color display should be used for long messages that, perhaps, require repetitive readings.

1. The CASSY system uses speech recognition, speech synthesis, and color display to interface with the crew. The used phraseology for speech output is based on the rules of radio communication. If the message is too complex and/or too long, the color display is used. Speech recognition is used as an input device. Knowledge is represented and reasoned using a frame-based system as described in section 2.2.1 for speech synthesis. Neural networks are used for speech recognition. Neural networks are described in section 2.2.6 (Onken, 1995).

2.3.2 System Status/Diagnosis

The System Status module should be used to monitor and analyze on-board systems to determine the current aircraft state and evaluate current system capabilities. Any detected aircraft malfunction should be evaluated to determine the degree of degradation of the overall system capability and assess the impact of the degradation on the mission plans. Where possible, the system should generate remedial plans. The following example is developed by Lockheed in the Pilot's Associate:

1. From the Pilot's Associate, the Diagnosis Function of the System Status module is a fusion of statistical fault detection techniques with AI techniques including rule-based logic (paragraph 2.2.1), blackboards (paragraph 2.2.3) and model-based reasoning (paragraph 2.2.2), (Ball, 1992). The diagnostic process has two phases: fault detection and fault isolation. Fault detection does not utilize AI. Fault isolation, which employees AI, consists of three functions: the Hypothesis Generator, the Hypothesis Evaluator, and the Hypothesis Tester.

   The Hypothesis Generator function begins fault isolation by creating a list of hypotheses or suspected faults that could explain the abnormality noticed by the Fault Monitor. In his 1992 paper, Ball writes that reasoning with uncertainty had not been developed at that time, because the focus of the development was a model-based approach. Previous model-based approaches included mathematical and classifications methods. In the mathematical method, the fault space is represented as a matrix of component states and influence coefficients, and the process of hypothesis generation is formulated as an optimization problem. Classification methods for hypothesis generation use various representations of the fault space, including IF-THEN rules, causal nets, and cause-effect sets. If the fault space representation uses IF-THEN rules, then breadth-first, depth-first, or best-first search strategies are used. If the representation uses causal-effect sets, the fault space is tractable when minimal solutions are considered first. The technique used is model-based reasoning which allows implicit modeling of the system.

   In the Hypothesis Evaluator function, the suspected faults are confirmed or ruled out by the use of a backward-chaining (goal driven) classification process. If it does not have sufficient information, the Hypothesis Evaluator calls the Hypothesis Tester function to run a test and generate data to analyze.

2.3.3 Situation Assessment

The Situation Assessment subsystem should monitor events external to the aircraft. It combines stored mission data with data from the aircraft sensors and other cooperative sources to provide context sensitive information to the pilot. Finding consistent, correct, and meaningful information is of paramount importance. The following are examples of situation assessment subsystems:

1. The Situation Assessment subsystem of the Pilot's Associate combines a need to react to nominal and unexpected objects and events. This requires the subsystem to reason about the situation in a data-driven way and focus attention on the data that is relevant to the current mission plans. The Situation Assessment function includes methods for reasoning with uncertainty, described in paragraph 2.2.4, allowing preliminary conclusions to be drawn based on imperfect or suspect data, and then be confirmed as contributing data is available and more inferences become certain (Chin, 1992).

2. The Situation Assessment subsystem of CASSY focuses on improving the pilot's situation awareness as to flight plan executability and is responsible for autonomous activation of replanning processes. If a conflict is detected, the crew is informed and the relevant replanning processes activated. The subsystem is implemented using object-based representation as explained in paragraph 2.2.2 (Onken, 1995).

2.3.4 Automatic Flight Planning

The Automatic Flight Planner should create and maintain a takeoff to landing mission profile, including routes, resources and time constraints. This system should be able to manipulate aircraft control surfaces and the propulsion to lead the aircraft along a predetermined path. Beginning with the start and end points and available fuel and other resources, this subsystem should generate all flight profiles, including route, space-time description, resource consumption, and resource consumption budgets. For fighter aircraft, a tactics planner should be developed to provide minimum risk trajectories to support both offensive and defensive tactical plans.

1. For the CASSY co-pilot, the inputs for route planning are the actual position, the destination, the generated altitude, and planning mode. All of these inputs are fed in the computer by the crew. The route planning module uses different search algorithms. A simple best-first search is used for very time critical planning tasks. A modified A* algorithm is used for rerouting in a very large airspace. The knowledge representation used is object-based representation as explained in paragraph 2.2.2 (Onken, 1995).

2.3.5 Aircraft Flight Control

Until recently, analytical algorithms were the sole basis of flight control systems. Now, neural networks and fuzzy systems, as described in paragraph 2.2.6, are proposed as superior substitutes in dealing with the non-linearities of aircraft flight control laws. These new systems may be used on the ground to provide computational techniques that aid in the creation of flight control laws. They may be used during flight tests and simulation to improve control law development. They may be use in operation to augment conventional flight control systems. The following examples provide a view of current research in the use of neural network to augment aircraft flight control:

1. Neural networks and fuzzy logic systems are computer architectures loosely based on the way the human brain processes information. There are many different types of neural networks, but common among them is the use of many simple processors in parallel. The primary use of neural networks and fuzzy logic systems in control problems is to synthesize a static or dynamic mapping from some subset of controller inputs to controller outputs. An example of this is the mapping that specifies actuator position as a function of sensor inputs. Neural networks have the ability to learn directly from data or learn on-line by altering or synthesizing non-linear mappings. Major categories of application for neural networks and fuzzy logic systems are aircraft flight control design tools, flight control augmentation systems, outer-loop mode controls, and replacing software components of the flight control system (Steinberg, 1992). The five basic neural network controller designs are: supervised control, direct inverse control, neural adaptive control, backpropagation-through-time (BTT), and adaptive critic methods (Werbos, 1990). With supervised control, learning is a simple case of direct teaching; the system will not be required to evolve. Direct inverse control is the same as supervised control. Neural adaptive control adapts system parameters to real time unexpected changes such as the loss of a section of an aircraft wing, which was totally unanticipated in design of the aircraft or the design and training of the neural network. "Backpropagation through time is used to solve problems of optimization over time because: 1) the user or designer is allowed to pick any utility function, performance measure, or cost function to maximize or minimize; 2) the method accounts precisely for the impact of present actions on future utility." (Werbos, 1990) Adaptive critic designs, like BTT, are capable of maximizing any utility function or measure of reinforcement over time, but they are approximations.

2. Currently being developed at NASA Ames Research Center is a neural network that rescues severely damaged aircraft by instantaneously relearning to fly the damaged aircraft (Jorgensen, 1996). Called Intelligent Aircraft Control, the objective of this project is to demonstrate a "flight control concept that can identify aircraft stability and control characteristics using neural networks, and then to utilize this information to optimize aircraft performance in nominal, off-nominal and simulated failure conditions"(Totah, 1996). A pre-trained multi-layer perceptron neural network with a hybrid Levenberg-Marquardt solution technique was employed to model the aerodynamic stability derivatives. The network consisted of two inputs, Mach and altitude, 20 processing elements in a single hidden layer, and 32 outputs corresponding to each aerodynamic stability derivative. Results "conclude that this

controller exhibits both stable and robust adaptive characteristics when subjected to mild and extreme changes in the aircraft aerodynamics."

3. In 1990, a paper was published addressing the use of neural networks to assist in the control aircraft autolanding problems using adaptive control (Jorgenson & Schley, 1990). The paper presents the mathematical models of a standard controller, a reduced complexity model of the airframe upon which the controller would act, and an environmental model. After the models were created, a series of experiments were conducted using variations on neural network architectures, training rules, and performance criteria. As a result of the experiments, it was observed that there was a necessity for refinements in a number of neural network concepts when faced with a complex application. Among them were the ability of a neural network to learn discontinuities, the importance of and the potential risks associated with grouping training data around discontinuities, the vital need for fast convergence methods, and the advantages of incorporating *a priori* knowledge into a network to facilitate convergence.

4. Recently, a research paper was published, titled "Linear and Neural Network Feedback for Flight Control Decoupling" (White, 1992). The paper proposes to develop a hybrid control system using an embedded neural network to decouple an aircraft nonlinear control system.

5. The research and use of fuzzy control for aircraft flaps is shown to be useful. Research was performed at Texas A&M University, supported by a NASA Training Grant. The test-bed was a piece-wise, linear, longitudinal simulation of the Boeing 737-100 (Painter, 1994).

3.0 Verification and Validation Approach for a Neural-Based Flight Controller

This section presents a verification and validation approach for the evaluation of a neural-based flight controller developed at NASA Ames Research Center and scheduled for flight demonstration on the F-15 ACTIVE aircraft at NASA Dryden Flight Research Center. The basic question addressed is "How can we trust this black box with the life of a pilot and a multi-million dollar aircraft?" As neural networks gain wider acceptance and are being used in the implementation of safety critical systems, a comprehensive approach for system verification and validation is necessary. Since neural networks are created using a very different paradigm from conventional software, a new framework must be developed. This new framework includes all conventional V&V activities, but tailored for neural networks (Suski, 1994).

This section begins with an introduction to neural networks, and control systems. A background is provided by describing the development and resultant neural-based flight controller. Next, an evaluation of traditional verification and validation processes for conventional safety critical software and their applicability to the testing neural networks is analyzed. As a result of this analysis, a revised, general method for the V&V of neural-networks is proposed. Lastly, a specific verification and validation approach for the subject F-15 Neural-based Flight Controller is described.

3.1 Background

The automatic control of aircraft is a straightforward task in a completely predictable environment with a perfectly functioning aircraft, but when an unplanned event occurs the problem can become very complex. Until recently, the pilot-vehicle interface of the aircraft was truly the domain of the human pilot. In recent years, the aerospace industry has developed knowledge-based copilot systems to provide assistance to pilots in the form of warning and advice, and when the pilot is overloaded, provides direct assistance in the execution of a variety of tasks. The newest research in this area is the use of adaptive neural networks to completely control aircraft. This introduction provides: 1) a definition of the term neural network, 2) a simple example of a neural network, and 3) an explanation of a generic adaptive flight control system.

3.1.1 A Simple Neural Network

A definition of the term neural network was given in the 1988 DARPA study:

> a neural network is a system composed of many processing elements operating in parallel whose function is determined by network structure, connection strengths, and the processing performed at computing elements or nodes. ... Neural network architectures are inspired by the architecture of biological nervous systems, which use many simple processing elements operating in parallel to obtain high computational rates (p. 60).

There are two basic types of neural networks, feed-forward and recurrent. In a feed-forward network, the links are unidirectional and there are no cycles. Basically, the network processes the current inputs, produces an output and waits for the next set of inputs. In a recurrent network, the links can form arbitrary topologies (Russell, 1995). Most control systems use feed-forward topologies. To understand the detailed computations of neural networks, Figure 8 is presented which shows the topology of a very simple feed-forward neural network.

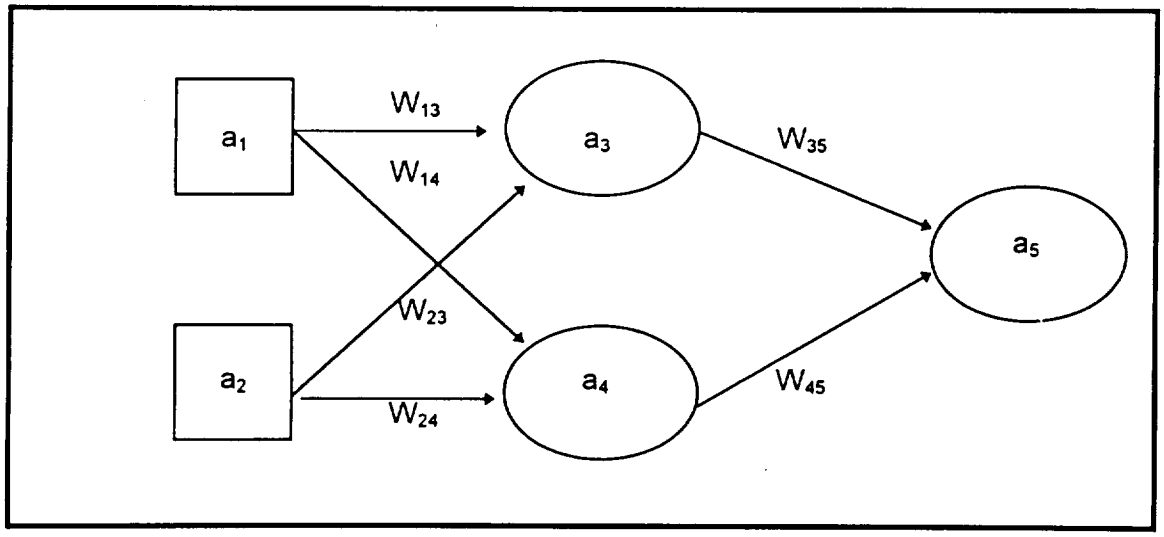

Figure 8 - Simple Forward-Feed Neural Network

The inputs are a_1 and a_2 , the hidden processes are a_3 and a_4 , and the output is a_5. This forward-feed network processes the following output function:

$$a_5 = g(W_{3,5}\, g(W_{1,3}\, a_1 + W_{2,3}\, a_2) + W_{4,5}\, g(W_{1,4}\, a_1 + W_{2,4}\, a_2))$$

where g is the nonlinear activation function. Unless the threshold value of a_3 is exceeded, stimulated enough to fire, the ($W_{1,4}\, a_1 + W_{2,4}\, a_2$) portion of the equation will not contribute to the output a_5 .

3.1.2 Control Systems

The problem of controlling a plant using a controller is shown in Figure 9. The plant and controller are in a feed-forward configuration.

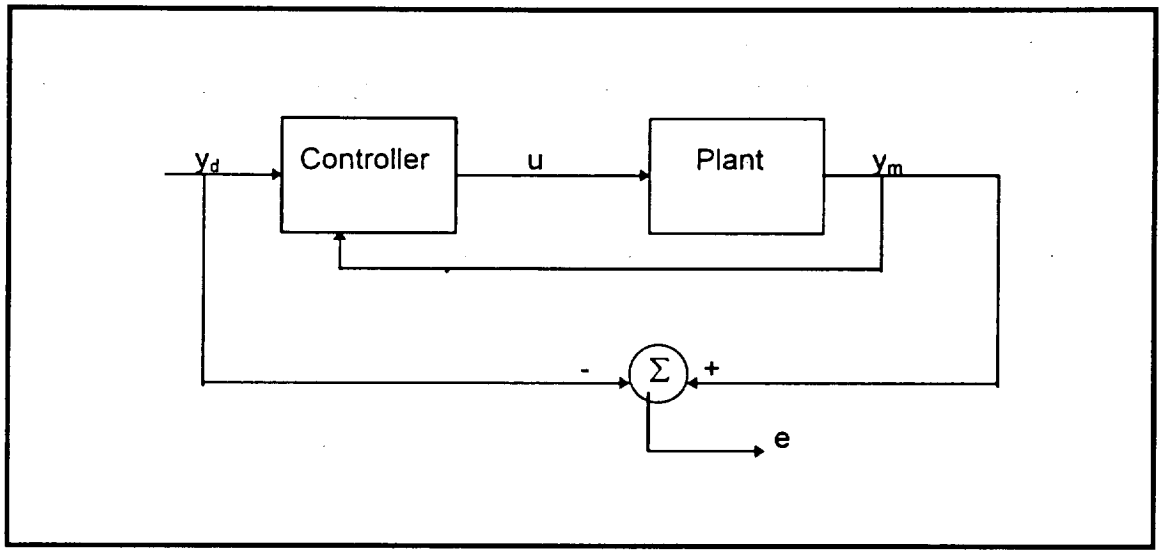

Figure 9 - Feed-forward Controller

The basic system control design problem is to find an appropriate control function from the measured plant output, y_m and the desired plant output, y_d to derive a satisfactory control action u.

3.1.3 Direct Adaptive/Learning Flight Controller

The motion of aircraft is usually expressed as a set of nonlinear dynamic equations. In the past, control of nonlinear systems using feedback linearization techniques required very detailed knowledge of the nonlinear plant dynamics, which in turn was a computationally intense effort for real-time onboard computers. Neural networks overcome these difficulties through the use of limited knowledge of the nonlinear plant dynamics and a combination of off-line and on-line, learning. The basic control system architecture of what can become, a neural-based adaptive/learning flight control system is shown in Figure 10.

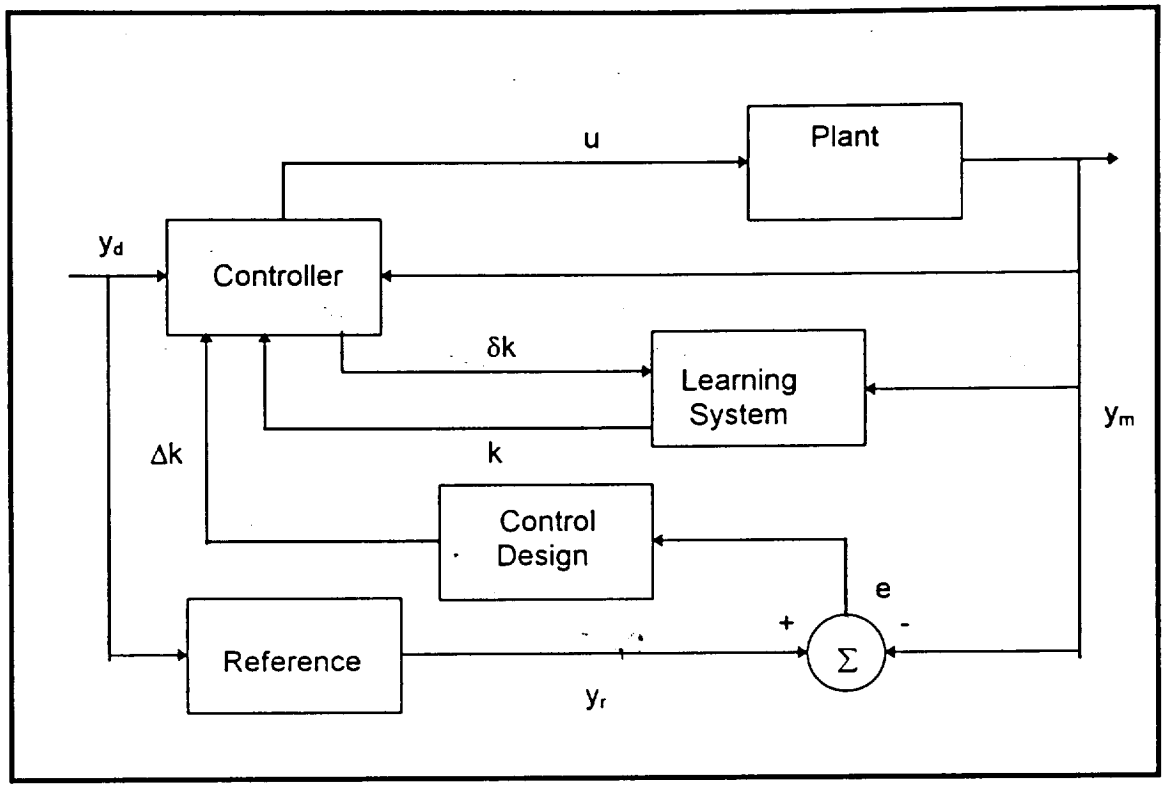

Figure 10 - Direct Adaptive/Learning Control System

The system control design problem is to find an appropriate functional mapping from the measured plant outputs and the desired plant outputs, to a control action that will produce satisfactory behavior in the closed-loop system. The controller in Figure 10 has the added capability of learning, which provides it the ability to improve its performance based on performance feedback and an objective function contained in the learning system. The controller in the above figure has four inputs and two outputs. The controller inputs are the measured plant outputs y_m, the desired plant outputs y_d, estimates of the control law parameters k, which are output from the learning system, and a correction factor Δk for the current control system parameters. The controller outputs are the resultant control action u and the signal to the learning system that is the perturbation in the control parameters δk to be associated with the previous operating cycle. The estimates of the control law parameters are adjusted based on the error e, which is equal to the sum of the measured plant outputs and the outputs of a reference system y_r. The reference represents the desired behavior for the augmented plant . Learning augmentation is accomplished by using the learning system to store the required control system parameters as a function of the operating condition of the plant. The system provides the required perturbation to the control parameter k generated by the learning system.

3.1.4 Direct Adaptive Tracking Control Architecture

The concept of a neural-based flight controller is based on the work being accomplished on the Advanced Concepts Program Research Project. It is lead by Dr. Charles Jorgensen at NASA's Ames Research Center, with flight tests underway at NASA's Dryden Flight Research Center using a modified F-15 jet fighter (Totah, 1996).

The direct adaptive tracking control architecture developed by Kim and Calise is the basis for the NASA flight controller described in Figure 11 below. In their paper (Kim, 1994), the development of a direct adaptive tracking controller using a neural network for an F-18 aircraft model is presented. The neural network presented overcomes difficulties that conventional controllers exhibit. The dilemma of limited knowledge of nonlinear plant dynamics is mastered through the ability to train the neural network both off-line and on-line in flight learning. The second difficulty is overcoming the processing speed required to perform the model inversion necessary to implement the feedback linearization. This problem is solved since the developed neural network performs its calculations using massive parallel processing.

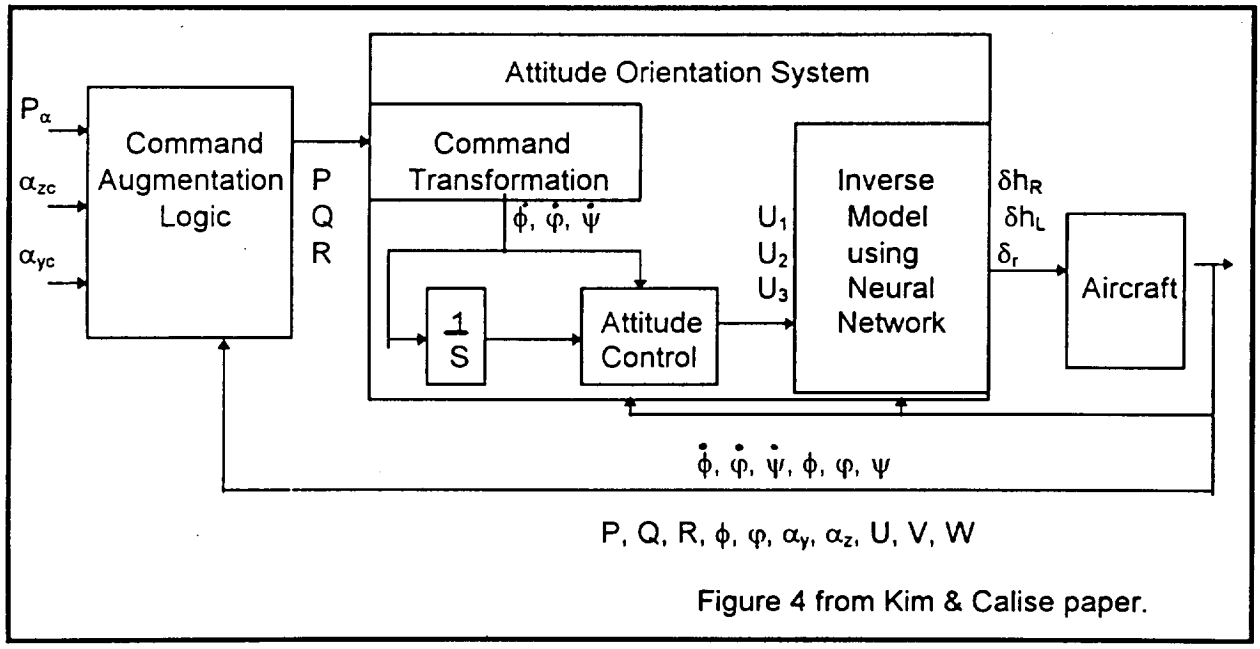

Figure 11 - Command Augmentation System

The command augmentation system has both a command augmentation logic and an attitude orientation system. The command augmentation logic is the outer loop function which tracks pilot commands. The attitude orientation system is an airframe stabilization system that accepts rate commands. The output of the attitude orientation system is transformed into commanded aircraft body axis rotational accelerations, which are used in an inverted model to calculate control surface deflections. The key purpose of the neural network is to approximately invert the vehicle attitude dynamics.

By inversion, it is meant that given the desired angular accelerations, U_i , determine the desired control surface deflection.

To facilitate a representation of a mapping that is less sensitive to altitude and Mach number variations, the neural network is design to represent the inverse mapping of plant dynamics from moment coefficients to control deflections, instead of from angular accelerations to control deflections. This reduces network complexity and the effort required in training. The results of the neural network in Figure 11 are the values for effective rudder deflection δr, effective elevator deflection $\delta e = (\delta h_L + \delta h_R)/2$ and the differential tail deflection $\delta t = (\delta h_L - \delta h_R)/2$.

3.1.5 Neural-based F-15 Adaptive Flight Controller

The control laws defining pilot input $U_{1,2,3}$ to the control surface deflections δh_L, δh_R, and δr and the design philosophy of Kim and Calise are the basis of the NASA neural-based F-15 flight controller. NASA modified the neural network to model aerodynamic stability derivatives rather than total aerodynamic coefficients. The control architecture employs a single pre-trained neural network to represent the nonlinear aircraft aerodynamics in the model inversion portion of the controller. The aircraft model used is the F-15 Advanced Control Technology for Integrated Vehicles (ACTIVE) aircraft. The inversion performed is expressed in equations T4 from Totah's paper provided in Appendix A of this report. The neural network estimates of the aerodynamic stability derivatives are denoted by an overstrike such as $\overline{C_{z\alpha}}$ in the T4 equations.

In Totah's F-15 paper, the network structure consisted of two inputs Altitude and Mach, 20 processing elements in a single hidden layer, and 32 outputs corresponding to each of the aerodynamic stability derivatives. The aerodynamic database consisted of an entire set of coefficients for sets of Mach and altitude ranging from $0.3<M<1.2$ at sea level, to $0.6<M<2.0$ at 50,000 feet.

3.2 Comparing Conventional Software and Neural Network Paradigms

The goal of this section is to evaluate which portions of the NASA V&V standard can be augmented to include the evaluation of neural networks. There are a number of standards for the formal testing of safety critical software. These formal test standards are expressed under the subject of Verification and Validation (V&V). The Department of Defense has its standard expressed in DoD-MIL-STD-2168, titled "Defense System Software Quality Program." The IEEE standard is ANSI/IEEE Standard 1012-1986, titled "IEEE Standard for Software Verification and Validation." The NASA V&V standard is provided in Appendix B of this report.

Since software systems control the any safety critical systems, formal software evaluation has been an important topic for both public and private enterprises. These software evaluation standards are well respected for there ability to field high quality software.

The NASA standard is composed of the following sections:

1. Concepts and Definitions
2. Activities
3. V&V During the Software Acquisition Life Cycle
4. Independent Verification and Validation
5. Tools and Techniques.

3.2.1 Concepts and Definitions

The entire Concepts and Definitions section of the NASA V&V standard is as follows:

> "Software Verification and Validation (V&V) is the process of ensuring that software being developed or changed will satisfy functional and other requirements (validation) and each step in the process of building the software yields the right products (verification). The differences between verification and validation are unimportant except to the theorist; practitioners use the term V&V to refer to all of the activities that are aimed at making sure the software will function as required.
>
> V&V is intended to be a systematic and technical evaluation of software and associated products of the development and maintenance processes. Reviews and tests are done at the end of each phase of the development process to ensure software requirements are complete and testable and that design, code, documentation, and data satisfy those requirements ".

At this level, these definitions would apply to neural networks. Unlike conventional software design and development which is based on an iterative procedure of requirements definition, analysis, specification, implementation and testing, neural

network based systems rely on training to formulate the control mechanism. Verification and validation of such systems for safety properties is extremely hard due to the lack of a complete system model. Because of this difficulty and a lack of evaluation framework, some neural network developers feel that they are not required to comply with the formal systematic and technical V&V evaluation. It is believed that as neural network become used more and more for safety critical system implementation, this attitude must change. Also, by providing this formal neural network V&V framework, the old beliefs will change.

3.2.2 Activities

The next section of the NASA V&V standard is Activities, which involve testing and reviews. Presentation of the neural network should be accomplished as a subset of the total system reviews. Evidence required by these reviews will only be available in the form of top level requirements. Neural networks are treated somewhat as a black box with little internal details, thus code inspection is less effective.

The NASA standard provides the accepted definition of testing:

> "Testing is the operation of the software with real or simulated inputs to demonstrate that a product satisfies its requirements and, if it does not, to identify the specific differences between expected and actual results. There are varied levels of software tests, ranging from unit or element testing through integration testing and performance testing, up to software system and acceptance tests".

Neural networks can be tested according to this definition. Black box testing is accomplished by generating test input data, executing the software, and observing the results. As the standard states, the objectives of these tests are: 1) computational correctness, 2) testing of boundary and extreme conditions, 3) state transition, 4) stress testing, and 5) adequate error detection, handling, and recovery. These are clear objectives for testing neural networks. Computational correctness can be evaluated with respect to the desired output given a particular input. Statistical analysis can be performed to evaluate performance characteristics, boundary conditions that produce greater error, and degradation of performance. Goals of this analysis are to determine robustness of the system and does it still produce meaningful results at boundary conditions.

White box testing is not applicable for a neural network. In white box testing the internal workings of the code are examined and tested. The neural network software only provides the network structure and the threshold algorithms. The system is not testable until it is trained.

Next, the NASA standard covers acquirer-approved/formal testing. At this level of testing, the neural network is often embedded into the system. As a result, there shouldn't be special consideration.

3.2.3 Verification and Validation During the Software Acquisition Life Cycle

The NASA standard lists eight phases for the software acquisition life cycle. In each phase there are V&V activities.

Software Concept and Initiation Phase

Goals of the phase are the same for conventional software and neural networks, but the results will differ. The major goal is "to develop a concept of how the system is to be reviewed and tested." Neural networks require different test cases, simulators, and other test capabilities. The creation of training and testing data for a neural network is a time consuming and costly process. It is extremely important to develop a sound V&V concept since the complexities and unfamiliarities of neural networks can drive cost and schedule overruns.

Software Requirements Phase

A preliminary version of an Acceptance Test plan should be developed. The development of test beds, training data, and testing data started. Only top-level requirements can be expressed, since a system model of the actual neural network can not be expressed. It will be difficult to write testable requirements. The typical Verification Matrix will not be applicable because of lack of system decomposition.

Software Architectural & Detailed Design Phase

The neural network developers must employ a set of evaluation tools to determine the effectiveness of the neural network architecture, and evaluate training and test data. These evaluation tools are statistical tools or system evaluation tools as described in section 3.2.5. The V&V activity should include a review and understanding of these analyses, completion of the Acceptance Test Plan, and participation in the Preliminary and Critical Design Reviews. For the Software Architectural (Preliminary) and Detailed Design Phases the neural network can not be expressed like conventional software. The development of definitive specifications for any real-world software system using conventional methods is difficult. A neural network system poses further challenge due to the fact that almost nothing is assumed to be known about the contents of the blackbox. The neural network learns and discovers its architecture and parameters of the underlying system through training.

Software Implementation Phase

During this phase the V&V detailed test procedures are developed. These test procedures should include the statistical evaluation tools included in section 3.2.5. The traditional software development activities of code inspection and unit testing will be replaced by the neural network training program, the developers evaluation of the neural network, and resultant modifications. The development of detailed test procedures should be supplemented by the development of a subset of the training data to be used for V&V test data. Statistical analysis and boundary conditions evaluations generated by the neural network developers should be thoroughly understood in order to correctly select this V&V test training data.

Software Integration & Test; Software Acceptance & Delivery Phases

The major V&V test and evaluation efforts occur in the Software Integration and Test Phase and the Software Acceptance and Delivery Phase. These efforts adhere to the standard, except for the fact that the test procedures will specify the evaluation of the neural network.

Software Sustaining Engineering and Operations Phase

The Software Sustaining Engineering and Operations Phase is different for a neural network than conventional software. Conventional software is inert; it does not change as a function of time or use. If training continues during operation, neural networks can become over trained and brittle. Therefore it is necessary to develop a sustaining engineering program in the same manner that hardware is required to have a sustaining engineering program.

3.2.4 Independent Verification and Validation (IV&V)

The use of an IV&V agent is necessary as a function of the safety-critical nature and the complexity of the system. As there is an increase likelihood that the system will effect human safety, that the result of this system will effect other systems, or the complexity of the system increases, the neural network should be independently evaluated by an IV&V agent.

3.2.5 Tools and Techniques

Tools that evaluate neural networks can be classified into two groups: 1) tools that assist in the decisions for the development of the neural network system and 2) tools that evaluate the performance of the neural network. The goal is to decrease both system output error and system output error variability. The usefulness of neural networks have motivated the implementation of many systems. Characteristics such as fault and noise tolerance, and learning capability have attracted many engineers. Unfortunately, the uncertainty of neural network output has not been addressed in many implementations. In safety-critical systems, output error variability must be evaluated.

Tools for the evaluation of neural network development:

Design of Experiments

Design of experiments is an organized series of tests in which the objective of each of the individual tests is to determine how results will vary when a particular input is changed. The procedure in performing a design of experiments is the following: 1) define the problem, 2) define measurable objectives, 3) choose the independent variables or factors, 4) choose the levels of the factors, and 5) run the experiment and analyze the results. The goal of the entire experiment is to obtain information about what combination of design factors provides the best results. Design factors for a neural network are noise in the training data, the network architecture, the time in which the training is stopped and noise in the test data.

The Taguchi Method

The Taguchi method is another analysis procedure, like the design of experiments that has been used successfully in improving manufacturing products and processes. It has been used successfully to generally evaluate neural networks (Peterson, 1995). As a conclusion Peterson indicates that "since relatively little is known about the capabilities and characteristics of neural networks, controlled experiments under varying circumstances must be run since the outcome of a single experiment can be misleading". The following presents some of guidelines for the design and construction of neural networks resulting from this paper: 1) Use a training sample that is as free of noise as possible, 2) use dense training data, 3) for best error control, use networks with two hidden layers, 4) for best smoothness of the approximating function, use one hidden layer, and 5) stop training when the error on the testing data begins to rise.

Tools that evaluate the performance of the system:

Statistical Analysis

Statistics is used to evaluate the performance of neural networks. The following definitions are fundamental:

Accuracy -- the difference between the average of the values and the true value.

Range -- the difference between the high and low values:

$$R = X_h - X_l$$

where

X_h = the highest measurement, and X_l = the lowest measurement.

Mean -- the sum of all values divided by the number of values:

$$\bar{X} = (X_1 + X_2 + X_3 + \ldots X_n)/n$$

where

X_1, etc. = the value of each individual value
n = the number of values.

Standard deviation -- the square root of the sum of the squares of the differences between the individual values and the mean divided by the number of values.

$$s = \sqrt{\sum (X - \bar{X})^2 / (n - 1)}$$

where

X = the value of the individual value
$\bar{X}$ = the mean
n = the number of values.

Variance -- is the square of the standard deviation (s^2)

Mean squared error --the variance of the error. The mean square error is used for many data modeling techniques including neural network modeling. The use of the mean square error in data modeling is known as the least mean squares (LMS) method. It attempts to optimize the fit of a model with respect to the training data by minimizing the square of the residuals. This method is well suited for uniformed training data sets. Unfortunately, neural network training set commonly are non-uniformed. Using LMS, a small sub-set of gross errors or outliers can influence the resulting training set unfavorably and cause inaccuracies. As developed in the paper, " Robust Error Measure for Supervised Neural Network Learning with Outliers", (Liano, 1996) the mean log squared error (MLSE) can eliminate this inaccuracy.

Fault Tolerance

Fault tolerance refers to the fact that no matter how well a system is designed and tested, faults will remain in the delivered system. Neural networks must be designed to be fault tolerant if their application requires high reliability. Aircraft are subjected to extreme reliability requirements. The reliability requirement for a passenger aircraft is extremely high: "the probability of catastrophic failure during a 10-hour flight should be less than 10^{-9}/hour" (Rushby, 1993). Such high reliability requirements force the development of redundant systems since the hardware can not live up the such high standards. Triple redundancy with voting is proposed in the paper titled "Complete and Partial Fault Tolerance of Feedforward Neural Nets" (Phatak, 1995).

3.3 Verification and Validation Program for the Neural-based F-15 Adaptive Flight Controller

The evaluation approach described in this section will be tailored to the current development maturity of the subject test article: the neural-based F-15 adaptive flight controller engineered at Ames Research Center.

3.3.1 V & V During the Software Acquisition Life Cycle

It is assumed that the subject neural network is in the System Demonstration Phase of the System Acquisition Life Cycle. As shown in Figure 12, the subject system is beyond the Concept Phase, but not mature enough to enter the System Production Phase.

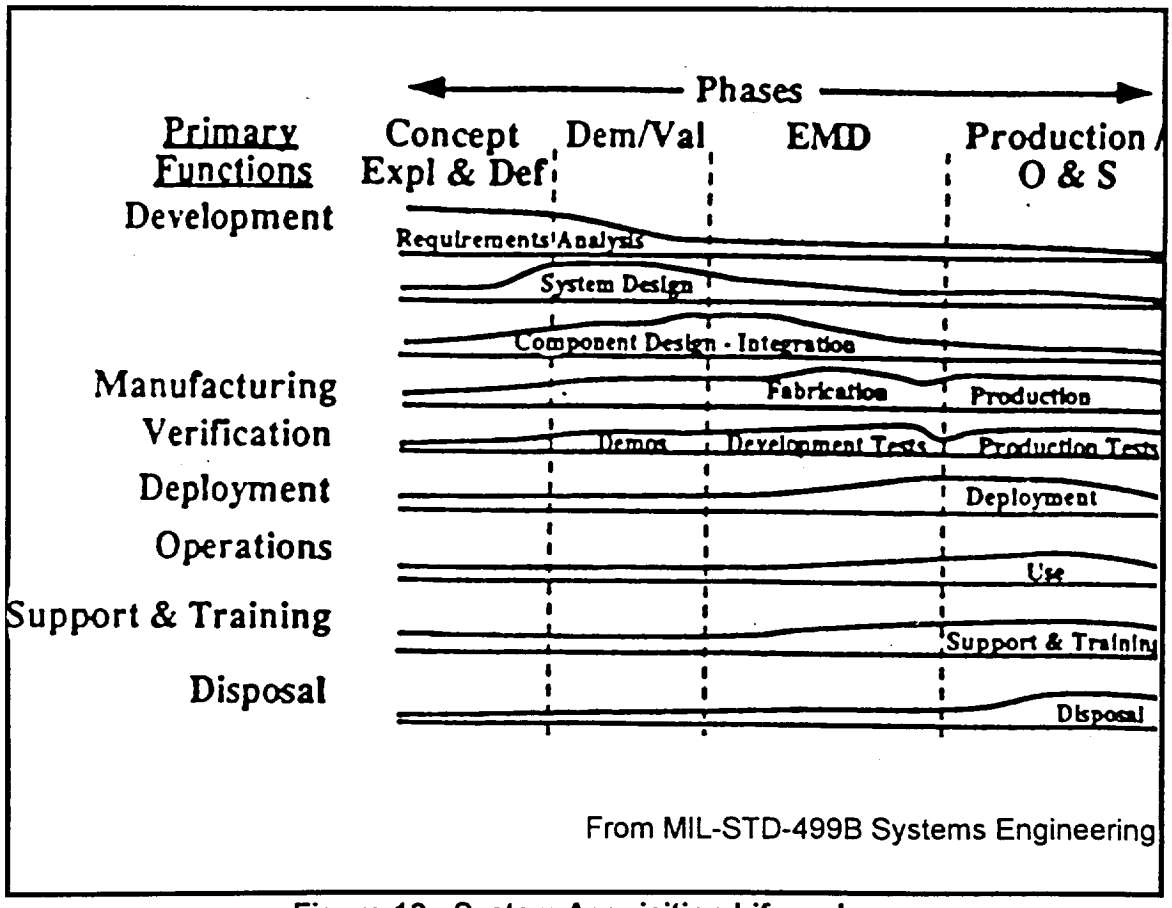

Figure 12 - System Acquisition Lifecycle

System development and evaluation efforts being taken are readying this system for the Production Phase. In the Production Phase, a contractor would incorporate this system into the avionics subsystem of a production line aircraft. For this reason it is very important that more formal system evaluation be conducted. More importantly,

during the upcoming flight tests, the system will have influence over the safety of the pilot and actual flight vehicle. So, as the neural-based adaptive flight controller and the F-15 avionics system is modified for flight worthiness, a more formal test effort is required.

Software Concept and Initiation Phase

Everything that is required for conventional software during this phase is applicable to neural networks. Both the F-15 avionics software that is changed to accommodate and interface with the neural network and the neural network require all aspects of this first phase. The first activity, the development of a "concept of how the system is to be reviewed and tested" is required. The evaluation to determine top level test cases, requirements of simulators or modification to existing simulators, and in the case of the neural network, test case data. This top-level evaluation activity also helps in the planning of an adequate V&V concept and plan and should be in sufficient detail to estimate cost, schedule, and test complexities. Since this neural network has been evaluated in a simulated test environment, important information from previous testing can assist in this top-level evaluation.

Software Requirements Phase

The V&V activities applicable are:

1) analyzing software requirements to determine if they are consistent with, and within the scope of, system requirements.
2) assuring that the requirements are testable and capable of being satisfied.
3) creating a preliminary version of the Acceptance Test Plan, including a verification matrix, which relates requirements to the tests used to demonstrate that requirements are satisfied.
4) beginning development of test beds and test data generators.
5) the phase-ending Software Requirements Review (SRR).

Both the F-15 avionics software that is changed to accommodate and interface with the neural network and the neural network software require all aspects of the Software Requirements Phase.

In addition, the V&V of the neural network will require further evaluation since it is partially non-deterministic software. In conventional software the basis of evaluation is a set of requirements. In conventional software, the requirements and behavior are both certain; a given input will result in a precisely specified output. For a neural network, there usually is a degree of uncertainty about what the output will result for a given input. The basis of evaluation for the neural network therefore should be a set of requirements, a set of constraints, and a set of goals that specify expected behavior. These set of requirements should be expressed in a system specification and reviewed at the SRR.

Software Architectural & Detailed Design Phase

During the design phases, PDR and CDR, the Acceptance Test Plan is expected to be completed and the system design will be expressed in both the system design specifications and the design reviews. During this period, the establishment and evaluation of training data should take place. It is very important that this training data represent the actual data the network will experience during flight demonstration. In conditions of rapid change, a greater volume of test cases are required.

Software Implementation Phase

During the Software Implementation Phase, the development of Detailed Test Procedures is required. Unit testing of the conventional F-15 interface software is required. The neural network training data is divided into two sets; one set for training (80%) and one set for testing (20%). During neural network training, potential problems are analyzed, such as, the identification of outliers, influential subsets and collinearities (Peterson 1993). Outliers are data points that are far removed from the rest of the data. Influential subsets are data points or subsets that could provide more than their share of influence on the outcome of the training process. Collinearities are situations in which the values of one feature are linear combinations of the values of other features. These problems will skew statistical evaluations.

Software Integration and Test Phase

During this phase, the actual Detailed Test Procedures are conducted, first using a simulate F-15 and then the actual F-15. The test cases should be identical, so correlation of output can be analyzed. Using a simulated F-15, tests of the F-15 interface software and the neural network are conducted. The Detailed Test Procedures are based on the system requirements and goals. The neural network tests should evaluate the following general characteristics:

1) safely -- there is no circumstance where the system produces output that could cause harm to the pilot or aircraft.
2) system error -- statistical estimation of error is estimated and acceptable.
3) sensitivity -- the output is properly sensitive to the inputs.
4) boundary -- the boundary of acceptable input is well understood and input is restricted within the boundary.
5) performance -- near the boundary of acceptable performance, the performance degradation is gradual.
6) robustness -- the system continues to provide useful results even when some inputs are missing or estimated.

4.0 Conclusions

The principal accomplishments of this project were 1) guidance in the selection of artificial intelligence methods to develop advance aircraft automation systems, and 2) a formal methodology to evaluate neural networks. Both topics are very current topics in the application of artificial intelligence in aircraft automation. The significant contribution of both accomplishments was greater use of the power of AI to address real world problems faced by engineers in the aerospace industry developing aircraft automation systems.

Guidance in the selection of AI methods to develop aircraft automation systems was expressed in the form of a relationship matrix. The matrix listed AI methods on the top and aircraft automation applications which were suitable candidates for AI implementation on the left. The matrix was a product of an extensive library search that was interdisciplinary in nature. Over one hundred journal articles, and ten books were read to assess AI and aircraft automation. Total quality management (TQM) methodology was performed to classify aircraft automation and AI methods. This matrix will be useful to a manager faced with the task of meeting systems requirements for the modification of an existing cockpit system or for the development of a new automated cockpit. The matrix will provide a starting point for an informed decision when selecting tools to implement aircraft automation.

The second problem addressed by this report was the development of a formal methodology to evaluate neural networks. This part of the report resulted from the initial feasibility report in which the evaluation of neural networks was a good candidate for further research. The approach of starting with the NASA software quality assurance policy and using the methods that are recommended was a starting point for the development of an approach to V&V neural networks. Neurocomputing has been slowly evolving as a mainstream tool through its utilization over the years. As system developers acquire more and more confidence in neural network technology, neural networks experience greater use. This comprehensive V&V approach provided the necessary evaluation methods and tools to use neural networks in safety critical systems.

This project accomplished the lifelong learning benefit of a interdisciplinary/ multidimensional project. This project allowed the author to study the fields of artificial intelligence, software engineering, engineering and production management, and aerospace engineering as they apply to aircraft automation. The multidimensional aspect was found in the need to work with the university resources and the NASA project personnel on real world problems. This was excellent preparation for the challenges that will be faced in the work place, with it's importance on integrated product development.

5.0 Future Recommendations

It is recommended to develop a NASA Web site to disseminate the matrix information contained in the Aircraft Applications vs. AI Methods Matrix. In addition, it is recommended that this Web site be used to encourage other aerospace managers to contribute information about their experience with AI applications. It is felt that even through the Lockheed Corp. had a contract to develop the Pilot's Associate, other companies like McDonald-Douglas, and Northrop-Grumman were also very active in the design and implementation of prototype systems. This added information would be valuable in enhancing the matrix with additional references.

The second recommendation is to develop an Acceptance Test Plan and Detailed Test Procedures for the evaluation of the neuro-based F-15 flight controller. Test and evaluation planning should address performance, functional and design requirements with appropriate quantitative criteria, test events, scenario descriptions, resource requirements, special test facilities, and test limitations. Test and evaluation efforts should provide an assessment of risks and verification of the technical performance objectives.

Appendix A: Derivation of the Aerodynamic Stability Equations

The following derivation is from J. J. Totah paper titled "An Examination of Aircraft Aerodynamic Estimation Using Neural Networks", SAE Technical Paper Series 952036, 1995

T1

$$P_c = \delta_{la} - p \quad (A.1)$$

$$Q_c = \left(\delta_{ln} - N_{Zc_\delta}\right)\frac{g\left(K_1 + \frac{\zeta_2}{S}\right)}{\left(U_0 + \right)} - q \quad (A.2)$$

$$R_c = \left(\delta_{pd} - N_{Yc_\delta}\right)\frac{g\left(K_3 + \frac{\zeta_4}{S}\right)}{\left(U_0 + \right)} - r \quad (A.3)$$

$$K_1 = t_n K_2 \quad (A.4)$$

$$K_3 = t_y K_4 \quad (A.5)$$

$$t_n = \frac{-mV_t}{\bar{q}S\bar{C}_{Z_\alpha}} \quad (A.6)$$

$$t_y = \frac{mV_t}{\bar{q}S\left|\bar{C}_{Y_\beta}\right|} \quad (A.7)$$

$$K_2 = K_4 = \frac{4.6}{t_{so}} \quad (A.8)$$

$$t_{so} = 2.4 \text{ sec} \quad (A.9)$$

$$\dot{\Phi}_c = P_c + \left(Q_c \sin\Phi + R_c \cos\Phi\right)\tan\Theta \quad (A.10)$$

$$\dot{\Theta}_c = Q_c \cos\Phi - R_c \sin\Phi \quad (A.11)$$

$$\dot{\Psi}_c = \frac{Q_c \sin\Phi + R_c \cos\Phi}{\tan\Theta} \quad (A.12)$$

T2

$$U_1 = K_{p\Phi}\left(\frac{\dot{\Phi}_c}{S} - \Phi\right) + K_{d\Phi}\left(\dot{\Phi}_c - \dot{\Phi}\right) \quad (A.13)$$

$$U_2 = K_{p\Theta}\left(\frac{\dot{\Theta}_c}{S} - \Theta\right) + K_{d\Theta}\left(\dot{\Theta}_c - \dot{\Theta}\right) \quad (A.14)$$

$$U_3 = K_{p\Psi}\left(\frac{\dot{\Psi}_c}{S} - \Psi\right) + K_{d\Psi}\left(\dot{\Psi}_c - \dot{\Psi}\right) \quad (A.15)$$

$$K_{p\Phi} = K_{p\Theta} = K_{p\Psi} = \frac{9.2}{t_{si}} \quad (A.16)$$

$$t_{si} = 0.8 \text{ sec} \quad (A.17)$$

$$K_{d\Phi} = K_{d\Theta} = K_{d\Psi} = \frac{K_{p\Phi}^2}{2} \quad (A.18)$$

T3

$$\dot{P}_c = U_1 - U_3 \sin\Phi - \dot{\Psi}\dot{\Theta}\cos\Phi \quad (A.19)$$

$$\dot{Q}_c = U_2 \cos\Phi - \dot{\Theta}\dot{\Phi}\sin\Phi + U_3 \sin\Phi\cos\Theta + \dot{\Psi}\dot{\Phi}\cos\Phi\cos\Theta - \dot{\Psi}\dot{\Theta}\sin\Phi\sin\Theta \quad (A.20)$$

$$\dot{R}_c = U_3 \cos\Phi\cos\Theta - U_2 \sin\Phi - \dot{\Theta}\dot{\Phi}\cos\Phi - \dot{\Psi}\dot{\Phi}\sin\Phi\cos\Theta - \dot{\Psi}\dot{\Theta}\cos\Phi\sin\Theta \quad (A.21)$$

T4

$$\begin{bmatrix}\delta_l \\ \delta_s \\ \delta_r\end{bmatrix} = \begin{bmatrix}b_1 & b_2 & b_3 \\ b_4 & b_5 & b_6 \\ b_7 & b_8 & b_9\end{bmatrix}^{-1}\begin{bmatrix}\dot{P}_c - L_1 \\ \dot{Q}_c - M_1 \\ \dot{R}_c - N_1\end{bmatrix} \quad (A.22)$$

$$b_1 = \frac{I_{xz}\left(\bar{C}_{n_{\delta_l}} + 2\bar{C}_{n_{\delta_{la}}}\right) + I_z\left(\bar{C}_{l_{\delta_l}} + 2\bar{C}_{l_{\delta_{la}}}\right)}{\frac{I_x I_z - I_{xz}^2}{\bar{q}Sb}} \quad (A.23)$$

$$b_2 = 0 \quad (A.24)$$

$$b_3 = \frac{I_{xz}\left(\bar{C}_{n_{\delta_r}} - \bar{C}_{n_{\delta_{ra}}}\right) + I_z\left(\bar{C}_{l_{\delta_r}} - \bar{C}_{l_{\delta_{ra}}}\right)}{\frac{I_x I_z - I_{xz}^2}{\bar{q}Sb}} \quad (A.25)$$

$$b_4 = 0 \quad (A.26)$$

$$b_5 = \frac{\bar{C}_{m_{\delta_s}}\bar{q}S\bar{c}}{I_y} \quad (A.27)$$

$$b_6 = 0 \quad (A.28)$$

$$b_7 = \frac{I_{xz}\left(\bar{C}_{l_{\delta_l}} + 2\bar{C}_{l_{\delta_{la}}}\right) + I_x\left(\bar{C}_{n_{\delta_l}} + 2\bar{C}_{n_{\delta_{la}}}\right)}{\frac{I_x I_z - I_{xz}^2}{\bar{q}Sb}} \quad (A.29)$$

$$b_8 = 0 \quad (A.30)$$

$$b_9 = \frac{I_{xz}\left(\bar{C}_{l_{\delta_r}} - \bar{C}_{l_{\delta_{ra}}}\right) + I_x\left(\bar{C}_{n_{\delta_r}} - \bar{C}_{n_{\delta_{ra}}}\right)}{\frac{I_x I_z - I_{xz}^2}{\bar{q}Sb}} \quad (A.31)$$

$$L_1 = \frac{1}{I_x I_z - I_{xz}^2}\left\{\left[\begin{array}{l} pqI_{xz}\left(I_x - I_y + I_z\right) - \\ qr\left(I_z^2 - I_y I_z + I_{xz}^2\right) \end{array}\right] + \right.$$

$$\bar{q}SbI_{xz}\left[\overline{C}_{n_\beta}\beta + \frac{b}{2(U_0+u)}\left(\overline{C}_{n_p}p + \overline{C}_{n_r}r\right)\right] +$$

$$\left.\bar{q}SbI_z\left[\overline{C}_{l_\beta}\beta + \frac{b}{2(U_0+u)}\left(\overline{C}_{l_p}p + \overline{C}_{l_r}r\right)\right]\right\} \quad (A.32)$$

$$M_1 = \frac{1}{I_y}\left\{\left[-rp\left(I_x - I_z\right) - I_{xz}\left(p^2 - r^2\right)\right] + \right.$$

$$\left.\bar{q}S\bar{c}\left[\begin{array}{l} \overline{C}_{m_o}\dfrac{u}{(U_0+u)} + \overline{C}_{m_\alpha}\alpha + \\ \dfrac{\bar{c}}{2(U_0+u)}\overline{C}_{m_q}q + \overline{C}_{m_{\delta_e}}\delta_c \end{array}\right]\right\} \quad (A.33)$$

$$N_1 = \frac{1}{I_x I_z - I_{xz}^2}\left\{\left[\begin{array}{l} -qrI_{xz}\left(I_x - I_y + I_z\right) - \\ pq\left(I_x I_y - I_x^2 - I_{xz}^2\right) \end{array}\right] + \right.$$

$$\bar{q}SbI_{xz}\left[\overline{C}_{l_\beta}\beta + \frac{b}{2(U_0+u)}\left(\overline{C}_{l_p}p + \overline{C}_{l_r}r\right)\right] +$$

$$\left.\bar{q}SbI_x\left[\overline{C}_{n_\beta}\beta + \frac{b}{2(U_0+u)}\left(\overline{C}_{n_p}p + \overline{C}_{n_r}r\right)\right]\right\} \quad (A.34)$$

Equations of Motion

$$\dot{p} = \frac{1}{I_x I_z - I_{xz}^2}\left\{\left[\begin{array}{l} pqI_{xz}\left(I_x - I_y + I_z\right) - \\ qr\left(I_z^2 - I_y I_z + I_{xz}^2\right) \end{array}\right] + \right.$$

$$\bar{q}SbI_{xz}\left[\begin{array}{l} C_{n_\beta}\beta + \dfrac{b}{2(U_0+u)}\left(C_{n_p}p + C_{n_r}r\right) + \\ C_{n_{\delta_a}}\delta_a + C_{n_{\delta_l}}\delta_l + C_{n_{\delta_{dc}}}\delta_{dc} + C_{n_{\delta_r}}\delta_r \end{array}\right] +$$

$$\left.\bar{q}SbI_z\left[\begin{array}{l} C_{l_\beta}\beta + \dfrac{b}{2(U_0+u)}\left(C_{l_p}p + C_{l_r}r\right) + \\ C_{l_{\delta_a}}\delta_a + C_{l_{\delta_l}}\delta_l + C_{l_{\delta_{dc}}}\delta_{dc} + C_{l_{\delta_r}}\delta_r \end{array}\right]\right\} \quad (A.35)$$

$$\dot{q} = \frac{1}{I_y}\left\{\left[-rp\left(I_x - I_z\right) - I_{xz}\left(p^2 - r^2\right)\right] + \right.$$

$$\left.\bar{q}S\bar{c}\left[\begin{array}{l} C_{m_o}\dfrac{u}{(U_0+u)} + C_{m_\alpha}\alpha + \\ \dfrac{\bar{c}}{2(U_0+u)}C_{m_q}q + C_{m_{\delta_s}}\delta_s + C_{m_{\delta_c}}\delta_c \end{array}\right]\right\} \quad (A.36)$$

$$\dot{r} = \frac{1}{I_x I_z - I_{xz}^2}\left\{\left[\begin{array}{l} -qrI_{xz}\left(I_x - I_y + I_z\right) - \\ pq\left(I_x I_y - I_x^2 - I_{xz}^2\right) \end{array}\right] + \right.$$

$$\bar{q}SbI_{xz}\left[\begin{array}{l} C_{l_\beta}\beta + \dfrac{b}{2(U_0+u)}\left(C_{l_p}p + C_{l_r}r\right) + \\ C_{l_{\delta_a}}\delta_a + C_{l_{\delta_l}}\delta_l + C_{l_{\delta_{dc}}}\delta_{dc} + C_{l_{\delta_r}}\delta_r \end{array}\right] +$$

$$\left.\bar{q}SbI_x\left[\begin{array}{l} C_{n_\beta}\beta + \dfrac{b}{2(U_0+u)}\left(C_{n_p}p + C_{n_r}r\right) + \\ C_{n_{\delta_a}}\delta_a + C_{n_{\delta_l}}\delta_l + C_{n_{\delta_{dc}}}\delta_{dc} + C_{n_{\delta_r}}\delta_r \end{array}\right]\right\} \quad (A.37)$$

$$\dot{u} = -g\sin\Theta\cos\Theta_0 + rv +$$

$$\frac{\bar{q}S}{m}\left[\begin{array}{l} C_{x_o}\dfrac{u}{(U_0+u)} + C_{x_\alpha}\alpha + \\ C_{x_{\delta_s}}\delta_s + C_{x_{\delta_c}}\delta_c \end{array}\right] \quad (A.38)$$

$$\dot{v} = g\sin\Phi\cos\Theta_0 - (U_0+u)r + pw +$$

$$\frac{\bar{q}S\bar{c}}{I_y}\left[\begin{array}{l} C_{y_\beta}\beta + \\ C_{y_{\delta_a}}\delta_a + C_{y_{\delta_l}}\delta_l + C_{y_{\delta_{dc}}}\delta_{dc} + C_{y_{\delta_r}}\delta_r \end{array}\right] \quad (A.39)$$

$$\dot{w} = -g\sin\Theta\sin\Theta_0 + (U_0+u)q - pv +$$

$$\frac{\bar{q}S}{m}\left[\begin{array}{l} C_{z_o}\dfrac{u}{(U_0+u)} + C_{z_\alpha}\alpha + \\ C_{z_{\delta_s}}\delta_s + C_{z_{\delta_c}}\delta_c \end{array}\right] \quad (A.40)$$

Appendix B: NASA Verification and Validation of Software Systems

The following is Section V of the NASA Software Quality Assurance Standard

V. VERIFICATION AND VALIDATION

A. Concepts and Definitions

Software Verification and Validation (V&V) is the process of ensuring that software being developed or changed will satisfy functional and other requirements (validation) and each step in the process of building the software yields the right products (verification). The differences between verification and validation are unimportant except to the theorist; practitioners use the term V&V to refer to all of the activities that are aimed at making sure the software will function as required.

V&V is intended to be a systematic and technical evaluation of software and associated products of the development and maintenance processes. Reviews and tests are done at the end of each phase of the development process to ensure software requirements are complete and testable and that design, code, documentation, and data satisfy those requirements.

B. Activities

The two major V&V activities are reviews, including inspections and walkthroughs, and testing.

1. Reviews, Inspections, and Walkthroughs

Reviews are conducted during and at the end of each phase of the life cycle to determine whether established requirements, design concepts, and specifications have been met. Reviews consist of the presentation of material to a review board or panel. Reviews are most effective when conducted by personnel who have not been directly involved in the development of the software being reviewed.

Informal reviews are conducted on an as-needed basis. The developer chooses a review panel and provides and/or presents the material to be reviewed. The material may be as informal as a computer listing or hand-written documentation.

Formal reviews are conducted at the end of each life cycle phase. The acquirer of the software appoints the formal review panel or board, who may make or affect a go/no-go decision to proceed to the next step of the life cycle. Formal reviews include the Software Requirements Review, the Software Preliminary Design Review, the Software Critical Design Review, and the Software Test Readiness Review.

An inspection or walkthrough is a detailed examination of a product on a step-by-step or line-of-code by line-of-code basis. The purpose of conducting inspections and walkthroughs is to find errors. The group that does an inspection or walkthrough is composed of peers from development, test, and quality assurance.

2. Testing

Testing is the operation of the software with real or simulated inputs to demonstrate that a product satisfies its requirements and, if it does not, to identify the specific differences between expected and actual results. There are varied levels of software tests, ranging from unit or element testing through integration testing and performance testing, up to software system and acceptance tests.

a. Informal Testing

Informal tests are done by the developer to measure the development progress. "Informal" in this case does not mean that the tests are done in a casual manner, just that the acquirer of the software is not formally involved, that witnessing of the testing is not required, and that the prime purpose of the tests is to find errors. Unit, component, and subsystem integration tests are usually informal tests.

Informal testing may be requirements-driven or design-driven. Requirements-driven or black box testing is done by selecting the input data and other parameters based on the software requirements and observing the outputs and reactions of the software. Black box testing can be done at any level of integration. In addition to testing for satisfaction of requirements, some of the objectives of requirements-driven testing are to ascertain:

Computational correctness.

Proper handling of boundary conditions, including extreme inputs and conditions that cause extreme outputs.

State transitioning as expected.

Proper behavior under stress or high load.

Adequate error detection, handling, and recovery.

Design-driven or white box testing is the process where the tester examines the internal workings of code. Design- driven testing is done by selecting the input data and other parameters based on the internal logic paths that are to be checked. The goals of design-driven testing include ascertaining correctness of:

All paths through the code. For most software products, this can be feasibly done only at the unit test level.

Bit-by-bit functioning of interfaces.

Size and timing of critical elements of code.

b. Formal Tests

Formal testing demonstrates that the software is ready for its intended use. A formal test should include an acquirer-approved test plan and procedures, quality assurance witnesses, a record of all discrepancies, and a test report. Formal testing is always requirements-driven, and its purpose is to demonstrate that the software meets its requirements.

Each software development project should have at least one formal test, the acceptance test that concludes the development activities and demonstrates that the software is ready for operations.

In addition to the final acceptance test, other formal testing may be done on a project. For example, if the software is to be developed and delivered in increments or builds, there may be incremental acceptance tests. As a practical matter, any contractually required test is usually considered a formal test; others are "informal."

After acceptance of a software product, all changes to the product should be accepted as a result of a formal test. Post acceptance testing should include regression testing. Regression testing involves rerunning previously used acceptance tests to ensure that the change did not disturb functions that have previously been accepted.

C. Verification and Validation During the Software Acquisition Life Cycle

The V&V Plan should cover all V&V activities to be performed during all phases of the life cycle. The V&V Plan Data Item Description (DID) may be rolled out of the Product Assurance Plan DID contained in the SMAP Management Plan Documentation Standard and DID.

1. Software Concept and Initiation Phase

The major V&V activity during this phase is to develop a concept of how the system is to be reviewed and tested. Simple projects may compress the life cycle steps; if so, the reviews may have to be compressed. Test concepts may involve simple generation of test cases by a user representative or may require the development of elaborate simulators and test data generators. Without an adequate V&V concept and plan, the cost, schedule, and complexity of the project may be poorly estimated due to the lack of adequate test capabilities and data.

2. Software Requirements Phase

V&V activities during this phase should include:

Analyzing software requirements to determine if they are consistent with, and within the scope of, system requirements.

Assuring that the requirements are testable and capable of being satisfied.

Creating a preliminary version of the Acceptance Test Plan, including a verification matrix, which relates requirements to the tests used to demonstrate that requirements are satisfied.

Beginning development, if needed, of test beds and test data generators.

The phase-ending Software Requirements Review (SRR).

3. Software Architectural (Preliminary) Design Phase

V&V activities during this phase should include:

Updating the preliminary version of the Acceptance Test Plan and the verification matrix.

Conducting informal reviews and walkthroughs or inspections of the preliminary software and data base designs.

The phase-ending Preliminary Design Review (PDR) at which the allocation of requirements to the software architecture is reviewed and approved.

4. Software Detailed Design Phase

V&V activities during this phase should include:

Completing the Acceptance Test Plan and the verification matrix, including test specifications and unit test plans.

Conducting informal reviews and walkthroughs or inspections of the detailed software and data base designs.

The Critical Design Review (CDR) which completes the software detailed design phase.

5. Software Implementation Phase

V&V activities during this phase should include:

Code inspections and/or walkthroughs.

Unit testing software and data structures.

Locating, correcting, and retesting errors.

Development of detailed test procedures for the next two phases.

6. Software Integration and Test Phase

This phase is a major V&V effort, where the tested units from the previous phase are integrated into subsystems and then the final system. Activities during this phase should include:

Conducting tests per test procedures.

Documenting test performance, test completion, and conformance of test results versus expected results.

Providing a test report that includes a summary of nonconformances found during testing.

Locating, recording, correcting, and retesting nonconformances.

The Test Readiness Review (TRR), confirming the product's readiness for acceptance testing.

7. *Software Acceptance and Delivery Phase*

V&V activities during this phase should include:

By test, analysis, and inspection, demonstrating that the developed system meets its functional, performance, and interface requirements.

Locating, correcting, and retesting nonconformances.

The phase-ending Acceptance Review (AR).

8. *Software Sustaining Engineering and Operations Phase*

Any V&V activities conducted during the prior seven phases are conducted during this phase as they pertain to the revision or update of the software.

D. Independent Verification and Validation

Independent Verification and Validation (IV&V) is a process whereby the products of the software development life cycle phases are independently reviewed, verified, and validated by an organization that is neither the developer nor the acquirer of the software. The IV&V agent should have no stake in the success or failure of the software. The IV&V agent's only interest should be to make sure that the software is thoroughly tested against its complete set of requirements.

The IV&V activities duplicate the V&V activities step-by-step during the life cycle, with the exception that the IV&V agent does no informal testing. If there is an IV&V agent, the formal acceptance testing may be done only once, by the IV&V agent. In this case, the developer will do a formal demonstration that the software is ready for formal acceptance.

E. Techniques and Tools

Perhaps more tools have been developed to aid the V&V of software (especially testing) than any other software activity. The tools available include code tracers, special purpose memory dumpers and formatters, data generators, simulations, and emulations. Some tools are essential for testing any significant set of software, and, if they have to be developed, may turn out to be a significant cost and schedule driver.

An especially useful technique for finding errors is the formal inspection. Formal inspections were developed by Michael Fagan of IBM. Like walkthroughs, inspections involve the line-by-line evaluation of the product being reviewed. Inspections, however, are significantly different from walkthroughs and are significantly more effective. Inspections are done by a team, each member of which has a specific role. The team is led by a moderator, who is formally trained in the inspection process. The team includes a reader, who leads the team through the item; one or more reviewers, who look for faults in the item; a recorder, who notes the faults; and the author, who helps explain the item being inspected.

This formal, highly structured inspection process has been extremely effective in finding and eliminating errors. It can be applied to any product of the software development process, including documents, design, and code. One of its important side benefits has been the direct feedback to the developer/author, and the significant improvement in quality that results.

REFERENCES

Allen, J. F., 1984 "Towards a General Theory of Action and Time", *Artificial Intelligence* 23(2):pp. 123-154.

Amalberti, R., and F. Deblon, 1992 "Cognitive Modeling of Fighter Aircraft Process Control: a Step Towards an Intelligent On-board Assistance System", *International Journal of Man-Machine Studies,* vol. 36, pp. 639-670.

Astrom, K.J. and T.J. McAvoy, 1992. "Intelligent Control: An Overview And Evaluation", *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches*, ed. White & Sofge, Van Nostrand Reinhold, New York, New York, pp. 3-34.

Ball, J. W. Sr., 1992. "System Status: the Diagnostic Edge of the Pilot's Associate", ISA, pp. 19-29.

Caudill, M., 1989. "Neural Network Primer: Part I", AI Expert, Feb. 1989

Chin H. H., 1992. "Knowledge-Based System Techniques For Pilot Aiding," *Control and Dynamic System- System Performance Improvement and Optimization Techniques and Their Applications in Aerospace Systems,* pp. 69-174.

DARPA Neural Network Study (1988). Alexandria, VA: AFCEA International Press.

Funk, K. H., and J. H. Lind, 1992. "Agent-Based Pilot-Vehicle Interfaces: Concept and Prototype", *IEEE Transactions on System, Man, and Cybernetics,* Vol. 22, No. 6, pp. 1309-1322, Nov./Dec. 1992.

Gonzalez, A. J., and D. D. Douglas, 1993 "The Engineering of Knowledge-Based Systems", Prentice Hall, Englewood Cliffs, New Jersey 07632.

Govindaraj, T., and C. M. Mitchell, 1994. "Operator Modeling in Commercial Aviation: Cognitive Models, Intelligent Displays, and Pilot's Assistants, NASA-CR 198609, Georgia Institute of Technology, Atlanta, GA.

Jorgensen, C.C., 1996. "NASA News Briefs", NASA Tech Briefs, pp. 24-26, Sept. 1996.

Jorgensen, C.C., and C. Schley, 1990 "A Neural Network Baseline Problem for Control of Aircraft Flare and Touchdown", *Neural Networks for Control*, ed. Miller, Sutton, and Werbos, MIT Press, pp. 403-425.

Kim, B. S. and A. J. Calise, 1994. "Nonlinear Flight Control Using Neural Networks", AIAA Paper 94-3646-CP.

Kim, K. and E.B. Bartlett, (1994) "Validation of a Nuclear Power Plant Fault-Diagnostic System Using Artificial Neural Networks", *Paper presented at the Artificial Neural Networks in Engineering Conference;* St. Louis, Missouri.

Kosko, B., 1992. "Neural Networks and Fuzzy Systems: a Dynamic Systems Approach To Machine Intelligence", Prentice-Hall, Englewood Cliffs, New Jersey.

Krobusek, R. D., R. M. Boys, and K. D. Palko, 1989. "Levels of autonomy in a tactical electronic crewmember," J. Emerson, J. Reising, R. M. Taylor, and M. Reinecke, Eds., WRDC-TR-89-7008. Wright-Patternson Air Force Base, OH: Wright Research and Development Center, pp. 124-132.

Liano, K., 1996. "Robust Error Measure for Supervised Neural Network Learning with Outliers", IEEE Transactions on Neural Networks, Vol. 7, No. 1, pp. 246-250.

Menon, P. K. A., 1993. "Nonlinear Command Augmentation System for a High Performance Fighter Aircraft," Proceedings of AIAA Guidance, Navigation and Control Conference, Monterey, CA, pp. 720-730.

Mitchell, C. M., 1994. "Human-Centered Design of Human-Computer-Human Dialogs in Aerospace Systems, NASA-CR-197379, 31 July 1994.

Nagel, D. C., 1988. "Human Error In Aviation Operations", *Human Factors in Aviation*, San Diego, Academic, pp. 263-303.

Noor, A. K., and C. C. Jorgensen, 1996. "A Hard Look At Soft Computing", Aerospace America, pp. 34-39, September, 1996.

Onken, R., 1995. "Functional Development and Field Test of CASSY - a Knowledge-Based Cockpit Assistant System", *Paper presented at the Mission Systems Panel of AGARD;* NASA AMES Research Center, California.

Owre, S., J. Rushby, N. Shankar, & F. von Henke, 1995. "Architectures: Prolegomena to the Design of PVS", IEEE Transactions on Software Engineering, Vol. 21, No. 2, pp. 107-125.

Painter, J. H., E. Glass, G. Economides, & P. Russell, 1994. "Knowledge-Based Processing for Aircraft Flight Control", NASA-CR 194976, Texas A&M University, College Station, Texas, NASA-CR-194976.

Peterson, G. E., 1993. "A Foundation for Neural Network Verification and Validation", in Science of Artificial Neural Networks II, Dennis W. Ruck, Editor, Proc. SPIE 1966, pp. 196-207.

Peterson, G. E., D. C. St. Clair, S. R. Aylward, W. E. Bond, 1995. "Using Taguchi's Method of Experimental Design to Control Errors in Layered Perceptrons", IEEE Transactions on Neural Networks, Vol. 6, No. 4, pp. 949-961.

Phatak, D. S., & I. Koren, 1995. "Complete and Partial Fault Tolerance of Feedforward Neural Nets", IEEE Transactions on Neural Networks, Vol. 6, No. 2, pp. 446-456.

Rouse, W. B., N. D. Geddes, & J. M. Hammer, 1990. "Computer-aided Fighter Pilots", IEEE Spectrum, pp. 38-41.

Rushby, J. M., & F. von Henke, 1993. "Formal Verification of Algorithms for Critical Systems", IEEE Transactions on Software Engineering, Vol. 19, No. 1, pp. 13-23.

Russell, S., P. Norvig, 1995. "Artificial Intelligence: A Modern Approach" , Prentice Hall, New Jersey.

Shelnutt, J. B., 1989. "Pilot vehicle interface management," in *The Human-Electronic Crew: Can They Work Together?* J. Emerson, J. Reising, R. M. Taylor, and M. Reinecke, Eds., WRDC-TR-89-7008. Wright-Patternson Air Force Base, OH: Wright Research and Development Center, pp. 104-107.

Steck, J. E., K. Rokhsaz, & S. Shue, 1996, "Linear and Neural Network Feedback for Flight Control Decoupling", IEEE Control Systems, pp. 22-30, August 1996

Steinberg, M., 1992. "Potential Role of Neural Networks and Fuzzy Logic in Flight Control Design and Development", *Paper presented at the 1992 Aerospace Design Conference;* Irvine, California.

Sucki, G. J., W. L. Persons, & G. L. Johnson, 1994. "The Safety Implications of Emerging Software Paradigms", *Paper presented at the International Federation of Automatic Control Emerging Intelligent Control Technologies,* Hong Kong.

Totah, J. J., 1996. "Simulation Evaluation of a Neural-Based Flight Controller", AIAA 96-3503.

Totah, J., 1995. "An Examination of Aircraft Aerodynamic Estimation Using Neural Networks", SAE Technical Paper Series 952036.

Wen, W., & J. Callaham, 1996. "Neuralware Engineering: Develop Verifiable ANN-based Systems", Technical Report, NASA/WVU Software Research Laboratory, *Paper presented at the IEEE International Joint Symposia on Intelligence and Systems;* Rockville, Maryland, pp. 60-66, 4-5 November 1996.

Werbos, P. J., 1990. "Overview of Designs and Capabilities", *Neural Networks for Control,* ed. Miller, Sutton, and Werbos, MIT Press, pp. 59-66.

Werbos, P. J., 1992. "Neurocontrol and Supervised Learning: An Overview and Evaluation", *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches,* ed. White & Sofge, Van Nostrand Reinhold, New York, New York, pp. 65-90.

Wilber, G. F., 1989. "Simulating intelligent missions." *Defense Computing,* vol. 2, no. 6, pp. 43-47 Nov. /Dec. 1989.

Wilber, R., 1989. "Expert systems aid on-board mission management," *Defense Computing,* vol. 2, no. 1, pp. 27-30 Jan. /Feb. 1989.

White, D. A., A. Bowers, K. Iliff, and J. Menousk, 1992. "Flight, Propulsion, and Thermal Control of Advanced Aircraft and Hypersonic Vehicles", *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches,* ed. White & Sofge, Van Nostrand Reinhold, New York, New York, pp. 65-90.

Glossary of Technical Terms

Algorithm. A set of step-by-step instructions for accomplishing a task.

Angle of attack. The acute angle between the chord of an airfoil, and a line representing the undisturbed relative airflow.

Angular acceleration. Rotational acceleration of an aircraft about a single or multiple axes.

Antecedent. A condition (premise, evidence) in a rule/logical statement.

Approximate reasoning. Inexact reasoning employing techniques such as probability theory, fuzzy logic, or certainty factors.

Artificial Intelligence. A field of study in computer science that pursues the goal of making a computer reason in the same manner as humans.

Associative networks. A method of knowledge representation using graphs made up of nodes and arcs, where the nodes represent objects and the arcs the relationship between the objects.

Automatic pilot. A control mechanism which initiates signals to the control surfaces of an aircraft to maintain a steady and preset course and attitude, without assistance from the human pilot. Also called copilot or autopilot.

Avionics. A field of applied research in which electronic devices and computers are adapted to use in aviation. Coined from *avi*ation electr*onics*.

Backward chaining. An inference control strategy. In a rule-based system, backward chaining begins with a goal and tries to prove it to be true by proving the premises of a rule that contains the goal as its conclusion.

Best-first search. Search technique that uses knowledge about the problem to guide the search.

Binary tree. A tree in which each node has only two children.

Binding. The process of assigning a value to a variable.

Blackboard. A system where several independent agents or knowledge sources share information in a common working memory.

Blind search. A search technique that does not make use of knowledge to search the problem space. It will find a solution, if it exists, but at a cost of time due to the exhaustive nature of the search.

Breadth-first search. A search technique that looks for a solution along all of the nodes on one level of a problem space before considering nodes at the next lower level.

Case-based reasoning. Theory in AI that proposes that the elements of human memory are based on specific historical events or cases. A problem is compared to a set of previous cases and the case most similar to the problem is used as the basis for solving the problem.

Causal reasoning. Reasoning based on cause-effect relationships between the problem's objects.

Certainty. Degree of confidence or belief in a fact or relationship.

Certainty factor. A number assigned to a fact or rule to indicate the confidence one has in the fact or in the rule's relationship.

Cognition. The mental process of knowing. Having knowledge.

Cognitive science. The study of the human problem-solving processes.

Cognitive modeling. Models of human reasoning, model of how the human mind works, a computer program that models human reasoning.

Competence model. A model of human reasoning sufficient to perform the task being investigated.

Consequent. The conclusion (action) of a rule/logical statement.

Contradiction. Condition where the antecedents of two rules are the same while the consequence are different.

Control. A procedure, explicit or implicit, that determines the overall order of problem-solving activities in a system.

Data-driven. An inference method which starts with all available data or evidence obtained or determined by the system and from this information deduces which hypothesis may or may not be true. Also called forward chaining.

Declarative knowledge. Descriptive or factual knowledge.

Declarative representation. Descriptive statements of static knowledge - facts about objects, events, and their relations. In contrast, procedural representation is executable statements of knowledge -- knowledge captured in procedures.

Deductive reasoning. Reasoning from the general to the specific.

Deep knowledge. Basic knowledge coming from first principles or physical laws of the domain.

Defuzzification. Process of converting a fuzzy value into a crisp value.

Degree of membership. The likelihood, expressed as a number from 0 to 1, that a particular object belongs in a fuzzy set.

Depth-first search. A search technique that explores each branch of a search space to its full vertical depth, then proceeds alone the left or right branches at that depth. Each branch is searched for a solution; in none is found, a new vertical branch is searched using the same rule of search.

Domain. The problem area. Example: medical diagnostics.

Domain-specific knowledge. Knowledge about the problem area.

Encapsulation. The hiding of data and procedures within an object.

Evaluation function. Procedure used to determine the value of a proposed path through a problem space.

Exhaustive search. A search technique where every possible path through a problem space is examined.

Existential quantifier. For some variable X there exists some object that could instantiate the variable.

Fact. A declarative assertion or statement that has the property of being either true or false.

Fault tree. A tree diagram containing an ordered representation of faults that may be found in a system.

Fire. To activate the conclusion of a rule if the premises are true.

First principles. Basic theory of the domain used to solve the problem rather than rules of thumb.

Forward Chaining. inference strategy where conclusions are drawn by first looking at the facts or data of the problem. Also known as data-driven.

Frames. A frame is a data structure or knowledge representation method that associates an object with a collection of features. Each feature or attribute is stored in a slot with a corresponding attribute value, or method for acquiring the value. Frames include declarative and procedural information in predefined internal relations. Action frame are frames that describe action. State-change frames are frames that contain an object slot that is filled with an application-specific object or quantity.

Frame-based system. A program that processes problem-specific information contained in the working memory with a set of frames contained in the knowledge base, using an inference engine to infer new information.

Fuzzy logic. A branch of logic that uses degrees of membership in sets rather than a strict true/false membership.

Fuzzy reasoning. Method of working with inexact information. Works with subjective or poorly understood concepts to determine an adequate solution.

Fuzzy set. Degree of membership distribution for membership of some object in a linguistic variable set.

Goal. A hypothesis to prove. A node in a search space containing a solution.

Goal driven. An inference technique that begins with a goal or hypothesis and works backward through the rules in an attempt to prove the goal. Also called backward chaining.

Heuristic. Knowledge, often expressed as a rule of thumb, that guides the search process.

Hypothesis. A statement that is subject to verification or proof. A goal in a goal-driven system.

Induction. Inducing rules from knowledge contained in a set.

Inductive reasoning. Reasoning from the specific to the general.

Inference. The process of deriving new information from known information.

Inference engine. Processor that matches the facts contained in the working memory and in the domain knowledge contained in the knowledge base, to draw conclusions about the problem.

Inference network. Graphical representation of the system's rules with the antecedents and consequences of the rules drawn as nodes and their supporting relationships drawn as links.

Inheritance. Process by which the characteristics of a parent are assumed by its child.

Instance. A specific object from a class of objects.

Intelligence. The ability to acquire and apply knowledge to solve a problem.

Interpreter. The inference engine of a knowledge-based system.

Knowledge. A collection of facts, rules, and concepts used to reason with.

Knowledge acquisition. The process of acquiring, organizing and studying knowledge.

Knowledge base. Part of the system that contains the domain knowledge.

Knowledge-based system. Systems whose performance depends on encoded knowledge.

Knowledge elicitation. The process of acquiring knowledge from a domain expert to enter into the knowledge base.

Knowledge representation. The method used to encode knowledge in a system's knowledge base.

Learning. The process of gaining knowledge and understanding through education or experience.

Learning control system. A system that has the ability to improve its performance in the future, based on experiential information it has gained in the past, through closed-loop interactions with the plant and its environment.

Logic. A system of reasoning based on the study of propositions and their analysis in making deductions.

Logic-based system. A system employing formal logic as its primary representation.

Membership function. A formula used to determine the degree of membership of some object in some fuzzy set.

Meta-knowledge. Knowledge in a system that explains how the system is controlled of reasons. Knowledge about knowledge.

Meta-rule. A rule that describes how to control the problem-solving process.

Model-based reasoning. A method of reasoning that draws conclusions about the state of a modeled system exposed to a set of simulated data.

Modus ponens. A rule of logic that assets that if we know A is true and A implies B, then we can assume B is true.

Monotonic reasoning. Method of reasoning that assumes once a fact is asserted it cannot be altered during the course of the reasoning.

Objective function. For a control system to improve its performance, its learning system must operate in context of the objectives of the system. Attainment of these objectives is based on system performance feedback which characterizes the appropriateness of the systems behavior.

Polymorphism. A characteristic of object-oriented programming or frame-based systems in which a given message may be interpreted and acted upon differently between objects or frames.

Predicate. A statement about the subject of a proposition.

Predicate calculus. A programming language or logic system where statements about objects and their relationships are made. Each element in the predicate calculus is an object and the statements are called predicated. It is an extension of propositional calculus.

Premise. A statement in the IF part of the rule that must be satisfied before the rule's conclusion is accepted.

Probability. A number representing the likelihood of a given event occurring.

Problem space. A tree or graph containing nodes and branches used for searching for a solution to a given problem. The nodes represent possible problem states and the branches possible paths between states.

Procedure. A well structured way of performing a given task.

Procedural representation. Procedural statement when executed derive knowledge. In contrast, declarative representation is descriptive statements of static knowledge.

Proposition. A declarative assertion or statement that has the property of being true or false.

Propositional calculus. Logical system for reasoning. Conclusions are obtained from a series of statements according to the processing of rules.

Pruning. Reducing the alternatives in a problem space during search when it appears that continual search in the pruned area will be fruitless.

Qualitative reasoning. A way to describe phenomena in terms of causal, compositional, or subtypical relationships among objects or events.

Reasoning. The process of working with knowledge, facts and problem-solving methods to draw conclusions or inferences.

Resolution. Inference strategy used in logical systems to determine the truth of an assertion.

Rule. A method of representing knowledge consisting of premises and a conclusion.

Rule-based system. A computer program that processes problem-specific information contained in the working memory with a set of rules contained in the knowledge base, using an inference engine to infer new information.

Rule of thumb. A rule based on good judgment, gained from experience rather than first principles.

Semantic network. A method of knowledge representation using graphs made up of nodes and arcs, where the nodes represent objects and the arcs represent the relationship between the objects.

Tactical. To arrange, position, or maneuver forces in contact, or near contact, with the enemy so as to achieve an objective or objectives in battle; near-term solution.

Tactical Procedures. Procedures of maneuvering military forces in action. For military aircraft, procedures that the pilot executes either defensively or offensively.

Taxonomy. Classification of objects that are alike.

Temporal reasoning. Reasoning about problem states as they evolve over time.

Threshold. A numeric that must be exceeded before some action is taken.

Unification. A rule of inference in logic that is used to unify two formulas to produce a clause.

Universal quantifier. A statement that indicates that a statement is true for all values of a problem.

Waypoint. A reference point between the point of departure and the destination, particularly a point on a course line the coordinates of which are defined in relation to an electronic aid to navigation.

Wingman. A pilot and his aircraft, who flies at the side and to the rear of the leader aircraft, commonly in a two-plane or three-plane formation.

Index