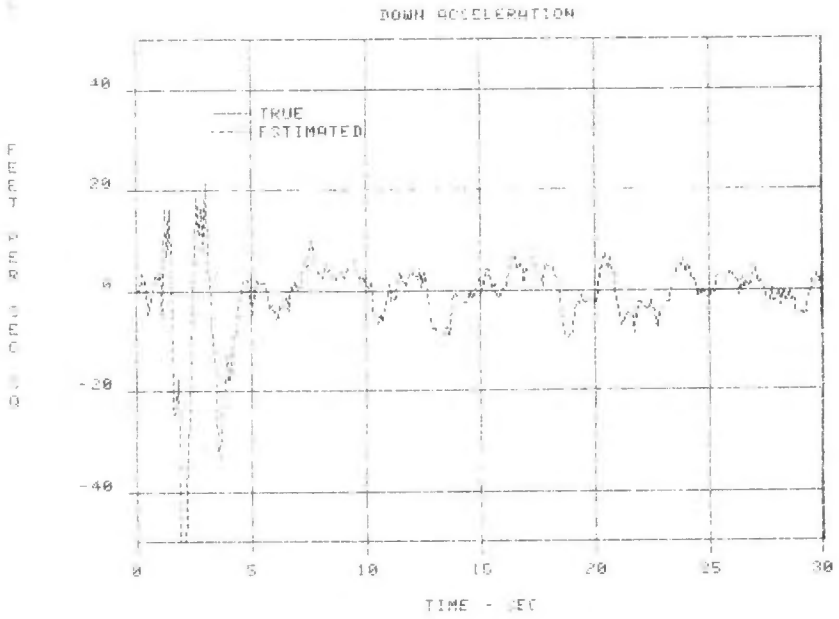


Figure 7. Mode 1 Downward Acceleration History

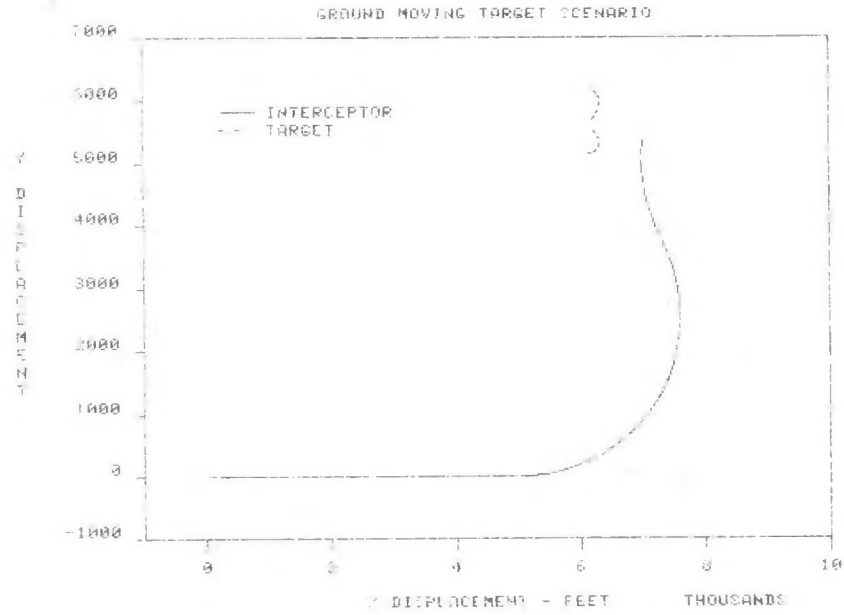


Figure 8. Mode 2 Scenario

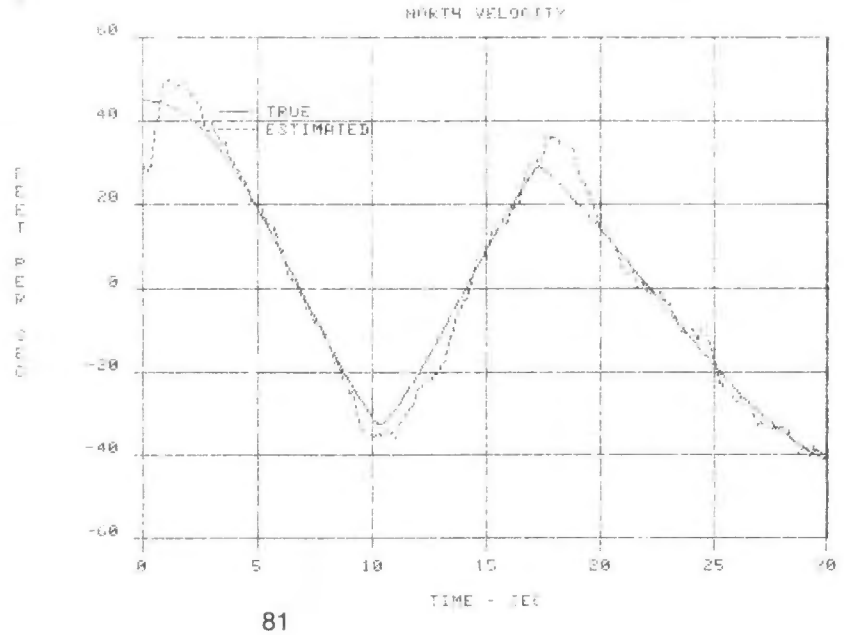


Figure 9. Mode 2 North Velocity History

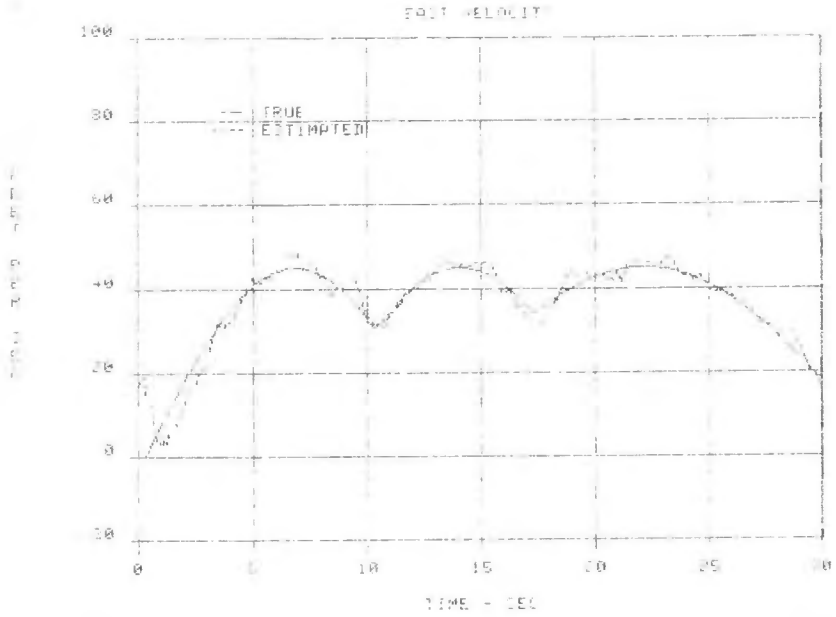


Figure 10. Mode 2 East Velocity History

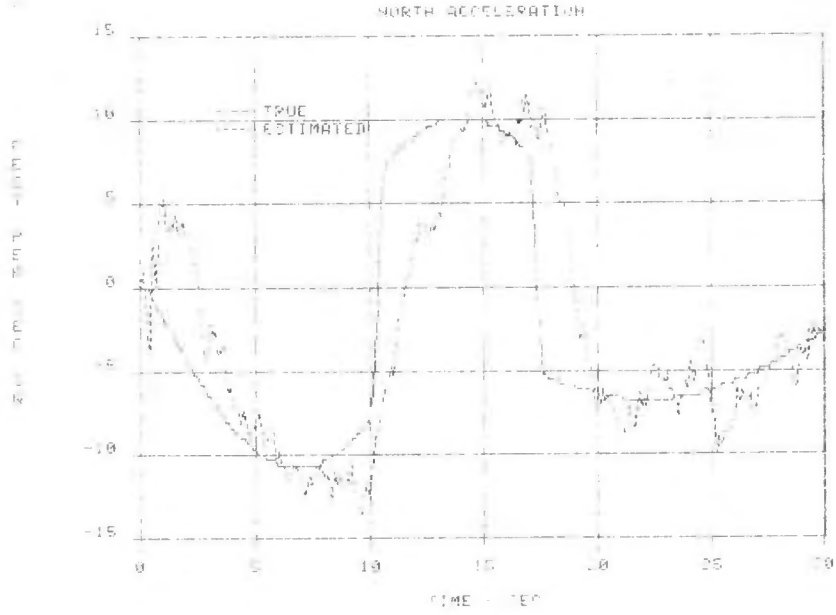


Figure 11. Mode 2 North Acceleration History

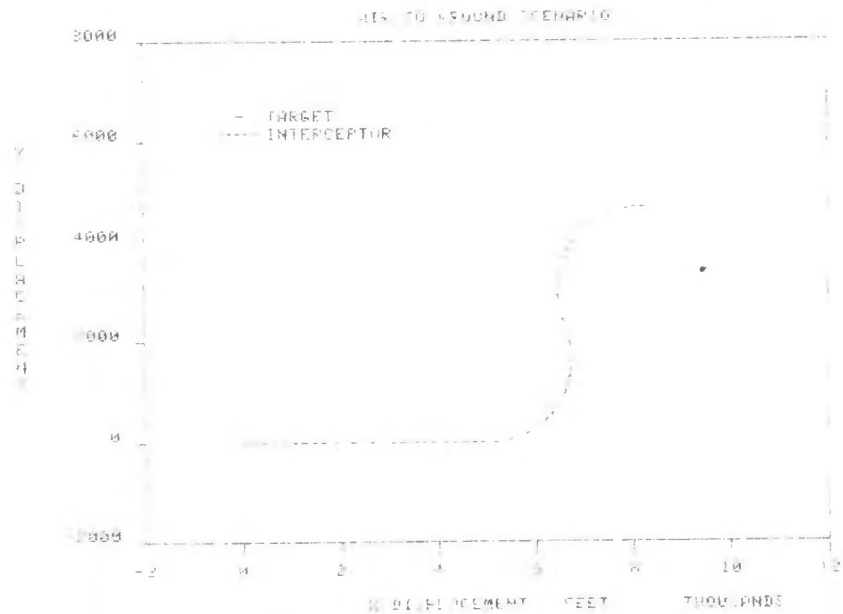


Figure 12. Mode 3 Scenario

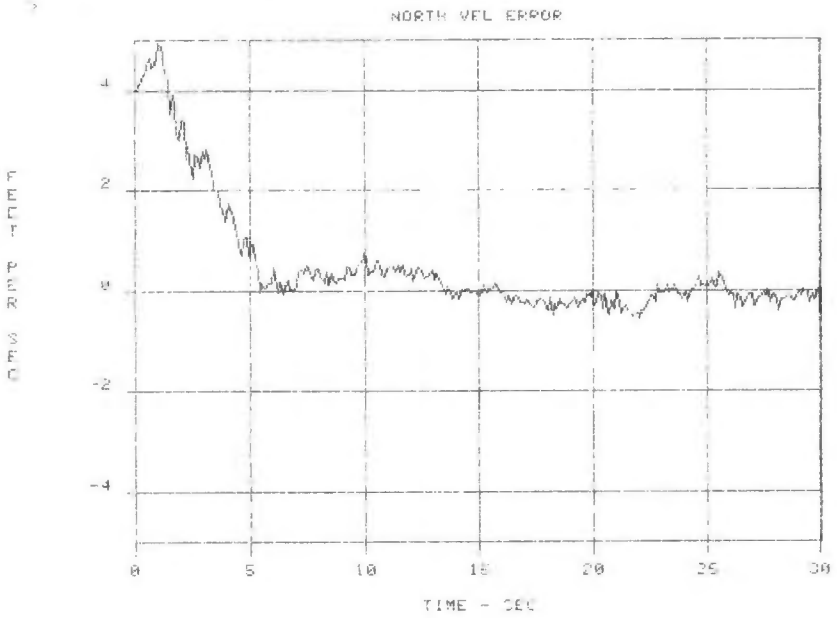


Figure 13. Mode 3 North Velocity History

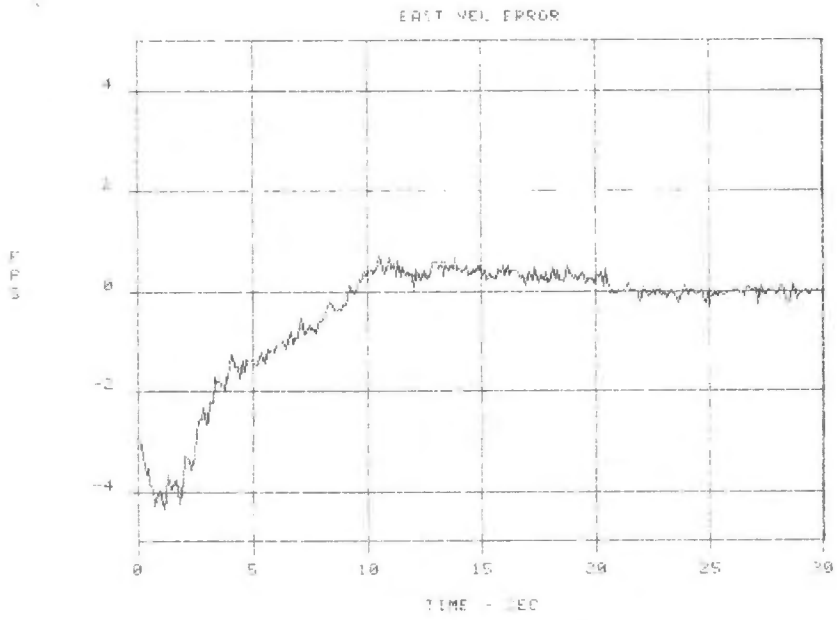


Figure 14. Mode 3 East Velocity History

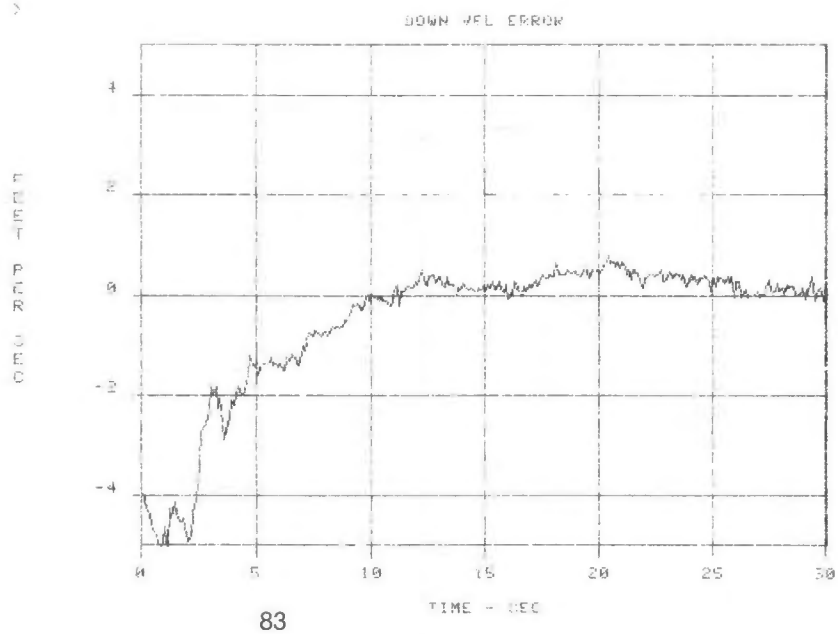


Figure 15. Mode 3 Downward Velocity History

SESSION 4 CREW SYSTEMS - HUMAN FACTORS AND ARTIFICIAL INTELLIGENCE

4

Chairmen:

Remus N. Bretoi
NASA Ames Research Center

Major Dean Cole
Air Force Aerospace Medical
Research Lab

This session focuses on the interface between crew and vehicle, including displays, controls and automation; and the impact of new technologies such as computer graphics and artificial intelligence on crew performance.

THE EVALUATION OF DISPLAY SYMBOLOGY: A CHRONOMETRIC STUDY OF VISUAL SEARCH

Roger Remington

NASA - Ames Research Center
Aero-Space Human Factors Research Division
MS 239-3
Moffett Field, CA 94035

Douglas Williams

Psycho-Linguistic Research Associates
2055 Sterling Avenue
Menlo Park, CA 94025

Abstract

Three single-target visual search tasks were used to evaluate a set of CRT symbols for a helicopter traffic display. The search tasks were representative of the kinds of information extraction required in practice, and reaction time was used to measure the efficiency with which symbols could be located and identified. The results show that familiar numeric symbols were responded to more quickly than qraphic symbols. The addition of modifier symbols such as a nearby flashing dot or surrounding square had a greater disruptive effect on the graphic symbols than the alphanumeric characters. The results suggest that a symbol set is like a list that must be learned. Factors that affect the time to respond to items in a list, such as familiarity and visual discriminability, and the division of list items into categories, also affect the time to identify symbols.

Introduction

The selection and evaluation of symbology for cathode ray tube (CRT) displays is an important part of display design, yet there are no generally agreed upon procedures for selecting symbologies or evaluating candidate sets. It is not difficult to design tests to compare a small number of alternate sets. The focus, however, is often not on the relative performance of different sets, but whether a given set is adequate. As a practical design issue it is not possible to generate sets of alternate symbols and check all possible combinations. The approach has been to provide guidelines for the use of certain symbol attributes (3,7) or the development of performance-based criteria (4). While these guidelines are useful, general principles are often overidden by situationally specific factors. It seems more fruitful to search for display principles that pertain to a restricted class of displays to be used under similar circumstances.

We were confronted with these problems in responding to a request to certify that a set of symbols selected for use on a 3-inch circular CRT helicopter traffic situation display were adequate for quick recognition and and low confusability. The symbol set is shown in Table 1. The rows of Table 1 are the symbols identifying a category of other aircraft; the columns are "marker" conditions giving additional status information. Each cell entry shows a given symbol in a particular marker condition, the first column being no marker. The numbers in the margins are the numbers used to refer to a given symbol or marker. Each symbol in Table 1 is defined in terms of the distribution of "on" cells in a pixel matrix. The set includes both numeric and graphic symbols. The numbers were chosen to designate certain characteristics of the crafts they represented; the graphic characters, however, were an entirely abstract representation. The goal of the experiments was to determine whether the selected symbols were, in all marker configurations, sufficiently distinguishable from one another so that each could be identified quickly and accurately.

This instance is typical of the design process. The symbols in Table 1 were chosen by committee, and the requirement for a semantic or graphical relationship between the symbols and their associated threats reflects some intuition common to the committee members that such a relationship would improve performance by providing a mnemonic aid for identification. This assumes both that symbol identification is the crucial time-consuming mental process, and that the graphic representation will facilitate identification. While such mnemonics may be useful with very large symbols sets, these relationships become less important when the set size is small and overlearned. In these cases, visual discriminability is an important determiner of display access time. It might be more important to choose symbols that are not visually confusable (3).

Identification involves memory access; hence, the use of graphic symbols will facilitate identification only when they facilitate memory access. Studies of lexical access have shown that the time to identify a letter string as a word is inversely related to its frequency of occurrence. High frequency words require less presentation time and are less subject to masking than low frequency items (1). The same relationship holds for picture naming latency. Subjects are faster to name or recognize objects whose names have a higher frequency of occurrence (2,5,6,8). Word frequency is often associated with frequency of exposure to an object, but there is no direct evidence showing that the frequency of a particular graphic symbol determines identification time. It would be important to know for display design whether unfamiliar, idiosynchratic symbols required more processing time than highly familiar digits or letters. Christ and Corso (3) have found that digits are responded to more quickly than letters or graphic symbols, but only on some tasks. We sought to clarify the role of symbol familiarity by focusing on a particular display type and confining our conclusions to similar displays.

A visual search paradigm was used to evaluate the discriminability of the symbols and assess the effects of familiarity. Measuring the time required to find a target symbol among a set of distractors has advantages over techniques that measure discriminability by ratings or by the confusability of items in noise. In practice, pilots search the display to find and identify other aircraft, or confirm a reported siting, and determine the appropriate action. Visual search then is an important component of display use. The time to react to potential collisions could determine the pilot's survival. Since the experimental display hardware is identical to the one intended for use, the arrangement of symbols on the display and symbol discriminability can be evaluated to determine their contribution to actual search time.

Experiment 1

Method

Apparatus. The subject sat in a sound attenuated booth in an adjustable helicopter seat and viewed a 5 inch Sony KV-5000 color monitor through the double- paned glass window of the booth at a distance of 25 inches. The rest of the window was shrouded with black material to eliminate reflections, and all but a 3 inch diameter portion of the CRT screen was masked. On a shelf in front of the subject was an 18 inch board covered in black cloth on which two microswitches were mounted, one labeled "yes" the other "no".

An Apple II+ computer was used to control the experiment, randomize the presentation of stimuli, and record responses and response times. The stimuli were presented by a SOL-20 computer using a Cromemco TV dazzler color board set programmed to give a 128 X 128 pixel display of cyan colored symbols on a black background. The Apple and SOL communicated over an RS232 serial line. On each trial the Apple sent an ASCII string to the SOL indicating which items to display, and a signal that blanked the display at the appropriate times. Since the Apple began timing before it sent the first symbol, the reaction times are inflated by a constant amount that reflects the time to transmit the information and the time for the SOL software to display the indicated items. This delay was approximately 700 ms, and was a constant affecting reaction times in all conditions equally.

Stimuli. Each symbol and marker from Table 1 was drawn as a character in a 16 X 16 pixel matrix using a Cromemco TV dazzler color board and associated software. There were 36 pixels per inch, making the 16 X 16 matrix a square of 0.44 inches per side. At a viewing distance of 25 inches each matrix subtended approximately 1 degree of visual angle horizontally and vertically. Not all stimuli were of an equal horizontal or vertical extent, and each used different numbers of cells in the matrix. In the first experiment only the first column of symbols, those without markers, were used. Thus not all of the matrix is needed to represent each symbol.

The stimuli were presented in frames. Each frame consisted of four symbols with one of the symbols designated as the target symbol for that frame and the three distractor stimuli chosen quasi-randomly with the constraint that each target appear at least once with every other symbol and that all symbols occur about equally often. The experimental displays were modeled closely on the actual display. Symbols could occur in one of two concentric circular rings. The inner ring was on the circumference of a circle with a center at the center of the display and a radius of approximately 21 pixels (1.34 deg visual angle). The radius of the circle for the outer ring was approximately 120 pixels (3.8 deg visual angle). Within a frame symbols could occur in both inner and outer rings, and at any of the 12 clock positions in each ring. The separation of symbols within a particular frame was manipulated by restricting the location of symbols in that frame to a specified number of quadrants. Thus, the highest density frames required all stimuli to fall into one randomly selected quadrant. In the least dense frames each symbol occupied a separate quadrant. This procedure does not insure that the distance, or dispersion of symbols in the 4-quadrant condition, for example, is always greater than in the 3-quadrant condition. It was, however, easy to implement, produced the required separations on the average, and the values were adjusted as the frames were produced to avoid any obvious discrepancies such as two adjacent figures on a quadrant boundary. There were 108 frames in all. Each symbol appeared equally often as targets, with 12 frames for each target, 3 frames for each target in each quadrant.

Design and Procedure. Experiment 1 was intended to assess the discriminability of the symbols without markers and the effects of symbol separation on search times. There were 432 trials in each experimental session, divided into 4 blocks of 108 trials each. Each trial began with the presentation of the target symbol which appeared in the center of the screen and remained on for one second. Half a second after the target was extinguished a display of four symbols appeared and subjects were to press one of two keys as quickly as possible to indicate whether the target was present (positive trial) or absent (negative trial). Half the subjects pressed the left key to indicate that the symbol was in the set, and pressed the right key when they failed to find the target symbol. For the other half of the subjects this key assignment was reversed. In actual use the display can have up to eight symbols, but the use of four symbols is representative of moderately severe threat situations, and it is unlikely that the pilot would have the opportunity to make reasoned tactical maneuvers with more than two or three threats. Subjects were instructed to respond as quickly as they could. Since both speed and accuracy were of interest, subjects were encouraged to respond as soon as they thought they knew whether the target was in the set or not, and not wait until they were sure. At the end of every 54 trials subjects were given a short break. Each 432 trial session lasted approximately 90 minutes.

Half of the trials were positive, half negative. Each symbol served as target an equal number of times in each of the four quadrant conditions. For positive trials, each target frame was presented equally often. On negative trials the non-target frame was chosen quasi-randomly from the set of frames not containing the target, with the constraint that each of the frames be uses about equally often.

Subjects. Sixteen non-pilot volunteers served as subjects. Subjects were student and staff volunteers as well as paid volunteers from the NASA - ARC subject pool who received $9.00 each for their participation. All subjects had normal or corrected normal vision.

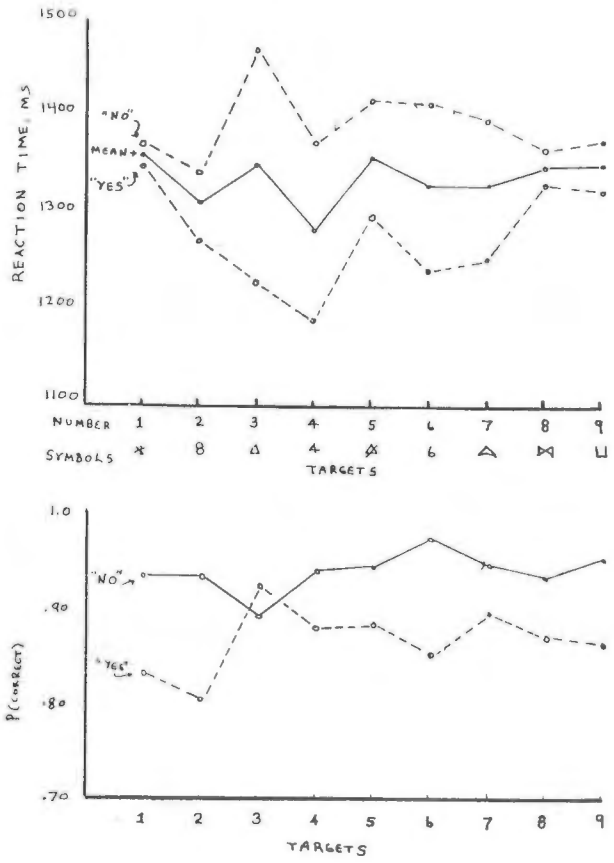
Results

A preliminary analysis of variance showed no main effects of key assignment nor any interactions of key assignment with other variables, so the results were pooled over key assignment. An analysis of variance on mean reation times in each condition showed main effects of targets ($F[8,15] = 5.521$; $p < .001$), quadrant ($F[3,15] = 10.612$; $p < .001$), and positive/negative trial type ($F[1,15] = 45.639$; $p < .001$). There were also significant interactions of target and quadrant ($F[24,15] = 3.453$; $p < .001$), target and trial type ($F[8,15] = 13.298$; $p < .001$), quadrant and trial type ($F[3,15] = 5.424$; $p < .01$), and a three-way interaction of target, quadrant, and trial type ($F[24,15] = 5.11$; $p < .001$). An analysis of variance on the arcsin transforms of the proportion correct for each subject in each condition (Myers, 1971) showed similar effects, expect that there was no significant main effect of targets nor an interaction of quadrant and condition.

The top of Figure 1 shows the mean reaction time, and reaction times for positive and negative trial types plotted separately for each target. Reaction times on positive trials represent the time taken to find the indicated target in the set, while reaction times to negative trials reflect the time to correctly indicate that the target was not in the set. A Duncan's range test showed that the main effect of targets resulted from significantly faster reaction times to target 4 than all other targets except target 2. Target 2 was significantly faster than all remaining targets except targets 6 and 7. No other differences were significant. The symbols arrange themselves into three groups with symbol 4 being the fastest, symbols 2, 6, and 7 being about 34 ms slower, and the remaining symbols being an additional 24 ms. slower.

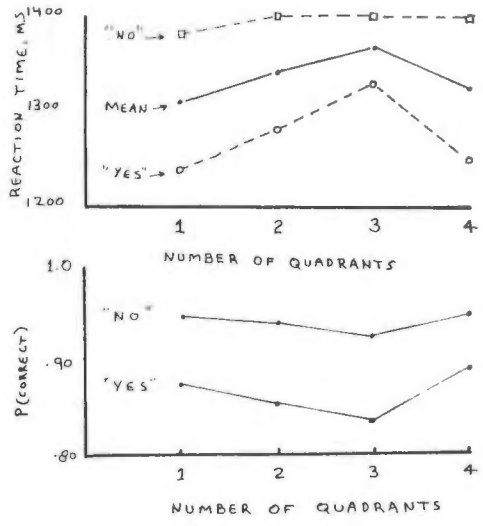
The bottom of Figure 1 shows proportion correct for each target on both positive and negative trials. There were more misses than false alarms. The average percent correct for positive trials was 86.6, for negative trials 93.7. There is no indication of a speed-accuracy trade-off. The significant interaction of trial type and targets reflects the fact that there were no differences in proportion correct for different targets on negative trials, but a Duncan's range test shows that for positive trials targets 1 and 2 were missed significantly more often than all others, while target 3 was missed less often.

FIGURE 1



The top panel of Figure 2 plots reaction time for each quadrant separately for positive and negative trial types along with the mean. There was no effect of quadrant for negative trials. For positive trials there was a steady increase in reaction time as the separation of the figures is increased, until the 4-quadrant condition where reaction time was unexpectedly fast. The bottom of Figure 2 shows proportion correct for each quadrant separately for each trial type. The advantage for the 4-quadrant condition on positive trials was not due to a speed-accuracy trade-off, since subjects were better as well as faster in that condition.

FIGURE 2



Discussion

Experiment 1 revealed potential problems with th symbol set. If only mean reaction times to targets are cor sidered the symbol set looks reasonably uniform. The rang of the means is only about 80 ms. An analysis of positiv and negative trials, however, reveals a potential probler with symbol 3, the triangle. Positive trials reflect the tim to find the designated target; correct responses on negativ trials reflect the time to reject all of the distractor symbols Fast reaction times for positive trials coupled with lon reaction times for negative trials could reveal a bias towar seeing a particular figure. This is the pattern shown by sym bol 3, the triangle. On positive trials it is one of the fastest second only to symbol 4. On negative trials, however, it i by far the slowest. When looking for the triangle, then, sub jects had great difficulty in rejecting the distractor symbols The accuracy data support this. Subjects made more fals alarms on the triangle than other symbols, but had a highe proportion of hits than any other symbol. Operators woul tend to mistake the threat designated by the triangle fo the one actually displayed, especially under time pressure It is important to note that similarity ratings or confusio matrices might have shown symbol 3 more confusable tha others, but that wouldn't have translated easily into a realis tic performance score. By using a visual search task tha captures salient aspects of the display's use the informatio can be directly related to aspects of performance.

Many of the effects of Experiment 1 conform to result from the psychological literature for search and matchin tasks. Responses to negative trials took longer as would b expected since on the average more symbols must be exam ined than in positive trials. The familiar numeric character were responded to more quickly than the less familia graphic characters. This speed advantage does not resul from subjects trading speed for accuracy since, with th possible exception of target 2, there is no difference i accuracy between the numeric and graphic characters.

The effects of separation determined by the quadran manipulation showed an unanticipated outcome. It wa expected that the time to find a target would increase a the separation increased. This was true for quadrant condi tions one through three, but the four quadrant conditio was unexpectedly fast. In this condition, symbols in eac frame were arranged one in each quadrant. This woul result in a more symmetric arrangement than other qua drant conditions, which could have facilitated search.

Experiment 2

Experiment 1 tested only the symbols themselves. T provide a more complete and representative test Experi ment 2 examined subjects' ability to locate target symbol in the presence of the markers shown in Table 1. Sinc these markers will appear in conjunction with the targets i the actual threat situations it is important to know ho much they will interfere with target detection. The additio of markers was expected to act as visual noise, makin visual comparisons more difficult. The target always occurred in conjunction with a marker, but was presented i isolation when given as the standard at the beginning o each trial. Thus, the similarity of target to standard is reduced. These conditions favor those symbols with robust internal representations. Numerals are very familiar and seen in under many different font and visual conditions. The graphic symbols, on the other hand, are distinguishable, but rarely, if ever, encountered. The exception to this, symbol number three (triangle), is very similar to symbol five and whatever familiarity advantages it may have may be offset by this similarity. There were indications in Experiment 1 that subjects had problems with these two symbols. Hence, the addition of markers should have less detrimental effect on the numeric symbols than on the graphic symbols.

Method

Stimuli and Apparatus. The same stimulus frames from Experiment 1 were used. The frames were modified to include the designated marker for each target symbol and any other markers required in that frame. The assignment

of distractor symbol to marker was random within each frame.

Design and Procedure. The design of Experiment 2 was not completely counterbalanced with respect to all the variables in Experiment 1, and reflects the primary concern of assessing the effects of target-marker combinations. Each target was paired equally often with each marker. Every target always occurred with one of the four markers, so that the subject had to identify the target on positive trials disregarding one of the four markers. Since each target occurred three times in each quadrant condition it was not possible to completely map the four markers into each target-quadrant condition. The assignment of markers to distractor symbols was done randomly with the constraint that each marker serve as distractor about equally often, and that there be an equal number of frames with 0,1, and 2 markers on the distractor items.

Subjects. Sixteen new non-pilot subjects were selected from the NASA - ARC subject pool and student and staff volunteers.

Results

As in Experiment 1 there were no effects of key assignment and the data were collapsed over that condition. An analysis of variance on mean correct reaction times showed main effects of targets ($F[8,120] = 17.01$; $p < .0001$), markers ($F[3,45] = 58$; $p < .0001$), and trial type ($F[1,15] = 117.13$; $p < .0001$). All interactions were significant ($p < .0001$) with the interaction of markers and trial type being the weakest ($p < .02$).

Figure 3 plots the mean reaction time to each target along with reaction times to positive and negative trials for each target. The bottom of Figure 3 shows the proportion correct for each target as a function of trial type. As expected, targets 3 and 5 were very difficult for subjects, especially on negative trials. The pattern seen in Experiment 1 is amplified here. Accuracy data shows the same effects as the reaction time data.

FIGURE 3

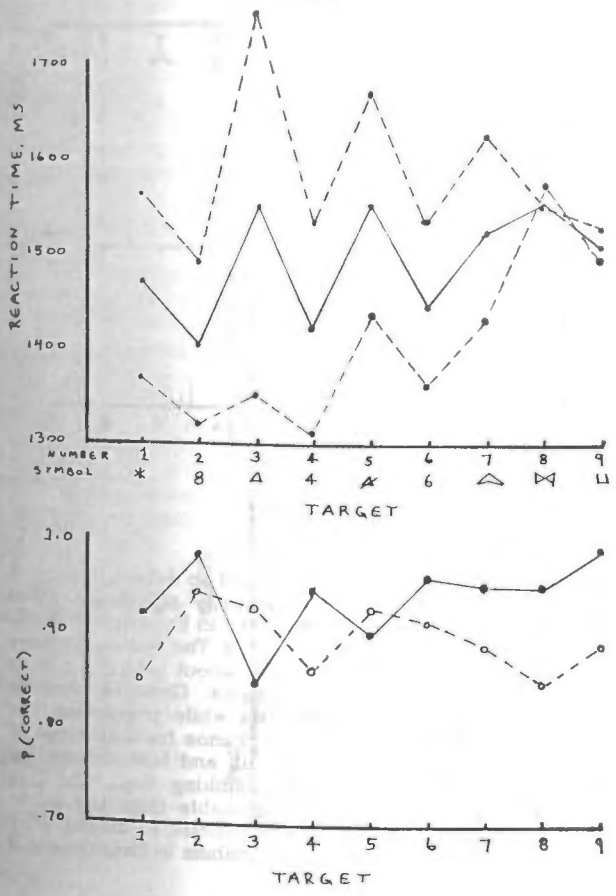
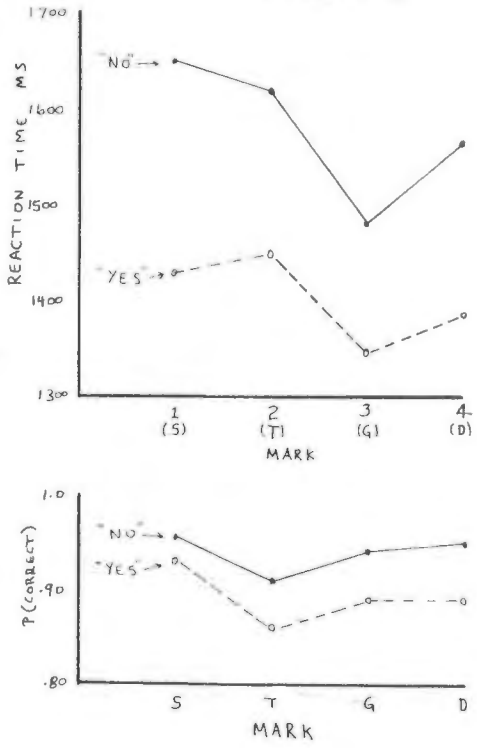


Figure 4 plots reaction time for each mark separately for positive and negative trial types. The bottom panel of Figure 4 shows proportion correct. Though there is a significant interaction in the reaction time data between mark and trial type it is clear that both follow the same pattern with respect to the different mark conditions. When reaction time for each mark is plotted for each target, as in Figure 5, the complex effects that lead to the interaction between target and mark can be seen. In general, the square and tail markers are more disruptive than the others; the ghosted targets being somewhat more easily recognized. The square and tail affect all targets about equally. The magnitude of the effects of the ghost and dot, however, depend more on the specific symbol.

FIGURE 4

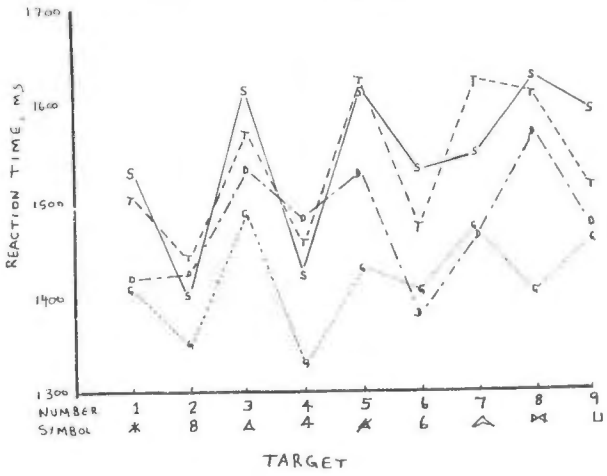


EXPERIMENT 2
MARK BY TRIAL TYPE

Discussion

The addition of visual noise, in the form of the marker conditions, increased overall mean reaction time and differentially affected the times to find or reject different targets. The general result was that the difficult cases noted in Experiment 1 were affected more than the easier conditions. This is especially clear in the case of symbols 3 and 5. These were troublesome for subjects in the first experiment, and the same performance patterns are amplified in Experiment 2. That the 4 markers affected individual symbols to different degrees doesn't change this conclusion. With the exception of the good performance of symbol 8 in the ghost condition and the poor performance of symbol 4 in the dot condition, symbols 3,5,7,8, and 9 are consistently poorer than symbols 1,2,4, and 6. The former is the set of graphic symbols used; the latter is the set of number symbols and the asterisk, a common lexical character.

FIGURE 5



EXPERIMENT 2
MARK BY TARGET

Experiment 3

The marker conditions provide important status information about other aircraft. An additional test was required to determine whether the subjects could quickly and reliably identify the marker associated with a target symbol. This procedure would also provide information about how performance would change when a source of difficulty other than visual interference was manipulated. Experiment 2 made the discrimination of visual symbols more difficult, Experiment 3 makes response selection more difficult, since the number of markers is four rather than the two-alternative yes/no response of the previous experiments. Visual discriminability plays only a part in the contribution to total response to a threat display. The selection of an appropriate action also contributes. If the response selection component is independent of visual aspects of the display then the response times for each symbol in Experiment 2 should be increased by a constant factor in Experiment 3. If the time to select and execute a response depends on the discriminability of the display items, then increasing the difficulty of response selection will change the relative reaction times for each symbol in Experiment 2.

Design and Procedure

In Experiment 3 the target symbol appeared on every trial, and the subjects' task was to identify which of the four markers was paired with the symbol. The same frames used in the previous studies were used. Subjects no longer sat in a sound isolated chamber, but were in a darkened room with the monitor at the same distance as before. The procedure was identical to the earlier experiments except that subjects were to press one of four micro- switches to indicate which of the four markers was associated with a given target. The assignment of the four keys to the four markers was determined by latin square arrangement, so that each target was associated with each key equally often. In all other respects Experiment 3 was identical to the previous experiments. Twenty volunteers served as subjects.

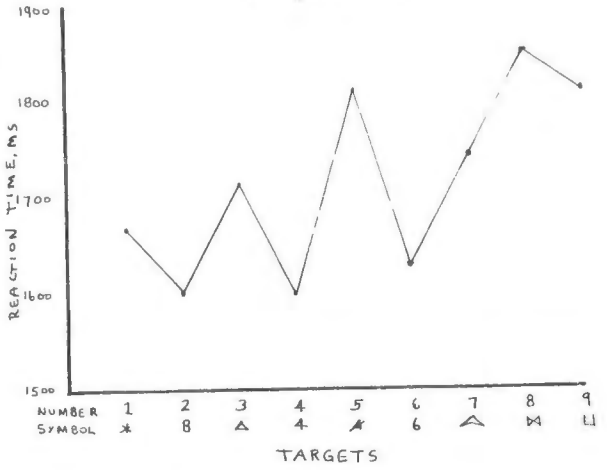
Results

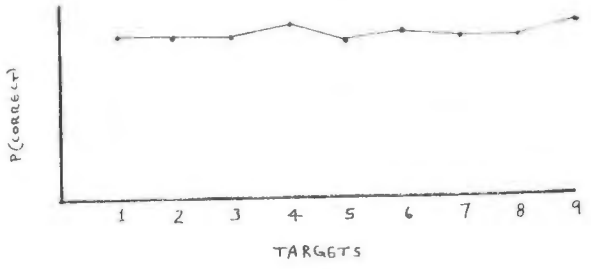
The data were collapsed over the response mapping condition and an analysis of variance was done with targets and markers as within subjects variables. The only significant main effect was target ($F[8,144] = 31.1$, $p < .001$). There was no main effect of markers nor any interaction between markers and targets. Figure 6 shows mean reaction time for each target. These mean values are highly correlated with mean target reaction times from Experiment 2 ($r = .83$), and only moderately correlated with the time from Experiment 1 ($r = .53$).

Discussion

Experiments 2 and 3 require different require subject to extract different information from the same display. I Experiment 2 the markers were not informative and had t be ignored in deciding whether a symbol was the target Experiment 3 requires a similar identification, but havin found the target subjects must then disregard the targe and respond to the markers. The longer reaction times i Experiment 3 reflect not only the increased difficulty c response selection, but the more complicated decision pro cess as well. Yet, this increased difficulty affected all tar gets equally. The pattern of reaction times across targets i due to the targets and the respective frames, not to an response or additional decisional complexities. The reactio time on each trial can thus be considered to have at leas two independent components: search time and respons selection and execution.

FIGURE 6





EXPERIMENT 3

Ghosting tends to be more difficult to detect in Expe ment 3, though this is not statistically significant. T trend is interesting since the tendency in Experiment 2 w for the ghost to be more easily ignored. The symmetry he is appealing, and it reveals something about both the effe of the markers and subjects' strategies. Ghosting consi of deleting selected figure elements while preserving t shape of the symbol. There is less chance for distortion masking than with the square or tail, and less chance diverting attention than with the blinking dot. Ghost would leave the target more recognizable than the oth markers, but would, itself, be more difficult to detect. W then was reaction time to ghosted symbols in Experimen

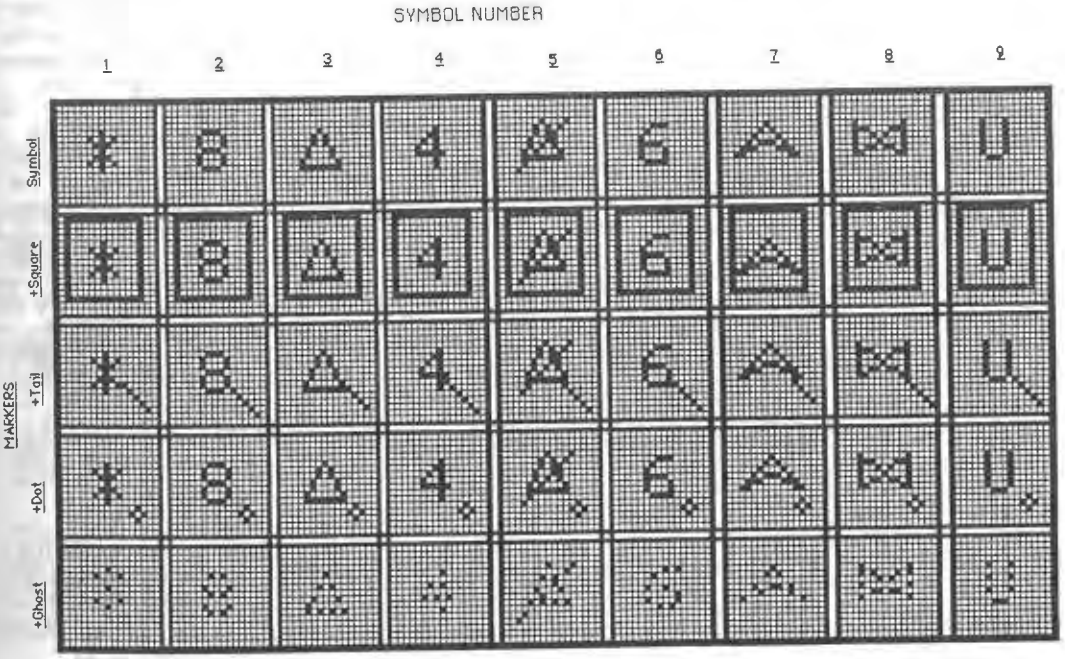
higher the reaction time to non-marked targets in Experiment 1? Since targets always had markers in Experiment 2, subjects could profit by searching for markers. The number of markers in a frame had a strong effect on the reaction time. Thus, we hypothesize that the greater difficulty in recognizing the ghost as a marker offset its advantage in preserving the shape of the symbol. that could lead to advances in display design procedures.

Summary

Three experiments examined search performance with a small situation display on which digits and graphic symbols designated other aircraft. The time required to locate and identify the individual symbols showed that the graphic symbols were in all cases inferior to the numeric stimuli. Increasing the difficulty of the visual discrimination magnifies the differences between graphic and numeric symbols, while increasing response difficulty uniformly increases response times. Unless the shape of the figure provides some elegantly coded multidimensional information about the symbol, or the symbol set is too large to capture by letters and numbers, highly familiar characters, such as letters and numbers, will yield better performance scores. Factors such as familiarity and categorization which have large effects on reaction time and retention in many psychological tasks, seem also to be influential in finding and identifying symbols, at least on the class of display investigated here.

There is a clear need for quantitative models of similarity that can be applied to the design of displays and the selection of symbology. Such models would provide the designer with a means of evaluating candidate symbol sets without the need for experimentation, thus enabling rapid prototyping of display formats. Quantitative similarity models exist, but are typically based on similarity ratings, not on performance measures. It would be useful to generalize the the existing similarity models either by extending their application to reaction time data, or establishing a firm connection between similarity judgements and operational performance.

TABLE 1



84-2617

ARTIFICIAL INTELLIGENCE IMPLICATIONS FOR ADVANCED PILOT/VEHICLE INTERFACE DESIGN

Kenneth J. Maxwell
James A. Davis
General Dynamics/Fort Worth Division
Fort Worth, Texas

ABSTRACT

The impact on pilot/vehicle interface (PVI) design for fighter aircraft from the introduction of artificial intelligence (AI) technology is discussed. Three prototypical models (pilot manager/AI associate, pilot/AI colleague, Autonomous assistant) of the operational relationship between the pilot and AI systems are defined. These models provide a structure in which PVI issues are discussed. Issues involving the resolution of possible disagreements between the pilot and the AI system, intelligent presentation of information including an intelligent interrupt capability, and natural language interaction are discussed. It is concluded that the introduction of AI into the aircraft will have a major impact on PVI design.

INTRODUCTION

This paper discusses potential changes in pilot/vehicle interface (PVI) design that the authors anticipate as a result of the use of artificial intelligence (AI) in onboard aircraft systems. Although the discussion will be limited to PVI design for fighter aircraft, many of the design issues are applicable to PVI design in general. To appreciate the design implications entailed by AI applications familiarity with current PVI design (without AI) is necessary.

The pilot/vehicle interface is that portion of the cockpit weapon system that acts as information and contral center The cockpit mediates the transmission of data and the presentation of information, control commands, and messages between the pilot and the aircraft, Figure 1. Modern fighter aircraft incorporate an array of sophisticated electronic systems that increase the capablility of the aircraft to operate in severe tactical environments. Today operational control of these systems is left primarily to the pilot, and as a result, the operational demands imposed on the pilot have increased along with the complexity of the onboard systems. Pilot/vehicle interface (PVI) designers have attempted to compensate for the greater volume of available information and the additional control requirements imposed by these systems in several ways, four of which are briefly discussed below.

First, by employing multifunction displays that present only a subset (functionally relevant) of the available information to the pilot at any given time. This facilitates the acquisition of relevant information and inhibits the acquisition of irrelevant information. Second, by packaging functionally related information together on a single display format (i.e., integrated display concept) in a manner that facilitates the acquisition of the information and important relationships and interactions among information from different systems. Third, by designing automated systems to function with greater autonomy from the pilot. Thus, the pilot would presumably be required to monitor the system only intermittantly, rather than actively control it. To the degree that the monitoring task is less demanding than performing the control task,

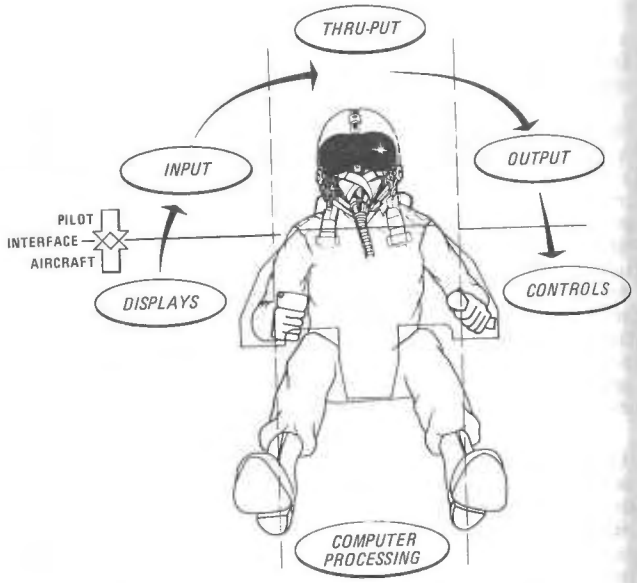


FIGURE 1. REPRESENTATION OF THE PILOT/ VEHICLE INTERFACE.

automation of the task should benefit the pilot. Fourth, by designing systems which are functionally integrated with one another. System integration allows the pilot to control the system at a level that requires less attention to the details of each individual subsystem's operation.

The discipline of artificial intelligence (AI) has developed a technology base that, although still emerging, seems promising for increasing the capability of the aircraft beyond current levels without a corresponding increase in the demands placed upon the pilot. This is the case, because programming techniques developed by AI researchers are able to produce a machine-based intelligence. Thus, functions that are currently performed solely by the pilot, can become the province of the machine. Simulation of these intelligent pilot abilities will allow some aircraft systems to function more autonomously than they could using only conventional automation techniques. Also, higher levels of functional integration in the weapon system are currently performed exclusively by the pilot. That is, one of the pilot's functions is to coordinate and integrate the capabilities of the individual systems and subsystems. Integration of functional capabilities is currently being performed for some systems, but the pilot is still the high level functional integrator. Artificial intelligence offers the possibility that higher level functional integration can, at least in part, be performed by the machine.

In order to realize the potential benefits from systems that can operate with increased degrees of autonomy and at higher levels of functional integration, it is necessary for AI programs to have the ability to communicate more

abstractly, with greater degrees of flexibility with the pilot, (reference 1). This ability to communicate successfully at the most efficient level and within the time constraints of the task is one of the primary limitations toward facilitating cooperative pilot/AI problem solving.

Consider how the pilot currently interacts with the aircraft. Since the various functions that need to be performed are not currently integrated to a great degree, the pilot performs this integration internally. That the integration is performed, is manifested in the performance of a task that requires the coherent action of several functions. Currently, since aircraft systems do not perform the integration, information is available to the pilot from an array of single sources of information. Integrated display concepts facilitate higher level tasks by presenting functionally related information from different sources together on a single display format. However, compiling information from low level systems on a single display does not necessarily provide the pilot directly with the information needed to perform higher level tasks. The pilot performs complex tasks and is concerned with high level problems, but must instruct his aircraft with relatively basic commands.

The advent of the application of AI techniques to real-time applications has been recent. Consequently, little research has been performed investigating how such machine capabilities should be integrated into the aircraft. Many questions need to be answered before computer programs will be able to successfully interact with the pilot in a way that facilitates, rather than burdens the pilot.

The design of a PVI can be described at different levels. At a hardware level, the PVI includes the devices employed to display information and to control the aircraft and its systems. The physical distances to these devices and their spatial layout in the cockpit are also included at this level. Other levels of description include the way in which information is formated on the display surfaces, the conditions under which information is displayed, what type of information can be accepted as commands and queries from the pilot, and the type of information that is displayed. The design issues concerned with what type and when information is presented and accepted by the PVI are included in an information level of description.

PILOT/AI RELATIONSHIP MODELS

It is possible that the introduction of AI into the system will require some form of hardware modifications to the displays and controls. But, it will be the information level of the PVI that will receive the greatest degree of modification due to AI systems. Therefore, this paper will restrict itself to changes at the information-level. Changes at the information-level of the PVI reflect changes in the relationship between the pilot and the system. These changes in relationship are primarily due to the increased information management capability of the system when incorporating AI techniques. Three basic types of relationships between the pilot and an intelligent system are outlined below. These pilot/AI relationships each differ from the current relationship that exists between the pilot and the aircraft in that they each involve a pilot/AI system interface. This difference is exemplified in Figure 2. Each of the three models describes a different form of cooperative problem solving between the pilot and the AI system. The responsibilities of the pilot and AI system change across the three models. It is expected that different operational problems and tasks will require different forms of cooperation which are represented by the three models and these combinations.

The Pilot Manager/AI Associate Model. There are several ways to envision the relationship between the operational roles of the pilot and the machine-based intelligence. One of these relationships is referred to here as the pilot manager/AI associate model. In this model, the pilot performs as system manager primarily at outer control loops. That is, the pilot directs, and the AI associate performs as directed. The AI associate buffers the pilot from the lower-level functions that he performs today. The model stereotypes the pilot as a supervisor of systems. The model sterotypes the AI associates's role as providing the underlying support mechanisms to allow the pilot to consent rate his attention at the outer control loops by performing lower level functions.

This model is most appropriately applied to operational situations in which the tasks can be partitional hiearchically. The pilot performs the top end of the hierarchy while the AI system performs the bottom.

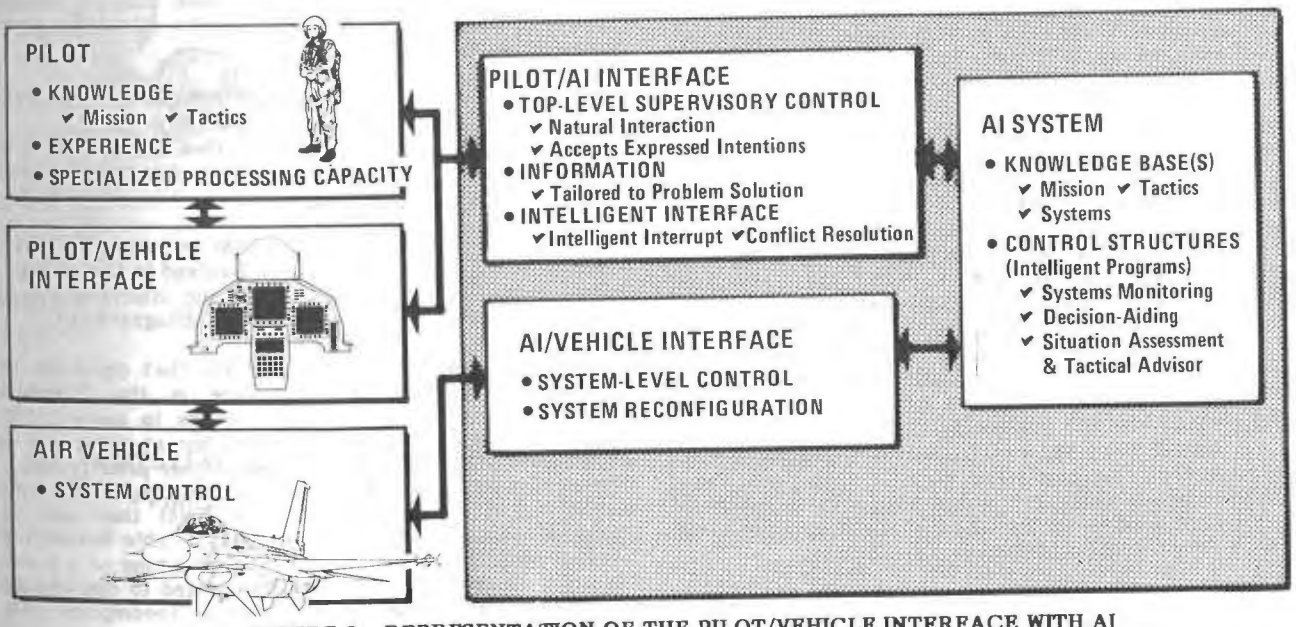


FIGURE 2. REPRESENTATION OF THE PILOT/VEHICLE INTERFACE WITH AI SYSTEMS INCLUDED.

The Pilot/AI Colleague Model. The pilot/AI colleague model differs from the pilot manager/AI associate model in several respects. Instead of supervising the AI systems, the pilot works with an AI colleague on a relatively equal buses and with equal responsibility. Each monitors the other, and each contributes in a variety of ways to the performance of operational tasks. This model is most appropriately applied to operational situations in which the tasks are not easily partitioned into a hierarchy, or in which partitioning into a hierarchy (although possible) is not as efficient as other partitioning schemes.

The Autonomous Assistant Model. This model differs from the other two in that it is the AI system has actual control of the aircraft. The pilot's role is to augment the machine's capability, if desired. This model is primarily appropriate for situations in which the pilot has become incapacitated to a degree, and is unable to function in either of the other defined roles.

These three models describe different levels of pilot and AI system responsibilities and manner of functioning. Each of them entail problems for the designer of the PVI. Supervisory control requires a different form of interaction than does pilot/AI colleague control or autonomous assistant control. The models described above are not meant to be mutually exclusive. They are described as prototypes of the kinds of roles the pilot may be placed in when working with intelligent machine systems. Actual operational situations will not be described purely as a single model. The particular blend of the models in an implemented design will depend on the particular characteristics of the tasks being performed and the problems being solved. Different situations will require different types of intelligent cooperative relationships. It is expected that real intelligent interface systems will incorporate aspects from all three models.

The models provide a framework to describe possible changes in PVI design. That is, a framework for describing what type of interactions are required to support those three distinct roles cooperative relationships.

PVI DESIGN IMPLICATIONS

Implications for PVI design from the use of AI spawn from general issues and problems that are created by the existence of the capabilities AI brings to the system. These capabilities include a greater degree of autonomous functioning and the ability to perform functions previously performed exclusively by the pilot.

The Disagreement Issue. One problem created by the AI system is that in the course of performing a task the pilot and the AI system may come into disagreement as to what the best course of action should be. How is this disagreement to be resolved? One approach, used in expert systems today, is that by backtracking through its logic the system can explain its reasoning. Presumably, by studying the rules the AI system used to reach its conclusion a resolution to the disagreement can be obtained.

The time constraints of many tasks during the missions flown by fighter aircraft make such a solution to the disagreement problem untenable. There is simply not enough time to have the system present a detailed history of its reasoning for the pilot to inspect. Such a solution is also undersirable because it works in opposition to one of the primary purposes for considering AI applications in this domain area; namely reduction in pilot workload.

The question is not one of who has ultimate responsibility or control. In all cases, the pilot will be totally responsible for the conduct of the mission and will have the ability to override the AI system. However, an interesting exception to this is the case where the pilot may be unconscious or in some other manner incapable of exercising control. The question is what is the pilot to do when the system disagrees with him? It is worth noting that having the AI system in disagreement with the pilot is not always an undesirable situation. In addition to reducing the amount of some kinds of work the pilot has to do, a major contribution of AI is that the system will enhance some pilot judgements by considering a wider range of information or a larger set of original alternative actions. If the pilot and AI system always agreed, this contribution would not be realized.

Without assistance in resolving the disagreement, the pilot is in a quandry. If he tries to determine why the disagreement exists, he is using up valuable time and may in the end determine that he was right all along. On the other hand, he may determine that the AI program is correct, but the opportunity for the action may be lost due to excessive deliberation. In short, if time is imperative, as it often is, the pilot may decrease his degree of success simply by using up valuable time determining if he or the AI program is correct. However, if he does not determine which conclusion is best, his degree of success will also decrease due to choosing the best conclusion only a portion of the time.

In the Pilot/AI colleague model the pilot is participating actively in the decision making process at lower levels than in the pilot manager/AI associate model. This means that in the pilot manager model the pilot is less likely to be in a position to quickly evaluate the AI programs low-level decision making than when he is participating as a colleague. Thus in the pilot manager/AI associate model the disagreement problem would seem to be more severe when it occurs. However, it is also the case that the disagreement problem is less likely to occur in the pilot manager model.

This is the case, because when acting as a manager, the pilot is not interacting with the AI program at lower levels of processing. Thus, disagreement at lower levels doesn't occur in this model. Disagreement is manifested at the higher levels of processing reflecting higher levels of functional integration. This problem, although less frequent, is more critical in the pilot manager model because from a supervisory position the pilot has little chance of tracing its locus.

In the Pilot/AI collegue model, disagreement is more likely to occur because the pilot is more involved in the processing. But it is also the case that the pilot is in a better position to trace the locus of the disagreement due to his active involvement.

The solution for this problem then has two aspects. First, the pilot should remain as actively involved in the problem solving activity as he can. Second, the interface should provide a means for timely resolution of disagreement.

The Interrupt Issue. Another problem that designers of intelligent PVI interfaces will face is the interrupt problem. Actually, this problem exists in current PVI designs in terms of presenting information at times and in ways that not distract the pilot from higher priority tasks. Rather than creating this problem, AI provides a means to resolve it at a more sophisticated level than can be achieved today. The AI technology may be able to monitor the tasks and functions the pilot is performing at a given time and decide how information presented to him should be prioritized. The programs that accomplish this prioritization will require knowledge about the mission, the intent of the pilot, the priority of objectives, and how tasks

relate to the accomplishment of objectives. Figure 3 depicts a situation in which the AI system decides to interrupt the pilot to inform him of a fuel leak problem and advises him that a return to base action is in order.

In the pilot manager/AI associate model, a program that handles interrupts, and the more general case of displaying the most appropriate and relevant information to the pilot at the right time, must account for the difference in level between the tasks the AI systems are performing and the tasks the pilot is performing. In the pilot/AI colleague model this difference does not exist (at least not to the same degree). The difference in level of tasks may make the interrupt problem at the supervisory level more difficult because in the supervisory role the pilot must rely more heavily on what the system tells him. Thus, the prioritization of information may be more critical.

As indicated above, the interrupt issue is a part of the more general issue that includes the rules for filtering and presenting information to the pilot. Artificial intelligence can be used to create an interface that acts as an information coordinator. This coordinator will receive information from the onboard systems, determine when it should be presented to the pilot, and possibly determine how it should be presented to the pilot. The information coordinator will make these determinations by monitoring the pilots activities, referring to its knowledge about the mission objectives, and noting what information the pilot has explicitly or implicitly requested.

The Natural Language Issue. The full development of an information coordinator will rely heavily on the ability to communicate efficiently with the pilot. A major issue for the design of intelligent communication is the use of natural language dialog as a medium for the interaction. Such a medium would allow necessary flexibility in the interaction. It would allow the high level commands to be issued by the pilot, the character of which would indicate the "intention" of the pilot. The AI system would then be capable of interpreting these high-level imperatives of intention in terms of the lower-level tasks and functions that needed to be performed in order to achieve the desired goals. Figure 3 depicts two aspects of the use of rational language. First, the pilot can direct the system with high level expressions of intent. Second, the pilot can query the system for necessary information.

This "intent-driven" aspect of the intelligent PVI is best viewed from the context of the pilot manager model. In this case the pilot is not concerned with attending to the details of particular functions. His attention is focused on the higher level tasks. This is not to say however, that intent would not be used to communicate at the colleague model level.

Note that in a dialogue, intention is expressed in two directions. This bidirectional flow of intent can also be designed into the intelligent PVI. Bidirectional flow of intent can occur in each of the models, but is probably most important in the pilot/AI colleague model in which there is much interaction at the same level of functioning.

The Role Transition Issue. The use of AI in the aircraft and in PVI design concepts will provide the pilot with alternative ways to interact with the system. These alternatives are specified by the three prototypical pilot role models. The applicability of these models will vary

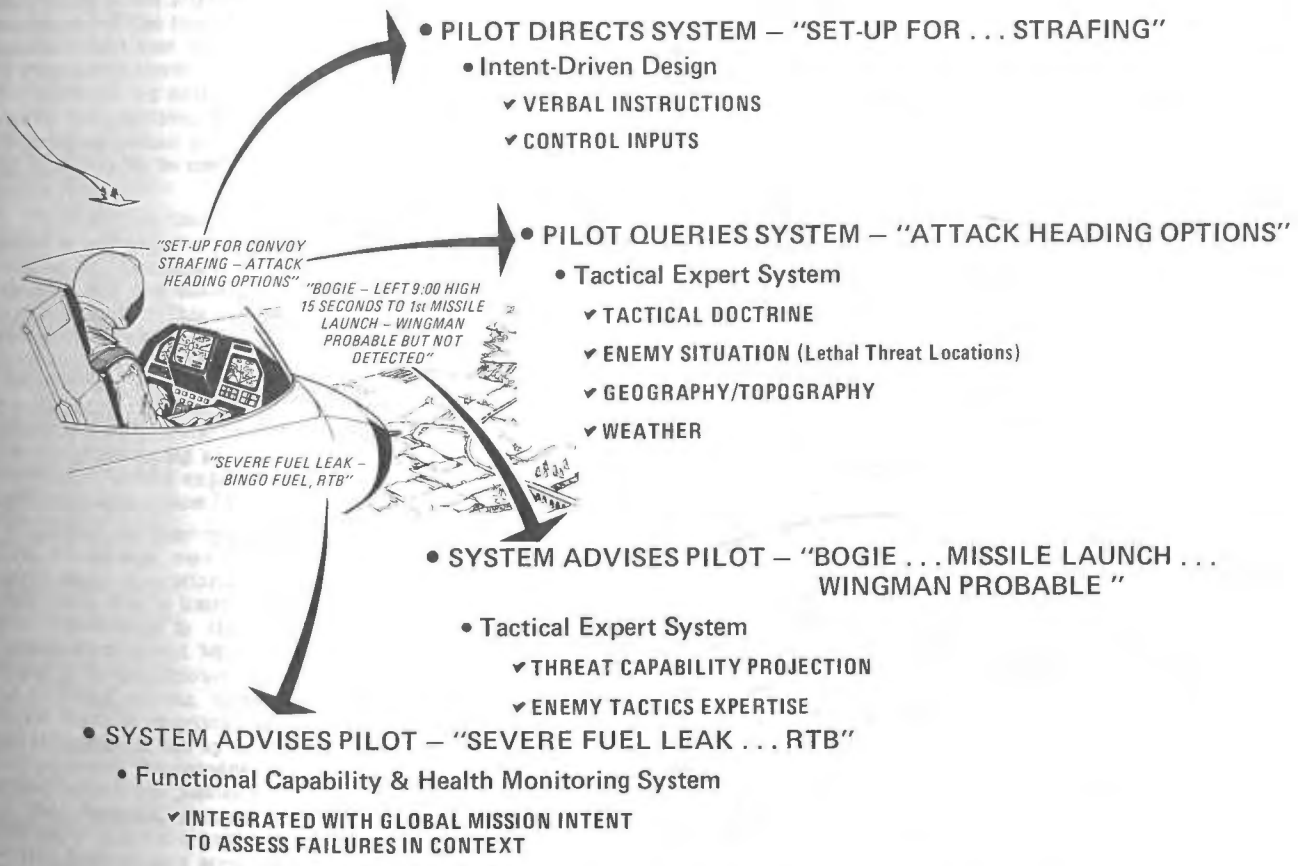


FIGURE 3. REPRESENTATION OF SOME ASPECTS OF NATURAL PILOT/AI DIALOGUE.

with different operational tasks and tactical situations which creates a problem in shifting the role of the pilot from one situation to the next. That is, for part of the mission the pilot may best function as a manager, while during another part may have to function as a colleague. The PVI must be able to provide a graceful transition for the pilot between the two roles.

Since real systems will not purely reflect the roles depicted in the models, transitions will be occurring among more than two states. The transition between role states may profit from analysis in terms of different levels of functional integration. In a sense, different levels of integration in the system present a finite number of role states that are more finely categorized than the three role models defined above. For example, the pilot manager model really demands that the underlying system be highly functionally integrated. Otherwise, the PVI could not allow him to exclusively operate at the outer control loops, because there would be no underlying support structure to perform the lower level functions.

AN APPROACH TO INTELLIGENT PVI DESIGN

The discussions above have focused on the alternative roles the pilot may assume when interacting with an AI system, and the issues that must be confronted by the designers of interfaces that support those alternative interactions. One approach that would aid the design process is to classify a set of pilot-role/design-issue pairs that would provide a framework for describing the variety of design situations with which the PVI designer must deal. A design situation is referred to here as the set of circumstances that characterize the interaction between a particular pilot-role model and a particular design issue. A more detailed analysis of how each design issue affects each of the alternative cooperative roles would provide a taxonomy of these design situations. What this means is that the design issues and the alternative roles interact with one another. Therefore, a description of this interaction is necessary to cover the range of possible cooperative pilot/AI relationships appropriate for the desired applications.

Design specifications for each of these design possibilities can then be formulated. These specifications would, of course, be stereotypical in nature because they would be derived from a set of generic design situations. Finally, operational tasks can then be matched with the design situations to which they most closely relate. At this point,the set of stereotypical specifications for that design situation can be refined for the particular task.

This design approach focuses on the relationships among: (1) the role of the pilot, (2) design issues, and (3) operational tasks. It emphasizes designing the interface with reference to these three factors and their interactions. It does not however, address the problem of moving gracefully from one design situation to another.

In conclusion, the use of AI in fighter aircraft systems will offer many alternative roles for the pilot and relationship with the aircraft. These alternative roles can be stereotyped and analyzed in terms of generic issues related to intelligent interfaces. It is proposed that the interactions among the design issues and the alternative roles be investigated with the purpose of developing general specifications for the set of pilot-role/design issue combinations (design situations). These design situations will form a framework that can be used to match and structure the specifications for individual operational tasks.

References

(1) H. Tanaka, S. Chiba, M. Kidode, H. Tamura, T. Kodera, Intelligent Man-Machine Interface. T. Motoka (Ed.), (Fifth Generation Compu Systems." New York: North Holland, 1982, 147-15

MODEL-BASED REASONING IN EXPERT SYSTEMS: AN APPLICATION TO ENROUTE AIR TRAFFIC CONTROL

Captain Stephen E. Cross, PhD
Artificial Intelligence Laboratory
Department of Electrical Engineering
Air Force Institute of Technology
Wright-Patterson AFB, Ohio 45433

Abstract - A working computer program, an air traffic control expert system, uses qualitative reasoning to justify heuristically generated plans. The justification is based on a model of aircraft performance which is computationally too complex for use in the normal planning process. Equations are represented in a semantic network. The reasoning algorithm is based on constraint propagation.

I. Introduction

Pilots often 'talk with their hands.' During a mission pre-briefing, a fighter pilot will use hand gestures to indicate how individual aircraft are to approach the lead aircraft, fly in formation, and accomplish other mission related goals. In essence, the pilot is justifying plans for specific mission goals using sensitivity analysis based on a naive understanding of the aircraft equations of motion. This type of reasoning is common to many human problem solving domains. For instance, the driver of an automobile can justify why one would decrease gears while climbing a hill. Air traffic controllers justify their plans and the plans of other controllers (including novel plans which may be more elegant than their own) through a similar reasoning process. A common aspect is that the physics of the machine or machines being controlled by the human can be modeled with mathematical equations and that humans understand justifications based on their naive physics understanding of the equations. In the research reported here, a reasoning capability, based on constraint propagation, is defined that is similar to that observed in human problem solving. The reasoning capability is shown to be useful in the justification of plans of an expert system.

The man-machine communication task is dependent on shared knowledge between the man and the machine. In his book on man-machine interface modeling, Rouse [1] discusses the need for the machine to be equipped with models of human behavior. This may include the representation of the human's naive knowledge of some domain (e.g., physics). In the research described here, a computer equipped with a naive physics understanding of Newton's laws can translate domain equations into knowledge that is meaningful at a heuristic level. The resultant representation and an automated reasoning process based on constraint propagation forms part of the explanation capabilities of an air traffic control expert system [2].

Expert systems typically utilize a declarative and uniform knowledge representation (Stefik [3]). The approach offers many operational advantages (e.g., a simple control structure), but is limited to expressing an expert's surface level knowledge in the form of pattern-decision pairs (-Chandrasekaran and Mittal [4]). The computer should have access to 'deeper' knowledge if it is to understand and justify its planning actions. Consider a domain where knowledge, in the form of equations and algorithms, is computationally too complex for use by the human practitioner. How should mathematical knowledge be represented to aid in the improvement and justification of plans? In the task domain of this research, enroute air traffic control, heuristically generated plans are justified by applying a qualitative reasoning process to a structure derived from aircraft performance equations. Equations are represented in a semantic network where nodes represent variables and links represent dependent variable influences. The linkages are determined in part by a symbolic series expansion of domain equations. The resulting structure explicitly represents the influences of each domain variable, hence a form of sensitivity analysis.

The approach is unique in three aspects. First, a level of abstraction is included. Domain equations may be computationally too complex for a human expert to use. However, the equations can be interpreted in terms of a naive representation of Newton's laws as applied to one dimensional motion thus abstracting the influences inherent in the equations. Second, the approach enables bidirectional reasoning. Qualitative knowledge can be used to direct quantitative reasoning. Additionally, when new equations are implemented, their meaning is represented explicitly and interpreted using the existing qualitative knowledge. Third, the computer constructs its own representation of the equations based in part on a symbolic series expansion.

II. Research Issues

This research is motivated by three relevant and related topics in artificial intelligence: explanation capabilities of expert systems, computer understanding, and reasoning about disparate knowledge. In the man-machine context, it is critical that the computer be able to explain and justify its solution to the human. By explanation, I mean supplying the user with sufficient information to understand what a plan or plan step means. Justification means that the computer describes the rationale of a plan or plan step. The process of plan justification may use information that is normally in the background (i.e., not used in the planning process). Often knowledge which the computer could use to form justifications is implicitly contained in a mathematical substructure. Each research issue is investigated in turn before showing a new application to justification in the domain of enroute air traffic control.

A. *Explanation Capabilities of Expert Systems*

Expert systems are often identified with production systems. A production system consists of a data base, a knowledge base, and a control structure. Typically, the data base is a centralized medium in which pertinent data is stored. The knowledge base is a data base of production rules (representation of heuristic knowledge) and the control structure defines how rules are chosen. Rules can be chosen to satisfy a goal (e.g., goal driven) or in response to data (e.g., data driven). The production system approach affords several advantages. The knowledge is acquired from experts in the form of 'condition/action' heuristics and production rules provide a natural representation. Since the control structure is separate from the knowledge base, rules can be added, deleted , or modified without impacting system control. The rules are logical constructs and solutions can be proven to be logically correct with respect to the current state of the knowledge base. Correctness has been an overriding concern in previous expert system research. The 'proof' serves as an explanation to the human user. This is especially useful in the performance testing phase. The production system answers questions like 'HOW' (how a conclusion was reached) or 'WHY' (why a question was asked) by reciting some portion of the rule chain that was used to achieve a goal. The advantages of the production system approach are chiefly implementational. Rule recitation is the usual expert system approach to automated explanation, but the approach does not constitute understanding.

B. Computer Understanding

The explanation capabilities of production systems do not constitute understanding. In the context of a production system, it is the user's responsibility to understand the explanation. The usefulness of the automated explanation is dependent on the ability of the user to understand the chaining among rules and on the context and clarity of the knowledge represented in the rules. These two assumptions are not always met. Clancey [5] describes how the rules of MYCIN do not capture the human expert's understanding of rules. For instance, a rule such as 'If the patient is less than eight years old, do not administer tetracycline' does not represent the causal knowledge that tetracycline can impair a child's bone development. Aikins [6] has investigated context-dependent knowledge organization of rules for medical based expert systems while Pople [7] has sought to implement reasoning strategies similar to expert clinicians.

It is argued that existing production systems are incapable of understanding their domains. Each rule contains a microscopic 'chunk' of knowledge that does not relate to the 'macroscopic' context of a domain. There is no convincing argument that humans represent knowledge in rule-form or that control structures such as forward- or backward-chaining are typical of human reasoning. The performance of the expert systems might be increased by including knowledge foreign to the human. For instance, one may wish to use detailed control algorithms that are computationally too complex for use by a human. One can imagine such an algorithm's possible but cumbersome representation in rule form. A third disadvantage concerns the domain of application. Many domains, such as natural language, are broad and do not require inferencing as deep as provided by a production system approach.

Research is language comprehension has addressed similar knowledge organization issues (Charniak [8]). Knowledge is represented in stereotypical structures called frames (or schemas, or scripts). These structures offer a natural way to partition knowledge. The structures are used to understand pieces of text. That is , the structure specifies the common-sense knowledge that is required to interpret the text. If one wished to interpret 'What color is a pear?' one would not retrieve facts like 'fire hydrants are red' (Charniak [9]). One of the major tenets of conceptual dependency theory is that any information in a sentence be made explicit in the computer's meaning representation of that sentence (Schank and Abelson [10]).

A frame representation is a necessary, but not sufficient step towards computer understanding in a task domain. The sufficiency argument is equivalent to the 'frame hypothesis' (Charniak [9]) which states that understanding an input is equivalent to finding a frame in which the input can be integrated. There are at least two problems with the hypothesis.

1. Often there is no one frame into which inputs can be integrated. There are two reasons for this. First, an applicable frame may not exist which implies that new knowledge is required. Second, the input may contain a goal interaction which cannot be handled by the present set of frames.

2. There may be a number of frames into which inputs can be integrated, but doing so is not tantamount to understanding. Consider the statement 'He painted the terminal screen white' (Charniak [9]). If one only had the 'normal' painting frame, the input sentence could be interpreted. But the computer also needs the ability to justify its interpretation. In this case, the justification would be based on world knowledge of 'painting' and 'terminal screens.' In the domain of air traffic control, at least part of the justification can be made, based on the computer's understanding of the aircraft equations of motion.

Another disadvantage to explanations in typical production systems is the opaqueness of the system's belief in its knowledge. MYCIN uses a Bayesian processing algorithm that computes an updated certainty factor based on apriori, human-specified certainty factors attached to each rule. The certainty factors do not capture the expert's rationale for belief and non-belief of each invoked rule. In contrast, Doyle [11] attaches symbolic statements of belief and non-belief to nodes in a reasoning tree. The resulting reasoning procedure, often referred to as 'data-dependent backtracking', allows the computer to construct justifications. A recent paper by Cohen and Grinberg [12], describes the use of endorsements for the representation of states of certainty. An attempt is made in this research to take advantage of the knowledge embedded in mathematical equations for the purpose of justifying heuristically generated actions.

C. Reasoning About Disparate Knowledge

Many artificial intelligence researchers adhere to the principle that a uniform knowledge representation is necessary. In fact, in many real-world domains, there are diverse sources of knowledge. In the application here, the computer develops its own meaning representation of complex equations. The equations are used at a deep level (deep meaning that their use is far removed from human observation) in the normal algorithmic sense. However, the equations can be interpreted at a symbolic level when justification is required of expert planning behavior. The equations are interpreted by qualitatively propagating constraints in a structure that explicitly represents the influence of each domain variable. The structure is generated by the computer using a symbolic series expansion, hence a form of sensitivity analysis. The structure is also interfaced to a naive physics representation. Recent work suggests that humans comprehend system performance based on their naive representations of the system.

McCloskey [13] relates that humans acquire remarkably well-articulated naive theories of motion based on everyday experiences. Hayes [14] discussed the need for naive theories and provided a theoretical framework for future work. de Kleer [15] explored the computational aspects of qualitative reasoning in the mini-world of the roller coaster and contributed the concept of envisionment. Envisionment predicts system behavior through qualitative simulation. In related research [16], he illustrates the use of Incremental Qualitative (IQ) analysis as a weak form of reasoning about perturbation. Forbus [17] uses a stronger approach that includes the sign and magnitude of a quantity's amount and derivative. The STEAMER project (Forbus and Stevens [18]) uses qualitative simulation of complex physical devices to construct explanations of the operation of those devices.

Qualitative reasoning is used to justify heuristically generated plans with knowledge that is computationally complex. The influences of domain variables are made explicit in a semantic structure suggestive of Rieger's common-sense algorithms [19], but purposely less expressive in the types of linkages. The structure is generated in part by a symbolic series expansion and unified by an explicit representation of Newtonian mechanics, abstracted to one dimensional motion. In this way, seemingly unrelated equations are accessible to the reasoning algorithm. The approach is motivated by conceptual dependency theory (Schank and Abelson [10]) where any implicit knowledge associated with a piece of text is made explicit in a meaning representation of the text.

III. The Air Traffic Control Domain

The United States consists of 23 Air Route Traffic Control Centers (ARTCC). An ARTCC is divided vertically into terminal, low-altitude enroute and high-altitude enroute control sectors. Each sector is manned by a team of controllers whose goals are to insure the safe and expeditious transit of each aircraft in the sector. For the purposes of this research, eight high altitude sectors of the Chicago ARTCC were modeled.

Controllers are subject to constraints from other sectors and supervisory controllers (e.g., a flow constraint), but are free to use one of five operators to achieve their goals. These operators are classified as the following aircraft maneuvers: turn, speed change, altitude change, and holding pattern. In

general, enroute aircraft have one of three intents: arrival (descend to a nearby airport), departure (depart a nearby airport), or overflight. Controllers learn rules, constraints, phraseology, and elementary problem solving strategies at the FAA Academy in Oklahoma City. However, the majority of their training takes place in the Dynamic Simulator (DYSIM) at their respective ARTCC's. In the DYSIM, realistic problems are solved. Representative DYSIM problems from the Dubuque and Joliet high altitude sectors were used in this research.

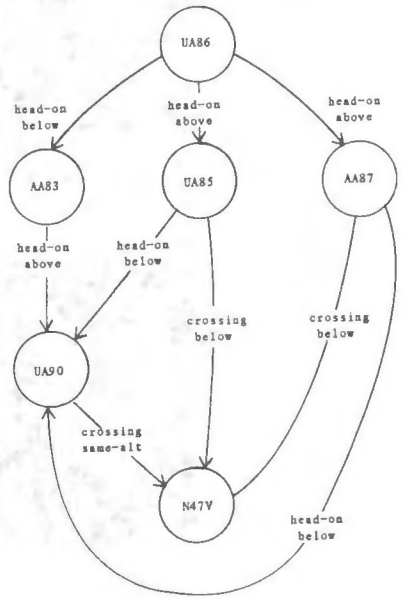
Consider two aircraft that are involved in a 'head-on' conflict. The controller must generate a plan that prevents a mid-air collision. The plan must involve the modification of one of the aircraft's flight plans. If collision avoidance were the only air traffic control goal, the solution would be trivial. Any legal operator (e.g., climb, descend) could be used. However, there are other important goals such as fuel efficiency. A significant portion of the controller's training involves assimilating heuristics useful for generating plans that achieve both goals. For instance, aircraft are usually more fuel efficient at higher altitudes. The human controller chooses an operator that prevents a collision and improves (or at least does not seriously degrade) fuel efficiency.

Many of the heuristics that controllers use are justified by mathematical knowledge. Assume that the controller requires one of the aircraft to climb, thus achieving the required separation. The climb was chosen because the controller recognizes that aircraft are more fuel efficient at higher altitudes. The declarative statement that 'aircraft are fuel efficient at higher altitudes' requires many equations for justification. Fuel efficiency is dependent on the fuel flow rate, which in turn is dependent on the required engine thrust. The required engine thrust is dependent on aircraft drag.Aircraft drag is dependent on aircraft geometry, airspeed, and air density. In the event that explicit statements of belief are nonexistent (failure of data-dependent backtracking), justification is based on a form of sensitivity analysis where qualitative constraints are assigned to domain variables and those values are propagated in a semantic structure which represents the domain equations and naive physics.

IV. An Expert System Approach

A successful application of artificial intelligence has been in the design of expert systems. An expert system is a computer program that solves problems normally deferred to a human expert. These systems use the knowledge of the human expert and an automated reasoning capability which may or may not resemble the reasoning processes of the human expert. An advantage is the ability of these programs to explain their solutions usually by reciting the knowledge that was utilized at each step of the reasoning process. The approach taken here combines the hierarchical structure concepts prevalent in MDLGEN [3] with script-based planning theories [20].

The expert system approach to problem solving in the domain of enroute air traffic control is briefly discussed. A problem is defined from aircraft flight plan data. The problems used in the research were taken from 'live' test problems used in the simulation facility of the Chicago ARTCC. The data is processed and a semantic structure which represents the global aircraft conflict scenario is created. The structure is shown in Fig. 1. It allows a global description of the air traffic control problem. Each node of the structure represents an aircraft and each link represents the type of conflict. The structure is decomposed into individual problems by applying problem decomposition strategies. Some strategies recognize goal interactions. For instance, an aircraft may be involved in recurring similar conflicts with many other aircraft. Each instantiated problem decomposition strategy in turn causes a problem solving strategy to be instantiated. A problem solving strategy specifies the knowledge that allows the computer to select which aircraft should receive a command and what type of commands should be used. The detailed knowledge about commands resides in tactics. Commands are criticized to insure they are do not exceed aircraft performance or cause additional



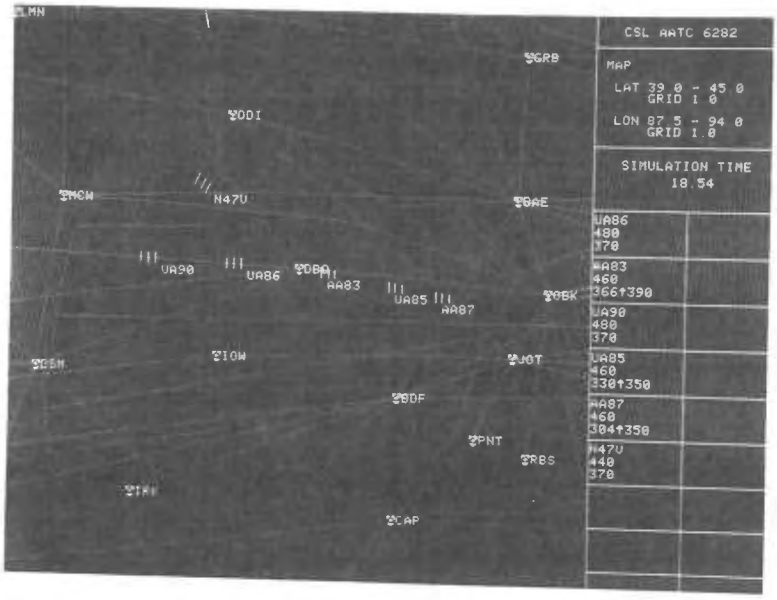
Conflict Structure
Figure 1

conflicts. The approach is modeled after human controller problem solving behavior.

Knowledge is represented in frames. Frames provide the structure in which the diverse knowledge of this task domain can be integrated. A frame language patterned after FRL (Roberts and Goldstein [21]) was implemented in Franz Lisp on a VAX 780. There were three distinct types of frames: aircraft, strategy, and tactic.

Fig. 2 presents an interesting problem. Without avoidance control, the six aircraft will be involved in nine near misses during a fifteen minute period. A typical problem solving approach would be to decompose the problem into nine separate problems, generate an avoidance command for each subproblem, and then try to combine the answers into a composite solution. This approach is similar to the behavior of a novice controller. An experienced controller would subdivide the problem into three subproblems that takes advantage of similar near miss situations. For instance, the conflicts between ua86 and and the set of aircraft aa83, ua85, and aa87 are sufficiently similar to be treated as one problem. A particular problem solving strategy is represented in a frame for this case. The set of problem solving strategies are prioritized in a strategy library.

The computer generated a solution for this problem that was similar both in solution and problem solving behavior to an experienced controller. The computer can explain its problem solving behavior using the approach advocated in production systems (e.g., why and how type responses by recitation of invoked rules). However, a human controller may wish to understand the solution at a deeper level. This is especially true as the domain becomes more automated. The controller will need insight into the equations used by the computer. However, either the equations or algorithms based on the equations may be too difficult for the human to understand. It therefore seems desirable for the computer to justify its plans using a model of naive physics which is probably comprehensible to the human.



A Representative Air Traffic Control Problem
Figure 2

V. Reasoning Component

The reasoning component consists of naive physics knowledge, domain equations, and an interface between these knowledge types and a data base of heuristics.

A. Naive Physics Level

Newton's laws are represented in a semantic network where nodes are primitive concepts (e.g., force, velocity) and links indicate their interaction. The structure represents the designer's understanding of Newtonian mechanics as abstracted to one dimensional motion of mechanical systems. The naive physics knowledge is made explicit by the designer. The linkages between the naive physics level and a domain are equation dependent. The representation serves as the basis for the interpretation of detailed domain equations and algorithms.

Nodes represent forces (propulsive, enabling, resistive), acceleration, velocity, position, mass, and power. Propulsive and enabling forces are referred to as positive forces. Propulsive forces require an external enablement which converts fuel (portion of system's mass) into the force. The rate of change of mass varies directly with the change in propulsive force. That is, when propulsive force increases, the fuel flow rate increases causing the mass rate of change to increase.

Another positive force is called an enabling force. Enabling forces do not directly influence mass. For instance, the air flow over a wing causes a pressure differential that enables lift. Lift is an enabling force that counteracts weight. Resistive forces counteract positive forces. Common examples are pressure and friction forces which vary with velocity and domain dependent variables such as weight, density, and surface area. Nodes are represented in a frame-like structure as shown below.

(frame F_{prop} a node with
(name = propulsive-force)
(influences = (F_+))
(level = naive-physics)
(instance = ((horz-level thrust))))

Nodes are related by links which specify how a dependent variable is 'influenced' by a changing independent variable. The links define a structure in which qualitative values are propagated. There are four types of links: influence, component, parent, and instance. Influence links are labeled positive or negative depending on a node's incremental affect on another node. Component links partition equations into terms. Parent and instance links indicate domain and naive physics relationships.

B. Search Strategy

Nodes are searched using a combination of forward and backward constraint propagation. Initially, the network describes a system in dynamic equilibrium. A node can be assigned one of three qualitative values (incr, decr, or no-change). The value is then propagated forward from the node until the network again reaches dynamic equilibrium. A node is in dynamic equilibrium when (1) its influences are constrained to a value of 'no-change' or (2) one of its influences was previously constraint to 'incr' or 'decr' and during another propagation an influence takes on an opposite value. Backward propagation is used to search for influence nodes that may cause some desired effect. Primary links are searched first. Secondary links are searched only when the search on primary links fail. In this manner, the computer has some control over an otherwise exhaustive search procedure. The assignment of primary and secondary linkages is context dependent. For instance, if one wishes to explain why velocity increases, one would typically first consider changes in propulsive force (unless the domain were that of hot air balloons and the velocity was vertical velocity, in which case mass would be a primary influence on acceleration). Each node is itself represented in a frame. Additional constraints can be placed on nodes. For instance, a node may in some context be constrained to remain unchanged. An attempt to influence a constrained node triggers a backward search for an influence that can be used to circumvent the affect of the original influence. Additionally, demons in a parent frame to all nodes represents common-sense facts such as 'a variable cannot increase beyond 100% of its maximum value.'

C. Domain Equations

Aircraft equations of motion are dependent on four forces: lift (L), thrust (T), drag (D), and weight (W). For level flight, dynamic equilibrium is defined as:

$$L - W = 0 \tag{1}$$

$$T - D = 0 \tag{2}$$

Each force is defined by an equation. Thrust is a function of throttle setting. Lift is a function of velocity, air density, angle of attack, and wing geometry. Weight is a function of aircraft mass and gravity. Drag is a function of air density, velocity, and aircraft configuration variables. Drag has two components: parasitic and induced drag. All subsonic aircraft performance capabilities can be derived from the drag equation (3). Consider an aircraft at a constant altitude and configuration. The maximum velocity then occurs when the drag equals the maximum thrust. Since drag is parabolic with velocity, the velocity at the minimum drag defines the 'best endurance' airspeed.

$$D = D_p + D_i \tag{3}$$

$$D_p = \frac{f V^2 \sigma}{295} \tag{4}$$

$$D_i = \frac{94}{\sigma e} \left(\frac{w}{b}\right)^2 V^{-2} \tag{5}$$

where,

v = velocity
w = weight
σ = altitude density ratio, and
b, e, and f are aircraft configuration variables

A symbolic series expansion is used to define the influence links. The sign of the first error term indicates a positive or negative influence. The magnitude of the influence is the amount of the first error term and is saved if ambiguity resolution is later required. There are four types of influence links: primary positive, primary negative, secondary positive, and secondary negative. The primary/secondary distinction is required for search efficiency. For instance, acceleration is primarily influenced by force and secondarily influenced by mass.

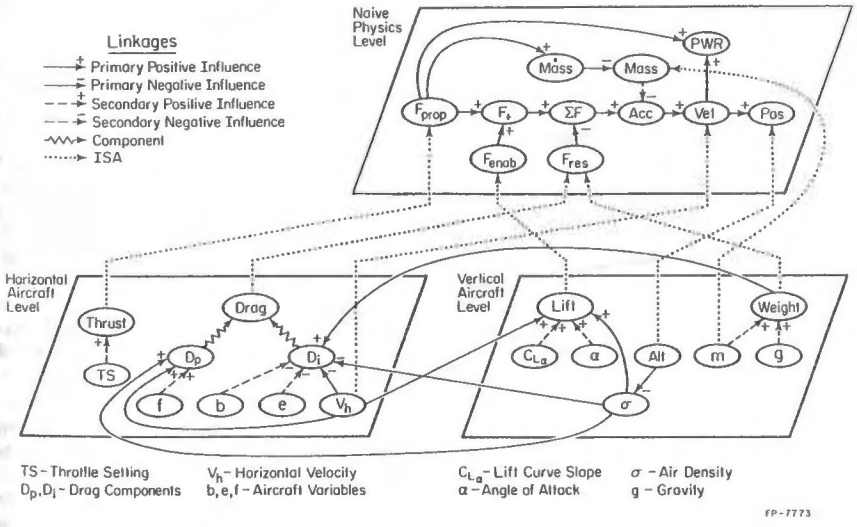
A variable is defined as a primary influence if it is an instance of an abstracted concept and that concept is also a primary influence. Aircraft airspeed (or horizontal velocity) is a primary influence because it is an instance of velocity which is itself a primary influence of position. Sometimes influences are found recursively. For instance, air density is a function of altitude (an instance of vertical position). Position can be a primary influence of resistive force (e.g., friction). Thus, it is inferred that air density is a primary influence of drag.

D. Interfaces

The signs of the terms in the force equations (1) and (2) are used to define instances of forces in the naive physics representation. For instance, T and L are positive forces. Semantic knowledge is also required. Thrust is a propulsive force while lift is an enabling force. In this context, weight and drag are resistive forces. The representation of both aircraft levels and the abstracted structure is shown in Fig. 3.

VI. Examples

In this section, several examples are presented to illustrate the reasoning capabilities. Consider a plan that requires an aircraft to increase its speed. Unless explicitly stated, the plan implies that a constant altitude be maintained. An important aspect of air traffic control is that the controller never invoke a plan that exceeds the performance capabilities of the aircraft. Given a plan to increase speed, how can the controller be sure that (1) the aircraft can increase its speed and (2) maintain a constant altitude? The computer can obtain the correct interpretation from the semantic network



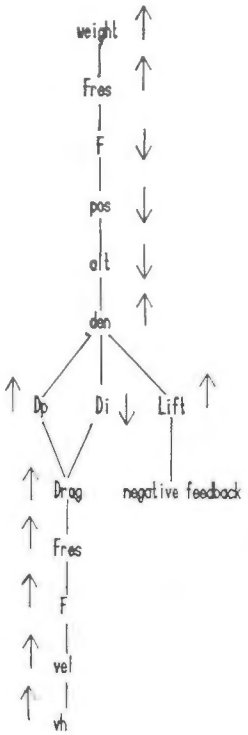
Interface to the Aircraft Levels
Figure 3

of performance equations. An increase of airspeed occurs when its parent, velocity (a naive physics level variable), is increased. An increase in velocity is caused by an increase in acceleration which is, in turn, caused by an increase in positive force or a decrease in drag. Drag cannot decrease because its influences are constrained (velocity is to be increased as air density is constrained since altitude is constrained). Thus, positive force must increase which implies an increase in throttle setting.

If the throttle setting is at 100%, then the aircraft cannot increase its speed. This common-sense fact is true for all variables and is stored at the generic variable level (a parent frame to all nodes). When throttle setting is increased, the value 'incr' can be propagated forward to find its influences. There is then a tendency for the increased airspeed to influence lift. Lift increases which tends to make the aircraft increase its altitude. But this violates an altitude constraint. A search is made for influences that counteract the positive influences. As there are no primary influences, a search is made for secondary influences. There are three: lift curve slope (C_{L_α}), wing area (S), and angle of attack (α). An intelligent choice from this set of variables requires the use of context related knowledge. For instance, the knowledge that the wing area is constant for a given aircraft must be represented in the frame for the node that represents wing area. Similarly, there must be represented the knowledge that the lift curve slope is constant for the context of enroute flight and that the lift curve slope changes in the landing context (e.g., flaps deployed). Angle of attack is pilot controllable and thus is the plausible answer. The computer performed an equivalent reasoning process by a combination of forward and backward constraint propagation.

Another example illustrates the reasoning process in the justification of a novel planning context. The expert system did not have an explicit encoding of what to do when an aircraft's weight increases during its flight through an air traffic control sector. Large military transport and bomber aircraft routinely increase their weights by several hundred thousand pounds during in-flight refueling operations. From the mathematical level, the aircraft would be commanded to climb to a new altitude. The justification of this action can be generated through qualitative simulation. When weight is increased, the aircraft will descend unless lift is increased. If the aircraft is to remain at its present altitude, then the only way to increase lift is to increase velocity. Velocity is increased by increasing thrust, but increasing thrust causes more fuel to be used which is fuel inefficient. Thus, if fuel efficiency is important, the aircraft must climb to a new altitude.. A trace of the reasoning process is shown in Fig. 4. The example shows the advantage of this reasoning approach. Previous approaches could only justify the novel plan step if a symbolic statement existed in a heuristic knowledge base or was in some way indexed to this plan's knowledge source. In this application, the computer generated the heuristic knowledge from reasoning about its mathematical knowledge.

A final example concerns a slightly different domain. The computer was given the domain equations which describe the propulsion capabilities and power requirement of a typical automobile. The reasoning process justified actions like 'why one would decrease the gear ratio when climbing a hill' (to maintain velocity while increasing power).



Reasoning About Increased Aircraft Weight
Figure 4

VII. Applications of the Research

Plan justification is a necessary component of computer understanding and improved explanation capabilities. The explanation for the climb command in an air traffic control context may be that it prevents a mid-air collision and is fuel efficient. The justification is that when drag decreases less thrust is required and since fuel use is proportional to thrust, it is therefore fuel efficient to perform maneuvers that minimize drag. In this context, the climb command minimizes drag since as altitude increases air density decreases. Knowledge implicit in many aircraft performance equations were used in the construction of the justification. The important point is that the computer represents the fuel efficient aspect in terms of the equations and their semantic structure. It can explain its justification at a more abstract level. The decrease in air density causes drag to decrease. Drag is a resistive force. Since resistive force decreases, the positive forces can also decrease.

The diagram shown in Fig. 3 is useful for common-sense reasoning about aircraft performance. A controller should never issue a command that cannot be implemented by an aircraft. Assume an air traffic controller commanded an aircraft to increase its speed. Implicit in the command is constraint on altitude (i.e., constant). Can an aircraft increase its speed without increasing its altitude? The answer is obtained by a combination of forward and backward constraint propagation in the naive physics and domain equation representation. By applying conceptual knowledge about aircraft (e.g., wing size is constant, lift curve slope changes only when the flaps are deployed) the computer reasons that (1) the throttle setting is increased resulting in the increased speed and (2) the angle of attack is decreased resulting in a constant altitude.

The approach is useful in the learning of new problem solving strategies. Human controllers acquire their skill by the justification and assimilation of an expert controller's plans. This type of learning is referred to as advice-initiated. A prerequisite of advice-initiated learning is that the computer understand the advice. I say that this process is advice interpretation. I define advice interpretation as the justification of another expert's plan. Reasoning about mathematical knowledge, through a process of qualitative simulation, allows the computer to construct justifications of novel plans.

VIII. Conclusion

Domain equations often contain knowledge which are useful for the justifcation of heuristically generated plans. The mathematical knowledge is contained at a 'deep' level and made transparent to a reasoning algorithm by a symbolic series expansion. A significant contribution of the research is that the computer constructs the representation that allows reasoning. The resultant representation makes explicit the influences of domain variables and abstracts those variables to a naive physics level.

Acknowledgments

The research was supported by the Department of Transportation, Federal Aviation Administration under contract number DOT-FA79WA-4360 while the author was a graduate student at the University of Illinois at Urbana-Champaign. The research continues under the sponsorship of the Air Force Wright Aeronautical Laboratories.

References

[1] W. Rouse, *Systems Engineering Models of Human-Machine Interaction*, New York, NY: Elsevier North Holland, Inc., 1980.

[2] S. Cross, *Qualitative Reasoning in an Expert System Framework*, PhD Dissertation, Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, CSL-T-124, 1983.

[3] M. Stefik, *Planning with Constraints*, PhD Dissertation, Stanford University, STAN-CS-80-874, 1980.

[4] B. Chandrasekaran and S. Mittal, 'Deep Versus Compiled Knowledge Approaches to Diagnostic Problem-Solving," *Proceedings of the National Conference on Artificial Intelligence*, pp. 349-354, 1982.

[5] W. Clancey, *The Epistemology of a Rule-Based Expert System: A Framework for Explanation*, STAN-CS-81-896, 1981.

[6] J. Aikins, *Prototypes and Production Rules: Knowledge and Representation for Consultation*, PhD Dissertation, Stanford University, STAN-CS-80-814, 1980.

[7] H. Pople, 'The Formation of Composite Hypothesis in Diagnostic Problem Solving: An Exercise in Syntactic Reasoning," *Sixth International Joint Conference on Artificial Intelligence*, pp. 1030-1037, 1979.

[8] J. Doyle, "A Truth Maintenance System," Massachusetts Institute of Technology, MIT-AI-521, June 1979.

[9] P. Cohen and M. Grinberg, "A Theory of Heuristic Reasoning About Uncertainty," *AI Magazine*, 4: 17-24, Summer 1983.

[10] E. Charniak, "A Common Representation for Problem-Solving and Language Comprehension Information," *Artificial Intelligence*, 16: 225-255, 1981.

[11] E. Charniak, 'On the Use of Framed Knowledge in Language Comprehension," *Artificial Intelligence*, 11: 225-265, 1978.

[12] R. Schank and R. Abelson, *Scripts Plans Goals and Understanding*, Hillsdale, NJ: Lawrence Erlbaum Associates, 1977.

[13] M. M^{c}Closkey, 'Naive Theories of Motion," in D. Gentner and A. Stevens Eds.), *Mental Models*, Hillsdale, NJ: Lawrence Erlbaum Associates, 1982.

[14] P. Hayes, 'The Naive Physics Manifesto," in *Expert Systems in the Micro-Electronic Age*, D. Michie (Ed.), Edinburgh, UK: Edinburgh University Press, pp. 242-270, 1979.

[15] J. de Kleer, 'Qualitative and Quantitative Reasoning in Classical Mechanics," *Fifth International Joint Conference on Artificial Intelligence*, pp. 299-304, 1977.

[16] J. de Kleer, *Causal and Teleological Reasoning in Circuit Recognition*, PhD Dissertation, Massachusetts Institute of Technology, MIT-AI-TR-529, 1979.

[17] K. Forbus, *Qualitative Process Theory*, Massachusetts Institute of Technology, MIT-AI-TR-664, 1982.

[18] K. Forbus and A. Stevens, 'Using Qualitative Simulation to Generate Explanations," Cambridge, Mass: BBN, Inc., 1981.

[19] C. Rieger, *The Commonsense Algorithm as a Basis for Computer Models of Human Memory, Inference, Belief, and Contextual Language Comprehension*, University of Maryland, TM-373, 1975.

[20] R. Wilensky, *Planning and Understanding*, Reading, Mass: Addison-Wesley Publishing Co., 1983.

[21] R. Roberts and I. Goldstein, *The FRL Primer*, Massachusetts Institute of Technology, MIT-AI-TR-408, 1977.

MISSION SCENARIOS FOR COCKPIT AUTOMATION TECHNOLOGY

Gilbert G. Kuperman
and
Philip V. Kulwicki

AIR FORCE AEROSPACE MEDICAL RESEARCH LABORATORY
Wright-Patterson Air Force Base, Ohio

Abstract

This paper describes the development and application of a detailed tactical fighter attack mission scenario package for exploitation in an Air Force advanced development program directed to developing and validating an advanced crew system design methodology. The scenario development process excluded explicit consideration of a specific avionics suite and, for that reason, is described as being "technology-free." Emphasis was placed on identifying and describing areas of mission uncertainty and aircrew decision nodes encountered during conduct of the mission.

Introduction

The Air Force Systems Command has initiated an advanced development program directed to the establishment of an improved crew system design process which emphasizes both the operational aircrew and the implementation of automation within USAF fighter aircraft. This program, Cockpit Automation Technology (CAT), is managed by the Aerospace Medical Division (AMD), with technical support from the Air Force Aerospace Medical Research Laboratory and the Avionics and Flight Dynamics Laboratories of the Air Force Wright Aeronautical Laboratories, and is being conducted with the participation of other Department of Defense organizations and of industry. An Air Force Summer Study report (2) highlighted the need to solve the problem areas that the CAT program is addressing. The following quotations were extracted from the board's findings and conclusions:

> "The complexity of today's missions and high-performance aircraft has created workloads that at times impose intolerable demands on combat pilots.
>
> "The aircrew's stated immediate need is for improved ability to fly low, at night, and during severe weather, using terrain for cover from enemy defenses. The critical and essential functions that could be automated to achieve this goal have not been completely identified...
>
> "The Air Force does not have an established position on the requirements for automation in aircraft.
>
> "There is a large gap between what is known in a laboratory setting of the basic characteristics of human psychomotor performance, and what is known about how pilots actually fly and react in modern combat aircraft. Much of the knowledge needed to design an automated aircraft that uses pilots' skills to the best advantage lies within that gap."

These quotations are intended to convey an appreciation for the broad and complex nature of the problem being addressed by the CAT program. The overall goal of the program is to develop and validate an advanced crew system design methodology. An important product of the program is the determination of automation requirements for future combat systems based on specific mission demands and taking full account of pilot capabilities.

Morgan et al. (7) describe the CAT program as being composed of four stages:

1. Mission Characterization
2. Function Allocation
3. Integration and Design
4. Validation

The mission characterization stage serves to define mission requirements and to decompose them into critical functions, together with required performance criteria. (Mission characterization includes the creation and analysis of the mission scenario.) The function allocation stage serves to rationally assign the accomplishment of system functions to the man, the machine, or to a combination of the two. This stage includes and applies the findings of an automation technology forecasting and assessment substage. The integration and design stage includes the conceptualization of alternative cockpit designs and their refinement (through analytic and man-in-the-loop simulation techniques), to produce a final crew system design that best supports the mission performance criteria set. The final stage, validation, assures the reliability of the predicted crew system (and, hence, weapon system) performance. This is accomplished through mission simulation and measurement of actual, achieved performance and workload data. (O'Donnell [8] provides a detailed review of psychophysiological workload measurement techniques while Eggemeier [3] discusses subjective workload measures.)

The CAT design process is based on specific mission demands; accordingly, a detailed mission scenario document is required to provide the context within which many of the technical activities must be carried out. The development and application of the CAT mission scenario are the subjects of this paper.

Background

Current practice in the design of combat aircraft has, until recently, resulted in crew systems which were at least managable by the crewmember(s). Within the last decade, however, crew workload has grown to the point where less than immediately critical tasks are often deferred (if not omitted) during some mission segments. This growth in workload has been attributed to the same three factors that drive the development of new or enhanced weapon systems:

1. Response to improved enemy threat systems.
2. Response to changed mission requirements.
3. Response to opportunities for technology insertion.

In very general terms, each of these "drivers" (alone or in combination) results in the addition of technology to the weapon system. Although often advocated, at least in part, as an amelioration to the high cockpit work situation, the result has been, in fact, a growth in the complexity of the man-machine interface. The problem is compounded to the extent that these technologies are typically introduced in isolation from any other system capabilities and, therefore, frequently impose the additional burden of requiring that information be assimilated across multiple information sources (and, possibly, multiple modalities) in order to perform a single task. Examples of individual technology-driven subsystems include:

- Multimode Radar Systems
- Multisensor Target Acquisition Systems
- Subsystem Caution/Warning Systems
- Threat Warning Systems

Adoption of individual technologies, with only limited attention to total system impact is, potentially, an example of an error of commission. Errors of omission are also possible. Eggleston and Kulwicki (4) describe a value analysis framework for estimating the worth of emerging fighter/attack system technology. They point out that previous methods are largely unable to relate "technical capability/performance to mission effectiveness."

The problems inherent in technology-driven systems, particularly crew systems, are compounded by the fact that the technology is explicitly specified in the weapon system procurement process rather than having the desired system performance capability described and having this description of operational capability guide the system design/specification process. From a human factors viewpoint, the "explicit specification" approach relegates the crewmember to the role of a "slack variable" that can be apportioned across subsystems in order to achieve total system functioning. The "capability description" approach (e.g., in terms of mission effectiveness), on the other hand, elevates the crewmember at least to the status of any other subsystem in performing trade-offs between system performance and subsystem capabilities and limitations.

The crewmember has not been totally excluded from the crew system design arena. Rather, he has functioned as an expert participant in an iterative process directed toward integrating subsystem technologies, including automation, at the man-machine interface, the crew system. Kuperman et al. (5) and Kuperman (6) describe the use of noninteractive cockpit mock-ups in refining crew system concepts, while Spencer (9) reports on his participation in a large scale, part mission, simulator exercise directed toward the same goal. In each of these cited cases, however, the crewmember's participation occurred after the avionics suite, controls and displays, etc., had been specified and his input was limited to responses concerning the adequacy of the overall crew system concept and the location or arrangement of specific control functions within the cockpit.

An associated problem is that, to the extent that the specified technologies represent automation of previously manual functions, the weapon system designer has lacked guidance on how to introduce automation into the crew interface. It requires only little imagination to posit an automatic target cuer that forces the crewmember to perform multiple target recognitions but allows him only a few seconds before the opportunity to launch weapons has passed. Similarly, little is known about the additional training burden required to assure that the crewmember can recover manual control of the (sub)system should the primary, automated mode of operation fail.

The CAT Mission Scenario

Formal direction for the CAT program required consideration of a conventional tactical attack mission. The scenario was to be created in the context of post-1990 threats, platforms, targets, and weapons. The air-to-ground (A/G) aspects of the scenario were to be emphasized, although the weapon system was to reflect a significant (air-to-air [A/A]) self-protection capability.

The A/G mission is, in general, viewed as being taxiing to the man-system interface because a great diversity of subsystems must be exercised in its accomplishment. A battlefield interdiction mission, in particular, places workload on the crew because of the need to ingress to the target, the variety of targets that may be attacked, the variety of threats that must be defeated, etc. (At the time of this writing, an A/A mission scenario package is being prepared. The A/A mission takes place in the same area, at the same time, and reflects the same threat "bed-down" as does the A/G mission. The A/A mission follows the same development process and is documented in a similar style. It includes both offensive and defensive counter-air engagements.)

The scenario includes elements of sector and engagement levels of operation. This affords opportunities to exercise the weapon system as part of the coordinated air battle while maintaining the finer detail (specific tactics, profiles, weapon/target pairings, etc.) inherent in the A/G attack. This level of consideration also affords the human factors researcher an opportunity to explore the complexities of crew tasking at the same level that the crew experiences the tasking. To assure completeness of the analysis of the crew interface, the scenario includes all

phases of the A/G mission, from premission planning through postflight/debriefing.

The European theater of operations is the locale of interest. This provides opportunities to explore the impact of diverse threat systems, diverse terrain types, and highly diverse weather patterns. For similar reasons, both day and night operations are considered. The European environment provides ample variability to support trade-off analyses while includng "worst case" conditions of weather, threat, etc. Again, the goal is to realistically motivate workload variation and to provide a framework for the exploration of crew decision making under uncertainty.

Certain system capability assumptions were required in the creation of the scenario. Aircraft capabilities (turn rates, instantaneous and sustained G loadings, etc.) were defined commensurate with a next-generation fighter aircraft. Weapon parameters were defined for follow-on versions of existing A/G weapons. Threat system capabilities were based on published, classified threat descriptions. Tactics were evolved with the participation of representatives of the Tactical Air Command's (TAC) Fighter Weapons School. None of these assumptions forced the declaration of specific avionics subsystems or cockpit control/display subsystems.

The Scenario Development Process

The mission scenario document is central to the mission characterization stage of the CAT program. It is intended to provide both a detailed chronology and description of mission events and serve as a first-order data base during the development of a computerized mission model (using a suitable analytic technique, such as network analysis) during the mission decomposition substage. In order to satisfy these goals, a structured process was followed in creating the mission scenario document.

Background Data. The most current sources of threat capability, distribution, and engagement tactics were reviewed and relevant information was extracted for appropriate surface-to-air and A/A threat systems. The scenario includes sufficient information on these systems to support engagement-level simulations that may be undertaken later in the CAT project to assess the worth of prospective uses for cockpit automation.

Weapons Data. Technical discussions were held with representatives of the Air Force Armament Laboratory concerning the capabilities of weapons that could reasonably be expected to be in the inventory during the time frame of interest. Precision-guided, autonomously launched, and gravity weapons were included. Any expected constraints on the delivering aircraft were identified. Target/weapon pairings for both fixed/hard and mobile/soft targets were also included in the scope of the technical exchange. Sufficient technical data were obtained (and included in the scenario document) to permit simulation of weapon delivery events.

Tactics. Tactical doctrine was explored with highly experienced TAC personnel. Areas of discussion included command/control/communication/intelligence practices, coordination with friendly ground forces, use of threat-defeating systems (flares, chaff, and countermeasures), and engagement doctrine. Aircraft maneuvers during threat evasion, penetration, ingress, attack, and egress were discussed in detail. The TAC Systems Office, located at the Aeronautical Systems Division, Wright-Patterson Air Force Base, provided continuing support and guidance during the entire scenario creation process.

Mission Events. Although the specific details of the CAT mission scenario are classified, the mission can be described in general terms. The mission begins with the receipt of the fragmentary order. Great detail is provided on the process of planning and briefing the combat mission. Departure is from a main operating base in West Germany. Cruise-out is in keeping with aircraft, mission, and friendly ground force requirements. Penetration is accomplished at an appropriate speed and altitude. Ingress to the target area, a railroad tunnel, is low and fast. The attack profile is commensurate with aircraft, weapon, and survivability requirements. Egress is similar to ingress and recovery is made at a forward operating base. Figure 1 presents an artist's rendering of the mission as it is planned and briefed in the scenario document.

During the course of the mission, numerous functions are accomplished (navigation updates are made, enemy threat systems are encountered and engaged or evaded, mission information is exchanged, etc.). Many of these tasks are decision nodes in which tactics are implemented or mission variations adopted under conditions of less than perfect information (mission uncertainty). Throughout the CAT mission scenario document, these decision nodes are identified, a set of possible alternatives is described, and the rationale for the specific alternative selected is explained. This was done to support the exploration of variations from the described mission and to investigate the possible impact of pilot-aiding automation.

CAT Mission Characterization

The CAT advanced development program is currently underway. Multiple contractor teams are performing cockpit automation and component technology forecasts and assessments and identifying/developing/refining man-machine simulation, crew system design, and workload prediction/measurement tools. The CAT mission scenario was provided to each team as government-furnished information (GFI). Each contractor team is also revisiting the CAT mission scenario in order to gain additional insight into the operational environment and required weapon system performance descriptions. A structured approach to dealing with the CAT mission has been defined by the Air Force. The mission characterization stage of the CAT program has been broken down into five substages. Figure 2 presents a conceptual flow diagram of the CAT mission characterization stage. The major substages are identified and described below.

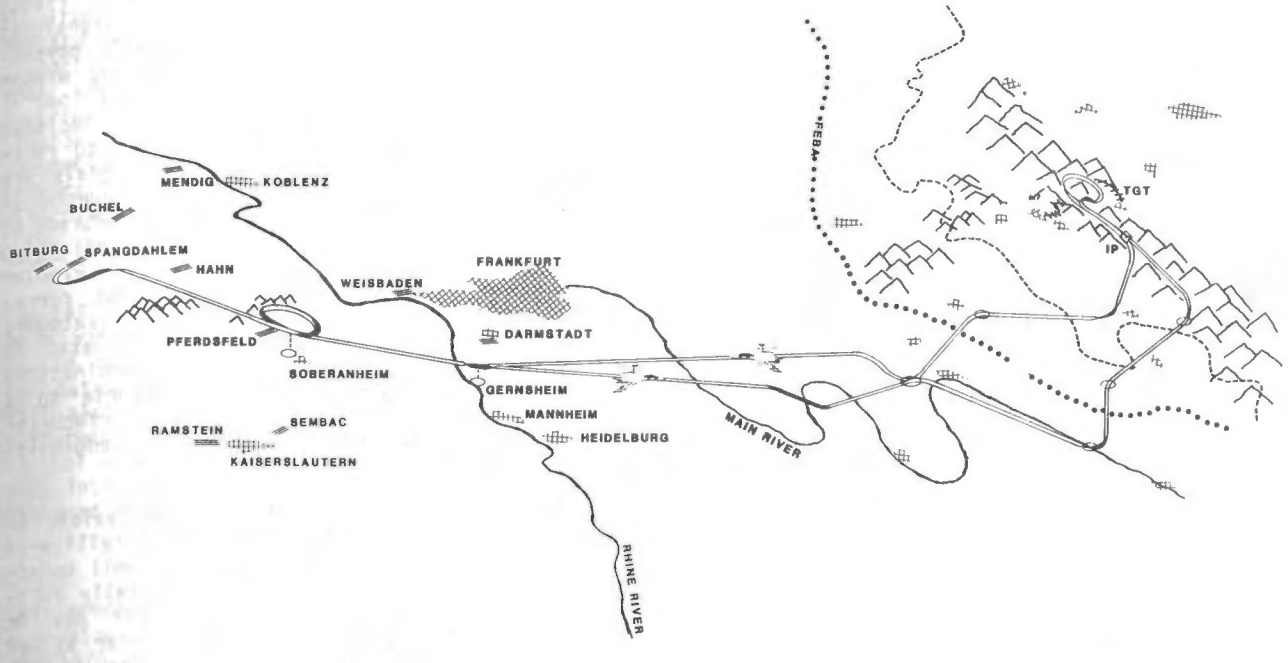


Figure 1. Artist's Concept of CAT A/G Mission

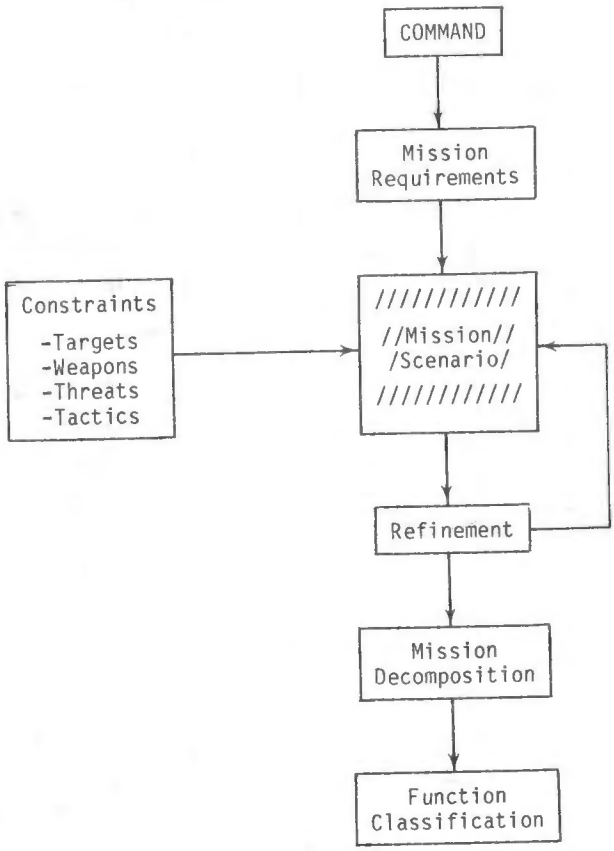


Figure 2. Conceptual Flow Diagram of CAT Mission Characterization Process

Review. Each team is performing a detailed operations analysis of the GFI scenario to assure its completeness and their understanding of its implications.

Baseline System. Each team is identifying a recommended baseline fighter/attack weapon system (airframe and avionics) to serve as the reference against which incremental benefits of cockpit automation are to be demonstrated.

Refinement. The time sequence of mission events recorded in the scenario is being analyzed in order to develop detailed mission timelines. The baseline weapon system is assumed and sufficient detail is developed to account for variations introduced by mission uncertainty (malfunctions, etc.), threats, and the environment. A second aspect of the refinement step is the determination of specific mission performance criteria by which to assess the accomplishment of mission objectives. Figure 3 presents a conceptual flow diagram of this substage.

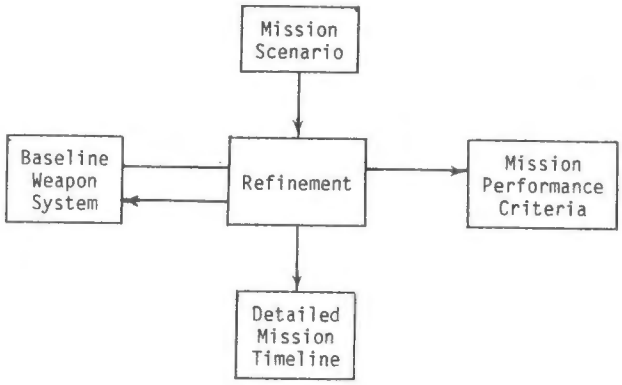


Figure 3. Conceptual Flow Diagram of Mission Scenario Refinement Substage

Decomposition. This step is a well documented effort which provides the bridge between the required system performance described in the CAT mission scenario and the conceptual evolution of the crew interface. An iterative, hierarchically organized approach allows the crew system analysts/conceptual designers to make successively more detailed assumptions regarding the man-machine interface, to examine the implications of the assumptions with regard to workload and total system performance, and to explore alternative crew system concepts until the point is reached at which all man-machine interactions are stated and are individually traceable to mission objectives, baseline system capabilities, technology assessments, and assumed mechanization approaches. Thus, an explicit path is traced between system performance requirements and the specification of the crew system interface.

Function Classification. The decomposed CAT mission is analyzed to identify all crew functions and to categorize these functions into:

- Operations Variables
- Decision Variables
- Problem Formulation Variables

Each class of variable corresponds to a distinct behaviorally-related category. Operations variables correspond to skill-based behaviors (i.e., functions in which a sensory input signal is received and the appropriate, frequently psychomotor, response is evoked). Recovering wings-level flight in response to a sudden gust load is an example of skill-based behavior. Decision variables are functions which correspond to rule-based behaviors (i.e., sensory input signals are recognized as requiring a patterned sequence of responses). Following checklist procedures in response to a subsystem warning/caution indication is an example of rule-based behavior. Problem formulation variables are functions that correspond to the knowledge-based behavior category (i.e., the response pattern must be created in response to the specific values and combinations of the sensed input signals). Response to a "pop-up" threat system, such as mobile surface-to-air missile, would be an example of knowledgeable-based behavior.

The three behavior categories are equivalent to levels of capability in artificial intelligence systems. If the required response is well understood, skill-based behaviors may be totally automated. An automatic pull-up maneuver may be integrated into a terrain following radar system to recognize and correct for violation of the preset minimum altitude. Rule-based behaviors may also be automated, at least to a large degree. The system might automatically carry out the flight manual procedures required in response to an engine fire warning, for example. Knowledge-based behaviors are less easily automated. Rather, a pilot-aiding capability might allow the crewmember to more effectively respond to complex situations. Situation awareness displays would be an example of pilot-aiding.

The identification of decision nodes and areas of mission uncertainty in the mission scenario document assist in the identification and classification of crewmember functions. The discussion of operational tactics, the detailed description of the mission planning process, and identification of possible response alternatives during decision making, all are intended to support the function classification substage of the CAT process.

Technology Assessment

Although the CAT program includes a separate, formal technology forecasting and assessment effort (included in the function allocation stage which follows the mission characterization stage), the mission scenario itself serves a closely related purpose. The scenario does not include explicit definition of the avionics suite. Rather, it identifies what the weapon system, including the crew and other subsystems, must accomplish in carrying out the mission. Thus, the crew might be described as "adopting terrain clearance flight" and then "adding terrain avoidance flight" without stating whether auto terrain clearance/avoidance modes are embodied in the multimode radar system. Similarly, the crew might be described as "observing a surface-to-air missile system's radar transition from search to track" without stating how that information is acquired and displayed. The scenario presents the context within which mission-related performance

criteria are developed during the mission scenario refinement substage. Thus, time and accuracy requirements for many mission tasks are implicit in the mission planning process and in the scenario description. Even though the primary goal of the mission scenario document is to describe system events, the level of detail provided makes it an excellent vehicle for supporting technology trade-offs, crew system integration concept formulations, and system-level effectiveness predictions.

Scenario Applications

The CAT program will draw on the mission scenario in a variety of applications. The decomposition of the mission into function variables and behavior categories has been described. The scenario will also be used in developing/refining operator performance/operator workload predictive mathematical models and as the "script" for man-in-the-loop interactive simulations through which the CAT designed crew interface will be validated.

Summary

This process of developing, refining, and applying a mission scenario has been described. The post-1990 A/G attack scenario, developed for the CAT program, offers particular attributes that make it extremely well-suited for its intended purposes. It is "technology free" in that neither the avionics suite nor the controls/ displays concept are explicitly stated. The document explicitly states what mission tasks must be accomplished in carrying out the mission. (Where imposed by external factors, the time and accuracy criteria for successful task accomplishment are also provided.) It is left to subsequent analysis to identify and substantiate the inclusion of specific technologies, particularly those associated with cockpit automation, in the weapon system. The mission scenario document emphasizes the crewmember's viewpoint. Tasks to be accomplished, events to be responded to, etc., are all presented from the perspective of the pilot. This orientation is important since crew workload may be the major limiting factor to achieving weapon system performance capabilities. The mission is not deterministic. Unplanned events are provided for and the crew's responses to them are defined in terms of decision nodes. This permits the analyst to create variations from the basic mission. Mission variations can be readily constructed by adding or deleting unplanned events and/or by changing the decision alternative adopted. The mission is based on realistic weapons, threats, targets, tactics, weather, and terrain. This realism provides the face validity of the document and supports the operational relevancy required by the CAT project. The mission scenario document serves as a primary resource in undertaking the development of cockpit automation technology.

References

1. Aretz, Anthony J., "Cockpit Automation Technology," Proceedings of the 1984 Human Factors Society Annual Meeting, October 1984.

2. Davidson, J. (Chairman), Report of the Committee on Automation in Combat Aircraft, National Academy Press, Washington, D.C., 1982.

3. Eggemeier, F. Thomas, "Current Issues in Subjective Assessment of Workload," Proceedings of the 1981 Human Factors Society Annual Meeting, October 1981.

4. Eggleston, Robert G. and Kulwicki, Philip V., "Estimating the Value of Emerging Fighter/ Attack System Technologies," Proceedings of the NATO Defense Research Group (DRG) Panel VIII Workshop, Shrivenham, England, April 1984.

5. Kuperman, Gilbert G., Moss, Richard W., and Bondurant, Robert A., "Crew System Assessment Methods Applied to Derivative Fighter Cockpits," Proceedings of the 1983 Human Factors Society Annual Meeting, October 1983.

6. Kuperman, Gilbert G., "Criteria for Selecting Subjects for the Assessment of Advanced Crew System Concepts," Proceedings of the Ninth Psychology in the Department of Defense Symposium, April 1984.

7. Morgan, D. Reed, Furness, Thomas, Aretz, A., Cole, Dean, and Kulwicki, Philip V., "Cockpit Automation Concepts," Proceedings of the NATO Defense Research Group (DRG) Panel VIII Workshop, Shrivenham, England, April 1984.

8. O'Donnell, R. D. Contributions of Psychophysiological Techniques to Aircraft Design and Other Operational Problems, AGARD-AG-244, NATO Advisory Group for Aerospace Research and Development, Neuilly sur Seine, France, July 1979.

9. Spencer, David C., "F-15 Dual Role Fighter Cockpit Integration," Proceedings of the Society of Automotive Engineers 1983 Aerospace Congress and Exposition, October 1983.

SESSION 5 DIGITAL FLIGHT CONTROLS

5

Chairmen:

Charles R. Abrams
Naval Air Development Center

George Vetsch
McDonnell Douglas Corp.

This session is concerned with recent and emerging technologies relating to digital flight control systems with emphasis on hardware implementation, processor architectures and data management.

84-2622

BACKUP MODES FOR THE F/A-18 DIGITAL FLIGHT CONTROL SYSTEM

H. E. Harschburger*
B. Glaser**
J. R. Hammel†
McDonnell Aircraft Company
McDonnell Douglas Corporation
St. Louis, Missouri

Abstract

The McDonnell Douglas F/A-18 Hornet is the first production high performance fighter aircraft with a digital fly-by-wire (FBW) primary flight control system. The requirements for backup modes for flight control systems have historically been based mainly on reliability and survivability experience with previous aircraft. The decision to use a quadruplex digital fly-by-wire mechanization in F/A-18 flight control system was a primary consideration in specification of the backup modes.

In the F/A-18 development, major emphasis was placed on designing the aircraft to minimize the effects of electromagnetic interference, and extensive software testing procedures were established to minimize the possibility of catastrophic software errors. These efforts in conjunction with extensive degraded mode capability of the quadruplex primary control system makes the probability of reversion to a backup mode extremely remote. The F/A-18 experience indicates that if the environmental factors are adequately defined, the systems are designed to meet those requirements, and adequate software verification and validation is performed, future aircraft should have minimal requirements for backup control system modes.

System Description

The F/A-18 flight control system, Figure 1, is a control augmentation system (CAS) configuration which is implemented using fly-by-wire techniques. All control law computations are performed by the four digital computers working in parallel (quadruplex redundancy). The computers use inputs from analog cockpit controls and motion sensors to compute commands to the redundant electrohydraulic servoactuators that provide the desired surface deflection and aircraft response. Aircraft control is provided by two ailerons, two rudders, two stabilators, two leading edge flaps, and two trailing edge flaps. The overall redundancy provides two-fail-operate capability for the primary control functions. Backup modes include mechanical control of the two stabilators and fly-by-wire analog control of the ailerons and rudders. The flight control computer, motion sensors and cockpit controls are supplied by General Electric (GE). The software coding for the computer is performed by GE using system requirements defined by MCAIR.

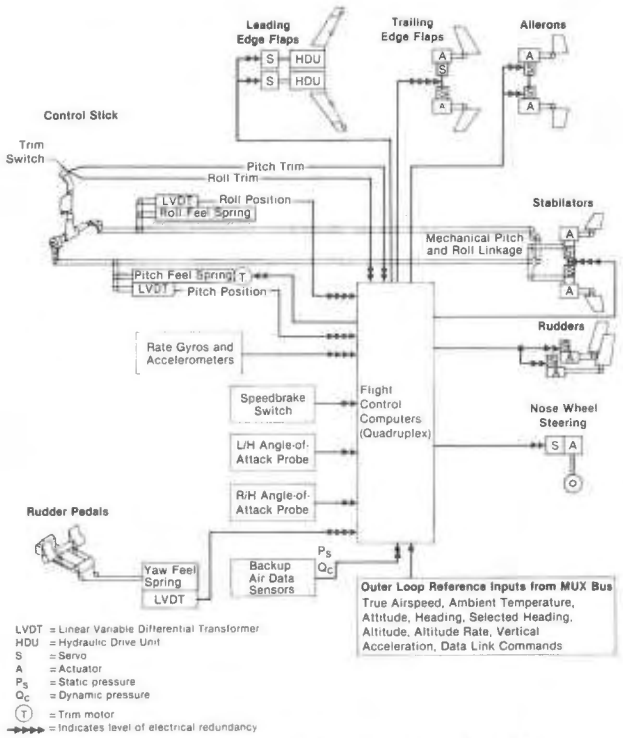


Fig. 1 F/A-18A Flight Control System Functional Diagram

Back-up Flight Control System Requirements

The configuration of the air vehicle plus the mission of the weapon system defines the complexity of the flight control system. The requirements imposed upon the system for operation following a failure and the allowable transient response define the complexity of the control system mechanization. In specifying the degraded mode operation and backup systems the military services generally draw on three areas of experience: 1) random failure history of similar parts (reliability), 2) combat damage due to the assumed threat (survivability), and 3) environmental factors which may not be sufficiently defined. The weighting factors placed on the three areas are usually a function of the seriousness of the problems encountered during the last comparable program.

During the initial F/A-18 design phase MCAIR made two decisions which had a significant impact on backup control system requirements: the basic aircraft would have positive aerodynamic static stability and the flight control system would be a quadruplex digital fly-by-wire mechanization. Backup control systems for aircraft with positive aerodynamic static stability can be very simple, with no augmentation feedbacks. However, the decision to incorporate a digital fly-by-wire

*Section Chief, Guidance and Control Mechanics, Member AIAA
**Senior Engineer, Guidance and Control Mechanics
†Engineer, Electronics

primary control system raised additional concerns which reinforced the requirements for backup control modes: the potential for software errors which could result in simultaneous shutdown of the digital processors ("generic software failure") and the concern that electromagnetic interference could shut down the digital computers or alter their operation. Other concerns raised in conjunction with backup mode requirements were the effects of lightning strikes and loss of all electrical power.

All of the concerns outlined above were considered in the specification of the degraded and backup modes for the F/A-18. The relevant requirements are shown in Figure 2. The EMI requirements are presented below.

SD-565-1, 3.3.1
"3.1.10.2 With mechanical pitch and roll controls only, and with no less than two like failures in the rudder control system, the aircraft shall be capable for returning and performing a field landing. Categories B and C, Level 3 longitudinal short-period and Dutch roll frequencies, time-to-bank, and cross wind requirements shall be met."

SD-565-1, Appendix G
"4.3.37 The flight control system shall incorporate design features to minimize loss of flight path control due to single hits from a 23 mm HEI-T or specified fragment. Routing and separation of electrical signal wiring and mechanical flight control systems shall be such that maximum protection against 23 mm HEI-T or specified fragment is afforded by masking and/or shielding."

MIL-H-5440F, 3.2
". . . The hydraulic system(s) shall be configured such that any two fluid system failures due to combat or other damage which cause loss of fluid or pressure will not result in complete loss of flight control . . ."

AS1291 (AV), 3.7.68
"b. Ability to withstand one electronic failure and continue to provide Level 1 performance as defined in MIL-F-8785. With two like electronic failures, the flight control system shall provide Level 3 performance."

Fig. 2 F/A-18A FCS Requirements

The uncertainty about the effects of EMI, the concern with software design faults, and the electrical power concerns dictated dissimilar control paths for the backup control systems. The first mode specified was a mechanical backup to provide pitch and roll control using the stabilators. If there were remote possibilities of a total electrical system failure, a generic software failure, catastrophic battle damage, or shutdown of the digital processors, it was desired to provide the pilot with sufficient control capability to at least return to the vicinity of the carrier before he ejected. The next backup mode defined was an analog "direct electrical link" (DEL) for use in the event that digital computations were denied. Since the stabilator was backed up by the mechanical control, it was decided to use the analog DEL mode only for the rudders and ailerons.

F/A-18 Flight Control System Mechanization

Reliability and survivability considerations in consonance with the F/A-18 aerodynamic configuration were used to establish the redundancy levels for each control function. Control functions which are critical from flying qualities or safety standpoints were designed with two-fail-operate/fail-safe capability. These functions include primary control commands, motion sensors, and stabilator and trailing edge flap actuators. Less critical functions or control surfaces with aerodynamic redundancy were designed fail-operate/fail-safe capability. The redundancy selected for each of the major flight control system components is shown in Figure 3. The reliability and fail operational requirements could have been met with a triplex configuration if in-line monitoring had been used. However, in 1975, when the F/A-18 flight control system design was started, in-line monitoring of digital flight control systems was not considered state-of-the-art technology. Also, the survivability requirements for separation of components made it advantageous to configure the quadruplex system in two identical packages, each with dual components. The flight control system channel arrangement is shown in Figure 4. This channel arrangement was established with survivability and vulnerability as major considerations, which include maximum separation of redundant channels and components, separating quadruplex sensors in two dual packages, and use of other components in less critical systems as a shield to reduce the vulnerability of flight control system components.

A major factor in the F/A-18 design philosophy is to provide maximum system capability with the loss of any control function. The failure mode capability for each of the major flight control system components is shown in Figure 5. The degraded mode categories are shown in Figure 6. The degraded modes for the F/A-18 can be relatively simple because the basic vehicle is statically stable. Open loop control of any of the three axes provides adequate handling qualities for get-home-and-field-landing capability.

The mechanical control system schematics are shown in Figures 7 and 8. The stabilator actuators contain a command select mechanism (CSM) which disengages the mechanical inputs during normal operation. In the event of a series of failures which results in shutoff of the electrical commands, full command capability is transferred to the mechanical backup system. The analog DEL control of the ailerons and rudders can be engaged only following the failure of three of the four digital processors. The analog DEL mode control laws are presented in Figure 9.

Figure 10 summarizes the failures which can result in reversion to the backup modes and the probability of failure for each case. From these computed failure rates it can be seen that the probability of reverting to a backup mode because of system component failure is very remote. Thus, EMI effects on system performance, such as shutdown or altered operation, and the potential for generic software failures become the major factors in the requirements for backup control systems.

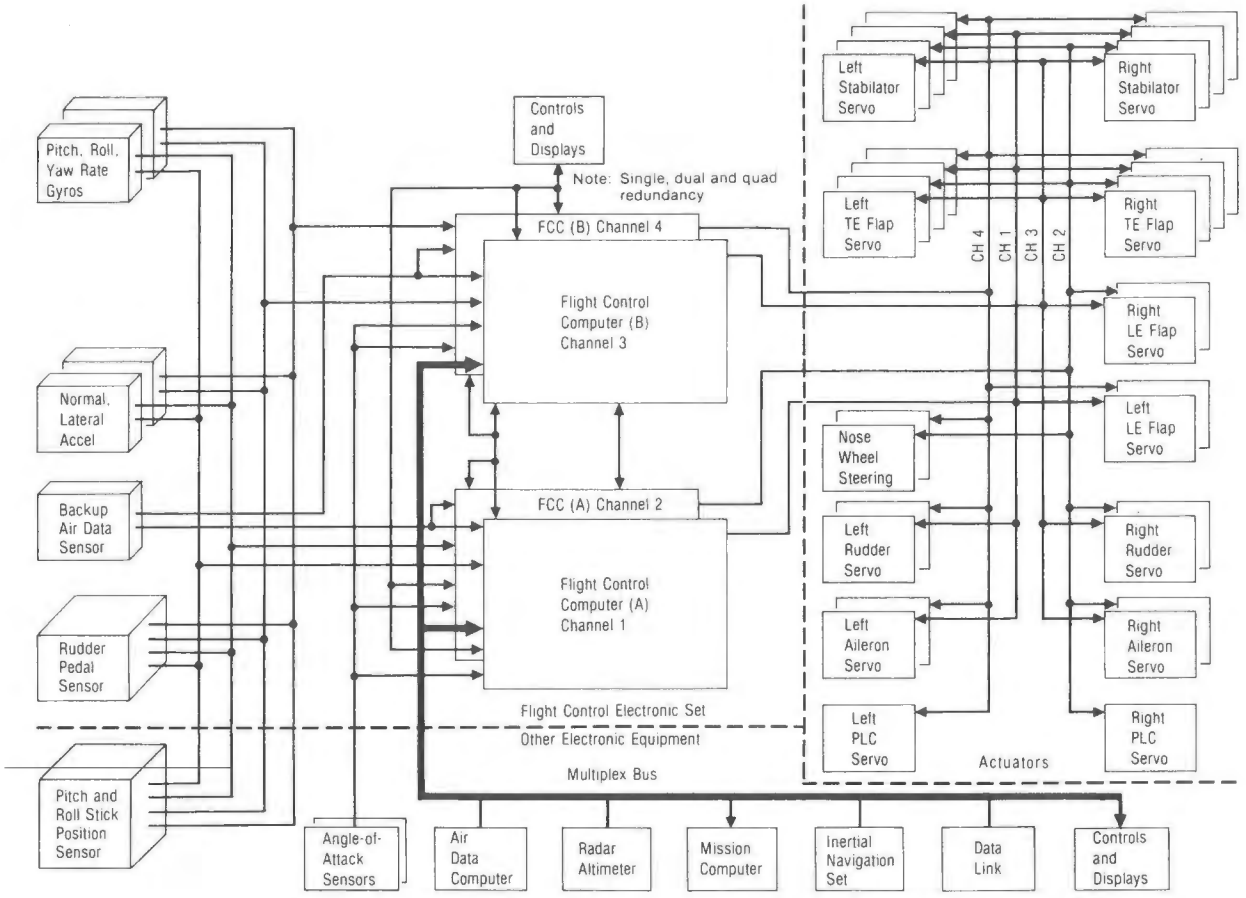


Fig. 3 Flight Control Electronic Set Interface

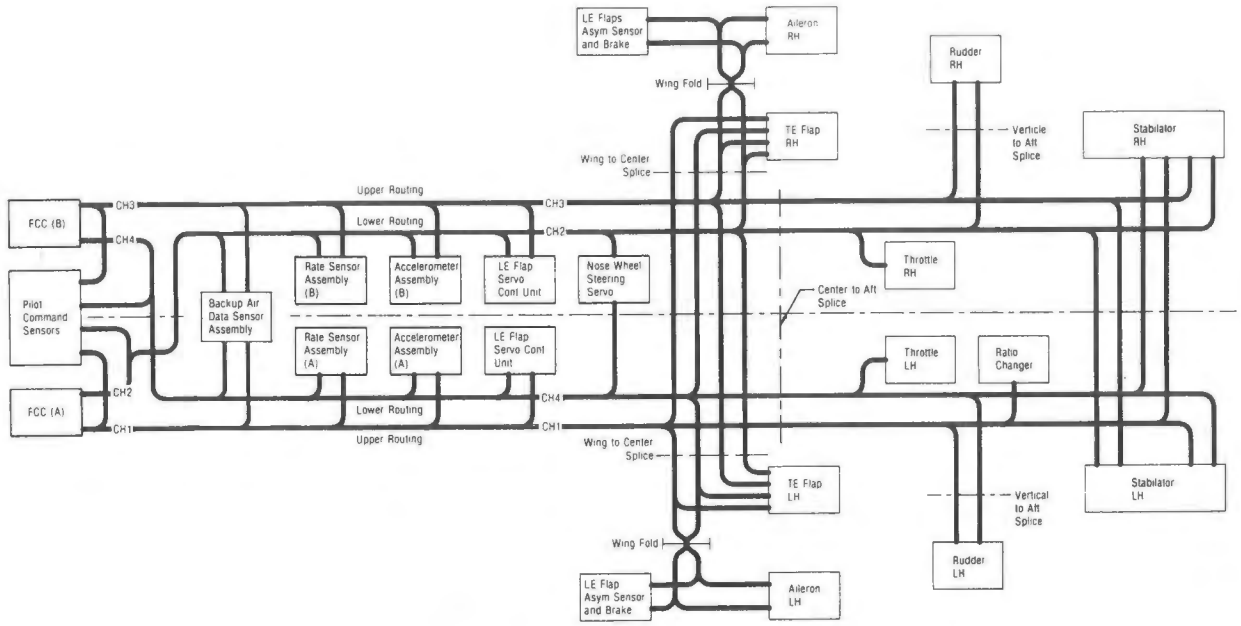


Fig. 4 Flight Control Channel Arrangement

Component	Redundancy	Capability
Flight Control Computers	Quadruplex	Two Fail-Operate/Fail Mech Mode
Cockpit Controls	Quadruplex	Two Fail-Operate/Fail Mech Mode
Accelerometers	Quadruplex	Two Fail-Operate/Fail Digital DEL Mode
Angle-of-Attack Sensors	Quadruplex Elec, Dual Mech	Two Fail-Operate/Fail Fixed Gain
Rate Gyros	Quadruplex	Two Fail-Operate/Fail Digital DEL Mode
Air Data Sensor	Quadruplex Elec, Dual Probes	Two Fail-Operate/Fail to Last Good Value, Fixed Gain Override
Stabilator Actuator	Quadruplex Elec	Two Fail-Operate/Fail Mech Mode
Trailing Edge Flap Actuators	Quadruplex	Two Fail-Operate/Fail to Zero Deg
Aileron Actuator	Dual	Fail-Operate/Fail Damped Trail
Rudder Actuator	Dual	Fail-Operate/Fail Damped Trail
Leading Edge Flap System	Dual	Fail-Operate/Fail Locked Position

Fig. 5 Primary Flight Control System Redundancy

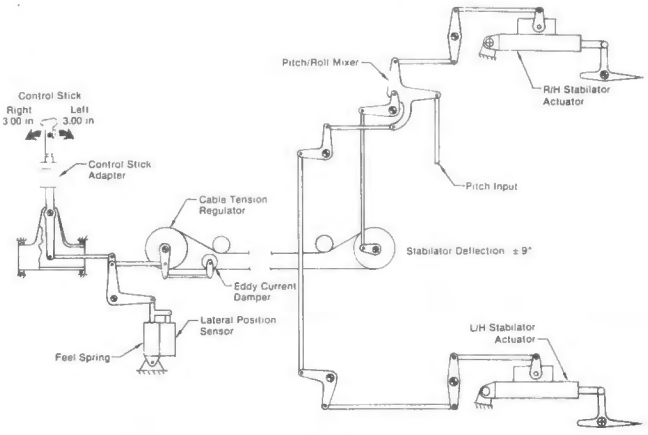


Fig. 8 Lateral Control System Schematic

- No Impact on Flying Qualities
  - — First or Second Failure of Quad Sensor or Actuator
  - — First Failure of Dual Actuator
  - — Computer "A" (Channels 1 and 2)
  - — Computer "B" (Channels 3 and 4)
- Degraded Control Augmentation
  - — Loss of Angle-of-Attack - Frozen at Last Good Value
  - — Loss of Air Data Sensor - Frozen at Last Good Value
  - — Third Lateral Accelerometer Failure
  - — Third Rudder Pedal Sensor Failure
- Digital Direct Electrical Link (DEL)
  - — Third Rate Sensor Failure (Pitch, Roll, Yaw)
  - — Third Normal Accelerometer Failure

Fig. 6 Degraded Mode Capability

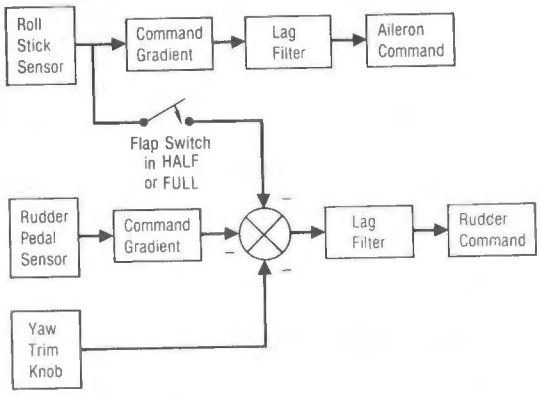


Fig. 9 Analog Direct Electrical Link Control Law

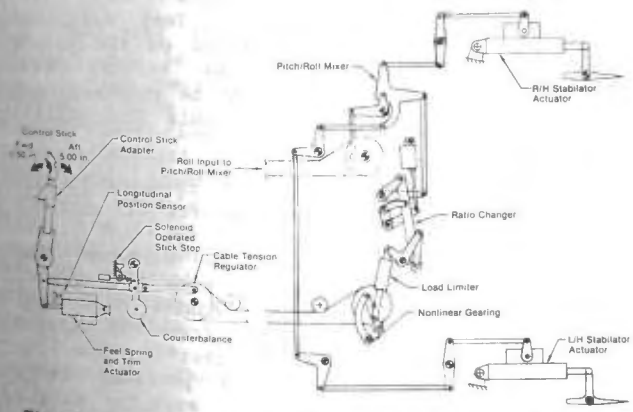


Fig. 7 Longitudinal Mechanical Control System Schematic

Backup Modes	Failures Causing Reversion	Probability of Reversion
Mechanical	Third Pitch or Roll Control Stick Sensor	3.2×10^{-14} Per 2 hr Flight
	Third Digital Processor	9.7×10^{-10} Per 2 hr Flight
	Third Computer Power Supply	1.6×10^{-9} Per 2 hr Flight
	Third Stabilator Actuator Servo Loop	7.7×10^{-7} Per 2 hr Flight
Analog DEL	Third Digital Processor	9.7×10^{-10} Per 2 hr Flight

Fig. 10 Reversion to Backup Modes

Electromagnetic Compatibility

The electromagnetic environment (EME) generated on aircraft carrier decks can be many times higher than those experienced by land based aircraft. Electromagnetic interference (EMI) was a major consideration in the design of the F/A-18 because of 1) the digital fly-by-wire flight control system and 2) the use of advanced composite structures which may degrade the shielding protection normally provided by aircraft skin. A major design emphasis was placed on electromagnetic compatibility (EMC).

The expected EME defined by the U.S. Navy is shown in Figure 11. These requirements were based on a survey of 13 aircraft carriers and have been adjusted for the anticipated EME through 1990. The design approach to meet these requirements is described in Reference (1). One of the major points in the design approach was the use of airframe as an EM shield. Testing had indicated that the carbon/epoxy aircraft skin provided significant shielding. As shown in Figure 12, as a result of MCAIR's accepting the responsibility for providing some EM shielding, the EME requirements for avionics components were significantly reduced, and these lower EME requirements permitted subcontractors to produce avionics that were lighter and more cost effective. Reference (1) describes other design approaches to reduce EMI effects such as: antenna EM radiation control, subsystem EMC control and ground plane interference requirements.

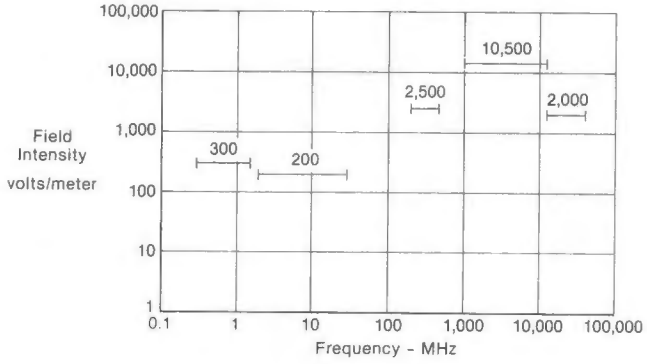


Fig. 11 Carrier Deck Environment Composite Transmitter Field Strengths

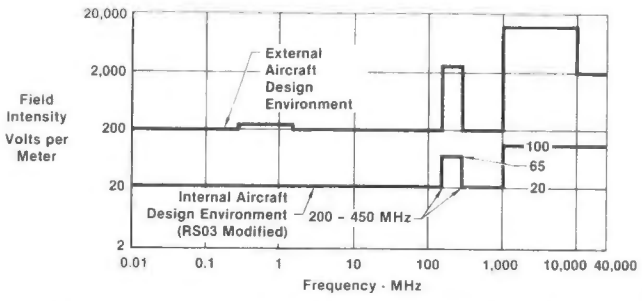


Fig. 12 Electromagnetic Environment Compatibility

The F/A-18 has been subjected to extensive ground and flight testing at the Navy EMC test facility, NATC Patuxent River, Maryland and during carrier qualifications. The F/A-18 has also been subjected to lightning testing at the Sandia National Laboratory. The effectiveness of the EMC design has been demonstrated by successful completion of these ground and flight tests wherein the F/A-18 met or exceeded all Navy imposed EMC requirements. The F/A-18 is completely qualified for day/night carrier deck operations in all-weather conditions.

Flight Control System Software Validation

Software errors which could cause total shutdown of the primary control system modes were a major factor in requiring backup modes. Concerns about software errors have led to an extensive verification and validation process which provides a high degree of confidence that the flight control system software will be error free.

The computer program for the F/A-18 flight control system is an assembly language program which performs all of the calculations for control system augmentation, autopilot functions, MUX bus communications, redundancy management, and built-in-test. The program performance is based on system requirements provided by MCAIR. The system requirements are translated into computer software requirements by General Electric (GE), who perform software coding and testing. Validation is a joint GE and MCAIR test effort. Figure 13 outlines the flight control system software validation process. These overlapping tests ensure with a high degree of confidence that the flight control system computer program will be error free and safe for use in flight test and production aircraft.

The GE software validation process is composed of three elements: module tests, hardware/software integration tests, and quality assurance review. In the module tests, small sections of the computer program are individually tested on a functional basis using an assembly language emulation on a VAX computer. All code paths are exercised. For the hardware/software integration tests, the actual flight control computers execute the software using a test bench hto supply the analog input/output to interface with the computers. Test procedures are written to ensure that MCAIR system requirements are fully satisfied. They cover all signal and logic paths, including software point-to-point tests and hardware input-to-output tests. Test data are compared to expected results based on the system requirements. All disagreements between test results and expected results must be resolved, and all test results are made available for quality assurance review.

MCAIR testing is performed concurrent with the testing at GE. The main source of software validation performed by MCAIR is the Flight Control Electronic Set Automated Software Test (FAST) system. The FAST system uses a Harrier/6 computer to provide excitation signals. A special architectural design of the control system software was required for this test method. The FAST system, mainly a FORTRAN program, reads special purpose command files to exercise flight controls software. The full test generates over

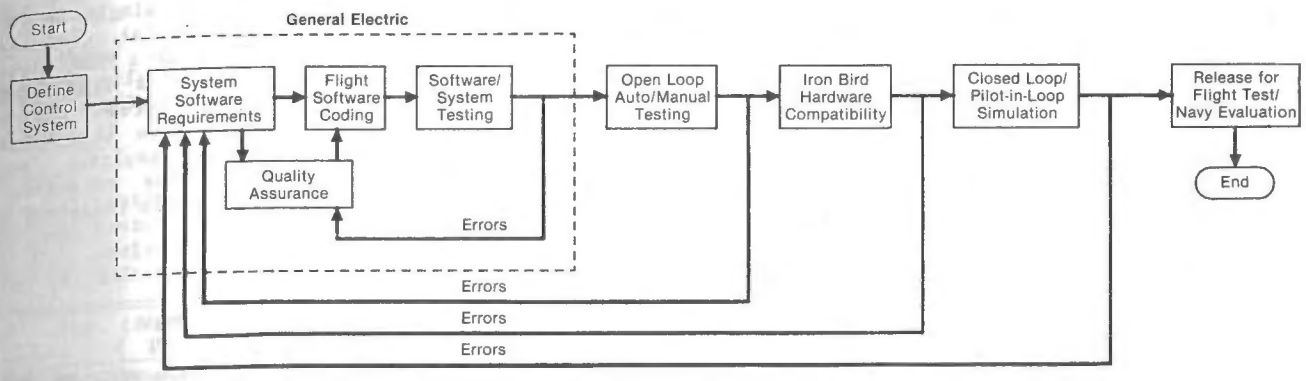


Fig. 13 Flight Controls System Software Validation Process

800 plot files in addition to many real-time data and logic tests. The complete FAST program runs automatically and requires 19 hours to complete. The test data generated are recorded back onto the Harris/6 for post-processing. A FORTRAN model of the flight control system is used for validating the test data. The model program executes the same test deck files as the flight control software and creates plot files similar to the test data. The test data are automatically compared to the expected results data, point-by-point. If the test data does not match the expected data within a specified tolerance, the data are then sent to a hard copy plotter for visual inspection. Figure 14 shows a set of data which matches the expected results and passes the automatic tolerance test. Typically, fewer than 20 plots require visual inspection. The FAST system provides a repeatable method for testing the flight control computer program and requires only a minimum amount of engineering time, just to start the test and analyze the final data. An analog test bench is also used to test parts of the system not available to the FAST system, such as input signal management, actuator signal management, power on recovery, cross channel data transfer, data management, and built-in-test.

Another MCAIR method for testing the flight control software is hardware compatibility tests on the iron bird. These tests provide hardware/software integration with the actuators and allow for a more realistic test of the built-in-tet portion of the software. Minor software changes which do not affect hardware compatibility can bypass the iron bird tests.

The FAST system and the iron bird tests perform open loop tests of the software. However, it is also necessary to verify that the flight control system operates properly in its mission tasks. This closed loop testing is done with the flight simulation facility, which provides the aerodynamic equations of motion that supply valid feedback signals for the flight control system. Also, the flight control computers are interfaced with the mission computer to porovide verification of MUX communications. A series of simulated pilot maneuvers at various flight conditions is used to verify proper system operation. Failure conditions are also tested in the closed loop environment. The final phase of the closed loop tests consists of a MCAIR test pilot flying various mission tasks. Altogether, these closed loop tests provide validation of the software that cannot be accurately accomplished by any other means.

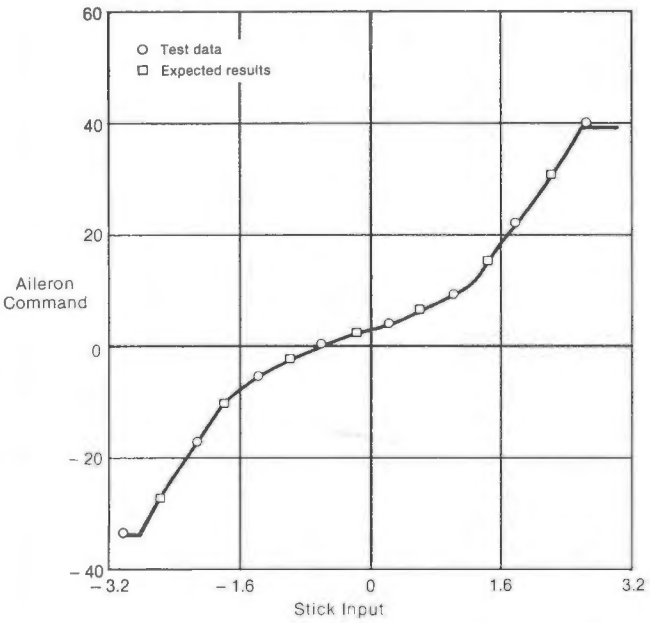


Fig. 14 Aileron Command to Roll Stick Input
LAIL vs CSLA Test No. 1.2.1 Step No. 4

When anomalies are found during the tests, an anomaly report is filed and sent to the software systems group at GE. They determine whether the anomaly is due to the software requirements not matching the control system requirements or to an error in programming. After the anomaly has been resolved, a new software tape is generated and tested. This process continues until the program is error free. After all of the test requirements are met, approval is given to GE to take the flight control software and create PROM's. The PROM's are then verified word by word against the software to ensure that the system to be flown is identical to the system that was tested.

In summary, the method for testing the F/A-18 flight control software gives a high level of confidence in the quality of the mission system. The MCAIR and GE testing provides cross-checks of the software which results in a complete check-out. The FAST system is an important part of the validation process because it uses a higher order language program to test the machine language program. The bench tests and Iron Bird tests provide a means of verifying the remaining portions of the software. The closed loop and pilot-in-loop tests provide the assurance that the software will perform properly in a typical mission task. This extensive software test program provides a high degree of confidence that the flight control system which will be used for flight is exactly as designed. The successful experience gained with our flight test program reinforces this position.

Impact Of Backup Modes On System Design

When designing a redundant flight control system with reversion mode capability the impact of backup modes capability must be considered. In the F/A-18 flight control system, there is a major distinction between degraded modes and backup modes. The degraded modes have very little impact on system cost and complexity because they make use of remaining system components following a series of failures or battle damage which prevent normal system operation. On the other hand, the backup modes were designed as dissimilar control paths and, as such, must minimize the use of hardware used during the normal modes of operation. The requirement for backup control modes had significant impact on the design of the basic control system.

The backup analog DEL modes required that the aileron and rudder actuators be designed with analog servoloop feedback and analog failure monitoring. Because the normal system was designed with analog servoloop electronics, there were no significant weight penalties. However, there were penalties for providing failure detection and switching logic which increased the flight control computer cost.

The mechanical backup control system requirement resulted in significant weight and cost penalties. The mechanical backup mode consists of all of the conventional push rods and cables from the control stick to the stabilator actuators and the command select mechanism which is built into the actuators to enable switching between the normal mode and the backup mode. Figure 15 summarizes the cost, weight, and software penalties resulting from the backup modes. Also, there are significant nonrecurring costs associated with design, development and flight testing of backup control systems.

Experience With Degraded/Backup Modes

As of mid-summer 1984 more than 200 F/A-18 aircraft have been delivered, and they have accumulated more than 70,000 flight hours. During this time there have been no reversions to a degraded or backup mode due to random system component failures. To date, the only failures encountered have been first level failures. These are single channel sensor failures, single servo-actuator failures, or single digital computer processor failures. There have been a few cases of reversion to degraded modes as a result of damage to the angle of attack probe during inflight refueling. Also, there have been some servo-actuator failure indications resulting from failure monitor thresholds or time constants, which fall into the category of nuisance failures.

Cost and Weight

	Recurring Costs (Percent of Total)		Weight (lb)		
	Computers	Actuators	Computers	Mechanical Controls	Actuators
Analog DEL (Ailerons/ Rudders)	2	—	0.5	—	—
Mechanical (Stabilators)	1	12	0.5	43	40

Software

	Memory (Words)		BIT Test Time (sec) (Total Test Time 120 sec)
	BIT	Redundancy Management	
Analog DEL (Ailerons/Rudders)	600	100	9
Mechanical (Stabilators)	1,500	50	48.7

Fig. 15 F/A-18 Backup Control Mode Penalties

Four of these nuisance failure indications have resulted in reversion to the mechanical backup mode. Only one of these reversions occurred in flight, whereas the other three occurred during ground operations. The reversions were related to a problem in the stabilator actuator mechanical mode command select mechanism. If the actuator had not had a mechanical mode, the conditions which resulted in the reversion would not have resulted in a degradation in actuator response or a failure indication. Design changes have been made recently to eliminate this type reversion problem.

In 5-1/2 years of flying the F/A-18, only two noteworthy errors in the software were detected in flight. The first involved a coding error in a gain schedule which defined the gain below 1.01 MACH and above 1.0 MACH, but not at all exactly at 1.0 MACH. When flying at exactly 1.0 MACH, the flight control computers declared a processor error due to overflow and immediately went into a recovery mode. The recovery was accomplished successfully and no degradation occurred in flying qualities. The second software error to be detected in flight involved the degraded mode logic which controls the limiting of control surface commands at high load factors. This could result in higher loads than expected on that surface, but the loads would remain below the design limit.

Summary

There are four primary factors considered when specifying backup control modes for aircraft with redundant fly-by-wire flight control systems. These are:

- o System reliability
- o System survivability
- o Environment
- o Software verification

The first two categories are under direct control of the designer. The system reliability can be calculated based on data from systems with comparable components, and redundancy can be added until the system requirements can be met. The system survivability for the specified threat can be met through separation of redundant channels or components and through shielding of critical components either with aircraft structure or with components in less critical systems. The need for a degraded mode capability can be established by performing suitable tradeoffs between aircrew safety and system cost.

However, the real "drivers" in requirements for backup control modes are the environment and software verification. The term environment includes the electromagnetic as well as the atmospheric environment. During design and qualification of the F/A-18 these drivers were given a higher priority, in an attempt to preclude situations which might require reversions to a backup control mode.

F/A-18 experience shows that even the simplest backup modes impose major penalties in terms of complexity, weight, and cost. Based upon probabilities of failure predictions the use of a backup system with dissimilar redundancy cannot be justified. Therefore, it is the responsibility of the control system designers and the customer to: 1) sufficiently and realistically define all of the environmental factors, 2) design and test to assure compliance with those requirements, and 3) make sure that adequate software verification and validation is performed to obviate the possibility of catastrophic generic software faults. Therefore, if we in the aircraft industry do our job properly, there should be minimal requirement for backup control system modes, which generally increase cost and complicate the flight control system design.

1. J. R. Ketterer and J. J. Fisher, "The Navy F/A-18 Hornet Electromagnetic Compatibility Program", NAECON, Dayton, Ohio, May 1981.

INTEGRATED FLIGHT CONTROL/NAVIGATION SENSORS

by

Robert E. Ebner
Shou Y. Wei, Ph. D.
Litton Systems, Inc.
Woodland Hills, CA 91365

Abstract

Litton is currently in the fabrication and test phase of the U.S. Navy's Integrated Inertial Sensor Assembly (IISA) program. IISA is a six ring laser gyro, six accelerometer system suited to combined navigation and flight control sensing needs of modern high performance fly-by-wire aircraft. The status of system development and preliminary test data are presented, extending that given by the author at the 1983 DASC in Seattle. The background of integrated navigation/flight control sensors is given along with a review of a spectrum of possible system configurations as a function of avionics redundancy requirements. Noise on angular rate and acceleration outputs from sources such as gyro dither vibrations, linear and angular effects from external vibration as modified by vibration isolators, and amplification of noise by lever arm compensations are major concerns to flight control design. Anti-aliasing filter characteristics plus laboratory and simulation data are presented in the paper. Simulations of transients in sensor outputs which might be expected during redundancy management switching are also presented. A photograph of the actual IISA unit is shown.

Background

Aircraft having gimballed inertial navigation or attitude heading reference systems commonly had completely separate angular rate and acceleration sensors for use by the flight control system. Angular rate derived from gimbal angles had insufficient accuracy and/or bandwidth for critical stability augmentation control loops. Because of the relative complexity of INS, reliability was also insufficient for flight safety without redundancy, unwarranted at that time for primary INS functions. Therefore, separate dedicated sensors were included for angular rate and acceleration inputs to the flight control system. This approach had the added benefit that the sensor location on the aircraft could be optimized for loop stability during flight test without costly modifications to analog computer transfer functions in the FCS computers.

A number of factors in avionics and aircraft designs have changed in recent years, however, and more efficient avionics partitioning methods can now be considered. These factors are as follows:

1. Inertial navigation systems are now available in nongimballed (strapdown) form usually utilizing ring laser gyros. Angular velocity and acceleration outputs in the aircraft coordinate system are available with required accuracy and bandwidth.

2. Flight control systems are now designed around digital computers making optimization for a given aircraft/sensor configuration more flexible.

3. Mechanical linkages between the pilot and control surfaces are becoming minimal or eliminated altogether (fly-by-wire), leading to complete reliance on inertial sensors, with attendant redundancy requirements for flight safety.

4. Skewed-axis approaches to redundancy have been proven to be a low cost alternative to straightforward axis-by-axis replication (1) (2) (3).

5. Inertial velocity data has become important to flight safety in both commercial and military applications such as for wind-shear control during landing for the former, and terrain-following or integrated fire/flight control functions for the latter. This leads to a requirement for redundancy in the inertial navigation function to meet flight safety probabilities.

These above considerations lead to the conclusion that the inertial navigation and flight control sensing requirements in future aircraft avionics should no longer be viewed as separate in order to achieve minimum avionics cost and weight and maximum aircraft performance.

The commercial community has already begun to take advantage of redundant RLG INS outputs for flight control. The introduction of the concept to military aircraft, however, appears to be progressing very slowly. While RLG INS is close to acceptance in U.S. Navy standards, features needed for use of these systems for flight control are not currently included. Skewed-axis configurations are planned for new aircraft but reductions of R&D funding may preclude their use for another generation.

The particular concerns that need to be validated prior to acceptance for a production aircraft are in the use of sensors whose location in the aircraft is not necessarily ideal for flight control. Accelerometers and gyros are colocated in an INS so one can't be put at a bending node and another at an antinode as it is currently done. The INS is also larger than typical FCS sensors, limiting installation flexibility. While simulations by various aircraft companies have indicated that software compensation for bending mode effects is practical, there is a very limited amount of real flight test data. Until this is obtained, there will be considerable reluctance to deviate from standard, proven methods in an area where flight safety is a major concern. Programs currently addressing these considerations are the Integrated Inertial Sensor Assembly (IISA) program (by Litton for the U.S. Navy), the Multifunction Flight Control Reference System program (by McDonell Douglas Corporation for the U.S. Air Force), and the Integrated Sensory Subsystem program (by Grumman for the U.S. Navy). These are described in References (1) through (6).

Spectrum of Configurations

There are a number of ways of integrating the INS with a flight control system depending on which functions are assumed to affect flight safety. If one assumes that the basic FCS redundancy requirement for angular rate and acceleration sensing is fail-operational/fail-operational/fail-safe, a spectrum of configurations can be devised as a function of the amount of redundancy required on attitude and navigation (velocity sensing) functions. Table I shows four different system configurations with variable levels of attitude/velocity redundancy.

Configuration 1

Configuration 1 assumes that neither attitude nor velocity are required for flight safety. The INS can be nonskewed relative to aircraft pitch/roll/yaw axes for minimum size. A companion unit would be required containing three gyros and three accelerometers, skewed relative to the INS axes so that no two are coincident and three are not in the same plane. This is achieved if the net six gyros and six accelerometers are equally spaced on a 54° cone (half-angle).

While the INS may be conventional in most respects it must have isolated electronics between the three axes, with no single point failure mode. It also must contain anti-aliasing filters on angular rate and acceleration outputs, solved at greater than twice the highest vibration frequency. Since all operational RLGs are dithered, solution rates must typically be greater than 800 Hz.

The rate gyro/accelerometer package should contain similar channel isolation and identical sensor anti-aliasing filters. Transfer functions including filters must be very similar to those of the INS to avoid complications in the redundancy management during transient conditions.

Configuration 2

Configuration 2 would be virtually identical to configuration 1 except that a computer is added to unit 2 to solve the strapdown equations in order to derive the aircraft attitude and heading. Considering the capabilities of modern computers, this is a relatively minor addition. Sensor accuracy would probably need to be somewhat better than configuration 1.

Configuration 3

This is the IISA design mechanization and is described extensively in Reference (4). It consists basically of two specially-designed INS units to provide the required FO-FO-FS angular rate and acceleration outputs, in addition to dual attitude and navigation outputs. The design features of the INS included to achieve this capability are:

1. Skewed sensors
2. Channelized partitioning (1 gyro, 1 accelerometer each)
3. Anti-aliasing filters solved at 1024 Hz, outputs at 1024 Hz
4. Fail-operational synchronization within each channel
5. Improved-resolution gyro and accelerometer outputs.

Configuration 4

FO-FO-FS navigation can be achieved by adding one gyro and one accelerometer to each INS. Since the four sensors within an INS are bolted together rigidly, with proper geometry good navigation can be achieved using any three of the four. Thus with four navigation solutions for each INS (eight total) plus one parity equation in each, degradation of navigation accuracy can be detected and isolated to the failed sensor. Computers must be added to maintain FO-FO-FS capability and throughput requirements are increased for the added solutions.

Summary

Use of INS sensors to reduce or eliminate special flight control sensors can be done in a number of ways (depending on the amount of redundancy needed in INS functions) by taking advantage of skewed-axes techniques. In order to be adaptable to such reduced-avionics configurations, however, INS designs must contain special provisions such as channelized design and anti-aliasing filters. Barring unforseen difficulties, the major impediment to implementation of such an avionics mechanization appears to be managerial - assuring a good background of test data, making the decision as to what level of navigation redundancy is required, and factoring the resulting equipment requirements into the specifications.

IISA Description

Configuration 3 above has been built and is being tested by Litton for the U.S. Navy under the name of IISA (Integrated Inertial Sensor Assembly). The sensors are contained in two Inertial Navigation Assemblies (INA), each of which provides full, independent inertial navigation outputs.

Within an INA, sensor axes are orthogonal but skewed relative to the aircraft yaw axis (see Figure 1). One accelerometer and one gyro in an

TABLE I
CONFIGURATION VS ATTITUDE/VELOCITY REDUNDANCY

Config	Attitude Redundancy	Velocity Redundancy	Unit 1	Unit 2
1	None	None	INS (nonskew)	Sensors only (skew)
2	Dual	None	INS (nonskew)	SD AHRS (skew)
3	Dual	Dual	INS (skew)	INS (skew)
4	Quad	Quad	INS (3NS+1S)	INS (3 NS+1S)

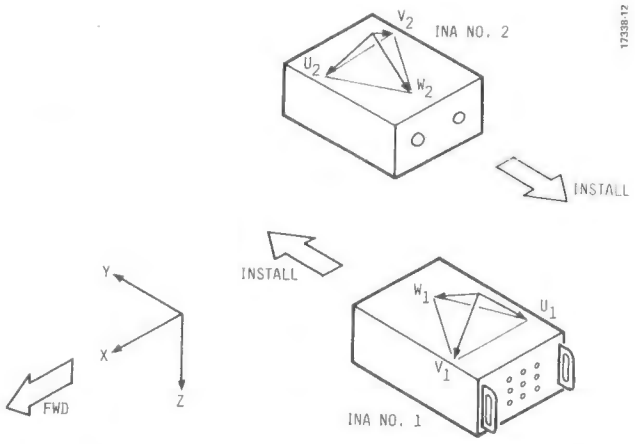


Figure 1. INA Installation Configuration

INA are oriented along each skewed axis. Figure 1 depicts the orientation of axes when the INA are installed into the equipment bays of the aircraft. When one INA is installed into the right equipment bay, with 180° rotation about yaw relative to the identical left INA, the six sensor axes are then distributed uniformly about a 54.7° half-angle cone. No two axes are coincident, nor are three in the same plane. Thus, any three sensors may be used to derive three-axis outputs in aircraft axes after suitable computer transformation.

An INA is divided into three, largely independent channels. Each channel contains data from one gyro and one accelerometer plus related electronics, a preprocessor, provisions for output of data to the FCS and to the navigation computer, and independent low/high voltage power supplies. The navigation processor and its MIL-STD-1553B I/O are on the same power supply as one of the three sensor pair channels.

The packaging arrangement for the INA is shown in Figure 2. The three channels of electronics are physically separated to eliminate commmon failure modes. Wiring from the sensors to the sensor electronics is also kept physically separated to avoid short-circuit, EMI, etc., failure modes common to two channels.

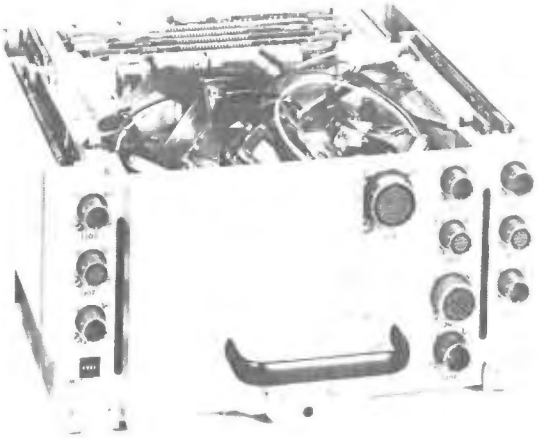


Figure 2. Inertial Navigation Assembly

The installation characteristics of an INA are:

Weight:	54 lbs
Dimensions:	14 x 14 x 7 inches
Power:	135 watts

Rate and Acceleration Noise from Gyro Dither

IISA utilizes ring laser gyros that depend on a small amount of angular mechanical dither motion to avoid lock-in between CW and CCW lasers to achieve full navigation accuracy. This mechanical dither produces accelerations sensed by the accelerometer. If nonlinearities occur somewhere in the process, difference frequencies between gyro dithers or aliasing with the sampling frequency can cause low-frequency beats to occur on acceleration outputs that could enter the flight control system and cause wander or actuator flutter.

IISA anti-aliasing filters are solved at an iteration rate of 1024 Hz, and consist of cascaded walking average filters. Filter sections are of different lengths to produce a good low-pass response with minimum time delay. Bandwidth is essentially 23 Hz with an additional time delay of 9.7 milliseconds. Attenuation at dither frequencies is over 80 dB.

The acceleration output from the actual operational IISA hardware with gyro dithering is random with a noise amplitude of 0.05 ft/sec^2 (less than 2 milli -g). Figure 3 shows typical Fast Fourier Transforms of the data showing the near white-noise response. No peaking due to gyro dither vibrations or beat frequencies is discernible.

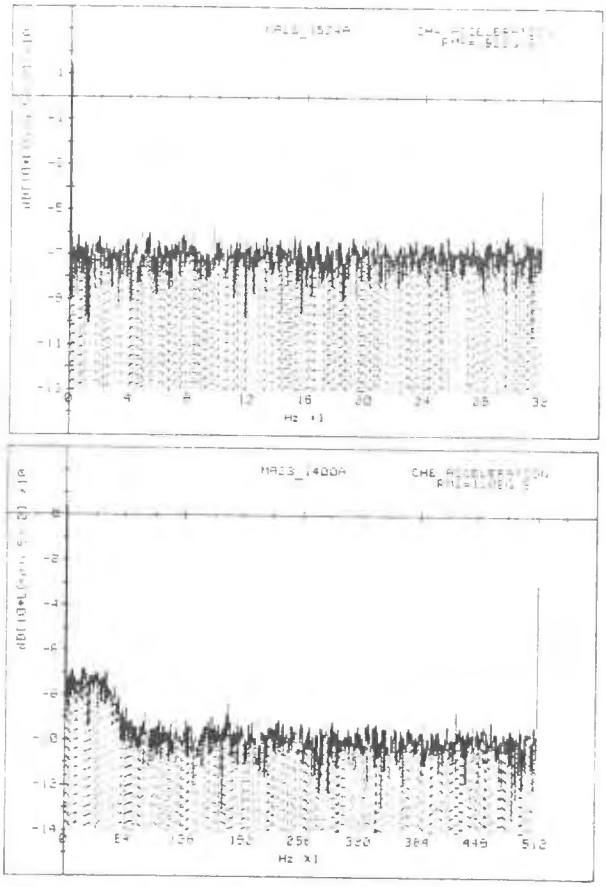


Figure 3. FFT of IISA Acceleration Noise with Three Gyros Dithering (418, 424.5, 428 Hz)

The primary cause of the acceleration noise is accelerometer quantization. Acceleration measurements are converted to digital form in the feedback path of the force rebalance loop. The quantization level of 0.0015 ft/sec at each end of the sample combined with the effective sample time of 0.017 seconds produces a maximum quantization error or 0.18 ft/sec^2, leading to very close to the measured 1-sigma value of 0.05 ft/sec^2. The gyro dither serves to randomize the error which would otherwise be a fixed-frequency for a given g-level. If an application requires a wider bandwidth with similar static acceleration noise, accelerometer quantization step size can easily be reduced.

Redundancy Management Performance

The six axes of skewed angular rate and acceleration are sent to an external computer for redundancy management processing to derive body-axis rates and acceleration, free from the effects of any two hard or soft sensor failures. In a production aircraft, this processing would probably be done in the flight control computers. In the IISA ADM, a separate unit is provided for easy adaptability to existing FCS for flight demonstrations.

The redundancy management consists of parity equations, sensor selection logic, and design equations. The parity equations combine the outputs of four sensors at a time (15 equations total) in such a way as to cancel vehicle rate (or acceleration) and thus expose sensor errors. Sensor errors above a predetermined level (threshold) are defined as failures. Sensor selection logic considers the state of all the equations and determines which sensors are to be used to derive angular rate and acceleration outputs. Based on selected sensors, design equations are used to derive the required outputs, removing skew angles, and accounting for redundant data where applicable.

The quality of the redundancy management process rests on:

1. Noise level of parity equations
2. Thresholds that are used to detect failures
3. Transients that may occur in outputs when failures occur or if different sensors are selected due to normal noise conditions.

The IISA ADM will not be ready for evaluation of these parameters until 1985. Realistic simulations have been performed, however, to evaluate the effects of factors such as vibration isolators, anti-aliasing filters, and misalignments on the redundancy management process.

Redundancy Management Simulator

The redundancy management simulator consists of three programs: 1) trajectory simulator, 2) system response simulator, and 3) parity and design equation simulator. The overall flow chart is presented in Figures 4 and 5. The process presented in Figure 4 is required for both INAs independently.

Trajectory Simulator. The trajectory simulator contains six channels to represent linear and angular motion. Each channel has two integrators to integrate an acceleration signal to velocity and then position. The coupling effect of pitch rate and forward velocity (in X direction) into Z acceleration is included. The lever arm effect that results in an acceleration due to angular

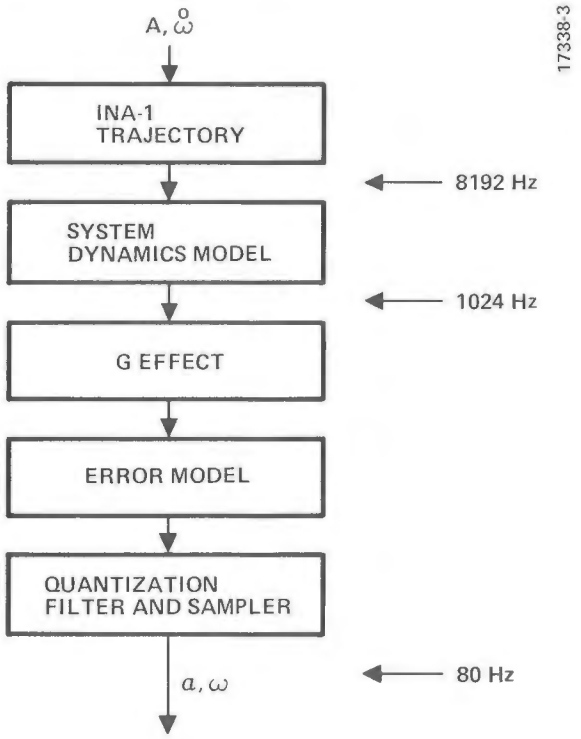


Figure 4. Trajectory Flow Chart and System Response Simulator

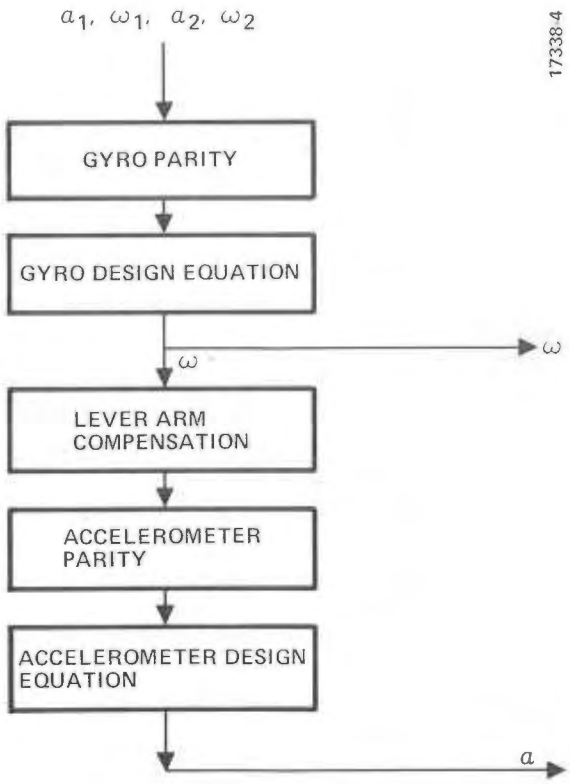


Figure 5. Parity Flow Chart and Design Equation Simulator

motion is also simulated. The capability of making a coordinated turn is included.

System Response Simulator. The system response simulator is based on the sensor assembly model that results from rigid body dynamics, including effects of vibration isolators and gyro dither mechanisms. The input excitation is obtained from the trajectory simulator. For trade-off study, the following parameters can be varied: 1) each isolator frequency and damping, 2) each gyro dither frequency and damping, 3) each isolator location, 4) cg location uncertainty (i.e., cg off-set), 5) mismatch of gyro dither center and cg of the assembly, 6) mass and the moment of inertia tensor, 7) each gyro dither excitation.

The equation of motion can be summarized in the form of:

$$\dot{X} = AX + B_1U + B_2U$$

where

$$X = \begin{bmatrix} \vec{V}_B \\ \vec{\theta}_B \\ \vec{\phi}_g \end{bmatrix} \qquad U = \begin{bmatrix} \vec{V}_s \\ \vec{\theta}_s \\ \vec{\tau} \end{bmatrix}$$

and

$\vec{V}_B$ = linear displacement of sensor block

$\vec{\theta}_B$ = angular displacement of sensor block

$\vec{\phi}_g$ = angular displacement of gyros along the dither axis

$\vec{V}_s$ = linear displacement of the shelf where the sensor assembly is mounted (resulting from the trajectory simulator)

$\vec{\theta}_s$ = angular displacement of the shelf (resulting from the trajectory simulator)

$\vec{\tau}$ = gyro dither excitations.

A, B_1, and B_2 are coefficient matrices.

To obtain the linear and angular motion of the block and gyro, the equation of motion is integrated using the transition matrix of the system.

Parity and Design Equation Simulator. This simulator consists of: 1) sensor error model, 2) parity equations, and 3) design equations. Sensor errors of misalignment angles, bias, and scale factor can be simulated. All 15 parity equations are included, and one of 29, 3 x 6 design equations can be selected based on sensor selection logic, to derive simulated system outputs.

Simulation Results

A parametric simulation has been performed based on a trajectory of a snap roll into a high-g coordinated turn. The effects of isolator imbalance and mismatch, unit misalignment, and large installation lever arms were independently evaluated. Samples of the final computer run with all mechanisms present simultaneously are now presented. Table II shows the assumptions used in this simulation.

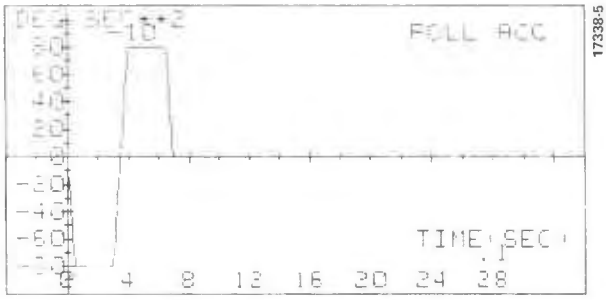
TABLE II
REDUNDANCY MANAGEMENT SIMULATION

- Aircraft velocity, 1000 ft/sec
- Snap roll (800°/sec^2) into 7-g coordinated turn
- Actuator response time, 0.05 second
- Lever arms
  - INA-1; X = -10, Y = -2, X = -1 ft
  - INA-2; X = 0, Y = 1.5, Z = 0.5 ft
- Vibration isolators
  - INA-1; 35 Hz resonance, 0.05-inch cg displacement
  - INA-2; 30 Hz resonance, no cg displacement
- Misalignments
  - INA-1; 0.2° yaw
  - INA-2; 0.2° roll
- Filtering
  - Gyro and acceleration measurements: 17 millisecond anti-aliasing filters
  - Parity equations: 25 millisecond low-pass filters

Flight Profile. The trajectory is driven by a roll acceleration with the time profile of Figure 6. Velocity is constant at 1000 ft/sec along the X (forward) direction, and a coordinated turn is started at time t=0 seconds. The total mission duration simulated here is 1.5 seconds. From Figure 6, it can be seen that the actuator response time is approximately 0.05 seconds.

With the roll acceleration described above, the roll, pitch, and yaw rates during the coordinated turn are presented in Figure 7. The linear accelerations due to the coupling of the coordinated turn, the pitch rate, and the lever arm effect (10 ft, 2 ft, 1 ft) are presented in Figure 8.

Parity Equation Outputs. The outputs from two gyro parity equations are shown in Figure 9 and are typical of the remainder. Equation Tij8 combines outputs from the U and W axes of each unit (Figure 1), while



17338-5

Figure 6. Roll Acceleration of Simulated Trajectory

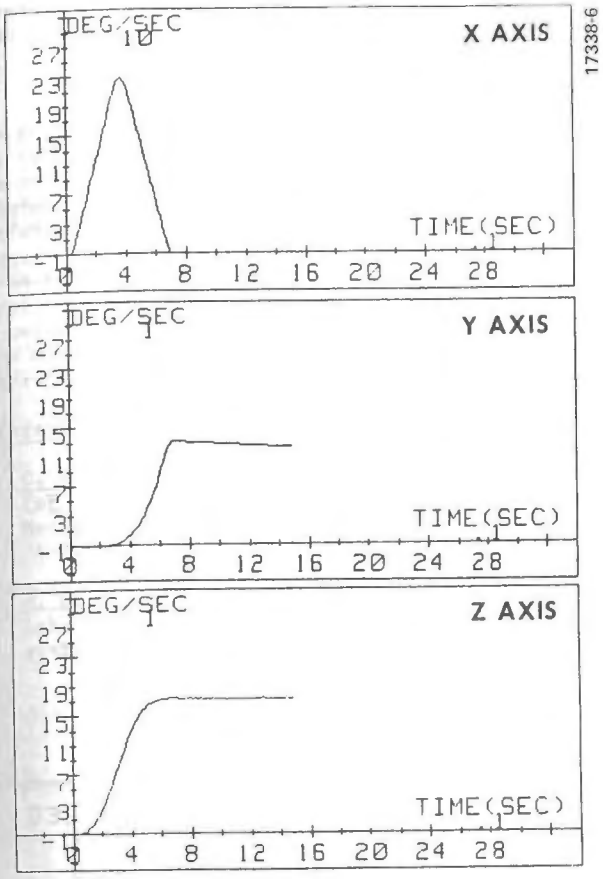


Figure 7. Simulated Maneuver, Angular Rate Profile

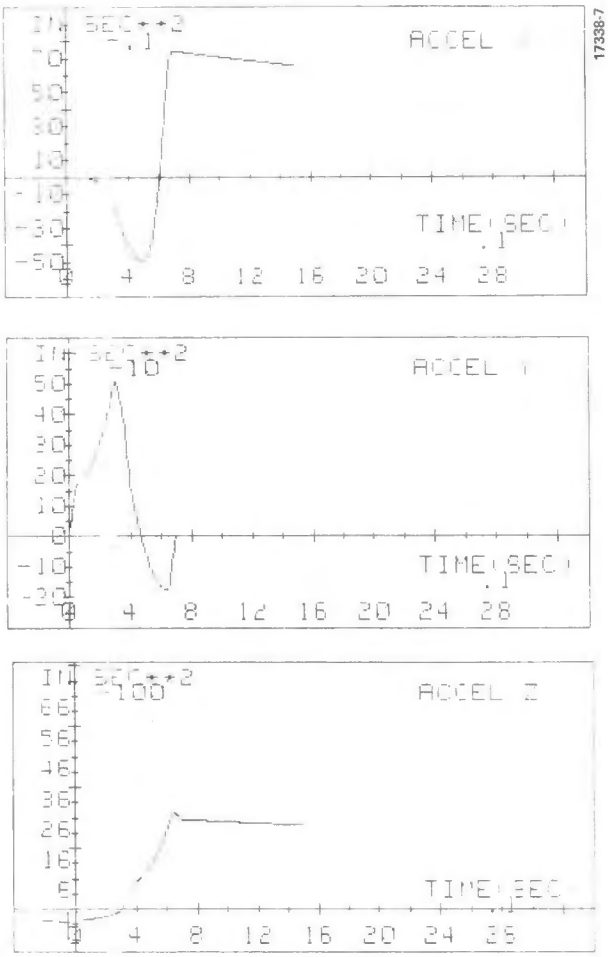


Figure 8. Simulated Body Axis Acceleration

Equation Tij9 uses both U and V axes. The dominant error of Tij8 is the misalignment angle between units. The error of equation Tij9 occurs mainly from the vibration isolator frequency mismatch.

Accelerometer parity equation outputs (Figure 10) uses axes W_1 with U, V, and W_2 for Tij3, and V_1 with U, V, and W_2 for Tij2. Errors are mainly due to lever arm compensations and anti-aliasing filter lags.

Output Transients During Sensor Failures. During a soft sensor or sensor I/O failure, an output transient can occur due to: 1) use of different design equations involving a different set of sensors causing a change in the propagation of normal errors to the output, and 2) sudden removal of an acceptable soft error that may have been present for some time prior to finally exceeding the parity equation thresholds.

The former error was evaluated during the simulation by solving two sets of design equations simultaneously and differencing their results. The design equations selected are one using two sensors in each unit and the other using two sensors in the left unit and one sensor in the right unit. The differences between the two output computations are shown in Figure 11 for angular rate and Figure 12 for acceleration. These curves represent envelopes of possible transients where the actual transient depends on when, in time, the sensor failure occurred.

The latter error is a function of the failure thresholds used on parity equations. These thresholds

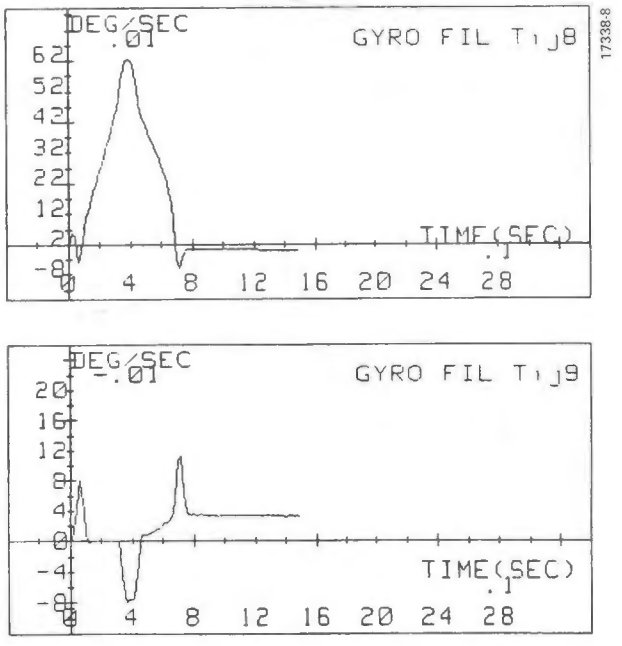
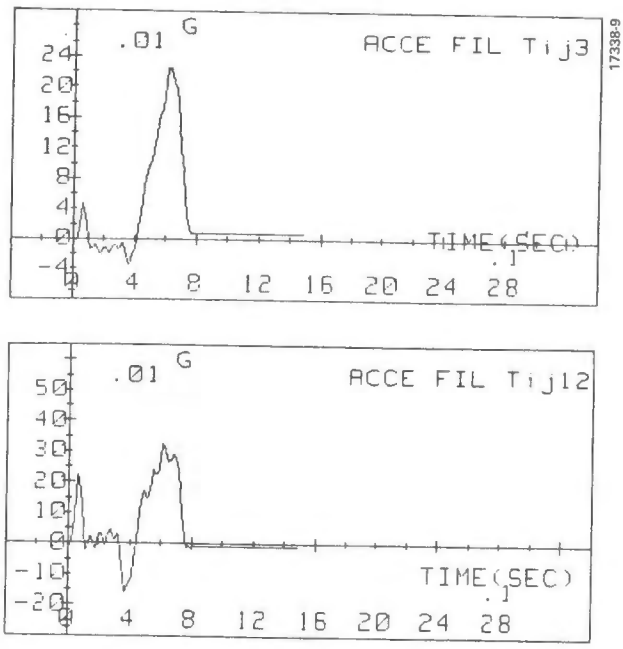
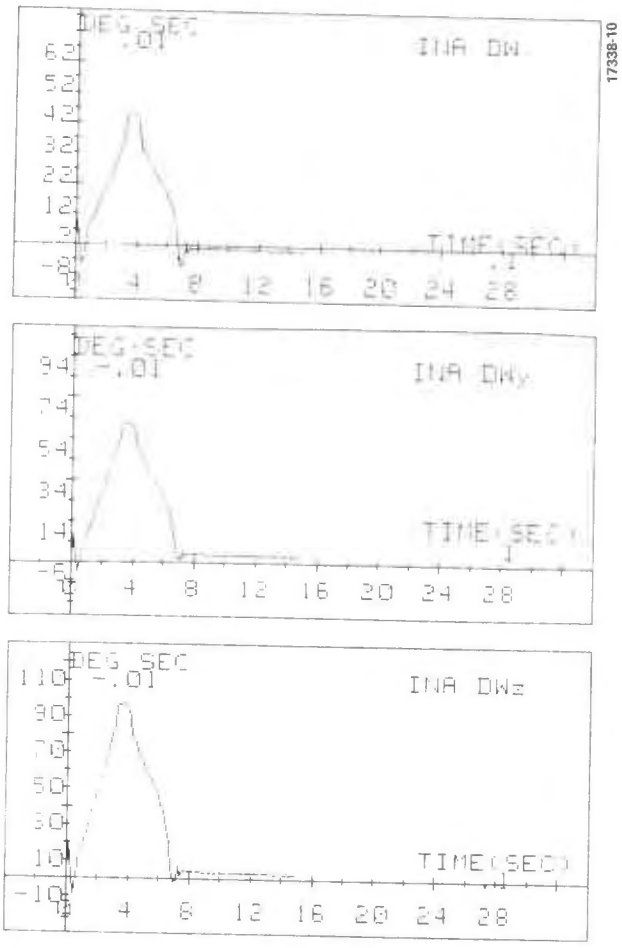


Figure 9. Sample Gyro Parity Equation Outputs



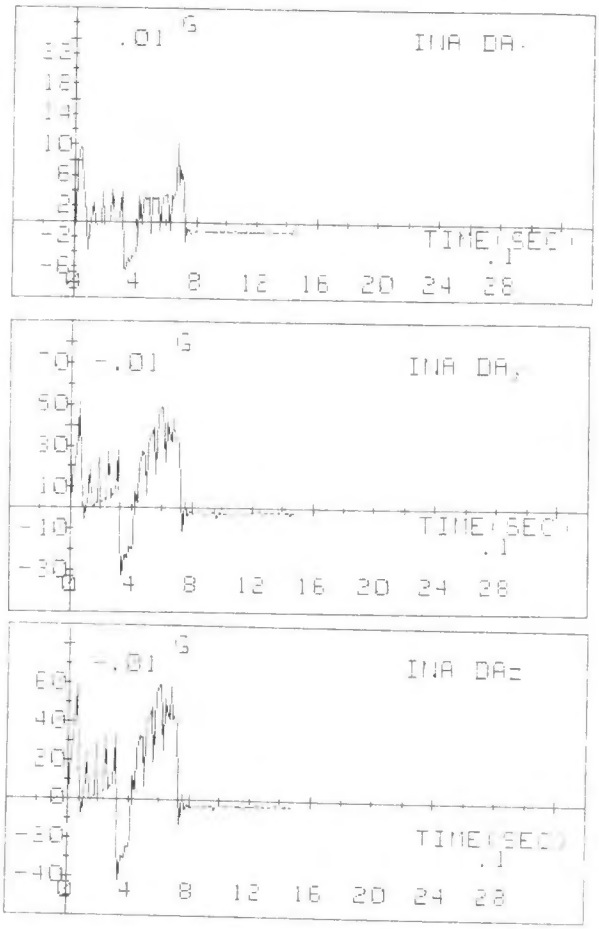
17338-9

Figure 10. Sample Accelerometer Parity Equation Outputs



17338-10

Figure 11. Maximum Angular Rate Output Transient



17338-11

Figure 12. Maximum Acceleration Output Transient

will be made variable as a function of angular rate, acceleration, and lever arms. They will be set high enough so that false alarm error detections will be extremely improbable. Thus, transients following slowly increasing soft errors could be two to three times those indicated by Figures 11 and 12.

Simulation Conclusions. For the extreme maneuver that was simulated, the following maximum conditions occurred:

	Parity Noise	Switching Transient
Angular rate (degrees/sec)	0.6	1.0
Acceleration (g)	0.35	0.6

These levels are relatively small considering the high g-forces and transients from the maneuver itself, and should not lead to loss of control or pilot complaints.

The major sources of these effects are misalignments and vibration isolators for angular rates, and lever arm compensations with anti-aliasing filter lags for acceleration measurements.

Perturbations of parity equations also occur due to vibration and fuselage bending modes. Simulations have not been performed at this time; however, the fuselage bending effects are expected to be quite significant for applications with widely separated

sensor units. Methods such as described in Reference (3) may be needed for flight control compensation and to minimize parity equation noise.

Overall Conclusions

The design and test of a system suitable for shared use between navigations and flight control functions has progressed significantly. Both laboratory and simulation data are available that support required performance. Effects of fuselage bending modes, however, require further investigation. Acceptance of an integrated sensor concept into future aircraft requires a good background of aircraft test data that is currently lacking. Absence of this data may postpone use of integrated sensors for another aircraft generation.

References

1) C. R. Abrams and W. D. Weinstein, "A New Concept for Angular Rate Flight Control Sensors," AIAA, Mechanics and Control of Flight Conference, Paper 74-868, Anaheim, Ca, 5-9 August 1974.

2) W. K. Toolan and A. M. Zeslin, "Development and Laboratory Test of an Integrated Sensory Subsystem (ISS) for Advanced Aircraft." Proc. 4th Digital Avionics Systems Conference, St. Louis, Missouri, November 1981.

3) W. K. Toolan and K. H. Grobert, "Development and Simulation Testing of an Integrated Sensory Subsystem (ISS) for Advanced Aircraft (Phase III)," Proc. 5th Digital Avionics Systems Conference, Seattle, Washington, November 1983.

4) David Krasnjanski and R. E. Ebner, "Design of the Integrated Inertial Sensor Assembly (IISA) Advanced Development Model," Proc. 5th Digital Avionics Systems Conference, Seattle, Washington, November 1983.

5) Capt. J. T. Young, J. M. Perdzock, and D. L. Sebring, "Design and Development of the Multi-Function Flight Control Reference System," Advisory Group for Aerospace Research and Development (AGARD), Guidance and Control Panel Symposium, ENSAE, Toulouse, France, 17-20 May 1983.

6) D. L. Sebring and Captain J. T. Young, "Redundancy Management of Skewed and Dispersed Inertial Sensors," AIAA 4th Digital Avionics Systems Conference, 17-19 November 1981.

84-2624

A HIGHLY MONITORED DIGITAL FLIGHT CONTROL SYSTEM FOR THE AV-8B HARRIER II

Gary G. Gaston
Subsystem Manager, Flight Controls, Navigation and Flight Aids
McDonnell Douglas Corporation
St. Louis, Missouri

Lynn E. Costlow*
Engineering Department Head, Tactical Systems
Sperry Corporation, Flight Systems
Albuquerque, New Mexico

James R. Perry
Engineering Section Head, AV-8B Flight Controls
Sperry Corporation, Flight Systems
Albuquerque, New Mexico

Abstract

The new AV-8B Harrier II V/STOL Strike aircraft for the U.S. Marines features an advanced digital flight control system called the Stability Augmentation and Attitude Hold System (SAAHS). This system is a limited authority Stability Augmentation System (SAS) that also includes classical autopilot functions for pilot relief. The system is single channel with mechanical backup and features extensive self-monitoring for fail passive operation.

Self-test capabilities are provided for maintainability as well as safety (fail passiveness). SAAHS monitoring is implemented in three primary categories: hardware, software monitor of hardware, and software monitor of performance. Total system health is determined in a comprehensive preflight Built-In Test (BIT) and system performance during flight is continuously monitored by the In-Flight Monitor (IFM).

The SAAHS self-test philosophy establishes a validated record upon which the rest of the testing sequences are based. The philosophy meets the SAAHS requirements for detection of 98 percent of all failures and isolation of 99 percent of all detected failures to a faulty weapons replaceable assembly (WRA) in the ground BIT mode. Furthermore, it meets the requirement of detection of 75 percent of all failures and isolation of 99 percent of all detected failures to a faulty WRA in the IFM mode.

Introduction

SAAHS represents the first digital flight control application on the Harrier V/STOL series of aircraft. It fully utilizes the self-monitoring capability of a digital processor based system. This paper presents an overview of Harrier control systems, the SAAHS architecture and functional aspects, description of SAAHS monitoring techniques and results of failure transient testing.

Historical Background on Harrier Control Systems

AV-8A Harrier

The U.S. Marines AV-8A Harrier, procured in the early 1970's, has a classical mechanical control system for aerodynamic flight. A Reaction Control System (RCS), which provides control during jetborne flight (hover), is integrated with this mechanical control system. Both systems provide control during semijetborne flight. An "autostab" system provides stability augmentation below 250 knots and is implemented with analog electronics technology. The hydraulic aileron and stabilator power control cylinders have series servos which provide autostab inputs. Autostab is not provided in the forward pitch RCS nozzle. The rudder is mechanically controlled while yaw reaction control is provided by the sum of pilot input and a yaw series servo input. This series servo provides yaw stability augmentation.

YAV-8B Prototype Harrier

An AV-8A was modified for prototype demonstration of the AV-8B in the mid-1970's. The resultant YAV-8B flight control system resembles the AV-8A except for two major changes: new aileron actuators were provided for the larger wing, and a series servo actuator was developed for the forward pitch reaction control nozzle. This series servo actuator, driven by a modified pitch/roll autostab computer, provided SAS inputs for the forward RCS nozzle.

AV-8B Harrier

The original approach for AV-8B automatic flight control was designed to provide pitch and roll attitude hold along with stability augmentation for 250 knots and below. This concept provided the SAAHS name for the AV-8B flight control system. The design approach was to use the YAV-8B mechanical control system and actuators to the greatest extent practicable.

*Member AIAA

The first major change to the flight control system was to add a rudder power control cylinder with a series servo for SAAHS inputs. The original yaw series servo was deleted. The new actuator features a manual reversion capability upon loss of hydraulic power. Once this actuator was added, the original SAAHS concept was expanded for use throughout the flight envelope. In addition, heading and altitude hold modes were added.

Another important change increased the stabilator power control cylinder series servo authority from ±1.5 degrees to ± 3 degrees of stabilator rotation. This added authority was required to control large trim changes which occur when the flaps and/or nozzles are lowered. With this change came the first stringent requirement for fast, efficient monitoring in the SAAHS computer.

Two-speed pitch and roll trim actuators are provided. For manual trim, the pitch and roll speeds are 2.45 and 0.89 surface degrees per second. In the automatic trim mode, the pitch and roll speeds are 0.3 and 0.25 degrees per second. These speeds were chosen to handle slow trim changes for situations such as the center of gravity changes due to fuel consumption or transfer. Large trim changes must be accomplished manually.

Schematics of the AV-8B longitudinal, lateral, and directional control systems are shown in Figures 1, 2, and 3.

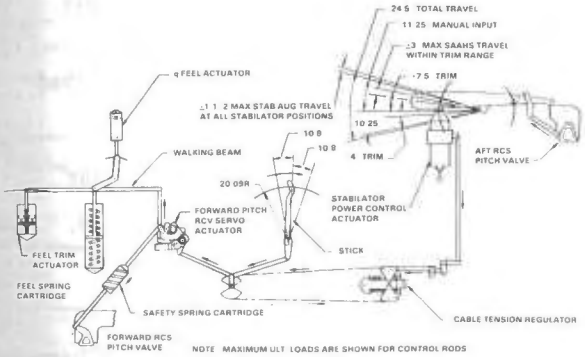


Fig. 1 AV-8B Longitudinal Control System.

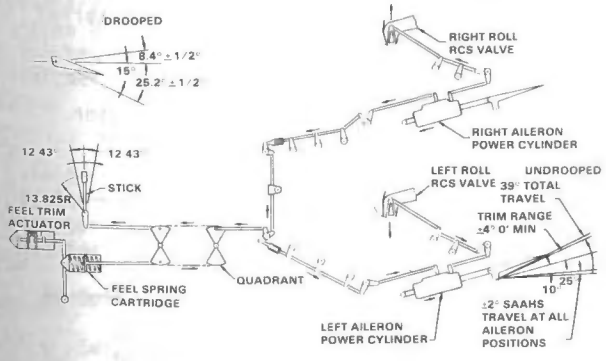


Fig. 2 AV-8B Lateral Control System.

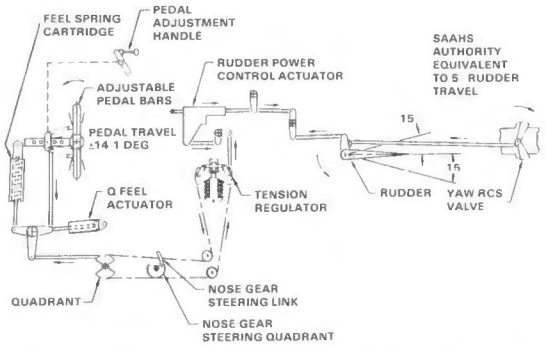


Fig. 3 AV-8B Directional Control System.

SAAHS Description

Architecture

SAAHS is implemented in a single channel architecture and uses automatic reversion to the mechanical control system upon failure. A block diagram of the SAAHS in Figure 4 depicts the interface with other AV-8B avionics systems, as well as sensors and actuators in the aircraft.

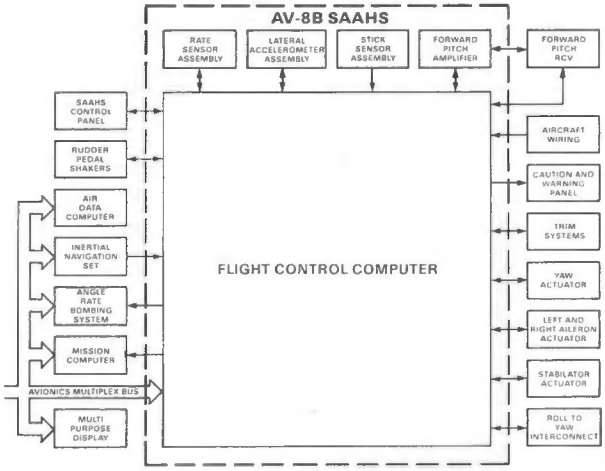


Fig. 4 AV-8B SAAHS Block Diagram.

Modes

SAAHS operational modes for SAS functions include three-axis rate damping in both vertical and cruise modes and the transition region between these two flight regimes. In addition, SAS functions include rudder/aileron turn coordination, independent axis engagement, and monitoring. Automatic flight control modes include functions for attitude hold, heading hold, altitude hold, automatic trim, control stick steering, and airspeed scheduled limits. SAAHS control of the AV-8B in all three axes is accomplished through normal aerodynamic surfaces in cruise modes, and by engine bleed air-powered reaction jets in vertical modes. A rudder pedal shaker function warns the pilot of dangerous sideslip in approach and hover.

Hardware Complement

The heart of the SAAHS system is the digital Flight Control Computer (FCC) which features a

2901 bit slice, 16-bit processor capable of 470 thousand-operations-per-second (kops) with ultraviolet erasable programmable read only memory (UVEPROM). The FCC program consumes 16k words of memory of which 6k words are control law code and 10k words are BIT. Remaining SAAHS units consist of a three-axis Rate Sensor Assembly (RSA), Lateral Accelerometer Assembly (LAA), Stick Sensor Assembly (SSA), and Forward Pitch Amplifier (FPA) as shown in Figure 5.

Due to the weight-sensitive V/STOL application, total SAAHS set weight is a very light 27 pounds, with the FCC being only 13.5 pounds of this equipment.

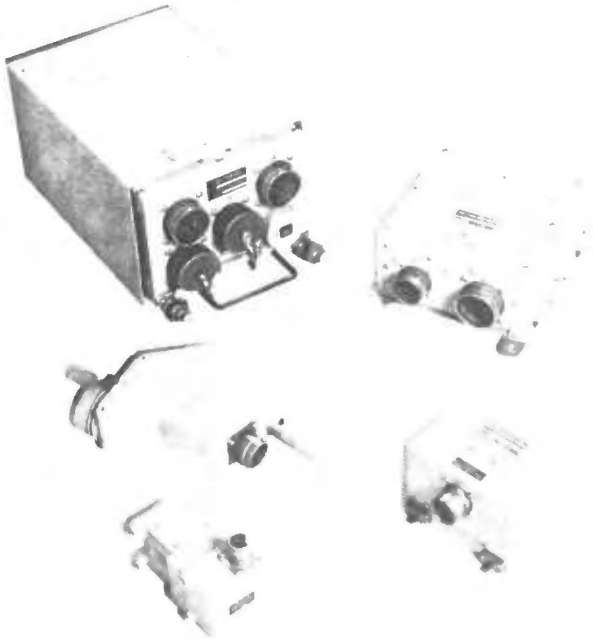


Fig. 5 SAAHS WRA Set.

Monitoring and Maintainability Requirements

Requirements for failure detection, indication, and isolation capability are divided into two categories: IFM during flight and Initiated BIT on the ground.

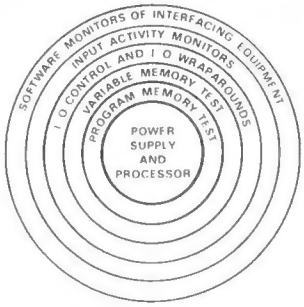
IFM. IFM will detect more than 75 percent of all failures. One hundred percent of the detected failures will be indicated by outputting digital status words. At least 99 percent of the detected failures will be fault isolated to a faulty WRA by outputting digital status word. Failures detected by IFM will result in the disengagement of the corresponding axis(es) computation. Operational capability unaffected by the detected failure will be retained.

Initiated BIT. Initiated BIT will detect at least 98 percent of all equipment failures, including failures occurring in modes not manually selected in multimode equipment. One hundred percent of the detected failures are indicated by providing corresponding digital status words. At least 99 percent of the detected failures are fault isolated to the faulty WRA by outputting digital status word. Maintenance actions are triggered by failure messages stored in the nonvolatile semiconductor memory during either Initiated BIT or IFM.

Summary of Self-Test Techniques

Philosophy

SAAHS uses a large array of self-test techniques. These techniques support a self-test philosophy which establishes a validated core of functions upon which the rest of the testing sequences are built as indicated in Figure 6.



THE SELF TEST PHILOSOPHY ESTABLISHES A VALIDATED CORE UPON WHICH THE REST OF THE TESTING SEQUENCES ARE BUILT

Fig. 6 Self-Test Philosophy.

Self-Test Modes

SAAHS self-testing is partitioned into six modes: Short Power Up (SPU), Long Power Up (LPU), Automatic BIT (AUTO), Maintenance BIT (MAINT), Acceptance Test Procedure Test (TEST), and In-Flight Monitor (IFM). These modes are used for various functions described in Table 1.

Self-Test Mode	Functional Coverage
SPU	Power supply, processor, ROM, RAM. Performed only in-flight, upon power recovery.
LPU	All SPU functions plus analog and discrete input/output. Performed only on ground upon power application.
AUTO	Preflight check of SAAHS without hydraulics. Checks SAAHS plus servos with hydraulics.
MAINT	Utilizes Digital Display Indicator (DDI) for failure indication for maintenance action.
TEST	Subset of AUTO used in factory acceptance test of SAAHS FCC.
IFM	Flight control system self-test executed continuously in-flight.

Table 1 SAAHS Self-Test Mode Summary

Category	Approach	Implementation S/W	Implementation H/W
Power Supply	Aircraft input power continuously monitored.	-	x
	Critical +5 volts continuously monitored.		x
	Other secondary voltages periodically monitored.	x	-
Processor	Test control store using subset of instructions.	x	-
	Test register and logic using key instructions.	X	-
	Heartbeat monitors test compute cycle frame time.	-	x
Memory			
-Program memory (ROM)	Compute checksum and compare with stored value.	x	-
-Read/Write memory (RAM)	Test cells with "marching bits" pattern and data bus with "walking bits" pattern.	x	-
-Maintenance nonvolatile read/write (MNOS)	Store test patterns in last two locations and read back. Store failure messages in both RAM and MNOS. Checksum of MNOS stored in RAM and MNOS.	x	-
I/O Control	Serial links between I/O Control and I/O boards for fault isolation between boards.	-	x
	Hamming code used in BIT address control circuit.	x	x
	Alternating bit pattern inverted and wrapped around.	x	x
	Detects any failure combination of stuck high/low or shorted data bits.	-	-
	Detects one or more address bus bit failures	-	-
	Detects one or two-bit failures of BIT PROM address.	-	-
I/O Interface			
-D/A - A/D	Dedicated BIT D/A used to test A/D for linearity/accuracy.	x	x
	All other D/A tested by wrap-around to A/D at five different voltages.	x	x
-Analog Inputs	Use "forced input" test via analog switch-in of BIT D/A.	x	x
	Test voltage at five different voltages.	x	-
I/O Interface (cont)			
-Discrete Inputs	Use "forced input" test via BIT voltage injection at input resistor divider.	x	x
	Use "walking ones" pattern.	x	-
-Multiplex Bus	Bus controller in Mission Computer deposits terminal test word in FCC RAM.	-	-
	FCC processor moves test word to another location.	x	-
	Mission Computer retrieves and checks test word from new location.	-	-
Servo Loops	Servo position feedback compared with command.	x	x
	"Center tap" monitors on servo position feedback LVDT/RVDT.	x	x
	"Offset" monitors on trim actuator position feedback potentiometers.	x	x
	Series servo software models compared with performance of real servo loops.	x	-
	Data reasonableness monitors check servo loop rate performance based on aircraft performance capabilities.	x	-
	Servo position limit monitors detect servo position errors scheduled with dynamic pressure (tighter limits at higher pressures).	x	-
	Techniques above detect all failures in D/A, servo amp, actuator, actuator feedback, and feedback amp.	-	-
Sensors			
-Lateral Accelerometer	Inject precise BIT current into torque coil.	x	x
	Monitor output voltage by FCC to detect failures in pendulum mass movement, pendulum position detector, and output electronics.	x	-
-Rate Sensor	Inject calibrated current into gyro torque coil.	x	x
	Monitor output voltage by FCC to check gimbal mechanism and electronics.	x	-
	Spin motor rotation detector (SMRD) signal verified gyro motor synchronous speed.	x	x

Table 2 Summary of Self-Test Techniques

Summary of Techniques

Self-test functions are divided into seven primary categories: Power Supply, Processor, Memories, Input/Output Control, Input/Output Interface, Servo Loops, and Sensors. A summary of techniques used is presented in Table 2, along with an indication of whether the technique is hardware-based, software-based, or both.

In-Flight Monitor Approach

The purpose of IFM is to detect flight control system failure in flight and identify the appropriate corrective action. The following monitors are used:

- Spin Motor Rotation Detectors (SMRD)
- Software Series Servo Models
- Position Limit Monitors
- Position Feedback Linear Variable Differential Transformer (LVDT) or Rotary Variable Differential Transformer (RVDT) Center Tap Monitors
- Data Reasonableness Monitors
- Offset Monitors

Spin Motor Rotation Detector (SMRD)

Magnets in each rate gyroscope spin motor wheel and a static coil comprise a magneto. When the wheel rotates, pulses are generated in the magneto as shown in Figure 7. The pulses are demodulated and a proportional dc voltage is output. As long as the wheel spins at the proper speed the dc output is the proper value. If the wheel quits turning, the dc voltage drops and the failure is detected by a dc level detector.

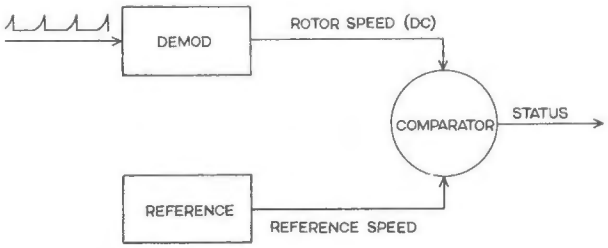


Fig. 7 Spin Motor Rotation Detector.

Software Series Servo Models

The series servo position is compared with a software actuator model (Figure 8). The differences are monitored in an error comparator. When a tolerance level is exceeded, the series servo is declared failed and is disengaged. Comparison monitors are most effective where delays in excess of 100 milliseconds are acceptable. In the AV-8B, the software series servo models are most effective at low dynamic pressures.

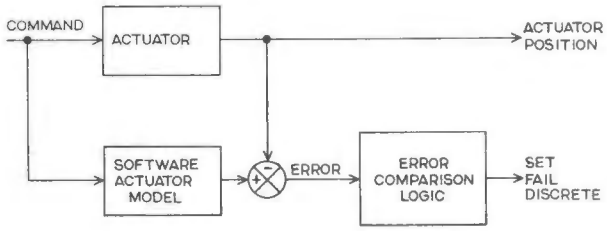


Fig. 8 Actuator Model Comparison.

Position Limit Monitors

Position limit monitors become increasingly effective with increasing dynamic pressure ($\bar{Q}$) (Figure 9). The absolute value of the servo position is compared with 120 percent of the position limit. If the threshold is exceeded, the FCC declares the actuator failed. This monitor works within 10 milliseconds of the failure.

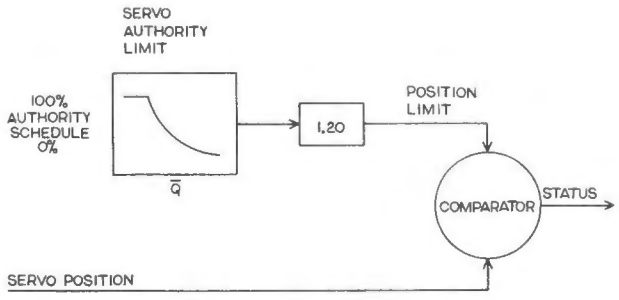


Fig. 9 Position Limit Monitor.

Position Feedback LVDT and RVDT Center Tap Monitor

Center taps are required to monitor three types of devices (Figure 10). In a differential transformer, the secondary voltages V_1 and V_2 are subtracted for the signal. When the center tap is grounded, the values of V_1 and V_2 can be added. In normal operation, the sum of V_1 and V_2 is a constant value. Any broken wire or loss of excitation causes the sum of V_1 and V_2 to decrease in voltage. This decrease is interpreted by the flight control computer as a failure. This monitor can detect failures within 10 milliseconds.

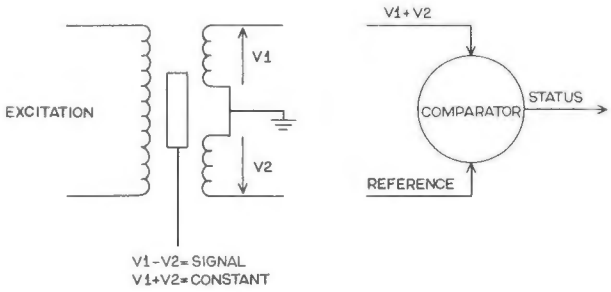


Fig. 10 Center Tap Monitor.

Data Reasonableness Monitors

These monitors allow a relatively long time to detect failures while preventing failed signals from commanding control surface movements. They are also useful in preventing transients from moving control surfaces. In designing a data reasonableness monitor, the maximum possible rate of change of a parameter must be determined. When a parameter's maximum rate is exceeded, the software holds the past value for control law computation and starts a counter. If the exceedance is a transient, the next input will be accepted providing it is within reason. The sensor is declared invalid if it continues to exceed the maximum possible rate of change.

The following is an example of a data reasonableness monitor for a roll attitude synchro signal. 300 degrees per second roll rate exceeds the roll capability of the AV-8B in controlled flight. Thus, the maximum allowable change in a computer with a 50 Hz update is: 300 degrees per second x 20 milliseconds = 6 degrees. The present value is subtracted from the past value:

$$|\phi_{past} - \phi_{present}| = \Delta\phi$$

If $\Delta\phi$ is less than 6 degrees, it is used. If $\Delta\phi$ is greater than 6 degrees, then the past value (ϕ_{past}) is used. If $\Delta\phi$ continues to be larger than 6 degrees, a failure is declared.

Offset Monitors

The concept does not allow hardover failures or failures to zero to go undetected (Figure 11). The parameter range is defined as being from -1 to +1. The allowable sensor output is between 1 and 3. If the output is less than 1 or more than 3, it is declared failed.

Figures 12 and 13 show that this concept can be expanded beyond digital computer use.

The AV-8B roll trim actuator feedback potentiometer is powered by a 16 volt dc source. End resistors in the feedback pot prevent the output from exceeding 12 volts or going below 4 volts. The FCC uses this feedback voltage for trim position. If the value exceeds 12 volts dc or is less than 4 volts dc, the signal is considered failed. The indicator is a 0 to 16 volt movement. The face is masked so that the needle cannot be seen when the voltage is greater than 12 vdc or less than 4 volts dc.

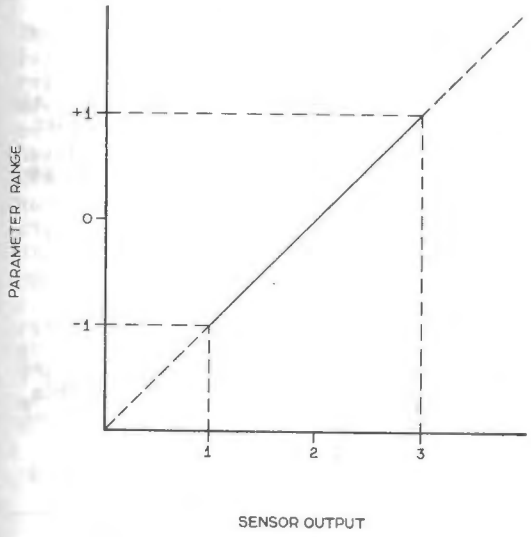


Fig. 11 Offset Monitor Concept.

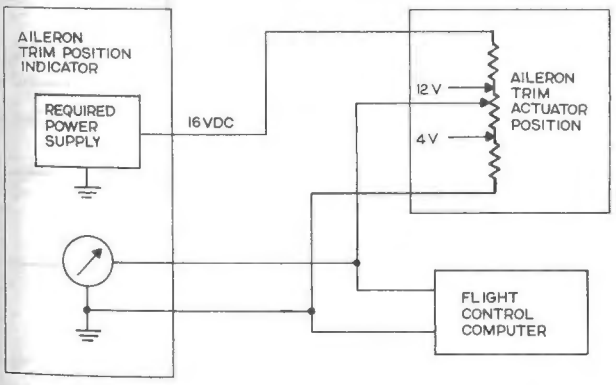


Fig. 12 Offset Monitor Example.

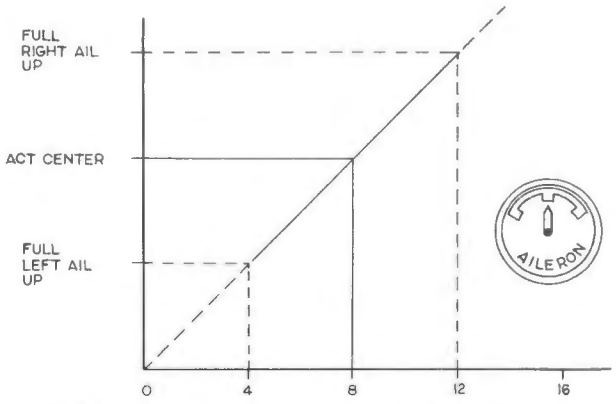


Fig. 13 Aileron Trim Offset Monitor.

Iron Bird Testing Failure Transient Results

More than 1,000 test runs were completed on the "Iron Bird" simulator to test both control laws and IFM. Selected examples are included to demonstrate the effectiveness of IFM. All failures were inserted with the Iron Bird "flying" at 0.85 Mach at sea level. This flight regime was chosen because hardover failures will have the most profound effect at these conditions. In each case, the Iron Bird was set up flying straight and level. The Iron Bird was unpiloted, and tended to drift after failures were inserted.

Stabilator Hardover LVDT Failure

The stabilator series servo feedback was opened at t = 1.2 seconds causing a nose down stabilator series servo movement. At t = 1.25 seconds, the software center tap monitor caused the pitch axis to disengage. Figure 14 shows the resulting pitch rate caused by the stabilator surface position transient of 0.35 degree. Figure 15 shows the associated normal acceleration. Angle of attack and pitch attitude change by less than 0.1 degree during the transient. All transients caused by this failure are negligible.

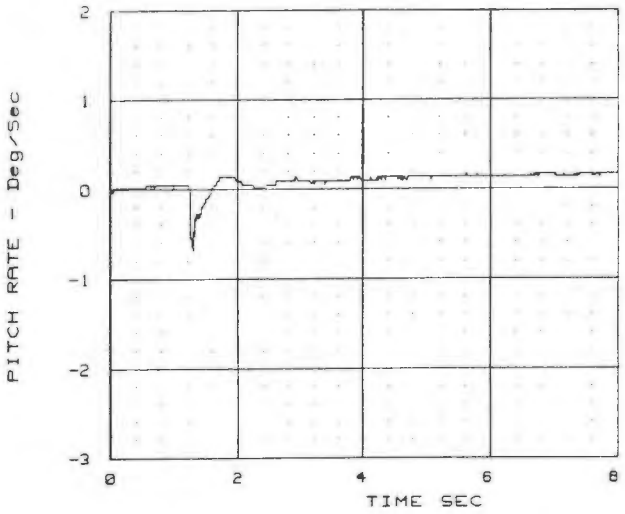


Fig. 14 Stabilator LVDT Failure Transient.

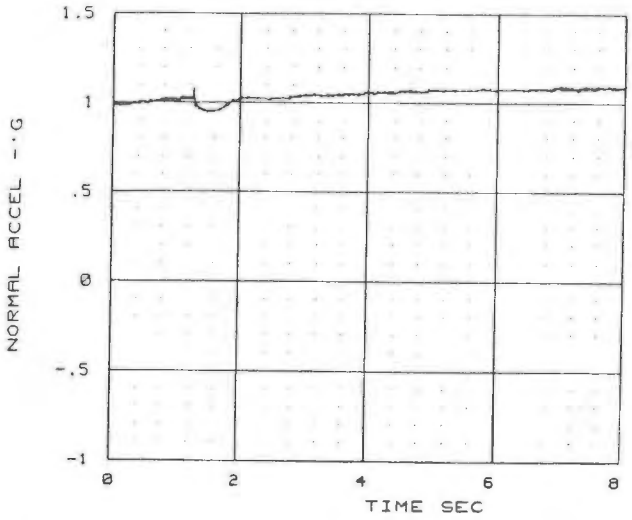


Fig. 15 Stabilator LVDT Failure Transient.

Stabilator Hardover Servo Amplifier Failure

This failure demonstrates a much larger effect than the center tap monitor detected failure, because the servo actuator must move to cause a failure detection. When a center tap monitor senses the monitored input voltage changes, it detects the failure prior to significant actuator movement. The example failure was inserted at t = 1.2 seconds. The software stabilator series servo limit monitor detected a 20 percent above command signal limit, and disconnected the pitch axis at t = 1.25 seconds. The stabilator series servo moved +0.45 inch causing a stabilator surface movement of -1.4 degree. Figure 16 shows the resulting pitch rate. Figure 17 shows the associated normal acceleration. Angle of attack changed by 1.2 degrees and pitch attitude by 1.6 degrees. The failure's effect lasts for less than 1 second, and then the aircraft returns to trim.

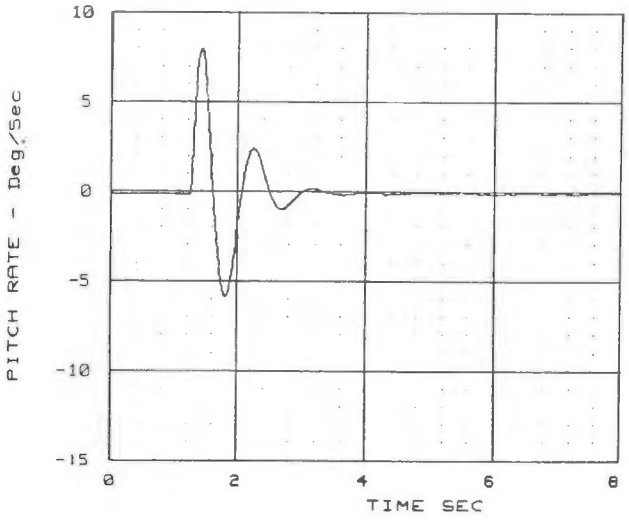


Fig. 16 Stabilator Servo Amplifier Hardover Transient.

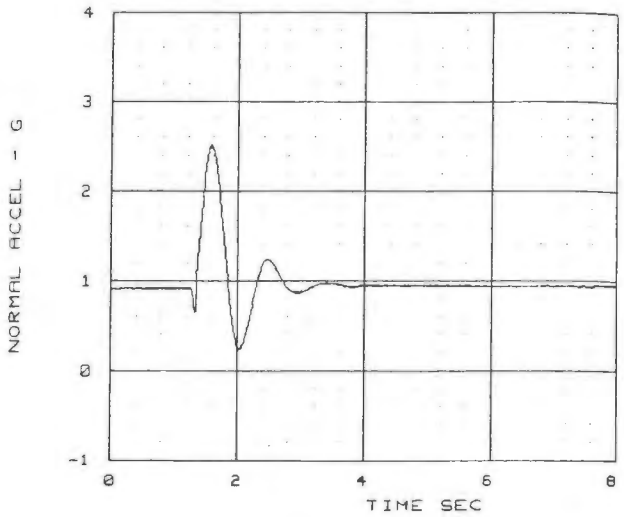


Fig. 17 Stabilator Servo Amplifier Hardover Transient.

Rudder Hardover LVDT Failure

The rudder series servo feedback was opened at t = 1.10 seconds, causing a nose right rudder series servo movement. At t = 1.15 seconds, the software center tap monitor caused the yaw axis to disengage. Figure 18 shows the resulting yaw rate caused by the rudder surface position transient of -0.8 degree. Figure 19 shows the associated lateral acceleration. The failure is quickly detected, but there is a tendency for the aircraft to oscillate for approximately 2 seconds. Sideslip changes 0.3 degree during the transient. There is also a coupling into the roll axis causing a change in roll rate of 0.7 degree per second.

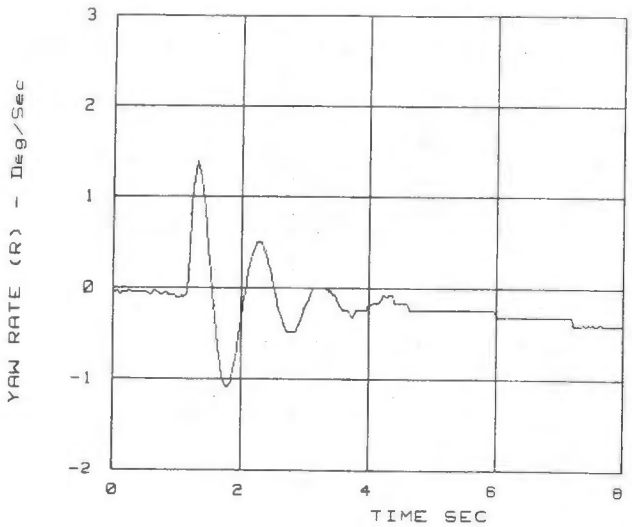


Fig. 18 Rudder LVDT Failure Transient.

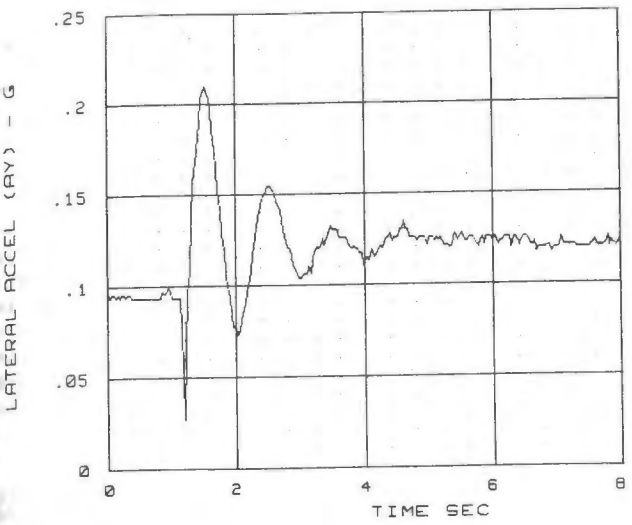


Fig. 19 Rudder LVDT Failure Transient.

Rudder Hardover Servo Amplifier Failure

The rudder series servo amplifier hardover failure was inserted at t = 1.05 seconds. The resulting nose right rudder series servo movement caused the software rudder series servo limit monitor to detect a 20 percent above command signal limit, and to disconnect the yaw axis at t = 1.10 seconds. The rudder surface position moved -3.5 degrees. Figure 20 shows the resulting yaw rate. Figure 21 shows the associated lateral acceleration. As a result sideslip changed by 1.2 degrees. Roll axis coupling caused roll rate to change by -3.0 degrees per second. Although a peak lateral acceleration of about 1.5 G for 0.4 seconds may be momentarily uncomfortable for the pilot, it is not a safety problem.

Fig. 20 Rudder Servo Amp Hardover Transient.

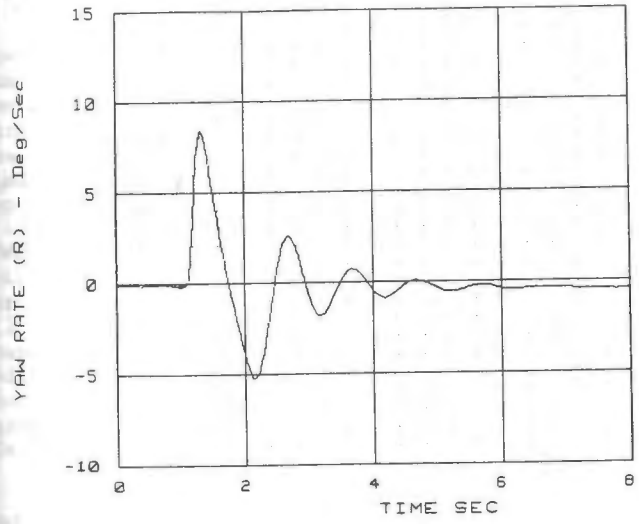


Fig. 20 Rudder Servo Amp Hardover Transient.

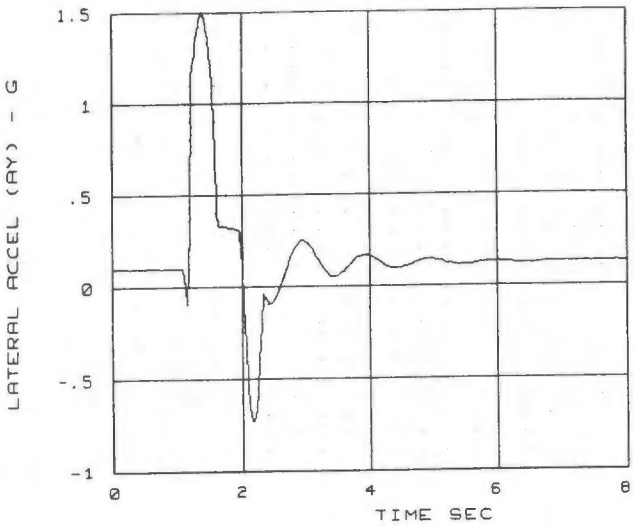


Fig. 21 Rudder Servo Amp Hardover Transient.

Conclusion

Retention of the basic mechanical controls from the AV-8A, coupled with the strict weight requirements for the AV-8B flight control system, dictated that the design be a single channel architecture. The additional weight required for multiple processors, I/O, and sensors was unacceptable. The challenge was to design a single channel system with extensive self monitoring that would safely revert to mechanical control in the event of a failure. This was accomplished by carefully integrating the aircraft sensors and actuators with the Flight Control Computer to create a total system design with the required failure detection and isolation capabilities. By using the Flight Control Computer's computing power, and placing the majority of the monitoring workload in software, the size, weight, and recurring cost of the system has been minimized.

System reliability and safety have been demonstrated during 3 years of flight and ground testing, and by production aircraft operation. Hands-off-the-stick vertical landings have been made during flight test, indicating the test pilot's confidence in the system's safety.

84-2625

MULTIVARIABLE CONTROL SYSTEM DESIGN TECHNIQUES: AN APPLICATION TO A SHORT TAKEOFF AND LANDING AIRCRAFT

R. J. Landy*, D. B. Kim**
McDonnell Aircraft Company
McDonnell Douglas Corporation
St. Louis, MO

ABSTRACT

This paper presents a technique for designing multi-input multi-output control systems that is superior to iterative applications of conventional single loop control system design techniques and leads to improvements in both accuracy and design cost.

This technique, termed Singular Value Synthesis (SVS) has evolved from the results of Doyle and Stein (Reference 1) and from subsequent work by their colleagues (References 2, 3, 4). It can be used both to synthesize a control system design and to analyze its performance and stability.

This paper presents the steps of this design and analysis procedure and shows an example of its application to the design of an integrated flight propulsion control system for a representative Short Takeoff and Landing (STOL) aircraft. In particular, the paper discusses the application of this technique to digital control system design. Two digital design approaches, Control Law Transformation (CLT) and W'-plane design are explored and their capabilities evaluated.

INTRODUCTION

The complex integration requirements of current and future weapon systems necessitate the development of techniques for the optimal design of multi-input, multi-output control systems. Conventional analytical methods involve an iterative "one control loop at a time" approach which is costly in time and manpower. These conventional techniques can also be misleading in predicting system robustness, that is the sensitivity in performance and stability to changing system parameters or modeling errors.

Although the theoretical mathematics exist for multivariable optimization, these techniques have had very limited application in the aircraft industry. The principal reason has been the difficulty in defining a relationship between the time domain performance criteria used in multivariable optimization and the frequency domain requirements imposed by military specification requirements. However, recent control publications (Reference 2) have described techniques in which the two can be linked.

This paper presents a practical control system design procedure based on these recently developed multivariable optimization techniques which results in improved integrated flight control system designs. We have termed this technique Singular Value Synthesis (SVS). Using SVS, an integrated flight/propulsion control system was designed for the landing mode for a three-surface short takeoff and landing vehicle. This design, accomplished first in the analog (s-plane) domain, met the design goals. A digital control system was then designed for the same aircraft using two different approaches.

SINGULAR VALUE SYNTHESIS TECHNIQUE

The objective is to generate control laws for multivariable systems which provide good closed loop performance and stability properties. In the SVS technique, this is accomplished by solving a sequence of optimization problems to match a designer-specified open loop frequency response matrix, called $T_D(s)$. This response matrix can be specified at either the input or the output of the plant $G(s)$. We term the associated design problems as input break and output break problems, respectively.

We must first quantify the concept of the size of a matrix in order to define the closeness of two matrices and hence to determine the quality of the SVS design, i.e., how close the SVS-designed open loop transfer function matrix is to the desired T_D matrix. Singular values are selected to represent the size of a matrix. Singular values of a complex-valued matrix A, $\sigma_i(A)$, are defined as $+\sqrt{\lambda_i(A^*A)}$, where λ_i denotes an eigenvalue and A^* is the transposed conjugate of A.

The system equations are written in state space format as follows:

$$\dot{x} = Ax + Bu + \zeta$$

$$y = Cx + \eta$$

The system noise, ζ, and measurement noise, η, are assumed to be Gaussian with zero mean and with constant covariances. Many noise models, for example the Dryden wind gust representation, can be incorporated in this form by including additional dynamics in the system model.

We can describe an input-output relation for this system as follows:

$$y(s) = G(s)u(s) + d(s) + \eta(s)$$

where:

$$G(s) = C(sI - A)^{-1}B$$

$$d(s) = C(sI - A)^{-1}\zeta(s).$$

Typical mission performance objectives are to make the output, denoted y_r, track a given commanded output, denoted y_c, with less than a specified error level over the control frequency range (Figure 1). This is mathematically expressed as

*Section Chief, Guidance and Control Mechanics
**Senior Engineer, Guidance and Control Mechanics
Now with Lear Siegler Astronics, Santa Monica, CA.

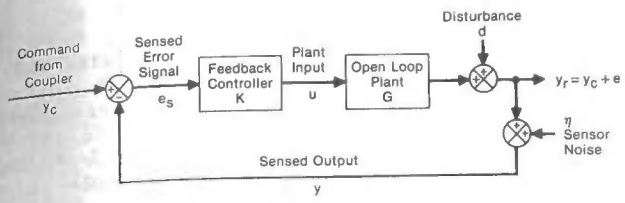


Fig. 1 Command Follower Feedback Loop

$$||y_r(jw) - y_c(jw)||_2 < \varepsilon$$

where the norm holds at each frequency over the control frequency range. ($||x||_2 = x_1^2 + x_2^2 + \ldots + x_n^2$ is the Euclidean vector norm.)

The tracking error response to commands, disturbances, and sensor noise is given by

$$e(s) = y_r(s) - y_c(s)$$

$$= (I + G(s)K(s))^{-1}\,(d(s) - y_c(s)) - (I+G(s)K(s))^{-1}\,G(s)K(s)\,\eta(s)$$

where GK denotes the loop transfer matrix with the feedback loop broken at the output y. This last equation relates the system performance to each error source.

Using the minimum and maximum singular values, denoted $\underline{\sigma}(.)$ and $\bar{\sigma}(.)$, respectively, we can define the approximate loop shape required for good command following, disturbance rejection, response to sensor noise, and robustness. The minimum and maximum singular values of a matrix can be interpreted as the minimum and maximum gains that the matrix can produce. Therefore, from the error signal equation, we can draw the following conclusions about the properties required in the feedback loop.

Loop Property 1: The size of $(I + GK)^{-1}$ should be small, or equivalently $(I + GK)$ large, when $||d - y_c||_2$ is large compared to a maximum allowable $||e||_2$.

Loop Property 2: The size of $(I + GK)^{-1}GK$ should be small when $||\eta||_2$ is large compared to a maximum allowable $||e||_2$.

The open loop shaping requirements are illustrated in Figure 2. The familiar classical open loop requirements are apparent. We desire (1) high loop gain at low frequency for tight tracking of the command and rejection of plant disturbances, (2) a 1/s type roll-off through crossover to guarantee good stability margins, and (3) a small loop gain at high frequencies to suppress sensor noise and effects of neglected high frequency dynamics in the model used for control law design. The SVS technique seeks a good closed loop design by proper choice of the desired open loop (GK) frequency response.

There are two stability properties useful in multivariable control system analysis. Defining GK_{true} as the actual plant and compensator, then the multiplicative perturbation on the plant output ΔG is defined as:

$$GK_{true} = (\Delta G)GK$$

For each of the two stability results to be stated, the following conditions are assumed: 1) If the nominal open loop transfer matrix G(s)K(s) has a pole on the imaginary axis, then so does the perturbed open loop system, and 2) The nominal and perturbed open loop systems have the same number of right half plane poles.

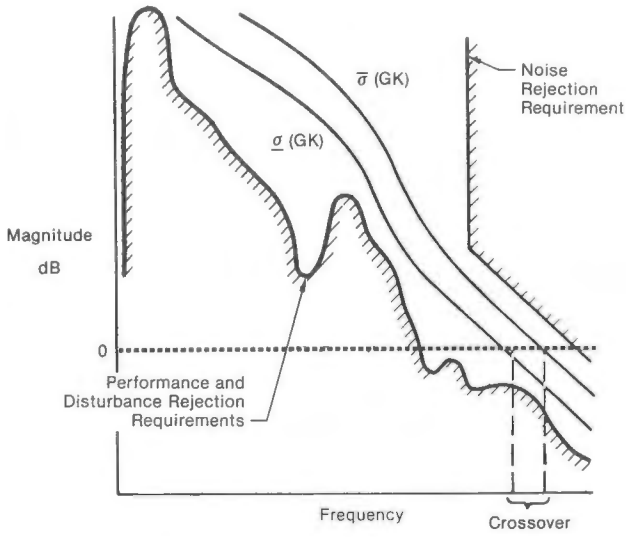


Fig. 2 Multiloop Open Loop Characteristics

The first stability property states that the closed loop system is stable if the unperturbed closed loop system is stable and if:

$\bar{\sigma}(\Delta G - I) < \underline{\sigma}(I + (GK)^{-1})$ for all s on the Nyquist contour,

where $\underline{\sigma}(A)$ is the minimum singular value and is mathematically equivalent to

$$\min ||Ax||_2$$
$$||x||_2 = 1$$

and $\bar{\sigma}(A)$ is the maximum singular value and is mathematically equivalent to

$$\max ||Ax||_2 = \frac{1}{\underline{\sigma}(A^{-1})}$$
$$||x||_2 = 1$$

The second stability property (Reference 3), which requires that ΔG have no zero or negative eigenvalues, states that the closed loop system is stable if the unperturbed closed loop system is stable and if:

$\bar{\sigma}(G^{-1} - I) < \underline{\sigma}(I + GK)$ for all s on the Nyquist contour.

These stability properties are sufficient conditions for stability and were used to generate stability margins presented in this paper.

SVS DESIGN PROCEDURE

The SVS methodology can be remarkably effective in achieving the open loop characteristics required by Figure 2. The resultant command following system architecture is given in Figure 3. The SVS procedure results in defining the optimal controller (Kc) and Kalman-Bucy filter (K_f) matrices for the controller K(s) shown in Figure 3.

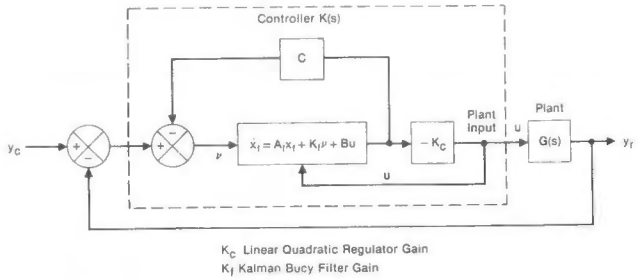


Fig. 3 Command Follower System Architecture

Four steps are used in a SVS design for an output command follower problem. A dual set of steps would be used for an input command follower problem. To regulate all system outputs independently, at least as many inputs are required.

Step 1 - The designer must specify a square matrix $T_D(s)$ of desired open loop transfer functions with left-half plane poles and provide a minimal (i.e., controllable and observable) state space realization of $T_D(s)$.

$$T_D = \begin{bmatrix} T_{D_{11}} & T_{D_{12}} & - - - & T_{D_{1n}} \\ & & & \\ T_{D_{n1}} & - - - - - & & T_{D_{nn}} \end{bmatrix}$$

$$T_D = \hat{C}\,(sI-\hat{A})^{-1}\,\hat{B}$$

The designer can draw on familiar concepts from single loop control theory to pick transfer functions which meet the open loop characteristics shown in Figure 2.

If the open loop plant G(s) has all its transmission zeros in the left half plane (i.e., minimum phase) and at least as many inputs as outputs, then the remainder of the compensation design is relatively routine. (Transmission zeros are defined as the roots of the numerator polynominal of det G.)

Step 2 - Solve the Kalman Bucy Filter (KBF) problem for the following augmented system:

$$A_a = \begin{bmatrix} A & 0 \\ & \\ 0 & \hat{A} \end{bmatrix} \qquad B_a = \begin{bmatrix} B \\ \\ 0 \end{bmatrix} \qquad C_a = [C \;\; \hat{C}]$$

$$\text{Plant Noise} = Q = \begin{bmatrix} \varepsilon I & 0 \\ & \\ 0 & \hat{B}\hat{B}^T \end{bmatrix}$$

Measurement Noise = R = I

(If A has stable roots, then $\varepsilon = 0$. If A has unstable roots, may have to be made nonzero).

The solution of the resulting Ricatti equation provides the innovation weighting matrix K_f shown in Figure 3. If G(s) is stable and $T_d(s)$ is chosen well i.e., det $(I + T_d(\omega)) > 1$ at all control frequencies then the Ricatti solution is not required and K_f can be chosen as:

$$K_f = \begin{bmatrix} 0 \\ \\ \hat{C} \end{bmatrix}$$

Step 3 - Calculate a sequence of Linear Quadratic Regulator (LQR) solutions for the following augmented system as μ is decreased:

$$A_a = \begin{bmatrix} A & 0 \\ & \\ 0 & \hat{A} \end{bmatrix} \qquad B_a = \begin{bmatrix} B \\ \\ 0 \end{bmatrix} \qquad C_a = [C \;\; \hat{C}]$$

$$\text{State Weighting } Q = \begin{bmatrix} C^TC & C^T\hat{C} \\ & \\ \hat{C}^TC & \hat{C}^T\hat{C} \end{bmatrix} \qquad \text{Control Weighting } K = \mu I_{pxp}$$

This will provide a sequence of K_c values. As μ is decreased, the singular values of the open loop system G(s)K(s) approach the singular values of the desired frequency response $T_d(s)$. The designer selects the value of μ which provides an acceptable match at reasonable K(s) "gain" levels.

Step 4 - Assess the stability using the minimum singular value of the inverse return difference matrix $I + (GK)^{-1}$ and the minimum singular value of the return difference matrix I + (GK) as outlined in the discussion on stability properties. If the stability margins are not satisfactory, repeat Step 3 until both the performance and stability tradeoffs are satisfactory.

SVS DESIGN EXAMPLE: LANDING MODE CONTROL SYSTEM DESIGN

The SVS technique was applied to the design of an integrated flight propulsion landing mode control system for a representative three-surface short takeoff and landing aircraft. Figure 4 shows the analog control system architecture used in this example.

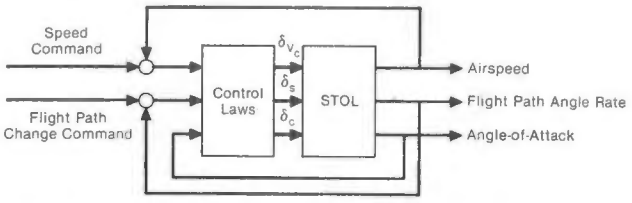


Fig. 4 STOL Landing Mode Control System Architecture

The two design objectives are (1) to control airspeed (u) with almost no change in flight path angle rate ($\dot{\gamma}$) and angle of attack (α), and (2) to control flight path angle rate with very little change in airspeed and angle of attack. These design objectives require the measurement of the three variables: airspeed, flight path angle rate, and angle of attack. Control surfaces available are the stabilator, canards, and collective rotating vanes on the nozzle to provide thrust modulation.

The design model is a linear time-invariant four state (u, α, θ, γ), model describing aircraft longitudinal and translational motion. The flight condition investigated is 115 kt airspeed, 12 degrees trim angle of attack at sea level. The unaugmented aircraft is unstable at this condition.

ANALOG DESIGN RESULTS - It is desired that the closed loop response of airspeed have a 1 radian bandwidth and the closed loop responses of flight path angle rate and angle of attack have bandwidths of 2 rad/sec. Furthermore, it was desired to have steady state errors no greater than 1%. To meet these requirements, the desired open loop frequency response matrix selected consisted of diagonal transfer function elements with crossover frequencies of 1 rad/sec for airspeed, 2 rad/sec for flight path angle rate, 2 rad/sec for angle of attack, and all with steady state gains of 100.

A controller was designed to provide a close match with this desired open loop response. The resulting closed loop frequency responses are shown in Figure 5. The frequency responses of airspeed and flight path angle rate to their respective input commands are well behaved with no resonance peaking. Figure 5 also shows frequency responses for airspeed (u) and angle of attack (α) to a flight path angle rate ($\dot{\gamma}$) command, along with frequency responses of α and γ to a u command. These results show that sufficient attenuation is achieved such that these responses are effectively decoupled from the command input.

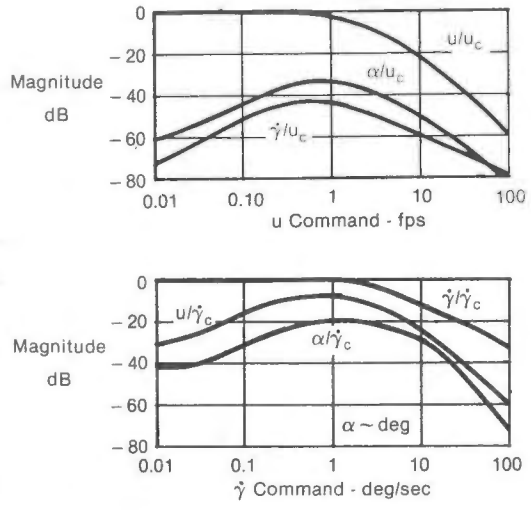


Fig. 5 Closed Loop Frequency Response
STOL Landing Mode 190 Kts at Sea Level

Time history responses for the SVS designed control system are shown in Figure 6. The airspeed response is very good, with satisfactory decoupling among flight path angle, angle of attack, and airspeed. A step command of 10 ft/sec in airspeed results in very small changes in flight path angle and angle of attack of less than 0.2° each. Note that the command inputs are removed at 6 seconds.

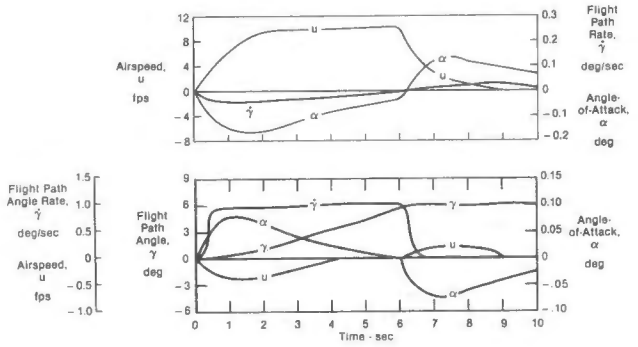


Fig. 6 Time History Response
STOL Landing Mode
190 Kts at Sea Level

Figure 6 also shows good flight path angle rate response for a step command input of 1 deg/sec. This change in flight path angle rate is accompanied by changes in u and α of less than 0.5 ft/sec and 0.1 deg. This command is also removed at 6 seconds.

The two methods described on the previous section were used to quantify multiloop stability margins. In order to present these results in the familiar phase margin-gain margin terminology, we assume a diagonal form for the multiplicative perturbation. That is, we allow simultaneous perturbations in all channels but do not allow cross-channel perturbations.

Figure 7 presents stability margins for the SVS design calculated using both methods for calculating multiloop margin and compares them with traditional margins, determined by allowing perturbations in only one channel at a time. The stability is measured at the aircraft output. The traditional single loop method always provides higher numerical margins, since it represents a more restricted (one loop at a time) perturbation. Nevertheless, the margins for the multiloop method are reasonable considering the general nature of the perturbation.

Output Break		Single Loop			Multiloop Margins					
		u	$\dot{\gamma}$	α	$\underline{\sigma}[I+GK]=0.9038$			$\bar{\sigma}[I+GK]^{-1}$ GK = 1.298		
Given Phase Perturbation	(deg)	0	0	0		0	±30		0	±30
Allowed Gain Increasing Margin	(dB)	38.41	∞	23.8		20.75	19.35		4.96	3.24
Allowed Gain Decreasing Margin	(dB)	-∞	-∞	-20.35		-5.61	-4.21		-12.79	-11.02
Given Gain Perturbation	(dB)	0	0	0	-2.81	0	6.0	-6.0	0	2.48
Allowed Phase Margin (±)	(deg)	85.99	88.2	71.36	41.06	54.02	54.02	45.33	45.33	35.12

Fig. 7 STOL Landing Mode Stability Results
Analog Design

DIGITAL DESIGN RESULTS - Figure 8 shows the digital control system architecture used in this example. To investigate the tradeoffs between the two design approaches, Control Law Transformation (CLT) and W'-plane, three sample rates were chosen: 10 Hz, 20 Hz, and 40 Hz. A zero order hold was used as the digital to analog (D/A) device. Prefilters were used at the quarter sample rate frequency to preclude aliasing, and postfilters were used at the half sample rate to smooth commands to the actuators. A computational delay of one half the sample time period was modeled using a first order Pad'e approximation and inserted into the loop for the stability and performance analysis.

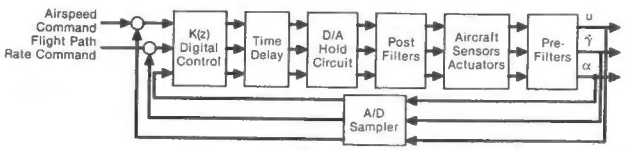


Fig. 8 Digitized STOL Configuration

Two digital control system design approaches were used to design an integrated flight propulsion control system for the STOL example. The first approach, Control Law Transformation (CLT), involves designing the control law using the SVS design procedure in the S-plane and then applying the Tustin transformation $S = \frac{2}{T}(Z-1)/(Z+1)$ to map the control law to the digital domain (z-plane), where T is the sample period.

The second approach, called W'-plane Design, involves z-transforming the analog system, the prefilters, and the postfilters from the s-plane to the z-plane, and subsequently transforming them via the bi-linear relationship $z = (1 + (T/2)W')/(1 - (T/2)W')$ to the W'-plane. The SVS design procedure outlined earlier in this paper then proceeds in the W'-plane. The advantage of W'-plane design is that the same software used for SVS design in the S-plane can also be used in the W'plane plane. The resulting control law, K(W'), is then mapped back to the Z-plane for stability and performance evaluation.

Figures 9 and 10 show frequency responses of flight path angle rate, angle of attack, and airspeed to the command inputs of flight path angle rate and airspeed. The responses of the analog system are compared with those of the two digital systems (CLT, W'-plane), for the sample rate of 40 Hz. All responses met the design objectives. Figures 11 through 14 show the same set of frequency responses for sample rates of 20 SPS and 10 SPS. Note that the CLT responses begin to deviate somewhat from the analog responses at the lower sample rates, but the W'-plane responses remain good even for sample rates as low as 10 Hz.

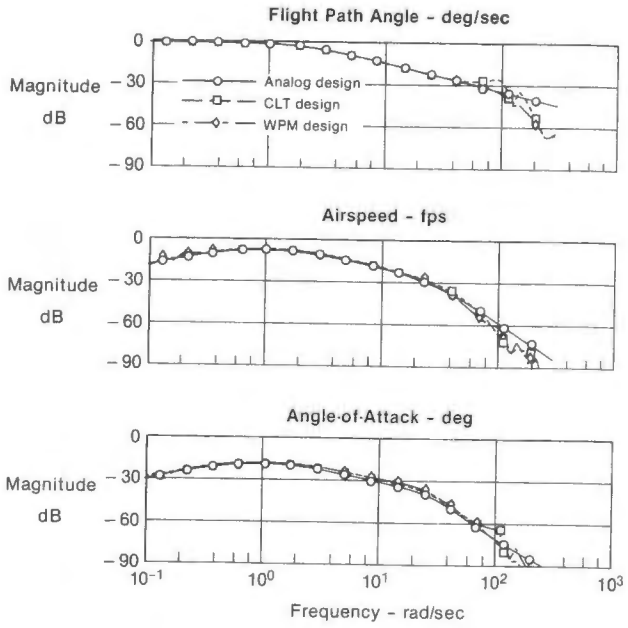


Fig. 9 Analog and Hybrid Responses
Flight Path Angle Command
40 Hz Sample Rate

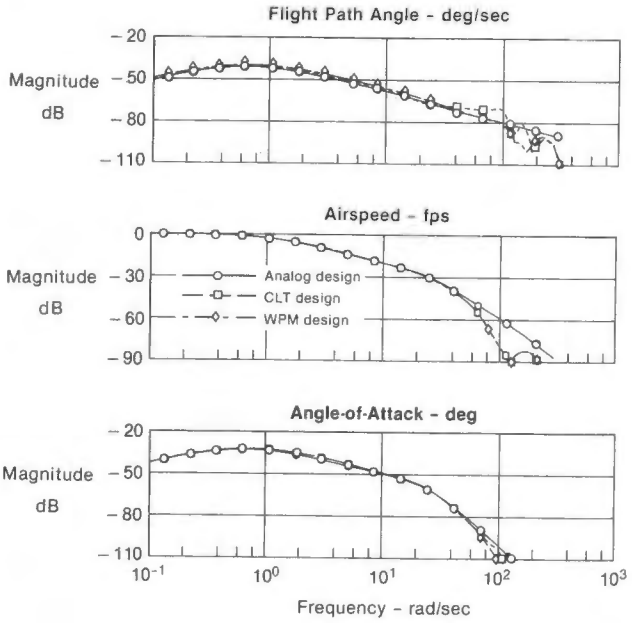


Fig. 10 Analog and Hybrid Responses
Speed Command
40 Hz Sample Rate

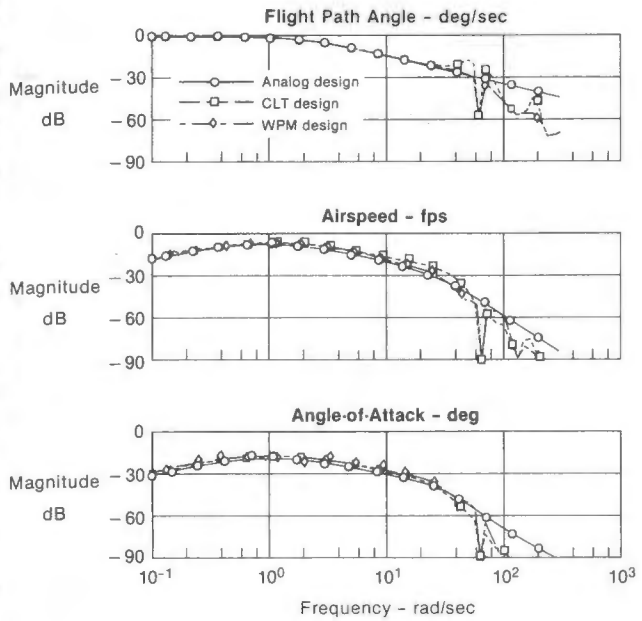


Fig. 11 Analog and Hybrid Responses
Flight Path Angle Rate Command
20 Hz Sample Rate

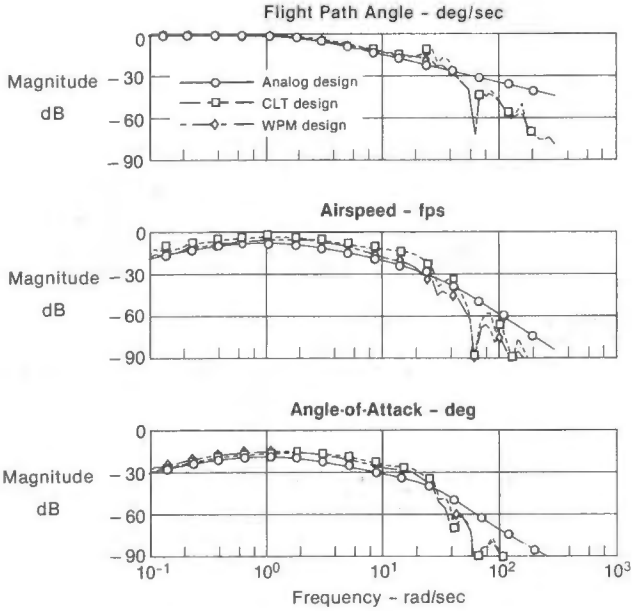


Fig. 13 Analog and Hybrid Responses
Flight Path Angle Rate Command
10 Hz Sample Rate

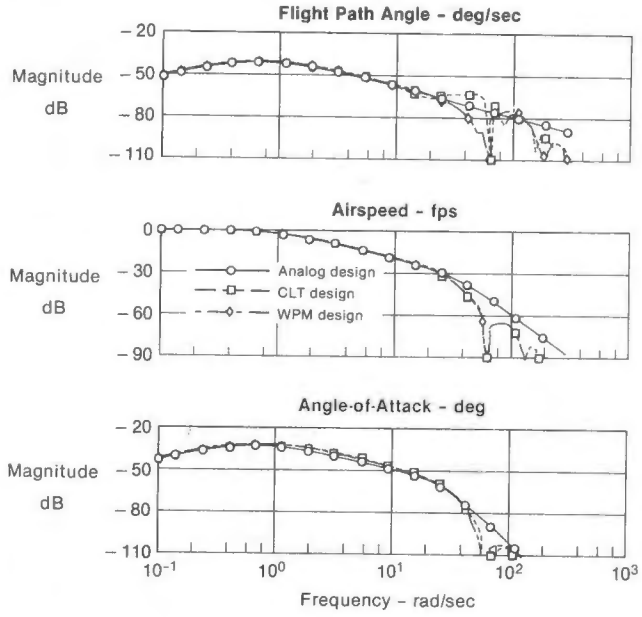


Fig. 12 Analog and Hybrid Responses
Speed Command
20 Hz Sample Rate

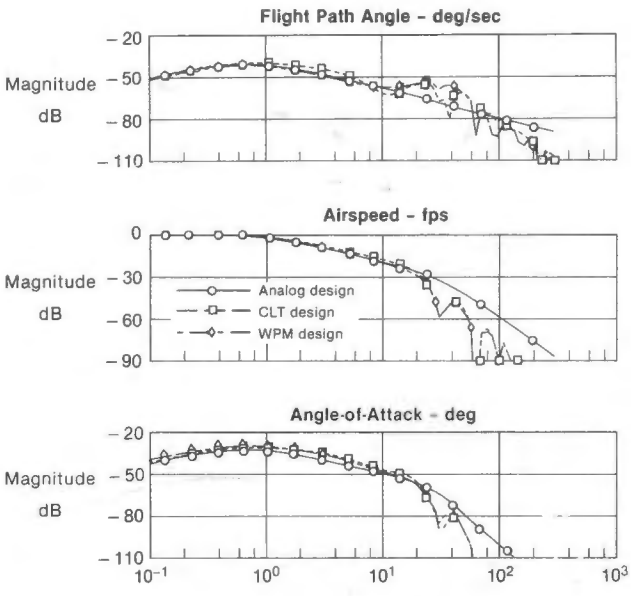


Fig. 14 Analog and Hybrid Responses
Speed Command
10 Hz Sample Rate

Figures 15 through 20 show the corresponding time history responses to step input commands for the analog and the two digital systems. At the 40 SPS sample rate, all responses met design objectives. However, as the sample rate is lowered to 20 SPS and 10 SPS, the W-prime design retains good responses while the CLT responses are degraded somewhat with respect to the analog system.

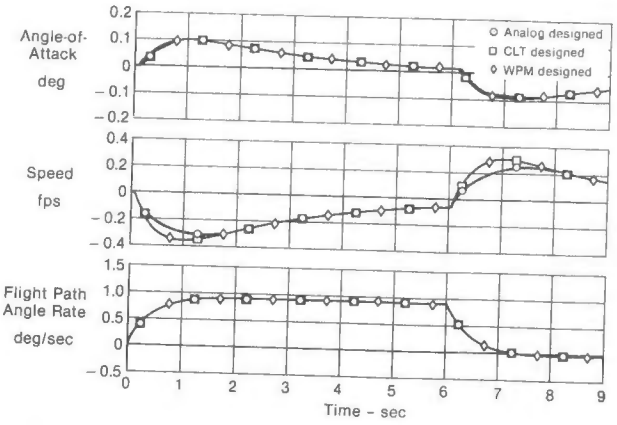


Fig. 15 Digital System Time Responses
Flight Path Rate Command
40 Hz Sample Rate

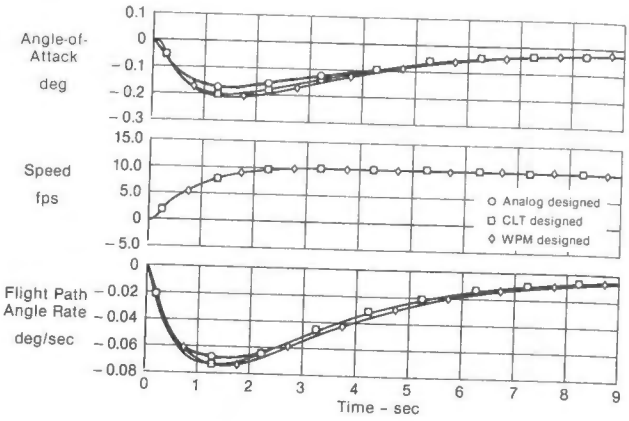


Fig. 18 Digital System Time Responses
Speed Command
20 Hz Sample Rate

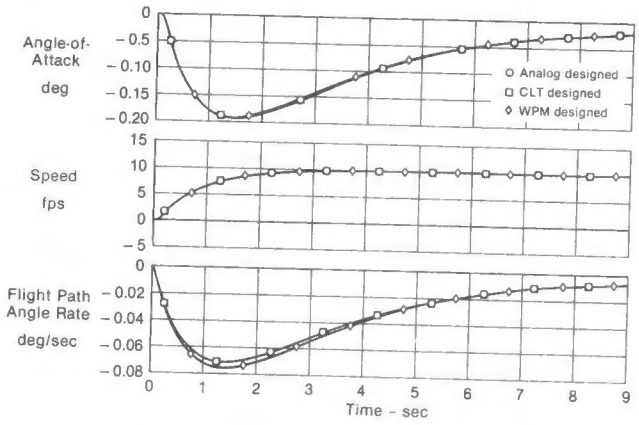


Fig. 16 Digital System Time Responses
Speed Command
40 Hz Sample Rate

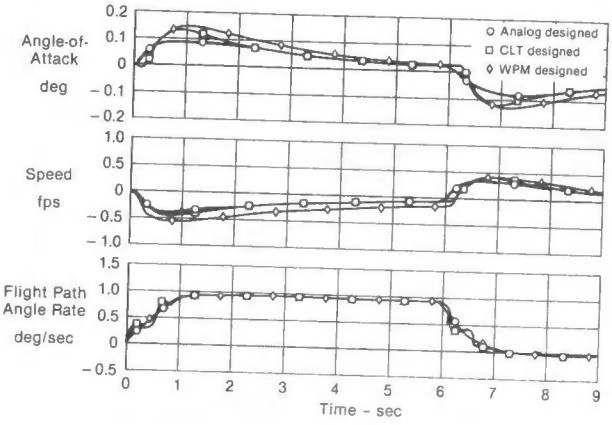


Fig. 19 Digital System Time Responses
Flight Path Command
10 Hz Sample Rate

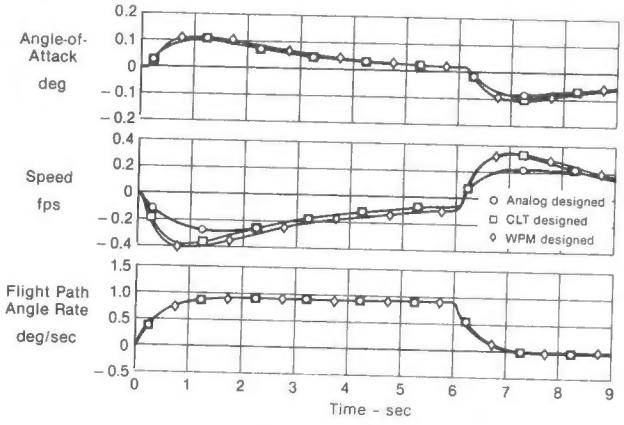


Fig. 17 Digital System Time Responses
Flight Path Rate Command
20 Hz Sample Rate

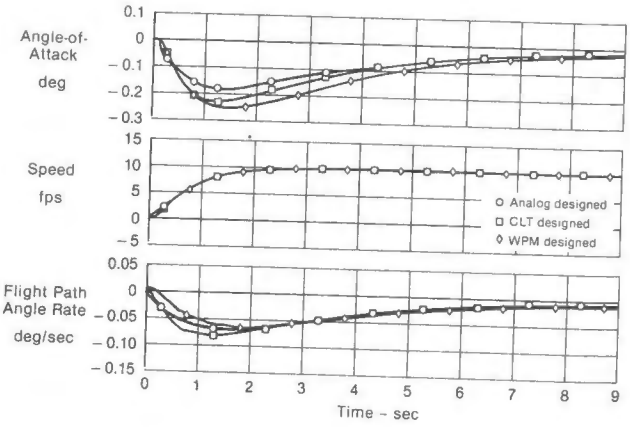


Fig. 20 Digital System Time Responses
Speed Command
10 Hz Sample Rate

The fact that the CLT responses degrade as sample rates are reduced can be related to the frequency response peak which occurs immediately below the Nyquist frequency. This peak is particularly evident in the flight path rate response at the three chosen sample rates. This peak is caused by Tustin transformation of high frequency modes in the s-plane designed controller which transform to lightly damped poles in the z-plane near the Nyquist frequency. At the lower sample rates, this response peak occurs at a sufficiently low frequency such that problems are induced in the control frequency range as shown. This problem can be eliminated by using order-reduction techniques to remove higher frequency s-plane controller modes prior to the Tustin transformation.

The W'-plane designed system responses do not contain lightly damped modes. The W'-transformation of the system inherently contains information about the sample rate. Therefore, if the SVS design process is successful in matching the desired open loop frequency response in the W'-plane, the resulting controller will not contain any low damped roots when transformed to the z-plane. Thus the extra step of removing high frequency dynamics from the controller prior to transforming it to the z-plane is not required in the W'-plane design method.

The stability margins were compared for the two digital control system designs. There were no significant differences between stability margins of the two systems. As expected, the digital system margins more closely approached those of the analog system for the higher sampling rates.

Thus, we conclude that the W'-plane design process is the better of the two techniques considered for accomplishing digital control system design. Its superiority is clearly evident when there is a requirement for minimum sampling rates.

CONCLUSIONS AND RECOMMENDATIONS

The results from this design of an integrated flight propulsion control system illustrate that the SVS technique can be a very powerful tool for designing integrated controllers for multivariable systems which provide the desired designer-specified frequency response characterisitcs.

Two digital multivariable control system design techniques were evaluated by application to this control system design. Both controller designs met design requirements when the sampling rate was relatively high. As sampling rate was reduced, the W'-plane technique provided superior designs. The CLT technique can be improved by developing additional software for controller order reduction. However, this still is an extra step not requied by one W'-plane method.

More work is recommended in two areas. The first is controller order reduction. In current SVS techniques, the controller has a larger order than the system model. This could challenge the throughput limits of onboard digital computers. Internal balancing techniques for order reduction are available but, in our experience, have not proved to be adequate in reducing controller order while preserving the performance and stability properties of the higher order controller. Frequency-weighted internal balancing techniques may be the answer. Certain techniques involving Hankle norms are also being pursued (Reference 2).

The other area requiring more investigation is that of stability assessment. The stability assessment techniques for multivariable systems are sufficient conditions which provide very conservative values for margins. This conservatism results from the very general nature of the allowable perturbation, namely, simultaneous and cross-channel perturbations among the loops. Some work is being performed in the area of structured singular values (Reference 4) to utilize the designer's knowledge of the allowable physical nature of the perturbations. Specifically, if the designer places some restrictions on the nature of the perturbation, larger values for the margins can be obtained.

REFERENCES

1. Doyle, J.C. and Stein, C., "Multivariable Feedback Design Concepts for a Classical/ Modern Synthesis", IEEE Transactions on Automatic Control, Vol. AC-26, pp 4-16, 1981.

2. Enns, F. Dale, "Model Reduction for Control System Design", PH.D. Thesis, Stanford University, March 1984.

3. Lehtomaki, N.A. et al., "Robustness Tests Utilizing the Structure of Modeling Error", CDC Conference on Decision and Control, San Diego, 1981.

4. Wall, Joseph E., Doyle, John C., and Stein, Gunter, "Performance and Robustness Analysis for Structured Uncertainty", IEEE 82CHI788-9/82 V-2.

84-2627

AIRCRAFT AUTOMATIC DIGITAL FLIGHT CONTROL SYSTEM WITH INVERSION OF THE MODEL IN THE FEED-FORWARD PATH

G. Allan Smith and George Meyer
NASA Ames Research Center, Moffett Field, California

Abstract

A full-flight-envelope automatic trajectory control system concept is being investigated at Ames Research Center. This concept was developed for advanced aircraft configurations with severe nonlinear characteristics. A feature of the system is an inverse of the complete nonlinear aircraft model as part of the feed-forward control path. Simulation and flight tests have been reported at previous Digital Avionics Systems conferences. A new method for the continuous real-time inversion of the aircraft model using a Newton-Raphson trim algorithm instead of the original inverse table look-up procedure has been developed. The results of a simulation study of a vertical attitude takeoff and landing aircraft using the new inversion technique are presented. Maneuvers were successfully carried out in all directions in the vertical-attitude hover mode. Transition runs from conventional flight through the region of lift-curve-slope reversal at an angle of attack of about 32° and to hover at zero speed in the vertical attitude showed satisfactory transient response. Simulations were also conducted in conventional flight at high subsonic speed in steep climb and with turns up to 4 g. Successful flight tests of the system with the new model-inversion technique in a UH-1H helicopter have recently been carried out.

Introduction

Some current high-performance fighter aircraft and helicopters, as well as many of those proposed for the future, have control and stability difficulties over some portions of their flight envelopes. These difficulties arise from 1) highly nonlinear aerodynamic and propulsion characteristics, 2) from undesirable coupling between axes, and 3) from the extreme range of flight conditions encountered over the flight regime. There may be instances when automatic flight control will be desirable for these aircraft over at least a portion of the flight regime. For example, precision trajectory control during night landing on a carrier at sea, terrain following with a helicopter, or hover-mode control of a vertical attitude takeoff and landing (VATOL) aircraft; of course, complete atuomatic control is required for remotely piloted vehicles and cruise missiles.

A powerful new automatic flight control system concept suitable for trajectory control of the aforementioned aircraft types is being investigated at Ames Research Center. The implementation of this concept is made possible by the airborne digital computer. This is a total aircraft flight control system which combines attitude and thrust control in a unified system for operation over the full flight envelope.

The flight control system is based on a model-following concept in which several models of different complexity are used in the system, including some that have essentially all of the aerodynamic and propulsion details that would be found in a complete model used for nonlinear, real-time, manual simulation studies. A unique feature of the control system is the continuous real-time inversion of such a complete nonlinear model of the aircraft for the purpose of defining the aerodynamic and propulsion control commands based on trajectory commands and regular outputs.

The development of this control system concept is reported in Refs. 1-3. Successful flight tests in the Augmentor Wing Jet STOL research aircraft (4), in the DHC-6 aircraft (5), and in a UH-1H helicopter (6) have been carried out. Simulator studies of a carrier landing were reported at the 1979 Digital Avionics Systems Conference (7), and a preliminary report of a simulation of hover-mode control for a VATOL configuration was given at the 1981 conference (8). The overall structure of the control system is described in the above references, and only a brief explanation of the general system configuration will be presented in this paper. Emphasis will be on a different technique for the real-time inversion of the nonlinear aircraft model and on a discussion of simulation results of its application to the control of a VATOL aircraft over the full flight regime from vertical-attitude hover through transition to conventional flight.

Figure 1 is an artist's conception of a Navy VATOL aircraft configuration suitable for launch and recovery at the side of a small ship. The aerodynamic and propulsion characteristics of this proposed aircraft, which was developed by the Vought Corporation (9-11), are used in the example discussed in this paper. This conceptual aircraft was chosen to illustrate the extensive flight regime and range of nonlinear characteristics that can be accommodated by the control system.

Control System Design

The Ames control system concept has several key features. Fundamentally, it combines feed-forward and feedback control, with the forward path split into a command section and a control section (Fig. 2). The feed-forward controller is shown in solid lines, and the feedback control is shown in dotted lines. Most signals in this control system are three-dimensional column vectors with appropriate subscripts.

The feed-forward command section generates smooth, executable, and consistent acceleration A_c, velocity V_c, and position R_c command vectors in the reference inertial axis system (north, east, and down) in response to corresponding rough trajectory command inputs A_i, V_i, R_i. These rough inputs may be supplied from an air-traffic-control system, from ground vectoring, or from a trajectory time-history either stored in the computer or generated on line by the pilot.

The control section of the feed-forward controller computes the aircraft thrust and moment controls U needed to produce the total command-acceleration vector A_T. It is a particular feature of the

system that the series combination of the control section and the actual aircraft is essentially linear with decoupled axes when viewed from the command section. This is achieved by embedding an inverse model of the aircraft in the control section that determines the aircraft controls necessary to produce the commanded acceleration. The regulator feedback loop is closed around this approximate identity and hence can be easily designed by linear methods. It is only called upon to compensate for disturbances and model uncertainty.

Control System Implementation

The command generator (Fig. 2) accepts the rough input trajectory commands that may not be executable because of discontinuities or deficiencies in some components of the vectors and produces a complete and consistent set of smooth, executable acceleration, velocity, and position vectors. This is accomplished by using a canonical model of the aircraft; the model consists of a string of four integrators for each of the three channels which correspond to the components of the input command vectors. Adjustment of gains and limits provides smooth, executable output vector commands (1). The smooth commanded acceleration (A_c) output of the command generator is added to the closed-loop feedback acceleration command (ΔA_c) output of the regulator to give the total commanded-acceleration vector (A_T).

Since the control section includes an inverse model of the aircraft it is appropriate to first consider the aircraft model itself (Fig. 3). This model has been specialized to represent the VATOL aircraft used in this simulation. The four controls (U) are normalized thrust and three angular acceleration controls for pitch, roll, and yaw which are converted to various aircraft control deflections. The VATOL jet engine has an output nozzle capable of swiveling through ±15° in both pitch and yaw. The four input control commands are divided appropriately between the various aircraft controls, which include throttle, elevons, rudder, flaps, engine-nozzle angle, and engine bleed-air reaction jets at the wing tips for roll control in the hover mode.

The outputs of the control actuators in Fig. 3 are input to the force and moment generating section where aerodynamic forces and moments in body axes are computed as a result of the angle of attack α and the velocity with respect to wind in body axes, V_{wb}. A detailed engine model with afterburner is used in calculations of thrust and ram drag, including such refinements as engine gyroscopic moments and thrust losses owing to nozzle turning angle and the bleed air used for roll control. The body-axis moments M_b are used with the aircraft inertia matrix J and the Euler angular-rate equations to calculate angular acceleration $\dot{\omega}$, angular velocity ω, and aircraft attitude, which is expressed as TBR, the direction-cosine matrix of rotation from reference axes to body axes. The inverse (transpose TRB) is used to transform F_b from body axes to reference axes. The addition of the gravity vector G and division by the aircraft mass m yields the acceleration vector A. This acceleration vector is the essential output of the direct model although integrations to give velocity V and position R are included in the figure.

The control section of the feed-forward controller (Fig. 2) is functionally the inverse of the aircraft model just described. The input is the total acceleration command vector A_T, and it is applied to the model inverse to determine the corresponding thrust and attitude control positions. The overall operation of the trajectory control system should now be clear (through reference to the figures) and attention will be directed to the details of the model-inversion process, which differs substantially from the scheme used for previous applications.

Model-Inversion Process

In the early implementations of this control system concept (4,5), the model inversion was carried out with the aid of extensive tables of nonlinear aerodynamic data that related lift and drag coefficients to angle of attack and thrust. Simulation and flight results were satisfactory for these aircraft for which the configuration allowed the force and moment equations to be treated separately. An exhaustive search technique was then used to solve two-dimensional tables.

The new model-inversion technique is suitable for more complex configurations with serious nonlinearities.

A symbolic explanation of the model-inversion scheme is presented in Fig. 4. The upper diagram indicates how the aircraft model shown in Fig. 3 is normally used in a conventional simulation study. For any particular set of velocity vectors, knowledge of the control inputs U and the aircraft attitude TBR allows the output accelerations to be calculated by using equations for the various aerodynamic and propulsion forces and moments. The lower diagram in Fig. 4 represents the model-inversion problem. It will be recognized by those engaged in aircraft simulation studies as the approach for computing the control positions required to trim the aircraft at a given flight condition. The aircraft velocities and commanded accelerations and sideslip are specified as inputs, and it is desired to calculate as outputs the corresponding aircraft attitude and controls that would produce the commanded inputs.

The inverse problem of Fig. 4b cannot be directly solved as illustrated in that diagram, because we do not have analytical expressions for directly calculating the trim variables. Instead, the trim method applies an iterative Newton-Raphson procedure to the aircraft model of Fig. 4a. The Newton-Raphson trim is essentially a very refined cut-and-try process. It is a multivariable adaptation of Newton's method for finding the root of an equation by calculating a local derivative and using a linear extrapolation to find an approximation to the root in an iterative way.

The trim procedure is quite standard so only a brief discussion of particular features will be given here. The trim procedure employs the force and moment vector equations in body axes:

$$F_b + TBR_c(G - mA_T) = EF$$

$$M_b + S(\omega_c)J\omega_c - J\dot{\omega}_c = EM$$

These equations (1) apply to the aircraft model embedded in the control system and have, therefore, commanded variables as indicated by the subscripts. These two vector equations represent six scalar equations. The first three are force-error equations, and the last three are moment-error equations. The force represented in body axes is F_b. The gravity vector G and the commanded acceleration vector A_T are both represented in reference axes. They are transformed by TBR so that all terms are in body axes. If the aircraft model is in force trim, the three components of the error (EF) are zero. The M_b term is the torque vector represented in body axes, ω_c is the angular velocity, $\dot{\omega}_c$ is the commanded angular acceleration, J is the aircraft inertia matrix, and $S(\omega_c)$ is the skew symmetric matrix function of angular velocity. If the model is in moment trim, the three components of EM are zero. For trim in conventional flight, the rotation matrix TBR_c is implemented as the product of five elementary direction-cosine matrices that represent a heading ψ_c and flightpath angle Γ_c defined by the commanded velocity vector V_c, a commanded sideslip β_c (usually zero), a roll angle ϕ_c, and angle of attack α_c. The first three angles are fixed during a trim cycle, and roll and angle of attack are two of the trim variables. The other trim variables are the four controls U.

The trim procedure first selects trial values of the six trim variables (input variables in Fig. 4a). These are rough estimates for the initial trim or values from the preceding trim cycle as the simulation proceeds. These trial trim variables are then used along with the commanded velocity vectors to calculate the forces and moments which are substituted into the trim vector equations along with the commanded accelerations to find the errors (EF and EM). If they are below specified tolerance values, the model is trimmed and the four trim variables that constitute the aircraft controls U are sent to the actual aircraft actuators. Nominal tolerances were taken as 0.001 g for the force equations and 0.005 rad/sec^2 for the moment equations.

If any errors exceed their tolerance, a perturbation procedure is initiated. One trim variable is perturbed by a small amount from the trial value, and the force and moment calculations are repeated to determine new values of the six trim-equation errors. This is done six times, once for a perturbation of each of the six trim variables. A six-by-six Jacobian matrix of error derivatives of each scalar equation with respect to each trim variable is formed where, for example, the third-row, fifth-column term is the partial derivative of vertical force with respect to angle of attack.

The Newton-Raphson procedure then inverts this matrix to give the estimated changes in the trial values of each trim variable to iteratively produce a set of balanced equations. These new trim variables are used to compute forces, moments, and the angular transformations required in the trim equations; if the six equation errors are less than the tolerance test, the model is trimmed. If not, the entire process is repeated with seven more passes through the model; one pass for each trim variable perturbation and one to verify trim. A single set of seven passes is sufficient for most flight conditions, but two or more sets are needed for parts of the trajectory when commanded accelerations are changing rapidly or when external disturbances are encountered.

The use of the error equations in the trim procedure should now be clear. There are, however, several interesting details involved in the actual system. One important consideration arises when one attempts to trim the aircraft in a vertical attitude. When the aircraft is in a vertical-attitude hover, all velocities may become zero, so that angle of attack is undefined. Therefore, for low velocities in the vertical attitude, the angle of attack is replaced as a trim variable by the pitch attitude θ_c. Alpha is, of course, still computed and used, though not as a trim variable, at all velocities where it is defined. Furthermore, in hover θ_c is nearly 90° with respect to reference axes which is a singular point for the transformation TBR_c. As a result, a change is made for the low speed, and aircraft attitude is specified by the three conventional Euler angles; the angles are measured with respect to an inertial system rotated 90° from the reference axis system. This inertial system is called the perpendicular system, and its axes are positive up, east, and north, respectively. For this condition, the transformation matrix from the perpendicular system to body axes is designated TBP_c and is implemented as the product of three elementary direction-cosine matrices representing the Euler angles. The pitch angle about the aircraft Y-axis and the yaw angle about the Z-axis are taken as trim variables. The other Euler angle is a commanded heading angle about the X-axis; it is the angle between north and the aircraft landing gear in the vertical attitude and remains constant during a trim cycle.

A constant transformation matrix TPR is used to rotate from the reference axes 90° about the east horizontal axes to the perpendicular system. These rotations are used in the force-error equations to get all terms in the perpendicular-axis system so that the force-error equation becomes

$$TPB_c\ F_b + TPR(G - mA_T) = EF$$

In the simulation, a preselected pitch attitude of 60° from the horizontal was used to switch between the two different sets of force-error equations and to substitute pitch attitude for angle of attack as a trim variable. Operation was such that only very slight transients were observable in the system. This was expected, because the transfer is merely a computational operation, and no switching of physical sensors would be involved in an actual aircraft installation.

Other details in the trim procedure may be seen in Fig. 5, which is an overall flow diagram of the actual system. Note that several differences exist between it and the simplified configuration shown in Fig. 2. The control section has been split into translational and rotational sections, each with a command generator and a regulator. We have been discussing the six-degree-of-freedom trim whose input signal is the total commanded-acceleration vector A_T and whose output is the rough commanded-attitude matrix TBR_c which contains two of the trim variables. Furthermore, the angular velocity ω_c and angular acceleration $\dot{\omega}_c$ are taken as zero for the six-degree-of-freedom trim.

The matrix TBR_c is input to the rotational-command generator which is functionally similar to the translational-command generator and whose output, a smooth commanded angular acceleration vector $\dot{\omega}_c$, is combined with the output of the rotational regulator

$\Delta\dot{\omega}_c$ to give a total commanded angular-acceleration vector $\dot{\omega}_T$. The rotational regulator compares the aircraft attitude TBR and angular velocity ω with a smooth commanded-attitude matrix TBR_{SC} and angular velocity ω_c to generate the closed-loop angular-acceleration command increment $\Delta\dot{\omega}_c$. The total angular-acceleration command is input to a four-degree-of-freedom Newton-Raphson trim section, which is entirely analogous to the previous six-degree-of-freedom trim section except that only four trim equations are used: the three scalar moment equations and the first of the three scalar force equations.

The four trim variables are thrust, roll, pitch, and yaw. The actual aircraft attitude matrix TBR and angular velocity vector ω are used in these trim equations. The outputs of the four-degree-of-freedom trim are the four trim variables, which become the controls U; they are input to the actual aircraft. This trim differs from the six-degree-of-freedom trim in that it has a commanded angular-acceleration input and an angular-velocity input in addition to the same commanded translational-acceleration input A_T. The Newton-Raphson technique again requires one pass through the model to determine initial errors and four additional passes for perturbation of the four trim variables for at least five passes each cycle time.

The simple diagram of Fig. 2 does not distinguish between the two trim implementations. The six-degree-of-freedom trim is used to determine aircraft attitude required for the commanded translational acceleration, but assuming zero angular velocity and acceleration. Thus, the four resulting control-trim variables at this point are only approximate and are not used further. The four-degree-of-freedom trim uses the actual aircraft attitude and angular velocity and determines the four control variables necessary to produce the commanded angular-acceleration vector. The four-degree-of-freedom trim switches from reference axes to perpendicular axes at the same time the six-degree-of-freedom trim switches.

Simulation Results

A series of simulation studies was performed using the control system just described in the VATOL aircraft previously discussed. Maneuvers were performed in the vertical-attitude hover mode in all directions, both individually and simultaneously and with various aircraft roll rates about the vertical axis. Accelerations up to 0.3 g and velocities up to 50 ft/sec were investigated. The response to gusts and steady winds was also studied. In the conventional-flight mode, major attention was directed to the transition between horizontal flight and hover. Tests were also carried out at high subsonic speed, with turns up to 4 g and flightpath angles to 10°. The results of two different flight trajectories are presented: maneuvering in the vertical attitude at low speed and making a transition run from conventional level flight to hover. Time-histories are shown for each trajectory; they include aircraft accelerations, velocities, and displacements, as well as attitude angles and control variations.

The results shown are preliminary in that no particular effort was made to optimize transient performance by gain or limit adjustments, since stable and reasonably satisfactory performance resulted from the initial settings.

Vertical-Attitude Maneuvering

Performance during maneuvers in the vertical attitude is shown in Fig. 6. The aircraft is initialized in hover in a vertical attitude at zero velocity with the landing gear facing north and thrust equal to weight; the initial altitude was 1000 ft. A step command of 3 ft/sec^2 horizontal acceleration was applied at 3 sec for a 15-sec period, followed by a command of 3 ft/sec^2 acceleration vertically at 30 sec for 10 sec. A roll command of 20°/sec about the vertical for 5 sec was applied at 50 sec. Thus, at 65 sec the aircraft was translating horizontally at 45 ft/sec, translating up at 30 ft/sec, and rolling at 20°/sec. Simultaneous commands were then given to bring the aircraft back to the hover condition, which was essentially achieved by 90 sec.

The first plot in Fig. 6 shows both the rough horizontal acceleration command which is A_i, the first component of the rough commanded-acceleration column vector in Fig. 2, and the smooth command A_c from the command generator. The curve for A_c has appreciable overshoot, because the command generator forces its position and velocity outputs V_c and R_c in the steady state to be equal to the rough commanded inputs V_i and R_i, respectively. Hence, transient lags in smooth velocity and in position that are inherent in the basic aircraft must be compensated by additional smooth commanded acceleration, even though the frequency response of the command generator canonical model is critically damped. The total commanded acceleration in the second plot in Fig. 6 is A_T and it includes the feedback corrective acceleration signal ΔA_c. The control system performance can be evaluated by noting how closely the actual acceleration A, indicated by the solid line on the same plot, follows A_T. The tracking is quite close except at the four points labeled a-d in Fig. 6, which show an initial aircraft acceleration in the direction opposite to the command. This reflects the nonminimum phase character (right-half plane zero) of the aircraft transfer function as the engine nozzle swivels forward to rotate the aircraft for translation and thus produces a momentary force opposing the intended translation. This is similar to the initial downward acceleration caused by elevator deflection when starting a climb in a conventional aircraft. In the third plot in Fig. 6, the velocity reaches a steady value of 45 ft/sec, and the horizontal translation in the fourth plot increases to almost 3000 ft, finally.

The fifth and sixth plots (Fig. 6) show the vertical-acceleration commands and responses. There is no nonminimum phase effect in the vertical channel since the acceleration is due to engine thrust. The actual acceleration follows the total command very closely except for a transient effect (point e in Fig. 6) during the roll-rate application. The seventh plot shows that vertical velocity responds with essentially no overshoot and reaches a value of 30 ft/sec. The altitude response is shown along with horizontal distance on the fourth plot. Both the rough roll-rate command and the roll-angle response are shown on the eighth plot.

In the low-speed, vertical-attitude regime the aircraft attitude is measured by Euler angles in the perpendicular axis system. The pitch angle in the next plot in Fig. 6 is not quite zero in hover, owing to an engine offset from the aircraft centerline. As shown in this plot, the aircraft pitches forward and the yaw angle remains zero until the aircraft starts to roll. Basically the aircraft thrust vector is deflected to provide horizontal acceleration; that is, a deflection from vertical of about 0.1 rad to give a forward acceleration of 0.1 g. No deflection would be required to sustain a constant velocity in the absence of aerodynamic forces. However, at an airspeed of 54 ft/sec an aircraft pitch of 12° is needed to overcome aerodynamic forces and engine ram-drag effects. As the aircraft rotates about the vertical, the thrust vector is deflected in yaw and less in pitch until, after 100° of roll, there are about 6° of deflection in yaw and 2° in pitch. The size of the angles indicates differences in aerodynamic and propulsion forces in the two axes. During all of these commanded rotational maneuvers, the trajectory deviated less than 2 ft from the smooth command. The thrust curve in the tenth plot shows a correlation with the vertical acceleration.

The normalized pitch channel signal in the eleventh plot in Fig. 6 reflects the pitching-moment requirements needed for angular acceleration and to counteract aerodynamic moments as airspeed increases. The yaw control indicates the shift from pitch to yaw control as the aircraft rolls. The roll control (shown by the dotted line) initiates the roll rate at 50 sec. The final plot shows the engine nozzle pitch and yaw deflections which remain well below their maximum limits of 15°.

At 65 sec, simultaneous commands (points f-h in Fig. 6) are applied to all three axes to reverse the original commands and restore the aircraft to a hover condition at an altitude of 2000 ft. The commands and responses are nearly the inverse of the original, except for coupling effects caused by the initial roll angle. It will be noted that the axes are quite well decoupled during the separate commands of the first 50 sec and that the major coupling effects for simultaneous commands occur in the controls and angular response so that as a basic translational acceleration control system, the trajectory response in the different axes remains nearly decoupled for all commands. The trim equations were successfully iterated in one cycle for over 95% of the run and required only two compute cycles during the transient adjustments to step commands.

Other vertical-attitude maneuvers showed similarly satisfactory response to lateral translational commands.

Transition Run from Forward Flight to Hover

Performance during a 60-sec transition run at constant altitude is shown in Fig. 7. The aircraft is initialized in level flight at a velocity of 180 ft/sec at an altitude of 1000 ft. A deceleration command of 3 ft/sec^2 is given at 5 sec and is increased to 12 ft/sec^2 at 33 sec. The commanded acceleration is reduced to zero at 41 sec when the corresponding velocity command reaches zero. The vertical-acceleration command is zero throughout the run.

The point of particular interest in this run is the response during lift-curve-slope reversal beginning at about 22 sec. The lift-curve-slope reversal near an angle of attack of 32° presents a problem to the essentially linear Newton-Raphson technique. In order to achieve trim, the predicted corrections were reduced to 80% to prevent hunting in the algorithm. The difficulty was compounded by the configuration with a large canard surface whose local angle of attack caused the slope of the lift curve for that surface to reverse at a different speed than for the main wing. Most of the same variables shown on Fig. 6 are also shown in Fig. 7 where they display many of the same correlations.

The step deceleration command A_i at 5 sec shown in the first plot of Fig. 7 results in the smooth command A_c on the same plot and the total command A_T and the aircraft response A on the second plot. The velocity V on the third plot begins to decrease and the position R in the fourth plot continues to increase but at a slower rate. A slight vertical-acceleration response occurs between 5 and 15 sec accompanied by a 2-ft loss of altitude. The pitch angle, measured from the horizontal, and the angle of attack begin to increase from their initial values of 19° as the aircraft responds to the pitch-control command in the ninth plot and the corresponding pitch nozzle in the tenth plot. The thrust decreases almost to its minimum value during the initial deceleration.

As airspeed decreases, the angle of attack in the eleventh plot (Fig. 7) increases to maintain lift. The aerodynamic lift curve of the wing reaches its peak at an angle of attack of about 32°, at which point it reverses following wing stall. This effect is seen at 22 sec where a vertical acceleration transient (point a) occurs as the trim process tries to obtain increased lift by increasing the angle of attack. At this point, the trim routine must maintain altitude by using additional engine thrust at a higher pitch attitude, shown by the thrust (point b) and pitch angle (point c) as the aircraft transitions to the vertical attitude mode. Airspeed is about 130 ft/sec at this time. Only 5 ft of altitude were lost in the transition. The aircraft continues to slow down and the attitude increases until at 30 sec a pitch attitude of 60° is reached, and the attitude direction-cosine matrix is switched from the reference-axis to the perpendicular-axis system. This is seen in the seventh plot (Fig. 7), where the pitch angle jumps from 60° to -30°. Only a slight response was observed in vertical acceleration. At 33 sec, the deceleration command is increased to 12 ft/sec^2, and transients similar to those seen in Fig. 6 occur in all variables. The nonminimum phase behavior on the second plot is denoted by points d and e. The rapid rise to a vertical attitude (i.e., zero pitch angle) is shown by f in the seventh plot in Fig. 7. When the commanded velocity reaches zero, the deceleration command is removed and the aircraft comes to a hover position at zero velocity.

The transition starting at 22 sec required two trim cycles for about 5 sec and was equally troublesome when other transition runs were made going from low to high speed while climbing at 30 ft/sec. There was less than 2 ft of sidewise deviation from the trajectory except when wind gusts were applied. This aircraft has such a clean aerodynamic profile that deceleration along the flightpath is difficult to achieve. An initial deceleration command of much

more than 0.1 g would have called for a thrust from the trim process of less than the engine idle setting. The maximum deceleration obtainable varies from about 0.1 g at low speed to 0.4 g at an airspeed of 700 ft/sec in conventional flight. Of course, much higher deceleration can be achieved in the vertical attitude mode.

Other Trajectories

Several other trajectories for which data are not shown were simulated. In particular, a turning, climbing, accelerating trajectory with speed change from 300 to 900 ft/sec and climb at a 10° flightpath angle showed very satisfactory performance, including the execution of turns of 2, 3, and 4 g. Some trajectories were subjected to wind gust and to initial offsets; the transient results were satisfactory.

Concluding Remarks

The objective of this simulation study was to apply a recently developed trajectory control system concept to a vertical attitude takeoff and landing aircraft model with severe aerodynamic nonlinearities. Tests were carried out over an extensive flight envelope from vertical-attitude hover through transition to conventional flight at high subsonic speeds. The control concept features the continuous real-time inversion of a detailed model of the aircraft. This study employed a new method of inverting the model by a Newton-Raphson trim technique instead of the inverse table look-up scheme that was originally used. Several specific comments regarding the study are presented below.

1) The results presented here (and those of other simulation runs of different trajectories which were not presented because of a lack of space) show that the control system performs satisfactorily over a large part of the flight envelope. The other tests included 4-g turns at Mach No. 0.8 and steep climbs, as well as low-speed, vertical-attitude maneuvers in the hover mode with simultaneous velocity commands in all three axes and rapid rolling maneuvers.

2) The lift-curve-slope reversal near an angle of attack of 32° caused instability in the trim algorithm initially, but it was overcome by minor modifications to the technique.

3) This investigation was undertaken to demonstrate the performance of the control concept with the Newton-Raphson model-inversion technique in a conceptual aircraft with a very large flight envelope. Although this objective was essentially accomplished, a number of areas warrant study in greater detail to improve performance and define limitations on maneuvering capability.

4) The use of a perpendicular inertial axis system for vertical-attitude hover maneuvers was quite satisfactory. No appreciable transients occurred, although the aircraft attitude matrix was switched at three places in the control circuits simultaneously. The switching point is not critical and could be anywhere between attitudes of 45° and 85°, so a suitable hysteresis loop could be designed to prevent any switching chatter when going through the switching region in either direction.

5) The system response to wind gusts (not shown here) was satisfactory for various reasonable combinations of wind magnitude and direction. However, trim failure could be induced for sufficiently severe disturbances. No attempt has yet been made to develop a logic to cope with such conditions.

6) The description of the control system concept mentioned that the rough trajectory command inputs could be generated on line by a pilot rather than by precomputed trajectory segments. Simulation runs were made for the conventional flight mode in which the pilot's fore and aft stick position commanded vertical velocity and the sideways stick displacement commanded the g level of a horizontal turn. Throttle position commanded airspeed. Under these conditions, the aircraft performance was quite satisfactory. It should be noted that only a few components of the rough command input vectors were generated and some of these were in a body-axis or heading-axis system instead of in reference axes. Nevertheless, suitable input transformations allowed the command generator to provide a full set of smooth commanded vectors in reference axes.

7) The credibility of these simulation results is supported by the flight-test history of the concept in three aircraft. Flight tests in a DHC-6 Twin Otter aircraft (5) and in the Augmentor Wing Jet STOL research aircraft (4), both of which used multidimensional table look-up for model inversion, showed good agreement between simulation and flight-test results. A current flight-test program using the Newton-Raphson model-inversion technique in a UH-1H helicopter has successfully controlled that aircraft over a complex maneuvering trajectory in spite of strong nonlinear coupling between axes.

References

(1) Smith, G. Allan and Meyer, George, "Application of the Concept of Dynamic Trim Control to Automatic Landing of Carrier Aircraft," NASA TP-1512, 1980.

(2) Meyer, George, "The Design of Exact Nonlinear Model Followers," Proceedings of the Joint Automatic Control Conference, University of Virginia, Charlottesville, Va., June 1981.

(3) Meyer, George, "Nonlinear Systems Approach to Control System Design," First Annual Aircraft Controls Workshop, NASA Langley Research Center, Hampton, Va., NASA CP-2296, 1983.

(4) Meyer, George and Cicolani, Luigi, "Application of Nonlinear Systems Inverses to Automatic Flight Control Design—System Concepts and Flight Evaluations," AGARDograph No. 251, Theory and Applications of Optimal Control in Aerospace Systems, July 1981.

(5) Meyer, George and Wehrend, William, "DHC-6 Flight Tests of the Total Automatic Flight Control System (TAFCOS) Concept on a DHC-6 Twin Otter Aircraft," NASA TP-1513, 1980.

(6) Meyer George, Hunt, Robert L., and Su, Renjeng, "Design of a Helicopter Autopilot by Means of Linearizing Transformations," Guidance and Control Panel 35th Symposium, Lisbon, Protuagl (AGARD Conference Proceedings No. 321, 1983).

(7) Smith, G. Allan and Meyer, George, "Total Aircraft Flight Control System—Balanced Open and Closed Loop Control with Dynamic Trim Maps," IEEE Cat. No. 79CH1518-0, Proceedings of the 3rd Digital Avionics Systems Conference, Fort Worth, Tex., Nov. 1979.

(8) Smith, G. Allan and Meyer, George, "Application of the Concept of Dynamic Trim Control and Nonlinear System Inverses to Automatic Control of a Vertical Attitude Takeoff and Landing Aircraft," Proceedings of the 4th Digital Avionics Systems Conference, St. Louis, Mo., Nov. 1981.

(9) Driggers, H. R., "Study of Aerodynamic Technology for VSTOL Fighter/Attack Aircraft, NASA CR-152132, 1978.

(10) "USAF Stability and Control DATCOM," Air Force Flight Dynamics Laboratory, Wright-Patterson AFB, Ohio, Oct. 1960 (Apr. 1978 revision).

(11) Fortenbaugh, Robert L., "A Mathematical Model for Vertical Attitude Takeoff and Landing (VATOL) Aircraft Simulation." Prepared under contract NAS2-10294 by Vought Corporation, Dallas, Tex., for NASA Ames Research Center, Dec. 1980.

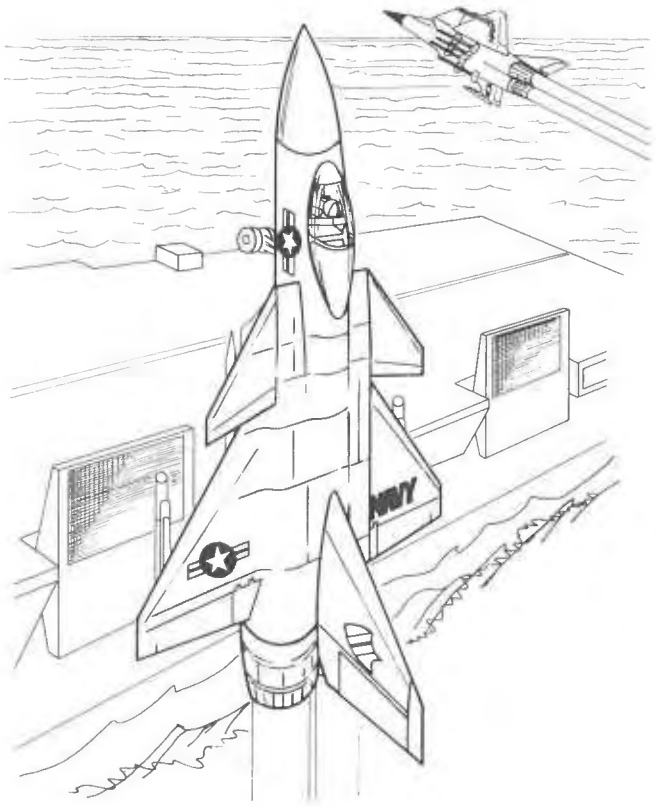


Fig. 1 VATOL aircraft used in simulation study.

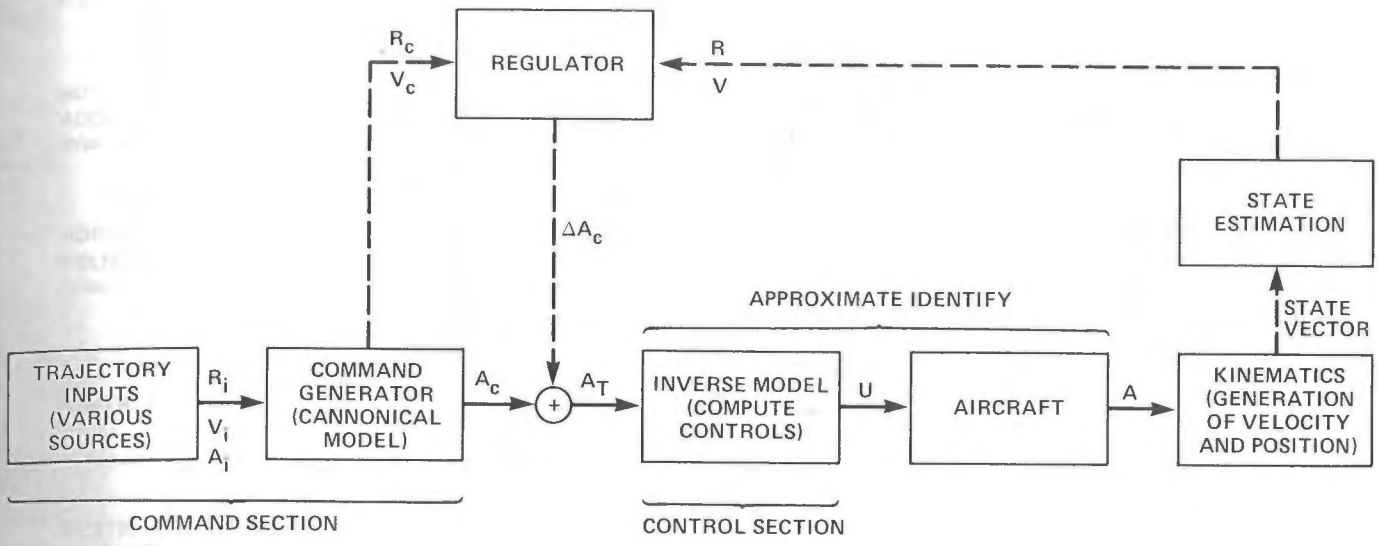


Fig. 2 Essential aspects of the nonlinear control concept.

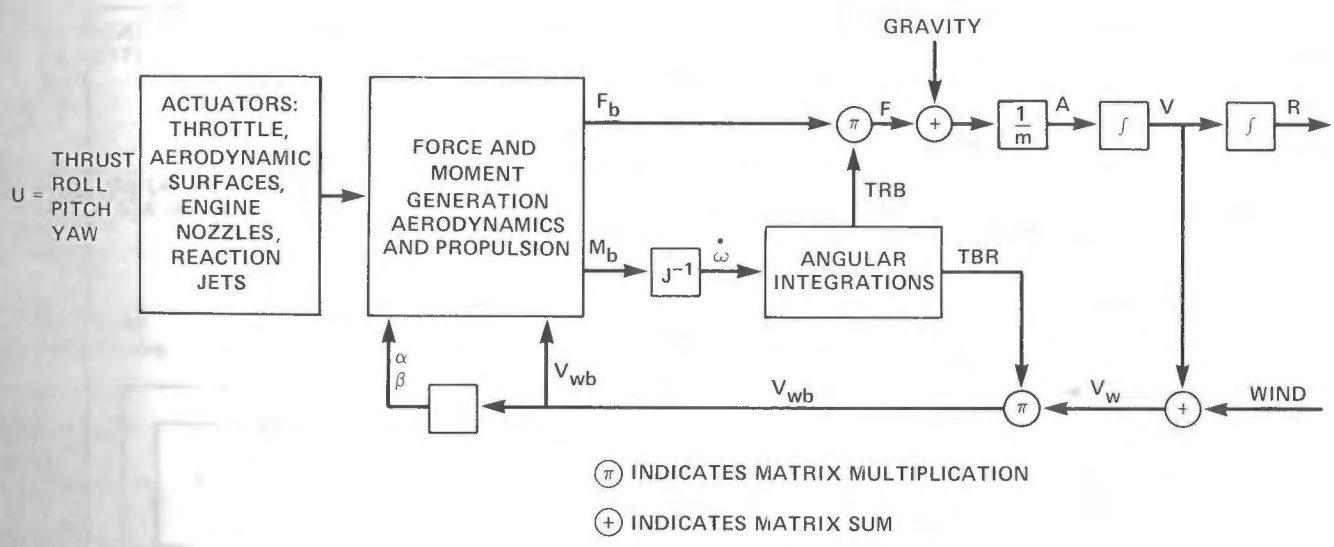


Fig. 3 Aircraft model.

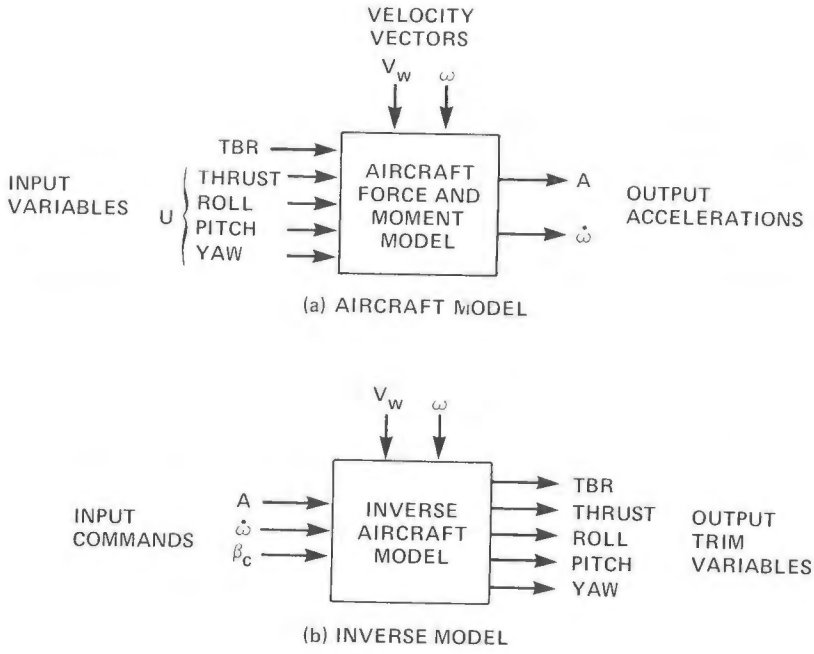


Fig. 4 Model inversion.

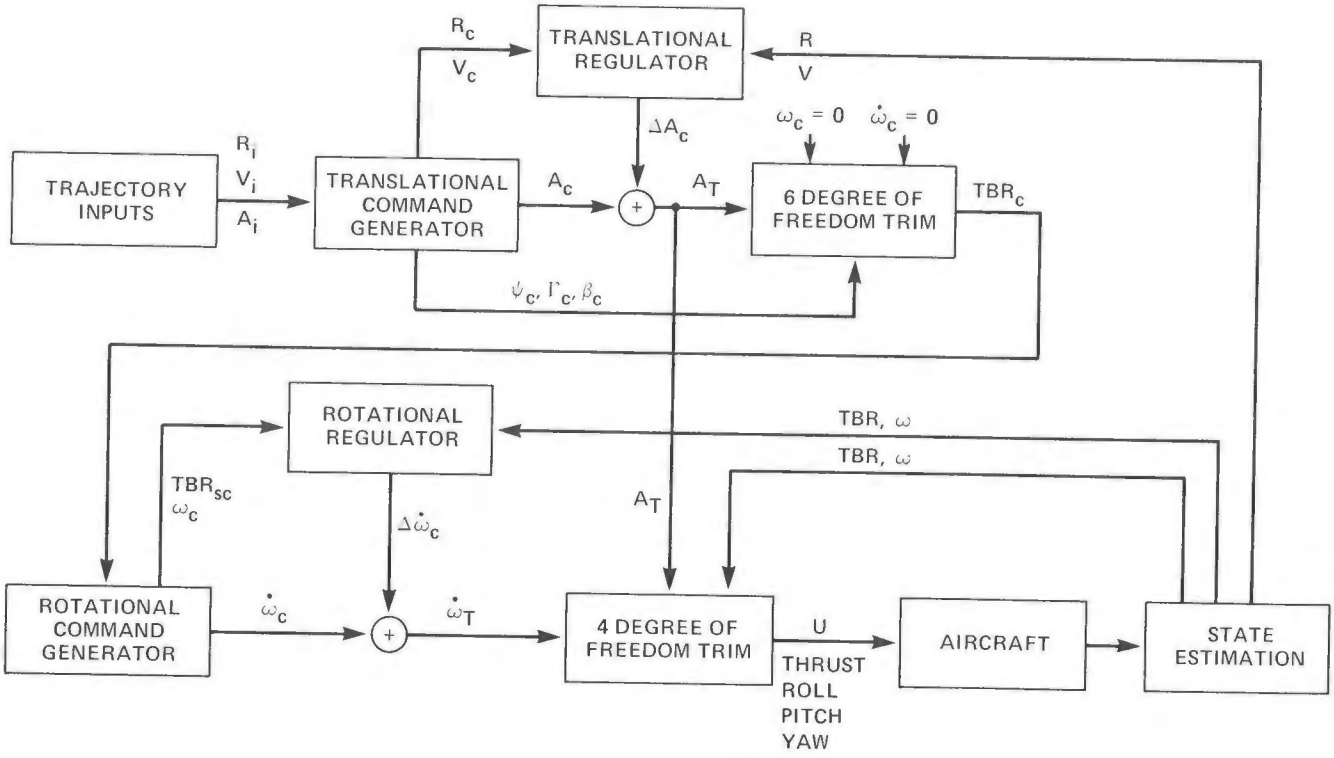


Fig. 5 Complete control system.

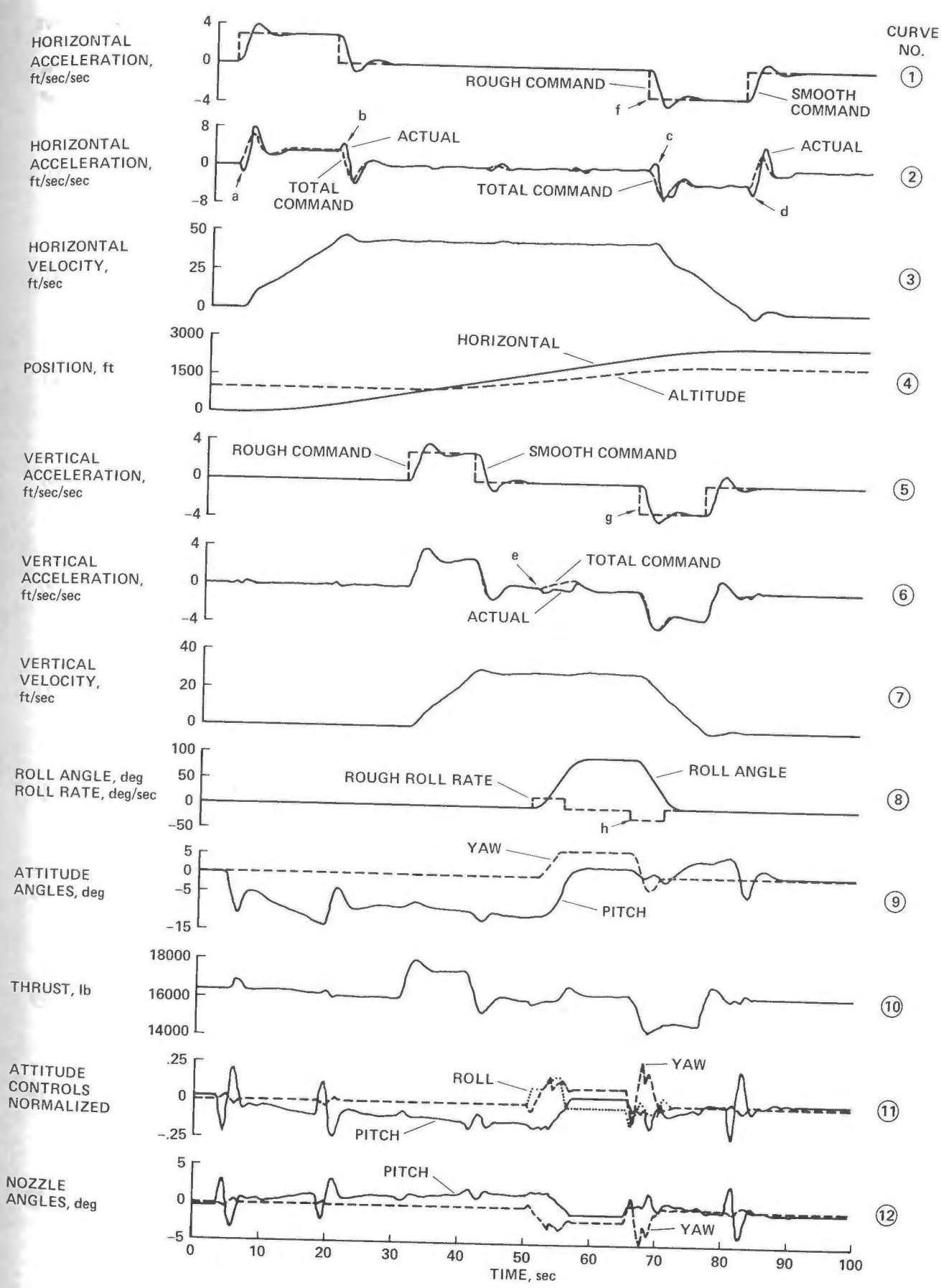


Fig. 6 Vertical attitude maneuvers about hover.

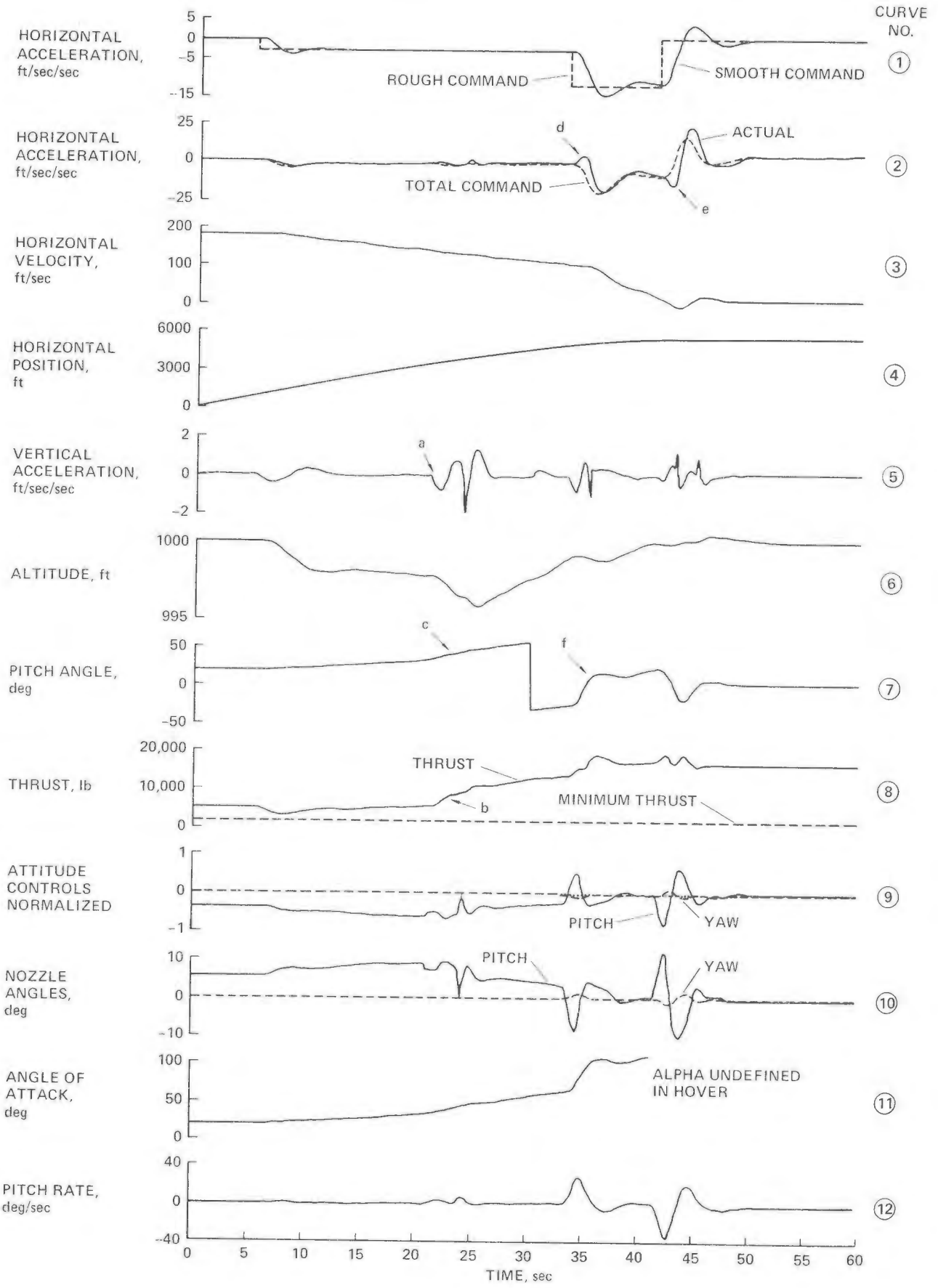


Fig. 7 Transition from forward flight to hover.

SESSION 6

SYSTEMS AND SOFTWARE - DEVELOPMENT AND EVALUATION TOOLS

6

Chairmen:

Ed Waltz
Allied Bendix Aerospace
Bendix Communication Division

Don Brown
General Dynamics
Fort Worth Division

This session covers tools used for avionic system development, such as system level concept development and definition tools, simulation aids, and system test tools for testing integrated hardware and software.

DEVELOPMENT OF A SUPPORT SOFTWARE SYSTEM FOR REAL-TIME HAL/S APPLICATIONS

Rudeen S. Smith*
NASA Langley Research Center
Hampton, VA 23665

Abstract

The demand for real-time operating systems and associated support software is increasing significantly with the expanded use of embedded computers to control time-critical actions of other systems. For example, these computers are being used as on-board flight systems to investigate new concepts in navigation, flight controls, and flight management of aircraft. Until recently, few operating systems and even fewer high-level languages provided the real-time features necessary to fully develop and test these new flight concepts. There has been significant advancement in real-time operating systems but still there is little real-time support by the high-level languages. This paper describes a real-time software system that has been developed for the language HAL/S in support of the Advanced Transport Operating Systems (ATOPS) flight project currently operational at the NASA Langley Research Center. This system provides an environment that supports the development and testing of both the real-time flight executive and the experimental real-time applications.

Introduction

The increased use of embedded computer systems for real-time applications has resulted in a rapid growth of real-time operating systems. Because of the complexity of real-time problems and the difficulty in code validation, it is crucial that considerable software support be given the real-time applications programmer. Unfortunately, few high-level languages can support the real-time requirements of the application user, nor do they have the capability to support the real-time features of the operating system. This paper discusses these issues in the context of the experience of developing support software for a particular flight project. The real-time system and application requirements of the Advanced Transport Operating Systems (ATOPS) project, operational at NASA/Langley Research Center (NASA/LaRC), prompted the evaluation of the higher level language best suited to support such a project. It was determined that of the languages available, the Shuttle language HAL/S was the reasonable choice. This language provides both the mathematically-oriented algorithms necessary to support the application user as well as the real-time constructs for utilization of a real-time operating system environment. To supply the ATOPS project with HAL/S support, however, it was necessary to develop a HAL/S software development system that operated either on the flight computers, two NORDEN PDP 11/70M's, or the large host mainframes available at LaRC, the CDC Cyber series computers. Because of the debug and verification aids available on the Cybers, the decision was made to develop the HAL/S support system on the large mainframe. The executable code generated could then be processed on either the host Cybers or the target PDPs.

To assure code reliability, the procedure adopted by the ATOPS project is that all the development and testing of flight software will be done on the host mainframes. This means that only validated code will be delivered to the flight computers. It also means that a full HAL/S test facility be resident on the Cybers. This facility includes a HAL/S syntax analyzer and code generator, a HAL/S interpreter, an assembly language translator, a flight computer simulator, an aircraft simulation, and various verification capabilities. The code verification process is an iterative one with each step being verified before proceeding to the next (see Fig. 1). The code is developed, debugged, and tested using the HAL/S interpreter. The next step is to execute the flight code on a PDP computer simulator. These results must agree with the interpreter results. To confirm that the control commands are correct requires a response from the aircraft. Combining an aircraft simulation with the computer simulator provides this closed-loop processing. When the verification process is completed, then and only then is the flight code ready to be taken to the aircraft. This procedure requires an elaborate HAL/S test facility but the cost effectiveness is evident when the cost of flying the research airplane is compared to the cost associated with debugging the code on the Cyber computers. Also, the availability of the flight hardware is no longer a factor for software development and testing. The HAL/S facility on the Cybers is available not only to the ATOPS users but to other projects with similar needs.

This paper will discuss the methodologies used to design, develop, and implement such a software support system; the management and validation techniques used in the development and the maintenance of the software; and an overview of each of the utilities required to support the real-time HAL/S operating environment. This software support system provides a production level system for the HAL/S application user that consists of the following processors:

1. A real-time HAL/S interpreter for execution on the Cybers;
2. A translator that produces assembly code for the PDP computer;
3. A real-time interpretive computer simulator (ICS) for the PDP 11/70 for open-loop testing; and
4. An aircraft simulation for closed-loop testing.

*Aero-Space Technologist, Flight Software and Graphics Branch, Analysis and Computation Div.

These primary processors are supported by:

1. A code generator that produces intermediate code that may be directed to the host or the target computers;

2. A simulated PDP assembler and loader to produce an ICS load module;

3. An instrumentation level for execution control of the ICS; and

4. A validation suite of HAL/S programs to insure system integrity.

With this total HAL/S development system, the ATOPS flight project has a software support facility that enables a HAL/S application program to be developed and executed on the Cyber computers in an interpretive mode. In addition, object code can be generated that can be executed on either the PDP flight computers or on the PDP 11/70 ICS that is resident on the Cybers. The ICS supports the real-time features of the PDP operating system and provides the visibility and control of the real-time processing not possible on the actual computer. The combination of the computer simulator and the aircraft simulation supports both open-loop and closed-loop integrated testing. During the development of this software project, it became apparent that the only way to effectively manage a software effort of this size is with a good configuration control mechanism and the early establishment of the intercommunication links between the many processors that must interface with one another.

This production level development system provides a separate verification and validation mechanism for flight systems with a real-time debug facility that is unmatched by standard system supplied software. Although this system represents a significant system software undertaking, it has enabled the real-time applications software to be developed, debugged, and integrated much more effectively and virtually unaffected by delays in hardware delivery and inaccessibility of the actual flight computers. Much that was learned in developing the support software for this application is applicable to other software projects facing similar needs.

Discussion

The NASA/LaRC ATOPS flight project is a research activity that demonstrates and evaluates flight procedures involved with improving approach and landing efficiency. The project is primarily concerned with identifying airborne systems needed for efficient operations in high-density terminal areas. The results of this research will include reduced approach and landing accidents, reduced weather restrictions, increased productivity for air traffic controllers, and a saving of fuel through the use of more efficient air terminal area techniques. The ATOPS airplane is a Boeing 737-100 that has been equipped with a special research flight deck located about 20 feet aft of the stand flight deck. An extensive array of electronic equipment and data recording systems is installed to support the flight research (see Fig. 2). The experiments are flown from the aft flight deck using advanced electronic displays and automatic control systems with two safety pilots in the front flight deck assuring flight safety.

The flight safety aspects of the ATOPS operations require that the experimental code be reliable, easily maintained, and expandable. These requirements are more easily met with a high level language than with code written in assembly language. With the upgrade of the flight computers to a pair of PDP 11/70M's, a major rewrite of the flight executive and application software was required. The safety demands combined with the rewrite of the software was the impetus for the reevaluation of the high level language that would best satisfy the needs of the project. HAL/S was selected because of the real-time and scientific capabilities it affords the system and application user. The decision to implement the language on the CDC Cyber computers, because of the availability and capability of the mainframes, somewhat determined both the elements of and the design of the HAL/S system. The code generated on the Cybers would be directed toward the PDP 11/70 target computer, but a fully developed test facility would also reside on the Cybers. This system would support the applications user with a HAL/S subset that was determined by the requirements of the ATOPS experiments (1, 2) (see Fig. 3). Because of the real-time features of HAL/S, the ATOPS flight executive would be implemented in the high level language instead of the target assembly language as has traditionally been the practice (3). Again, a real-time subset of HAL/S was implemented that allowed the needs of ATOPS to be satisfied. The HAL/S language is described in Ref. 4 and 5.

The design of the HAL/S system as implemented on the Cybers includes several separate processors. The combination of these processors provides the user with a HAL/S interpreter, a translator that produces code for the target computer, and a software simulator for the PDP 11/70 that executes the PDP assembly code on the Cybers. Most of the system processors are written in Pascal. Pascal was chosen as the system implementation language primarily because of the data type specification and record capabilities. The language also lends itself well to structured, modular programming. These features are most important when designing, developing, and debugging processors that will be implemented by more than one programmer and which are designed to be only one of several interacting processors. Phased releases of the system were possible because of the modular design of the processors. The orderliness of the releases was managed under the configuration control mechanism that was applied to the total software support system. Each of these processors and their related functions will be discussed in the following sections.

HAL/S Interpreter

The interpreter consists of three processors: (1) a HAL/S syntax analyzer (front-end) that produces the intermediate language HALMAT; (2) a HALMAT to PCODE generator that produces a machine independent, assembly type language PCODE; and (3) a PCODE interpreter that executes on the Cybers, producing final results for the HAL/S application program (see Fig. 4).

The HAL/S syntax analyzer performs a syntactic and semantic analysis of the HAL/S source. The output is object code written in the intermediate language HALMAT. HALMAT is a machine independent representation of the HAL/S source being compiled. This compiler front-end also provides the necessary facilities for separate compilation of HAL/S modules. The HAL/S front-end was originally developed by Intermetrics, Inc. for the IBM S/360 and was written in XPL (6).

The HALMAT to PCODE processor is a code generator that takes the HALMAT code and translates it into a PCODE object file. PCODE is a modified version of the Pascal intermediate language and is designed for a virtual stack machine. Being machine independent it can easily be targetted to various computers. On the Cybers, the PCODE file is executed by the PCODE interpreter. For separately compiled modules, a linking process takes place before the interpreter phase. A run-time library supports the interpreter to provide the mathematical functions required for the applications. The real-time operating environment implemented on the PCODE interpreter supports the real-time features of the PDP RSX-11 operating system. This implementation supports the scheduling of tasks with a run-time priority and an optional repeat cycle. The task interrupts are handled through wait states and event flags. The termination or cancellation of a task is provided for an orderly closeout of cyclic tasks. The real-time process will be discussed further in the Real-Time Operating Environment Section.

The PCODE interpreter supplies the HAL/S user with meaningful error diagnostics for run-time errors as well as a post mortem dump with function and procedure trace back.

PCODE to MACRO-11 Translator

The PCODE to MACRO-11 translator takes the output from the PCODE generator and translates it into MACRO-11 assembly code for the target PDP-11. This processor was designed and implemented to reflect the requirements of the ATOPS flight project and the capabilities of the RSX-11 operating system. It supports separate compilation, the ATOPS HAL/S subset, and translates the HAL/S real-time features to corresponding RSX-11 executive calls. To produce more efficient flight code, the translator has incorporated a register allocation scheme and some operational optimization.

The MACRO-11 assembly code can be down-linked from the Cyber to the PDP-11 where it will be assembled, taskbuilt, and executed on the flight computer. The final results produced on the PDP-11 and by the HAL/S interpreter on the Cyber must agree (see Fig. 5). Instead of down-linking to the PDP, the MACRO-11 code can be executed on the Cyber by a PDP interpretive computer simulator.

Interpretive Computer Simulator

To provide a full HAL/S test facility on the Cyber requires the capability of being able to execute PDP-11 assembly code on the host computer. To accomplish this, an interpretive computer simulator (ICS) for the PDP 11/70 was implemented on the Cybers. The ICS functionally models, in software, the design and action of the PDP-11/70 hardware (7). This modeling takes the form of a many-to-one instruction simulation which interpretively implements the processor instruction set, instruction timings, exception conditions, memory management specifications, register allocation, addressing modes, word lengths, and data representation (8).

The PDP-11 ICS that resides on the Cybers also simulates the real-time environment of the RSX-11 operating system (9). With an ICS the visibility and control necessary to develop and debug a real-time problem is possible without modifying the operating environment. Because any monitoring of a real-time process alters the system state, debugging a real-time application program is extremely difficult and time consuming on the actual computer.

Several other processors are required to support the ICS (see Fig. 6). The first is a MACRO-11 cross-assembler. This cross-assembler takes the MACRO-11 object file generated by the translator and produces PDP machine object code. The assembler in turn is supported by a MACRO library for the real-time RSX-11 executive system calls.

The next processor is a task builder that generally models the RSX-11 task builder (10). The object code files generated by the cross-assembler are specified as input to the task builder and are combined into a single task image output file. The task builder supports the separate compilation of program modules by linking each named object module, resolving global references and references to the supporting run-time library, and finally produces a single task image that is ready to be installed and executed. A subset of the RSX-11 task builder switches and options have been included in the HAL/S task builder to support the ATOPS applications. The switches may be used to generate a cross-reference, a map, or a trace of the task image. The possible options include specifying partition areas, task priorities, and shared regions.

To model the RSX-11 operating system the ICS support system has an installer that makes a specific task image known to the operating environment. It is the installer that manipulates main memory by installing tasks and common areas in particular partitions or memory locations. The layout of memory is contained in the map generated by the installer.

It is the load map and memory file created by the installer that is input to the PDP-11 interpretive computer simulator. The memory file is the HAL/S source program that has been translated to PDP-11 assembly language, assembled, and linked producing an object file that can be executed on the Cybers by the ICS. This execution produces the same results as if executed on the target PDP-11/70. The PDP-11 ICS models the design, action, and timings of the target computer and simulates the RSX-11 real-time operating system environment. This simulation provides for the scheduling of tasks with time constraints for cyclic scheduling. The scheduling of tasks is based on priorities with the higher the number, the greater the priority. The priorities are specified at task build time but may be dynamically altered when the task is scheduled for execution. The ICS

provides for intertask and intratask communications through global and local event flags, respectively. Event flags are also used to control task execution by putting a task in a wait state until a predetermined event takes place. To assure an orderly shutdown of the system, all scheduled tasks are terminated with the cyclic tasks being cancelled first.

Run-Time Library

A run-time library is required to provide the mathematical and I/O support for the ATOPS application programs. In addition to the basis arithmetic and trigometric functions, the library provides for the HAL/S vector/matrix operations. For example, the vector/matrix routines include vector-matrix add and subtract, vector cross product, vector dot product, vector-matrix divide and multiply by a constant, vector times matrix, and vector magnitude. The I/O routines simulate the I/O functions of the PDP 11/70 hardware. This library is resident on the Cyber host with the mathematical routines also supporting the PDP 11/70 system. On the Cybers the library supports the application programs being executed by the ICS, and on the PDP it is the resident library that is linked to the executing task at task build time.

Real-Time Operating Environment

The HAL/S real-time operating environment that supports the HAL/S interpreter and the PDP-11 ICS are both designed to model the RSX-11 real-time operating system. The real-time environment is built around significant events, event flags, and system traps. The RSX-11 executive directives that are incorporated into this system are for task execution control, priority of a task, the setting and clearing of event flags, and the declaration of an event (11).

The task execution control directives principally deal with the starting and stopping of a task which will result in a change in the state of the task. States of the task are inactive, blocked, waitready, waitsuspended, ready, suspended, or running. A task is inactive if it has not been scheduled to execute. This is the initial state of all tasks. Once a task has been scheduled for execution, its status changes to waitready, ready, or running. A waitready task cannot execute until a time delay has elapsed. This is for a cyclic task that requires a mark time period before and/or between processing. A ready task can execute immediately but, because of the executing of a higher priority task, it goes into a wait state. The task actually executing is in the running state. If a task with a higher priority is scheduled or becomes ready for execution, the currently executing task will be suspended and the higher priority task will be put in the running state. If the running task suspends itself by going into a wait state, the task state becomes waitsuspended. A task that is waiting for an event to occur is blocked. The four states--waitready, waitsuspended, ready, and suspended--are used to handle the cyclic nature of some of the tasks. When a cyclic task begins processing, a reschedule is requested based on some time delay. These four states are required to distinguish between cyclic and non-cyclic tasks.

The priority of a task is established at task build time. Because of the dynamic aspect of a real-time system, it may be desirable to change the priority when the task is scheduled or during the processing activity. This is accomplished with an alter priority directive.

Intertask and intratask communications are best handled through event flags and significant events. A significant event is a change that causes the system state to be reevaluated and the eligibility of all active tasks to run to be reassessed. These significant events include I/O completion, task termination or cancellation, task priority change, elapse of a mark time, and execution of a declare significant event directive. Event flags are a means by which tasks recognize specific events. A task may associate an event flag with a particular operation. When the operation occurs, the event flag is set, activating the change to the system. There are two sets of event flags: the local event flags that are used by a task for communication internal to itself, and global event flags for communications beween tasks. A task is blocked when waiting for an event flag to be set.

Configuration Control

The real-time HAL/S support system consists of ten separate processors with a composite of more than 90,000 lines of code. The source code for these processors including several support libraries are written in four languages: Pascal, FORTRAN, COMPASS, and MACRO-11. To adequately manage and develop a software project of this size while maintaining the reliability and useability of the system required that a configuration control mechanism be exercised over the entire system. This control assured the project management and the system users that system integrity would be maintained during the development phase as well as in the production phase of the project.

During the development phase, it was necessary to devise a management technique that provided for timely system releases. These system releases were not based on a predetermined schedule but were in response to the user needs and module completions. The releases were for the purpose of adding new processors to the system, incorporating new features to existing processors, and correcting errors in the current production system. These releases were managed and maintained through configuration control. Before a system was released to the users, a complete validation of each processor in the system was exercised. The validation process consisted of testing each processor with a set of test programs developed for this purpose (see Fig. 7). The same validation process is followed whether the new release contained modifications to one processor or to all ten. Since each processor except the compiler front-end, is dependent on another processor, the validation starts with the front-end and tests both the interpreter path and the ICS path until the two sets of test results match.

The validation suite is made up of in excess of 90 HAL/S programs designed to test the HAL/S features implemented. The programs are self-checking, and normally only a TEST CASE XX PASSED comment is reported. In case of errors, intermediate results are produced. The Interactive

Software Invocation System (ISIS) installed on the Cyber computers provides a five-level hierarchical naming scheme for files that is used to manage the HAL/S support software and the validation suite data bases (12). The ISIS command language feature was used to automate the validation process.

Aircraft Simulation

With an ICS, flight code can be developed, debugged, and tested before ever being put on the flight computers. This allows open-loop and module integration testing but does not verify that the aircraft response is as expected. This verification process is known as closed-loop testing. To provide a closed-loop test environment, an aircraft simulation was developed and interfaced to the ICS (see Fig. 8). This facility means aircraft characteristics can be studied, algorithms tested, and new experiments evaluated before flight.

The aircraft simulator receives control commands from the ICS at a regular time interval of 10 milliseconds, performs the aerodynamics, and sends the resulting actions back to the ICS. This closed-loop simulation process on the Cybers verifies that the action of the control laws and guidance procedures have the expected results before the code is actually flown.

Instrumentation that controls execution of the ICS allows the actions of the aircraft simulation and the flight code commands to be dynamically monitored. This instrumentation includes a checkpoint/restart capability, the dumping and patching of memory locations and register values, an instruction trace within address bounds, the control of graphic outputs, and gives detailed statistics about the systems performance. The timing statistics provided are crucial in verifying that the time-critical demands of the experiment can be met before taking the code to the airplane.

Concluding Remarks

The introduction of operating systems and high level languages that support a real-time operating environment have resulted in several changes in the approach to flight projects. Traditionally, the flight application code and the flight executive or system scheduler has been written in assembly language. Upgrading the ATOPS flight computers to a pair of NORDEN PDP 11/70M's computers, the supporting RSX-11 operating system, and the need for reliable and maintainable code prompted the use of the high level language HAL/S as the language for both the flight system and application.

Using the high level language HAL/S for both the flight system and application software, the verification aids, the simulation of a real-time operating environment, and the strict configuration control of the development and test facility has assured that the flight code will be more reliable, maintainable, and expandable. The flight safety aspects of the ATOPS operations require that the flight code be tested extensively and a validation process be exercised before flight. The research aspect of the program requires that modifications and extensions to the system be easily incorporated. The HAL/S facility satisfies all these requirements. Even though the implementation of the HAL/S system was a significant software undertaking, the short term and long term benefits to the project more than justify the associated cost.

The HAL/S system as implemented includes processors for executing HAL/S in an interpretive mode on the Cybers. Assembly code generated for the target machine can be shipped to the PDP for execution or be executed on the Cybers by the PDP simulator. The ICS can execute code in open-loop mode or in closed-loop mode when interfaced to an aircraft simulation (see Fig. 9).

Both the interpreter and ICS systems have a HAL/S real-time operating environment to suport the scheduling and control of the flight application tasks that have time critical requirements. This environment models the real-time features of the target RSX-11M operating system. Having this facility means the same code that flys can be first tested on the Cybers with the increased capability and availability of the large mainframes. During the debug phase, the ICS provides the visibility and control of the real-time processing that is virtually impossible on the actual flight computers.

The real-time HAL/S facility resident on the Cyber host is unmatched in the development and testing features it offers the application users. Having this system allowed for the development of the flight code to be completed and integrated module testing under way before the delivery of the flight hardware. Final closed-loop integration testing was done on the Cybers before code was transferred to the flight computers. The flight application software development was able to meet the scheduled delivery date due primarily to the HAL/S facility on the Cybers. The delivery and availability of the flight hardware was not a determining factor in being able to continue the development and testing of the flight code.

Currently, there is interest being shown in HAL/S as the language for several new flight projects under development at the Center and at other agencies. It is felt that this facility can accommodate these projects with little modifications to the total system. The modularity of each individual processor allows for extension or modifications with minimum impact to any other part of the system.

For others considering such a software effort, the following recommendations should be given careful attention.

1. Make sure the software product is suited to the project. The HAL/S language was designed for engineering problems with time-critical constraints. The HAL/S subset implemented reflects both the ATOPS project requirements and the flight computer operating system.

2. Implement the software system under strict configuration control and against an implementation schedule. Out of necessity, the HAL/S test facility was being developed while supporting application code development. Strict software management was the only way to successfully accomplish this task.

3. Consider obstacles and their relative importance to the total effort. Several incompatibilities existed between the host and target computers but the advantages of providing the test system on the Cybers justified the Cyber implementation.

4. Consider the short term and long term benefits of the undertaking. The HAL/S facility is currently the test base for the ATOPS project but is available for use by other projects and installations. One of the most notable long term benefits is the acquired in-house expertise. This software expertise will be invaluable in developing and evaluating future software support systems.

Acknowledgement

It would be presumptuous to conclude that an undertaking of this magnitude could be accomplished except through a team effort. The author would like to acknowledge the people whose diligence and relentless support, over an extended period of time, made the development and delivery of this system possible. The effort they extended, even in the face of major obstacles, made it possible to produce the system support software for the project upgrade that allowed the application development and testing to proceed in a timely manner and meet the projected schedules.

My gratitude and appreciation goes to the members of the ATOPS System Software Team: Arthur L. Gooden, Marie S. Noland, Karen E. Spriggs, and Ralph W. Will of NASA Langley Research Center; and James L. Donaldson, Karen F. Gray, Roy W. Hamm, Dana P. Hammond, and Christopher J. Slominski of Computer Sciences Corporation.

References

1. "Syntax Diagrams for the HAL/S Language," Flight Software and Graphics Branch, ACD, NASA Langley Research Center, July 1984.

2. Pratt, Terrence W., "HAL/S Formal Semantic Definition," University of Virginia Report No. UVA/528164/AMCS79/102, August 1979.

3. Smith, Rudeen S., "Designing a Priority Driven Multi-Frame Rate Flight Executive," proposed for the AIAA 23rd Aerospace Sciences Meeting, Reno, Nevada, January 14-17, 1985.

4. "HAL/S Language Specification," Intermetrics, Inc., NASA, Version IR-542, September 1980.

5. Ryer, Michael, T., "Programming in HAL/S," Intermetrics, Inc., NASA Contract NAS9-13864, September 1978.

6. "HAL/S-FC and HAL/S-360 Compiler System Program Description," Intermetrics, Inc., NASA Lyndon B. Johnson Space Center, IR-182-2, March 1977.

7. PDP-11 Processor Handbook, Digital Equipment Corp., 1981.

8. Smith, Rudeen S., "Avoiding Pitfalls in Simulating Real-Time Computer Systems," 1984 Summer Computer Simulation Conference, Boston, MA, July 23-25, 1984.

9. PDP-11 Software Handbook, Digital Equipment Corp., 1981.

10. RSX-11M/M-PLUS Task Builder Manual, Version 3.2, Digital Equipment Corp., Order No. AA-H266A-TC, 1979.

11. RSX-11M/M-PLUS Executive Reference Manual, Version 3.2, Digital Equipment Corp., Order No. AA-H265A-TC, 1979.

12. Noland, Marie S., "An Evaluation of the Interactive Software Invocation System (ISIS) for Software Development Applications," NASA TM 81924, January 1981.

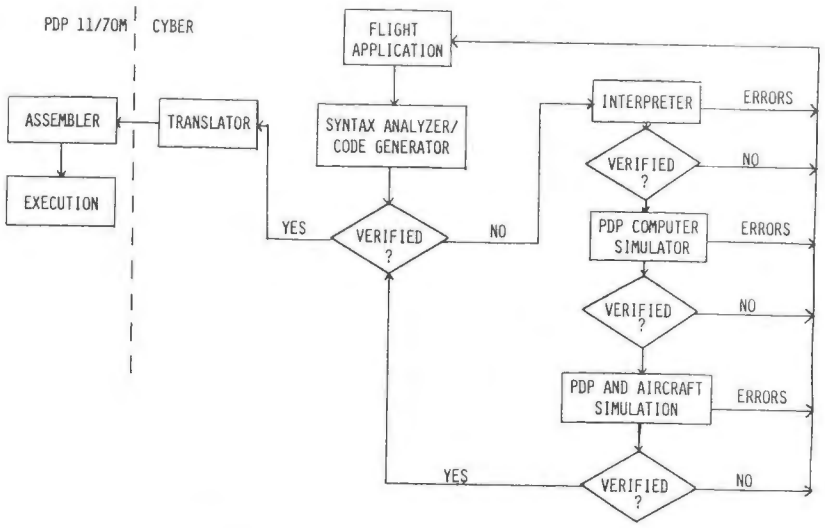


FIGURE 1: CYBER DEVELOPMENT AND TEST FACILITY FOR HAL/S

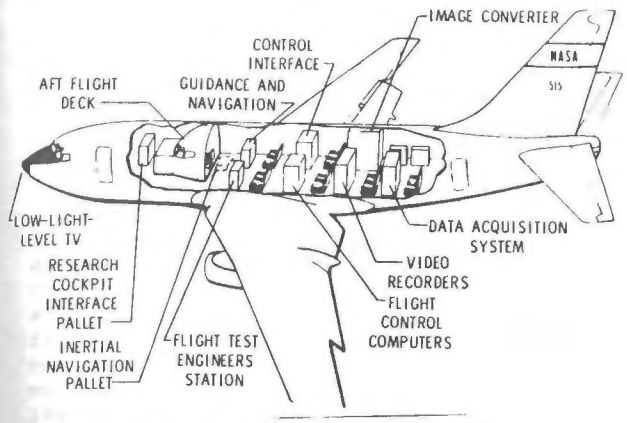


FIGURE 2: ATOPS B-737 RESEARCH AIRPLANE

PROGRAM STRUCTURE	EXPRESSIONS
PROGRAM	IF-THEN-ELSE
PROCEDURE	CASE
FUNCTION	WHILE
TASK	UNTIL
COMPOOL	REPEAT
SEPARATE COMPILATION	INTERATIVE DO FOR
	DISCRETE DO FOR
DATA SPECIFICATIONS	
INTEGER	**FUNCTIONS**
SCALAR	USER-DEFINED
CHARACTER	BUILT-IN
BIT	SHAPING
SUBBIT	
ARRAY	**REAL-TIME**
VECTOR	SCHEDULE TASK
MATRIX	PRIORITY
STRUCTURE	REPEAT EVERY
NAME	WAIT FOR
REPLACE	SET EVENT
	RESET EVENT
	CANCEL
	TERMINATE

FIGURE 3: HAL/S LANGUAGE FEATURES SUPPORTING ATOPS

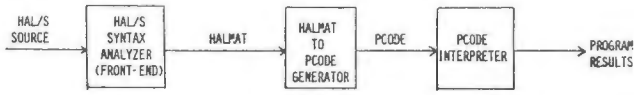


FIGURE 4: HAL/S INTERPRETER

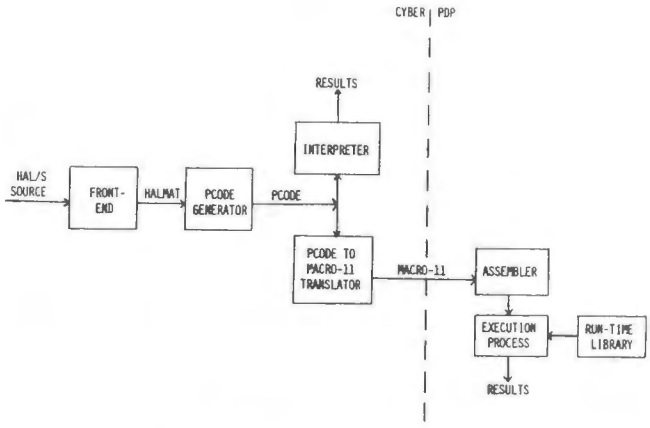


FIGURE 5: HAL/S INTERPRETER AND TRANSLATOR

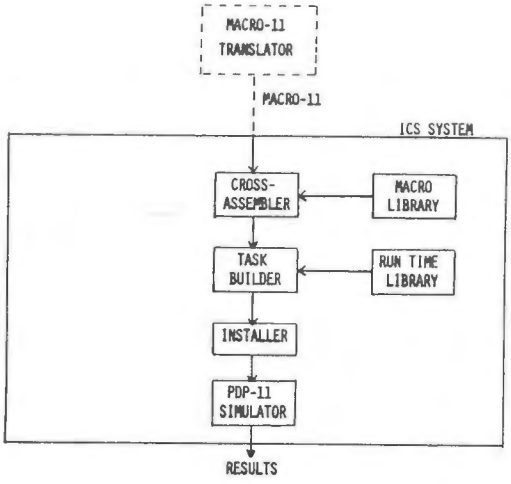


FIGURE 6: HAL/S INTERPRETIVE COMPUTER SIMULATOR SYSTEM

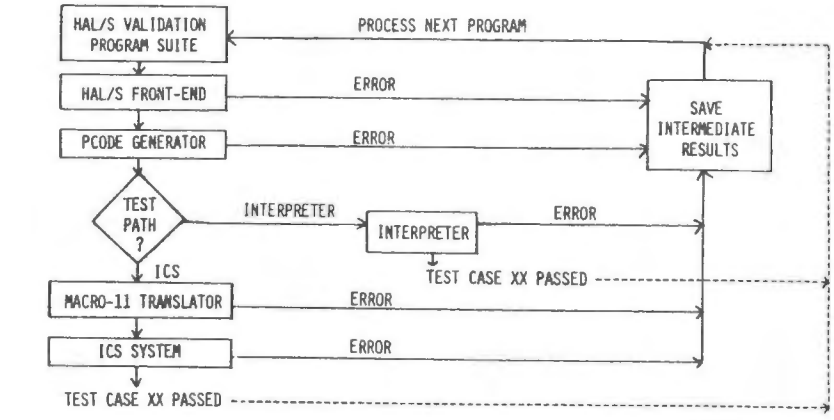


FIGURE 7: VALIDATION OF HAL/S SUPPORT SOFTWARE SYSTEM

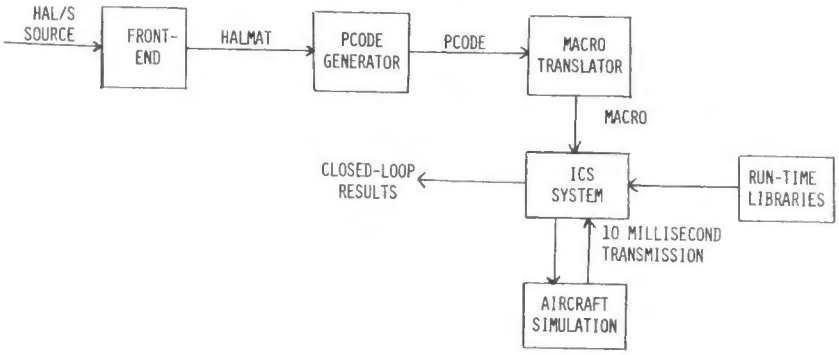


FIGURE 8: CLOSED-LOOP TEST SYSTEM

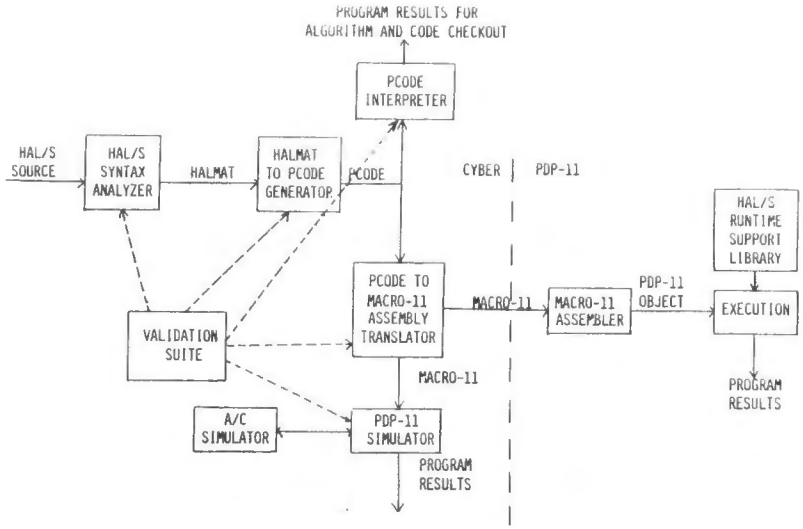


FIGURE 9: HAL/S SOFTWARE DEVELOPMENT AND TEST SYSTEM

SOFTWARE TRACEABILITY, REQUIREMENTS TESTABILITY, AND AUDITING MODEL

Joseph Sciortino (Manager Software Engineering Group)
Diane Dunning (Software Engineer)

ARINC Research Corporation
2551 Riva Road
Annapolis, Maryland 21401

Abstract

In recent years software development for both management information systems (MIS) and embedded computer systems (ECS) has suffered dramatic cost and time overruns, has resulted in user dissatisfaction, and has often yielded software that does not work in spite of extra time and money spent.(1) There are several major factors leading to these software product deficiencies:

- Original requirements that are incomplete or not validated
- Software design that is not traceable to the requirements and diverges during development
- Software code that is not maintainable because of poor enforcement of standards
- Documentation of the system that does not reflect as-built code
- Software that is insufficiently tested(2)

ARINC Research has developed a software traceability tool for reducing those risks in software development. The ARINC Research Software Traceability, Requirements Testability, and Auditing (STRATA) model is a microcomputer-based tracking system developed to support and automate the top-down structured approach to software development. It has already been applied to the environment set forth in the Department of Defense (DoD) Automated Data Systems (ADS) Documentation Standard 7935.1S and Air Force Regulations 300-12 and 300-15; it has been conceptualized to support the embedded software environment set forth in Military Standard 490 (MIL-STD-490) Specification Practices; MIL-STD-483, Configuration Management Practices; and the 800 series Air Force Regulations.

Introduction

STRATA was first used in the support of Air Force development of the Comprehensive Engine Management System (CEMS) Increment IV, a MIS for jet engine diagnosis. ARINC Research, acting as the software quality assurance contractor, verified and validated the software configuration components produced by the CEMS IV development contractor and identified validation testing requirements through application of the tool. Various life-cycle documents were produced during ADS software development:

- Functional Description
- User's Manual
- System Specification
- Operations Manual
- Program Specification
- Development Test Plan
- Data Base Specification
- Validation Test Plan

As the documents were developed, STRATA was loaded with pertinent key information for each document. In addition, data base "set" structures were loaded with key values that mapped the relationship of each document to subsequent documents in the ADS structure (see Figure 1). At the appropriate reviews during the development cycle, STRATA was used to produce several output products showing functional dependencies among documents and highlighting where the functional dependencies were not consistent or were not present. Additional, special supporting products demonstrated the effects across documents when various requirements were deleted, added, or modified.

The use of STRATA greatly assisted the independent verification and validation (IV&V) effort performed by ARINC Research engineers by automating the time-consuming and labor-intensive effort of tracing design consistencies throughout documents and verifying the functional completeness of the total design. This left the engineers with considerably more time to perform additional IV&V tasks such as identifying inconsistencies of implementation, identifying added requirements or poor programming practices, and performing software sneak analyses. Since embedded system development follows a specification process similar to that for an MIS development, it was noted that STRATA could very easily be adapted to assist in IV&V efforts performed on embedded computer systems, including those supporting digital avionics.

This paper provides an introduction to the CEMS IV MIS development, the first application in which STRATA was used to support system development in a DoD ADS 7935.1S environment. The model's structure, software, and use are described, together with the output products prototyped during the CEMS IV development. The valuable contributions of STRATA to the Air Force and the software contractor in demonstrating design inconsistencies and test schema deficiencies are also discussed. Finally, this paper briefly examines the environment of embedded systems development and the advantages of applying a tool like STRATA to the software development of digital avionics systems.

CEMS IV MIS Development

The CEMS is a system developed by the Air Force Logistics Command for implementation throughout the Air Force to satisfy engine management requirements and associated information needs. The system is being developed in four increments to facilitate implementation and reduce management burden. ARINC Research has supported various increments of CEMS since 1979. STRATA was developed to support a verification and validation effort for Increment IV.

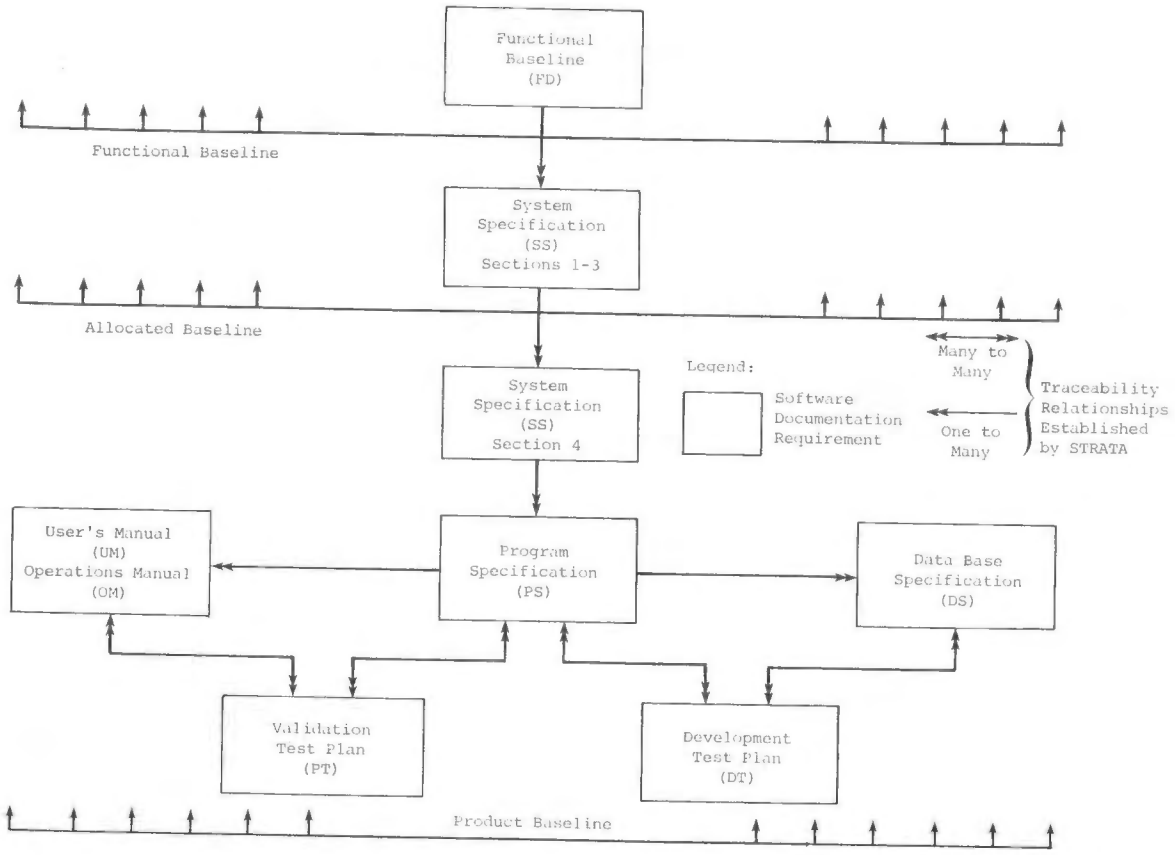


Fig. 1 The STRATA Model (Adapted to Support the DoD Standard 7935.1S, AFR 300-15, and AFR-30012 Environment)

In 1980 ARINC Research was awarded a contract for Increment IV, base level, to provide technical support to the project office at all development milestone reviews and to perform an independent verification and validation of the software contractor's final product through development and execution of a Validation Test Plan (PT). The following paragraphs describe the 7935.1S standard environment within which these development milestone reviews were accomplished

ADS Development Phases

Documentation and reviews are the primary channels of communication to be used in assessing software quality at each phase of development. Reviews must not be allowed to be ignored or their importance minimized. They must be successfully completed before further development activities are started. Too often, the governing agency does not have the time or resources to properly examine the documents and prepare for a review. Consequently, the documents are allowed to be too general, and little or no analysis is performed at the time of the review to support or contradict the requirements or design being presented. In addition, reviews are often held as passive tutorials for high-level officials instead of as active evaluations of the specification details. The software developer is then allowed to present the best picture of the work that has been accomplished at the time of any particular review.(2) When the development advances through the DoD Standard 7953.1S phases described in the succeeding paragraphs, STRATA can leverage the human resources needed for an adequate review of the software documentation. This permits more time for detailed analysis of the software design.

Before an ADS becomes operational, its development following the aforementioned standards progresses through four phases: conceptual, definition, development, and test. In each phase various documents are prepared and updated or examined to ensure that the changes set forth at the previous review were properly implemented. Successful completion of technical reviews marks the end of each phase. The phases are illustrated in Figure 2 and described below.

Conceptual Phase

The need for the ADP support is determined. Functional (mission) requirements are identified by the user and formally expressed in a Data Automation Requirement (DAR). A projected automation requirement is prepared to identify resources needed in the system life out-years. The functional manager must describe and justify the requirement for the system to execute a certain function or carry out a certain operation. Such requirements must be stated and analyzed to define any problems that will be encountered in providing, changing, or adapting a management or operational capability to meet them. After the requirements have been identified and evaluated for alternative solutions, they must be documented for management evaluation. If automation is selected as the solution, the following formal documents must be prepared for review: the DAR, the Data Project Directive, the Data Project Plan, and the Functional Description (FD). These

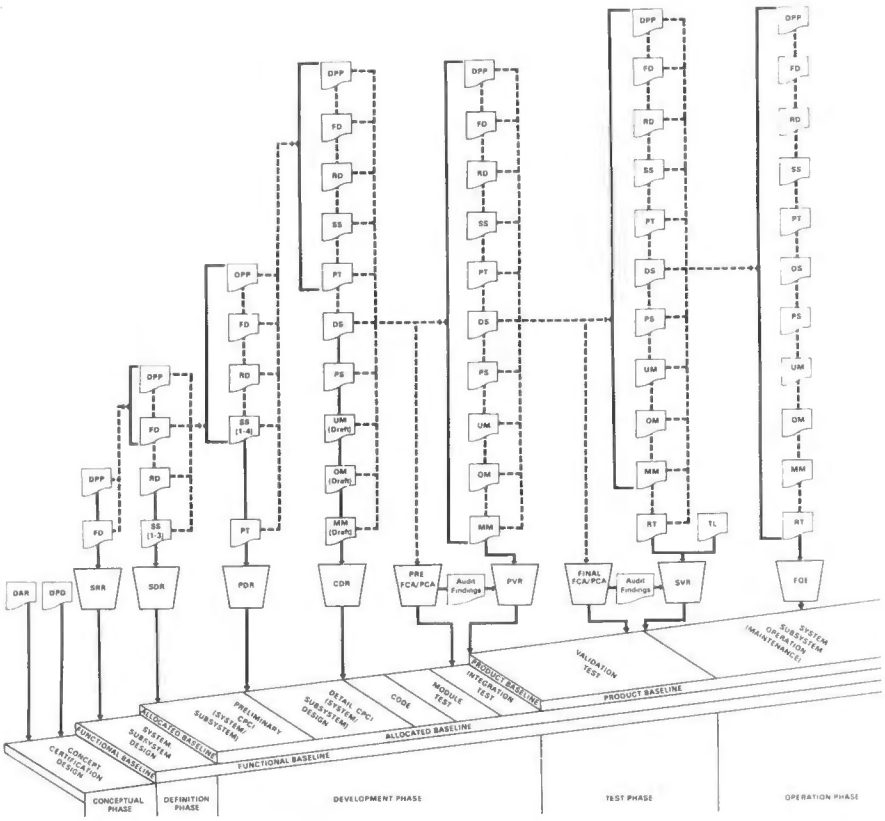


Fig. 2 ADS Development (Using DoD ADS Documentation Standard 7935.1-S and AFR 300-12, 300-15, prepared by HQ AFLC/LME August 1982)

documents are reviewed at the System Requirements Review (SRR). Successful completion of this review ends the conceptual phase.

Definition Phase

The developers must define the design requirements for the major components of the system. This effort includes developing system interface control requirements, precisely defining all functions the system must accomplish, and specifically allocating each function to a system component. The first three sections of the System Specification (SS) and the Data Requirements Document (RD) must be prepared. The System Design Review is then conducted to verify that the system design is correct and all user requirements will be met. If the system passes this review successfully, the definition phase is completed.

Development Phase

This phase is divided into three distinct efforts separated by three reviews: the Preliminary Design Review (PDR), the Critical Design Review (CDR), and the Product Verification Review (PVR). In addition, a preliminary Functional Configuration Audit (FCA)/Physical Configuration Audit (PCA) is held during this phase. The three distinct efforts, described below, are preliminary design, detailed design, and coding and testing.

Preliminary Design. The tasks to be performed in this first effort are a review of the design already called out in Sections 1 through 3 of the SS and the initiation of preliminary design through completion of Section 4, the final section of the SS. A PDR is held to validate this effort.

Detailed Design. This effort calls for the completion of the detailed functional design and the initiation of support documentation. The detailed design is an expansion of the preliminary design called out in Section 4 of the SS. It is documented consistently in the Program Specification (PS) and the Data Base Specification (DS). The support documents to be prepared include preliminary drafts of the User's Manual, Computer Operations Manual, Program Maintenance Manual, and Test Plan (PT). These efforts are validated by the CDR. When the Program Specifications and Data Base Specifications are approved during the CDR, the design of the ADS is completed.

Coding and Testing. The tasks performed during this effort include the coding, compilation, and verification of all program modules. Development testing is conducted on each coded program until complete programs are integrated and tested. All documents initiated during detailed design are updated to reflect the system after the coding and testing effort. A preliminary FCA/PCA is

held at this point to verify that the software products and documentation prepared are in agreement, that code is according to standards, and that development test results indicate that requirements will be met. After the audits have been completed, a Product Verification Review (PVR) is held to ensure that the system is ready for formal testing and to ensure that preparations for the test phase have been completed.

Test Phase

An independent test group performs validation testing. System requirements are tested, problems are corrected, and documentation is updated. In validation testing, the support documents (OM, UM, MM) are validated. A Test Analysis Report is prepared to document the results of the execution of the validation tests called out in the PT. A final FCA/PCA is then held to ensure that all documentation is updated to reflect the final configuration of the system. Following the FCA/PCA, the System Validation Review is held to determine that the system performs as originally stated in the FD and SS, and to obtain certification for operational use from the functional manager. Completion of this phase constitutes the end of the ADS development and passes the product into the operational phase.

STRATA Implementation

The STRATA model was designed to run on the CP/M-based* data base management system dBase II.** STRATA was hosted on the Superbrain+ to support the CEMS IV base-level development. STRATA has since been hosted and applied on many CP/M-based systems. The current implementation of STRATA follows the general textbook approach to a relational data base, which can be generally explained as a two-dimensional table with the following characteristics: The columns of a relation are referred to as attributes; each attribute has a distinct attribute name to distinguish it from all other attributes; each relation must also have a key composed of an attribute or a combination of attributes that can uniquely identify the row (data reach) of the relation.(3)

Model Structure

The complete structure of the model is illustrated in Figure 3. Each rectangle represents a software documentation requirement. Each software documentation requirement can be represented as a data base. These data bases (which are called relations) are structured into a relational data base by means of a relational data base management system such as dBase II. A link between the relations is established by using the unique keys of the two relations that are being associated. The following paragraphs describe in detail each relation as referenced in 7935.1S. The cross-referencing among documents as set forth in 7935.1S is also described.

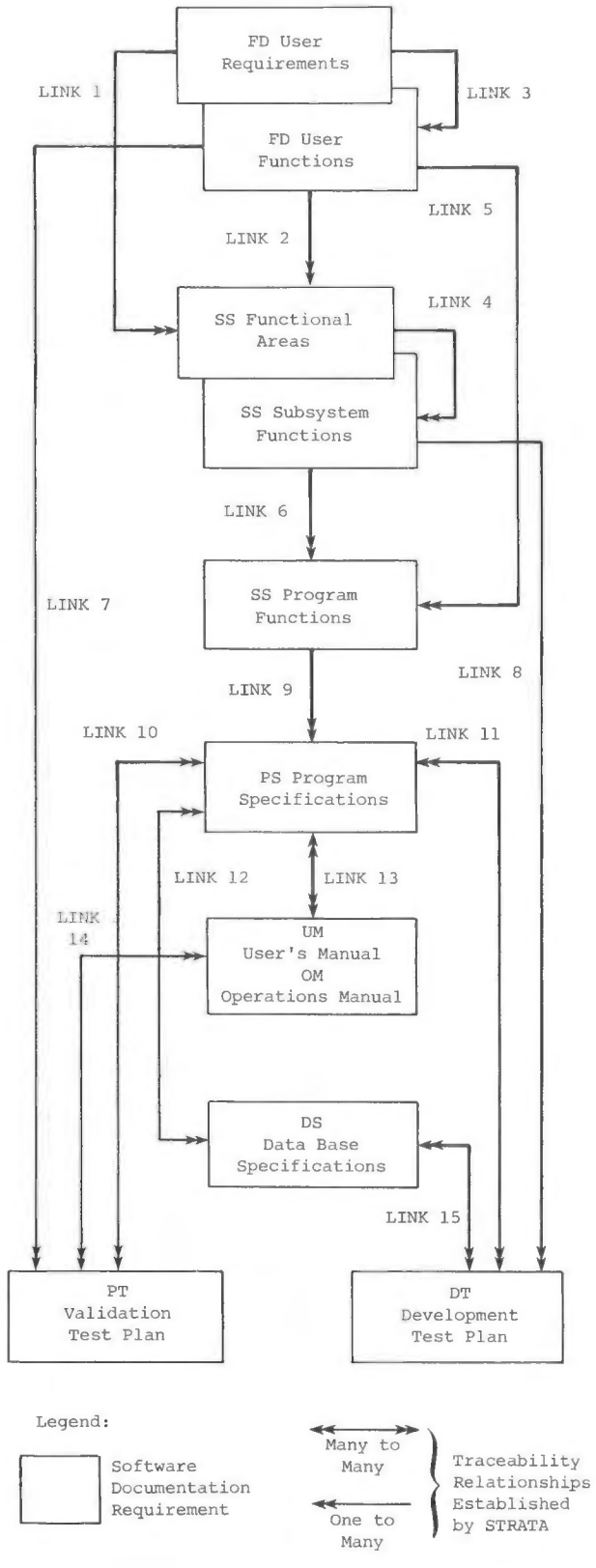


Fig. 3 STRATA Model Structure

*CP/M is a trademark of Digital Research, Inc.
**dBase II is a trademark of Aston-Tate.
+Superbrain is a trademark of Intertec Data Systems.

User Requirements Relation

The relation that describes the FD User Requirements consists of the FD Requirement Number, FD Requirement Name, Status, and Base Line Change Request (BCR) Number. The BCR number is the control document number of a request to delete or alter the user requirement. When a request to delete a requirement is approved, the status field is turned on and all processing of that record is skipped. The user requirements are the highest level of requirements in the FD. The unique key for the user requirements segment is the FD Requirement Number. This number corresponds to the paragraph number in the FD and is used to link this relation to other relations shown in Figure 3. The content and cross-referencing of this relation are described on page 3-09, paragraph 3.1, Specific Performance Requirements, of 7935.1S as follows (see Figure 3 for the LINK structure, as set off in parentheses below, to other relations): "This paragraph delineates the specific performance requirements to be satisfied by the ADS and is evolved from the system analysis. The requirements are stated in such a manner that the system functions in paragraph 3.2 (FD USER FUNCTIONS Relation through LINK 3) and the system tests necessary for implementation (PT VALIDATION TEST PROCEDURES Relation through LINK 3 and LINK 7) can be related to them."

Functional Areas Relation

The relation that describes the SS functional areas consists of the SS Area Number, SS Area Name, Status, and BCR Number. The content and cross-referencing for this relation are described on page 3-25, paragraph 2.1, System/Subsystem Description, of 7935.1S as follows: "This paragraph provides a general description of the System/Subsystem to establish a frame of reference for the remainder of the document."

User Functions Relation

The relation that describes the FD User Functions consists of the FD Function Number, FD Function Name, Status, and BCR Number. The content and cross-referencing of this relation are described in page 3-10, paragraph 3.2, System Functions, of 7935.1S as follows: "This paragraph describes the individual functions performed by the proposed ADS. The description relates the functions to the performance requirements (FD USER REQUIREMENTS Relation through LINK 3) and to the subsystem (SS SUBSYSTEM FUNCTIONS Relation through LINK 2) or computer programs (SS PROGRAM FUNCTIONS Relation through LINK 5) that will provide the functions and will show how the aggregate of these functions satisfies the specific requirements in paragraph 3.1."

Subsystem Functions Relation

The relation that describes the SS Subsystem Functions consists of the SS Function Number, SS Function Name, Status, and BCR Number. The content and cross-referencing for this relation are described on page 3-25, paragraph 2.2, System/Subsystem Functions, of 7935.1S as follows: "This paragraph describes the system/subsystem functions. There will be both qualitative and quantitative descriptions of how the system/subsystem functions will satisfy the requirements (FD USER REQUIREMENTS Relation through LINK 2 and LINK 3). Although the descriptions of the system/subsystem functions may be refined and more detailed as a result of the ongoing analysis and design, they must maintain a direct relationship to the system functions established in paragraph 3.2 of the FD (FD USER FUNCTIONS Relation through LINK 2) and be stated in such a manner that the system/subsystem environment in Section 3 can be related to them."

Program Functions Relation

The relation that describes the SS Program Functions consists of the SS Program Function Number, Program Identification, Program Name, Program Description, and Status. The content and cross-referencing for this relation are described on page 3-30, paragraph 4.4, Program Descriptions, of 7935.1S as follows: "Paragraphs 4.4.1 through 4.4.n shall provide descriptions of the program functions, related to paragraph 2.2 of the SS (SS SUBSYSTEM FUNCTIONS Relation through LINK 6) and the computer programs in the system/subsystem."

Program Specifications Relation

The relation that describes the Program Specifications (PS) consists of the Program Specification Number, Program Identification, Program Name, Logic Diagram Number, Logic Diagram Description, and Status. The content and cross-referencing for this relation are described on page 3-40, paragraph 4.5, Program Logic, of 7935.1S as follows: "This paragraph describes the logic flow of the program. Logical flow may be presented primarily in the form of charts. A narrative presentation, when appropriate, will be used to supplement charts. All charts will be keyed to high order charts in the SS (SS SUBSYSTEM FUNCTIONS Relation through LINK 9 and LINK 6) or the FD (FD USER FUNCTIONS Relation through LINK 9 and LINK 5)."

Validation Test Plan Relation

The relation that describes the Validation Test Plan consists of the PT Test Number, PT Test Name, PT Test Area, and Status. The content and cross-referencing for this relation are defined in Attachment 2 of Air Force Regulation 300-15 as follows: "The Test Plan (PT) is based on the requirements of the FD (FD USER FUNCTIONS Relation through LINK 7) and the SS (SS FUNCTIONAL AREAS Relation through LINK 7 and LINK 2), and specifies the test conditions for acceptance testing of a computer program."

Development Test Plan Relation

The relation that describes the Development Test Plan consists of the Development Test Number, Development Test Name, Development Test Description, and Status. The content and cross-referencing for this relation are defined in Attachment 2 of AFR 300-15 as follows: "This document specifies the method and content for developmental testing from the lowest compilable level (PROGRAM SPECIFICATIONS Relation through LINK 11) up through the complete computer configuration item (SS SUBSYSTEM FUNCTIONS Relation through LINK 8)."

Data Base Specification Relation

The relation that describes the DS consists of the Record Name, Local Name, Page Number, Record Description, and Status. The content and cross-referencing for this relation are described on page 3-50, paragraph 3.4, Records, of 7935.1S as follows: "This paragraph describes the major programs accessing, creating, or modifying the data base records as well as the system utility used to accomplish this operation (PROGRAM SPECIFICATIONS Relation through LINK 12)."

User's Manual and Operations Manual Relation

The relation that describes the User's Manual and Operations Manual consists of the Command Number, Command Line, Page Number, and Status. This relation was included to facilitate the automation of validation test procedures and to ensure that all program elements are documented in user and operations procedures. There is no cross-referencing specifically called out in any regulation.

Output Products

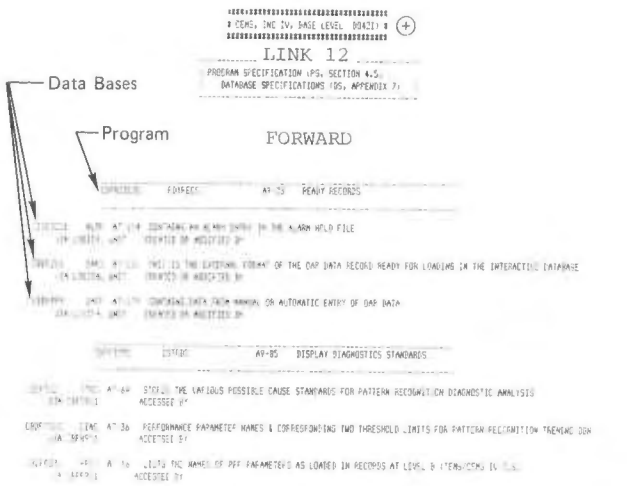
There are several standard products that can be produced by use of the STRATA model as well as several special products that the user can define within the LINK structure. For every LINK shown in Figure 3, there is an associated output product, which the user can produce by keying in the appropriate LINK number. The user can also employ the status field in each of the segment files to produce specified outputs and can employ the dBase II report generator to list the segment files in sorted or "on condition" order.

Standard Output Products

The standard output products can be accessed through a menu-driven utility in the STRATA software or through a direct print call. The heading at the top of the output products displays the relationship between the two relations and the LINK structure. Each of these output products can be called upon throughout the ADS life cycle to ensure synchronization among the various life-cycle documents.

Because each of these relations is representative of the top-down approach set forth by the 7935.1S ADS documentation standards, most of the relations represent a one-to-many relationship. That is, for every FD User Requirement there are many FD User Functions, and for every FD User Function there are many SS Program Functions. However, when there is a many-to-many relationship between segments, the output products can be produced both forward and backwards by a simple sorting of the LINK files.

Two examples are the many-to-many relationship between the data bases and the programs and the relationship between the validation test procedures and the user and operator commands. Each program can access and write to several data bases, and each data base can be accessed, created, or modified by many different programs. LINK 12 can be sorted by a program key to produce a LINK 12 Forward output product for use during the formation and review of the program specification (PS) to ensure its consistency with the data base specification. Figure 4 presents an excerpt from a LINK 12 Forward output product. This product shows, by program, which data bases the PS reflects as being accessed, created, or modified. Similarly, LINK 12 can be sorted by a data base key to produce a LINK 12 Backwards output product, which can be used during the formation and review of the data base specification to ensure its consistency with the program specification. Specifically, this product can be used to audit the data base specification and verify that for every data base documented, the description contains all programs specified in the PS as accessing, creating, or modifying it.



⊕ This material furnished by the U.S. Government.

Fig. 4 LINK 12 Forward

Validation test procedures specify which user and operator commands must be executed for each validation test. Again, this output product can represent a many-to-many relationship. LINK 14 can be sorted by user or operator command to produce LINK 14 Forward, an output product that shows the validation tests that exercise every user or operator command. Alternatively, LINK 14 can be sorted by validation test to produce LINK 14 Backwards, an output product that shows which commands are executed during the performance of every validation test. These products are useful during the validation test period to develop dependencies for test executions involving commands that must be executed overnight or over a period of time, such as file manipulator or purge commands. If a command line fails or cannot be performed, the LINK 14 output products will immediately recognize other tests that will be affected.

Special Output Products

The status fields in each of the segment files can be used in any number of ways to produce specialized output products, including the following examples. The user functions that involve external interfaces can be flagged and used to produce an output product that displays all

programs and all data bases executed or called during an external transfer of data. That output product can then be produced for all external interfaces or for a specified interface within an external system.

In addition, the status flag can be used during an output product run to flag the segments that participated in the LINK structure. For example, during a LINK 11 run, the programs and all data bases called by those programs that were called out in a development test procedure can be flagged. At the completion of that run, a special product can list all the programs and data bases whose status flags were not turned on. That product would display all programs and data bases not exercised during development testing or not correctly documented as being exercised in the Development Test Plan. The results of such a run would be useful in determining the earnestness of the software developers in testing their system. It would also provide the governing agency with sufficient evidence to postpone a review because the system had not been thoroughly tested, thus saving the agency time and travel money.

Application to ECS Environment

The ECS engineering and test flow is depicted in Figure 5. There are several differences, primarily in terminology, between this process (MIL-STD-1521A) and the ADS process required by 7935.1S. These differences can mostly be attributed to the fact that MIL-STD-1521A specifies a complete system testing and integration, encompassing hardware, software, equipments, and other elements. The reviews, however, are the same -- SRR, SDR, PDR, CDR. For embedded computer program development, they address the same level of detailed specification at each review as an ADS. In addition, the baselines are established in much the same way. The specifications required for ECS by MIL-STD-490 are represented by a STRATA model adaptation in Figure 6. Additional relations can be added to meet the ECS developer's needs. STRATA output products can then be employed to support the design reviews called out in Figure 5 in much the same way the reviews are supported in an ADS environment.

Conclusion

With the increased availability of processing power at a lower cost, the task of searching through development documentation for consistency and completeness can be accomplished most efficiently and economically through automated methods. Both the developer and the project manager should employ tools such as STRATA so that they can spend more time on more significant design details. In particular, the application of this concept can significantly contribute to development and independent verification and validation efforts for digital avionics systems.

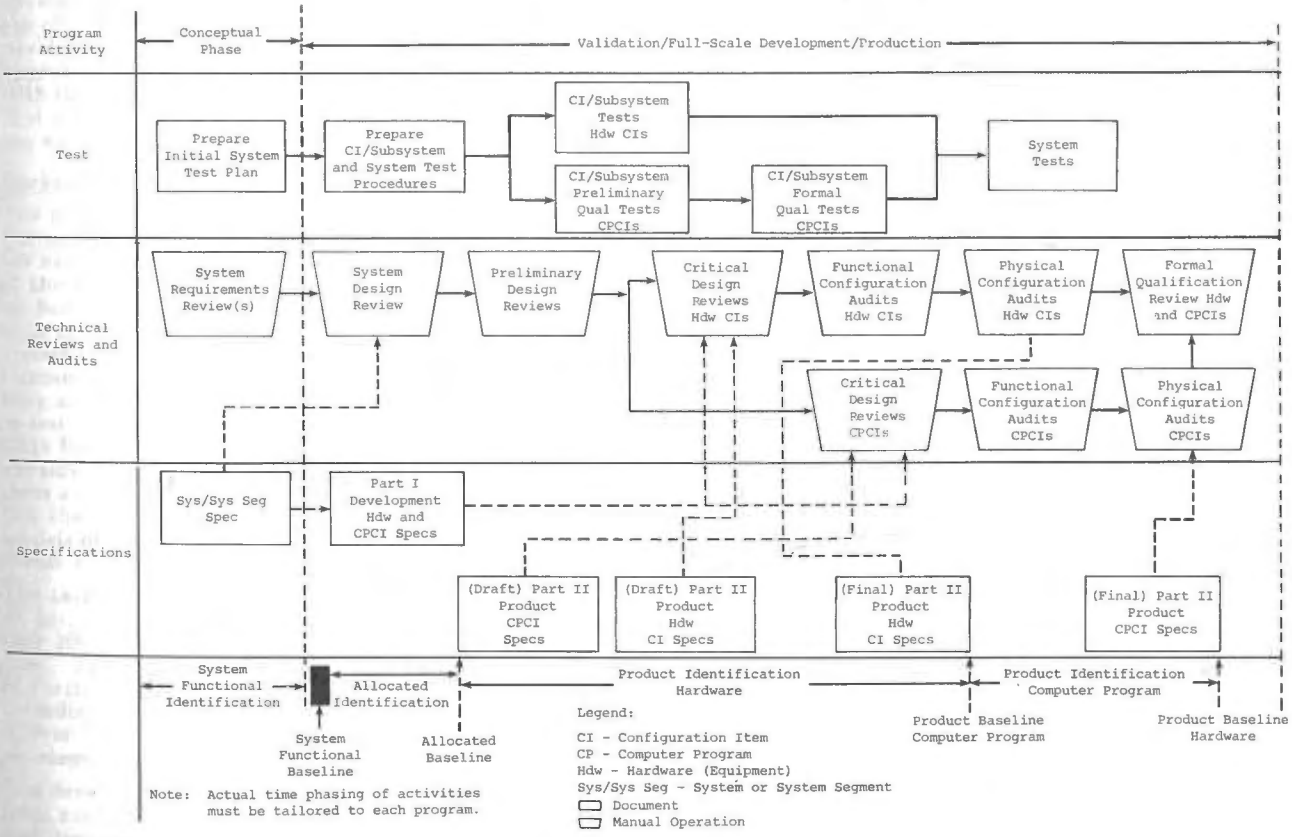


Fig. 5 Engineering and Test Flow (taken from MIL-STD-1521A)

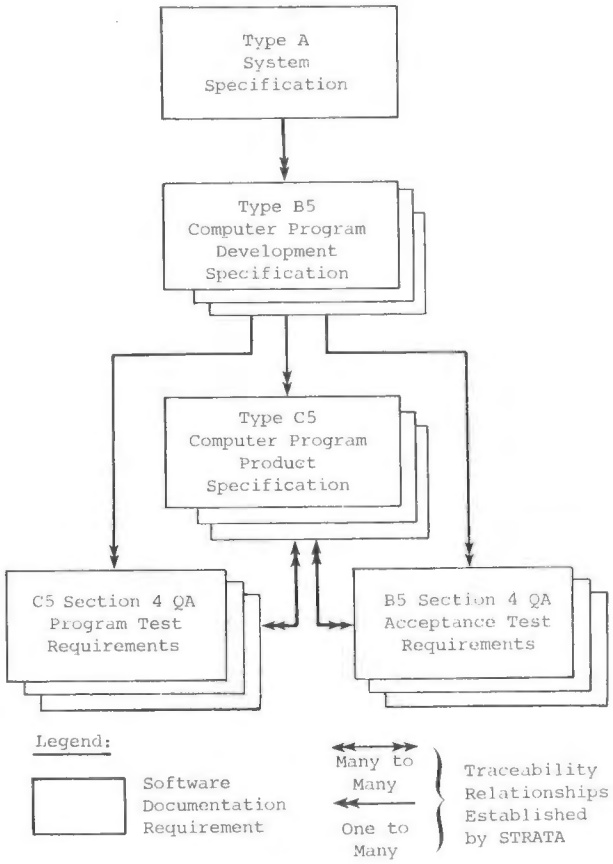


Fig. 6 The STRATA Model (Adapted to Support the MIL-STD-490, MIL-STD-483, and MIL-STD-1521A Environment)

References

1. "Contracting for Computer Software Development - Serious Problems Require Management Attention to Avoid Wasting Additional Missions," U.S. General Accounting Office Report No. FGMSD-80-4, November 9, 1979.

2. Major General Jerry Max Bunyard, USA, and James Mike Coward, "Today's Risks in Software Development - Can They Be Significantly Reduced" The Journal of Defense Systems Acquisition Management, Autumn 1982.

3. David Kroenke, Database Processing - Fundamentals, Modeling, Applications, Chicago, Science Research Associates, Inc., 1977.

DEVELOPMENT TOOLS – CASE STUDY FOR LARGE SYSTEMS

84-2635

Katherine Hornbach

Software Tools Section Manager
Lear Siegler/Instrument Division
Grand Rapids, Michigan

Abstract

Software development tools can be an important aid in controlling the complexity of large digital avionics systems. This paper describes the successful application of modern software tools to the development of the Flight Management Computer System for the Boeing 737-300 aircraft. Tools were used to increase productivity and quality during the entire software life cycle. Source code management tools provided thorough, ongoing configuration management of code. Static analysis and path coverage of the source aided in meeting stringent verification requirements. Fourth generation language techniques were used to produce many of the tools cost-effectively; and text formatting tools were used to increase documentation productivity. These and other tools, some in use for the first time, helped in the production of a high-quality software product on a very tight schedule. Special attention was paid to the problems of scaling up tools for use on a large project, and to careful tailoring of the tools to correspond to the specific ways the project chose to structure software development.

Introduction

The complexity of large digital avionics systems has reached the point where effective automated support for software development is essential. Due to advances in software technology, a large variety of productivity tools are now available. Despite this, many shops are still developing software little differently than they did ten years ago. This paper describes the successful application of modern software tools to the development of a large digital avionics system. Several areas of tool use, which are applicable to most large software development projects, are discussed in detail. Our initial apprehensions and hopes in using these tools are examined with the retrospective of two years experience, with the goal that other projects considering tool use can be reassured by our successes and avoid our mistakes.

Background

The project described in this paper is a Flight Management Computer System (FMCS) for the new Boeing 737-300. As the name implies, the FMCS manages all aspects of the flight of the aircraft. It interfaces with all other major systems on board, including the auto-pilot and auto-throttle, and with the pilot and first officer. Major functions which the system is required to perform are: navigation, aircraft performance calculations, flight path prediction, vertical guidance and steering, lateral guidance and steering, and built-in-test. It also contains two large data bases: the Navigation Data Base, which contains all the information on waypoints, navaids, airports, airways, standard routes, altitude restrictions and other data for the area covered by the user airline; and the Performance Data Base, which contains numerical models of the aerodynamics of the aircraft and of the engine thrust characteristics [1].

The target computer contains three concurrently-operating 16 bit, custom designed processers, employing a total of over 200K words of code and 100K to 200K words of data base. The source consists of over 100,000 executable lines of Fortran-77 and assembler, divided into 2,000 subroutines. Including comments and common blocks, the source code is over 500,000 lines. Over 80 people were involved in the development and testing of the software.

The development time available was slightly under two years, from contract award to start of FAA certification – a very ambitious schedule for a project of this size.

Development Methodology

On a prior project, requirements and design had been specified in large, textual documents. In addition, the time spent on those phases had often been slighted, since code production was a primary means of measuring progress. This was workable on lesser projects, but it had become apparent that something more was needed for a project of this magnitude. For these reasons, versions of Yourdon's *Structured Analysis* (SA) and *Structured Design* (SD)[2][3], modified for real-time systems, were used to formalize the development methodology and to provide adequate up-front requirements and design. Many of the tools described in this paper are designed specifically to support SA/SD. See [1] for a description of our real-time SA techniques.

Development System

FMCS was developed on a dedicated VAX 11/782 running the VMS operating system. VMS provided a robust, user-friendly base for developing, integrating and hosting the tools described in this paper. Facilities such as the on line HELP libraries for tool documentation, and the Command Line Definition utility, for automated command line parsing, were widely used when implementing tools.

Software Tools on FMCS

Tools were used to automate, to some degree, all portions of the software life cycle. This emphasis on tools was due to the need to meet a tight schedule, as well as previous successful experiences with a subset of the tools. Figure 1 summarizes the software tools used on this project. While all of these tools contributed to the success of FMCS, there were five tools in particular that will be examined in some depth. These five tools are similar in that they provided large savings to the project; they are widely available for different machines and languages; and they can be used for many different types of software development. They are:

- *Code Management* – for full, automatic tracking of source code
- *Static Analysis* – for error checking and documenting of source code
- *Path Coverage* – for ensuring all statements were executed during verification
- *Fourth Generation Languages* – for rapid generation of tools
- *Text Formatting* – for assistance in rapid production of documentation

Before deciding to use these tools, we spent considerable time and resources investigating them, to forestall any potential problems, and to ensure they would meet our needs. Many of the areas we investigated are also described in the following sections.

Code Management

What Is It?

Code management is the automatic tracking of the evolution of source code (or any other ASCII file, for that matter). Code is stored in a central "library". Changes to code and (optionally) program build configurations are recorded in this library. Every change to code is automatically identified as to date, reason, and author. Any version of a module, or any build configuration, can be accurately recreated at any time. It will notify the user if concurrent updating of a module is attempted, and, if desired, will coordinate concurrent updates by allowing the evolution and merging of variant versions of a module.

The code management tool used by FMCS was *Code Management System* (CMS)[4], from Digital Equipment Corporation (DEC). Additional procedures were written in-house, to expedite large builds and to use the CMS data base to

provide up-to-date, accurate and automatic data on development and test progress.

Applying CMS to the Project

Developers were required to check a module into CMS after initial debug, at the time it was assigned a part number by Configuration Management. Any time they needed to make a change, they *reserved* it, which gave them a copy in their own area to work on. After editing, compiling, testing, etc. until satisfied, they *replaced* it back into CMS, at which point CMS automatically assigned it the next higher *generation number*, noted date and author, and asked for a change reason. There was no limit on the length of time a module could remain checked out of the library - but it could not be taken into the lab for testing until it was checked back in. Any time a build was done for lab checkout, the subroutines and common blocks comprising it were *fetched* (i.e. obtained as read-only) concurrently with their name and generation number being permanently placed in a *class* - a CMS grouping mechanism. This procedure assured us that the exact contents of each build was permanently recreatable. In addition, the Verification team used CMS to find the latest, stable version of each module for unit test. Figure 2. shows an example of history of module evolution that is provided automatically by CMS.

Risks in Switching to CMS

Source code control tools have been well examined in the literature (see for example [5]), but are not usually used in the context of such large scale development - five or six people is typical, vs. 50 in our case. We identified several areas of risk when deciding whether to use automated source code control. These were balanced against problems we had experienced on earlier projects in maintaining adequate control of source code. These apprehensions, and the eventual outcomes, are discussed here.

Could It Handle Required Volume of Source? FMCS was a very large project, and we were extremely concerned that the tool simply would not be able to handle the extremely large load and heavy concurrent access that we required; or that the overhead required to retain the back versions and history would consume inordinate amounts of disk space. After a year and a half of experience, these are our conclusions:

- *Concurrent Access* - CMS successfully mediated concurrent access to the library, and even to the same module. On busy afternoons, the library was in non-stop use. Over 50 people accessed the library during the project.
- *Disk Space* - Disk space overhead turned out to be a minor issue. After one year of use, during which all accesses and changes to the source were recorded, the entire overhead was no larger than a single clear copy of the source. Figure 3 shows the distribution of disk space. The change history - that which is needed to recreate any previous

Tool	Where Obtained	Description	Volume of Use
Requirements Tools			
Reqmts Data Dictionary	In-house (Datatrieve/FMS)	Centralized database of all data/control flows including rates/units/resol and where used. from Structured Anaysis	5000 records
Traceability Data Base	In-house (Datatrieve/FMS/DBMS)	Data base linking reqmts - - design - - code - - test cases	8500 records
Design Tools			
Design Data Dictionary	In-house (Datatrieve/FMS)	Centralized database of global variables and all modules that reference them	21 000 records
Interfunctional Interfaces Report	In-house (Datatrieve/Fortran)	Cross-references interfaces between major system functions	
Implementation Tools			
Common Block Generator	In-house (Fortran)	Generates all FMCS common blocks directly from Design Dict	850 commons
VAX Debug	DEC	Interactive symbolic debugger that displays source during execution. breakpoints tracepoints watchpoints	
Code Management System	DEC	Source code tracking and configuration management tool	2500 modules
Verification Tools			
Problem Report DB	In-house (Datatrieve/FMS)	Tracks all software problems found during reqmts thru final system test	4500 records
RXVP-80	General Research Corp	Static Analysis/Path Coverage for Fortran	1000 subrtns
Target monitor/debugger	In-house (Fortran/FMS)	Target computer monitoring/debugging tool. VAX-hosted Includes symbolic debug of target breakpoints memory snapshots. trace buffers etc	
Dynamic Simulator	In-house (Fortran)	Full real-time aircraft simulation	
Static Environment Simulator	In-house (Fortran/FMS)	Simulates aircraft I/O VAX hosted with ability to file and recall test cases	1000 test cases
Documentation Tools			
SPELL	SW Tools User's Group	Spelling checker	80 000 entries
Runoff	DEC	Text formatter for document production	3000 files
TEX	Stanford University	Sophisticated text formatting package	
ECO Is/Was Generator	In-house (Fortran)	Automatically generates formatted is/was listing required for Engineering Change Order forms	6000 ECOs
Tellagraf	ISSCO	Very high level presentation graphics	20 plots/wk
Tools for Building Tools			
Forms Management System	DEC	Very high level tool for the creation of sophisticated interfaces	140 forms
Datatrieve	DEC	Fourth generation language for rapidly generating update/query applications	400 procedures 45 000 records
Common Data Dictionary	DEC	Central repository for data file descriptions	200 descr
Data Base Management System	DEC	CODASYL-compliant network data base	10.000 records
Differences	DEC	Lists differences between two files	
Software Tools Virtual Operating System	SW Tools User's Group	set of 80 UNIX-like* utilities for pattern matching. text manipulation. etc	used in 30 tools
HELP	DEC	On-line help for all tools including DEC. in-house and 3rd-party	175 topics

Figure 1
Software Tools Used on FMCS

```
 date of access          | auth. | action | module     | gen | prob no
29-MAY-1984 20:05:46 BUSH   RESERVE CORCTLN.FOR(13) PR 3428
  reason entered for change
 BLANK IRS DRFT RATE IF NO RADIO UPDTS IN ENTIRE FLGHT"
  module checked back into library: assigned higher gen num
29-MAY-1984 21:53:13 BUSH REPLACE CORCTLN.FOR(14) "EX1101
 PR 3428-ALLOW TIME SINCE TAKEOFF TO CUT UP FOR 1 HR"
  insert gen 14 into "package 6, pass 1" bld class; fetch for bld
29-MAY-1984 22:13:52 JOAG INSERT CORCTLN.FOR(14)
 NAVPK6PASS1 "build"
29-MAY-1984 22:54:37 JOAG FETCH CORCTLN.FOR(14) "build"
7-JUN-1984 13:19:20 BUSH RESERVE CORCTLN.FOR(14) "PR 3943
 DON'T CLEAR UNREASONABLE RECIEVER COUNTERS
 WHEN PREF MODE GOES TO ZERO"
7-JUN-1984 13:42:07 BUSH REPLACE CORCTLN.FOR(15) "EX1201
 PR 3943 -- DON'T CLEAR RCMSTN WHEN PRFMOD GOES TO 0"
  indicate which generation was used for function level test
7-JUN-1984 18:46:36 MAHLER FETCH CORCTLN.FOR(14)
 "FETCH for NVH function test"
29-JUN-1984 06:26:42 WOODRUFF FETCH CORCTLN.FOR(16) ""
  earlier generation fetched for documentation purposes
30-JUN-1984 12:07:26 BENNETT FETCH CORCTLN.FOR(13) ""
```

(Explanations of history data are in slanted text)

Figure 2.
Snapshot of CMS History for One Module

version of source - required only 14.5% of the CMS library's total disk space. The entire access history took somewhat more - 33.4%, but it was possible to store most of that off line.

- *Volume of Use* - We used CMS to track 2800 modules and common blocks. There were an average of 1500 accesses to the library a day, of which 40-80 were updates; there were over 500,000 accesses to the library in the first year of operation.
- *Execution Speed* - Execution speed was the only area we experienced difficulty in. During peak periods, library response grew unbearably slow - sometimes two minutes to fetch one subroutine. This was somewhat alleviated by keeping a clear copy of the source for quick reference, and by relegating large fetches for builds to third shift, but nonetheless remained an annoyance throughout the project. Future releases of CMS promise to greatly improve performance.

Is It Reliable? Another major fear at the outset was the reliability of the tool. Since source would be stored in an encoded form that retained history information, we were worried about the possibility of corruption or outright loss of source code. Source code loss/corruption of any type was completely unacceptable.

We instituted a special, twice-a-day backup for the CMS library, just in case. We only had to use it once - the disk drive broke, and we restored the library to an alternate disk to continue development while the drive was being fixed.

The library automatically recovered itself from minor problems such as execution being forcibly terminated in the middle of a source update. Occasional other problems required the CMS library manager to intervene - something that happened about once a month. The CMS library survived inexperienced users, devious users, managers, power failures from a squirrel in the transformer, bomb threats... without losing a line of source code over the year and a half of use.

Would Engineers Be Conscientious in Using It? The library would be effective only if all engineers cooperated and used it to control their source code from the time of initial module debug. Since this was a major change from previous source control strategies, and since there was no way to *force* them to use CMS, or to enter meaningful history comments, we were uncertain how they would react.

Although grumbling at times because of slow execution, we found that they were *extremely* conscientious in using it to track code changes. History comments were formatted, by

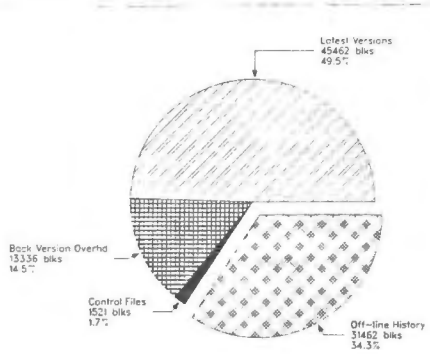


Figure 3.
CMS Disk Space Usage

convention, to include the version/mod the change reflected, and the Problem Report number, if any, that the change fixed; as well as a description of the fix. We even had requests to edit the history file, to correct misentered text!

Would It Alleviate Control Problems? Our final concern, and the reason we considered code management tools in the first place, was the need to coordinate source code control on a large and dynamic project - an extremely difficult task. Despite elaborate precautions, we had experienced expensive control problems on earlier projects - cases where we could not identify the source that went into a build; where the wrong modules went into a build; where the verification team wasted time testing out-of-date modules; or where concurrent updates of the same module were unknowingly made by two people.

The combination of CMS and the procedures we established for its use almost completely alleviated these problems. The Configuration Management and Software Quality Assurance groups heartily endorse its use, and themselves use it to track major releases to the customer. Mechanisms we used to solve control problems include:

- *Unknown Builds* - by convention, every time a major build is done, a class containing the specific generations of subroutines, common blocks and link control file composing the build is generated - an example of a class might be JULY19NAV-PASS2. This uniquely and permanently identified the build, and allowed it to be reproduced at any time.
- *Concurrent Updates* - Users are automatically notified if they access a module currently checked out for update to someone. In addition, when concurrent updating *is* desired (for example, when adding new functionality while retaining a stable baseline that incorporates only bug fixes), CMS allows the creation of variant lines of descent, and their eventual merging, if desired.
- *Progress Tracking* - Generation numbers were used to uniquely identify specific versions of modules over time. For example, the Problem Report Data Base recorded which generation of a subroutine the problem occurred in, and in which generation the fix was made. This provided the data for a report which accurately shows what problem reports are fixed in a previous release. Function/unit test status was recorded by generation number, to ensure that the modules being tested were the same ones going into the builds.

Code Management Conclusions

CMS is now being used on all new projects. Additionally, it is being retrofitted to projects that reactivate for maintenance, as appropriate. It has solved the source control problems we had experienced in the past, and has had the side effect of providing progress data to management. Some projects are using it to control documentation as well as source code, and we expect this trend to widen. Its only drawback is lack of execution speed during heavy concurrent access.

Errors Detectable by Static Analysis

- Mismatch in number of arguments
- Mismatch in types of arguments
- Used but never set local variables
- Unreachable code
- Mixed mode expressions
- Global variables never referenced

Documentation Generated by Static Analysis

- Calling tree
- Common block set/use matrix
- Global symbol cross reference
- Line counts by statement type
- Called-by list for each subroutine

Figure 4.
Typical Outputs from Static Analysis

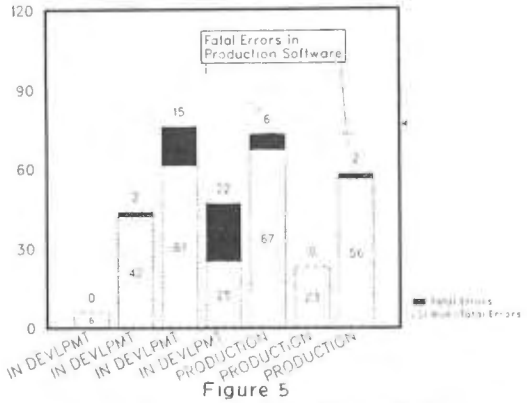


Figure 5
Errors Uncovered by Static Analysis

Static Analysis

What Is It?

The second software tool that had a major impact on software development for FMCS was a *static analyzer*. A static analysis tool is one that inputs the source code, and examines it for certain classes of errors. It is also used to provide extensive local and global documentation about the structure of a program. Figure 4 lists the types of errors that can be detected by a static analysis tool, and the types of documentation that can be produced. The static analyzer we used was RXVP-80, from General Research Corporation.[6]

Applying Static Analysis to the Project

Static analysis was a required part of unit testing for all Fortran routines on FMCS. A module was required to be free from any static errors before it could pass unit test. The testing runs are permanently filed to document testing thoroughness. We have been using static analysis for over three years; FMCS is the second major project it has been used on.

Risks in Switching to Static Analysis

We originally acquired a static analyzer as a way to cut down on the costs and tedium of the many inspections that were required during unit test. The following were our initial apprehensions and hopes when acquiring the tool, and our experiences in actual use.

Could it Handle Large Volumes of Code? We were concerned that the tool would not be able to handle the large amounts of source we routinely dealt with. Through experience, we have found up to 100,000 lines have been analyzed at one time, and execution speed is not a problem – it is about the same as a cross-compiler. We estimated that we have analyzed over ten million lines of source total.

Would It Find Errors? A major reason for acquiring the tool was the difficulty in checking for consistency over many hundreds of subroutines. We found the static analyzer to be extremely helpful in this regard. When we first acquired RXVP-80, we analyzed several large pieces of support software that had been in production for over a year, as a means of stress testing the static analyzer. To our surprise and chagrin, the static analyzer performed fine – but our production software had several fatal errors in it, uncovered by the analyzer! These included mismatch in number of calling arguments, and variables used but never set. Figure 5. shows the errors uncovered in seven programs the first time they were run through the static analyzer. Although many of the errors were potentially innocent things like mixed-mode expressions, some were severe. We also found several instances of global variables that were no longer accessed anywhere – but they had never been removed, because there was no way to be certain that there was not *one last reference*, somewhere.

Would It Save Time? One of the major justifications for acquiring the tool was time savings – to both reduce manpower costs, and pull in schedule. Before the acquisition of RXVP-80, unit test had required a number of checks to be done by hand. Static analysis automated almost all of these, with the result that unit test time was reduced from an average of four hours per subroutine, to .73 hours per subroutine, a savings of over 80%. At this rate, the tool paid for itself in just a few months.

Uncovering Program Structure

In addition to its use on FMCS, we find static analysis a useful tool in other situations. A primary use is uncovering the structure and data flow of a large, unknown piece of software that has to be debugged. For example, finding everywhere a common variable is referenced can take hours by hand, but only seconds with a static analysis listing.

Static Analysis Conclusions

Static analysis tools are still relatively unknown – and yet are indispensible on a large project. Their use of FMCS saved money and cut schedule. More advanced languages, like Ada[4], will include some of the functions of static analysis.

Path Coverage

What Is It?

A *path coverage* tool shows which statements have and have not been executed during a test. It also shows which paths out of a decision statement have and have not been taken. The path coverage tool reads the original source code, and outputs an augmented source file, identical in logic to the original, but also containing probe statements to trace its execution. This instrumented source is then compiled, linked and executed as usual, producing a file of probe data in addition to normal output. This data is then run through an analyzer that summarizes which paths through the code were not executed. The tool we used to perform path analysis on FMCS was an option of the RXVP-80 tool, described above, and many of the same comments apply.

Risks In Path Coverage

Besides the initial concerns about volume, etc., discussed in the previous section, there were areas in path coverage that we we were cautious in examining.

Host vs. Target Execution. The systems that we develop are cross-targeted, meaning that they execute on a different computer than the one they are compiled on. It is not feasible to do path coverage on the target machine, since it does not have the disk file for storing probe results. Instead, since the source code is written in Fortran, we simply used the host computer's Fortran compiler, and ran the unit tests on the host computer.

We were initially apprehensive that the differences in architecture (different instruction sets; 16 vs. 32 bit) would yield inconsistent testing results. After many thousands of tests, we have found this not to be the case. Not only are results

[4] Ada is a trademark of the DoD

consistent across machines, but the emphasis on unit test resulted in very clean loads once testing was switched to the target environment – the program was running in the lab in a matter of days, rather than weeks and months as on previous projects.

Would Path Coverage Help Thoroughness? As part of our Verification effort, we must show 100% path coverage for all subroutines, and that every decision is taken every possible way. This is something that is almost impossible to do without an automated aid – even in cases where a tester was sure all paths had been covered, RXVP-80 would reveal one or two that had not been covered.

Path Coverage Conclusions

Path coverage is an invaluable aid when thorough testing is a necessity. We do not usually do path coverage on non-critical software, but find it a necessity on most airborne programs.

Fourth Generation Language Techniques

What Is It?

Fourth generation language is a loosely-defined term for the very high level report writers, data inquiry and screen formatting tools that have become widely available in the last few years. The key factor in all these tools is that they are non-procedural – they allow you to specify *what* is to be done without saying *how* to do it. They allow for extremely rapid development of software – perhaps an order of magnitude over what is possible with traditional languages.

These techniques are only applicable to a certain class of problems; mainly information acquisition, analysis and reporting. However, on a large software project, there are a tremendous number of needs that fall under this category. In the past, these needs were either ignored or attempted manually. The cost of writing programs to automate them; plus the short lead time and rapidly evolving needs, made them impossible to implement cost effectively. However, fourth generation techniques now make the automation of the "project knowledge base" feasible.

The tools we used to automate information management for FMCS are part of an integrated environment called the *VAX Information Architecture* (VIA) from DEC[4]. The pieces we used included *Datatrieve*, a data query and report writing language, *Forms Management System* (FMS), a tool for designing fill-in-the-blank screen interfaces, *Data Base Management System* (DBMS), a CODASYL-compliant data base, and the *Common Data Dictionary* (CDD), a global data description repository that ties it all together.

Applying Fourth Generation Languages to the Project

These tools were used to generate a host of tools that we used to more effectively track and control FMCS. Some of these include:

- *Problem Report Data Base* – for tracking all problems found in the source, ensuring they were fixed, and monitoring problem causes and trends.
- *Requirements Data Dictionary* – for tracking all data and control flows during Structured Analysis
- *Design Data Dictionary* – for tracking global variable, their rates/units/descriptions, etc., in Structured Design; and for automatically generating common blocks
- *Requirements Traceability* – for linking requirements to the design that meets it, to the code that implements it, and to the test cases that test it.
- *Action Items*

and many more. Figure 6. gives an example of Datatrieve syntax.

Unlike the first two tools described, for which we had definite requirements and made risk tradeoffs, our widespread use of fourth generation techniques was something that happened almost without anyone realizing it. The VIA tools were purchased for reasons other that project software support. However, we found that decisions were made to implement tools in Datatrieve vs. Fortran, because of the large savings in development time. Even more commonly, individual engineers saw savings possible by automating the tracking of

```
DEFINE PROCEDURE INDIVIDUAL-REPORT
   READY PROBLEMS SHARED READ
   FIND PROBLEMS WITH ORIGINATOR = "SMITH" SORTED
            BY MODULE, DATE-FILED
   REPORT CURRENT ON REPORT.LIS
   SET COLUMNS-PAGE = 132
   SET REPORT-NAME = "PROBLEMS FOUND BY SMITH"
   PRINT MODULE, PROBLEM-DESC, DATE-FILED, CURRENT-STATUS
   AT BOTTOM OF MODULE PRINT "TOTAL BY MODULE:",COUNT
   AT BOTTOM OF REPORT PRINT "TOTAL PROBLEMS:",COUNT
   END-REPORT
END-PROCEDURE
```

Figure 6.
Example of Datatrieve Syntax

areas under their responsibility, and either set aside a couple of hours to do it themselves, or requested it from project support personnel. These trends continued, until now we have several dozen types of project data being tracked with fourth generation tools – *data that was simply lost on earlier projects.*

Another unforseen outcome was the shift of tool development from software engineers, who are in short supply, to local business school graduates with some programming experience, who are able to pick up and become proficient with Datatrieve in a short amount of time.

Rapid Protoyping With Fourth Generation Languages

For some of our tool applications, we had little idea what the problem really was or how it could best be solved – Requirements Traceability is an example. Rather than go through extended requirements, design, and implementation phases, only to find that the tool we developed did not fit the problem, we used the fourth generation tools to quickly put together a first cut at a solution, then try it out. It was often possible to put together a prototype in one or two days; the engineers would try it, find what they didn't like, and the next version, incorporating their suggestions, was quickly generated. In some cases, the evolved prototype was kept as the final version, with only documentation added. In others, the prototype was discarded and the final version developed from scratch.

Difficulties Encountered

We experienced many growing pains as our use of fourth generation techniques expanded. We eventually recognized and controlled most of them; they seem to be a common occurrence for shops who use these techniques.

Unmaintainability. We found that these applications were often developed casually, and tended to grow continuously and in an unstructured fashion as new features were added. They were poorly documented, and often impossible to understand by anyone but the original author. The structure of the data and the user interfaces were haphazard. As this problem became recognized, we began applying the strict structuring and documentation rules that govern the rest of our tools.

Uncontrolled Source/Data. For much the same reasons as above, the fourth generation source code was not initially placed under configuration management, and the data was not protected in any way other than normal system backup. We had cases where this source was inadvertently lost, and cases where data was irretrievably corrupted or lost. This was eventually solved by requiring that the Datatrieve, CDD, and FMS source code, for all but personal applications, follow the same configuration management practices as the rest of the support software. The data files are overseen by a Data Base Adminsitrator.

Inefficiency. Fourth generation techniques are very powerful, and hence rather CPU-intensive. We consider this tradeoff reasonable for the time/labor it saves. However, it is occasionally possible to do extremely inefficient things without realizing it. Without help from someone knowledgeable in the internals of Datatrieve, we had applications that started taking inordinate amounts of CPU time. This help is now

available from individuals who have extensive background in the product.

Conclusions on Fourth Generation Techniques

The ease of producing tools with fourth generation language techniques - something we stumbled on almost by accident proved to be invaluable for supplying tools to FMCS. Without it, most tools would have gone unwritten because it was not possible to implement them cost-effectively in traditional languages.

Text Formatting

What Is It?

Text formatters are common tools that have been available for a couple of decades. But, despite the potential savings to projects, they are not nearly as widely used as one would expect. A *text formatter* is essentially a word processor, but with a slightly different orientation. Commands are usually embedded control strings, vs. What-You-See-Is-What-You-Get with traditional word processing systems. Their abilities are often more sophisticated, and include producing indices, tables of contents, renumbering paragraphs, etc.

The importance of powerful documentation tools for engineers is emphasized by the fact that about two thirds of the time spent on a large software project results in documentation as its direct product, and only one third results in code as its direct product [7]. And, engineers traditionally dislike spending much time on documentation.

We used two text formatters for FMCS documentation. Runoff [4], from DEC, was used from the onset of the project, starting with the requirements documents. In all, over 3000 documentation files were formatted with Runoff, including Requirements Specs, Design Specs, Test Plans and Procedures, and most project memos. Towards the end of the project, we acquired a laser printer and TEX, a public-domain typesetting quality formatter that handles multiple fonts and equations [8]. (This paper was produced by TEX.)

Applying Text Formatting to the Project

On earlier projects, documentation had been done in a centralized Word Processing facility that was frequently backlogged, resulting in a turnaround time of days to weeks. When Runoff became available, engineers switched to it immediately, without being asked to, and were highly enthusiastic about its use. Documents were done in a more timely manner, and were of higher quality - the latter because the instantaneous turnaround time encouraged frequent revisions. A spelling checker further increased document quality. Almost all draft documentation was produced on the VAX in this matter, until the final release, at which time the documents were electronically transmitted to the Word Processing (WP) computer for final cleanup and release.

Automated Generation of Documentation

Another effect we began to notice was that we began to produce a good portion of our documents automatically. Data was pulled out of data bases, formatting commands inserted automatically, and run through the formatter. For example, the documentation on system data flows could be generated from the requirements and design data dictionaries. On earlier projects, this work had all been done slowly and oftentimes inaccurately, by hand.

Problems with Text Formatting

We found two main shortcomings with text formatting. The first had to do with the functionality we required from the tool - Runoff simply did not have the power to do the things we needed, like mathematics, alternate character sets, specialized layout, etc. In addition, the dot matrix printer was not of high enough quality for formal documentation. (These problems have since been solved with the acquisition of TEX and a laser printer).

The second problem, indirectly related to the first, was our practice of transmitting the documents to the centralized Word Processing Computer. The formatter on the WP computer of course differed drastically from Runoff. Although

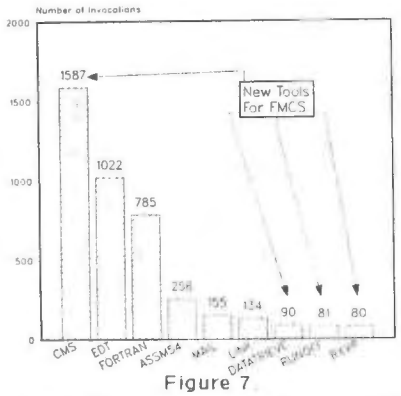


Figure 7
Tool Use by Number of Invocations

we transmitted the documents after formatting by Runoff, there still remained much work that had to be done by WP - both correcting Engineering typing and formatting mistakes, *plus* correcting errors introduced by the reformatting on a different machine. This overhead made it impractical to transmit documents freely between WP and Engineering - something that would have been useful.

Text Formatting Conclusions

Text formatting is one of the most promising areas for continued productivity gains. More sophisticated tools, including ones capable of integrating text and graphics, will be investigated for future projects.

Indirect Effects of Tool Use

Changes in Work Habits

By supplying all of these aids, most of the engineer's work revolves around the computer - encompassing much more than just software development tasks. One effect of this is that a terminal becomes prerequisite for most of the tasks an engineer must do, from software development to documentation to status reporting. *A terminal on every engineer's desk becomes a necessity.* While this might seem expensive at first glance, it becomes reasonable when one observes that you would not expect an engineer to go to a "pencil room" when he wanted to write something; nor would you expect him to share a pencil with the other person in his cubicle. Computer access through the terminal has become as fundamental to the software engineer's job as a pencil was to his earlier counterparts.

This is seen by the frequency of invocation of some of these new tools (figure 7) - tools that weren't even used on earlier projects are among the most heavily used on this one.

Improved Communications

We found this de facto "Office Automation" helped counter the N^2 communication paths problem that has traditionally been a serious problem on large projects. Mail began to replace telephone tag and disruptive drop-in's; memos were delivered electronically; note taking, reminders, action items, and other small clerical tasks were automated. An on line network link to a VAX at Boeing further facilitated communications, with the ability to instantaneously transfer memos, to remotely log Problem Reports into our data base, etc. The three hour difference in time zones between LSI and Boeing made the ability to communicate asynchronously even more important.

Management Tracking Information

In addition to their primary task, many tools had the secondary ability to provide project tracking information to management. In most cases, extracting the tracking information consisted of nothing more than counting and summarizing records. For example, net open problem report graphs, (figure 8), showed us overall trends in problem resolution. The Problem Report Data Base also summarized what phases of testing were uncovering the most bugs, and

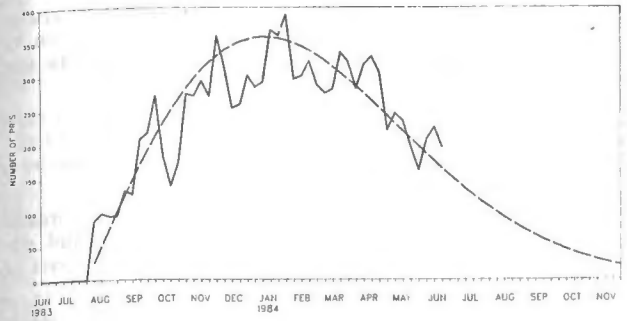


Figure 8
Net Open Problem Reports

what the sources of those bugs were. This quicker and more thorough knowledge of project status helped management find out when things were going off track earlier, and get them back on track without as much disruption.

Project Knowledge Goes On Line

The preceeding paragraphs demonstrate pieces of an overall trend - the *project knowledge base* is now going on line, rather than being stored in peoples heads and desks. As this happens, everyone has access to the latest, correct information. Things are much less likely to get lost or forgotten; data can be summarized, counted and interrelated. In addition to the immediate benefit this has, we expect to draw further benefit in the future, after the project is complete. We can analyze what is essentially the entire history of the project, decide what things we did right and what things we can improve on, and use that knowledge on later projects.

The benefit of this on line knowledge base cannot be underestimated - this is all information that was lost on earlier projects, and having it as input to the decision making process is invaluable.

Supporting the Tools

There is one additional aspect to using tools in the real world, that cannot be overlooked - getting the best possible tools, and providing comprehensive support for them. The next paragraphs outline briefly some of the strategies we used.

Software Development Environment Plan

One of the most valuable techniques for overcoming lack of familiarity with tools and their benefits, and for putting together an overall strategy for tools acquisition and use, is something we call a "Software Development Environment Plan". This plan analyzes where you are, where you want to be, and the step to be taken to get you there. Figure 9 shows an outline of our current Plan. An SDE Plan can go far in creating a positive atmosphere for tool acquisition and use. Some of the benefits we have experienced as a result of our SDE Plan include:

- *An SDE Plan forces recognition of tools as a valid area of concern.* The plan introduces management and programmers to the concept of software tools, and shows how tools fit into the current structure of the organization. As a formal written document, it goes into the budgeting process, with money being allocated specifically for the purchase of additional tools.
- *An SDE Plan helps integrate the tools into a coherent "whole".* It forces planning from the top down, thereby recognizing the environment as an interrelated set of tools that must work together and interact in predictable ways.
- *An SDE Plan produces a "shopping list" of tools.* At the completion of the plan, a number of tools will have been identified as being needed to provide an automated software development environment. Constraints and requirements will also have been identified. This list of tools can be used to prioritize tool acquisitions; to provide feedback to vendors on those tools you would like to see provided; and to begin a search to find someone who provides a tool that meets your needs.
- *An SDE Plan can force a formalization of your software development methodology.* One of the first things you may find as you start to try to match automated tools to the way you develop software is that everybody does it differently, with different intermediate documents, different ways of representing requirements and design, etc. It is impossible to automate anarchy; one of the first things required may be the adoption of a formal software development methodology and life-cycle model. Without such a methodology, it is possible to support only the implementation phase

What an SDE Is: Why it is Important
Summary and Recommendations
What has been accomplished since last Plan
Underlying tools and philosophy
Tools for software development
    tools for requirements
    tools for design
    tools for implementation
    tools for testing
Tools for document production
Tools for project management
Tools for office automation
Tools for information management
Non-software issues
Long Range Possibilities

Figure 9.
Outline of Software Development Environment Plan

Criteria for Selecting Tools

Whenever possible, we buy the tools we need, rather than make them, for simple economic and manpower reasons. The cost of a tool is only one small part of what we look at when evaluating a tool; other qualities are of equal or more importance. It is interesting to note that the criteria used below, in addition to helping us choose one tool over another of the same type, at times will make us choose *no* tool - if it cannot meet these criteria acceptably, we simply will do without any tool in that area. Criteria to use in narrowing down the list include:

- *Suitability of the Tool.* Does it meet all the functionality requirements you have? Does it have a good user interface?
- *Reputation of the vendor.* Do his products have a reputation of being easy to use and containing few errors? Does he acknowledge and fix quickly bugs that are found?
- *Frequency of enhancements.* Once you get a tool, it won't be long before you are thinking of additional capabilities for it. Does the vendor have a record for continually enhancing the functionality of the product? Is there a user group or other means for you to provide inputs? Are the enhancements included in the yearly maintenance fee?
- *Integration with other tools/data.* The more closely a tool will work with other tools and data you have, the more benefit it will have. For example, a statistical program that expects its input in a form that is incompatible with the output of your data-generating programs is of little use.
- *Can you get it on 30 day trial?* Many companies will allow you to try a piece of software for 30 days, and return it for little or no cost if you are not satisfied.

When the choices have been narrowed down to two or three possibilities, a good way to find out how well it *really* works is to get the name of a current user from the tool's seller. Call that person up and see what he thinks of it. Although the person obviously is somewhat satisfied with the tool (or you would not have been given his name), most users are quite frank about the strengths and shortcomings in a tool. Many of our tool choices have been made or broken based on feedback from current users.

Successful Introduction of Tools

After a tool has been purchased, there still remains much work to do before it can be considered a success. Strong support of a tool during its introduction and ongoing use will impact how often the tool is used, and how effectively it serves the purpose for which it was obtained.

The three most important factors in successful introduction of a tool are support, support, and support. No matter how easy it is to install and use, there still must be a local "champion" who will promote the use of the tool and will always be available to answer questions and solve problems. The duties of this support person include:

- Installing it on the system.
- Customizing as necessary, including command procedures that tailor it to local needs.
- Getting copies of, or writing, User Manuals, and putting up on-line help.
- Providing initial training (whether through formal class or informal memos).
- Being available to patiently answer the same questions over and over again as every new user tries the tool.
- Aggressively searching out new ways the tool can be applied, and new people who can use the tool.
- Tracking down the inevitable bugs that come up, filing trouble reports with the vendor, and providing a workaround for the user (if there is one) or commiserating with the user (if there is not one).
- Providing a focal point for suggested enhancements from users; implementing those that can be done locally and forwarding the the rest on as polite requests to the vendor.

It has been our experience that each tool requires a large block of a support person's time up-front. One to three months, full time, is not unusual for a large, complicated tool. (Smaller tools take correspondingly less time.) The tool itself is usually up and working within the first half day or so. The extra time comes in tailoring it to make it as easy and natural as possible for users to invoke, in gaining a full understanding of all its idiosyncrasies, and in continual training/question-answering. After the initial implementation is complete, we still find that a support person spends on average a couple of hours a week on tracking bugs, helping new users, installing later versions of the tool, etc.

It is important to recognize that this level of support is required to gain full benefit of the tool use, and to be willing to support the tool in this way. The support person chosen should get along well with people, and be eager to help - he should go out of his way to make sure the tool succeeds.

Finally, support also must be available in terms of hardware. If the tool requires significant hardware resources, these must be planned for or the tool will be disliked by users, who do not get adequate response time, and will be resented by non-users, who see it stealing their already limited resources.

Words of Advice

Finally, some subjective words of advice on things always to keep in mind when considering software development tools for large project support:

- *Get "Industrial Strength" tools* - the tools must be exceptionally robust, and should fail gracefully if they do break. Most tools cannot stand up to the heavy volume and unusual cases generated by a large project.
- *Spend time on the user interface* - the tool must be extremely easy to call up, should not require a manual to use, and should default to the ways the project usually does things
- *Supply the needed hardware resources* - hardware is cheap compared to software engineers. Skimping on hardware resources is not a way to save money.
- *Don't wait til the last minute to get started* - the contract award date is not the time to decide which tools you need. If the tools are not ready up front, engineers will be forced to change the way they do things midstream, when the new tools come on line - a big disruption to the project, and a real disincentive to engineers to use the new tool.
- *Get management behind you* - Without management support for tool purchase, and especially for their continuously and strongly-stated desire for tool use on a project, some people just will *not* change.
- *You can't automate anarchy* - unless you have a formalized, consistent methodology for requirements and design, you will only be able to provide support for part of the software life cycle - coding and verification.
- *Progress is evolutionary, not revolutionary* - Tool use is continually changing - partly because it is impossible to understand fully, ahead of time, all the ways and nuances of a tool's use. Equally important, but often unrecognized, is that the fact the the engineer's work habits are evolving along with the tool. He will want to use the tool in new ways as time goes on, and he incorporates it into his workstyle. *Tools evolve based on feedback from engineers; and engineers' workstyles evolve because of tool use, causing more tool evolution. . . .*
- *Tools are not enough* - Tools can be an important aid in controlling the complexity of large software project, but they are not a panacea. Without proper management and methods, any project can fail.

Conclusions

A number of advanced tools and techniques were used to increase software productivity and quality for a Flight Management Computer System. Many benefits were gained, and many lesson were learned in attempting to use these tools for the first time. Five tools, in particular, turned out to have large paybacks and to be applicable on many types of projects - code management, static analysis, path coverage, fourth generation language techniques, and text formatting. An emphasis on obtaining top quality tools and providing strong user support played a major role in the success of the tools. The end result was a high quality, on-time, maintainable product.

References

[1] Hatley, Derek. "A Structured Analysis Method for Large, Real-time Systems", DECUS Languages and Tools SIG Newsletter, February 1984.

[2] DeMarco, T. *Structured Analysis and System Specification*, Yourdon Press, New York (1978).

[3] Yourdon, E; Constantine L. *Structured Design*, Yourdon Press, New York (1978).

[4] Digital Equipment Corporation. "VAX Systems and Options Catalog", 1984.

[5] Rochkind, M. "The Source Code Control System", IEEE Transactions on Software Engineering, December 1975.

[6] General Research Corporation. "RXVP User's Guide", Santa Barbara, CA, 1982.

[7] Boehm, Barry. *Software Engineering Economics*, Prentice-Hall Inc., Englewood Cliffs, N.J., 1981.

[8] Knuth, D. *The TEXbook*, Addison-Wesley Publishing Company, Reading, MA, 1984.

Acknowledgments

The credit for the success of tools on FMCS belongs with the project engineers, who mastered and effectively used so many new tools under tight time constraints, and with the Support Software staff, who did an excellent job in providing and supporting the tools, under the same tight constraints.

MULTI-SENSOR SURVEILLANCE PERFORMANCE EVALUATION TOOL

84-2631

Dr. Christopher L. Bowman

Mr. Gleason Snashall

VERAC, Incorporated
9605 Scranton Road, Suite 500
San Diego, California 92121

Abstract

The Multi-Sensor Integration (MSI) software testbed is described. The use of fusion algorithms within a layered fusion processor illustrates the possibilities for off-line simulation of real-time avionics systems in order to define system requirements and develop MSI algorithms.

Introduction

Aircraft surveillance data is segmented among many complementary multispectral sources. These tangled surveillance pictures must be integrated to effectively engage Beyond Visual Range (BVR) threats with minimal exposure to counter-attack.

VERAC has developed a software testbed to address this problem in an off-line mode. In analyzing the results, the ability of the VERAC fusion algorithms to identify the target, develop state vectors of sufficient accuracy, and to reject false alarms requires that not only must the software generate state vectors, but sufficient instrumentation must be implicit to evaluate each stage.

The Problem

In generating the surveillance picture from multi-spectral sensors, reports from each sensor must be associated with the corresponding reports of other sensors. Due to size and weight limitations and the differing spectral observables, the capabilities of each sensor in the suite are not consistent. Often for example, passive sensors do not develop a range or range rate measurements which would be useful for associating their reports with each other and with radar. Hence, the problem is to associate incomplete, misaligned, and inaccurate overlapping sensor measurements to generate accurate ID and kinematic state vectors. This is done for example in an environment where the number of emitters in a radar resolution cell can make report association not only ambiguous but many-to-one.

Avionics System Architecture

Although avionics systems are not completely alike, they often are similar enough that a general approach to the simulation of sensors and sensor post-processing is possible. An example of such a system is shown in Figure 1. Shown are N sensors (i.e., radar, ESM, IRST, IFF,...) which comprise the system sensor suite. Each sensor typically has a dedicated post-processor to track a target, merge split reports, compute measurement covariance, etc. The data is then placed on a bus for routing to the MSI processor. The MSI processor fuses the data into a consistent picture for use by the System Manager (e.g., pilot, sensor allocator, internet controller,...). The System Manager then sends the appropriate control

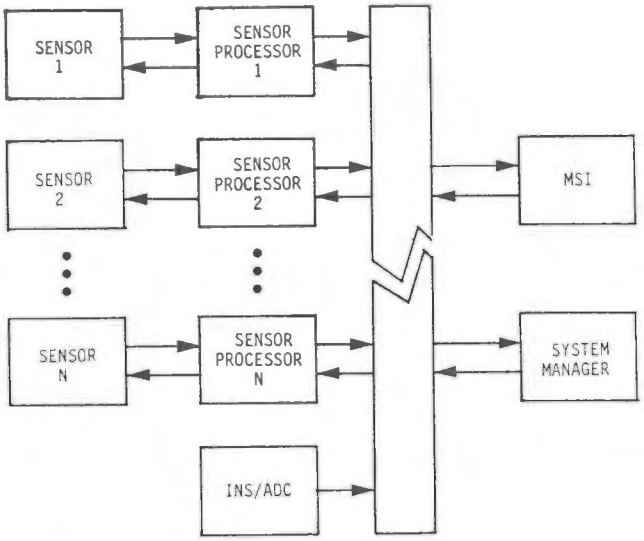


Figure 1. Example Avionics System

commands to the sensor processors, and MSI processor. Usually, the INS/ADC data is available to the processors for use in transforming data to a useful coordinate system.

An important point from a simulation designer's point of view is that the interface bus forms a natural division of functions. The interface itself can either be asynchronous (packetized) or synchronous (TDMA). In either case, the sensor report which is transmitted to the MSI software from the sensor processors typically consists of sensor mode, report time, sensor identification data (what kind of target is the source of the data), state vector, and possibly measurement error estimates. The data from the MSI processor to the system manager also consists of the same type of data.

Software Design Requirements

The MSI simulation package directly addresses at least 3 issues:

1. Memory Requirements: The Fortran code can be scaled to estimate the data and program memory requirements of the real-time avionics software.

2. Software Timing: The modules which require the most computational burden can be determined. By use of representative scenarios, these results may be scaled to estimate the real-time software loading and latency.

3. Full Instrumentation for Performance: In real-time systems, the testing required to measure transient phenomenon is often difficult and repeated experiments are required in which the parameters are varied. Figure 2 illustrates the traceability requirement in tracking a target report generated by the simulated sensors through report association and vector estimation. The traceability permits an analyst to verify whether a report is correctly or incorrectly associated and if the predicted accuracy of the resultant ID and kinematic vector reflects (in the statistical sense) the actual vector accuracy.

The MSI simulation package has been specifically designed to facilitate the investigation of these issues.

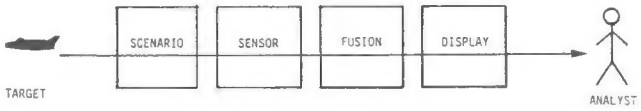


Figure 2. Traceability

MSI Simulation Architecture

Figure 3 represents the high level software structure of the MSI package while Figure 4 illustrates the data flow. From the scenario file, the scenario simulator generates a trajectory file. The trajectory file is input to both the sensor simulator and the display software. The sensor simulator uses the emitter/track characteristics file and the sensor management file as well as the trajectory file to generate reports for input to the fusion and display software. From the reports and emitter class file, the fusion package develops state vectors. The display software then uses the data output at each step to generate the plots and printouts as defined in the display commands file.

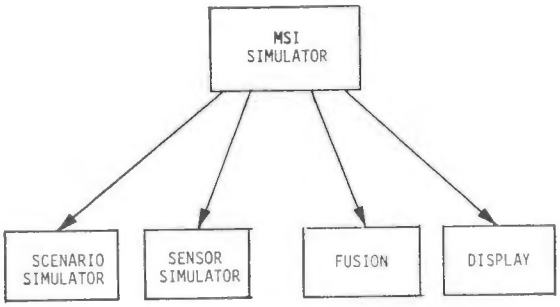


Figure 3. MSI Simulator Software Structure

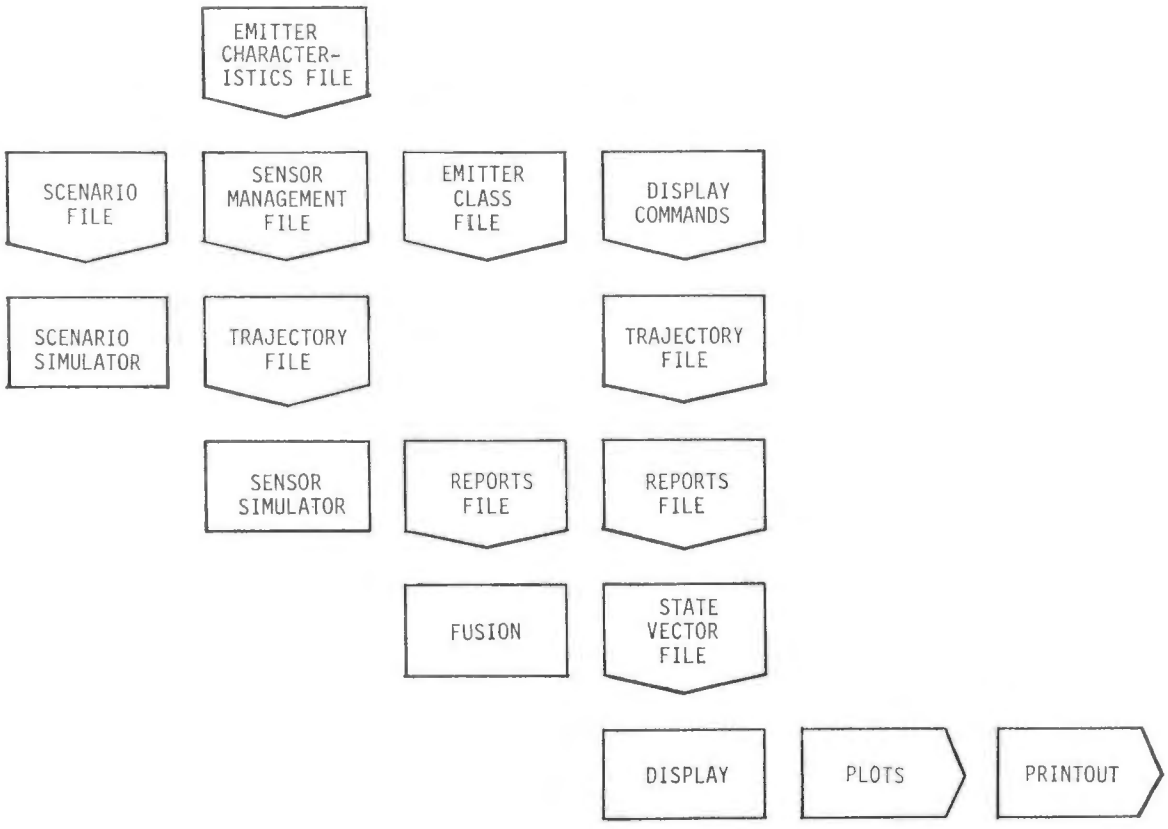


Figure 4. MSI Data Flow

Fusion Software Layers

Figure 5 illustrates the software layers of the fusion processor. The problem structure made it possible to view the software development in terms of the International Standard's Organization (ISO) reference architecture. The report entering the fusion software comes in as either a synchronous message or asynchronous packet. In either case, the event software accepts the data and queues the report in the database for service by the fusion routines. The fusion routines de-queue the data and then either fuse the data with a previous vector or initiate a new vector. Figure 6 illustrates the typical content of a vector. The vectors are saved in the database for use by the data output modules. The data out module formats the vector for use by the display package.

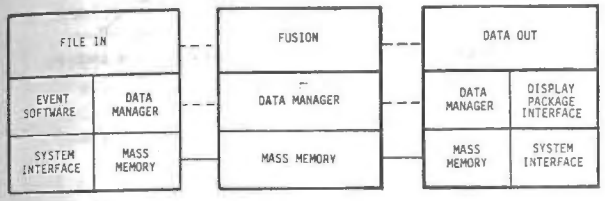


Figure 5. Software Layers

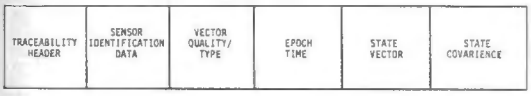


Figure 6. Data Element

Fusion Algorithm

For sensor suites composed of dissimilar (i.e., noncommensurate) sensors such as on a single aircraft the sensor identification data for each report can be sufficiently represented as a track class tree (TCT), see Figure 7. A multiple level target class tree is used since some sensors have difficulty in distinguishing much more than that there is a possible target while others can distinguish target class or target type (e.g., MIG 25, or F15). When integrating similar sensors as in multiple aircraft data fusion, sensor observables (e.g., RF, PRI) are needed. By automatically fusing the data from disparate sources in MSI to generate improved confidences, the pilot is relieved of the correlation burden.

VERAC has developed several fusion algorithms which select the report associations based simultaneously upon all the data (e.g., attribute, kinematics, and a priori). In general, the sensor reports are successively fused starting with the higher and proceeding to to the lower quality

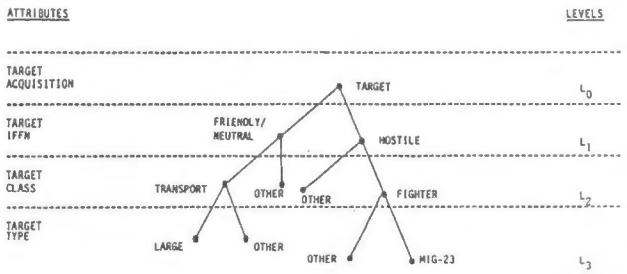


Figure 7. A Simplified Target Class Tree Structure

vector classes, as shown in Figure 8. This reduces the high cost and computational burden associated with a complete simultaneous treatment of all reports. The partitioning of the data over time, sensors and reports per sensor for data fusion specifies a fusion tree. The fusion processing at each node in this fusion tree consists of three steps as follows: common reference alignment, report association, and target state estimation (both kinematic and ID). Report association is accomplished via the generation, evaluation and selection of association hypotheses.

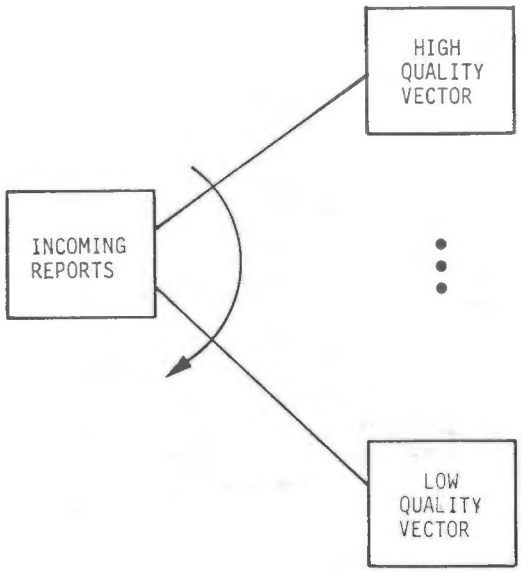


Figure 8. Fusion Sequencing

RESULTS

Figures 9-12 illustrate some results from a aircraft data fusion study, Reference [1-7], performed with the MSI software. Figures 9 and 10 represent the independent sensor errors while Figure 11 is the MSI error. This fusion process permitted the MSI software to generate a sensor misalignment bias estimate to improve future

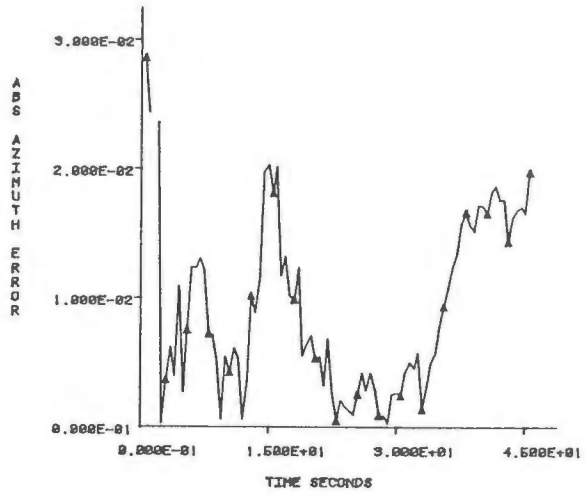


Figure 9. ESM Filter Azimuth Estimate Error

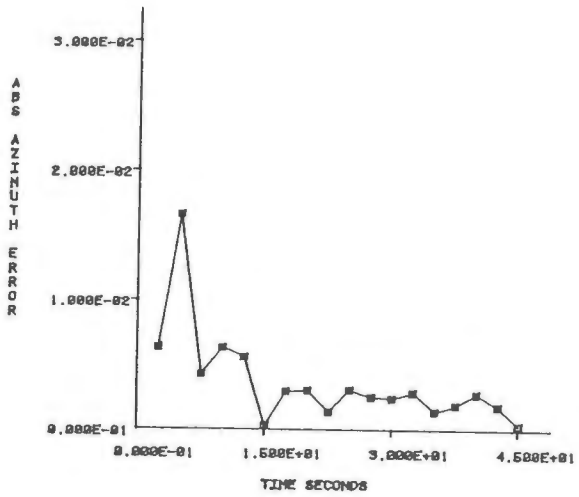


Figure 11. MSI Filter Combined Radar and ESM Azimuth Estimate Error

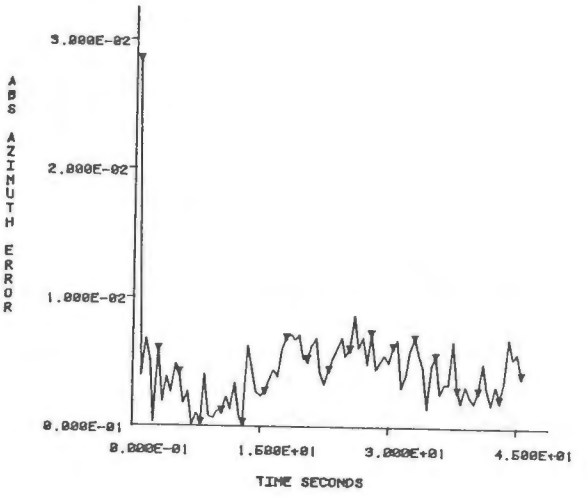


Figure 10. Radar Filter Azimuth Estimate Error

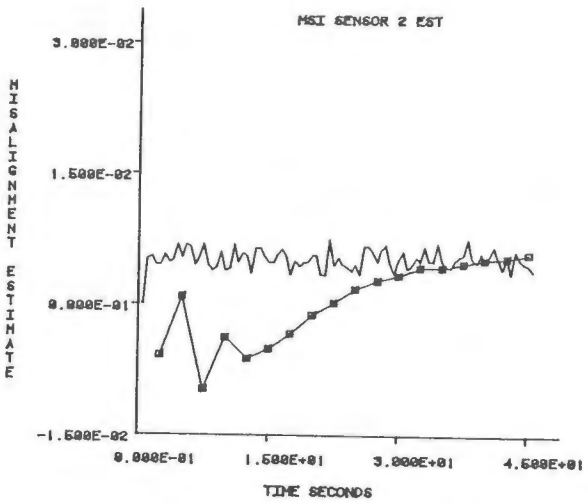


Figure 12. MSI Intersensor Misalignment Estimate

fusion estimates (Figure 12). Within the same study, the computer size and speed requirements for both autonomous anhd cooperative data fusion were defined for representative scenarios. The overall performance evaluation measure used to compare alternative MSI algorithms was the probability of satisfying the mission raid count, track accuracy and track ID requirements defined over time.

References

[1] M. Witt, "An Introduction to Layerfo Protocols", Byte, Sept, 1983, pp. 385-398.

[2] C.L. Bowman and R.K. Munzer, "Sensor Data Fusion," Interim Technical Report, AFWAL-TR-82-1013, February 1982.

[3] C.L. Bowman, "Passive Detection, Tracking and Identification," ADPA Advanced Concepts in Fire Control Conference, May 1983.

[4] C.L. Bowman, L. Gross, H.J. Payne, "Data Fusion with Intelligent Assistance: INCA", 16th Asilomar-Conference on Circuits, Systems, and Computers, November 1982.

[5] C.L. Bowman and M.S. Murphy, "An Architecture for Fusion of Multisensor Ocean Surveillance Data," 20th IEEE Conference on Decision and Control, December 1981.

[6] C.L. Bowman and J.M. Nash, "Passive Air Tracking System Accuracy Analysis," Second Tactical Air Surveillance and Control Conference Proceedings, RADC, November 1981.

[7] C.L. Bowman, C.L. Morefield and M.S. Murphy, "Multisensor Multitarget Recognition and Tracking," 13th Asilomar Conf. on Circuits, Systems and Computers, November 1979.

SESSION 7 GENERAL AVIATION AVIONICS

7

Chairman:

Myrl W. Kelly
Beech Aircraft Corp.

This deals exclusively with the applications and issues regarding digital avionics in general aviation aircraft. Topics include data busing, testing, crew interface, effects of precipitation static, and EFIS.

84-2637

GENERAL AVIATION DATA BUS UPDATE

David L. Stanislaw

Advanced Electronic Engineering
Cessna Aircraft Company
Wichita, Kansas 67277

Abstract

Ever since the Signal Corps installed the first radio in an aircraft in August of 1910, we have been adding more and more black boxes to accommodate the increasingly demanding requirements of the modern day aviator. Not only do we need electronic devices for air to ground communication, but complex computers have also found there way into today's cockpits, as well as the electronically controlled jet engine. The trend towards complexity never seems to rest. Developments such as the transistor and the integrated circuit have made these advances possible. While electronic engineers have dramatically changed the appearance and performance of the inside of these boxes, they have largely ignored the wiring between the boxes. This paper discusses the efforts that are under-way to eliminate the wiring complexity now used in modern aircraft.

Technology Evolution

In the beginning Avionics Manufacturers used vacuum tubes in the design of their equipment, like the RDR 1 Bendix Radar. Since this equipment was relatively complex, multiple black boxes were used throughout the aircraft to contain all the necessary circuitry. Therefore, considerable inter wiring was necessary. Sub-chassis with vacuum tubes was the norm for the day. Most tubes were single sections, however some bottles contained multiple elements. The RDR-1 Cable Diagram shows 73 interconnects for the radar alone. Most equipment stood alone and rarely interconnected with other equipment or instruments.

Semi-conductor technology became available in the early sixties and transistors were widely used throughout the seventies. This was the first opportunity to use Data Bus techniques, since this circuitry could support the necessary protocol to begin reducing aircraft cable weight. But by this time, the avionics designers were accustomed to specifying wire at will, therefore; the transistorized systems were just as complicated to install as the tube equipment. In the late seventies integrated circuits became widely available and this was another occasion to use Data Bus techniques in the aircraft, however; few designers took advantage of this opportunity.

Aircraft Utilities

In addition to the classical avionics equipment, modern aircraft contain additional electronic equipment that controls or regulates the airframe utilities. These utilities include pressurization, flaps, environmental, and deice type systems. The cabling burden for these utilities is just as excessive as the avionics system.

Modern Factory Installation

Now let's take a moment to describe how we operate today. At Cessna, we make computer graphic models of each avionic system schematic that is expected to be used in our aircraft. All interconnects are stored in a digital form, and can later be extracted for compiling. Since there is such a wide variety of avionics that can be specified, and in different configurations with other equipment, we maintain a file of approximately 1,800 avionics system schematics that can be mixed and matched in accordance with what the customer ordered.

Citation III will use approximately 25 individual avionics system schematics. A computer searches through these schematics to make the proper interconnects and to pick out only the wiring for a certain section of the aircraft since each system schematic contains end to end aircraft wiring. It is compiled and printed out in a form that is convenient for production to use in fabricating the nose, fuselage, and tail cable assemblies.

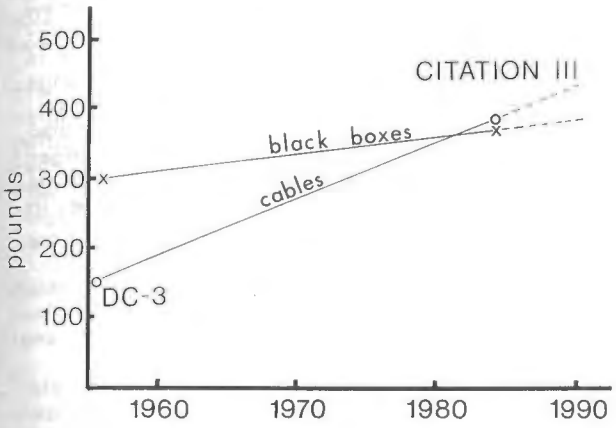
Most avionics and utility equipment connect in some in some way to switches or instruments in the cockpit. The area behind the instrument panel usually has the most dense cabling of any area in the aircraft.

The Avionics and Electrical Planning Group, in the factory, is responsible for compiling the system schematics into cut and solder sheets that the factory uses in fabricating these cables. The first step in fabrication, is marking each wire with the appropriate number, and cutting it to the appropriate length. The Wire Shop uses the cut and solder sheets, made by Planning, to build up each cable assembly on a fixture, that represents the geometry of that section of the aircraft, however, in many cases the aircraft configuration changes at the last minute and equipment is either added or deleted. Rewiring causes waste. After each cable assembly is fabricated, it is subjected to a complete ring-out. The Hughes computerized wire analyzer looks for proper continuity, opens, and shorts on every wire. A high-pot test for sensitive leads is also available. This automatic wire ring-out equipment can do a job in ten minutes, that would otherwise take two men 24 hours with an ohmmeter.

Wiring Comparison

Twenty-five years ago, corporate operators were flying DC-3's, and the like, for company transportation. Remote equipment was ininstalled, where the box weight averaged 300 lbs, and the wire was 1/2 that amount. Over the years, black boxes have become more sophisticated, where you can now buy five

times the utility for the same weight. However, the wiring burden, as shown for the Citation III, has grown to more than double what it once was, even with our new fancy integrated circuits. We have now tipped the scales, to where the cables in a modern corporate jet, weight more than the black boxes it serves.



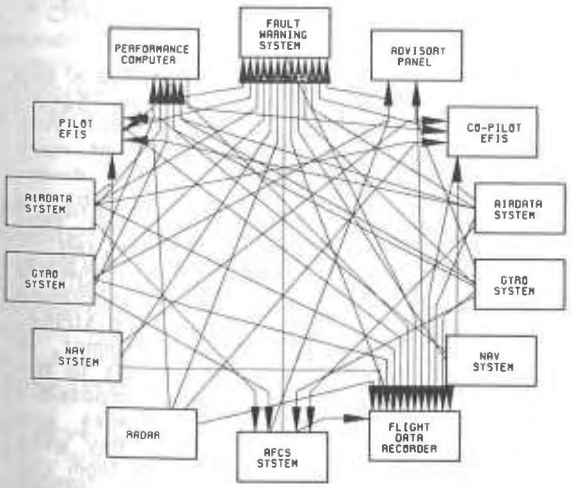
Recent Broadcast Bus Efforts

Numerous organizations have been working on Data Bus technology to simplify aircraft wiring. ARINC and Collins have developed a "broadcast" bus where each unique signal is transmitted to every user, by its own dedicated twisted shielded pair.

ARINC 429

Was approved in September of 1977, and it is the airline Data Bus standard for the Free World. Virtually every airline and airline equipment manufacturer subscribe to this system. Some equipment that is in design for business aircraft will also use this form of Data Bus, but in spite of its popularity, 429 does not offer the wire weight savings that the airframe companies are looking for.

DIGITAL AVIONICS INTERFACED WITH A BROADCAST BUS



Collins RS422

Data Bus System was adopted in December of 82 and it is a unique Data Bus System that is useful only for Collins equipment. It does not replace a substantial amount of inter-box discrete wiring, therefore, it does not offer the desired wire weight savings.

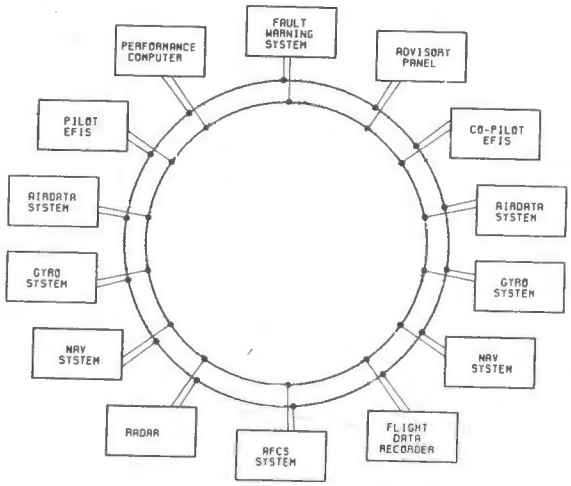
Recent Bi-directional Efforts

Sperry ASCB and MIL-STD-1553B use the "bi-directional" technique, where a single cable can transfer data in both directions.

Sperry ASCB System

Was adopted April of 1982, and it has been installed in several aircraft models in various configurations. It takes advantage of the commercial High Level Data Link (HDLC) integrated circuits that are available from numerous manufacturers, making the ASCB cost effective.

DIGITAL AVIONICS INTERFACED WITH SPERRY ASCB



MIL-STD-1553B

Approved in September of 1976, it is virtually used in all US military aircraft for avionics and electrical systems. It has many features that are desirable in a general aviation data bus system, but we have found that its protocol is extremely complex, making it too costly for our use. No commercial integrated circuits are available or planned, therefore, the price of this system will be high forever.

SAE has just completed a draft for MIL-STD-1773 which adds a fiber optic capability to 1553. They also have another committee (AE-9B) working on a High Speed Data Bus that is expected to operate in the 50 MHz range, a replacement for MIL-STD-1553. Three concepts have been reviewed:

Token Bus (proposed by Dassault, Hughes, RCA and Sagem)

Token Ring (proposed by FMC)

Switch Network (proposed by General Dynamics)

The Evaluation Board of 14 chose the Token Ring as the basic technique to write a spec around but the voting committee of 96 couldn't achieve a 75% plurality; therefore, the technical subcommittee will prepare a spec for both Token Ring and Token Bus.

Recent IEEE Efforts

In addition to these "Aviation" data bus systems, the computer industry, with the help of IEEE, has developed several advanced distributed control bi-directional data bus standards, mainly for ground based equipment. These are:

IEEE 802.3 Carrier Sensing Multiple Access with Collision Detection the CSMA/CD network, that is favored by Hewlett Packard, provides random access for each user. The message length can be quite long, however; most avionic messages are short, and very periodic in nature, therefore this method is not ideal for the aircraft environment. The distributed control, however does look appealing.

IEEE 802.4 Token "passing", is somewhat better, in that each user can depend on periodic or regular access, and it is a linear type network therefore, any user can be "unplugged", without effecting the remaining users.

IEEE 802.5 Token "ring" network basically hooks all users in series, where each terminal "repeats" the message of the last user. This is advantageous where long distances are involved, however, it is not desirable for aircraft, since one bad box can shut the whole system down.

IEEE 802.6 Metropolitan Area network was originally proposed as a time-division multiplex access standard, but the company that proposed it has gone out of business and this committee is currently in limbo.

Data Bus Requirements

The general aviation airframe companies have been watching these developments over the last several years, in an attempt to determine which technique would best serve the modern airplane. Several objectives were established:

1. The data bus we need today must provide for control and information transfer of all avionic and utility electronics systems that might be used over the next 25 years on general aviation aircraft. These systems include radio, flight control, guidance, annunicator, flight management, air data, electrical power, environmental, fuel, engine instruments, rain, ice and ground systems.

2. The future bus should have substantial room for label expansion, since a proliferation of new avionics systems will probably continue for the next 25 years, and maybe forever.

3. Future bus protocol and architecture should permit the use of reasonably priced commercial hardware for input-output circuitry.

4. Future digital data bus should be bi-directional and if done correctly, can provide substantial weight savings in new aircraft. Current cable weight is averaging 374 pounds in the Citation III. It is expected that a 75% cable weight savings can be achieved with a new data bus, which would increase the useful load enough to to allow one more passenger plus baggage to board. Other models would realize a proportional increase in their useful loads.

AIRCRAFT CABLE WEIGHT
WITHOUT DATA BUSING

MODEL	WEIGHT LBS
414	170
441	251
550	224
650	374

5. The hierarchy, architecture, and protocol of any future systems should be compatible with both wire and fiber optics. Wire may be preferred until 1990 because of familiarity, however, fiber optics will gradually replace wire, as it has in the telephone system, mining, robotics and nuclear power plants. Fiber optics offers good immunity from RFI, EMI, and lighting upset.

Lightning Consideration

Electronic equipment in our current all metal aircraft is relatively immune from catastrophic failures due to lightning strike, since conducted and induced voltages inside the aluminum fuselage during a 200 K amp lightning strike, have generally been at tolerable levels. Soon the all-composite aircraft will be here, therefore, steps must be taken to provide electrical isolation from the horrendous voltages that will be induced into solid state electronic equipment, by a lightning strike.

Last year the F18 was subjected to 200 K amp simulated lightning strikes at the Sandia National Laboratories in Albuquerque, New Mexico. A hanger housed a series of marx generators, that have the capability of developing 2 million volts. This simulated lightning charge was imposed on the aircraft nose, tail, wings, etc. The Sandia Lab attempted to measure voltages and currents at numerous places through the fuselage and wings during the presence of this simulated lightning strike. They found however, that any attempt to transmit these signals from the aircraft to the ground recording equipment, was impossible over wires, due to the high voltages and currents imposed on the aircraft. They were forced to replace the wire data lines from the aircraft, with fiber optic cables, to successfully record electrical events in the aircraft. This exercise demonstrates the immunity fiber

optics has from high currents and voltages associated with lightning. Bell labs and numerous other organizations have recognized the benefits that fiber optics offer to communication, computers, and other systems that require noise free data transmission. Fiber optics transmitters and receivers are now becoming commonly available from numerous companies and competition is driving the price down to reasonable levels. This technology is well established now, it's time to use it.

The forward center console of the Citation III weighs 6.75 pounds and contains twenty eight switches, as well as several annunicator lights. Each particular function has its unique switch; that is to say.....single pole, double pole, or single throw, double throw, as well as any mechanical latching mechanisms.

Each switch is connected to a short pigtail that terminates in one of four multiple pin connectors that feed the aircraft wiring harness.

This switch console could be substantially simplified through the use of Digital Data Bus technology, where all switches are connected to a multiplex integrated circuit, via printed circuit board. The multiplex integrated circuit would only require power, ground, and the data bus connection to the aircraft wiring. Of course in many cases, redundancy would be required for reliability, should one data bus fail. This could be accomplished with dual data bus architecture, where each switch would be a double pole version, and each pole connected to its own multiplex circuit. This provides for simultaneous commands to be fed to both Data Bus A and B. Should one data bus fail, commands would still be routed through the operating data bus.

This proposed switch console, with dual channel data buses, would be obviously simpler to manufacture and much lighter than the current Citation III switch console. Each switch would be directly attached to a printed circuit board, which would connect to the data bus integrated circuit. The data bus integrated circuit would be programmed with the appropriate labels for each switch, to send each command in the appropriate time slot. Remember the weight of the Citation III switch console is currently 6.75 pounds, this new assembly would weigh approximately 2.75 pounds. As one can imagine, it is obviously much simpler to manufacture and probably easier to service.

Committee Established

A petition was sent to the General Aviation Manufacturers Association (GAMA) requesting the formation of a committee to address the data bus questions, and hopefully choose or create a standard that would be useful in general aviation. The Executive Board, did just that, and an Ad Hoc Committee was formed to address these issues.

I serve as the Chairman. The initial meeting was held in January at Kansas City; since then meetings have been held at Dallas, Phoenix, Minneapolis, Cedar Rapids, and Fort Lauderdale.

To date three tasks have been identified:

TASK I

The current "broadcast" type airline standard, ARINC-429, has official labels and data content definitions for all the parameters used on commercial aircraft. Several avionics companies have also started using this standard on their general aviation equipment, and have created new labels that are unique to this service. Each company was assigning labels and data content definitions without knowledge of what others were doing, sometimes assigning different labels to the same message, other times using different scaling factors in the data. The committee decided, as a first step, it would review all these unique general aviation labels and work out a common list. Then a request went to ARINC, asking them for guidance in this endeavor, and petitioned them to publish these labels in their table. They graciously accepted.

TASK II

Sperry has developed a "bi-directional" data bus standard called the Avionics Standard Communication Bus (ASCB) over a period of six years. Initially, this bus was intended for Sperry equipment only, however, it occurred to other members of the committee that it should be made available to everyone, should they decide to "connect" up. Therefore, the committee requested, and Sperry responded with a detailed briefing and spec handout in May of this year. Several beneficial changes to the architecture were proposed by the committee and they are being worked out now.

Labels for additional functions and their data formats were also added to the existing specification.

TASK III

With all the other data bus efforts in work, by the avionics and computer industry, the committee felt it should investigate them to determine if one could become the "ultimate" business aircraft standard. SAE, in the AE-9B committee is creating the Bi-directional High Speed Data Bus for the Military. Boeing has been working on their Digital Autonombus Terminal Access Communication Data Bus, called DATAC, since 1981. It too is "bi-directional", with several novel features.

Maybe one of these efforts will result in a system that general aviation can use as their future standard. If not, possibly we'll create one.

I am very optimistic about this committee, and expect its efforts will provide us with the tool to make substantially lighter cable assemblies in future aircraft.

SYSTEMS CONCEPT FOR SPEECH TECHNOLOGY APPLICATION IN GENERAL AVIATION

Robert A. North
Honeywell Systems and Research Center
Minneapolis, MN

Hugh Bergeron, NASA Langley Research Center
Hampton, VA

December 1984

Abstract

Single pilot IFR flight often presents the pilot with difficult workload challenges, especially in deteriorating weather conditions and unfamiliar surroundings. Visual switching between the outside and inside world, and manipulating switches, keyboard devices, and rotary controls while attempting manual flight present particularly high workload situations. Interactive speech technology, using speech recognition and synthesis offers a promising method of aleviating the above workload problem. The study to be reported on in this paper focussed on identifying the most promising applications of this technology for the general aviation single-pilot IFR cockpit.

A high performance, twin-engine aircraft with typical current avionics and a future avionics system was the subject of the study for voice-interactive applications. Flight scenarios were studied to determine the pilot task demands. A set of candidates for speech technology was identified for in-depth study. Tradeoffs are discussed concerning the utility of each application from a pilot's point of view vs. application implementation feasibility.

Introduction

The single-pilot IFR cockpit is one of the most challenging and difficult general aviation scenarios, especially under the increasing traffic load of busy terminal area operations. The availability of new avionics devices that will allow access to new data sources in flight, more complex area navigation capabilities, and expanded weather radar options require dedicated pilot interaction to adequately explot their power during flight. Manual and visual information channels are taxed heavily during single-pilot IFR flight under normal conditions. The goal of designers of future IFR cockpits must be to make the pilot transactions with new systems as easy and error-free as possible while allowing the pilot to attend to normal cockpit tasks.

Military cockpit research and development laboratories have been exploring the utility of speech recognition and synthesis devices as methods of reducing the manual/visual workload problems in single-seat advanced fighters. McCauley, Cotton, and North (1982)[1] provide a review of speech application candidates in a recent study of Air Force air-to-ground fighters and North and Adsit (1983)[2] have reviewed applications for a single-seat advanced attack helicopter. The recent flight testing of such devices on the AFTI-F-16 program by the Air Force is evidence that flightworthy systems are developing rapidly.

Extension of voice recognition and synthesis technology to the general aviation cockpit will be possible within the next five years due to the increase in microprocessing capability, decrease in system costs, and miniaturization of components. Voice output systems are already available and in use in commercial aircraft. Successful application in the general aviation IFR aircraft will depend upon several key factors. These factors, stated as criteria for application development are: (1) The application(s) must derive some positive pilot benefit (e.g., reduce workload, save time, increase efficiency); (2) The voice-interactive system must be easy-to-use, and not create additional workload within any task; (3) The voice-interactive system must not interfere with critical information flow (e.g., communications with ATC); and (4) The system must be affordable and cost-effective within the applications it addresses.

This study explored the application potential of speech technology to the general aviation single-pilot IFR (SPIFR) scenario. The methodology used in asessing application strengths and weaknesses follows techniques developed by North and Lea (1982)[3] in an earlier Air Force study of B-52 applications.

Task analysis, workload analysis, and pilot assessments were used to identify the strongest voice applications from a pilot viewpoint. The technological feasibility of the application was then assessed by employing knowledge of current and future speech system capabilities. This provides a tradeoff analysis capability to guide system developers and research development teams toward the most beneficial set of candidate applications for SPIFR cockpits.

Study Methodology

Subject Aircraft Description

A twin-engine, high performance general aviation aircraft was the subject for data reported in this study. Both baseline and future avionics suites were studied. The baseline aircraft avionics included the equipment instrumentation package shown in Table 1. This list was derived from a pilot survey conducted by Weislogel and Chapman (1982)[4] for NASA to determine the most likely set of IFR avionics for this type of aircraft. The future cockpit included a multi-function display and control system with the capability of sophisticated route planning, systems status, and checklist retrieval. The Demonstration of Advanced Avionics System (DAAS) being evaluated and tested by NASA-Langley Research Center, and originally developed at NASA-Ames, will be used as an example of our "future" system.

TABLE 1. AVIONICS EQUIPMENT PACKAGE FOR STUDY AIRCRAFT

Baseline Avionics Package

- Conventional Flight Controls
- Conventional Engine Instruments
- Conventional Flight Instruments
- Flight Director
- Autopilot
- Pressure Altimeter
- Transponder
- Glideslope Indicator
- Variable Omni Range (VOR)
- Distance Measuring Equipment (DME)
- Automatic Direction Finder
- Radio Magnetic Indicator
- Horizontal Situation Indicator
- Directional Gyro
- Area Navigation Equipment (RNAV)
- Communications Transceivers (2)
- Anti-Icing and Deicing Controls

Future Avionics Package

- Radar Altimeter
- Waypoint Programming/Editing
- Checklist Retrieval
- Performance Chart Retrieval
- Electronic HSI (Map)
- Flight Warning/Advisory System
- Uplink of Messaging/Information

Initial Screening for Voice Candidates

An initial screening of possible voice interactive candidates was obtained by a detailed analysis of typical single-pilot IFR flights. A recent study by Parker (1982)[5] was used to define a timeline of events for a cross-country flight with takeoff and landing at busy terminal areas. This scenario was augmented events by Honeywell-corporate IFR-rated instructor pilots to ensure completeness of the event history. The complete list of pilot tasks was condensed to the list appearing in Table 2.

History has indicated that voice control is more suitable for discrete tasks where the possibilities are limited and finite, than continuous tasks where precision adjustment is required. Thus, we eliminated actual control movement adjustments from consideration in our initial screening.

Voice display or messaging to the pilot has been shown to be effective for directing attention to specific problems, confirmation of actuations, and as a way of unburdening the visual channel for certain classes of information. It is not useful in conveying information that must be remembered or referred to continuously, or conveying complex spatial relationships such as navigation charts, etc. The short, simple transmission of verbal information is ideal for voice output.

The above screening criteria were applied to the IFR flight scenario and produced 27 voice recognition and 18 voice synthesis candidates. These candidates were subjected to the evaluation procedures described below.

Evaluating the Pilot Utility of Voice Applications

Each of the candidate applications was subjected to a rating scheme to determine the pilot utility of voice control or display for that task. The pilot utility attributes for Voice Recognition were:

1. Workload Reduction addressing the potential avoidance of attention diversion from competing manual control tasks and/or simplification of executing the task by voice.

2. Time Savings addressing the potential improvement over present task execution methods.

3. Communication Disruption addressing the possibility that voice input may occur at a time when radio transmissions are taking place.

4. Cost of Error addressing the detrimental effects of having to repeat execution of the task due to voice system error.

For voice synthesis, Time Savings was changed to time saved in recognizing a critical or hazardous situation and Cost of Error was changed to cost of misunderstanding message.

Three IFR-rated instructor pilots used the above rating scheme to derive scores on the Pilot Utility dimension. The four attributes were evaluated on a -2 (low utility) to +2 (high utility) scale. For example, for workload reduction, a -2 would indicte a large increase in workload, while a +2 would indicate a large decrease in workload, and thus, high utility for voice implementation.

A total score was computed for each application by forming a linear combination of the scores on each attribute. A higher weighting factor (2.0)

Implementation is technologically easy. was applied to Workload Reduction because we believe this to be the strongest criteria for choosing whether the candidate has utility for increasing safety in SPIFR operations. The weights for all other factors were 1.0.

Evaluating Implementation Feasibility

Each voice application candidate was evaluated for factors pertaining to the ease of implementation of voice technology for that application. The criteria used for recognition were:

1. System speech requirements referring to the need for a simple isolated word system, or a more complex system such as connected or continuous speech.

2. Speech I/O software support referring to the complexity of supporting software to handle feedback, error correction, or intelligence to extract the meaning of an input by the pilot.

3. Electromechanical Linkage referring to the potential ease or difficulty of interfacing an external command signal from the voice recognition system to the designated avionics system. For the purposes of this discussion, we assume an augmented avionics system with interfaces such as those in NASA's DAAS cockpit.

4. Frequency of Occurrence was added to the list of implementation attributes to filter out those tasks that occur infrequently, making their implementation less cost-effective.

For voice synthesis, the System Specification attribute was changed to asses requirements for a fixed vocabulary vs. a flexible, multi-alternative system. (Implementation costs will increase if the more flexible type of system is needed.)

A weight of 2.0 was applied to the Electro-Mechanical Linkage attribute. This was seen as the most important attribute in determining implementation feasibility. Weights of 1.0 were applied to all other attributes.

Experts in speech technology application design, avionics design, and human factors collaborated on the rating of the above attributes for system implementation feasibility.

Results and Discussion

The results of rating the speech I/O applications on pilot benefit and implementation feasibility are presented in Tables 2 and 3 for voice recognition and voice synthesis, respectively. These columns show the mean score across the three raters for each dimension.

Voice Recognition Tradeoffs

Data generated by the process described in this paper can be used to make preliminary decisions about the priority of application implementation. The analysis is useful for tradeoff analyses that should be made prior to designing a system. We can examine the position of the applications on both dimensions, and seek the candidates that are high on both scales. These would constitute the top priority set for future development.

For voice recognition, the high utility, high feasibility candidates center around capabilities that would be provided with the on-board computing power of the future avionics suite that includes a DAAS-like system. Individual points regarding specific applications are discussed below.

1. Interactions with uplinked data by voice command request seen as highly useful, and could be implemented with a low-cost isolated word system using a modest vocabulary (winds aloft, pilot reports, weather bulletins, etc.). Minimal software change required.

2. Retrieval of checklists/performance charts useful because of elimination of searching during high workload segments. A small vocabulary with limited connected speech could be used. Software changes would be minimal.

3. Requesting direction to facility or present location seen as highly useful when pilot has lost orientation and must recover quickly. Implementation dependent on navigation system information. Vocabulary very simple to implement. Some software changes needed to format requested information.

4. Autopilot interchanges were moderately useful, and would require limited connected recognition to be practical, reducing feasibility for these applications.

5. Tuning of VOR/DME to specific stations was highly useful, and could be implemented with a number of two-step isolated word commands. Use of facility names in vocabulary instead of entering particular frequencies increases feasibility but requires more complex software to perform table lookup to insert frequencies.

6. Tuning communications radios was only moderately useful, and would require limited connected recognition to be practical. Again, the preset concept would be needed to increase ease of pilot interaction.

7. Selection of a waypoint, although very easy to implement with a simple isolated word command, was not seen as an improvement over present methods.

8. Selection of a course to intercept was moderately useful, but will require some moderately complex software to handle incoming data. Connected phrase recognition (e.g., "Intercept 330 radial of MSP") would be needed with a fairly large vocabulary, decreasing feasibility.

9. Setting of directional gyro and pressure altimeter were seen as highly useful, but are less feasible because of the need for connected digit recognition and/or present lack of electro-mechanical linkage to the avionics bus.

10. Recording of reference information (eg., using voice-store and forward capability as a verbal scratchpad) was not seen as useful due to

TABLE 2. RECOGNITION APPLICATIONS RANK-ORDERED ON PILOT BENEFIT FACTORS

APPLICATION	PILOT BENEFIT SCORE	IMPLEMENTATION FEASIBILITY SCORE
High Pilot Benefit Group		
1. Interface with uplinked data	6.0	3.0
2. Reprogram waypoint	5.3	3.0
3. Retrieve performance charts	5.0	4.0
4. Retrieve approach charts	5.0	3.0
5. Retrieve emergency checklists	4.6	3.0
6. Set up a course intercept	4.0	1.0
7. Retrieve routine checklists	3.6	4.0
8. Interact with electronic map	3.6	5.0
9. Request direction to facility	3.6	4.0
10. Request present location	3.3	4.0
11. Tune reset VOR/DME facility	3.3	3.0
Medium Pilot Benefit Group		
12. Tune communications radios	2.3	2.0
13. Set pressure altimeter	2.3	-2.0
14. Set/change autopilot heading	2.0	2.0
15. Set/change autopilot altitude	2.0	2.0
16. Set/change autopilot air speed	2.0	1.0
17. Set HSI heading bug	2.0	0.0
18. Set directional gyro	2.0	-2.0
19. Set transponder code	2.0	0.0
Low Pilot Benefit Group		
20. Set/change rate of descent/climb	1.6	2.0
21. Select new waypoint	1.6	5.0
22. Engage/disengage autopilot	0.6	5.0
23. Control external aircraft lights	0.6	0.0
24. Switch fuel tanks	-0.3	1.0
25. Change landing gear state	-0.3	-2.0
26. Control internal aircraft lights	-0.3	0.0
27. Set flaps	-0.6	-2.0

TABLE 3. SYNTHESIS APPLICATIONS RANK-ORDERED ON PILOT BENEFIT FACTORS

APPLICATION	PILOT BENEFIT SCORE	IMPLEMENTATION FEASIBILITY SCORE
High Pilot Benefit Group		
1. Report direction to facility	4.6	1.0
2. Announce fuel level increments	3.6	4.0
3. Announce approaching altitude	3.6	4.0
4. Announce altitude deviations	3.6	2.0
5. Call out altitudes on approach	3.3	3.0
6. Call out airspeeds on takoff	3.3	4.0
7. Announce system warnings	3.3	-1.0
Medium Pilot Benefit Group		
8. Alert to approaching stall speed	3.0	3.0
9. Announce course deviation errors	3.0	2.0
10. Read checklist item-by-item	3.0	2.0
11. Alert nearing maximum cruise speed	3.0	3.0
12. Announce rate of descent deviations	2.6	4.0
13. Announce altitude deviations	2.6	2.0
14. Read instrument approach procedure	2.6	2.0
Low Pilot Benefit Group		
15. Announce approaching course	2.0	3.0
16. Announce approaching heading	2.0	3.0
17. Alert nearing never-exceed speed	1.6	3.0
18. Announce autopilot engage/disengage	1.3	4.0

the time required to input the message. Also, this method would eliminate the visual reference provided by conventional scratchpad.

11. Control of aircraft lighting was neither useful nor feasible. This task would require too many presets (dim to bright) for the internal lights. Frequency of occurrence too low.

12. Flaps and gear settings were seen as simple enough in their present manual configurations. Also, pilot verification is very important in the execution of these tasks, and there is present reluctance to trust these to the voice modality.

Voice Synthesis Tradeoffs

Voice synthesis is a much less complex technology to implement in the aircraft cockpit and there are several existing commercial and general aviation aircraft with this capability. Therefore, decisions regarding what tasks to implement should primarily be driven by pilot factors. Memory costs and speech encoding techniques will make voice output a very practical low-cost add-on within the next few years.

Individual points regarding applications are discussed below.

1. Simple alerts, announcing the approach of some aircraft state or an out-of-tolerance condition were generally seen as having high utility because of their attention focusing nature, and allowing early detection of deteriorating conditions. They would generally require a fixed, simple vocabulary stored in digital form.

2. Reporting direction to facility (e.g., "Des Moines is 50 miles at 330 degrees", etc.) was seen as highly useful, but will require additional software to collect and format information from the navigation system. Also, a fairly large vocabulary would be needed for this application.

3. Callout of altitudes on approach was rated useful, which is consistent with an available system utilizing a synthesizer for 100 and 50 foot interfal enunciations.

4. Announcement of system warnings was judged useful, but would involve the insertion of a voice capability in each avionics device unless each system were linked to an electrical bus. Therefore, implementation feasibility would be lower until this link is more common across aircraft.

5. Readoff of each checklist item was seen as highly useful, but primarily for checklists that are not normally used (e.g., emergency). Otherwise, this may slow down the checklist process unless individual items are kept very short. System implementation would require a large vocabulary, but messages would be fixed and software changes minimal which increases the feasibility of this application.

Packaging the Voice System for SPIFR Aircraft

The results of this study indicated that a majority of useful applications of voice technology would reside in the more sophisticated systems that are becoming available to the general aviation pilot. Ease of implementation of voice display and control will depend on what signals are linked to these new avionics systems. The most useful packaging strategy for a total voice system would seem to be a centralized, integrated system possibly housed in the CPU of the long-range navigation system or other on-board systems with sufficient computing power. This would seem more cost-effective than retrofitting each individual avionics device with its own voice capability.

The centralized method offers the additional benefit of standardization of pilot-systrem transactions through a dedicated set of software routines that can extract the meaning of each pilot command and translate the command into the appropriate action. This "transaction-processor" will pose a significant design problem that would best be attacked by a design team including a linguist, aviation human factors engineer, and software engineer.

Conclusions

The major conclusion of this study is that voice recognition and synthesis would offer benefits to the single-pilot IFR cockpit by reducing the demand on manual and visual channels, especially in high workload segments such as takeoff, approach, and landing. The most beneficial speech command tasks appear to be centered in the data retrieval domain, which would allow the pilot access to uplinked data, checklists, and performance charts. Data entry tasks, such as changing the status of communication and navigation radios and reprogramming autopilot settings also appear beneficial because of the elimination of diversion of manual and visual attention to the flight task.

Speech synthesis, already in use on commercial and some private aircraft, appears to have a major contribution in focusing pilot attention on problems that may develop in the course of the flight, such as course errors, glideslope deviation, airspeed and altitude deviation, and fuel status. Automated readouts of airspeed and altitude would help reduce visual workload during takeoff and approach.

The primary technological problems in successful implementation of speech technology into the SPIFR cockpit will be in designing the pilot-system transactions to be as simple, and easy to use as possible. For speech command, this design problem is complicated by the lack of sophistication of current speech hardware, which requires pretraining of the system and short inputs of words or phrases. For speech output, the primary problem will be provision of sufficient computer storage for high intelligibility speech. Neither of these problems would appear insurmountable within the next five years.

The high-payoff application candidates for speech recognition and synthesis appeared in pilot transactions with the avionics provided in our "future" cockpit with the advanced multi-function display/control system provided by DAAS. Thus, it would appear that the most cost-effective method of implementing a speech capability in SPIFR aircraft will be inclusion of the hardware into a system like DAAS, or an integrated navigation system which will eventually have the power of addressing most of the avionics devices on the aircraft.

References

1. Cotton, J. C., McCauley, M. E., North, R. A., and Strieb, M. I. Development of Speech Input/Output Devices for Tactical Aircraft. Report AFWAL-TR-83-3073, Wright-Patterson Air Force Base, OH: USAF Wright Aeronautical Laboratories, July 1983.

2. North, R. A., and Adsit, D. Applications of Interactive Voice Technology for the Advanced Attack Helicopter. Honeywell Systems and Research Center Report No. 83-SRC-26, Minneapolis, MN, July 1983.

3. North, R. A., and Lea, W. A. Applications of Advanced Speech Technology in Manned Penetration Bombers. Report AFWAL-TR-82-3004, Wright-Patterson Air Force Base, OH: USAF Wright Aeronautical Laboratory, March 1982.

4. Weislogel, G. S., and Chapman, G. C. Statistical Summary: Study to Determine the IFR Operational Profile of the General Aviation Single-Pilot. NASA-CR-165805, NASA-Langley Research Center, Hampton, VA, May 1982.

5. Parker, J. F. Jr., Duffy, J. W., and Christensen, D. G. A Flight Investigation of Simulated Data-Link Communications During Single-Pilot IFR Flight. NASA-CR-3461, Washington, D.C., August 1981.

EFFECT OF PRECIPITATION STATIC ON GENERAL AVIATION DIGITAL AVIONICS

Leonard O. Hendry*

Group Engineer EMC/EMI/TEMPEST & Requirements
Military Aircraft Group
Beech Aircraft Corporation
Wichita, Kansas

Abstract

Particle precipitation static, or electric charging, can be highly disruptive of avionics operation in general aviation aircraft. As these aircraft become more capable, they also acquire more sophisticated avionics systems. Digital engine controls, fly-by-wire systems, VLF navigations sysems, digital displays, and memory devices are some of the systems that can be sensitive to precipitation static (also known as P-static). The use of nonconductive composite materials also accents the charging problem. This paper examines the effects of P-static in general aviation aircraft, how it affects digital avionics systems, and how it can be guarded against or dissipated in an expedient manner.

Introduction

Historically, precipitation static, or P-static, has not been a major concern to general aviation aircraft. With the advent of new and more sophisticated avionics equipment and airframes, this situation has changed. In order that the installed avionics equipment be able to function as expected, it becomes necessary to understand the manner in which P-static charging occurs, how it is distributed in the airframe in all flight regimes, the levels to which it can build, and how it can be dissipated in the least disruptive manner. Pilots must be educated to recognize P-static as it occurs and how it affects their aircraft. They must also understand the conditions under which it occurs. It is also important to understand the similarities and differences between P-static and a lightning discharge.

Nature of P-Static

When aircraft started using radio transmitters and receivers, a problem surfaced. At times, the radio's receiver would become unusuable because of static. This interference occurred when flying through particles in the air such as dust, ice crystals, sometimes moisture in clouds, or the charged fields between clouds. This interference is now known as precipitation static (P-static). The uncontrolled static discharge repetition rate can be as high as 35 kHz. This can cause the AGC of a radio to shut down its operation. The pilot will not know anything is wrong until he tries to contact a ground station. Studies were made to determine what caused this condition, resulting in the development of devices known as static dischargers (or static wicks). Static dischargers are now available from numerous sources. The type of static discharger to be selected is determined by the size and speed of the aircraft on which they are to be installed.

*Senior Member IEEE

Difference Between P-Static and Lightning

P-static charging is a distinctly different phenomenon than lightning discharges. P-static charges are created by the aircraft itself moving through the air. Lightning discharges occur because an aircraft enters an existing charged field. However, the conditions that are conducive to P-static charging are also, to a certain extent, conducive to lightning discharges.

Beech Experience

At Beech, additional problems with precipitation static interference began to surface soon after the first turboprop aircraft were produced. These were all-weather, pressurized aircraft which flew at higher altitudes where the temperature was lower. In general, the temperature around freezing (plus or minus a few degrees) is where maximum electrical charging occurs. Beech customers began noting lost communications while flying in or near thunderstorms. There would be times of several minutes that the communication or navigation signals were unusable due to electrical static discharge interference. The aircraft usually had been equipped with some type of a static discharger. However, they were placed on the aircraft mainly by guesswork near the extremes of the wings and tail surfaces. Little testing was done to actually verify the performance of these devices.

Testing

It became apparent that testing was needed to better understand the problems involved.

Initial Testing

One customer made his aircraft available to be used for testing. At that point in time, Beech had no high voltage testing capability. McDonnell-Douglas was put under contract for use

of their high-voltage facilities. The aircraft was placed on plexiglass isolating pads and was connected to one wire of a high voltage source. The other end of the high voltage source was grounded. The area around the aircraft surfaces were then examined by using a grounded sphere. The avionics radios, operating on battery, were monitored onboard the aircraft to see if they experienced interference as the grounded sphere was moved along the trailing edges of the control surfaces. When the grounded sphere was brought close to the outer trailing edge of the wing control surface, the avionics radios became unusable. Upon examining this area, it was found that an aluminum stiffener used in the end surface had a sharp point on the outermost end. There was also a 1/16th inch air gap at the other end of the stiffener, between it and the metal control surface. The sharp point went into corona, and the gap arced over, creating a spark generator or spark transmitter. The stiffener was changed to fiberglass, and one additional static discharger was added at the junction of the control surface and the fiberglass stiffener. This eliminated the interference problem on the aircraft.

Beech Conducted Tests

Beech then purchased their own high voltage equipment. A consultant, Mr. Bob Truax, was contracted for the first charging tests, of a Model 200 aircraft. During this test, such items as glass bead filled decals were found to take on a high potential charge. The streamer currents to the aircraft caused interference. Further work on gaps and the resulting spark transmitters was performed. A gap of 1/64th inch and 21 microamps of arc current can cause a low frequency navigation system with an "E" field antenna to go to dead reckoning. The "H" field antennas are not affected by this mode. New digital avionic equipment can be affected by the arc reoccurrence rate and degrade the data stream by placing extra or nonsynchronized pulses in the data stream.

Aircraft Configuration For Test

As knowledge of the phenomena was gained over several tests, it was determined that collectors as suggested by Mr. Truax should be used behind all surfaces. The current collected by them could then be measured and compared to the input current. A large difference of current level was measured. Then, using a noncontact electrostatic voltmeter, the charged aircraft voltage was measured. A very nonuniform voltage distribution was found, with as much as 30,000 volts difference between measured areas of the aircraft. Upon careful examination of the voltage distribution, it was noted that the voltage on the wheels and landing gear doors was always highest. The earth ground acted as one plate of a capacitor, and the greatest electrical stress was on the components nearest the ground. This is a configuration that is not typical of in-flight P-static discharges. It should be noted here that peculiar problems can occur when a charged aircraft is in landing configuration. In general, the landing gear will not be the point that first discharges to ground. Round tires do not discharge readily. The first point to discharge is apt to be a belly-mounted antenna. This can subject the associated radio to high voltage stress. We devised a test fixture to allow the landing gear to be retracted for testing. Transformer ceramic insulators were procured, and corona guards were placed on top of the insulators to fit the jack points of the aircraft. This allowed it to be raised off the ground so that the wheels could be retracted and doors closed. The voltage distribution was then uniform over the entire aircraft. Another benefit resulted -- the input current and that current absorbed by the collectors were almost equal. It was determined there had been streamering current over the plastic pads when using the previous isolating methods. With this configuration, and the input and collected current being almost equal, it was possible to determine the current required to cause radio interference. This test method proved that the low wattage, small, general aviation dischargers would not withstand the current being discharged.

Test Results

Beech defined, through testing, the discharge current level for their turbine engine powered aircraft. Two manufacturers were contacted to develop the required 300 microamp static dischargers to Beech specifications. These units were then tested in the Beech EMI Lab to determine both their electrical quieting (P-static discharge) capability and their ability to withstand the 300 microamp current for long durations without opening up. It was also demonstrated that the result of an open static discharger is like a spark generator and is worse than no static discharger being installed at all.

Tests With New Static Wicks

After the development of the new dischargers, Beech tested several aircraft to determine the discharge areas of the aircraft without static dischargers. To determine that the most outboard discharger on the wing and tail control surfaces would not be damaged by vortex wind currents, tufts were installed on each model. These were then flown, followed by a chase plane, and observed to determine optimum position of static dischargers in the air stream. The spacing of static dischargers should not exceed twice the length of the static dischargers for best electrical quieting; usually about four static dischargers on each outer control surface is sufficient for aircraft of this class.

Design Precautions

Other areas were also examined for corona discharge and placement of static dischargers. On Beech aircraft, the thin ventral fin trailing point must be protected. In general, any sharp extremity can be a suspect area. With the new composite material now being used to manufacture aircraft structures and outer surfaces, additional protection is required. All nonconductive material exposed to triboelectric charging must have a conductive coating type of paint used as the inner coating and must be

electrically connected to the conductive parts of the aircraft. Graphite-epoxy (GR/E) composite material is conductive enough not to be a problem; however, all surfaces must be electrically connected to prevent small electrical arc transmitters. With the higher resistance GR/E materials and the advancement of digital electronic display cockpits, new techniques must be developed to keep impulsive voltage or magnetic transients from being coupled into digital data lines. The design of new avionic systems must also consider the difference in potential between various installed equipment. This increased voltage is developed by the high resistance of GR/E structure. This becomes very important when considering possible results of a lightning strike to the airframe.

Summary

When proper attention is not given to the effects of precipitation static on today's higher performance aircraft, it will seriously degrade the usefulness of that increased performance.

CURRENT AND FUTURE GENERAL AVIATION EFIS DEVELOPMENTS

James A. Ogann

Program Manager, Flight Instrument Systems
Collins General Aviation Division
Rockwell International Corporation

Abstract

General aviation EFIS systems have become the most popular avionics topic in recent months. Airframes and avionics vendors are clearly concentrating on EFIS. Starting with the air transport systems, general aviation has taken a slightly different path to the all-digital aircraft (which is difficult to avoid addressing when EFIS is discussed). From the various formats of electromechanical instruments, EFIS has grown to a level that truly is now taking advantage of the technology. Coupled with that development is a very progressive attitude by the FAA and other regulatory agencies. The gestation period is over and the stage is set for the next generation of the "glass cockpit". The following discussion touches on both the past developments and the future trends in EFIS.

Electromechanical Displays

Flight instrumentation began with electromechanical displays. Early systems integrated the localizer, glideslope and compass data into one instrument. As time passed, other combinations became popular. The standard was finally established in the 1960's when the airlines and the military started to use the well-known ADI (Attitude Director Indicator) and HSI (Horizontal Situation Indicator). Several companies produce these instruments today and they are the basis for the first EFIS systems.

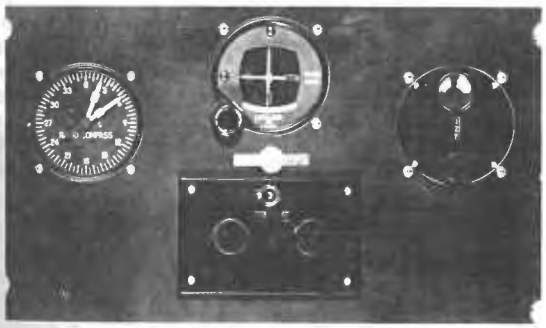


Fig. 1 Early LOC, GS Display

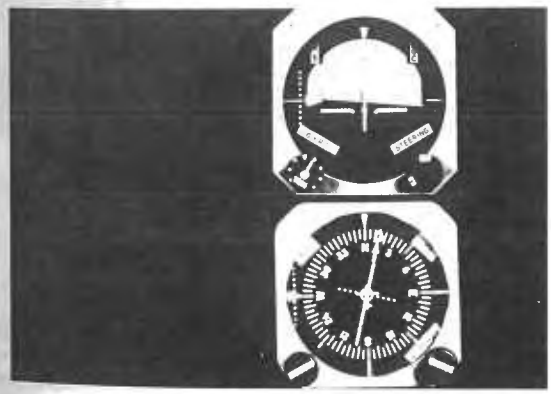


Fig. 2 FD-105

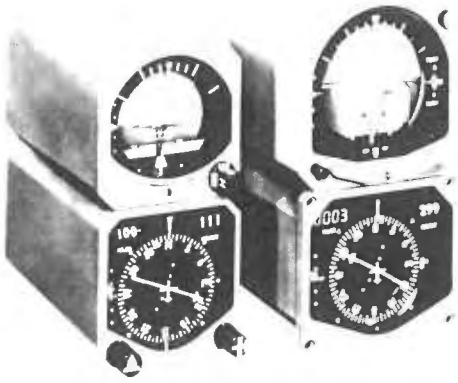


Fig. 3 FD-108, FD-109

Early CRT Development

Since the early days of commercial television, pilots and avionics engineers have been dreaming of the "Glass Cockpit." Raster scan systems were never bright enough and could not provide the resolution required for the primary flight instrument systems. However, several raster-based applications found their way into the cockpit in other systems. The most popular use of raster-based systems is in the navigation system CDU applications and weather radar. Though the requirements for brightness and resolution in these applications is not as great as in primary flight instrumentation, their widespread use has resulted in several important accomplishments. Operational experience has aided development of hardware, software, man-machine interface and pilot acceptance of CRT's in the cockpit. These accomplishments were necessary ingredients in acceptance of EFIS today. As the technology became available, EFIS was more readily accepted than most of the industry would have predicted.

Fig. 4 Early Monochrome Raster HSI

Color CRT Displays

Building on the acceptance of CRT's in the cockpit, the technological advances that followed made EFIS for general aviation aircraft a reality in 1982.

Color CRT Technology

During development of the Boeing 757/767 aircraft, the shadow-mask CRT became available at economical price levels and with the ability to withstand aircraft vibration. All of the avionics vendors bid shadow mask tubes. With the shadow mask tube, higher resolution and more choices of color are possible than with the older penetration style tubes. Several Japanese companies are producing these tubes for the avionics manufacturers in sufficient quantities. The CRT has proven to be one of the most reliable components in the systems.

Other Technological Issues

Several other innovations were required to make todays general aviation EFIS a reality. They included:

1. High voltage power supplies that could be certified to 55,000 feet and were economical to produce.

2. High-speed processors and A/D converters capable of controlling the beam of the CRT while stroke-writing at 30,000 to 50,000 inches per second.

3. High level languages that would allow speedy and economical conversion from format concept to actual working/dynamic software.

All of the above developments had to be available at reasonable cost because of the nature of the business and the relatively good price/performance of electromechanical display systems. In the end, the aesthetic appeal of EFIS would not be enough to make it a commercial success (even in general aviation where it is sometimes the prime driver). The cost of ownership, MTBF, and all of the other parameters normally used to make an intelligent decision in the avionics business were brought into the equation. EFIS has passed the test.

The current EFIS systems (available in the general aviation market) can most easily be described if their electromechanical equivalents are used. As we will see later, this luxury is about to become less useful to us as the EFIS systems gradually progress away from the "basic T" established by the electromechanical instruments.

Basic ADI (PFD) Instruments

EFIS ADI instruments have been developed in three basic formats. All of them (so far) resemble the electromechanical ADI for the most part.

Boeing 757/767 ADI

From the photo it can be seen that the standard round attitude display was retained and the CRT was used to add several useful and desirable pieces of information. These include:

1. Flight mode annunciators

2. Radio altitude

3. Ground speed

4. Choice of single or double que commands

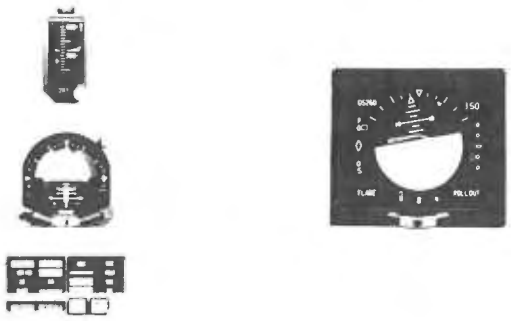


Fig. 5 Boeing ADI

Elliptical Format ADI

The elliptical format is a logical outgrowth of the round attitude display shown above. It allows room for more information to be displayed — such as airspeed as shown in the next photo. The advent of the elliptical format also gives the pilot more pitch information than the round, more traditional formats. When comparing the elliptical formats to the round formats, the following features were added:

1. Airspeed scales (both linear and numeric)

2. Airspeed trend vector

3. Airspeed reference bug set

4. Mach number

5. Longitudinal acceleration during takeoff

6. Vmo and Mmo information

7. Altitude (still in development)

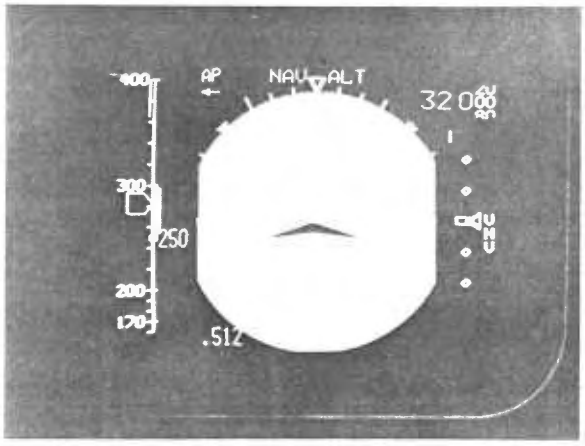


Fig. 6 Elliptical ADI

Full Sky Format ADI

The full sky ADI optimizes the attitude portion of the display. So far, it has been implemented with approximately the same data as the elliptical display but in different formats. The basic version, shown here, is very popular and contains the necessary data to easily transition to an EFIS equipped aircraft.

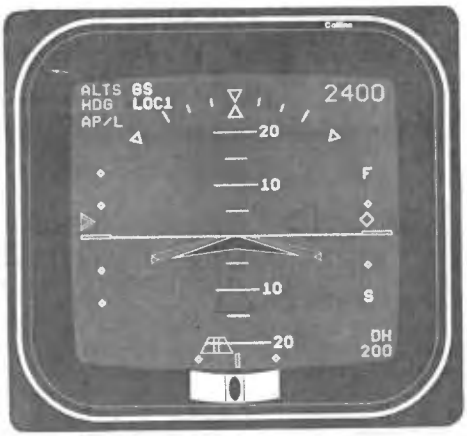


Fig. 7 Full Sky ADI

Basic HSI (ND) Instruments

Since its invention, the HSI has been subject to more changes that the ADI. This is primarily because it is always thought of as a "secondary" (at least to the ADI) instrument and also because the HSI has been the site of the navigation system display. Navigation system development in general aviation has been innovative and steady compared to the airline environment.

Boeing 757/767 HSI (ND)

The Boeing version of the HSI, while able to display a circular compass rose, is more dependent on the flight management system and its ability to create a map for the pilot than any other display available at the time. As shown, the format contains a sector of the compass which can be displayed either track-up or aircraft heading-up. It shows some of the basic information that normally appears on the HSI. Other information is displayed on the ADI. The selected course is displayed digitally while the lateral deviation is displayed, full time, on the bottom of the ADI. The vertical axis is portrayed using a standard vertical scale similar to glideslope but also having digital range to altitude shown next to vnav waypoints. Other standard items appearing in this format are:

1. Distance and time to go
2. Selected heading
3. Heading lubber line and type of heading

There are several unique symbols available on the map display including:

1. Flight plan path
2. Straight trend vector
3. Selected vor radial
4. Vortacs and waypoints not on the flight plan path.
5. Airports
6. Curved trend vector
7. Range marks
8. Weather radar overlay

This display, being of a different size and shape than the PFD, accommodates new symbology with relative ease and represents a logical transition for the flight crews operating the system.

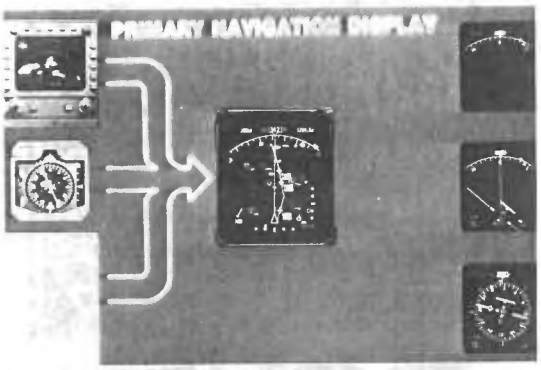


Fig. 8 Boeing HSI

Mechanical Look-Alike HSI's

The initial general aviation EFIS systems were developed to give the pilot a display with which he was already familiar. Thus, the first option that he is given is the look-alike. This is important not only for safety reasons, but also for market acceptance. Since a large number of the general aviation systems are being installed in non-type rating required aircraft, the transition must be very smooth. It has become a real challenge for the avionics manufacturers to sell the system to the high-time, ex-military chief pilot who is not impressed with the new formats or even the innovation oriented pilot. But that challenge has been met by offering both a traditional compass rose and a sector display with map and radar capability. The rose mode shown simply duplicates the mechanical HSI and adds only the necessary symbols required to identify the navigational sources. Color coding of on-side and off-side data is used but little else is added. The second course shown is a pilot option as is the bearing pointer. All are easily recognizable without much identification. The colors of the legends match the data in all cases. Green is used for active information.

Fig. 9 Mechanical Look-Alike HSI

Arc and Sector Formats

These formats are similar to the rose displays but are modified to accommodate the less sophisticated navigation systems currently popular in general aviation. The example shown below has no map information but is capable of displaying weather radar. A simple expansion of the forward sector of the compass rose is the goal here.

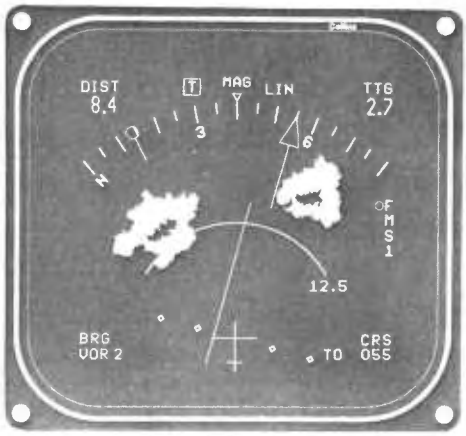


Fig. 10 Sector HSI

Map and Radar Formats

Another format option available is the map mode. It is usually dependent on the navigator and is currently able to display a waypoint string, range marks and weather radar. Generally, a heading up display is used. As the data becomes available from the popular navigators in general aviation, the map displays will start to take on the same types of data as the Boeing displays shown earlier.

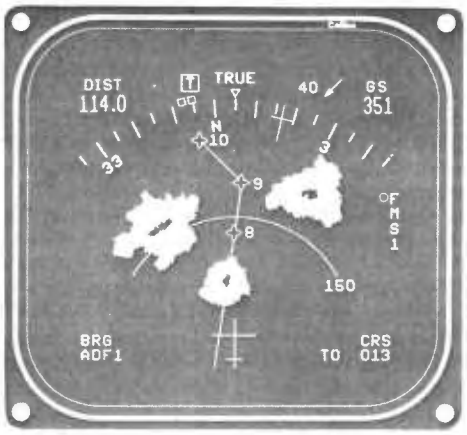


Fig. 11 Map HSI

Symbology Considerations

Due to the wide use of EFIS in general aviation and the dominance of non-type rating requirement aircraft, the FAA has been approaching the subject in a conservative but open manner. The initial fifty or so certifications have been done with full involvement of a team of FAA pilots and engineers. That team (the SCR team) has now been disbanded and a less formal and rigid group (with some of the same members) has been called together to evaluate systems and approve formats. The intent here is to gradually transition to an advisory circular environment for certification. This preamble is necessary to approach the consideration of symbology.

The guideline driving the requirements is the advisory circular draft that is evolving at the present time. The subject has been broken down into several parts.

1. Critical display functions
   A. AC 25.1309-1 total loss of critical information considerations
   B. Continued safe flight after power failure considerations. (Generally 30 minutes minimum)
   C. Hazardous or misleading attitude and navigation information during a failure

2. Color and symbology for information separation
   A. Standard colors
      1. Red is for immediate action
      2. Amber is for caution and subsequent action
      3. Brown is for earth
      4. White is for fixed scales
      5. Green is for active data selected by the pilot
      6. Cyan (light blue) is for fixed captions and sky
      7. Magenta (light purple) is for computed commands and analog raw data pointers.
      8. In addition, some guidelines are given for color separation and common use between the pilot and copilot displays.

   B. Color vs workload (clutter and interpretation)
   C. Symbol stability (repeatable locations, timing and recognition of symbols)
   D. Interpretation of 2-dimensional displays
   E. Attention getting requirements

3. Display visual considerations:
   Brightness, contrast, chromaticity, focus, line width, symbol size, flicker and dynamics all must be evaluated and considered.

4. Information arrangement
   A. The basic T cannot get any worse than previously certified arrangements in that aircraft type.
   B. Compacted (composite) formats
   C. Side-by-side displays are considered a major test program and require extensive evaluation
   D. Requirement for analog scales on some items
   E. Ability of the format to clearly show unusual attitudes

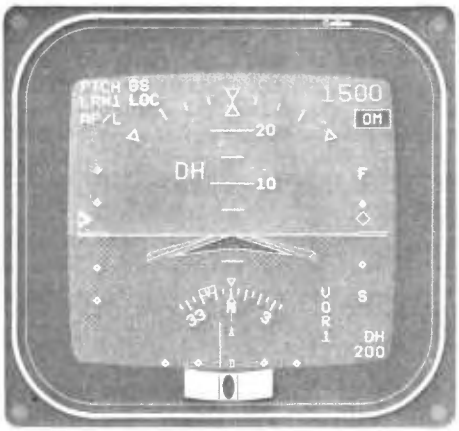


Fig. 12 "Compacted" ADI

Other Certification Issues

The previous section discussed the symbology-related issues. There are others, however, that are related to EFIS and not to today's electromechanical displays.

1. Cooling has become a certification issue at the regional level. There have been wide differences in requirements but, at this time, a consistent approach appears to be evolving. The main consideration here is to ensure that there are no unannunciated failures that the pilot cannot recover from using reversion, etc. As new technology becomes available with lower power consumption, this issue will be eliminated.

2. Training is becoming an issue in that some aircraft are now complex enough (with EFIS contributing but not the only cause)

to cause the FAA to ask for type rating in some cases where it was not required before.

Diagnostic Capabilities

EFIS brings to the general aviation marketplace the ability to display a very comprehensive window into the entire digital avionics shipset for diagnostic purposes. Aside from the normal EFIS related tests such as alignment of the tubes and input/output data, EFIS is able to act as a terminal for other systems on the aircraft when they are in their own diagnostic modes. Only with the all-digital general aviation shipset is this possible. Until recently, the EFIS diagnostics were limited (though much more useful than any previous systems) to displaying inputs and outputs of the EFIS system itself. The following photos show the alignment, EFIS input/output data and flight control diagnostic capability of the system. Future use of the system is almost unlimited since the EFIS system typically is connected by digital buses to almost all of the other avionics systems in the aircraft for normal operation.

Fig. 13 Flight Control System Diagnostics

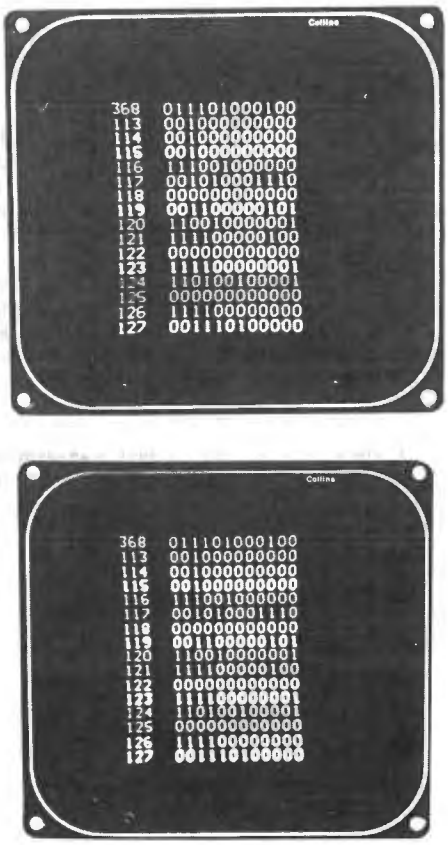
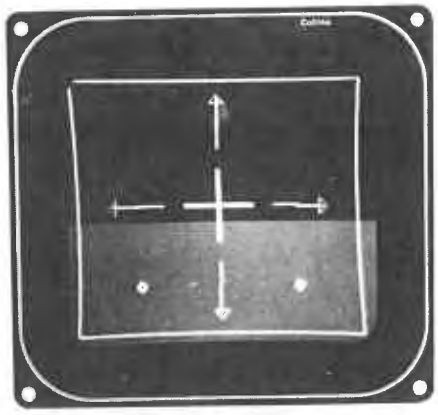


Fig. 14 Input/Output Data



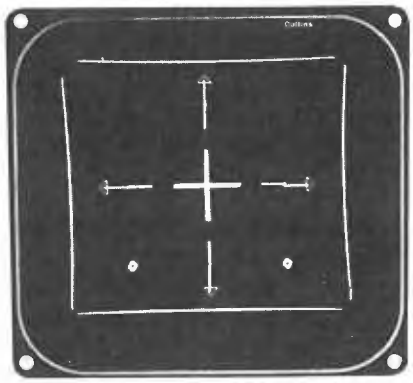


Fig. 15 Alignment Pattern

Multifunction Displays

When the first EFIS systems were developed for general aviation it was obvious that the area used by the raster-scan radar indicator was not as productive as it could be. The integration of that real-estate into the EFIS system became of prime importance. Reversion (covered later), consistent display quality and cockpit integration have driven all of the major general aviation EFIS manufacturers to the MFD (multifunction display) concept. They are capable of several formats:

1. Standard weather radar
2. Map overlays
3. Reversion for the HSI and ADI

Fig. 16 Multifunction Display