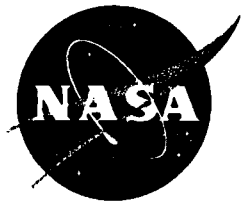
NASA/TM—1999–208788



GT-CATS: Tracking Operator Activities in Complex Systems

Todd J. Callantine
San Jose State University Foundation
San Jose, California

Christine M. Mitchell
Georgia Institute of Technology
Atlanta, Georgia

Everett A. Palmer
Ames Research Center, Moffett Field, California

June 1999

The NASA STI Program Office ... in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) Program Office plays a key part in helping NASA maintain this important role.

The NASA STI Program Office is operated by Langley Research Center, the lead center for NASA's scientific and technical information. The NASA STI Program Office provides access to the NASA STI Database, the largest collection of aeronautical and space science STI in the world. The Program Office is also NASA's institutional mechanism for disseminating the results of its research and development activities. These results are published by NASA in the NASA STI Report Series, which includes the following report types:

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

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

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

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

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

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

Specialized services that complement the STI Program Office's diverse offerings include creating custom thesauri, building customized databases, organizing and publishing research results ... even providing videos.

For more information about the NASA STI Program Office, see the following:

- Access the NASA STI Program Home Page at ***http://www.sti.nasa.gov***

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

- Fax your question to the NASA STI Help Desk at (301) 621-0134

- Telephone the NASA STI Help Desk at (301) 621-0390

- Write to:
  NASA STI Help Desk
  NASA Center for AeroSpace Information
  7121 Standard Drive
  Hanover, MD 21076-1320

GT-CATS: Tracking Operator Activities in Complex Systems

Todd J. Callantine
San Jose State University Foundation
San Jose, California

Christine M. Mitchell
Georgia Institute of Technology
Atlanta, Georgia

Everett A. Palmer
Ames Research Center, Moffett Field, California

National Aeronautics and
Space Administration

Ames Research Center
Moffett Field, California 94035

June 1999

Acknowledgments

This report describes the most recent development in a research project begun in the early 1980s. The operator function model (ref. 1) grew out of a need to describe operator activities in the control of complex dynamic systems. The description needed to be human-readable by the personnel being modeled, researchers, and software developers. Hence, the 'bubble' syntax replaced the traditional prose of task analysis. Moreover, the dynamics of finite state automata captured much of the behavior change, initiation, and termination in operational activities in carried out in complex systems.

Used as both a notation for description and specification for the design of model-based interfaces, the operator function model turned out to be both effective and useful. As computer interface technology became both more available and cheaper, the need to specify the semantics of operator workstations became more critical.

At the urging of Everett Palmer the project expanded to attempt to extend the operator function model into software, a kind of 'living tasks analysis.' Both conceptually and computationally this turned out to be a challenging but rewarding task.

As we were addressing this challenge, DARPA proposed the exploration of a pilot's associate. In many ways our research and its methods paralleled the DARPA work (refs. 2 and 3). Both included a hierarchic description of operator/pilot intentions that were 'expected' top-down; and both attempted to interpret actual operator/pilot actions bottom-up by considering them in light of expectations. Our research had the advantage of being conducted in the context of safety-critical control systems that were much more benign than air combat. This allowed us to incrementally design, implement, and evaluate various concepts.

Thanks in large part to the combined work of Kenny Rubin and Patty Jones the initial design, concept and implementation of OFMspert, or operator function model expert system, occurred in 1986 (ref. 4). For a simulation of a NASA satellite ground control system, OFMspert was applied to the task of understanding operator actions in controlling the system (Jones and Mitchell, 19xx). We were delighted to find that for many measures OFMspert was quite good at explaining operator actions. For some measures, including operator browsing through the interface display pages, however, there was room for improvement.

Our next application was to the tasks of piloting a B-727 (ref. 5). As often happens with research, the outcome was not as successful as anticipated. OFMspert, depending on a operator model represented as a finite-state machine, did not represent well the primarily continuous activities of navigation in a 'non-glass' cockpit.

Concurrently, OFMspert was extended and enhanced as both an aid (ref. 6) and a tutor (ref. 7). Application and evaluation in satellite ground control showed that OFMspert provided a useful knowledge structures for intelligent aiding and training.

GT-CATS, the research described in this report, built on and extended various flavors of previous OFMspert applications and used our accumulated wisdom. For researchers at Georgia Tech, past and current, and at NASA Ames, GT-CATS represents a major milestone. GT-CATS runs and empirical data show it to be reasonably successful. It also offers many productive avenues for follow-on research. The on-going support and assistance of many people is gratefully acknowledged. They include T. Govindaraj, Richard Robison, Alan Chappell, Jim Williams, and Delta Airlines Training and Flight Operations Departments. As all modelers know, success or failure depends in large part on the quality of the model. GT-CATS was greatly facilitated by the input and flying expertise of many pilots including Captains Arnie Kraby, Alan Price, and Bill Jones of Delta Airlines, and Captain Jim Irving of United Airlines.

Available from:

NASA Center for AeroSpace Information
7121 Standard Drive
Hanover, MD 21076-1320

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22161

Table of Contents

Summary

Human operators of complex dynamic systems can experience difficulties supervising advanced control automation. One remedy is to develop intelligent aiding systems that can provide operators with context-sensitive advice and reminders. The research reported herein proposes, implements, and evaluates a methodology for activity tracking, a form of intent inferencing that can supply the knowledge required for an intelligent aid by constructing and maintaining a representation of operator activities in real time. The methodology was implemented in the Georgia Tech Crew Activity Tracking System (GT-CATS), which predicts and interprets the actions performed by Boeing 757/767 pilots navigating using autopilot flight modes. This report first describes research on intent inferencing and complex modes of automation. It then provides a detailed description of the GT-CATS methodology, knowledge structures, and processing scheme. The results of an experimental evaluation using airline pilots are given. The results show that GT-CATS was effective in predicting and interpreting pilot actions in real time.

1. Introduction

Human operators increasingly use automation to control complex dynamic systems.1 Such operators function as supervisory controllers, monitoring and intermittently programming the automation to control the task environment. As computer technology has become more powerful, more aspects of the operator's former control task have become automated, and the automation itself has become more complex. The proliferation of automation has changed the human supervisory controller's task. Humans now make less frequent—albeit more complicated—inputs to the automation, and must monitor more complex information about both the controlled process and the operation of the automation (refs. 8–9). This transformation has placed new demands on the human operator. The operator must understand how automation is to be used in light of the current operating situation; the operator must be able to trade off operational objectives as necessary to use the automation effectively; and, the operator must be able to manage the monitoring and mental bookkeeping required to assess the situation correctly and "stay ahead" of the automation (ref. 10). If the operator experiences difficulties meeting one or more of these demands—in times of high workload or abnormal operation, for example—he or she becomes susceptible to errors that can compromise system safety (ref. 11)

Breakdowns in human-machine interaction have motivated a broad spectrum of research that attacks the problem from three interrelated angles: improving the design of the human-machine interface, improving operator training, and devising ways to aid the operator. For example, some research seeks to present information in a way that emphasizes important features of the system crucial to operator understanding (ref. 12), or to dynamically tailor displayed information to the situation at hand (refs. 13 and 14). Other research addresses improved training for operators of complex systems (refs. 7 and 15). Still another approach is adaptive aiding (refs. 16–18). Adaptive aiding combines dynamic task allocation, to keep operators "in the loop," with error-resistant, error-tolerant systems to keep operators from making errors wherever possible, and to detect and alleviate the effects of errors that do occur (refs. 8 and 19).

One way to foster error tolerance and support dynamic task allocation is to develop intelligent operator aiding systems that monitor the human-machine interaction and supply timely advice and reminders to the operator (refs. 4 and 20). This research addresses a facet of such systems—referred to as operator's associates (ref. 4), or intelligent operator assistants (ref. 21). The USAF/ Lockheed Pilot's Associ-

1 The term "complex system" is used herein to refer to engineered systems controlled by well-trained operators that are assumed to be well-motivated, and for which system state data are available via computer.

ate is an example of the operator's associate concept designed for fighter pilots (refs. 2, 22 and 23). The Pilot's Associate and other operator's associate systems construct and maintain a dynamic, context-specific representation of what the operator is doing—and will be doing—and why. Such a representation provides the knowledge required for the associate to monitor the human operator to detect errors and provide assistance.

Developing reliable dynamic representations of operator activities to support human-machine interaction is the focus of intent inferencing (refs. 4 and 18). *Activity tracking* is a type of intent inferencing explored in this research, so called because it focuses not on the psychological aspects of human intent, but on the context-specific manifestations of operator intentions as overt control activities. By tracking operator activities—like one human monitoring and interpreting another human's behavior—an operator's associate can maintain a dynamic representation of operator activities which can be used to support human-machine interaction.

As proposed in this research, activity tracking has four elements. The first element is the capability to *hypothesize* how the operator will perform the next set of activities in the current operational setting. The second is the capability to *confirm* the hypotheses based on actual operator actions. The third is the capability to *interpret unexpected operator actions* that were not hypothesized, to determine whether the unexpected operator actions are errors, or part of an alternative, but valid, method for using the automation. Finally, activity tracking includes the capability to *identify missed or late operator actions* so that possible errors of omission can be detected.

This research proposes, implements, and evaluates a methodology for activity tracking. The methodology embodies a theory that, first, establishes conditions on the types of knowledge that must be available in a domain to support activity tracking. Specifically, the methodology applies to engineered systems in which information about the state of the system, goals of the operator, and standard operating procedures is available.

Second, the theory underlying the activity tracking methodology establishes an organizational structure for the available domain knowledge. The activity tracking methodology uses a model of human-machine interaction based on the Operator Function Model (OFM) (refs. 1 and 24) to represent knowledge about how operators use automation. This enhanced OFM is called an OFM for systems with Automatic Control Modes (OFM-ACM), in deference to the role modes play in complex automation (ref. 25).

Third, the methodology is theoretically founded on the capability to transform the available knowledge of the state of the controlled system and goals of the operator into knowledge for predicting activities represented in the OFM-ACM. Using the conditions on available knowledge, this capability provides a flexible means of constructing a representation of current and future operator activities.

Fourth, the activity tracking methodology embodies a theory for processing the available knowledge. The theory provides that updated knowledge about the state of the controlled system can be used to interpret unexpected operator actions. In addition, it offers a means by which the required knowledge can be used to track operator activities in real time. Real-time interpretation of operator activities enables an operator's associate to supply timely advice and reminders.

The processing architecture was used to implement the methodology in a computer system called GT-CATS (Georgia Tech Crew Activity Tracking System). The thesis of this research is that the GT-CATS architecture can construct a real-time representation of operator automation usage. As a proof-of-concept, GT-CATS was implemented and evaluated in the domain of glass cockpit aircraft. The results of the evaluation showed GT-CATS to be effective in tracking pilot activities.

The remainder of this document is organized as follows. Chapter 2 discusses the potential impact of activity tracking and its roots in intent inferencing research. Chapter 3

describes human-automation interaction, with a focus on modes of automation and the errors modes engendered as identified through research on glass cockpit aircraft. The GT-CATS methodology is not limited to modal systems, and is in no way bound to glass cockpit automation; however, complex systems with multiple modes present a particularly challenging domain for the application of activity tracking, and glass cockpit automation is a well-studied example of such systems.

To support later discussions, Chapter 3 opens with a general description of the Boeing 757/767 glass cockpit intended to familiarize the reader with noteworthy displays, controls, and modes.

Chapter 4 describes the GT-CATS methodology and computer architecture, including the OFM-ACM. Chapter 5 describes the implementation of GT-CATS for the glass cockpit. Chapter 6 describes the GT-CATS evaluation study. It opens with a discussion of the evaluation methods applied by other researchers, then presents the GT-CATS evaluation procedure in detail. The results of the evaluation are given in Chapter 7, including the insights gained from a micro-analysis of action-by-action activity tracking outcomes. Chapter 8 summarizes GT-CATS research and its findings, and outlines important avenues for further research.

2. Intent Inferencing

Introduction

In the domain of complex dynamic systems, intent inferencing can be thought of as the process of inferring the intentions of a human operator controlling a complex system from the state of the system and observed operator actions. A computer system that can infer operator intent can then use this representation to support "intelligent" human-machine interaction (refs. 4, 18, and 26–29). Aid, advice, or reminders are intelligent when based on a model of what the operator is doing and why. Activity tracking is a form of intent inferencing that focuses on explanation of operator activities without addressing the precise psychological nature of the formation of human intentions.

This chapter summarizes intent inferencing research as conducted in the area of human operators responsible for the safety and effectiveness of complex dynamic systems. It first describes the motivations behind previous inquiries into the application and feasibility of intent inferencing. The chapter then describes models that can effectively support intent inferencing; such models represent both the physical and cognitive aspects of the operator's task in the domain of interest. Finally, the chapter presents a review of intent inferencing and related research.

Motivations for intent inferencing and related research

This section briefly describes several avenues of human-machine systems research that have substantiated the need for intent inferencing. All were developed in response to the transformation of the operator's role in increasingly automated complex systems. Although the use of advanced technology imposes new demands on operators, human abilities to anticipate and adapt to novel or uncertain situations preclude replacing human operators in complex dynamic systems (ref. 30). The research discussed in this section attempts to improve human-machine interaction while honoring the significance of both the human operator and the computer components (i.e., machine agents) of the controlled system.

Intelligent Decision Support Systems

Intelligent decision support systems (IDSSs), in which the human operator can allocate tasks to a machine agent, were one early attempt to wed human versatility and the analytical power of computers (ref. 31). IDSSs are expert systems. In the IDSS paradigm, humans guide the problem-solving process by supplying information to the IDSS, and the IDSS performs the complex reasoning required to solve the problem. Through a sort of question and answer session, human operators supply the information necessary for the IDSS support complex tasks, such as fault diagnosis. However, studies exposed deficiencies in the human-machine interaction fostered by IDSSs. The allocation of tasks between human and machine is designed into the system, and is therefore static. The problem solving process could be led astray by unanticipated variabilities, uncertainty about applicable types of knowledge, and deficiencies in the understanding between human and IDSS (refs. 31–34). To enhance human-machine interaction, researchers instead sought ways to use computer technology to develop cognitive tools that allow the human operator to effectively exploit machine capabilities in concert with his or her own (refs. 35–37).

Adaptive aiding

The concept of adaptive aiding in a sense foresaw the difficulties that IDSSs would encounter (refs. 16 and 17). Adaptive aiding is founded on two concepts: dynamic task allocation and error tolerance. Dynamic task allocation can enhance human-machine interaction by using the current operating context to determine how tasks should be allocated to human and machine agents, and to keep the human operator "in the loop." Error tolerance is a property of the machine agent that can enhance human performance by detecting

errors and helping to correct them or minimize their effects.

Both dynamic task allocation and error tolerance can benefit from intent inferencing. By incorporating an intent inferencing element, a machine agent can use its knowledge about the operator's current objectives in order to identify tasks it can support, and to distinguish operator errors from valid actions.

Hammer, Rouse, and Rouse developed an aid to assist pilots in the detection and remediation of procedural errors (refs. 26 and 27). The aid used a hierarchical script of flight procedures to identify correct actions, ommitted actions, and inexplicable actions (i.e., actions that did not fit into any scripts). A display was developed that dimmed each procedural step as it was performed.

To evaluate their computer aiding concepts, simulator data from four two-person crews flying a twin engine aircraft were collected for three scenarios: a normal flight and two emergency flights that involved engine and landing gear status indicator failures. The data included aircraft state variables, discrete operator actions, and transcripts of verbal crew communication. These data were used as off-line input to the computer aid. In a comparison between hard-copy procedures checklists and the computer-based procedures aid, the computer-based system detected and virtually eliminated procedural errors. Thus, the research demonstrated the potential usefulness of computer-based cockpit aiding systems.

Human-centered automation

Billings' (ref. 8) concept of human-centered automation is a philosophy for automation design that incorporates the need for intent inferencing. The philosophy is intended to address common shortcomings of automation in complex systems. Automation is often technology-driven: new technology enables some aspect of the human operator's task to be automated, but takes the human operator "out of the loop" in the process. Human-centered automation seeks to keep the operator in command, which in turn requires that the operator is informed and involved with monitoring the automation (ref. 14). As conceived by Billings, human-centered automation requires cross-monitoring, where both human and machine agents monitor the others. For cross-monitoring to be effective, each element in the system must have knowledge of the others' intent; thus, an intent inferencing component is a critical element of human-centered automation.

Operator's associate

An operator's associate is a machine agent that acts like a human assistant—subordinate and cooperative with respect to the operator, able to assume responsibility for tasks on demand, and able to monitor and anticipate situations and events (refs. 4, 21, 38, and 39). Intent inferencing is vital for providing the operator's associate with an understanding of what the operator is doing—and will be doing—and why. Using this knowledge, the control component of the operator's associate can provide timely advice and reminders, detect and remediate errors, and carry out tasks allocated to it by the human operator. Thus, the concept of a computer-based operator's associate encompasses both adaptive aiding and supporting the cognitive activities of the operator.

Researchers have pursued the operator's associate concept along several broad fronts. For example, the DARPA-funded USAF/Lockheed Pilot's Associate system is an operator's associate for fighter pilots (refs. 2, 22, and 23). The Pilot's Associate includes an intelligent pilot-vehicle interface that uses inferred intent to predict pilot performance, required resources, and the consequences of errors. The review of research later in this chapter discusses this and other important research on operator's associate systems in detail.

Intelligent information displays

Another application for intent inferencing systems is to guide when and how to display information to the human operator. For example, researchers have found evidence that information requirements of operators vary according to the plans they are currently pursuing (ref. 40). Operator performance can be

supported by configuring displays according to the information requirements of a particular plan; the plan operators are currently pursuing can be determined through intent inferencing. The intelligent pilot-vehicle interface in the Pilot's Associate, for example, incorporates intelligent information displays as one means of aiding pilots.

Intelligent tutoring systems

Another important application of intent inferencing is intelligent tutoring systems. Due to the prohibitive cost of 'complete' training, training programs typically result in operators that are far from experts—at best they are "trained novices" (ref. 15). Intelligent tutoring systems can help eliminate some of the on-the-job training normally required to achieve expert performance.

Intent inferencing can support dynamic student and expert models in intelligent tutoring systems (ref. 7). A student model that uses intent inferencing to understand the actions of the trainee can guide the instructional process. In addition, expert models can benefit from the predictive capabilities provided by intent inferencing. Rather than using "canned" scenarios that limit the scope of training, an expert model can predict what activities the operator should perform in varied contexts, thereby extending training to reflect real-world situations automatically.

Operator models for intent inferencing

Intent inferencing systems require domain-specific knowledge about the operator's task and the controlled system, and a means for controlling processing of this knowledge. Using updated information about the state of the system and the actions the operator performs, an intent inferencer processes knowledge about the operator's task to produce a dynamic representation of operator activities in the current operational context.

Human-machine systems research has established the importance of well-defined models of the human operator (refs. 1 and 16). A variety of models of the human operator have been developed; different models can be characterized conceptually by their purpose, structure, content, and specificity (ref. 41). Descriptive models include Rasmussen's (ref. 42) decision ladder and abstraction hierarchy; the OFM (ref. 1) and goal-means network (ref. 36), on the other hand, exemplify normative models. Model structure can be computational, as with control theoretic models (ref. 43), discrete control models (ref. 44), and OFMs—or conceptual, like the abstraction hierarchy. The content of models ranges from mental representations of the task derived from psychology (refs. 12, 45, and 46), to engineering models of overt operator activities (e.g., the OFM)—the type of model often referred to generally as an "operator model." Finally, models can be specific to a particular device, a class of machines (i.e., task models), or they can focus on cognitive processes independent of the machine agent.

Cognitive engineering models of the operator that are capable of dynamically and computationally representing salient physical and cognitive aspects of the operator's task in the domain of interest provide one way to support effective human-machine interaction in complex systems (ref. 24). Given the current system state and system goals, the model represents what interventions the operator should undertake and why, along with the control options the operator can exercise to attain the desired system state. The model also specifies a hierarchy of activities, in order to represent the complexity of the system in a manner that is cognitively compatible with the operator's actual representation of the task. Thus, to effectively provide the intelligence necessary to aid the operator (via intent inferencing), a model should be both normative, in that it can generate expectations of operator activities, and interpretative, in that it can 'understand' operator activities in the current context.

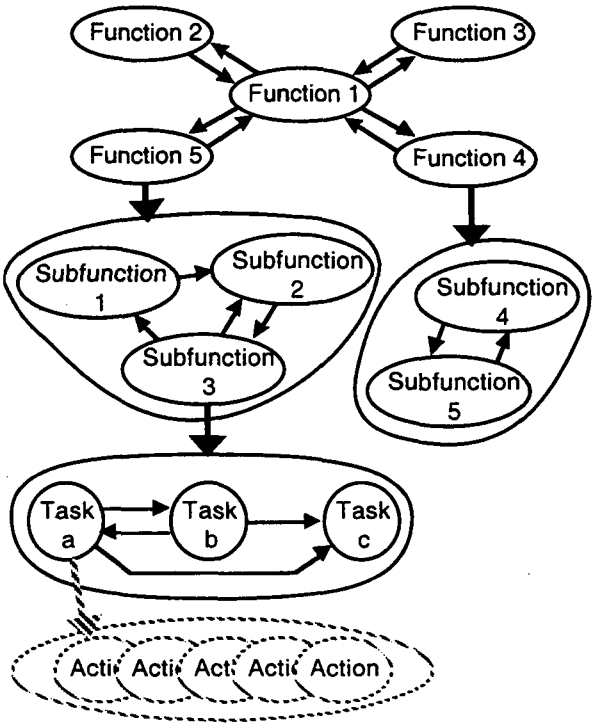


Figure 1. A generic OFM.

Operator Function Model

The Operator Function Model (OFM) is an example of a model developed towards these ends (ref. 1). The OFM (figure 1) is a hierarchical-heterarchical network of finite-state automata, based on the discrete control models of Miller (ref. 44). Nodes in the network represent operator activities; arcs represent enabling conditions that initiate or terminate operator activities as dictated by system events or the results of other activities. These enabling conditions are non-deterministic in that they identify a set of activities plausible for the current context, rather than a unique next activity.

The OFM hierarchy represents how operators might decompose control functions, from high-level functions down to individual manual or cognitive actions. The OFM heterarchy represents collections of activities at a particular level in hierarchy that are performed concurrently—a feature that enables the OFM to represent how operators dynamically coordinate activities and focus attention. Thus, the OFM provides a flexible framework for representing operator activities in complex systems (refs. 1 and 24). Chapters four and five discuss enhancements to the OFM that led to the OFM for systems with Automatic Control Modes (OFM-ACM) used in this research.

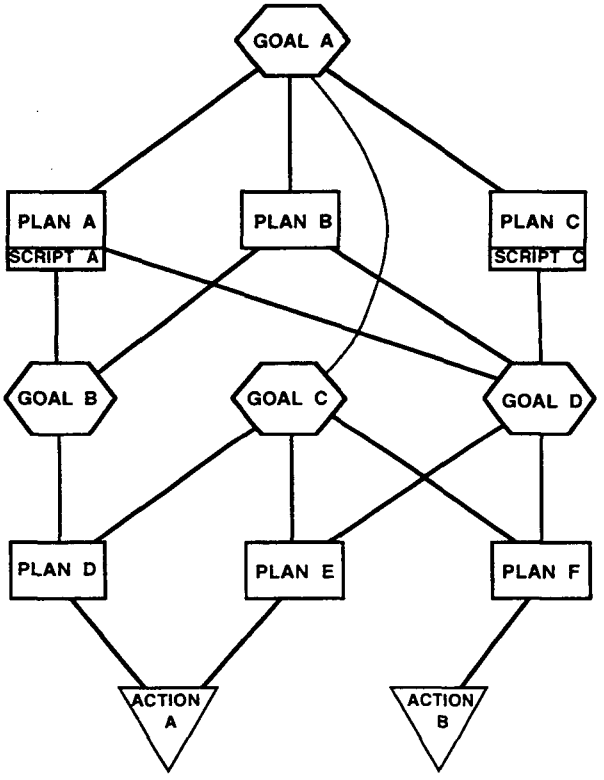


Figure 2. Generic Plan-Goal Graph.

Plan-Goal Graph

Another model that can support intent inferencing is a Plan-Goal Graph (PGG) (ref. 29). A PGG is a network of plans and goals. Unlike the OFM, the PGG derives from research in psychology and artificial intelligence using Shank and Abelson's (ref. 47) concepts of scripts, plans, and goals as cognitive structures of understanding (ref. 48).

In a PGG (figure 2), each high-level operator goal is decomposed into a set of plans that can be used to achieve it. Plans are then decomposed into subgoals, which in turn are decomposed into lower-level plans. The lowest-level

plans in the PGG are decomposed into the individual operator actions required to execute each plan. Plans may also have scripts that represent loosely ordered sequences of required actions. With this structure the PGG can represent the options available to the human operator in a complex system.

The links in a PGG are important for representing system-dependent constraints on relationships between plans and the goals they satisfy. Feasibility constraints express the range of system parameters within which a plan may be effectively used to satisfy a goal. Ambiguity constraints, so called because they are used to resolve the ambiguity present when a plan has multiple goals, represent the normative approach to satisfying a goal given the values of current system parameters. In addition to the constraints represented by the links in the PGG, each plan and goal has a list of other plans or goals with which it is mutually exclusive. Such an exclusion can be associated with values of pertinent system parameters, if required. These constraints, together with its structure, enable the PGG to represent the domain knowledge associated with the controlled system. The PGG is similar to the OFM in that it represents operator activities in a hierarchy, and contains information about normative activities given current system state.

Intent inferencing and related research

This section describes intent inferencing research in terms of the models and processing used, the domain of application, and the implications for future research. In cases where an intent inferencing system has been implemented in an intelligent aiding and/or training system, this work is also discussed.

OFMspert

The Operator Function Model expert system (OFMspert) research program focuses on the design of an operator's associate for complex dynamic systems (ref. 4). OFMspert was implemented in the context of a satellite ground control system (ref. 1). OFMspert uses the OFM as the source of knowledge about the controlled system and related operator functions. OFMspert's intent inferencing component, the Actions Interpreter (ACTIN), is responsible for maintaining a dynamic, context-specific representation of current best

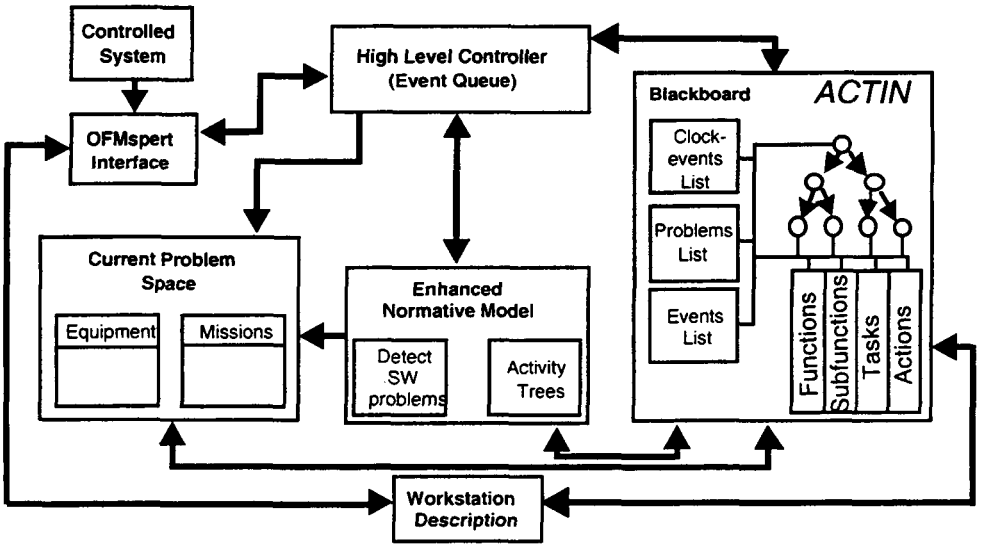


Figure 3. Generic OFMspert architecture.

hypotheses about operator activities. Other components are responsible for providing system-state knowledge, and controlling real-time processing (figure 3).

ACTIN was implemented as a blackboard system (ref. 49). Given system state information, ACTIN posts functions, subfunctions, and tasks from the OFM on its blackboard. As OFMspert detects operator actions, they too are posted on the blackboard and linked to every task they can support according to the OFM. These functions, subfunctions, tasks, and actions represent the inferred intent of the operator (figure 4). An important property of OFMspert's intent inferencing process is maximal connectivity; actions are interpreted to support as many tasks as possible. In this way, OFMspert explains operator actions in terms of all feasible tasks given the current system state.

Once actions have been linked to the specific task(s) they can support, ACTIN assesses the blackboard. The blackboard knowledge sources check to to ensure that constraints on the temporal ordering of actions involved in procedures, and constraints on the semantic content of actions that have values associated with them are all satisfied. For example, an action to replace a particular piece of equipment is constrained by the availability of the replacement equipment. Blackboard knowledge sources check to ensure that the replacement equipment is available. Thus, the assessment procedure provides the final 'understanding' of operator actions in OFMspert.

ALLY

OFMspert's understanding capabilities were subsequently augmented with control capabilities, and the capability to use inferred intentions to guide user interaction. The resulting operator's associate, called ALLY, was empirically evaluated by comparing the performance of one satellite ground controller using ALLY to the performance of a team of two human controllers (ref. 38). No significant performance differences were found, which provides empirical evidence for the efficacy of an operator's associate.

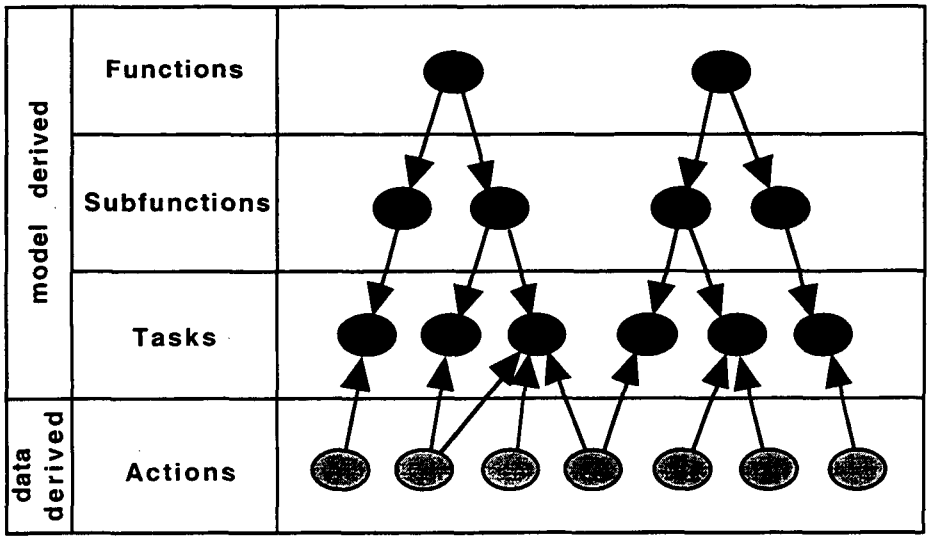


Figure 4. ACTIN (Actions Interpreter)—a dynamic, hierarchical representation of operator intentions

GT-MOCA

A second extension to OFMspert research is the Georgia Tech Mission Operations Cooperative Assistant (GT-MOCA). GT-MOCA is an operator's associate designed according to a theory of human-computer cooperative problem solving that embodies five principles: human authority, mutual intelligibility, openness and honesty, management of trouble, and multiple perspectives (ref. 39). GT-MOCA uses the ACTIN intent inferencing module to provide an interactive, inspectable model of expected operator's activities, along with context-specific reminders. Through empirical evaluation, these features were shown to promote improved performance. Furthermore operators received it positively, supporting the claim that the design principles GT-MOCA embodies are valid.

GT-VITA

The Georgia Tech Visual and Inspectable Tutor and Assistant (GT-VITA) uses the OFM and OFMspert to structure student and expert models for an intelligent tutoring system (ref. 7). GT-VITA uses these models to control student interaction with the tutor. An implementation of GT-VITA was empirically evaluated using actual NASA satellite ground controllers as subjects. GT-VITA was so effective that it reduced the estimated training time required from three months of on-the-job training to just days. It has since become an integral part of NASA's orientation program for ground control personnel. In combination with GT-MOCA, GT-VITA also conceptualizes the tutor-aid paradigm, in which the same knowledge structures used to support training gradually shape an operator's associate that supports the experienced operator (ref. 50).

OPAL

A second important body of intent inferencing research centers around OPAL (Operater Plan Analysis Logic), an intent inferencing system that uses the Plan-Goal Graph to anticipate the context-driven activities of the human operator (ref. 29). This research was also motivated by the need for intelligent aiding systems to detect and help remediate errors (ref. 17), and to design and control intelligent, intent-driven interfaces to complex systems (ref. 18).
OPAL's intent inferencing process creates a representation of the operator's current intent expressed as active instances of goals, plans, and scripts. Initially, a set of active goals and plans is identified with the overall mission of the operator. As operator actions are detected, OPAL first attempts to associate them with active scripts. If an action matches an active script, OPAL explains the action as supporting the procedure that the script represents. If the action cannot be explained in this manner, OPAL next attempts to use the PGG to determine if the action can be explained as supporting a known active plan. If not, OPAL uses the structure of the PGG and its associated constraints to attempt to locate other plans and goals that the action can support in the current situation. Failing this, OPAL identifies the action as a possible error.
OPAL is similar to OFMspert in several ways. First, both use network models that establish a hierarchy of operator activities. Both use domain-specific conditions specified in the model to postulate the activities operators should address in the current context. OPAL differs from OFMspert in that it uses scripts to explain actions involved with procedural activities, in the manner of systems designed for natural language understanding (ref. 48). OFMspert, on the other hand, uses a blackboard architecture to maintain a dynamic representation of operator activities. Both systems assess constraints on operator actions in generating explanations.

USAF/Lockheed Pilot's Associate

OPAL was initially evaluated in the context of a small process control system (ref. 29), but has since been used as the intent inferencing module in the Pilot's Associate (refs. 2, 22, and 23). OPAL's predictions and explanations for operator actions are used as input to an intelligent pilot-vehicle interface. The pilot vehicle interface uses this information to predict the pilot's performance, to predict the

resources (e.g., information, weapons systems) the pilot will require, and to classify pilot errors and predict their consequences. OPAL also supplies input to the tactical planning module of the Pilot's Associate.

As part of the intelligent pilot-vehicle interface, the Pilot's Associate also incorporates an information management module that uses inferred plans and goals to intelligently manage displays (ref. 3). This system uses OPAL's output as input to an algorithm that selects displays based upon the information required by the operator. The algorithm selects more displays until either all required information is presented, or there is no space left on any device to display the rest. In the spirit of the Pilot's Associate, a Rotorcraft Pilot's Associate that uses these principles is also under development (ref. 51).

Task Support System/Cockpit Task Management System

The Task Support System is another design for an intelligent interface based on intent inferencing for military pilots (ref. 52). An interesting feature of the Task Support System is that it employs a distributed model of the pilot's task. Specifically, the Task Support System is agent-based, in that it is comprised of collection of software objects. System agents encapsulate the current state of the actual aircraft system or subsystem they represent (including cockpit displays and controls), along with static knowledge about these systems and subsystems. Task agents receive information from the system agents, which they use in conjunction with internal knowledge to assist the pilot in performing the task they represent. Other task agents use their knowledge to coordinate lower-level task agents. The agents and the communication among agents thereby represents the model of the pilot's task.

The Task Support System provides several types of assistance to the pilot, in accordance with Funk and Lind's (ref. 52) recommendations for an integrative pilot-vehicle interface. Each task agent determines when its task should be initiated, and alerts the pilot if he or she is late in initiating it; task agents notify system agents representing displays when particular information should be displayed, and in what mode; pilots can instruct task agents to either monitor actions during the task, recommend actions, or perform the task automatically; and, task agents provide system alerting functions and monitor successful task completion. In addition, the Task Support System displays active and pending tasks. The Task Support System was evaluated against a baseline interface and found superior in both performance and pilot preference.

Research on the Task Support System paved the way for the Cockpit Task Management System (ref. 20). The Cockpit Task Management System is designed to aid the pilot in "the process of initiating, monitoring, prioritizing, and terminating tasks (p. 1521)." As in the Task Support System, system agents and task agents were instantiated to represent task and domain knowledge in a distributed fashion. The Cockpit Task Management System agents provide pilots with knowledge about task state (i.e., latent, upcoming, in-progress, suggested, or finished) and task status (i.e., satisfactory or unsatisfactory) using color-coded displays. Furthermore, this information is prioritized to emphasize important tasks. A simulator study comparing pilot performance with the Cockpit Task Management System to performance without it showed the Cockpit Task Management System significantly improved task completion, and indicated positive effects on pilot response time, task prioritization, and control of important aircraft parameters. Cockpit Task Management System research is being followed by work on an Agenda Manager that assists pilots in highly automated systems in which most lower-level tasks are performed automatically.

Cockpit Assistant System and Intelligent Flight Path Monitor

The Cockpit Assistant System (CASSY) is another pilot's associate system developed in Germany (ref. 53). The Intelligent Flight Path Monitor is under development in the United

Kingdom (ref. 54). These systems are notable for several reasons. First, they draw on "human-centered automation" concepts developed in the U.S. (ref. 8) as well as work on operator's associates such as the Pilot's Associate, in an effort to produce an operator's associate for commercial airline pilots. They use advanced voice interfaces for interaction, and integrate different types of modeling techniques (e.g., fuzzy logic, petri nets, and neural networks). Both systems are being aggressively developed by consortia of universities and/or government and industry, and like the Pilot's Associate, both are ambitiously designed to integrate assistance for a full range of aviation problems. CASSY is reported to have passed in-flight feasibility testing, and cost estimates for commercial certification have been calculated to include the full re-design of cockpit automation required for fully integrated implementation. Due to their similarities to the Pilot's Associate discussed above, CASSY and the planned Intelligent Flight Path Monitor are not detailed here; rather, the point is that European researchers have strongly embraced the operator's associate concept, and devoted considerable resources to its development. Their studies have shown it to be promising, and now they are actively pursuing the goal of certifying such systems for use on the flight deck. In addition to CASSY and the Intelligent Flight Path Monitor, Robson et al. (ref. 54) indicate that several other research programs aimed at developing operator's associates are afoot elsewhere in Europe; indeed, this has been the case for some time (ref. 55).

Summary

This chapter described the concepts of the operator's associate, human-centered automation, intelligent interfaces, and intelligent tutoring systems. It also described the importance of operator models to support such systems. Finally it reviewed important operator's associate systems and related systems to provide a theoretical and applied foundation for this research.

The next chapter, on modes in complex systems, summarizes a considerable body of research related to intent inferencing research. Problems with modes in complex systems—and aviation in particular—have contributed to the focus on operator's associates for the cockpit. Furthermore, as the next chapter describes, the function of an operator's associate is complicated in situations where operators must supervise the operation of multiple modes, making effective coordination and interaction between the human operator and an associate even more crucial. While several research projects have explored intent inferencing to support intelligent aiding systems, extant data are either classified (due to their military significance), proprietary, or pertinent to tasks that are less complex than flight deck mode management. The present research therefore seeks to provide publically available data on real-time activity tracking for a class of systems in which the Pilot's Associate is included. In the process, it posits theoretically important properties of the proposed activity tracking methodology, and demonstrates its effectiveness.

3. Modes In Complex Systems

Introduction

Modes are an important feature of automation in complex systems. Modes have proliferated as a useful means of formatting displays, entering data, and providing control options to the human operator; however, modes can contribute to operator confusion. Accidents involving glass cockpit aircraft (refs. 56 and 57)—as well as an abundance of less serious mode-related incidents (refs. 58–60)—provide grim evidence for this claim.

Early human factors research on cockpit automation addressed a broad range of issues, including modes (refs. 61–65). While some researchers stressed the importance of communication, coordination, and cooperation among pilots (ref. 66), Wiener's (ref. 67) survey of pilots helped focus attention on modes. Wiener found that automation can increase workload at times when it is already high—evidence of Bainbridge's (ref. 10) "irony of automation"—and characterized the automation as "clumsy." Pilots at times fell "behind the airplane," often wondering "What is it doing now?," "Why is it doing it?," and "What's it going to do next?"

A survey and subsequent simulator study of a different glass cockpit aircraft sought reasons for "automation surprises," and areas of misunderstanding (ref. 68). The research identifies several mode-related difficulties, including mode availability or disengagement, tracking automatic mode transitions, Vertical Navigation (VNAV) mode target values and logic, infrequently used modes, and selecting from multiple modes. Subsequent studies addressed pilot "mode awareness" in other glass cockpit aircraft (refs. 69 and 70). Again, the research indicates the "strong and silent" nature of advanced automation can compromise mode awareness; the automation can surprise pilots by taking unexpected actions, and by failing to take expected actions. In some cases, pilots experience these problems when they prepare a mode for use, then forget to engage it.

The capability to track operator activities in complex systems with multiple modes is an important step toward operator's associates, intelligent tutoring systems, and interfaces that can effectively neutralize mode-related problems. Although GT-CATS' domain of application need not have modes, it recognizes this requirement. This chapter provides a foundation for understanding modes in complex systems and the problems they engender. The chapter classifies modes, then focuses on modes of automation used to control complex systems. After characterizing control modes, the chapter outlines the demands that modes impose on operators' cognitive resources, and how demand-resource mismatches can cause breakdowns in human-machine interaction.

For the reader unfamiliar with 'glass cockpit' airplane modes used as examples, the chapter first provides an overview of the automation found in the Boeing 757/767, a typical glass cockpit aircraft.

The Boeing 757/767 glass cockpit

Glass cockpit aircraft like the Boeing 757/767 have complex automation that pilots monitor using CRT-based (i.e., "glass") displays. The automation requires pilots to supervise the operation of multiple modes. A range of modes offers specific control advantages, but also results in a wide variety of behaviors and possible transitions in different contexts of which pilots must be aware.

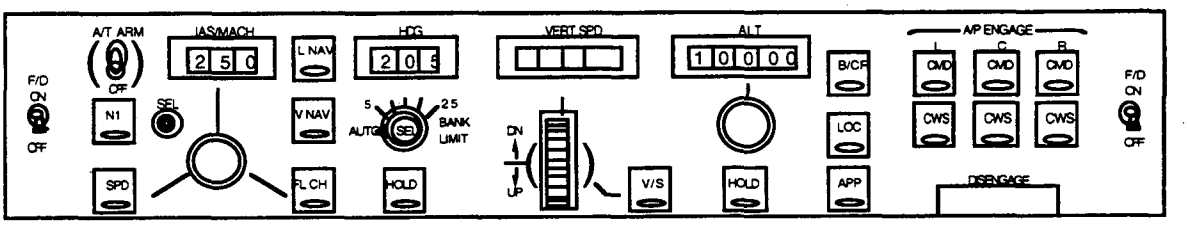


Figure 5. 757/767 Mode Control Panel (MCP).

Like other glass cockpit aircraft, the 757/767 Autopilot Flight Director System, or 'autoflight system,' has a mode control panel (MCP) that allows pilots to coordinate control of autopilot, flight director, autothrottle, and altitude alert functions. The MCP provides the control and display functions used by the crew to manage different modes. It houses all the switches for selecting modes, as well as knobs for selecting heading, altitude, airspeed/mach, and vertical speed (figure 5). The values selected on the MCP are target states to be acquired in certain modes.

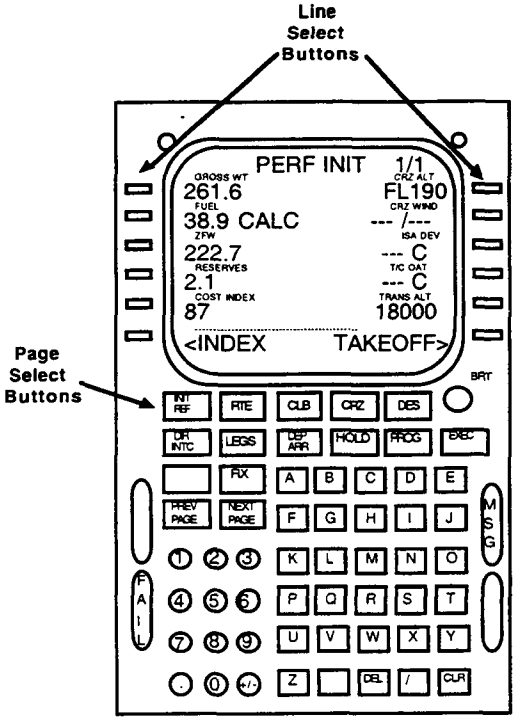


Figure 6. 757/767 FMS Control and Display Unit (CDU).

Pilots can couple autoflight system operation with the Flight Management System (FMS) by selecting certain modes on the MCP. The FMS provides computerized navigation functions; information programmed in the FMS defines the flight profile the autoflight system follows, instead of MCP-selected target values. Both 757/767 crew members have a FMS Control and Display Unit (CDU). The CDU has multiple display pages that enable flight profiles to be viewed and modified, as well as pages for addressing other flight management functions (figure 6).

Each crew member also has two "glass" displays critical for monitoring the operation of the autoflight system and FMS. These are the Attitude Director Indicator (ADI) (figure 7) and Horizontal Situation Indicator (HSI) (figure 8). The ADI shows the attitude of the aircraft, as well as other information important for monitoring the operation of selected modes. In particular, the ADI displays Flight Mode Annunciators (FMAs) that indicate which modes are engaged or armed for automatic engagement (see figure 7). The HSI displays the position of the aircraft relative to lateral navigation information programmed in the FMS. The HSI enables crew members to tailor this information by selecting the desired display range and viewing mode.

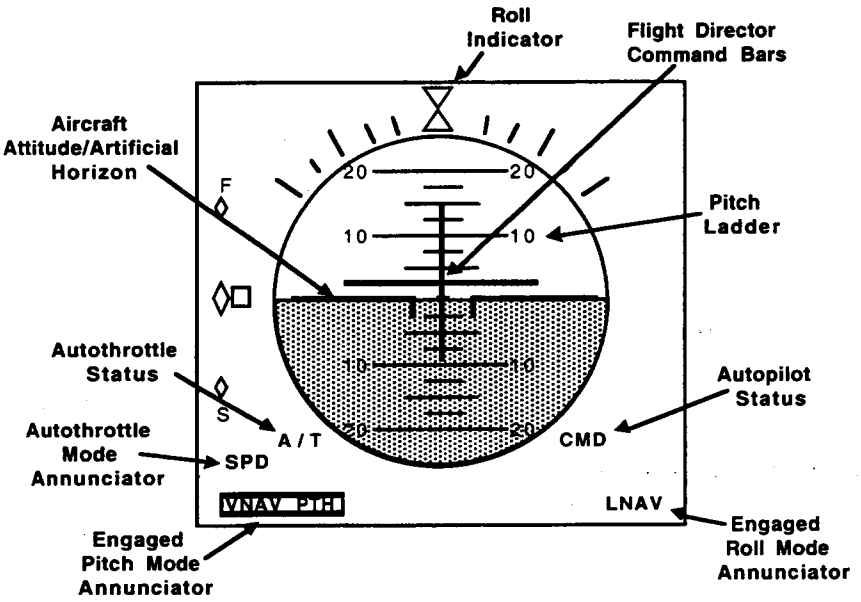


Figure 7. Attitude Director Indicator (ADI).

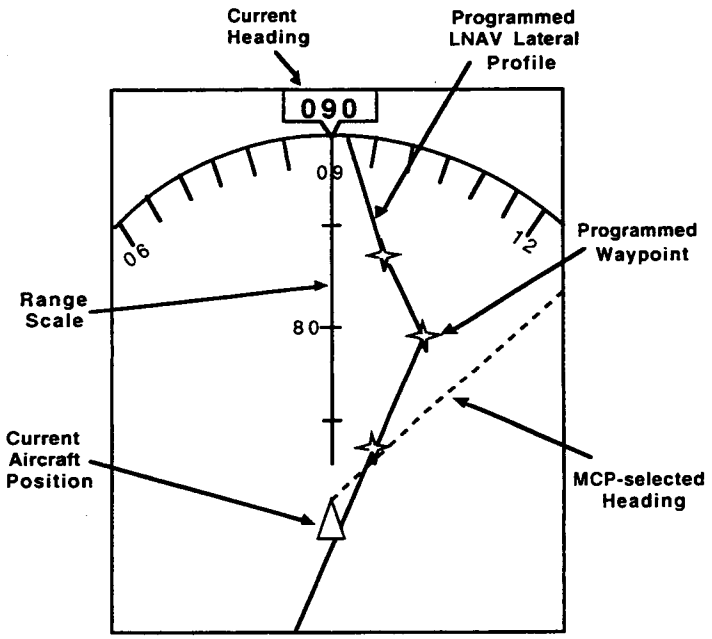


Figure 8. Horizontal Situation Indicator (HSI).

Figure 9 depicts the layout of these controls and displays on the flight deck—a configuration typical of all glass cockpit aircraft. The MCP is mounted on the glareshield between the two pilots. The ADIs are located on the main instrument panel in front of each pilot. The HSIs are located below each ADI. The CDUs are located on the pedestal between the pilots. The location of the CDUs is significant because pilots must look down to use them.

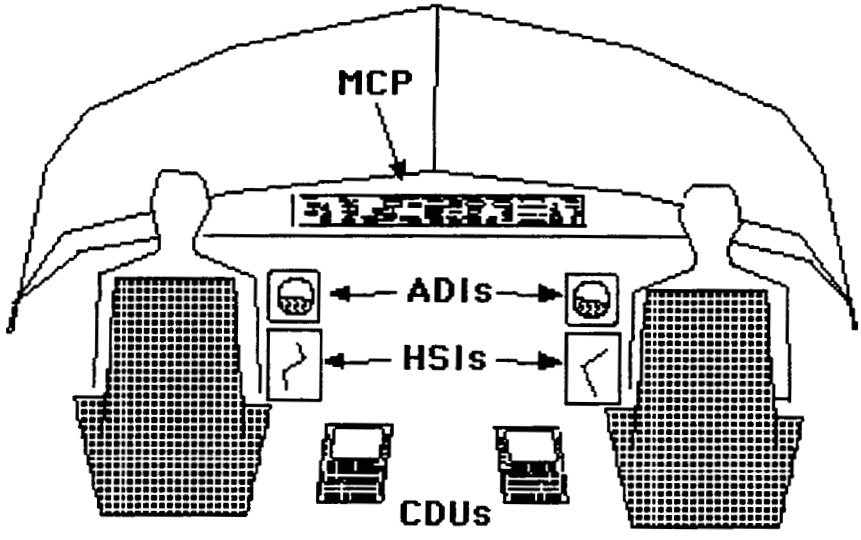


Figure 9. Glass cockpit layout.

Boeing 757/767 automation

This section describes the structure of 757/767 cockpit automation used for flight control and navigation. The first subsection describes the autopilot. The second describes how the modes are used in particular contexts.

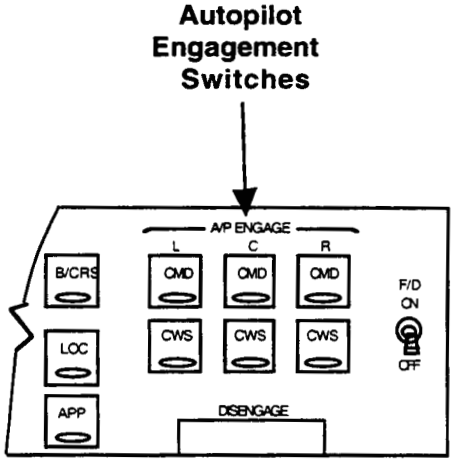


Figure 10. MCP autopilot engagement switches.

757/767 Autopilot

During autopilot operation, pilot inputs made using the MCP (and, in appropriate modes, the CDUs) automatically command the flight control surfaces of the aircraft. The 757/767 has three autopilots, any one of which can be engaged for automatic flight control. Pilots typically engage an autopilot soon after takeoff, using switches on the MCP (figure 10). Pilots select autopilot and autothrottle modes using switches on the MCP. Like the autopilot engagement switches, the mode selection switches are push-on, push-off switches with an integral "on" light to indicate a particular mode is engaged. An engaged mode can be disengaged by pushing a switch again, conditions permitting. A mode's switch light goes off if disengagement is automatically inhibited, or if the mode disengages automatically. An autopilot can be used in either command mode or control wheel steering mode. In control wheel steering mode, the autopilot allows the pilot to use light force on the yoke to control flight manually with assistance from the autopilot servos. Command mode provides fully automatic flight control. When an autopilot is engaged in command mode, the autopilot provides all the capabilities required to reach and maintain the target values set on the MCP. Pilots typically engage an autopilot in command mode soon after takeoff.

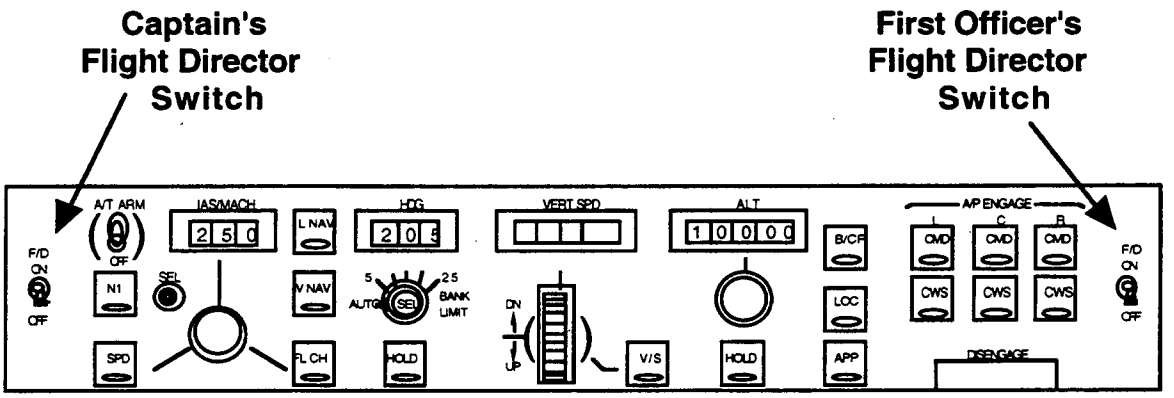


Figure 11. MCP Flight Director switches.

Each crew member has a flight director. When engaged, it positions command bars on the pilot's ADI (see figure 7). If the autopilot is not engaged, the pilot can still select modes and track the flight director command bars manually to follow the profile that the autopilot would command if engaged. Flight director switches are also found on the MCP (figure 11). The autopilot and flight director systems are commonly used together; the flight director command bars provide a means of verifying the control actions of the autopilot. The autopilot command and control wheel steering modes, together with the flight director, provide the pilot with several levels of assisted flight, from manual flight, to control wheel steering, to flight director only, to flight director with control wheel steering, to command with or without the flight director.

The MCP also has a switch for arming the autothrottle system (i.e., making it available for use) (figure 12). The autothrottle is normally engaged prior to takeoff and used throughout a flight. The autothrottle system automatically controls engine thrust by commanding servos for ea ch throttle. Limits on thrust are selected via a separate panel called the thrust selector panel.

Pilots may choose from Climb, Climb-1, Climb-2, or Takeoff thrust—each provides a specific level of engine performance and economy.

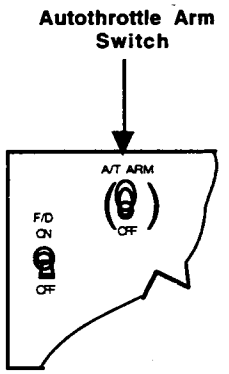


Figure 12. MCP autothrottle arm switch.

The autopilot works closely with the autothrottle. Different autopilot modes may automatically engage specific autothrottle modes, in order to control thrust in a manner complementary to control of the flight control surfaces. This coupling permits the aircraft to fly the desired vertical profile.

757/767 modes

The 757/767 automation modes are organized according to the dimensions of flight they are used to control. The autopilot has roll modes and pitch modes, and the autothrottle provides modes for automatic thrust control. There are eight roll modes, ten pitch modes, and seven autothrottle modes on the 757/767. Although not all of these modes can occur in combination (many are used only for brief or abnormal periods of flight), the 757/767 autoflight system provides pilots with numerous control options.

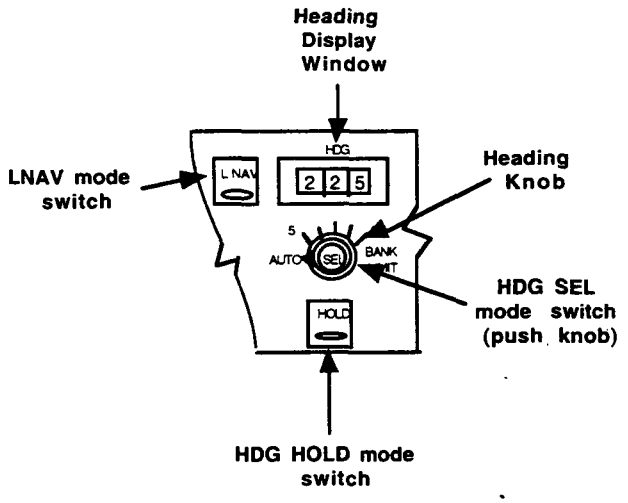


Figure 13. 757/767 lateral mode MCP controls and displays.

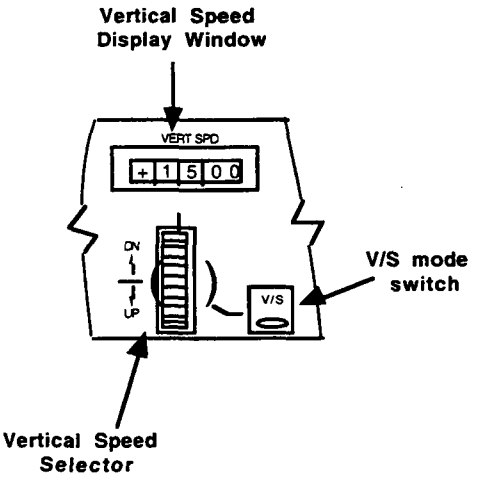


Figure 14. MCP controls and displays for V/S mode.

Modes are structured to provide multiple levels of automation, just as the autopilot and flight director make possible multiple levels of assistance. For example, pilots commonly use three different roll modes to control lateral profile: heading hold (HDG HOLD), heading select (HDG SEL), and lateral navigation (LNAV) . The area of the MCP dedicated to these modes is shown in Figure 13. When the HDG HOLD switch is pushed, HDG HOLD mode maintains the current heading. HDG SEL mode enables the pilot to select a heading on the MCP, and acquire the selected heading. LNAV offers the highest level of automation. LNAV takes input from the FMS to intercept and track a programmed lateral profile from the aircraft's origin to destination.

Modes also differ in the way in which they control a specific aspect of flight. For example, vertical speed (V/S) mode is used to climb or descend at a selected rate by adjusting the aircraft's control surfaces. V/S is an autopilot pitch mode commonly used in combination with the autothrottle speed (SPD) mode, which adjusts thrust to control airspeed; the V/S-SPD mode combination is referred to simply as V/S mode. (In later discussions, commonly occurring pairs of pitch and autothrottle modes are treated together and referred to as "vertical axis modes," or "vertical modes." The mode combination is usually referred to by a single name, e.g., a flight level change (FL CH) autothrottle mode combined with a speed (SPD) pitch mode is referred to as "FL CH mode" for parsimony.) Pilots can engage V/S mode by pushing the V/S mode switch on the MCP. Once V/S mode is engaged, the current vertical speed is displayed on the MCP, and pilots use the thumb wheel to adjust the target vertical speed (see figure 14).

Whereas V/S mode uses the autothrottle SPD mode, flight level change (FL CH) mode uses the FL CH autothrottle mode in conjunction with the autopilot speed (SPD) mode (i.e., there exists a SPD mode for both the autopilot and the autothrottle). In FL CH, the autopilot adjusts pitch to hold the current airspeed, while the autothrottle adjusts thrust to climb or descend. Pilots speak of speed being "on pitch" in FL CH mode (i.e., speed is controlled via pitch adjustments), and "on thrust" in V/S mode (i.e., speed is controlled via thrust adjustments). In both V/S and FL CH mode, the MCP airspeed/mach display window allows speed to be adjusted (see figure 15).

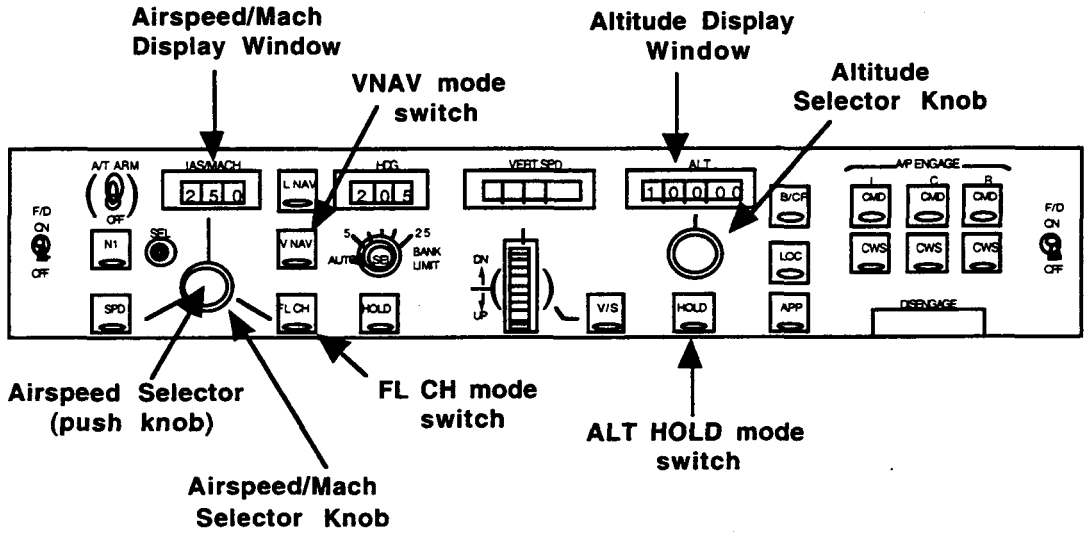


Figure 15. MCP vertical mode controls and displays.

V/S and FL CH are two of the 757/767's vertical modes; the two others are altitude hold (ALT HOLD) and vertical navigation (VNAV). ALT HOLD is used in a manner analogous to HDG HOLD; pushing the MCP ALT HOLD switch levels the aircraft at the current altitude. Figure 15 shows the MCP controls and displays required to use vertical axis modes.

Vertical navigation (VNAV) enables fully automatic FMS control over the programmed vertical profile. In VNAV mode, autothrottle modes are "slaved" to provide the appropriate thrust control. ("VNAV mode" is a very general term, as VNAV can be thought of as having multiple submodes that occur in combination with different autothrottle modes.) VNAV mode is the highest level of vertical profile automation, and maximizes fuel economy; FL CH, on the other hand, enables fast climbs or descents.

The MCP-selected altitude is one of the most important inputs pilots make. Pilots set the MCP altitude to the altitude cleared by Air Traffic Control (ATC) before engaging a vertical mode. In fact, if an altitude different from the aircraft's current altitude is not set on the MCP, neither FL CH nor VNAV will engage. In VNAV, the MCP-selected altitude limits the aircraft's climb or descent, regardless of the programmed vertical profile. This gives rise to a number of difficulties, including: forgetting to set a lower altitude in cruise, so that VNAV cannot descend; or, setting an altitude beyond a speed/altitude restriction, then inadvertently erasing the restriction from the CDU, so that the restriction is ignored on the way to the MCP-selected altitude. To further complicate matters, V/S can fly away from the MCP selected altitude (e.g., an altitude can be reached via V/S climb, then a negative vertical speed can be used to fly into terrain with no altitude protection). Some pilots/airlines standardize the use of the MCP-selected altitude, requiring that the nearest cleared altitude directed by ATC is always set before a vertical mode (other than ALT HOLD) is engaged.

Another automation feature that impacts vertical mode use in the automatic altitude capture (ALT CAP) mode. ALT CAP engages automatically, disengaging the vertical mode, when the aircraft is approaching the MCP-selected altitude (ALT CAP only engages automatically, so it has no mode switch on the MCP). ALT CAP smoothes the g-forces involved with the capture maneuver, then ALT HOLD mode engages automatically to hold the MCP-selected altitude. These mode transitions are tied to the AFDS altitude alerting system, which provides visual and aural alerts as the aircraft approaches the MCP-selected altitude. The altitude alerting

system also warns pilots of deviations from the selected altitude (e.g., in V/S mode).

Automation use

Pilots are trained to use the automation in a manner consistent with the philosophy and guidelines of the managing air carrier. Guidelines vary slightly among carriers. This subsection describes how the automation is normally used following one major carrier's guidelines. It also notes some other mode usage techniques that, although not officially taught, are widely accepted and used by line pilots.

Automation use begins before takeoff, when the pilots program the planned flight information and performance parameters into the FMS via the CDUs. Information about the flight's origin and destination airports, planned departure procedures and (if known) arrival procedures are programmed, along with the waypoints to be crossed during the high-altitude portions of flight. This information defines the lateral and vertical profiles. With this information, the autopilot can use information from the FMS waypoint database and the aircraft's inertial reference system to navigate in LNAV and VNAV modes.

Also before takeoff, the pilots turn their flight directors on and position the autothrottle switch to ARM. This arms the autothrottle in takeoff mode—a special purpose mode only used for takeoff. HDG HOLD is engaged with the runway heading selected on the MCP. To takeoff, the pilots advance the throttles and the autothrottle assumes control of thrust. At rotation speed, one crew member, designated the "pilot flying" (PF), rotates the aircraft to the pitch indicated by the flight director, and holds the heading indicated by the flight director. Once airborne, the pilot-not-flying (PNF) retracts the landing gear, and begins to retract the flaps according a speed schedule specified before takeoff. At the point at which the aircraft exhibits a positive rate of climb, the climb phase of flight begins.

At 1,000 feet above ground level (1,000 feet AGL), the PNF engages a vertical mode to be used for climbing, engages the autopilot in command mode, and sets the limit thrust on the thrust selector panel. The pilots now select a vertical mode. Guidelines dictate that if the appropriate departure information is programmed in the FMS, VNAV should be used; otherwise FL CH should be used. Even if the FMS is properly programmed, the crew may opt to use FL CH in order to expedite the climb to a required altitude because of traffic, terrain, and/or weather. FL CH mode might also be used to enable rapid modifications to the flight plan, without reprogramming the FMS

After the autopilot is engaged by selecting command mode following takeoff, pilots use HDG SEL mode to fly heading(s) specified by ATC until the FMS-programmed lateral profile can be intercepted. When on a heading that intercepts the route programmed in the FMS, the PNF arms LNAV. Figure 8 shows how the HSI looks when an intercept heading is selected; the dashed line indicates the MCP-selected heading intercepts the FMS programmed route. When the FMS route is intercepted, LNAV mode engages automatically and the aircraft turns onto the route. As long as the lateral profile is valid, pilots normally remain in LNAV mode. If, however, ATC requires a different heading, pilots either revert to HDG SEL or, if they have time, reprogram the FMS. Pilots may also use HDG HOLD to maintain a heading—a way to stop a HDG SEL turn or prevent a programmed LNAV turn. They may also use a heading hold submode of LNAV to hold the aircraft's heading after flying beyond the last programmed waypoint. HDG SEL, HDG HOLD, and LNAV are the roll modes commonly used to handle lateral navigation.

Unlike lateral axis modes, the use of vertical modes is closely tied to phase of flight. VNAV, in particular, performs differently depending on the phase of flight. Figures 16 shows the various manifestations of VNAV during the climb and initial cruise phases of flight; figure 17 shows how VNAV works during the cruise-to-descent and descent phases.

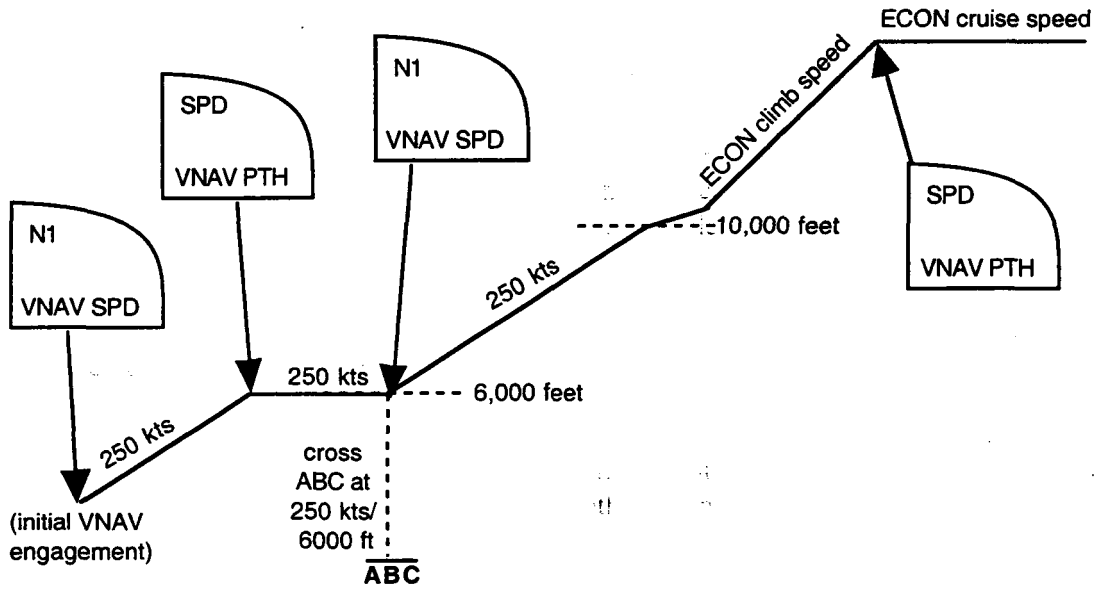


Figure 16. Typical VNAV profile and mode annunciations during climb from VNAV engagement to cruise flight.

The VNAV profile shown in Figure 16 begins with initial VNAV engagement following autopilot command mode engagement and thrust selection after takeoff. If no waypoint crossing restrictions are programmed into the FMS, pilots fly a default (i.e., federally mandated) 250 knot climb profile to the 10,000 feet mean sea level (10,000 feet MSL) transition altitude using the VNAV SPD submode of VNAV. In cases where a speed/altitude restriction is programmed at a waypoint (e.g., cross ABC at 250 knots and 6000 feet), VNAV SPD changes to VNAV path (VNAV PTH) at the altitude restriction and remains in VNAV PTH until after the waypoint is passed.

To comply with a speed/altitude restriction, such as that at waypoint ABC in figure 16, pilots must exercise care in setting the MCP altitude. If ATC cleared the aircraft to 10,000 feet MSL before takeoff, the crew may set10,000 feet on the MCP and "trust the automation" to handle the level-off at the programmed crossing restriction and resume climbing to 10,000 feet after the waypoint is passed. If, however, ATC only cleared the aircraft to the crossing restriction, then the crew must set 6,000 feet as the cleared altitude on the MCP. In this latter case, the autopilot will automatically transition through ALT CAP mode into ALT HOLD at 6,000 feet, disengaging VNAV in the process. When ATC clears the aircraft to a higher altitude the crew must set the new altitude on the MCP and reengage VNAV. Above 10,000 feet VNAV commands the most economical thrust setting for the climb. Each VNAV climb that is terminated by an MCP-selected altitude lower than the FMS-programmed cruise altitude causes an automatic transition to ALT CAP, then to ALT HOLD at the MCP altitude. When the aircraft reaches the programmed cruise altitude at the FMS-computed top-of-climb (T/C) point, VNAV PTH engages in conjunction with the autothrottle SPD mode to maintain the most economical cruise speed.

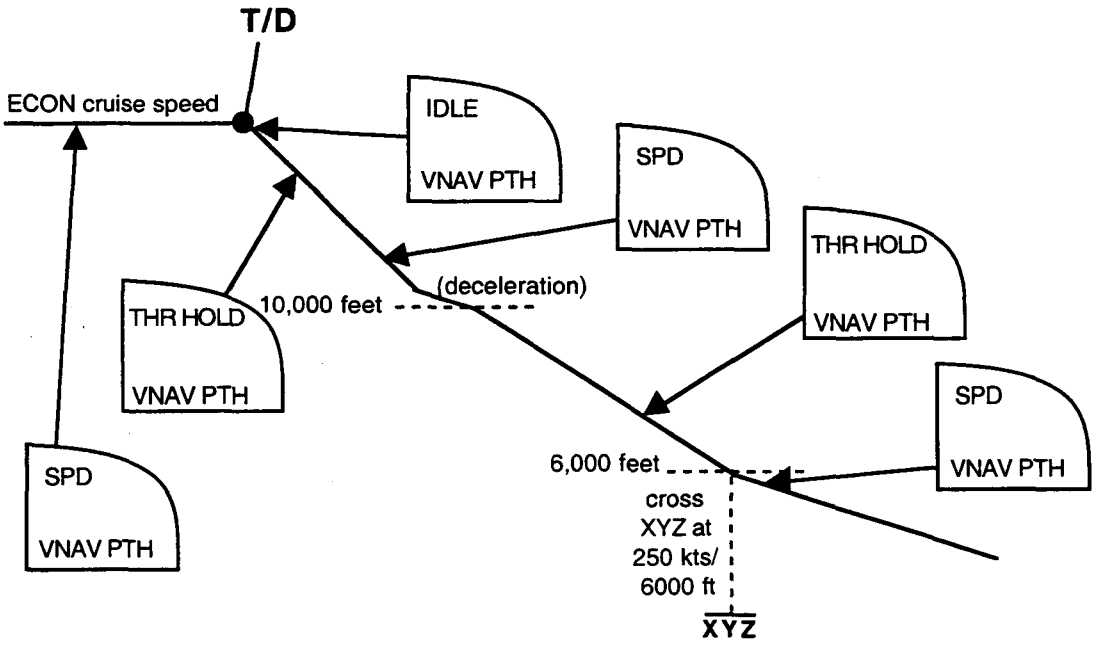


Figure 17. Typical VNAV profile and mode annunciations during descent from cruise to approach.

VNAV PTH is used throughout the cruise phase of flight, until the FMS-computed top-of-descent (T/D) point is reached (see figure 17). At the top-of-descent, VNAV PTH mode adjusts its associated autothrottle to command thrust in such a way to track the programmed descent profile. Depending on winds aloft, the autothrottle may reduce thrust to idle (IDLE mode), hold a specific thrust (THR HOLD mode), or add thrust (SPD mode). VNAV PTH is used with SPD mode to decelerate prior to a transition altitude (i.e., an altitude where a speed change is required). In general, VNAV uses a FMS-computed path to meet a speed/altitude restriction at waypoint on the descent profile. In cases where a strong tailwind is present, pilots may extend spoilers or speed brakes to slow the aircraft.

During VNAV operation, speed is commanded by the FMS; the MCP speed display window is blank. If ATC requires a speed that differs from the FMS-programmed speed at any time while VNAV is engaged, the pilot may engage a speed intervention submode of VNAV to achieve the desired speed without reprogramming the FMS. Speed intervention is invoked by pushing the MCP speed selector knob to display the current airspeed in the MCP speed display. The pilot may then adjust the airspeed with the speed selector knob (see figure 15). To revert to normal VNAV operation, the pilot pushes the speed selector knob a second time to blank the MCP speed display.

To use FL CH, the pilot first sets the new cleared altitude on the MCP. The pilot then presses the FL CH switch (see figure 15). As soon as FL CH engages, the MCP speed display changes to the current airspeed. The pilot should then check and adjust the speed as appropriate. In contrast to FL CH, V/S is used less often because of its capability to depart from a set altitude. It may be used, however, in situations where the pilot desires smooth level-offs that might cause passenger discomfort if left to VNAV or ALT CAP modes.

Other vertical modes may also be used at top-of-descent. Pilots may use V/S to smooth the initial descent; they may also use V/S at other points in the descent where a gradual descent profile is desired. FL CH affords the fastest descent to a lower altitude; it is especially effective in busy, low altitude situations.

Besides the major lateral and vertical modes described here, several additional special-purpose modes are available on the 757/767. For example, the glideslope (G/S) and localizer

(LOC) modes enable the glideslope and localizer beams to be intercepted on approach.

Summary of 757/767 automation operation

The 757/767 glass cockpit automation provides autopilot modes to control the aircraft's lateral and vertical profile. Lateral profile modes include LNAV, HDG SEL, and HDG HOLD. Vertical profile modes include VNAV, FL CH, V/S, and ALT HOLD. In addition, ALT CAP mode engages automatically whenever the aircraft approaches the MCP-selected altitude, and smoothes the automatic transition to ALT HOLD. As noted above, vertical modes are, in actuality, combinations of an autopilot pitch mode and an autothrottle mode; where insignificant, these distinctions are eliminated for parsimony.

Pilots use four major components in the glass cockpit to control and monitor the 757/767 automation, in addition to standard flight instruments. These components are the MCP, CDUs, HSIs, and ADIs. The CDUs enable information to be programmed into the FMS, for use when the autopilot is coupled to the FMS in LNAV and/or VNAV modes. Other autopilot modes acquire MCP-selected target values. In the next section, general classes of modes are characterized.

Classes of modes

A mode, in general, is a manner of behaving (ref. 71). In supervisory control systems, the behavior referred to can be either that of a display or input mechanism, or that of automation used to control the system. Modes related to display or input mechanisms are called "interface modes" or, using Degani et al.'s classification, "format/data-entry modes;" modes that determine the behavior of automation used to control the system are "control modes."

Format/data-entry modes

Format/data-entry modes first arose in human-computer interfaces; multiple interpretations of the same keys were needed to support expanding functionality. A boon to interface designers, modes were used to group related commands into a unit operated on as a whole (e.g., enabled or disabled). Modes could "correspond to a meaningful activity in the user's mind, such as 'editing,'" and thereby simplify the user's choices in a given mode (ref. 72, p. 440). Users, however, were not necessarily convinced. Tesler, an advocate of modeless interfaces (ref. 73), defined a mode as follows:

> "A mode of an interactive computer system is a state of the user interface that lasts for a period of time, is not associated with any particular object, and has no role other than to place an interpretation on operator input (ref. 74, p. 659)."

Modes nonetheless proliferated, and with them a growing need to understand their associated pitfalls. Mode errors—already identified as a category of unintentional, erroneous slips of action that occur when humans incorrectly assess a situation, then perform an action inappropriate for the actual situation (ref. 75)—took on new meaning as modal devices (e.g., text editors) entered widespread use. Humans sometimes lose track of which mode of the device is currently active, then perform an action inappropriate for the mode (ref. 76). The *vi* text editor, with its "command" and "insert" modes, is a popular illustration of format/data-entry modes (ref. 59). Another example is the degrees/radians mode distinction found on calculators. Unlike the *vi* example, the difference in behavior is not immediately evident: when a user inputs 3.14159, it is displayed as 3.14159. However, the mode affects the interpretation of 3.14159, once this value becomes part of a trigonometric calculation; it also affects the correct interpretation of the result.

In general, format/data-entry modes succeed if the user can always ascertain the state of the system, and if actions available during the mode are always relevant to the mode (refs. 74

and 77). Early studies on feedback and mode usage include that of Monk (ref 78), who showed that auditory feedback can help reduce mode errors, and Sellen, Kurtenbach, and Buxton (ref. 79), who examined the utility of visual and kinesthetic feedback. Enduring computer interface features such as menus and dialog boxes were developed to constrain user actions in a particular mode (ref. 80).

Control modes

The purpose of control modes is to provide the human operator with options for controlling the behavior of automation. A given control mode, once engaged, varies or maintains a certain set of parameters in a particular fashion. The dynamic response of the controlled system created when a control mode is engaged is therefore a factor that occasions its use. Automobile cruise control is an example of a simple control mode. Control modes have five important characteristics (figure 18). First, a given control mode has specific *engagement conditions*. The engagement conditions for a mode encompass target values that must be set so the mode can attain and/or maintain them, and the mode(s) that are currently in use. For example, Flight Level Change (FL CH) mode requires the pilot to enter an altitude target on the Mode Control Panel (MCP) that is different from the current altitude target. Vertical Speed (V/S) mode engages if FL CH is engaged and the autothrottle is engaged in N1 mode.

Second, some control modes can be armed for later automatic engagement. In such cases, *arming conditions* govern when the mode can be armed; engagement conditions dictate when the mode will engage automatically. For example, VNAV can be armed if a valid vertical profile is programmed and the glideslope is not captured. With VNAV armed, if a valid MCP altitude target is entered and the aircraft intercepts the programmed vertical profile, VNAV engages automatically.

Third, a control mode has *disengagement* conditions that govern when the mode disengages; a mode may disengage when another

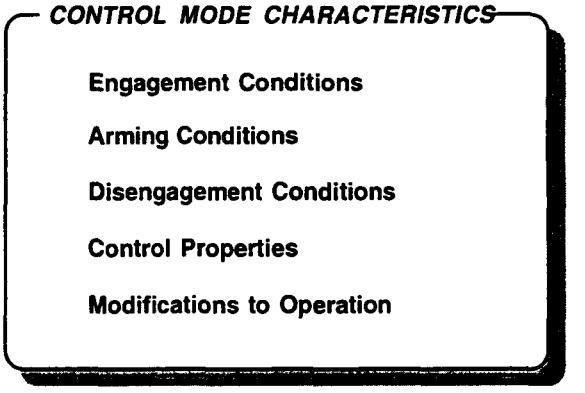


Figure 18. Characteristics of control modes.

mode is engaged, or when critical target value information no longer applies. For example, the Lateral Navigation (LNAV) mode disengages when Heading Select (HDG SEL) mode engages. VNAV disengages if the programmed vertical profile is no longer valid.

Fourth, a given control mode has characteristic *control properties* that include the subsystems used by or controlled by the mode, the specific set of parameters that the mode controls, and the manner in which the mode controls them. One mode may control the same set of parameters as another, but it may use different sources of information, and a different means of controlling the parameters. These properties are, in effect, the reason for including the mode in a system's automation suite—the mode provides control properties that are desirable in certain operational situations, and are not provided by another mode. In addition to the specific parameters that a mode controls, the level of automatic control excercised over the parameters also defines its control properties.

Levels of automation

Historically, each new element of automation and its enabling technology is added to the previously existing automation without replacing it. This design affords the human operator the opportunity to disengage the latest additions to the automation and revert to a familiar manner of controlling system. It also permits safe operation of the controlled system should the new automation fail (ref. 8).

Today's complex control automation typically follows suit, allowing human operators to choose among several levels of automation (figure 19). At low levels of automation, the operator performs control tasks manually with assistance from the automation. Higher levels of automation enable the operator to input desired system state values for the automation to achieve and maintain. The highest levels of automation essentially control the system autonomously, while the operator monitors the automation to ensure desired system performance. Billings (ref. 8) views these levels as a controland management continuum, similar to the levels of supervisory control discussed by Sheridan (ref. 9). As the level of automation increases, direct operator control decreases and monitoring responsibilities increase (figure 19).

Controlled subsystems and parameters

In complex supervisory control systems, a given system is comprised of several subsystems. The control properties of a mode are also characterized by the subsystems and parameters that the mode controls. Modes exist for controlling salient aspects of the performance of each subsystem. In glass cockpit aircraft, for example, three aspects (i.e., pitch, roll, and thrust) must be controlled simultaneously to achieve the desired flight path. Furthermore, several modes are available for controlling each of these parameters at each level of automation. Pilots can invoke a single mode at a high level of automation that integrates control over more than one of these parameters (e.g., pitch and thrust) to reach a desired altitude at a desired time. Alternatively, they can use multiple modes concurrently to control each

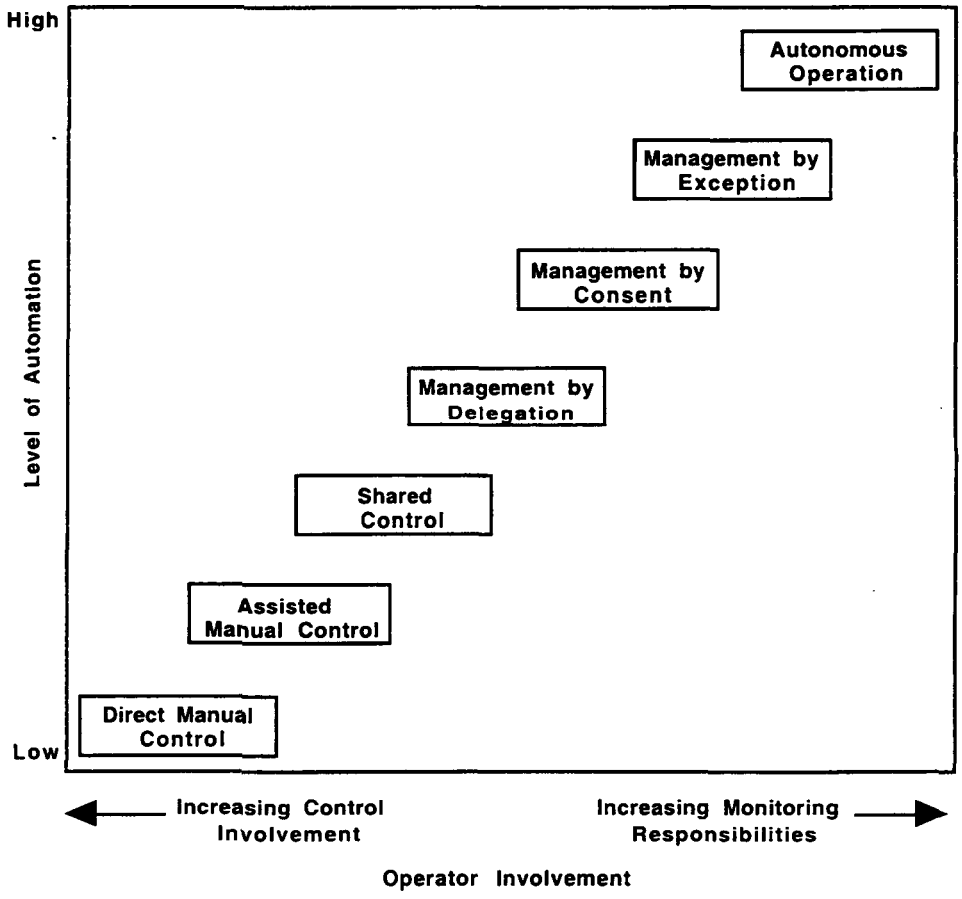


Figure 19. Levels of automation (adapted from Billings, (ref. 8)).

parameter separately at a lower level of automation to maintain, for example, a desired altitude and airspeed.

Thus, the control properties of a mode can be thought of as two-dimensional: one dimension corresponds to the parameters of each sub-system that are controlled; the other dimension corresponds to the level of automation. At high levels of automation, a single mode can control more than one performance parameter by automatically "slaving" another mode for its own use. VNAV, for example, routinely changes the autothrottle mode that controls thrust as necessary for its pitch commands to produce the vertical profile prescribed by the information programmed in the FMS.

Finally, a control mode is characterized by allowable *modifications to operation* that human operators (or other automation) can make while the mode is engaged. A mode may have submodes that allow temporary specification of target values different from those programmed prior to engaging the mode. For example, VNAV's speed intervention submode allows pilots to override the target speed programmed into the FMS if the desired airspeed differs from the programmed value. Thus, submodes provide a way for human operators to temporarily revert to a lower level of automation in which more direct control of the system is possible. Submodes can also refer to the automatic input of a default target value in a situation where the current input fails to meet specified criteria (e.g., envelope protection in the Airbus A320) (ref. 81).

Format/data-entry modes for control modes

The purpose of format/data-entry modes is to provide increased functionality of a system while using the same input mechanism and display space. The important feature of format/data-entry modes is that the same input results in different behavior. In isolation, format/data-entry modes are reactive—nothing happens until the operator performs another action. Control modes, on the other hand, are proactive in that they automatically transform the controlled system.

Operator interfaces to control modes in complex systems, however, routinely incorporate format/data-entry modes (e.g., to allow input of target values, and configure displays for monitoring the automation). This relationship imparts a proactive quality to the format/data-entry modes. Mode errors related to format/data-entry modes can propagate to create control problems unbeknownst to the operator—erroneous inputs that the operator would usually discover are immediately honored by associated control mode. Thus, while this research is concerned primarily with control modes, the importance of format/data-entry modes should not be understated.

Mode structure

The characteristics of individual control modes give rise to specific relationships between modes. Each subsystem may have its own set of modes, and therefore the modes of a given subsystem can interact with the modes of another. Degani et al. (ref. 59) use the term *mode structure* to refer to the hierarchy of modes in a system, the transitions among modes and associated transformations in the controlled system, and the interactions between modes of different sub-systems. The hierarchy of modes in a system derives from the characteristics of the individual modes and the level of automation at which they operate. Interestingly, this concept of mode structure is little changed from that embraced by early research on format/data-entry modes: "The natural relationship among these modes gives the space of modes its structure, which governs the allowable transitions between the various modes (ref. 72, p. 440)."

Mode transitions

Mode transitions are an important facet of mode structure. Degani et al. (ref. 59) state that a mode transition can result from three types of input: manual, automatic, or automatic/manual. A related view of mode transitions is offered by Vakil et al. (ref. 81). They

also identify three types of mode transitions: commanded, uncommanded, and automatic/conditional. The difference appears to be that Degani et al. characterize the inputs required to transition to the mode, while Vakil et al. characterize the transition itself.

For purposes of this research, there are four types of mode transitions. First, *manual* mode transitions are those that can only be directly and immediately effected by the human operator. For example, a transition to HDG SEL can only occur directly as a result of a pilot pressing the HDG SEL engagement switch. Second, *automatic* mode transitions are those that only occur automatically as a result of some target state being attained. For example, a transition to Altitude Capture (ALT CAP) mode to capture a set target altitude only occurs automatically; no engagement switch exists for this mode.

A third type of mode transition is *automatic/manual*. Automatic/ manual mode transitions are those that can occur either as a result of pilot input, or attainment of a specific target state. An example is a transition to Altitude Hold (ALT HOLD) mode, which occurs automatically following the altitude capture maneuver, or can be effected immediately by the pilot to hold the current altitude by pressing the ALT HOLD engagement switch.

Fourth, *conditional* mode transitions refer to modes that can be armed for later engagement, or engaged immediately if the target state conditions are already met at the time of input from the human operator. An example is LNAV, which can be armed to intercept the lateral profile programmed in the FMS, or can engage immediately if the aircraft happens to be on the programmed profile already when the pilot presses the mode switch.

For completeness, a fifth type of mode transition is one that cannot occur because the engagement conditions for the mode are not met (or the disengagement conditions for the current mode are not met), regardless of whether the human operator attempts to engage the new mode. For example, FL CH will not engage unless the pilot first enters an MCP target altitude different from the present altitude; attempting to engage FL CH without first setting a new altitude constitutes a (benign) pilot error.

Base-modes and macro-modes

Vakil et al. (ref. 81) provide an additional perspective on mode structure by distinguishing between base-modes and macro-modes. Base-modes simply maintain an invariant set of targets, while macro-modes consist of a linked sequence of base-modes. Because each base-mode in the macro-mode has its own set of targets, the macro-mode, in effect, has a set of targets which vary over the course of the its operation. They offer the autoflight system Autoland sequence as an example of a macro-mode, in which automatic transitions from a vertical mode, to Glideslope capture, to Flare, and finally to Rollout occur. Another example is a standard altitude capture maneuver, in which the aircraft transitions from the mode used to change altitude, to ALT CAP, and finally to ALT HOLD at the desired altitude. Sherry, Youssefi, and Hynes (ref. 82), in their specification of a formalism for the development of next generation automation, provide a related view. They first define primitive modes, then construct supermodes from the primitive modes. This approach holds promise for designing mode structures that are mathematically consistent in their behavior—one potential solution to automation surprises.

Cognitive factors impacting mode usage

Mode structure affects the cognitive demands placed on operators of complex systems, and therefore influences the performance of human operators using modes of automation. The more complex and highly automated the task environment is, the more susceptible operators are to mode errors (ref. 25). Four cognitive factors affect the performance of human operators of complex systems: knowledge factors impact knowledge use in various problem solving contexts; strategic factors drive tradeoffs in the face of changing objec-

tives, limited time, and high risks; attentional dynamics affect situation awareness and effective attention allocation in high workload periods; and, finally, bounded rationality leads to satisficing behavior that makes sense to humans in light of the other factors (ref. 11). The tasks involved in the selection and use of automation modes in complex systems provide examples of these factors at work. In some situations, the mode management task can cause demand-resource mismatches that lead to mode errors.

Mode usage tasks

When an operational objective is communicated to the operator, the first task is to select a mode from the modes available to accomplish the objective. The operator next programs or configures the automation with information required by the selected mode, and engages the mode. Upon mode engagement, the operator monitors the operation of the automation to ensure that the desired mode engages properly, and that the behavior of the controlled system meets expectations. Some modes require the operator to arm the mode, then monitor the conditions for automatic mode engagement. In certain situations, the operator may meet an operational objective by adjusting the operation of a mode that is already engaged by reprogramming target values required to use the mode, or by engaging a submode of the mode that provides the required control behavior.

Knowledge factors

To use modes effectively, operators must understand how a particular mode should be used in conjunction with other modes, and the type of control needed in the current operating context. A clear understanding of mode structure is critical to an operator's ability to properly adjust the operation of a given mode, effect a transition between modes, or monitor mode transitions effected automatically by the automation. In complex systems, such knowledge requirements are a significant addition to the operator's task (ref. 11).

In complex systems, some subsystems are highly automated—much more so than other subsystems. The FMS, for example, requires an disproportionate increase in the depth of knowledge required to use it (refs. 67 and 58); hence, operators may develop "buggy" or incomplete mental models of how this automation functions (refs. 69 and 70).
Operators must also be aware of the completeness and accuracy of their knowledge. Because their knowledge of FMS function is often incomplete or inaccurate, pilots are known to develop a small set of reliable strategies, involving a few modes—which may not be adequate in critical or abnormal situations (ref. 11). Other knowledge factors are inert knowledge, when facts about mode structure are known but cannot be applied in actual operating contexts (ref. 69), and oversimplifications that result when heuristics are used inappropriately (ref. 83).

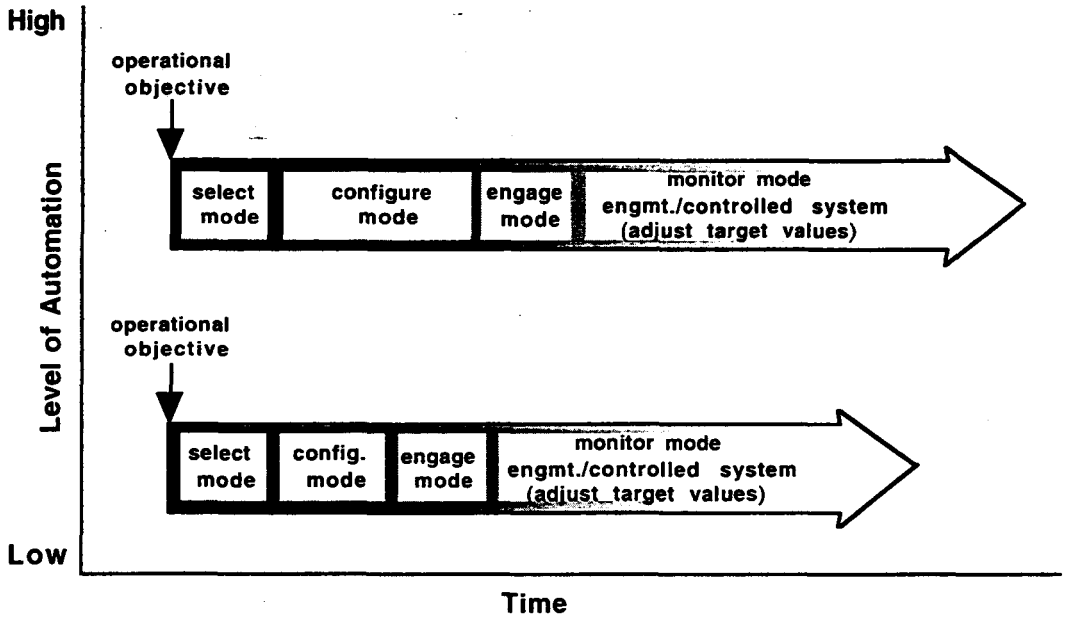


Figure 20. Operation of modes at different levels of automation.

Strategic factors

When selecting a mode, operators must consider the urgency with which a desired system state must be achieved, and the need to achieve safe and efficient system performance. Operators are often faced with tradeoffs—the penalty for not meeting a particular operational objective may be greater than the reward for meeting a competing objective. For example, modes that provide a high level of automation control the system more precisely and efficiently than modes at lower levels of automation. However, high-level modes ordinarily require more time to prepare for use (figure 20). Eldredge et al. (ref. 58) found the use of high levels of automation provided by the FMS detrimental in high-workload situations. If sufficient time is not available to program the automation, or operational objectives are likely to change in the near future, operators typically sacrifice the improved efficiency offered by a high level of automation for more direct control at a lower level of automation (ref. 69).

Another type of strategic factor plays a role in situations where the operator is responsible for the safe operation of the system, but has transferred authority to high-level automation. Woods et al. (ref. 11) term such difficulties "responsibility-authority double binds." Operators must correctly adapt ambiguous or inadequate guidelines for using the automation to the situation at hand (ref. 84), rather than permitting high-level automation to mishandle the situation. Pilots can be surprised by the automation failing to take expected actions, or taking uncommanded actions (ref. 70). Pilot strategy also includes "tricking" the FMS to achieve, for example, an early VNAV descent. Wiener (ref. 67) warns: "It does not speak well for automation that pilots of a modern airliner must deliberately enter incorrect data into a sophisticated computer to achieve a desired objective (p. 171) ."

Attentional dynamics

Attentional dynamics encompass "the factors that operate when cognitive systems function in dynamic, evolving situations (ref. 11, p. 67)," including workload management and control of attention. Many processes, including directed attention, perceptual processes, mental simulation, and mental bookkeeping,

are referred to generally as situation awareness (ref. 70). In complex task environments where modes are present, the term mode awareness has come to refer to an abstract level of vigilance and acumen required to manage the operation of multiple modes concurrently with other tasks. To maintain mode awareness, pilots of glass cockpit aircraft must know "who/which system is in charge of controlling the aircraft, what the active target values are, and whether they can anticipate the status and behavior of the FMS (ref. 69, p. 36?)." In short, they must be able to answer Wiener's (ref. 67) three questions: "What is it doing now?," "Why is it doing it?," and "What's it going to do next?"

Clumsy automation often increases workload at times when it was already high, and reduces workload at times when workload is typically light (ref. 67). Pilot workload becomes especially high at low altitudes near airports. Because pilots must meet objectives concerning both the lateral and vertical profiles, the selection and use of one mode must be performed simultaneously with another mode. This can create numerous attentional conflicts that may lead to a loss of awareness about the operation of the automation. For example, figure 21 shows that configuring one mode can compete with monitoring another (at time t_1), two time t_2), or a selection decision for one can compete with monitoring another (at time t_3). Operators can also become fixated on one element of the automation at the expense of other attentional demands (such as monitoring and collision avoidance)— pilots go "head down," for example, to program the FMS when they should be attending to other tasks, such as monitoring traffic (refs. 19 and 67).

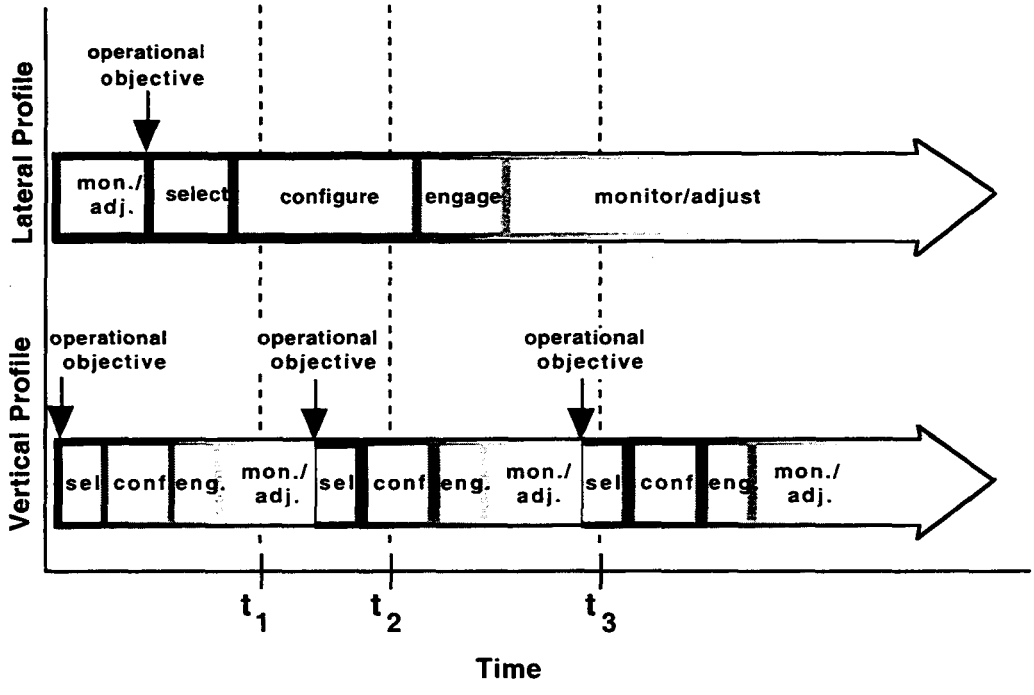


Figure 21. Setup/Engagement or monitoring/adjustment of one mode can compete for attention with another mode during high workload periods.

Attentional factors are compounded by automation. Automation is often machine-centered, leading researchers to characterize it as "strong and silent"; no increase in observability accompanies an increase in automation (ref. 70). The 1985 incident involving a China Airlines 747 (ref. 65) is an example of strong and silent automation. The aircraft experienced a loss of power in its outer right engine, which the autopilot compensated for—until it could no longer. The aircraft then plummeted 31,500 feet and sustained serious damage before the crew could recover, because the autopilot failed to alert the pilots that it could no longer compensate for a loss of engine power. Norman (ref. 85) suggests this example shows automation is not powerful enough—if it were made more powerful, perhaps it could provide the feedback necessary to better inform operators about its control capabilities.

Combined effects: bounded rationality

Bounded rationality—the idea that human problem solvers possess limited cognitive capabilities—leads to "satisficing" behavior in which humans do what seems reasonable in light of their knowledge, objectives, and limited attentional resources (refs. 86 and 87). Indeed, rationality must be bounded—to bring all potentially relevant information to bear would be overwhelming. Knowledge factors, strategic factors, and attentional dynamics interact to determine which resources are brought to bear on mode selection and use in a given situation. For example, the amount of time required to configure a mode for use is a function of the complexity of the programming task; the operator must understand the characteristics of the mode and decide whether attention can affordably be allocated to its use in the face of changing objectives. Knowledge, strategic, and attentional factors also impact operator decisions to decrease the amount of attention they need to devote to monitoring by opting for a lower level of automation when they are anticipating a period of high workload (ref. 11).

Automation modes can also discretize aspects of the otherwise continuous operation of the controlled system. For example, in the Bangalore crash (ref. 88), the pilots inadvertently engaged an Airbus A320 mode called Open Descent, which provides no altitude protection and led to the crash. Open Descent mode engaged automatically when the pilots entered a lower target altitude when the aircraft was already within 200 feet of the MCP altitude. If the pilots had entered the new target altitude when the aircraft was 205 feet from the set altitude, the accident might not have occurred.

A combination of factors might have played a role in this disaster; indeed, a detailed analysis of the accident highlights several factors (ref. 56): the captain was conducting a check flight, and the proper division of labor between crew members was not followed; the trainee pilot disengaged one, not both, flight directors, then became confused and fixated on the failure of the autothrottle to leave idle descent (once he realized he was in that mode); and, the crew relied on an A320 envelope protection feature to recover, but a time delay designed into the system caused the protection feature to engage too late.

Summary

Modes are useful because they provide control options to the human operator. However, the number of available modes, along with possible interactions between modes that occur when several modes can be used in combination, increases the potential for mode error. A range of possible mode configurations makes it easier to lose track of which modes are currently controlling the system, especially since the same controls and displays are often used differently depending on the modes in use.

The use of a given mode encompasses knowledge, attentional, and strategic factors depending upon its implementation. Supervising the concurrent operation of multiple modes, besides resulting in increased workload, can lead to misunderstandings about how

or when a particular mode should be used in conjunction with other operational modes, and misunderstandings about the type of control needed in a given situation. In situations where the operator must adjust the operation of a given mode, effect a transition between modes, or monitor mode transitions effected automatically by the automation, several of these factors can conspire to cause errors. Mode management is error-prone; therefore, operators supervising the operation of multiple modes to control complex systems are likely to benefit from operator's associates, context-sensitive displays, and intelligent tutoring systems—three important applications of intent inferencing.

This chapter provided background on modes in complex systems, which must first be understood in order to design a methodology for correctly predicting and interpreting operator actions. An understanding of modes is especially important for developing models suitable for supporting intent inferencing. The OFM-ACM is a model designed to represent knowledge required to effectively manage the operation of multiple modes. The OFM-ACM, along with the other elements of the GT-CATS activity tracking methodology and architecture is the subject of the next chapter.

4. A Methodology and Architecture for Activity Tracking

Introduction

This chapter describes the GT-CATS activity tracking methodology, along with a computer architecture for implementing the methodology to track operator activities in real time. An activity is simply something the operator does, expressed at any level of abstraction. *Activity tracking* is a machine capability analogous to the human supervisory controller's task of tracking the status and behavior of the controlled system, and anticipating future changes (ref. 11) Activity tracking entails predicting operator activities, explaining operator actions, and flagging possible operator errors, in light of the status and behavior of the controlled system and anticipated future changes.

Following an overview of the methodology, this chapter describes the components of the methodology. It then describes an architecture that connects these components to provide activity tracking capabilities. Finally, the chapter compares the GT-CATS methodology and architecture to related intent inferencing research.

Overview of the GT-CATS methodology

The GT-CATS methodology has four elements (figure 22). First, the methodology hypothesizes the next set of activities the operator will perform. It predicts one way of using of using the available control automation in the current operational context; specifically, the methodology predicts which mode(s) the operator is likely to select, and when, to achieve a desired system state. It also predicts how and when the operator will setup, engage, monitor, and adjust the selected mode. The methodology produces hypotheses at multiple levels of abstraction, in terms of high level activities (i.e., mode selections, tasks, and subtasks), as well as individual actions. By predicting when a new mode will be used, the methodology indirectly predicts when a mode transition is imminent, and to which mode.

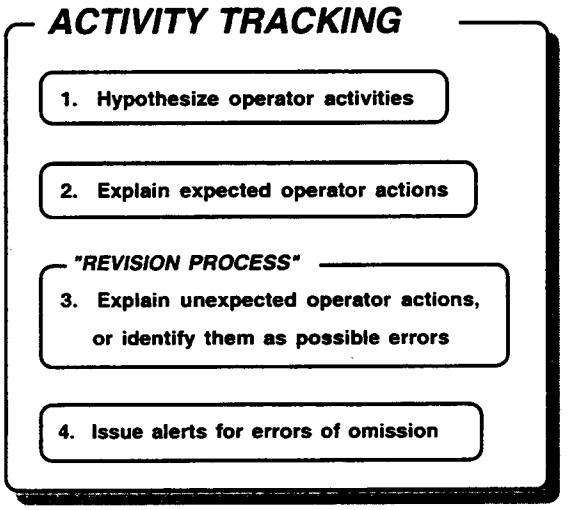


Figure 22. Elements of activity tracking.

The second element of the GT-CATS methodology is to explain operator actions that support its hypotheses. By confirming that an expectation is met by an actual operator action, GT-CATS produces an explanation for the action. GT-CATS' produces explanations at multiple levels of abstraction, in the manner of the initial hypothesis.

Automation that offers the human operator several mode choices for accomplishing a goal makes explaining an operator's choice of modes more difficult. Operators may switch between modes at will, seeking to exploit some perceived advantage of the new mode. The third element of the GT-CATS methodology, called the "revision process," addresses this problem. The name refers to how GT-CATS revises hypotheses about the mode it expects the operator to use, and explains "unexpected" actions as supporting an alternative mode that is also applicable in the current situation. This capability is vital to understanding operator activities in multi-modal supervisory control environments.

To explain unexpected actions, GT-CATS' revision process uses updated information to

assess whether the operator's mode choice is valid; if it is, GT-CATS explains the action as supporting the alternative mode. The second function of the revision process is also extremely important: detecting possible operator errors. If an unexpected action cannot be explained via the revision process, GT-CATS identifies the action as a potential error. The fourth element of the GT-CATS activity tracking methodology is to identify predicted actions that have not been detected or superseded by alternative actions. Expectations for operator actions that have not been met, and have not been superseded by actions related to an alternative mode, suggest a possible error of omission. Thus, the GT-CATS methodology is designed to note the possibility that the operator has forgotten a required action, or is unaware that a mode change is required in a particular situation.

The GT-CATS methodology is predicated on the additional requirement that the predictions and interpretations of operator actions should be produced in real time. A real-time understanding is vital because intelligent tutors and aids must keep up with the operator-automation interaction as it unfolds.

Components of the GT-CATS methodology

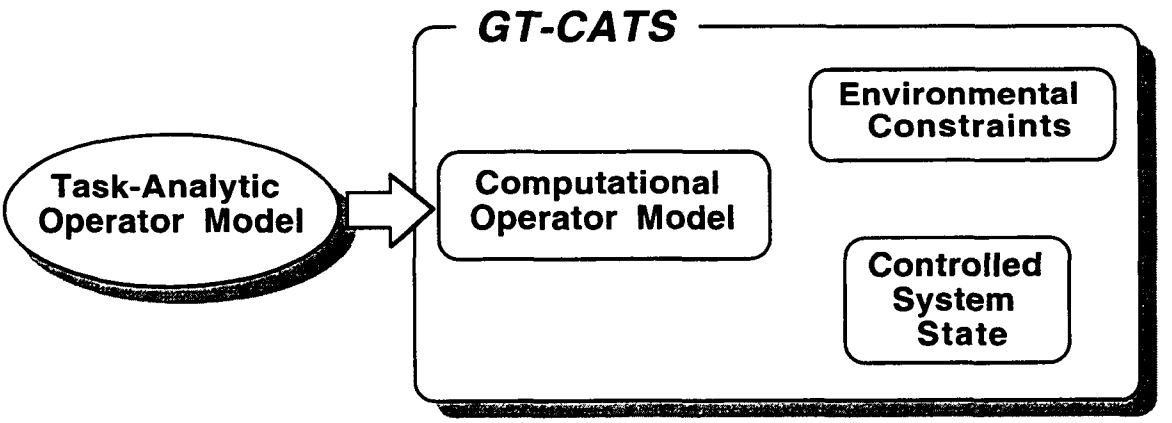


Figure 23. Knowledge representation in the GT-CATS methodology.

The GT-CATS methodology uses four knowledge representations that are linked through processing (figure 23). The first is a static task-analytic model of operator activities. The second is an instantiation of the task-analytic model in a computational form that is dynamically annotated during run time. The methodology uses this instantiation to interpret the current operator actions. During processing, knowledge about the current status of activities is added to the computer instantiation of the model to produce expectations. The computational operator model is also critical for producing explanations, because the methodology explains actions at the levels of abstraction represented in the model. The remaining two knowledge representations provide current knowledge about the constraints imposed by the environment and the state of the controlled system.

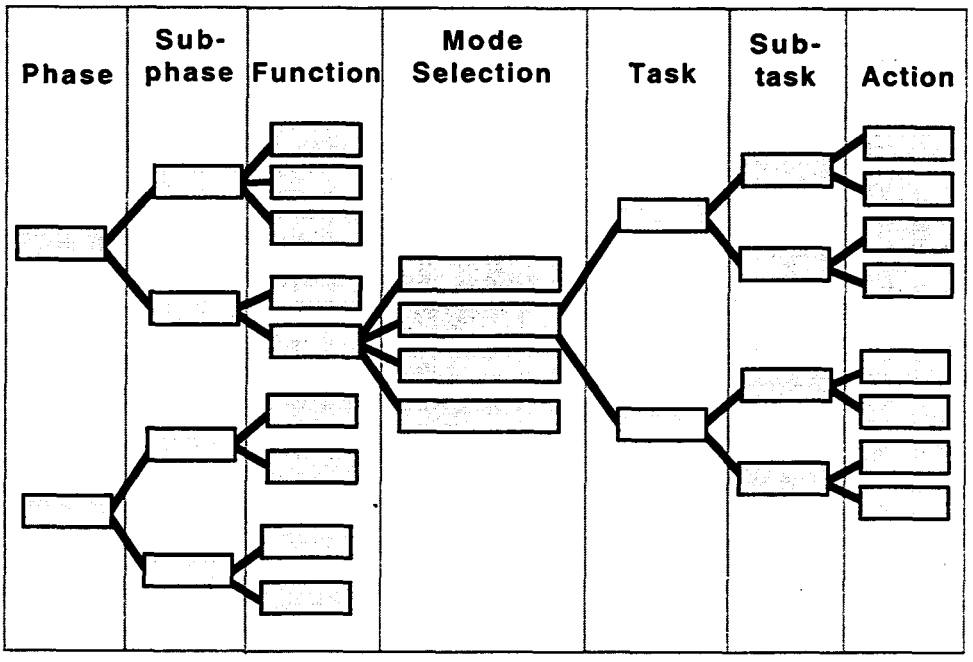


Figure 24. Generic structure of the OFM-ACM.

Representing the operator's task: The OFM-ACM

Knowledge about the operator's task is represented by an explicit, task-analytic model based on the OFM. Called an Operator Function Model for systems with Automatic Control Modes (OFM-ACM), the model specifies how operators use automation modes to achieve desired performance from the controlled system. The OFM-ACM imparts an explicit mode orientation to the OFM. Like the OFM, the OFM-ACM is structured as a heterarchical-hierarchical network of nodes that represent operator activities at relevant levels of abstraction (figure 24).

In the hierarchical dimension, the OFM-ACM decomposes operator functions that must be performed to meet operational goals into the modes that can be used to perform them, and in turn decomposes each mode into the tasks, subtasks, and actions required to use the mode depending on the situation. As with the OFM, such a decomposition is referred to as an "activity tree." The OFM-ACM's structure is also heterarchical like the OFM. The heterarchy is important for representing multiple functions that can be performed concurrently, and because the use of a particular control mode often allows or requires operators to perform tasks or subtasks concurrently.

The OFM-ACM enhances the OFM heterarchy by including an explicit hierarchical decomposition of operator activities for each phase of system control in the manner of Jones et al. (ref. 50) and Thurman and Mitchell (ref. 89). This enables operator control responsibilities to be represented explicitly in systems whose operation is generally thought of as consisting of several mutually exclusive phases, each of which requires operators to undertake a particular set of control functions. This structural feature of the OFM-ACM allows differences in how a mode is used to perform a required function in a given phase to be explicitly represented.

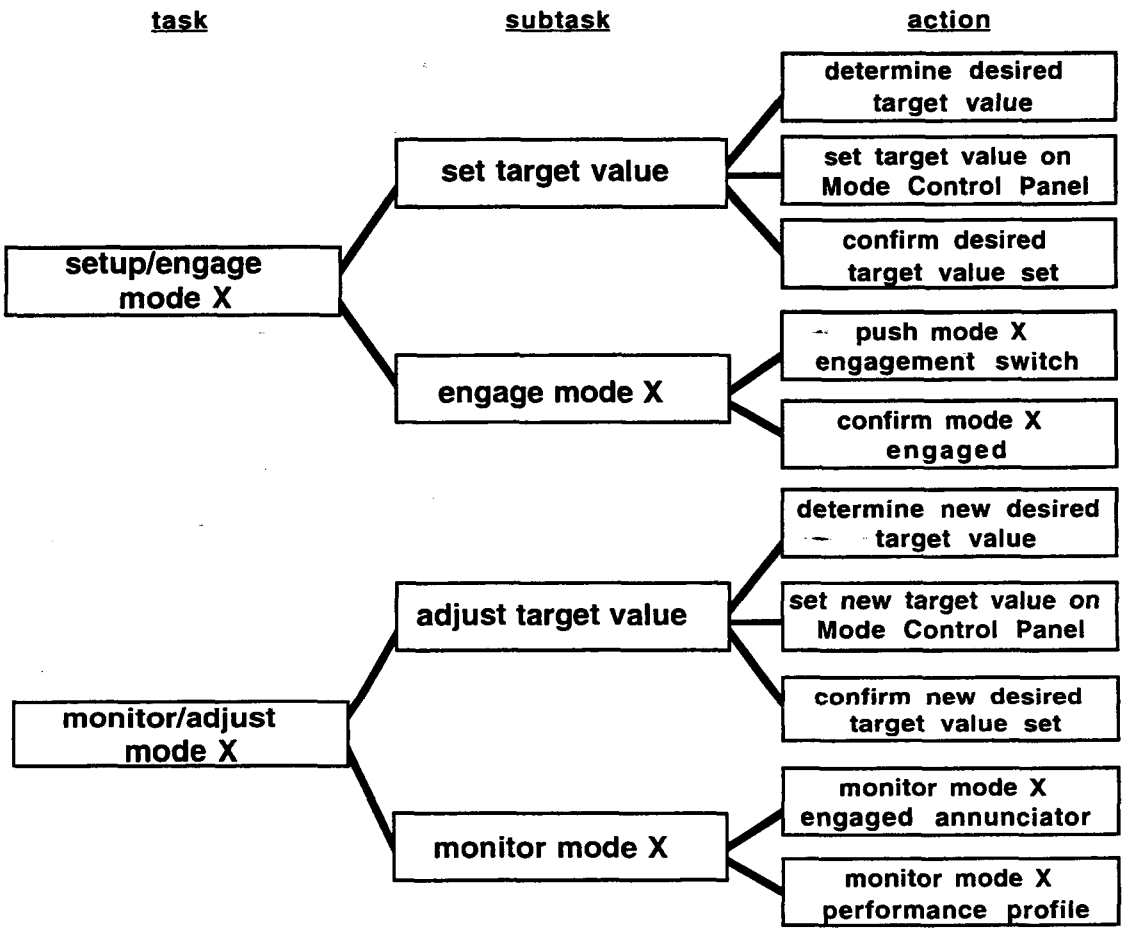


Figure 25. Generic decomposition of mode selection X into "setup/engage" and "monitor/adjust" task subtrees.

The structure of the OFM-ACM provides a theoretical framework for organizing knowledge about the operator's task. A control function is decomposed into the mutually exclusive mode selections available for performing it, each representing a control option available to the operator. Each applicable mode is decomposed into task "subtrees" that represent a task, its subtasks, and the supporting actions required to use the mode. For modes that are engaged manually, one task subtree commonly represents mode setup and engagement activities; a second represents monitoring and adjustment activities. A generic view of the task subtrees used in the OFM-ACM structure is depicted in figure 25.

Another important structural feature of the OFM-ACM is that activities above the mode selection level must be uniquely determinable (figure 26); activities above the mode selection level must be structured such that there is no ambiguity as to when the operator is expected to perform these activities. This is because GT-CATS must first be able to isolate the mode choices that the operator has in a given situation, in order to expect and explain operator mode usage. GT-CATS must be certain about the high-level activities that should be performed in order to determine the set of mode selections applicable to the situation.

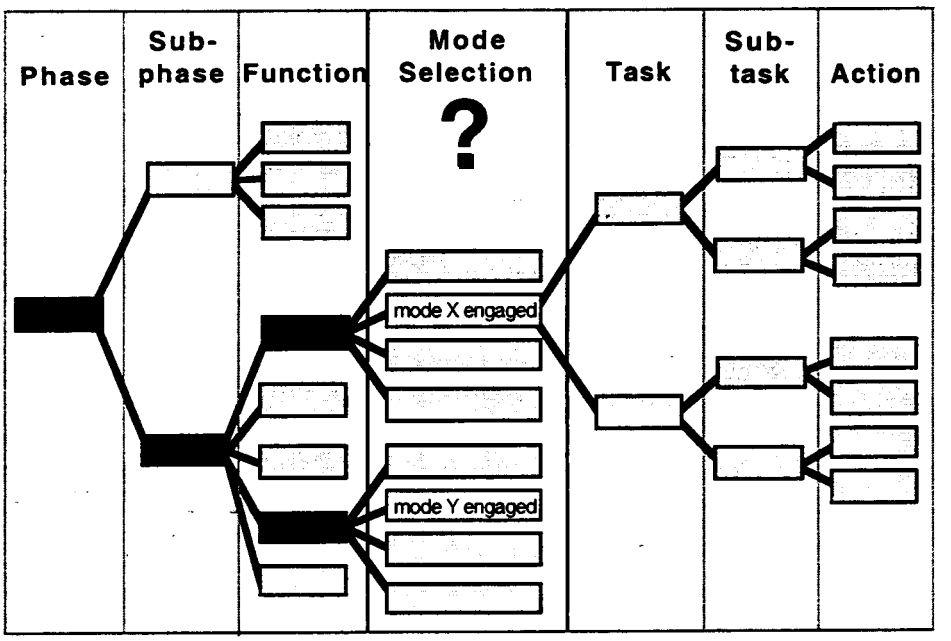
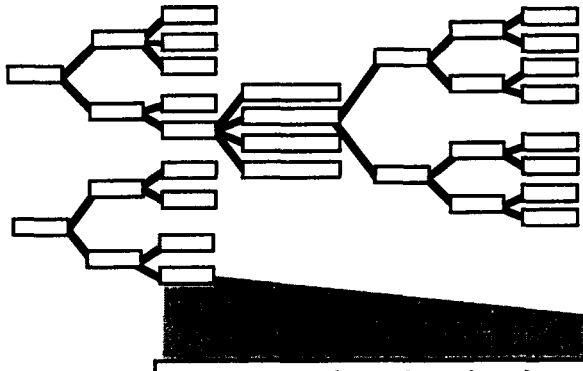


Figure 26. Functions are uniquely determinable; mode selections are uncertain.

Like the OFM, the OFM-ACM is generalizable with respect to the number of levels of abstraction required to adequately represent knowledge about the operator's task. GT-CATS' processing scheme, however, uses the mode selection level as a "pivot-point" for resolving uncertainty. In determining whether an unexpected action can be explained as supporting an alternative valid mode, the revision process refers to occurrences of the action that can support the possible set of modes. From a top-down perspective, the mode selection level is the first level of abstraction at which uncertainty is encountered because several modes may be applicable; from a bottom up perspective, the mode selection level is the first level at which uncertainty can be resolved because each mode selection corresponds directly to a mode that must be engaged in the controlled system if the operator has performed an action that supports a task related to using that mode.

The contents of the individual nodes that comprise the OFM-ACM activity trees are also important to the GT-CATS methodology. Each node encapsulates knowledge about the activity (figure 27). Basic knowledge includes the name of the activity, an identification number, and the level of abstraction at which the activity resides. A reason for the activity that reflects its inclusion in the OFM-ACM at the present location is also included for reference. This reason provides additional knowledge about the activity that might be useful for an intelligent tutoring or aiding system. Knowledge about the type of activity is also contained in a node (see figure 27). This information distinguishes manual, perceptual, cognitive, or verbal operator actions. Although the actions that the GT-CATS methodology can track computationally are limited currently by affordable technology to detectable manual actions, the methodology is also applicable to tracking perceptual, cognitive, or verbal actions. The task subtrees shown earlier are structured to represent activities of all of these types; the GT-CATS methodology includes them because such activities are important for the operator to perform in monitoring the behavior of the automation and controlled system. Although the methodology cannot explain perceptual, cognitive, or verbal actions because they are undetectable, by including them, the methodology can expect when such activities should be performed.



name: perform function A
identifier: 2021
node-type: function
up-links: 1007 (identifier of subphase that includes this function)
down-links: 3022, 3023, 3024 (identifiers of mode selections into which this function is decomposed)
reason: function A should be performed to bring state variable X within limits
conditions: state variable X outside limits
agent: (for low level tasks/subtasks/actions allocated to a specific crew member)
activity-type: (for actions; manual, perceptual, cognitive, or verbal)
automation mode: (for mode selections; indicates automation mode corresponding to the mode selection)

Figure 27. Knowledge contained in an OFM-ACM activity node.

Nodes in the OFM-ACM also represent knowledge about the agent responsible for performing the activity (see figure 27). The GT-CATS methodology is oriented toward systems where the "operator" may actually be a team of operators; this is the "crew" in GT-CATS. In such cases, knowledge about task allocation among the crew members is crucial for understanding human-automation interaction. With this knowledge, an action appropriate for a given situation may be identified as a departure from operational guidelines if performed by the wrong member of the crew.

Finally, each node has conditions that specify the operational context in which the operator is expected to perform the activity (see figure 27). The conditions in a node consist of a set of "context specifiers." Context specifiers play a critical role in the GT-CATS methodology. They link dynamic context knowledge from the representations of the environment and controlled system to knowledge of the operator's task represented in the computational instantiation of the OFM-ACM (figure 28).

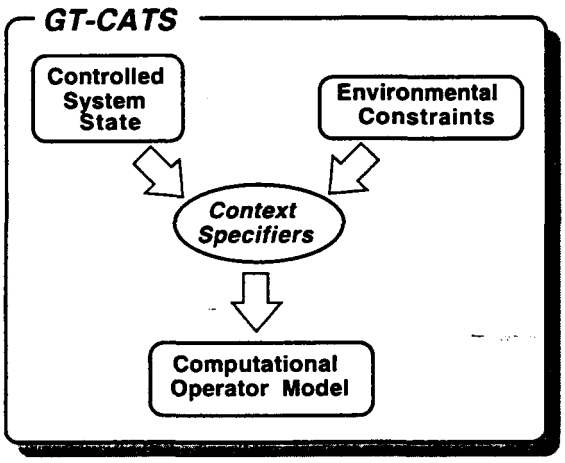


Figure 28. The central role of context specifiers.

Context specifiers are activated based on the representations of the controlled system and environmental constraints. An individual context specifier summarizes the relationship between a particular aspect of the state of the controlled system and an environmental constraint. A generic example of this is shown in the conditions in the OFM-ACM activity node in figure 27: the state variable X, when related to constraints on its value imposed by the environment, does not meet the constraints (i.e., the value of X is "outside limits"). The value of using context specifiers to condition when an operator is expected to perform some activity is shown by this same example. Regardless of the actual value of the state variable X (which is dynamic), and the particular environmental constraint that binds it (which is also dynamic), the resulting context specifier takes a static form which can be used as a condition for expecting the activity as modeled in the OFM-ACM.

The conditions contained in activity nodes in the OFM-ACM may consist of multiple context specifiers. The group of context specifiers used as conditions in a given activity node together reflect the relationship between multiple state variables and environmental constraints. Further, the GT-CATS methodology allows that the conditions in a node may consist of two groups of context specifiers. If either one group or the other is present, the operator is expected to perform the represented activity.

The conditions in the OFM-ACM are specialized for activities at each level of abstraction (figure 29). Generally, high-level activities have conditions comprised of context specifiers that relate general environmental knowledge to general state knowledge to express the current relationship between the goals of the operator and the state of the controlled system. Conditions at the mode selection level are comprised of context specifiers that reflect the state of the control automation vis a vis the preferred control mode (figure 29). Nodes that represent mode selections in the OFM-ACM also contain an additional piece of knowledge: the corresponding automation mode that should be engaged if the operator has selected the particular mode. This knowledge is specific to the mode selection level of the OFM-ACM, and is vital to the revision process.

Below the mode selection level, conditions are constructed from context specifiers that relate specific knowledge about the state of the automation to specific knowledge of environmental constraints. Context specifiers of this sort are used to identify tasks and actions relevant to the preferred mode selection that produces the required response in the controlled system (figure 29).

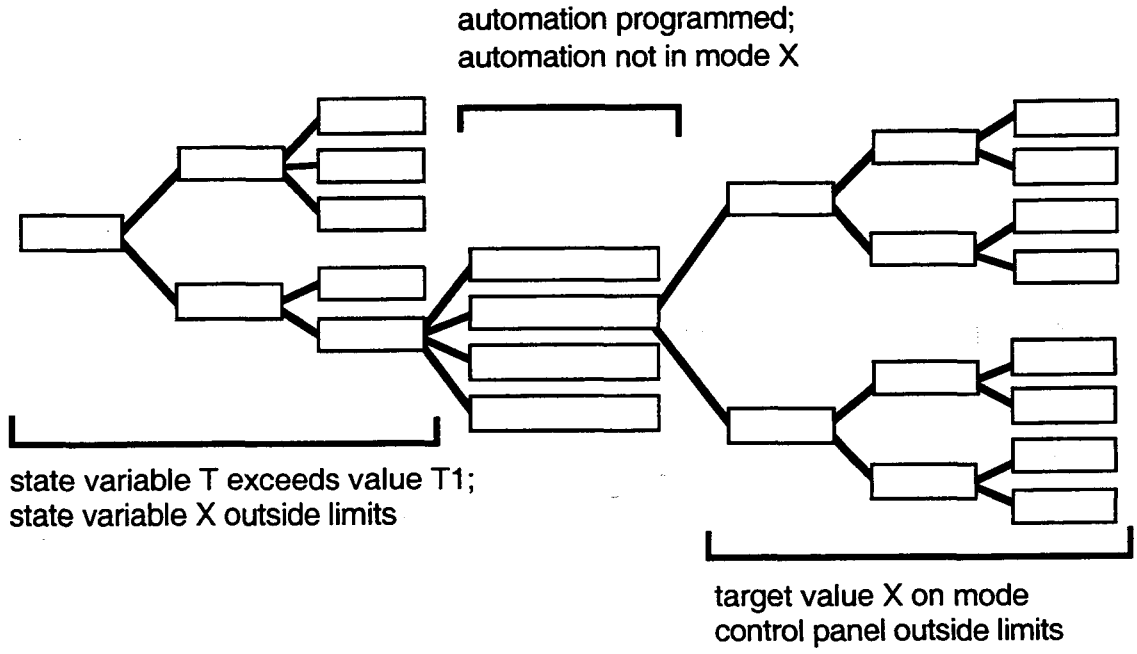


Figure 29. Generic examples of context specifiers with different characteristics as conditions in the OFM-ACM at different levels of abstraction.

The context specifiers that serve as conditions on activities in the OFM-ACM are also important for specifying procedural or concurrent activities. Concurrent activities have conditions that include the same, or similar, context specifiers. Activities that comprise steps in a procedure have as conditions context specifiers that reflect the effects of earlier steps in the procedure. For example, the "setup/engage" and "monitor/adjust" tasks shown in figure 25 form a procedure; the "setup/engage" task is followed by the "monitor/adjust" task. In this case, the conditions under which the "monitor/adjust" task is relevant reflect the fact that the "setup/engage" task has been performed (i.e., the automation is now in mode X).

To summarize, context specifiers form the conditions in the nodes of the OFM-ACM that indicate when an activity is expected. Knowledge about the reason for the activity contained in each node, noted above, essentially states why the activity is preferred under the conditions designated by context specifiers in the node. The fidelity of the context specifiers that comprise each node's conditions affects the methodology's activity tracking capabilities. The context specifers that serve as conditions must specify the operational context in which the activity is appropriate to afford unambiguous expectations.

Representing the state of the controlled system: The state space

The GT-CATS state space encapsulates all relevant knowledge about the state of the controlled system. This includes the state of the controlled system, as well as the state of the control automation, as shown in figure 30. The state space is updated dynamically to reflect changes in the state of the controlled system. The fidelity of the state space is defined by the granularity of the state knowledge (i.e., how detailed the representation is), along with how frequently it is updated. The state space must be updated frequently enough to accurately reflect current state information, in order to produce context specifiers that accurately portray the current operational context of the system.

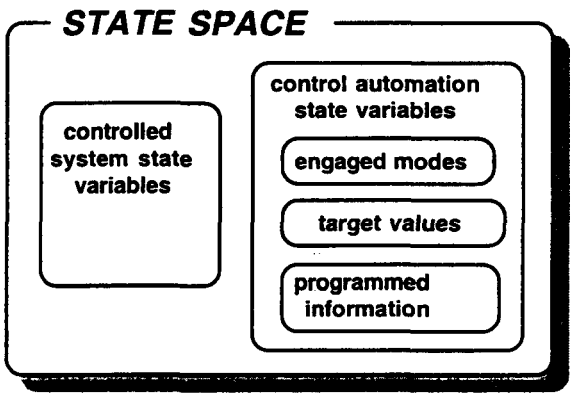


Figure 30. GT-CATS' state space knowledge representation.

Representing environmental constraints: The Limiting Operating Envelope

"For realistically complex problems there is often no one best method; rather, there is an envelope containing multiple paths each of which can lead to a satisfactory outcome (ref. 11, p. 16)." In the GT-CATS methodology, the structure of the OFM-ACM represents knowledge of these paths. An analogous concept is applicable to dynamic constraints placed on the operation of the controlled system by the environment; these constraints define an envelope in which the system must operate. The GT-CATS methodology terms this representation the "limiting operating envelope." The limiting operating envelope (LOE) is constructed in a manner similar to that of the space of feasible solutions in the field of computational optimization. Each environmental constraint is imposed on the space of possible system operations, and the limiting constraints are identified. This set of limiting constraints defines the space of feasible operations of the controlled system.

The GT-CATS LOE summarizes the constraints placed on a controlled system derived from safety concerns, regulatory agencies, the operating organization, and the capabilities of the controlled system itself. Assuming a well-trained and motivated operator, the constraints on system operation represented by the LOE define operator objectives. The LOE is dynamic, because the GT-CATS methodology is concerned with systems in which the state of the system and the goals of the operator are dynamic. When the constraints change, the LOE representation must change to reflect the new constraints.

The GT-CATS LOE is therefore designed with two distinct elements. The first element represents the operator's goals and environmental constraints as far into the future as they are known, expressing them as a series of "limit states" to be attained. A limit state is simply a collection of state values that reflect the goals to be achieved. As the "active limit state" is attained, the LOE's binding constraints become those reflected in the next limit state, and so on, as system operation progresses (see figure 31).

The second element of the LOE represents any temporary modifications to the active limit state; this knowledge, when applicable, effectively overrides portions of the active limit state such that a short-term limit state takes precedence in constraining system operation. A short-term limit state is expressed as a set of state values in the LOE. Only some of the state values in the short-term limit state may be important in representing the short-term goal. The active limit state constrains all aspects of operation except for those constrained by valued parts of the short-term limit state representation (i.e., the short-term limit state can override all or some of the currently active long-term limit state. The LOE may contain redundancies where the short-term limit state specifies constraints on operation that are also specified by the active limit state. Figure 31 illustrates this principle. In figure 31 the active limit state calls for the value of state variable V to be V2, while the short-term limit state specifies that the value of V should be V*. Thus, the value V2 is overridden by the value V* specified in the short-term limit-state, and all other values in the active limit state still reflect current goals.

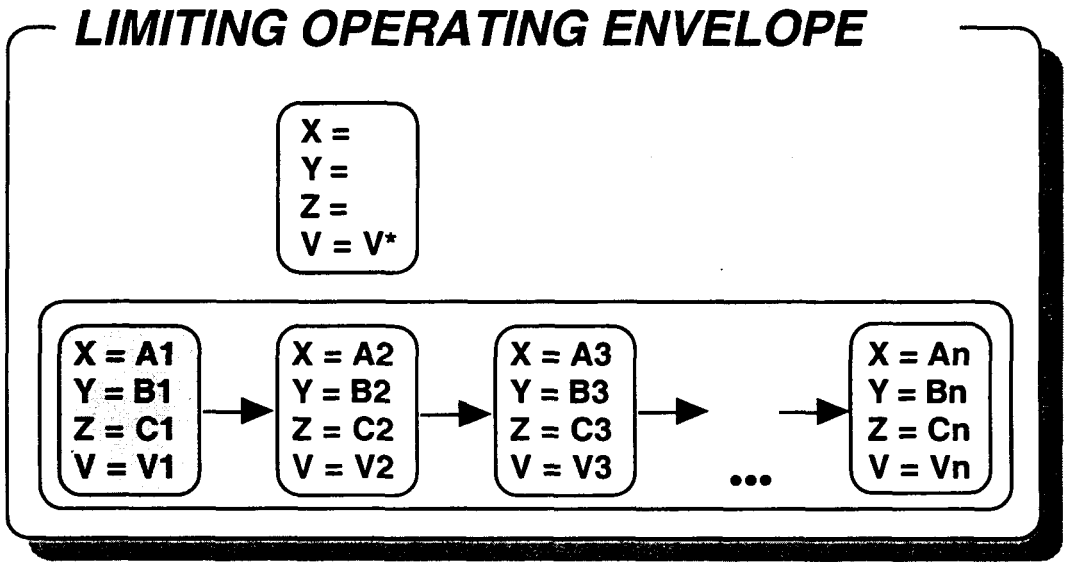


Figure 31. A generic LOE. The first limit state has been attained; the second limit state represents is the active limit state. The value V* in the short-term limit state representation overrides the value V2 in the active limit state.

The LOE representation is used to supply knowledge critical for determining the current operating context, and is therefore subject to requirements that affect its utility for generating context specifiers. Because the context specifiers are derived from the LOE and the state space, the knowledge in the LOE is specified at the same fidelity and at the same level of abstraction as the knowledge in the state space. Thus, state space knowledge can be compared with knowledge in the LOE to activate context specifiers.

The Dynamically Updated OFM-ACM

In the GT-CATS methodology, a computational operator model is derived directly from the OFM-ACM, as shown above in figure 23. The Dynamically Updated OFM-ACM (DUO) is a computational instantiation of the OFM-ACM that serves the dual purposes of representing the knowledge contained in the OFM-ACM and representing the current operator interaction with the control automation. DUO contains all of the knowledge specified by the OFM-ACM, and dynamically annotates it with knowledge to support real-time activity tracking.

When the OFM-ACM is instantiated in DUO, nodes in the OFM-ACM become computational objects, with slots to hold the descriptive knowledge about the activity represented by the node, as well as the conditions knowledge. In addition, a node in DUO contains slots to hold knowledge about the status of the activity in the current operational setting, and the history of the activity's status (figure 32). The status of the activity node reflects its relevance to the activity tracking process at the current time. The history of the activity node is a time-stamped record of the status of the node over the course of system operation.

The instantiation of the OFM-ACM in DUO also requires that activities at the action level that can support multiple tasks must be represented uniquely for each task. The GT-CATS methodology seeks to disambiguate actions with multiple purposes when producing expectations and explanations. When generating expectations, the methodology seeks to determine the precise activities at each level of abstraction that are preferred in current operating context. An action that supports several tasks or modes is represented as a unique instance of the action. This enables any

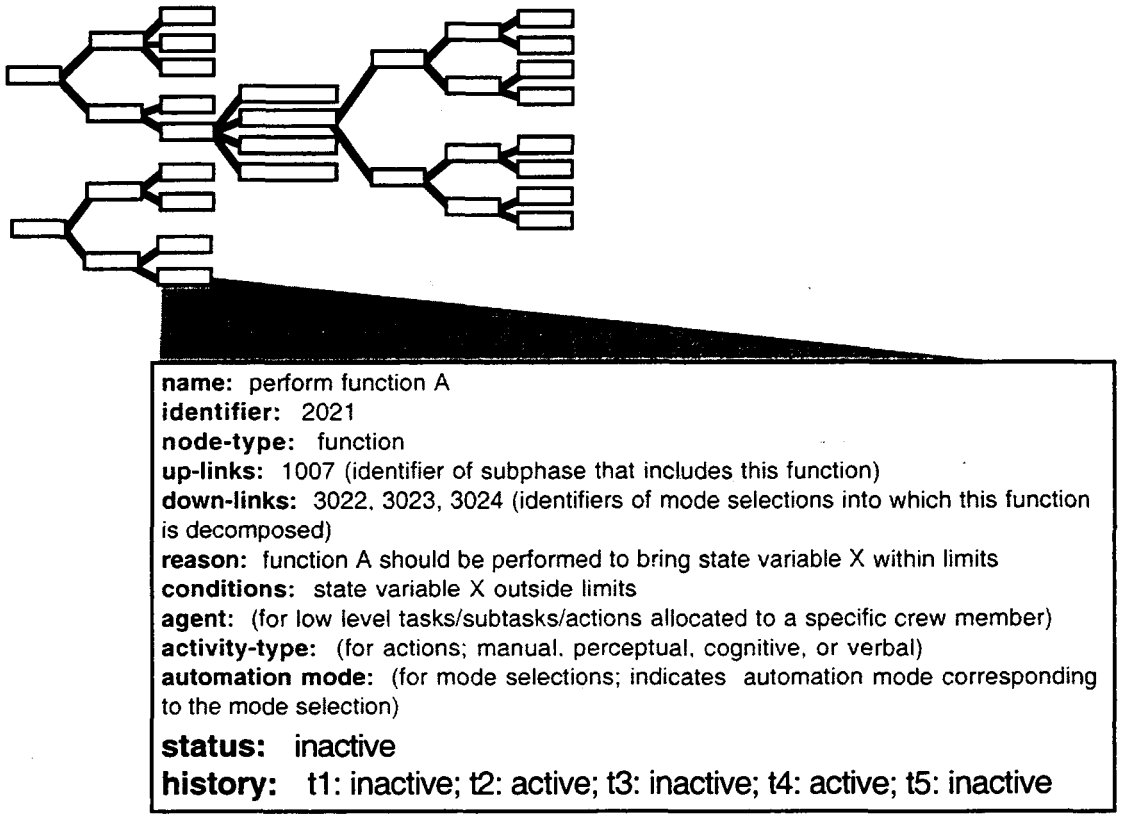


Figure 32. Knowledge about the status and history of the activity is added when the OFM-ACM is instantiated in DUO.

differences in the operational context in which the action should be expected to be represently explicitly, differentiating the action from other instances of the same physical action. Further, different agents may be responsible for the same action depending on the operational context and the high level activities it supports; here the responsible agent differentiates one instance of the action from another.

When explaining a detected operator action that was expected, GT-CATS links the detected action with an action in DUO that has as its parent activities a task and mode selection that explain the action (figure 33). Similarly, when determining whether an unexpected action can be explained, or might be an error, the methodology uses the revision process to determine if another instance of the action in DUO supports a task and mode that can explain the action. Thus, representing actions explicitly for each task in the OFM-ACM enables DUO to disambiguate actions that can support multiple tasks.

The GT-CATS methodology uses a processing scheme that updates DUO with each update to the state space and LOE. On each processing cycle, the state space and LOE are used to activate a set of context specifiers. The processing cycle assigns the status "active" to nodes in DUO whose conditions are a proper subset of the current set of active context specifiers; the processing cycle assigns the status "inactive" to nodes not meeting this criteria. If the status of a node changes, GT-CATS updates the node's history list. At the end of the processing cycle, active nodes represent activities that are expected at that time; inactive nodes are not currently expected. Thus, the processing scheme uses DUO to maintain a dynamic representation of expectations about operator activities.

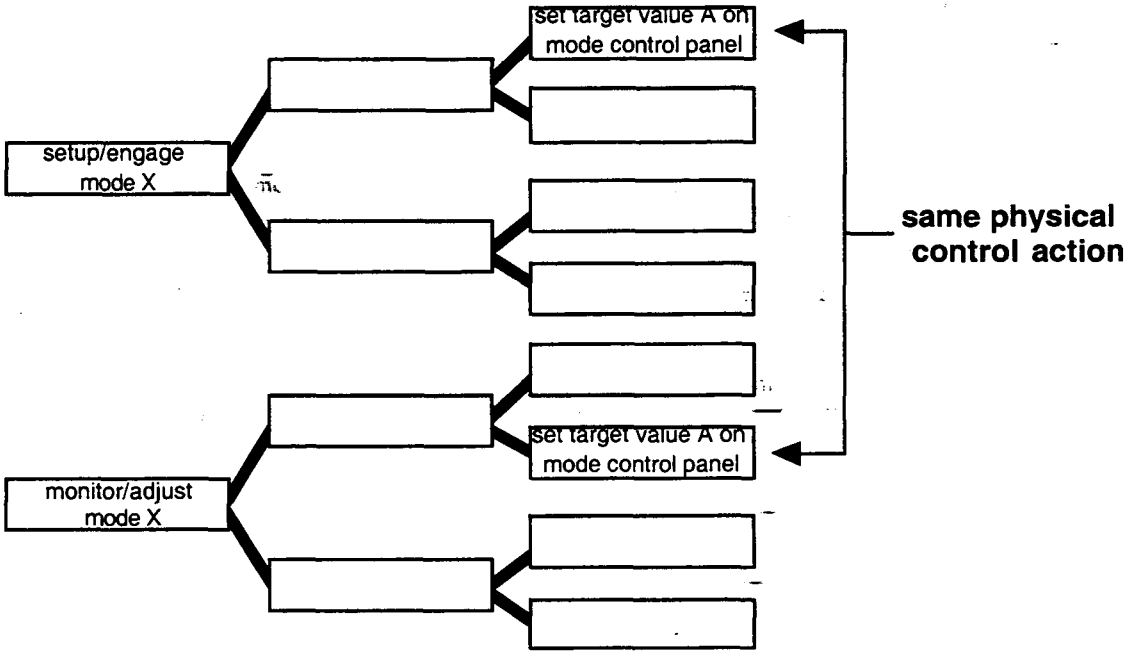


Figure 33. By representing actions uniquely for each task they support, as in this generic example, the OFM-ACM disambiguates equivalent control actions in expecting and explaining such actions.

DUO is updated with a top-down, breadth-first search procedure. The procedure starts with the nodes at highest (phase) level in DUO, and matches their conditions against the currently active set of context specifiers to determine the active phase. Inactive phase nodes, and all their subnodes, are inactive and not considered further. This process is then repeated for the subphases of the active phase. Recall that at the phase and subphase levels, the OFM-ACM nodes are mutually exclusive. Again the active one is found and the others are removed from consideration. At lower levels in DUO, the search of subnodes of an active node can identify multiple active subnodes, representing operator activities that are expected to be performed concurrently. The general processing scheme applies to all levels of abstraction in DUO, with the exception of the mode selection level. As the "pivot-point" in the OFM-ACM representation of the operator's task, the active mode selection is determined by taking into account the related mode knowledge encapsulated only in nodes at the mode selection level. Three cases can arise in determining the active mode selection (see figure 34):

Case 1: The mode selection is active because its conditions are a proper subset of the currently active set of context specifiers, and the state space reflects that the corresponding mode is engaged in the control automation.

Case 2: The conditions of the mode selection node in DUO match a subset of the currently active set of context specifiers, but the corresponding mode is not engaged. In this case, the mode selection is assigned the status"active," because the mode selection is expected in the current situation according to the conditions specified in the OFM-ACM. In this way the methodology derives an expectation that the mode will be used.

Case 3: If the conditions in the mode selection node do not match a subset of the current set of context specifiers, but the automation mode that corresponds to the mode selection in DUO is engaged, a new status designator, "obsolete," is assigned to the mode selection. The "obsolete" status desig-

nator means that, although the mode is engaged, conditions indicate that another mode is needed, and thus expected, in the situation. The obsolete node is processed like an active node, in that active tasks below it are designated active or inactive according to their conditions. In this situation, the activities designated active will be those involved with monitoring the operation of the mode as long as it remains engaged before the transition to the expected mode.

The conditions on alternate mode selections are structured so that if the higher level activity that any of the available modes supports is active, the conditions for one of these alternate mode selections must match the current set of context specifiers. Therefore, in this case, the mode selection that corresponds to the engaged mode should be monitored, but is (about to become) obsolete; there exists another mode selection to which the operator is expected to transition, because its conditions match the current set of context specifiers according to case 2.

In examining the interplay between cases 2 and 3, it is important to note the reason why a mode selection in case 2 may not have an alternative competing mode with status "obsolete." This is because the active function may have changed. Only when the same function remains active can one mode selection become obsolete and an alternative mode selection be active as in case 3.

The processing scheme in which active and inactive nodes in DUO are identified by matching the current set of context specifiers to the conditions knowledge in each node (and related mode knowledge at the mode selection level) produces hypotheses about the currently relevant set of operator activities. This process works top-down through the levels of abstraction instantiated in DUO. The search is pared at each level by limiting the next level of search to the nodes into which active (or obsolete) nodes are decomposed. Processes that explain operator actions and detect errors, however, require knowledge about actual actions. The revision process is the centerpiece of these processes. The revision process and related processes that work bottom-up using knowledge from DUO along with knowledge about actual operator actions are the responsibility of the GT-CATS action manager.

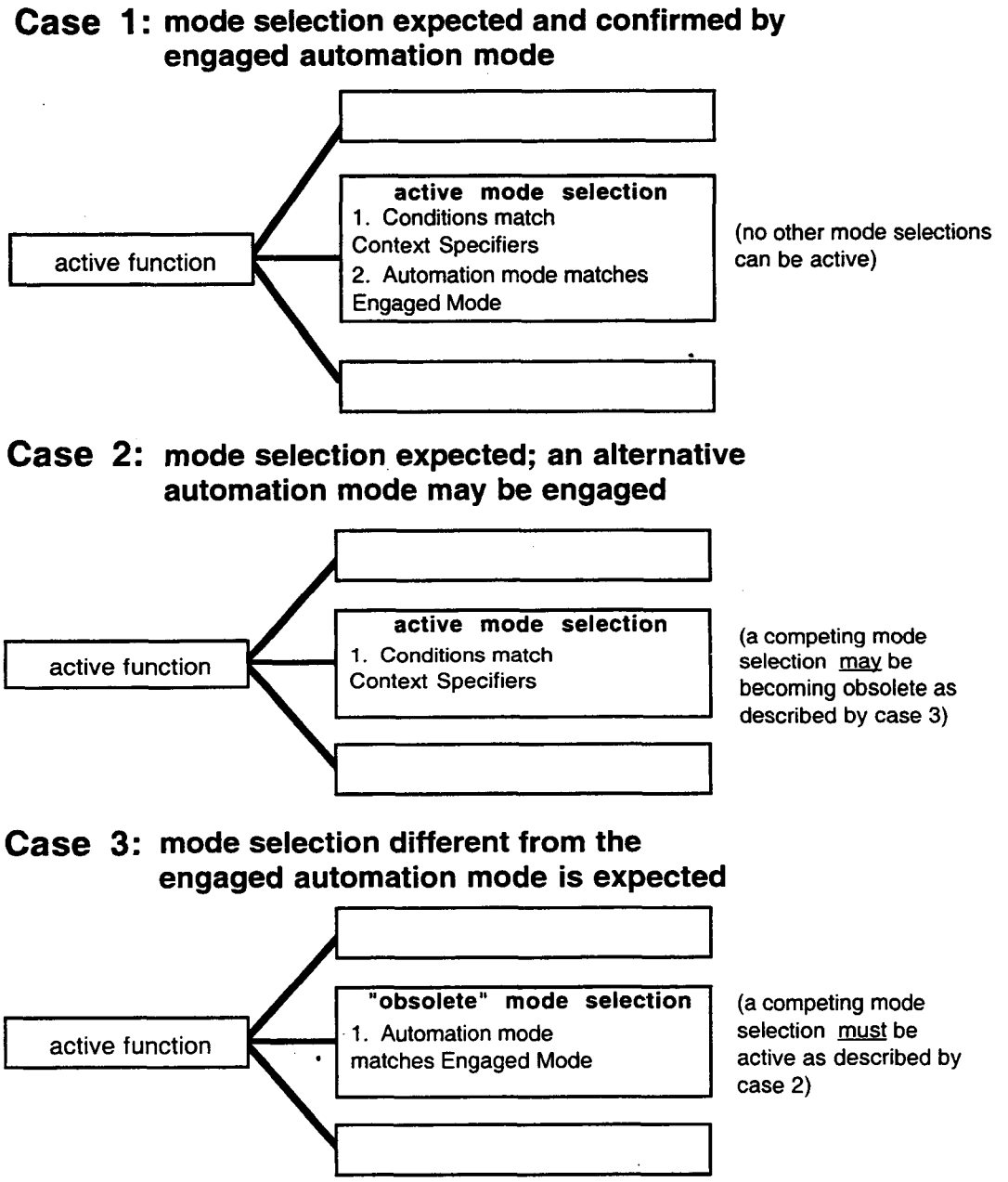


Figure 34. Mode selections may be "active" or "obsolete."

The GT-CATS action manager

The GT-CATS methodology uses DUO to predict operator activities; however, DUO is supplemented by the action manager, a mechanism for managing the remaining activity tracking functions. All of these functions involve examining detected operator actions in light of the knowledge in DUO. First, the GT-CATS action manager attempts to confirm that an actual operator action meets expectations, and explains it accordingly. Failing that, the action manager initiates the revision process to determine whether an unexpected operator actions suggests a valid alternative mode, or whether it is possibly in error.

The action manager's first function is to generate explanations for operator actions that match expectations represented by active

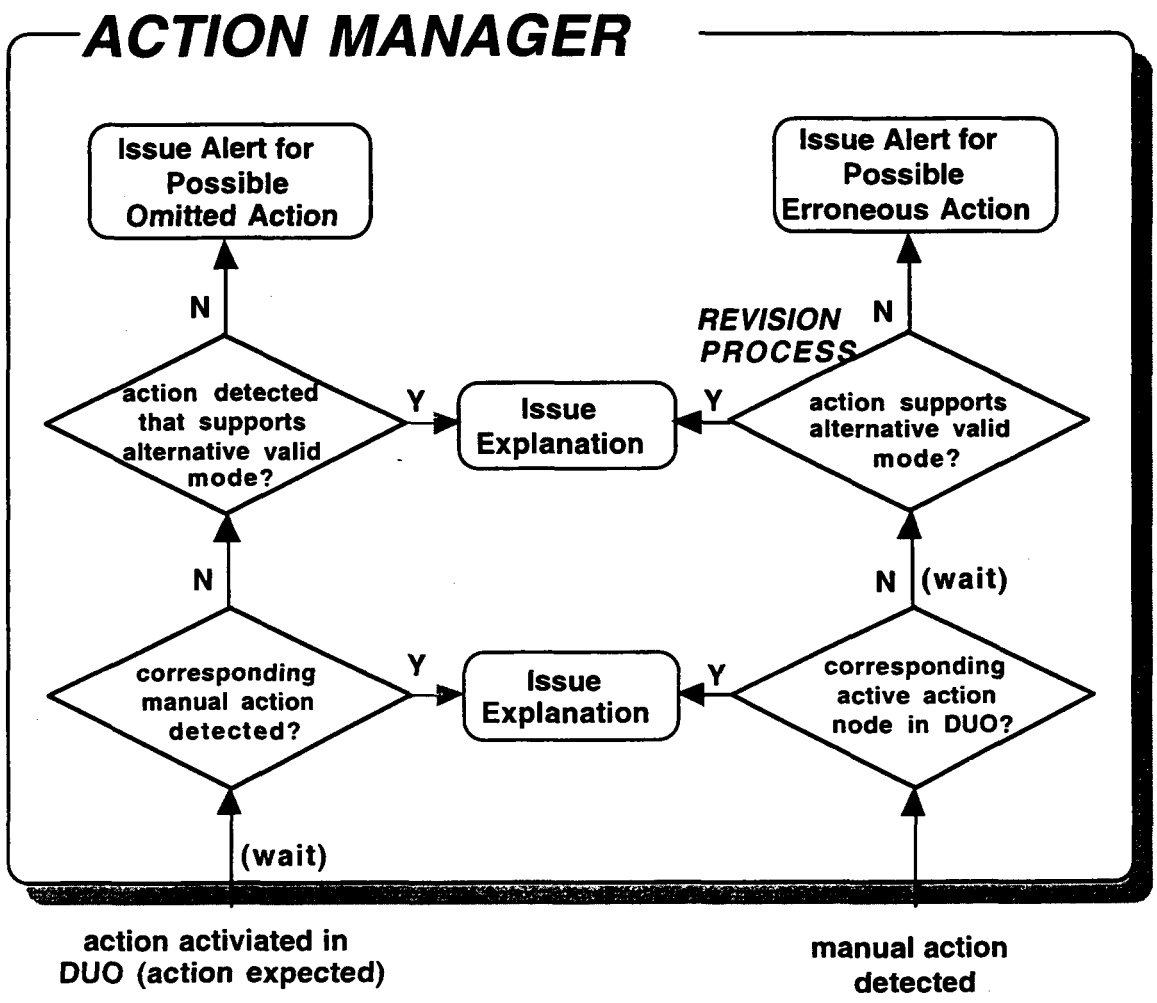


Figure 35. GT-CATS action manager.

nodes in DUO (see figure 35). This entails determining whether a detected operator action has a corresponding active action node in DUO. If so, the action manager assigns the action a status of "explained."

If the action manager cannot locate a corresponding active action in DUO, the observed action is defined to be unexpected. The action manager allows a period of time to elapse, then applies the revision process to the unexpected action (see figure 35). The revision process either explains the action, or identifies it as a possible error. To formulate a new explanation for the action, the action manager uses DUO in a bottom-up manner. It first identifies all action nodes that are instances of the unexpected action that support active functions. It then checks the mode selection that the action supports to see if it is now active, or, according to its history list, has been active in the time since the action was detected. If so, the action manager removes any other instances of candidate action nodes from consideration and explains the unexpected action.

Explaining the action involves first removing hypotheses about actions expected to support an alternative mode selection in DUO. GT-CATS identifies the alternative mode selections using the structure of the DUO, and checks their status. If any are active, the active manager makes them and all of their

subnodes inactive. It then sets the status of the node that corresponds to the unexpected action to "revised-explained." Finally, the action manager explains that the unexpected action supports a task that supports an alternative valid mode selection. Thus, to perform the revision process, the action manager uses DUO—by now updated to reflect active context specifiers—along with status and history knowledge encapsulated in DUO's nodes, and the structure of the OFM-ACM from which DUO derives.

If the revision process cannot explain an unexpected operator action, the action manager determines that the action was not understood. Such an action is possibly an operator error. The revision process cannot explain actions represented by instances of action nodes in DUO that support mode selections that have not been active between the time the action was detected and the time the revision process was applied. In this case, the action manager issues an alert signaling that no alternative instance of an action node in DUO explains the action, as shown in figure 35.

The GT-CATS methodology uses the expectations represented by the nodes with active status in DUO to flag actions that the operator may have omitted. When an action node becomes active, the action manager allows a period of time to elapse. If the action is not detected within this time period, and no other actions are detected for which the revision process can determine that the operator chose an alternative mode, the action manager issues a warning that the operator may have omitted the hypothesized action (see figure 35).

Figures 36 through 39 show a generic example of the revision process. Figure 36 shows a generic portion of DUO that represents two modes (and their supporting tasks, subtasks, and actions) that can both be used to perform a particular function. Shaded activities have active status, and are therefore expected. Among these activities is the action "push mode 1 switch." In figure 37, the operator has performed the action "push mode 2 switch" instead. This action is unexpected, because an action node representing this action is not active in DUO. The GT-CATS action manager therefore schedules an event to perform the revision process on the action. In figure 38, the fact that the operator performed "push mode 2 switch" has since been reflected in an updated version of DUO; when mode 2 actually engages in the controlled system, its corresponding mode selection becomes active in DUO, along with the monitoring activities required to use mode 2. When the action manager executes the revision process (figure 39), the "push mode 2 switch" action is explained to support the use of mode 2 in the situation, which is valid according to the structure of the OFM-ACM embodied by DUO.

Summary of the GT-CATS methodology

In the GT-CATS methodology, knowledge about the operator's task represented in the OFM-ACM is instantiated in DUO. State space and LOE knowledge is used to activate a set of current context specifiers, which are in turn used to activate nodes in DUO to produce expectations of activities that are preferable in the current context. When the operator performs an action, the action manager either explains it according to an existing expectation, explains it via the revision process, or identifies it as a possible error. If no action is detected that can be explained, an error of omission is indicated.

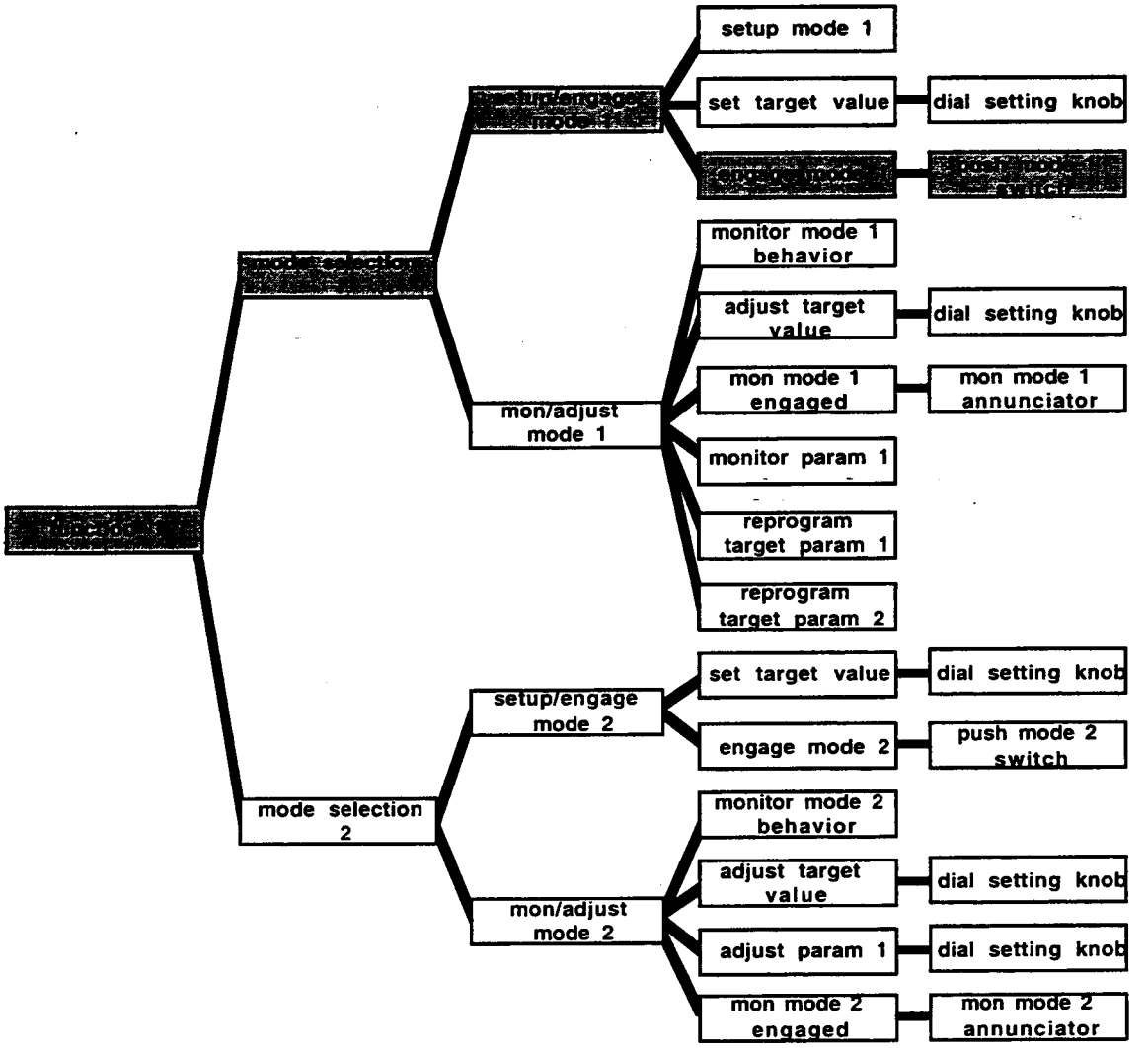


Figure 36. Mode selection 1 is expected to be engaged by pressing the mode 1 engagement switch.

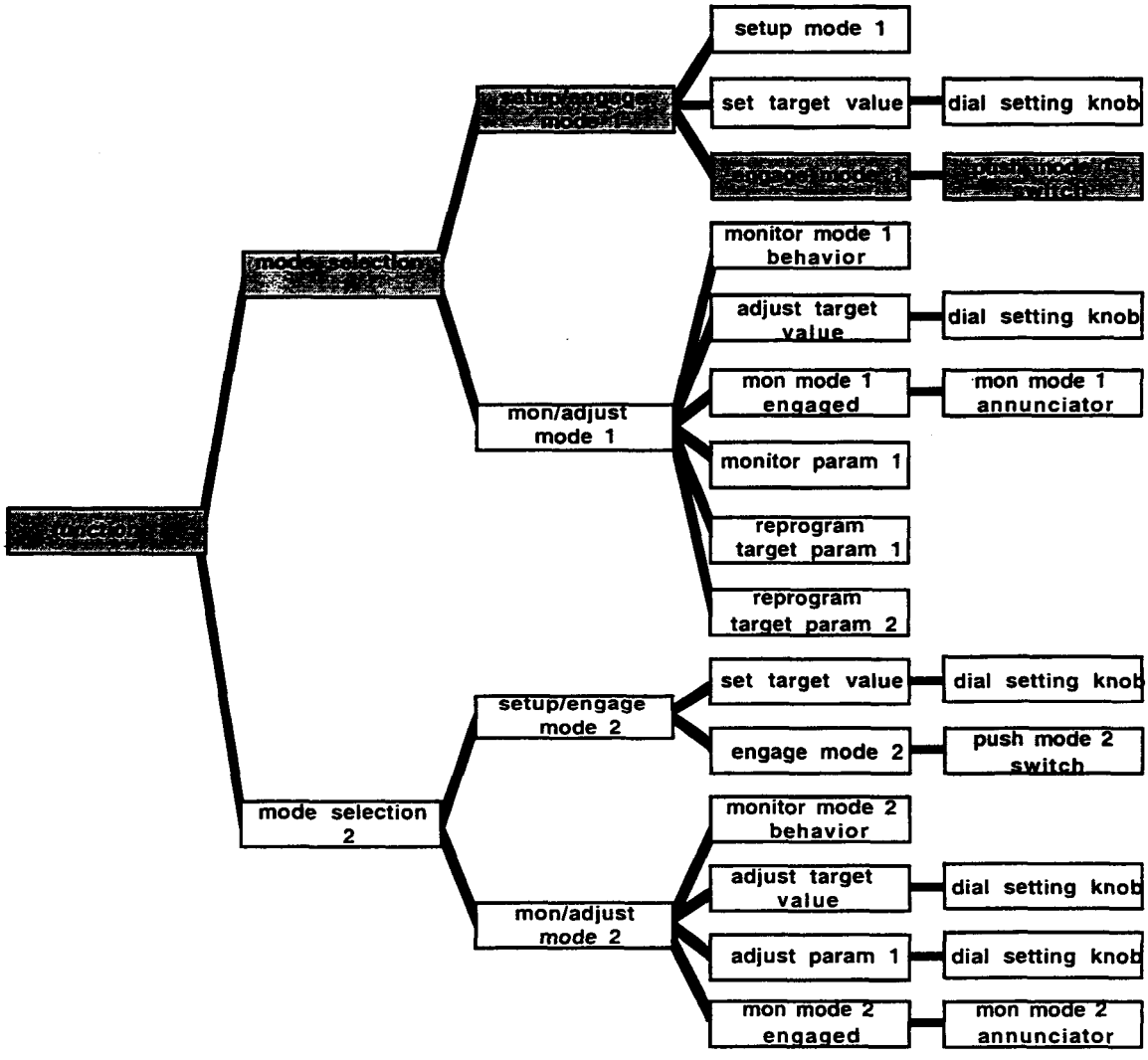


Figure 37. An unexpected action is detected (push mode 2 switch).

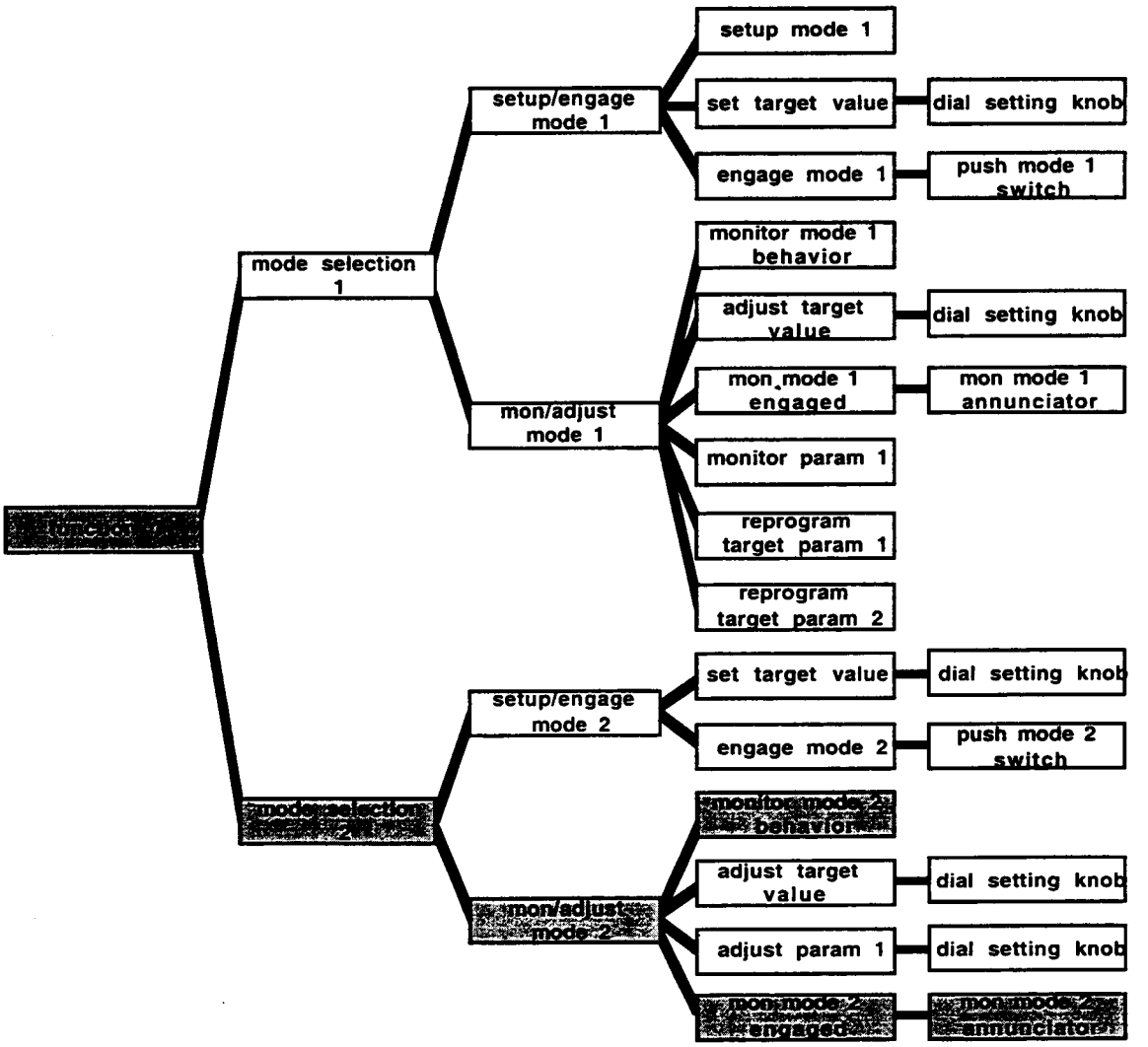


Figure 38. DUO is updated to reflect that mode 2 is engaged in the controlled system.

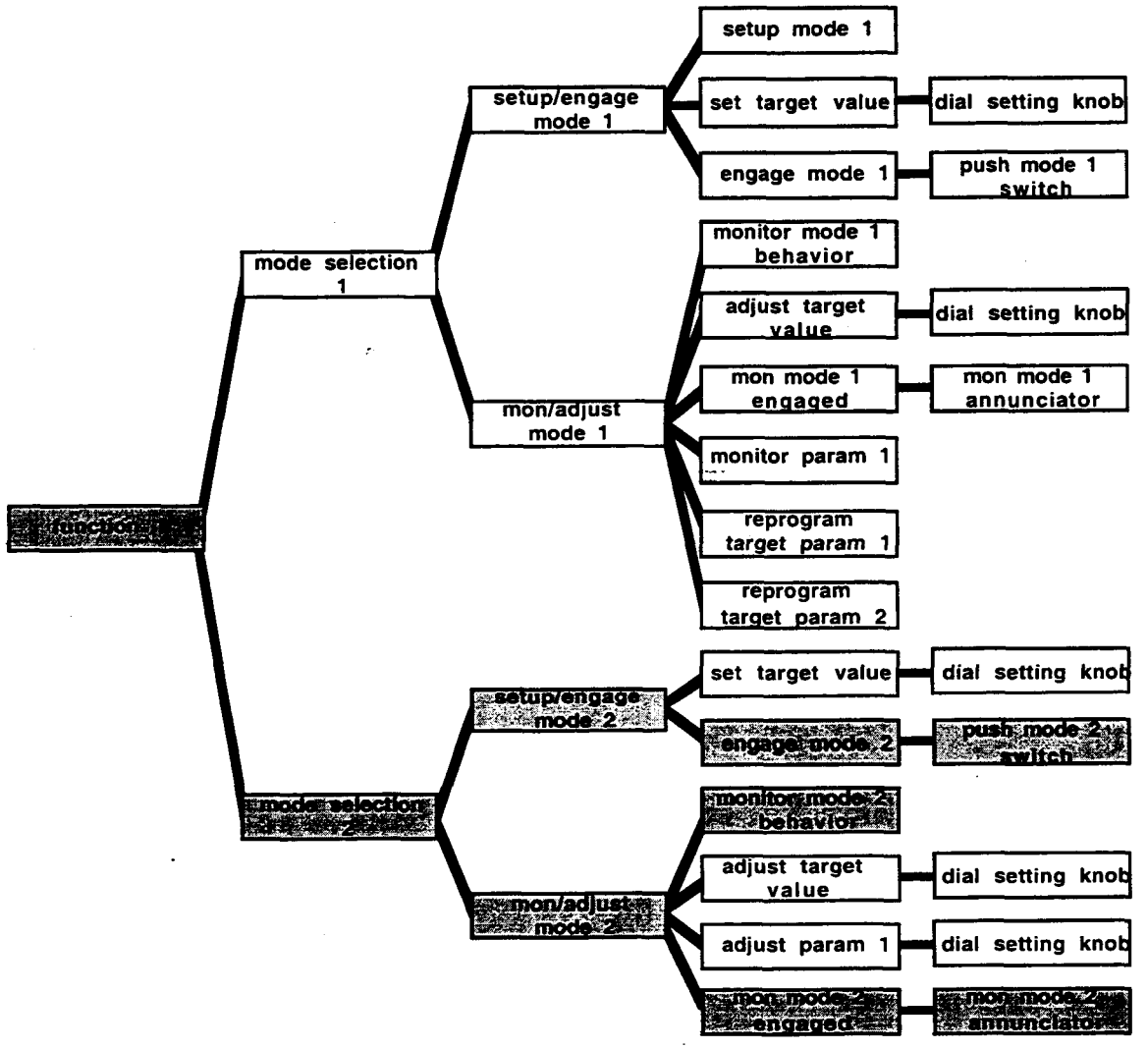


Figure 39. The revision process explains the action as supporting the use of mode 2.

The GT-CATS architecture

A computer architecture has been developed for implementing the GT-CATS methodology. The architecture provides a computational framework for tracking the activities of operators using automation to control a complex dynamic system in real-time. The GT-CATS architecture has structures for representing the knowledge required by the GT-CATS methodology and methods to control the processing of these knowledge representations.

Figure 40 shows a functional view of the GT-CATS architecture. Components derived from the GT-CATS methodology (i.e., the OFM-ACM, DUO, the LOE, state space, context specifiers, and action manager) are outlined in bold in figure 40. Figure 40 also depicts additional components needed to receive data (i.e., the interface/parser), coordinate real-time processing (i.e., the controller), and log and display expectations and explanations (i.e., the output interface). Arrows represent information flow between components.

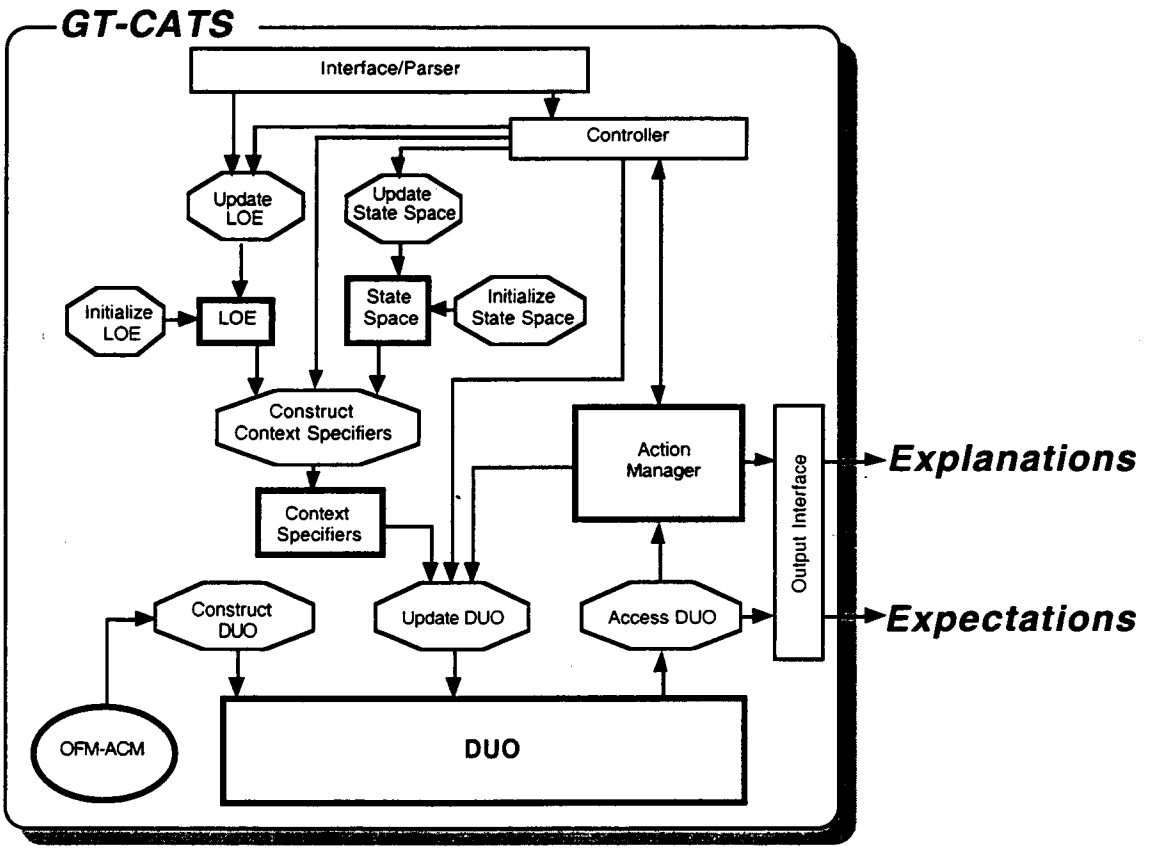


Figure 40. Functional view of the GT-CATS architecture.

Figure 40 also shows the methods used to process the knowledge representations. One set of methods instantiate the OFM-ACM in DUO, and initialize the LOE and state space. During run time, other methods update the LOE and state space as new objectives and system state information are received, activate context specifiers, and update and access DUO. Figure 40 also depicts the relationship of the GT-CATS action manager to the other components of the architecture. The action manager uses methods that access DUO in the process of explaining detected actions. The action manager also has methods that change the status of nodes in DUO when executing the revision process.

The OFM-ACM in the GT-CATS architecture

The OFM-ACM model of operator-automation interaction is the critical knowledge representation in the GT-CATS architecture. The OFM-ACM must therefore be represented at a level of detail and fidelity suitable for instantiating its knowledge in DUO. Each node in the OFM-ACM is represented in an ASCII file that GT-CATS reads to instantiate the DUO representation (Table 1). This arrangement affords easy inspection of the OFM-ACM, and simplifies modifications.

The OFM-ACM file structure uses nested brackets to represent the hierarchical structure of the OFM-ACM. The OFM-ACM heterarchy is represented by activities represented by brackets at the same nested level. Keywords indicate the type of knowledge specified after them. An exclamation point demarcates a comment line; blank lines are ignored.

GT-CATS uses a recursive method to construct DUO from the OFM-ACM text file representation.

Table 1. Generic OFM-ACM file structure.

```
! activity-1.ofm-acm
! a generic example of the OFM-ACM file structure

{ activity-level-1 activity-1
     reason information about why activity-1 is preferred
     ! conditions are optional--they may be inherited
     active-when context specifier 1
     active-when context specifier 2
     active-when context specifier 3
     active-when-1 context specifier 2
     active-when-1 context specifier 4
     ! activity type and agent may be used only where necessary
     activity-type type-designator
     agent agent-designator
     ! automation-mode is used only at the mode-selection level
     automation-mode mode-designator
     { activity-level-2 activity-2
          reason information about activity-2
          active-when context specifier 5
          active-when context specifier 6
          active-when context specifier 7
          active-when-1 context specifier 6
          active-when-1 context specifier 8
          activity-type type-designator
          agent agent-designator
          subfile lower-level-activity-1.ofm-acm
          subfile lower-level-activity-2.ofm-acm
          subfile lower-level-activity-3.ofm-acm }
     { activity-level-2 activity-3
          reason information about activity-3
          active-when context specifier 9
          active-when context specifier 10
          active-when-1 context specifier 6
          active-when-1 context specifier 11
          active-when-1 context specifier 12
          activity-type type-designator
          agent agent-designator
          subfile lower-level-activity-2.ofm-acm
          subfile lower-level-activity-4.ofm-acm
          subfile lower-level-activity-5.ofm-acm } }
```

The method is described in detail in the next section; here the discussion will be limited to the specification rules. Table 1 shows an activity (activity-1) at level "activity-level-1" with two sub-activities (activity-2 and activity-3) at level "activity-level-2." The inclusion of these activities within the brackets of activity-1 captures the hierarchical decomposition. Figure 41 shows the structure that is built from the file shown in table 1. Depending on the

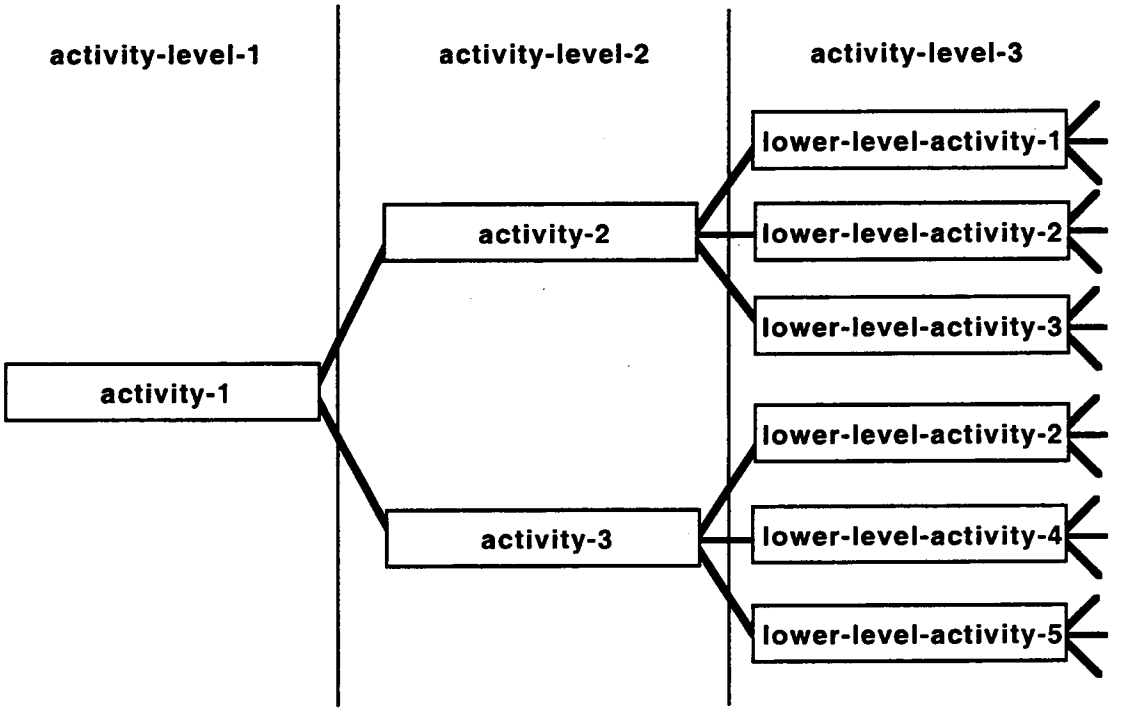


Figure 41. Structure of decomposition that results from the file specification shown in Table 1.

conditions for activity-2 and activity-3, these activities may be concurrent or serial.

Activity-1 has two sets of conditions under which it is active. The keywords "active-when" and "active-when-1" provide a simple method for specifying these conditions. These keywords specify that activity-1 is active either when hypothetical context specifiers 1, 2, and 3 are present, since they all follow the keyword "active-when," or when context specifiers 2 and 4 are active, since they all follow the keyword "active-when-1." Similarly, activity-3 is active either when context specifiers 9 and 10 are active, or when context specifiers 6, 11, and 12 are present. As table 4-1 shows, the conditions are grouped with the appropriate keywords. Because of the way in which the search for active activities proceeds in GT-CATS, conditions need not always be specified; an activity with no conditions is active if its parent activity is active.

The keyword "reason" is followed by a statement of why the activity is preferred under the specified conditions. The keywords "activity-type," "agent," and "automation-mode" are used to specify the corresponding knowledge. Automation-mode knowledge is included only in activities at the mode-selection level of the OFM-ACM. This knowledge consists of the automation mode that is engaged if the mode selection is chosen. Activity-type and agent knowledge is used at the action level to indicate whether the activity is manual, perceptual, verbal, or cognitive, and the agent responsible for performing the activity.

The last important feature of the OFM-ACM file specification is the "subfile" keyword. The subfile keyword is followed by the name of the file that contains the knowledge about the activities into which an activity is decomposed. For example, in table 1 activity-2 and activity-3 are both decomposed into three activities at the next level of abstraction, as indicated by the three subfile designators inside their brackets. Each subfile may specify further decomposition of these activities. The subfile keyword is a convenient shorthand because activities that occur multiple times in the OFM-ACM can be specified in a single file that is read multiple times. In table 1, activity-2 and activity-3 are both decomposed such that

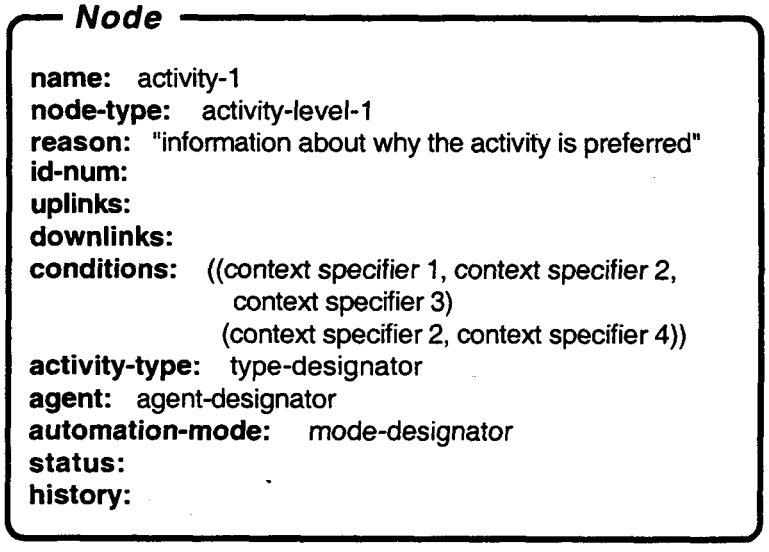


Figure 42. A generic node structure in DUO.

"lower-level-activity-2" appears in their decompositions. Reading the file "lower-level-activity-2.ofm-acm" for each occurrence makes lower-level-activity-2 a sub-activity of both, as shown in Figure 41.

The Dynamically Updated OFM-ACM

The Dynamically Updated OFM-ACM (DUO) is instantiated as a collection of instances of activity nodes. An activity node contains all of the knowledge specified for an activity in the OFM-ACM, plus knowledge to support processing. Figure 42 shows an activity node that would be created for activity-1, as specified in table 1.

DUO construction procedure

The nodes that comprise DUO are constructed from the OFM-ACM by a recursive procedure. The procedure takes as input the highest level file (or files) in the OFM-ACM and outputs completed nodes. It works by maintaining two stacks: one for nodes that have been created but not yet completed, and one for completed nodes. The generic OFM-ACM specification in table 1 will serve as an example of how the procedure works. When the filename "activity-1.ofm-acm" is passed to DUO's construction procedure, the file is opened for reading. Comment lines are ignored, as are blank lines, so the first line read is "{ activity-level-1activity-1." The left bracket signals the creation of a node instance whose name slot is filled with activity-1 and whose node-type slot is filled with activity-level-1. At the time the node is created, the construction procedure assigns the node a unique identifier. The identifier is a number placed in the id-num slot. The construction procedure then reads the reason knowledge and creates a string used to fill the node's reason slot as shown in figure 42. The procedure then encounters the conditions knowledge. Context specifier 3-tuples appearing after the active-when keywords are grouped into one sublist in the conditions slot; those following active-when-1 keywords are grouped into another sublist. Activity-type, agent, and automation-mode knowledge are inserted into the appropriate slots if applicable.

The next piece of knowledge in the OFM-ACM specification file is "{ activity-level-2 activity-2." The left bracket again signals that new node should be created. The first node is therefore placed on the "not-yet-completed nodes" stack, and the next node is created according to the keywords in its specification. The identification number of the first node is placed in the uplinks list of activity-2 at this time to signify that activity-2 is part of the decomposition of activity-1.
When constructing the activity-2 node, the procedure encounters the "subfile" keyword.

The occurrence of this keyword has the effect of placing the partially constructed activity-2 node on the stack of not-yet-completed nodes. The procedure is then called recursively with the filename "lower-level-activity-1.ofm-acm." To understand the recursive behavior of the construction procedure, the role of the right bracket must be examined. A right bracket indicates that all the knowledge required to specify the node and all of its sub-nodes has been read. When a right bracket is encountered, the partially completed node is removed from the stack of not-yet-completed nodes, and placed on the stack of completed nodes. At the same time, its identification number is placed on the downlinks list of the first node in the stack of not-yet-completed nodes. This signals to the procedure that it has returned to processing the higher-level node. Because all OFM-ACM specification files end with a right bracket, a "subfile" keyword results in all the information in the specified file being processed into completed nodes. Thus, when the procedure returns to the previous level of recursion, all of the information in the subfiles has been processed.

When DUO's construction procedure encounters the line "{ activity-level-2 activity-3" in table 1, the stack of completed nodes contains activity-2 and all of its subnodes. The stack of not-yet-completed nodes contains activity-1 with the identification number of activity 2 already on its downlinks list. The same recursive procedure is then repeated: the new activity-3 node is instantiated, given an identification number, its conditions slot filled, its activity-type and agent slots filled, and the identification of the first node on the not-yet-completed stack (activity-1) is placed in the uplinks slot of activity-3.

When the last right bracket in table 1 is encountered, all of the nodes in DUO have been instantiated with the exception of the node for activity-1. This final bracket signifies that activity-1 and all of its subnodes have been constructed, so activity-1 is finally placed on the list of completed nodes, and the procedure terminates.

The lowest level nodes (i.e., actions) in the OFM-ACM representation are specified in the same manner as nodes at higher levels. Actions differ from other nodes in that they have no subnodes. Thus, they are specified beginning with a left bracket and ending with a right bracket, but there are no brackets signaling further decomposition nested within the action node specification brackets, and no subfile keywords, signaling a further decomposition is not specified in another file.

During the construction procedure, nodes in DUO are assigned the initial status "inactive." The process of assigning a status to a node has the side effect of placing the status, along with the time it was attained, on the node's history list. Once constructed, DUO is ready to be used in the activity tracking process.

DUO update procedure

DUO encapsulates all the knowledge in the OFM-ACM and, furthermore, information about each activity's status in the current operator-automation interaction. When an activity attains the status "active," it is preferred in the current operating context, and is therefore expected to be performed by the operator. Thus, the process of updating DUO is the process by which GT-CATS produces expectations. DUO's updating procedure is a recursive procedure similar to the construction procedure.

Figure 43 illustrates how the DUO updating process proceeds according to the GT-CATS methodology. It works top-down, beginning with the highest level of activity (i.e., the phase level). It performs a breadth-first search, seeking nodes whose conditions are a proper subset of the current set of context specifiers. Nodes that meet this criterion are assigned active status. Nodes not meeting this criterion are assigned the status inactive. Inactive nodes are of no further interest in updating DUO; the statuses of all of their subnodes are immediately set to inactive (see figure 43).

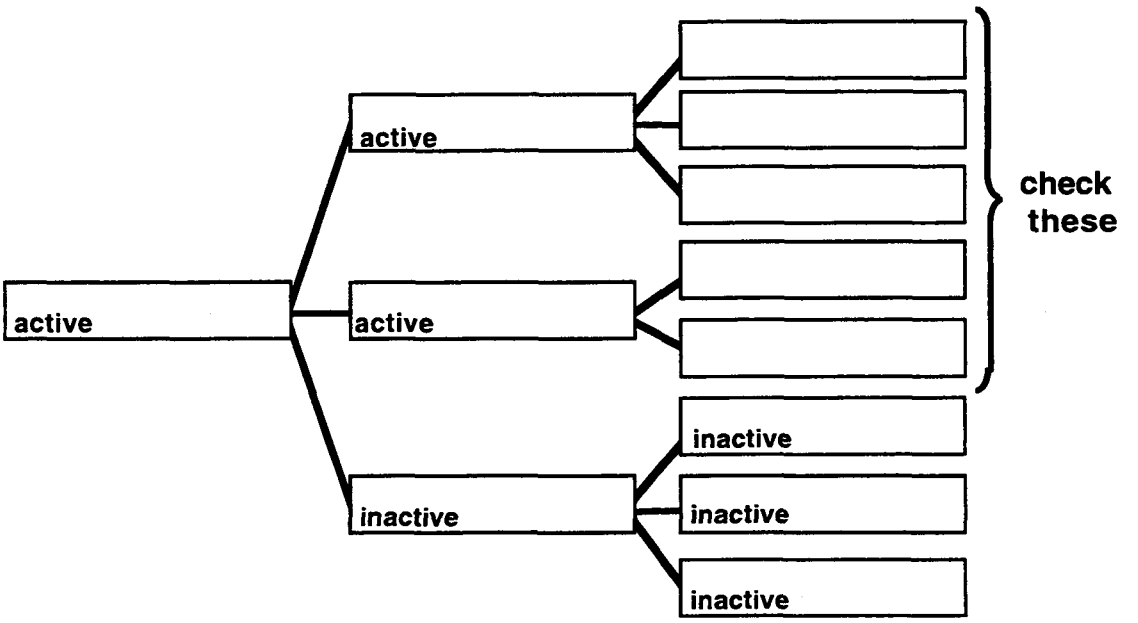


Figure 43. The DUO updating procedure pares the search at each level. When a node is found to be inactive, all its subnodes are assigned the status inactive. The search for active nodes proceeds only beneath active nodes at the previous level.

The update process next searches for active subnodes of active nodes at the next highest level. The downlinks slot of a node identifies the subnodes of the node. The subnodes of active nodes are subjected to the updating process, which continues until nodes having no subnodes (i.e., actions) have been updated.

As prescribed by the GT-CATS methodology, the DUO update procedure departs from the general process at the mode selection level. At themode-selection level, the procedure involves testing not only the context specifiers found in a mode-selection activity's conditions slot, but also the information in the automation-mode slot, which indicates the mode that should be engaged in the controlled system if the operator has in fact chosen the mode.

To reiterate (see figure 34), the rules to determine the status of the activities at the mode selection level are as follows. In the first case, if the mode selection is active according to its conditions (i.e., its conditions are context specifiers that are a proper subset of the currently active set of context specifiers), and the corresponding automation mode is also engaged in the controlled system automation, then the mode selection is active. In the second case, if the mode selection is active according to its conditions, and another applicable mode selection matches the engaged mode in the controlled system state, then the first mode is active and the second is (becoming) obsolete. In the third case, if the mode selection's automation mode matches the engaged mode, then the mode selection is obsolete, and another mode is active according to its conditions. Of course, if none of the three cases applies, the mode selection is inactive.

Insofar as the DUO updating procedure is concerned, an obsolete mode selection is treated like an active mode selection. Activities into which the obsolete mode are decomposed are still tested to see if they are active. The result of the application of this method is that activities that support monitoring the obsolete mode can be active at the same time as activities that support setting up and engaging the expected new mode selection. This is because the context specifiers used as conditions in "monitor/adjust" task subtrees below the mode selection level require that the mode selection is engaged for the monitoring

activities to be active—true by definition if the mode selection has status obsolete.

The state space

The state space in GT-CATS is comprised of a collection of instances of state variables (see figure 44). A state variable represents knowledge about a particular variable in the controlled system. This knowledge includes the name of the state variable, its latest value, the time it was updated, and its previous value and update time.

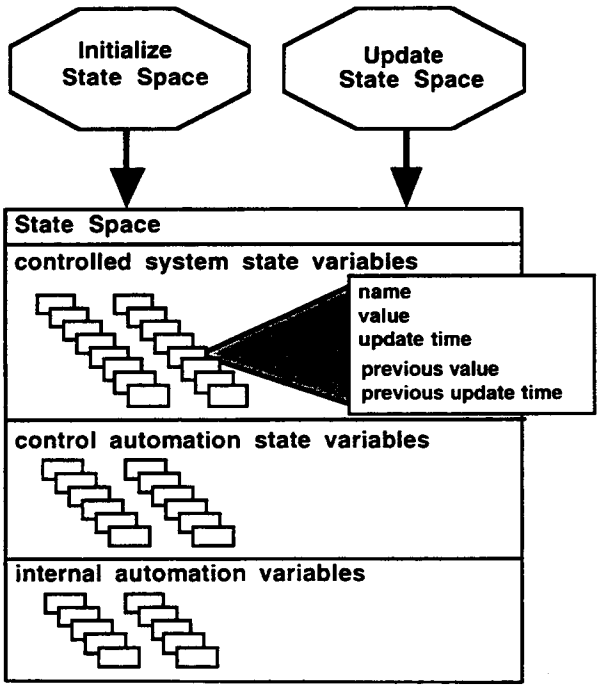


Figure 44. The GT-CATS state space.

The state space required for activity tracking in a highly automated complex dynamic system has state variables of three classes, as shown in Figure 44: (1) controlled system state variables, which are basic system performance measures; (2) control automation state variables, which represent information about engaged or armed control modes, and target values that the automation is currently attempting to achieve; and (3) internal automation variables, such as programmed target values or predictive information computed by the automation itself.

The state space is constructed by instantiating state variables for each parameter. The state space is updated with new state information as it is received from the controlled system via the interface/parser. The update process involves replacing the previous value with the old latest value and inserting the new value in place of the old latest value. Time stamp information is similarly recorded.

The Limiting Operating Envelope

The limiting operating envelope (LOE) is represented as structure with two parts (figure 45): (1) a short-term limit state, and (2) a series of long-term limit states. Each limit state consists of a set of state values to be achieved. The long-term limit states represent a series of desired limit states. The short-term limit state represents desired state values that override the values contained in the currently active long-term limit state.

Initialization of the long-term limit states entails instantiating a limit state that contains the set of values required to specify each goal in a sequence of goals to be attained. The first long-term limit state in the sequence that has not already been achieved is called the "active limit state" (figure 45). The short-term limit state is initialized with any amendments to the initial active limit state.

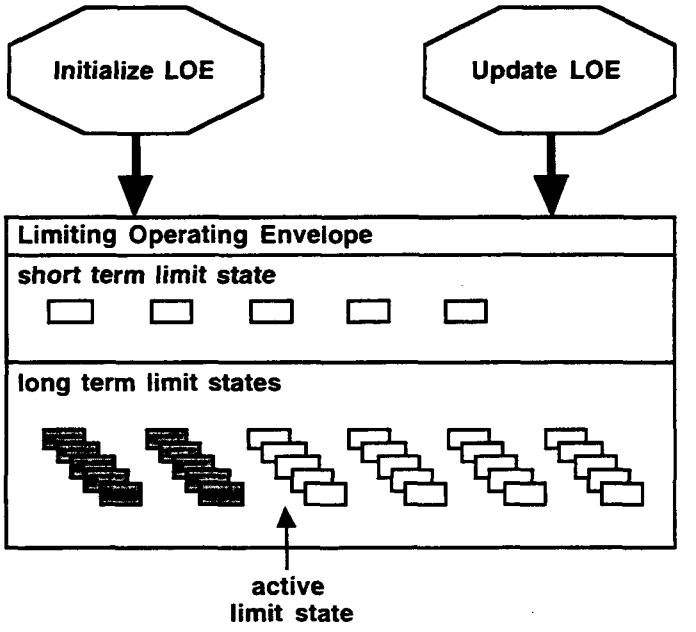


Figure 45. The Limiting Operating Envelope.

The GT-CATS LOE is updated in two ways: First, on each processing cycle, the latest state information is used to determine whether the active limit state has been achieved. If it has, it is designated "passed," and the next long-term limit state becomes the active limit state. The second type of update addresses the short-term limit state. Whenever amended goals arise from changes in environmental constraints on system operation, the amended goal values replace the corresponding values in the short-term limit state. Thus, the updated LOE representation reflects both the currently active long-term limit state and any short-term modifications to it imposed by dynamic environmental constraints.

Context specifiers

Context specifiers are a crucial component of the GT-CATS activity tracking architecture. They transform knowledge from the state space and LOE into a summary of the current operational context, which then serves as the means for referencing the conditions in DUO under which a particular activity is expected (figure 46). Context specifiers are constructed at run-time, and are designated as active or not.

The specific form used to represent context specifiers is not pivotal; any scheme that ensures unique context specifers, each with a specific connotation about the current operational context, will suffice. The GT-CATS architecture was developed with context specifiers represented as 3-tuples (figure 47). The first element indicates the type of state variable that is referenced to construct the given context specifier. The second element is the specific state variable. The third element is a qualifier that indicates the relationship of the value of the particular state variable with respect to the desired value represented in the LOE (e.g., "within limits" or "outside limits"). These qualifiers are qualitative so that the overall set of current context specifiers provides a qualitative summary of the current operational context.

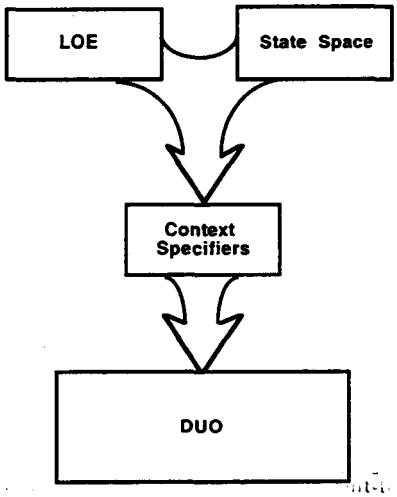


Figure 46. Context specifiers provide a dynamic summary of the current operational context, as a means of referencing the conditions in DUO.

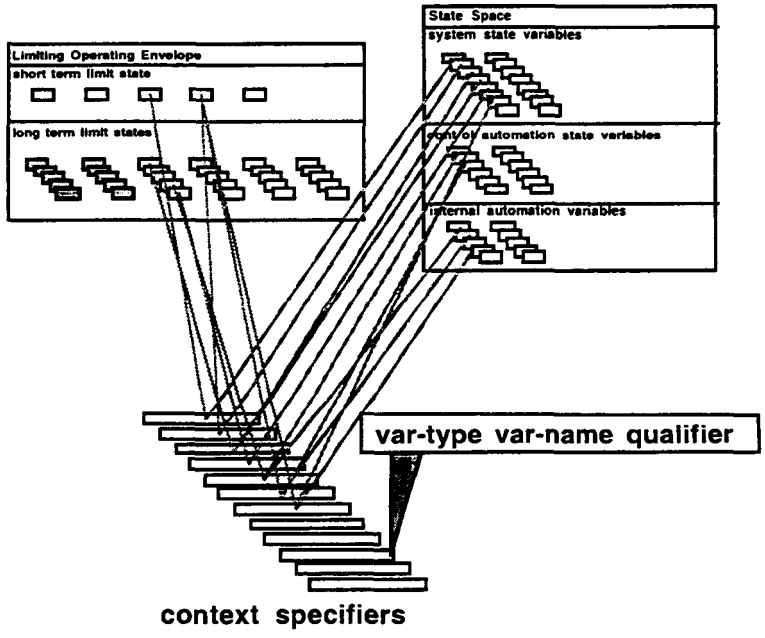


Figure 47. Activation of context specifiers.

An unconventional case arises when a context specifier is needed to represent the aggregate relationship of several state variables vis a vis the LOE. This is necessary when a one-to-one mapping between the state space and a relevant component of the active limit state, or short-term modifications to it, does not provide the required context information. Such aggregate context specifiers are useful, for example, in determining the operational context that results from a number of internal automation variables. A generic example of such a context specifier is "automation-state profile-information programmed." Here, several internal

automation variables are examined to arrive at the summary context specifier.

Context specifiers are activated based on information from updated state space and LOE representations. The GT-CATS architecture provides that the required comparisons are directly encoded—the state variables, limiting operating envelope elements, and tolerances used to activate a context specifier are directly specified in code. This allows extra predictive functions to also be encoded if such information is not directly available in the state space.

The action manager

The GT-CATS action manager plays an important role in the GT-CATS architecture. It handles all detected operator actions. It attempts to explain expected actions, explain unexpected actions, and detect errors. The action manager uses the GT-CATS controller to schedule events, accesses DUO to check to the status of activities, and updates DUO to reflect the state of operator-automation interactions. The functional relationship between the action manager, DUO, and the controller is shown in figure 48.

The action manager's first function is to explain expected operator actions (see figure 48). When an action attains active status in DUO, it is expected. Thus, when an operator action is detected, the action manager first checks whether an action of the same type is active in DUO. The action manager uses general DUO access routines for this purpose. If the action is found to be expected, the action manager explains the action based on the structure of the OFM-ACM embodied in DUO; the action is explained as supporting the associated subtask, task, and mode selection in the OFM-ACM structure. At the time the explanation is produced, the action manager assigns the action the status "explained," and the event to check whether the expected action has in fact been performed is cancelled.

The action manager's second function is to explain unexpected operator actions via the revision process. The action manager initiates the revision process upon detecting an operator action that is not expected according to DUO. After first verifying that the action is not expected, the action manager identifies all instances of the action in DUO that can support the expected function-level activity. After the action manager identifies one or more instances of the action that support the required function, the action manager schedules an event to attempt to produce an explanation for the unexpected operator action. The controller signals the action manager after the prescribed time interval to examine each of the instances of the unexpected action to find one that supports an alternative, but valid, mode selection. The action manager looks specifically to determine if a mode selection and supporting task that the action supports has become active in the time since the action was detected. To determine this, the action manager examines the history information contained in the corresponding nodes in DUO. If the action manager finds an instance of the action that meets this criterion, the action manager assigns the node a status of "revised-explained" and explains the action as supporting the task and mode selection it supports in DUO. Attempts to explain instances of the action not checked thus far are annulled.

If the action manager cannot explain a detected action that has instances in DUO, the action may be in error. The action manager therefore issues a statement that it could not explain the detected action. Although the action manager does not positively identify operator errors, an action that it cannot explain through the revision process is identified as an error candidate.

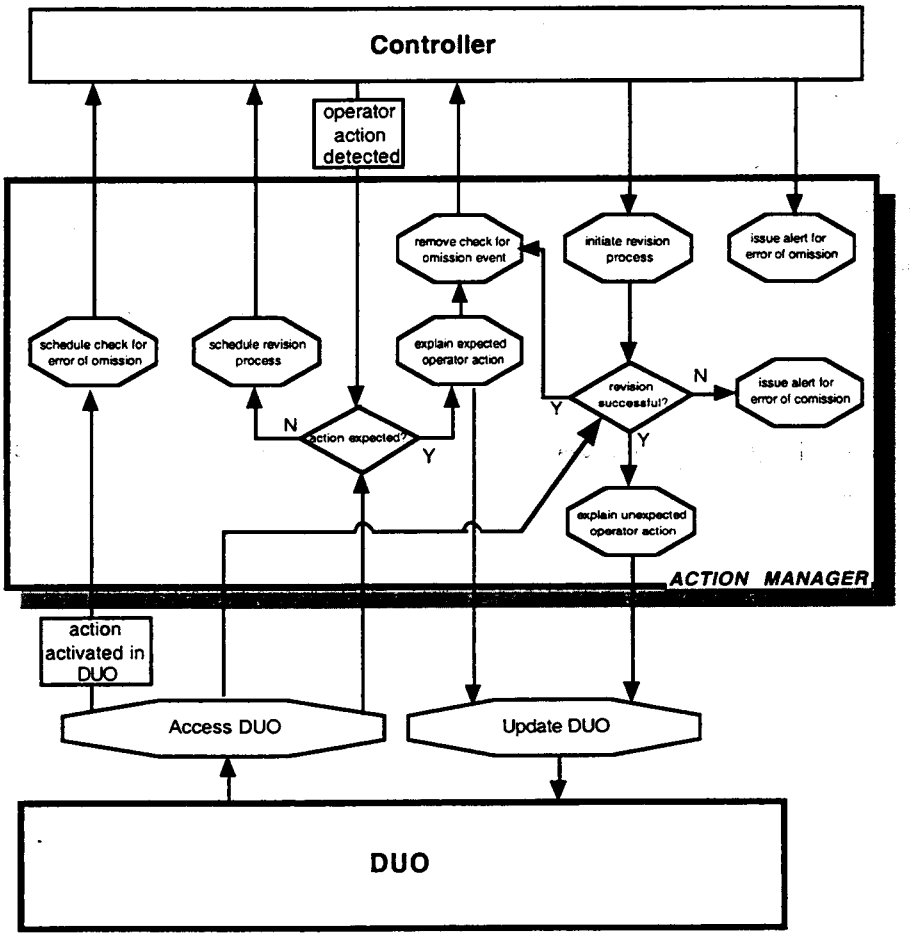


Figure 48. Relationship between the GT-CATS action manager, controller, and DUO.

The action manager's last function is to check that actions expected according to DUO are in fact performed, or that alternative valid actions are performed in their place. When an action attains active status in DUO (and is, therefore, expected), the action manager schedules an event to check that the action has been performed. If neither the expected action nor a valid alternative has been detected in the time before the event is processed, the action manager issues an alert for a possible error of omission. If the action was superseded by an alternative, but valid, action, alerts for an error of omission related to the action originally expected are annulled.

Control of processing in GT-CATS

GT-CATS requires an interface/parser, output interface, and controller to coordinate real-time processing. The architecture includes an interface/parser that can accept input in real-time from a controlled system, or read logged data from an input file. The real-time interface includes a data port used to receive data. The port is polled for data, and when data are received, the data are parsed and processed. All data are time-stamped; this information is used to update the timing information of the GT-CATS controller. The data appearing after the time stamp identifies whether the data are updated state information, new environmental constraints, or operator action data. Simulator data of these types is sent to either the appropriate updating procedure, or the action manager. The file interface operates in an analogous fashion; time-stamped data that would come from the simulator in real-time are simply read from a file.

The GT-CATS output interface has two components: a data file, and a real-time display. The file records all determinations made by GT-CATS. When actions are activated (expected) in DUO, an output file records the time the action was expected, and all other activities in DUO that are active at the time. All output from the action manager is similarly recorded. The manner in which explanations for actions are produced (i.e., from a prior expectation, or through the revision process) is also logged. Possible errors of omission or commission are similarly logged.

The GT-CATS real-time display shows the current status of all representations in the GT-CATS architecture. Windows for the state space and the LOE display the current values of state variables, and constraints from the LOE. Another window shows the status of nodes in the currently active phase/subphase of DUO. Active nodes and explained actions are color-coded. Unexpected actions are also color-coded, as are successful revisions Another window shows output similar to that logged in the output file.

The GT-CATS controller is responsible for scheduling and processing events according to its timing functions. The controller maintains a queue of events that are scheduled for later processing. When its timing functions indicate that an event is ready for processing, the controller calls the appropriate component of the GT-CATS architecture to perform the event.

The GT-CATS controller uses a simple processing cycle to coordinate updates to the state space, LOE, and DUO with action manager events. When new state information is received, the processing cycle schedules state space updates. These updates are processed, then the LOE is updated by checking whether the updated state information indicates that the active long-term limit state has been passed. After the LOE is updated, the controller initiates the DUO updating process using the new LOE and state space information. A new set of context specifiers is activated, and the DUO updating procedures use them to generate expectations by determining which nodes in DUO have active status.

The GT-CATS controller maintains an event queue for coordinating action manager events with the updating cycle. The action manager schedules events with controller in accordance with the time periods specified for detecting errors of omission or executing the revision process. When the controller processes these events the action manager is called upon to perform the required assessments and updates to DUO.

GT-CATS compared with other intent inferencing systems

Two intent inferencing systems developed previously are OFMspert (ref. 4) and OPAL (ref. 29), described in Chapter II. GT-CATS' activity tracking methodology enhances the OFM-based intent inferencing approach embodied in OFMspert. OFMspert's intent inferencing component, ACTIN, uses a blackboard architecture. Given current system state, ACTIN posts functions, subfunctions, and tasks from the OFM on the blackboard. The intent inferencing process involves mapping operator actions onto these OFM-derived functions, subfunctions, and tasks. A currently instantiated function corresponds to a current operator goal; by linking actions to the functions, subfunctions, and tasks they can support, and assessing the blackboard to ensure temporal and semantic constraints on the connected actions are met, ACTIN produces an understanding of the operator's current intentions. GT-CATS' OFM-ACM and processing scheme, in particular, differ from OFMspert. The OFM-ACM extends the OFM beyond functions, subfunctions, tasks, and actions, adding the mode-selection level and explicitly representing activities in mutually exclusive phases and subphases of system operation in the manner of Jones et al. (ref. 39) and Thurman and Mitchell (ref. 14). GT-CATS' OFM-ACM is explicitly represented in easily editable files.

GT-CATS instantiates its OFM-ACM in DUO. By using the currently active set of context specifiers to activate activities in DUO, GT-

CATS uses DUO itself to represent currently expected operator activities. Like ACTIN, GT-CATS understands actions by linking them to the mode selection, task, and subtask they support. Unlike ACTIN, however, GT-CATS attempts to determine the precise next set of activities the operator will perform, rather than all feasible activities. In this way, GT-CATS predicts one mode selection an operator will use to perform a currently active function from among several available alternatives.

GT-CATS also differs from ACTIN in the manner in which actions are explained by mapping them to the activities they support. ACTIN uses the concept of "maximal connectivity," connecting an action to all feasible tasks that it might support. GT-CATS instead associates detected actions with a single predicted mode selection. In this manner, GT-CATS explains actions as supporting a preferred mode—one of several modes available to the operator in the current situation. GT-CATS' method of associating operator actions with only one mode selection necessitates a means for explaining actions that do not support the preferred mode. GT-CATS therefore includes the revision process to explain actions that it does not expect. The revision process enables GT-CATS to use updated state information to associate unexpected actions with alternative operator mode selections. GT-CATS' revision process can determine if an unexpected action is an operator error, or associated with an alternative, but valid, mode.

OPAL (ref. 29) uses a network of plans and goals, called a Plan-Goal Graph, to establish a hierarchy of operator activities. Like both GT-CATS and ACTIN, OPAL understands operator actions by associating them with active plans and goals. OPAL differs in that it also uses scripts associated with some plans to represent procedural activities. If an active plan has an associated script, OPAL first attempts to match actions to the script and explain them as supporting the represented procedure. This script-based reasoning simplifies the intent-inferencing process, as the next action(s) are specified in the script.

If OPAL cannot match an operator action to a script, it attempts to explain the action by associating it to an active plan. Failing this, OPAL uses a procedure similar to the revision process used by GT-CATS: it uses the structure of the Plan-Goal Graph to attempt to locate other plans and goals that the action can support. OPAL identifies an action as a possible error if it does not support any plan applicable to the current situation.

Summary

The GT-CATS methodology and supporting computer architecture are designed to track operator activities in real time. The OFM-ACM imparts a specific mode orientation to the OFM, to provide GT-CATS with the capability to understand operator actions in complex systems with multiple modes. GT-CATS uses the OFM-ACM to predict the mode selection and associated activities an operator will perform in using a preferred mode selection to control the system in the current operating situation.

GT-CATS explains operator actions according to its expectations where possible; if a mismatch exists between expected and actual actions, GT-CATS uses its revision process to attempt to explain a particular action as supporting an alternative mode applicable in the current situation. If GT-CATS does not detect any action that supports a mode applicable to perform a required control function, it indicates a possible error of omission; if a detected action cannot be associated with an applicable mode, the action is flagged as a possible error. The next chapter describes an implementation of GT-CATS to track the activities of pilots using modes to navigate in glass cockpit aircraft.

5. GT-CATS Implemented for the Glass Cockpit

Introduction

This chapter describes the proof-of-concept implementation of GT-CATS for the Boeing 757/767 glass cockpit aircraft. Using an OFM-ACM developed for Boeing 757/767 pilots, GT-CATS predicts and interprets the actions pilots perform as they use the available automation modes to navigate. This chapter discusses how each of the components of GT-CATS architecture is instantiated for this application. First it describes the OFM-ACM developed for the Boeing 757/767. Next, it presents the state space and limiting operating envelope implementations. It then describes the context specifiers used in GT-CATS, and the use of these context specifiers as conditions in the OFM-ACM. The chapter next describes GT-CATS' implementation of DUO. The GT-CATS action manager is then discussed. The chapter concludes with examples of GT-CATS operation.

OFM-ACM for the B757/767

GT-CATS uses an OFM-ACM to model the activities of 757/767 pilots. The OFM-ACM decomposes the navigation task into phases, subphases, functions, mode selections, tasks, subtasks, and actions. Conditions to enable the expectation of each activity, along with activity type, agent, and reason information, complete the OFM-ACM. This discussion is divided into two parts. This first segment focuses on the decomposition of pilot activities. The second segment discusses how context specifiers are used to set up expectations for pilot activities. The reader unfamiliar with the operation of the Boeing 757/767 automation may refer to the description presented in Chapter III.

Structure of the OFM-ACM

In developing the OFM-ACM, the flight was first divided into mutually exclusive phases and subphases (figure 49). Climb phase is decomposed into three altitude-dependent subphases: climb-1000 (climb to 1,000 feet), climb-3000 (climb to 3,000 feet), and climb-cruise (climb to cruise). In the climb-1000 subphase, pilots fly the takeoff profile by manually tracking the flight director in takeoff and HDG HOLD modes. The crew configures the autopilot and autothrottle systems for the first time in the climb-3000 subphase, which begins at 1,000 feet. By the time the aircraft has reached an altitude of 3,000 feet, normal use of automation for climbing is established. The climb-cruise subphase of climb represents these functions.

Cruise phase begins when the aircraft levels off at cruise altitude. Cruise phase is divided into two subphases, init-cruise (initiate cruise) and cruise-to-descent, that differ primarily in the descent briefing the pilots perform when aproaching the top of descent (T/D). When the top of descent is passed, the descent phase of flight commences. In the GT-CATS OFM-ACM, descent is divided into two subphases, init-descent (initiate descent) and descent-to-apprch (descent to approach). Like the subphases of cruise, the subphases of descent differ primarily in the approach preparations required during the latter subphase. The OFM-ACM decomposition of the flight into phases and subphases is shown in figure 50.

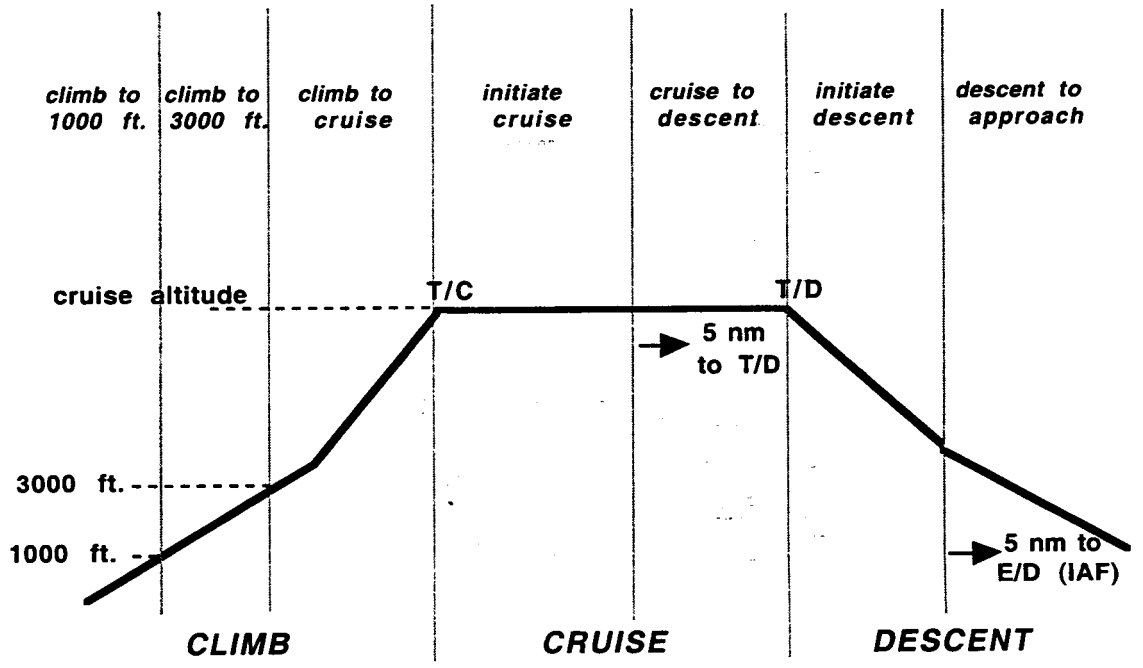


Figure 49. Phases and subphases of flight.

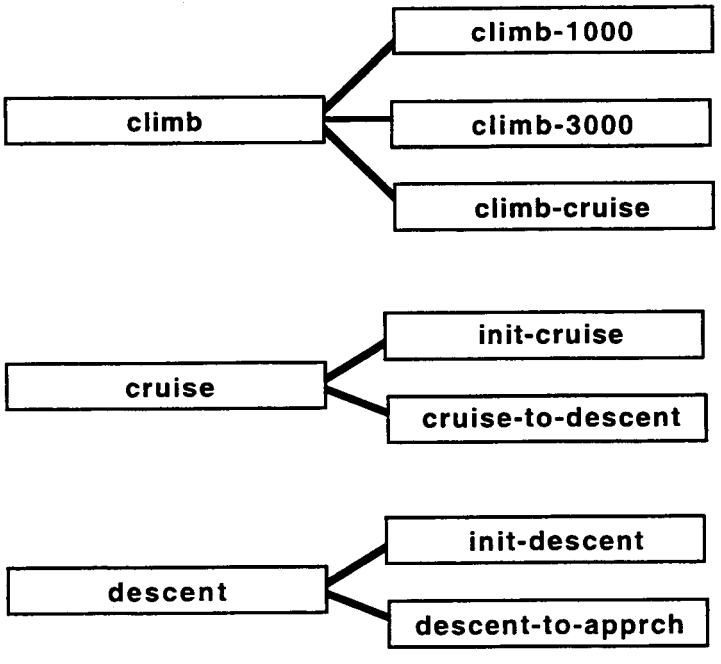


Figure 50. Phases and subphases in the GT-CATS OFM-ACM.

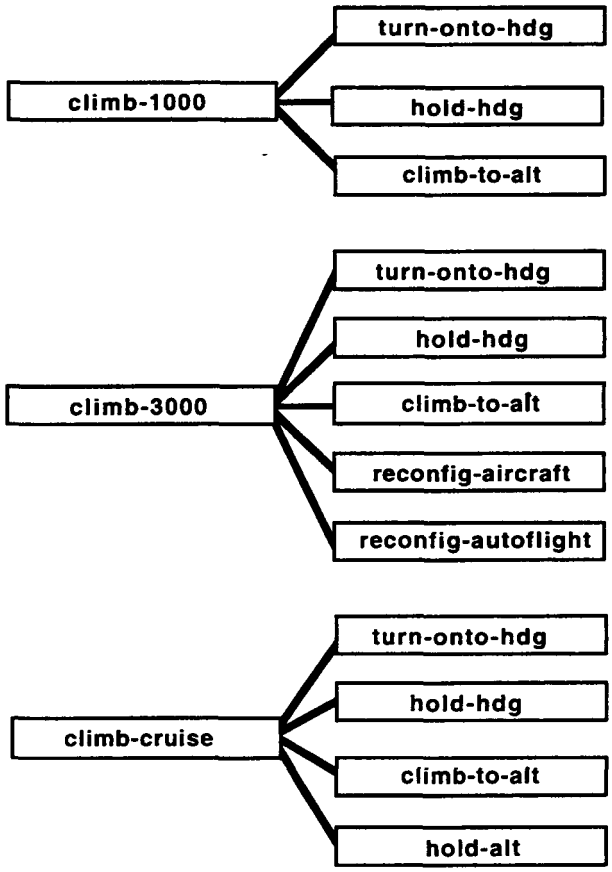


Figure 51. OFM-ACM decomposition of climb subphases into functions.

The next step in structuring the OFM-ACM is to decompose each of the subphases into the functions that the crew performs (figures 51 through 53). During the climb-1000 subphase, three navigation-related functions are required: turn-onto-hdg (turn onto a heading), hold-hdg (hold a heading), and climb-to-alt (climb to an altitude).

The climb-3000 subphase is decomposed into five functions: turn-onto-hdg, hold-hdg, climb-to-alt, reconfig-aircraft (reconfigure the aircraft), and reconfig-autoflight (reconfigure the autoflight system). The two additional functions are important during the climb-3000 subphase because the crew adjusts the aircraft configuration (i.e., flap settings, after takeoff checklist) from the takeoff configuration at this time, and configures the autoflight system (i.e., engaging an autopilot in CMD; setting climb thrust).

In the climb-cruise subphase of the OFM-ACM, pilot functions focus solely on the use of automatic modes to navigate. The functions important during the climb-cruise subphase are: turn-onto-hdg, hold-hdg, climb-to-alt, and hold-alt (hold an altitude). The hold-alt function is important above 3000 feet because ATC or published departure procedures may require temporary level-offs at altitudes below the cruise altitutude. Descend-to-alt (descend to an altitude) is not included as a function during climb phase, because a descent is not normally required during climb.

When the aircraft levels offs at cruise altitude, the cruise phase of flight commences with the init-cruise subphase (see figure 52). This subphase may require pilots to perform the following five functions: turn-onto-hdg, hold-hdg, step-climb-to-alt (step-climb to an altitude), hold-alt, and step-descent-to-alt (step-descent to an altitude). Step climbs and step descents are a means of changing the planned cruise altitude (i.e., "stepping" up or down to a new cruise altitude) as required by

ATC or weather considerations. The cruise-to-descent subphase includes an additional pilot function for reconfiguring the aircraft. The reconfig-aircraft function involves completing the descent checklist prior to the top of descent.

When the aircraft reaches the top of descent, the flight enters the descent phase (see figure 53). The first subphase of the descent phase, init-descent, is decomposed into turn-onto-hdg, hold-hdg, descent-to-alt, and hold-alt. The descent-to-apprch subphase also has a reconfig-aircraft function that encompasses approach preparations; because the required activities are not detectable, and because similar functions in the climb phase were deemed sufficient to demonstrate how GT-CATS tracks activities not related to mode selections, this function was represented by a place-holder in the GT-CATS OFM-ACM.

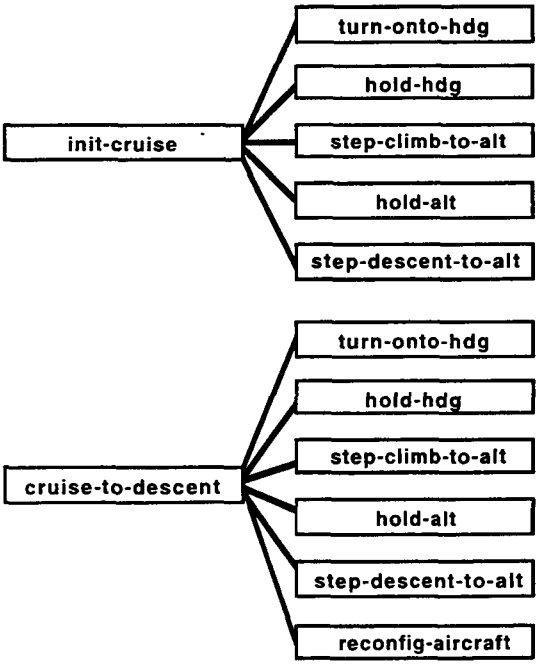


Figure 52. OFM-ACM decomposition of cruise subphases into functions.

An important requirement of the function decomposition is that the functions are uniquely determinable. As discussed in the Chapter IV, this means that there is no uncertainty as to which functions the pilots should perform at a particular time.

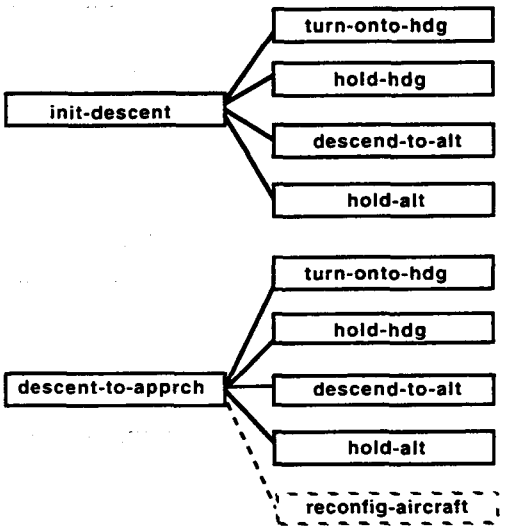


Figure 53. OFM-ACM decomposition of descent subphases into functions.

The GT-CATS OFM-ACM has functions related to controlling lateral profile (i.e., turn-onto-hdg and hold-hdg) and vertical profile (i.e., climb-to-alt, descend-to-alt, and hold-alt). One function from each set is always active at any given time (e.g., the pilot will never want to turn and hold a heading simultaneously). A lateral control function is always active concurrently with a vertical control function, but there is no ambiguity concerning which lateral and vertical control functions should be active. Figures 51 through 53 illustrate how the OFM-ACM decompositions of subphases into functions follow this principle.

The level of abstraction below the function level in GT-CATS' OFM-ACM is the mode selection level. At the mode selection level, each of the functions is decomposed into the mode selections available for performing them using the 757/767 automation modes (figures 54 through 60). Functions that are not supported by automation modes are not decomposed into mode selections; for example, functions that address reconfiguring the aircraft and autoflight systems are decomposed directly into required tasks, so no mode selections are shown to support them in figure 55.

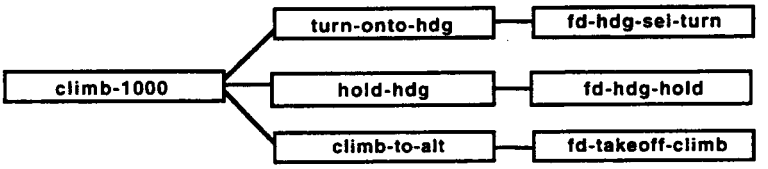


Figure 54. OFM-ACM decomposition of functions into mode selections in the climb-1000 subphase.

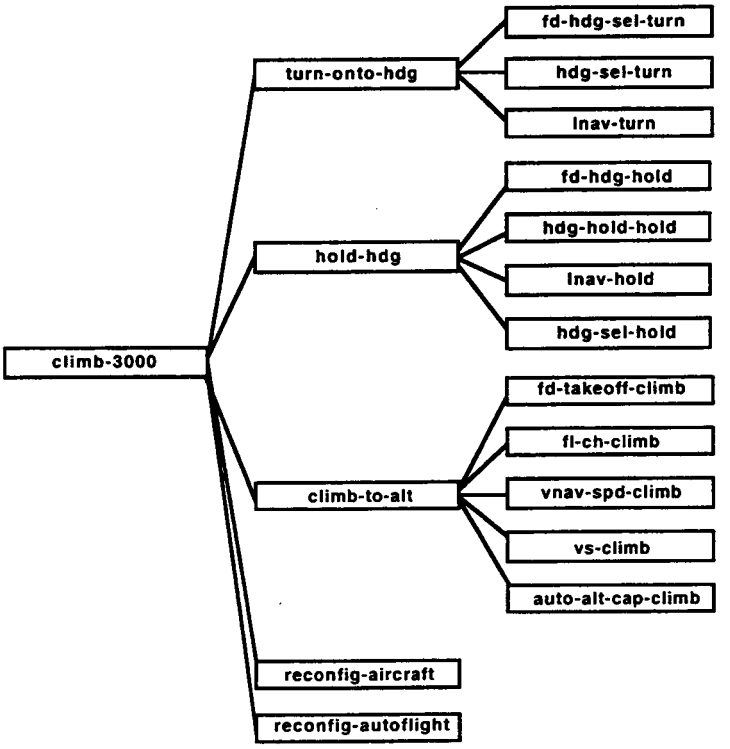


Figure 55. OFM-ACM decomposition of functions into mode selections in the climb-3000 subphase.

Depending on the subphase, the crew may have different mode selections available for accomplishing the same functions. For the climb-1000 subphase, fd-hdg-sel-turn (flight director HDG SEL turn), fd-hdg-hold (flight director HDG HOLD), and fd-takeoff-climb (flight director TO climb) are the only mode selections available to perform the respective functions. For navigation functions during the climb-3000 subphase (figure 55), a range of mode selections is available, depending on when an autopilot is engaged in command mode (CMD). The turn-onto-hdg function can be accomplished by fd-hdg-sel-turn or, after CMD mode engagement, hdg-sel-turn (HDG SEL turn) or lnav-turn (LNAV turn). The hold-hdg function is decomposed into fd-hdg-hold, hdg-hold-hold (HDG HOLD hold), lnav-hold (LNAV hold), or hdg-sel-hold (HDG SEL hold). A range of vertical axis modes similarly become available during the climb-3000 subphase (see figure 55). Besides fd-takeoff-climb, the mode selections fl-ch-climb (FL CH climb), vnav-spd-climb (VNAV SPD climb), vs-climb (V/S climb), and auto-alt-cap-climb (automatic ALT CAP climb) are available. Although the crew never actually "selects" ALT CAP mode (because it always engages automatically), it is included at the mode selection level of the OFM-ACM. This allows the auto-alt-cap-climb mode selection to be decomposed into monitoring tasks required to monitor the operation of ALT CAP as it completes the climb-to-alt function begun with another mode selection.

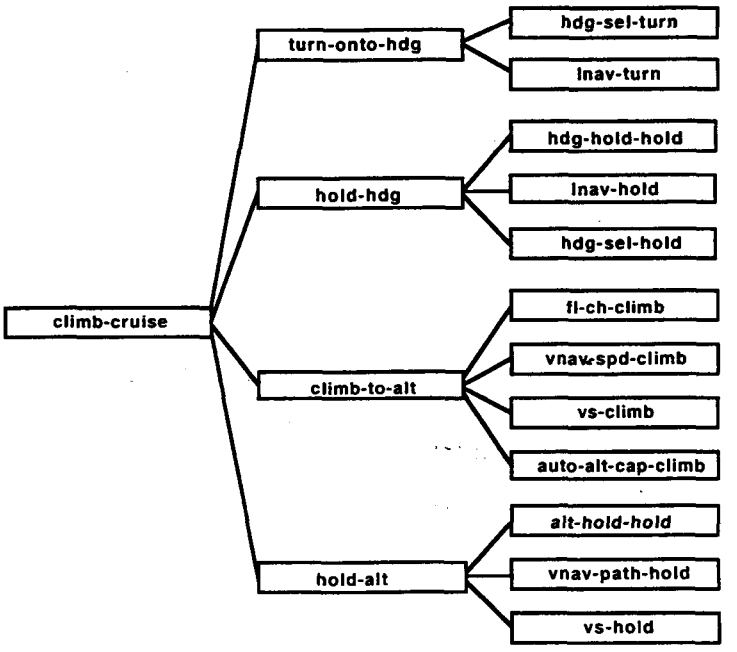


Figure 56. OFM-ACM decomposition of functions into mode selections in the climb-cruise subphase.

The climb-to-alt function for the climb-cruise subphase are decomposed into the same mode selections as those available for use in the climb-3000 subphase following autopilot CMD mode engagement. In the climb-cruise subphase, however, the hold-alt function is introduced (see figure 56). The hold-alt function is decomposed into the mode selections alt-hold-hold (ALT HOLD hold), vnav-path-hold (VNAV PTH hold), and vs-hold (V/S hold). The vnav-path-hold mode selection reflects a possible automatic VNAV transition to the VNAV PTH submode from the VNAV SPD submode, if VNAV levels the aircraft to meet a FMS-programmed waypoint crossing restriction. Both the alt-hold-hold and vnav-path-hold mode selections may be manually engaged by the pilot, or automatically engaged by the AFDS (vnav-path-hold as just discussed; alt-hold-hold via a mode transition from ALT CAP mode to ALT HOLD mode). The vs-hold mode selection represents the possibility of selecting a vertical speed of zero on the MCP while in V/S mode.

In the GT-CATS OFM-ACM, the mode selection alternatives for the turn-onto-hdg and hold-hdg functions are available in each subphase of flight after their introduction in the climb-3000 subphase. Mode selections pertaining to vertical axis modes in the descent phase are analogous to those in the climb-cruise subphase (see figures 57 and 58). Instead of fl-ch-climb, vnav-spd-climb, vs-climb, and auto-alt-cap-climb, the alternatives are fl-ch-descent, vnav-descent, vs-descent, and auto-alt-cap-descent. A notable difference in these decompositions is the unspecified submode information in "vnav-descent." Both VNAV PTH and VNAV SPD are available in descent, but they are not distinguished in the decomposition. The reason for this is twofold: first, the distinction is not important in predicting manual operator actions (i.e., the same set is applicable to either VNAV PTH or VNAV SPD); second, as discussed in detail later, the set of context specifiers as implemented cannot predict when a transition between VNAV PTH and VNAV SPD will occur.

The GT-CATS OFM-ACM models the cruise phase of flight as having climb-to-altitude, hold-altitude, and descend-to-altitude functions. In the mode selection decompositions of these functions (see figures 59 and 60), the

vnav-step-climb and vnav-step-descent mode selections represent the use of VNAV to change a cruise altitude by reprogramming the CDU according to FMS-computed projections about fuel economy at different cruise altitudes.

Each mode selection in GT-CATS' OFM-ACM is decomposed into the tasks required to use the corresponding 757/767 automation modes. As figure 61 shows, using FL CH as an example, mode selections are commonly organized in to "setup/engage" and "monitor/adjust" tasks. Two subtasks comprise the setup-eng-fl-ch task: set-mcp-alt (set the MCP altitude) and eng-fl-ch (engage FL CH). The set-mcp-alt subtask is supported by the action dial-mcp-alt (dial the MCP altitude knob). The eng-fl-ch subtask is supported by the push-fl-ch-sw (push MCP FL CH switch) action. Four subtasks comprise the mon-adj-fl-ch-climb task: mon-fl-ch-climb-profile (monitor FL CH climb profile), adjust-mcp-alt (adjust the MCP altitude), adjust-mcp-ias (adjust the MCP indicated airspeed), and mon-fl-ch-engd (monitor that FL CH is engaged). The mon-fl-ch-engd (monitor FL CH engaged) subtask is supported by the action mon-fl-ch-adi-annc (monitor the FL CH ADI annunciator).

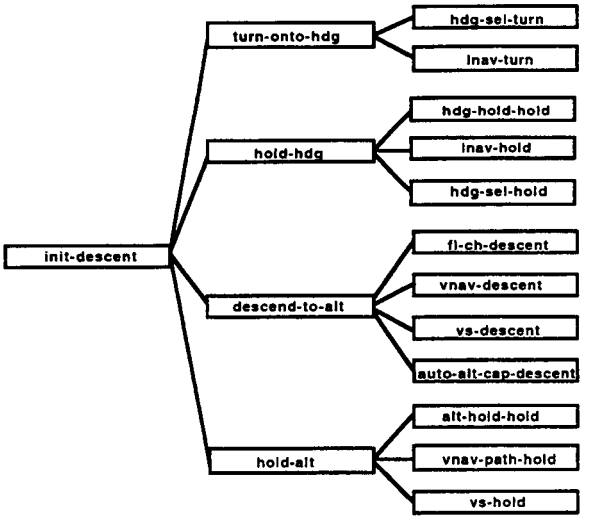


Figure 57. OFM-ACM decomposition of functions into mode selections in the init-descent subphase.

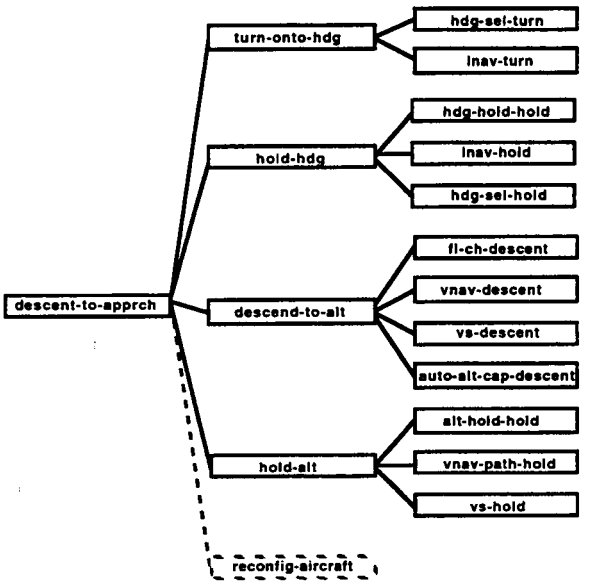


Figure 58. OFM-ACM decomposition of functions to mode selections in the descent-to-apprch subphase.

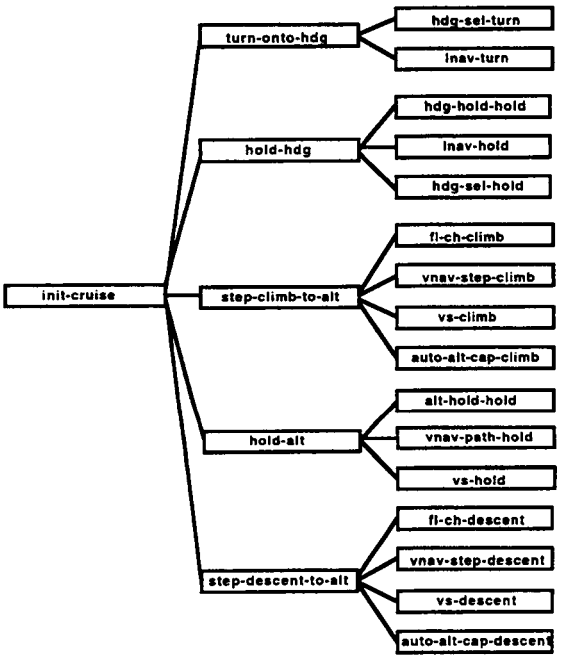


Figure 59. OFM-ACM decomposition of init-cruise subphase functions into mode selections.

The remaining subtasks of the mon-adj-fl-ch-climb subtask illustrate key features of the OFM-ACM modeling approach. First, two important adjustments can be made to FL CH once the mode is engaged. If ATC clears the

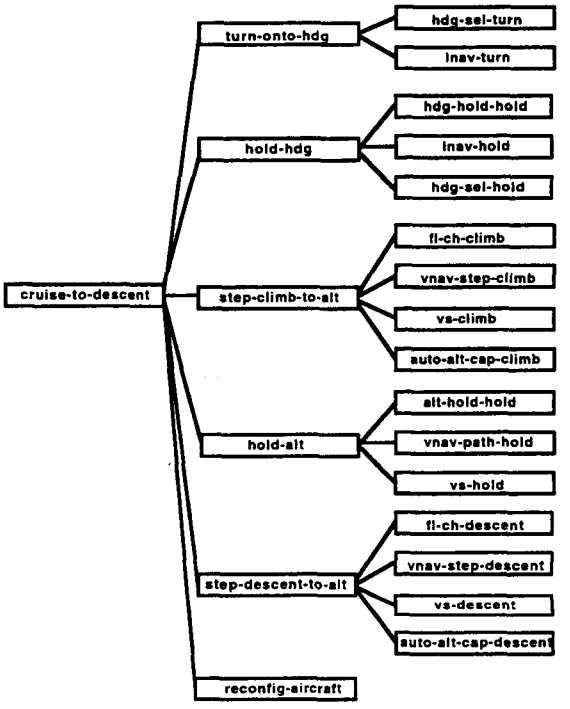


Figure 60. OFM-ACM decomposition of cruise-to-descent functions into mode selections.

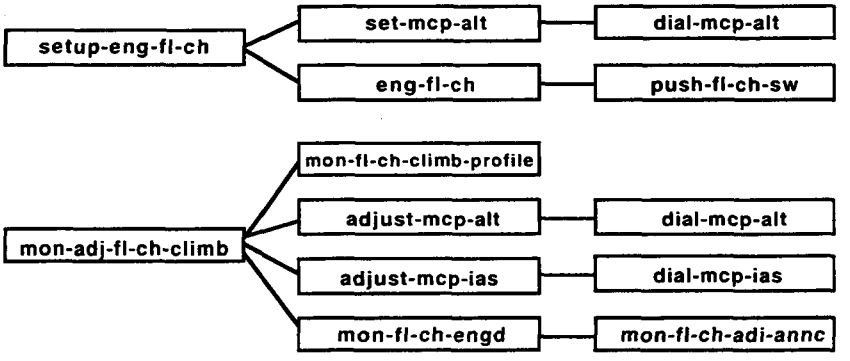


Figure 61. Tasks, subtasks, and actions supporting the fl-ch-climb mode selection.

aircraft to higher altitude before FL CH transitions to ALT CAP (to capture the previously cleared altitude), a pilot can adjust the MCP altitude and continue the climb in FL CH. This adjust-mcp-alt subtask is supported by a dial-mcp-alt action. Another aspect of FL CH mode usage involves the autopilot SPD mode that engages when FL CH engages. Upon FL CH mode engagement, the MCP speed window displays the current airspeed. Pilots are trained to adjust the airspeed using the MCP speed knob following FL CH mode engagement. Thus, the mon-adj-fl-ch-climb task also has a adjust-mcp-ias subtask, supported by a dial-mcp-ias action.

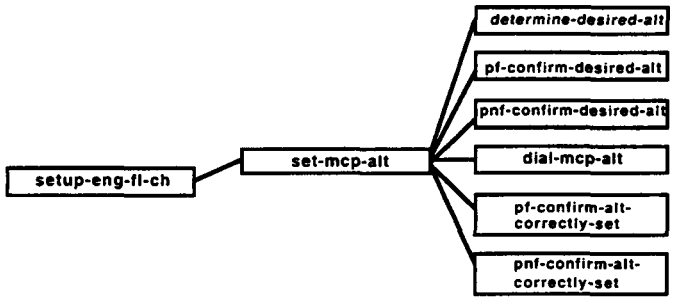


Figure 62. Cognitive, verbal, and perceptual actions that also support the set-mcp-alt subtask of the setup-eng-fl-ch task.

The mon-fl-ch-climb-profile subtask is not further decomposed in the GT-CATS OFM-ACM, because none of the actions required to monitor the profile are manual. It is nonetheless included in deference to its importance to the mon-adj-fl-ch-climb task. A similar modeling perspective leads to the decomposition of tasks into subtasks supported by a single action. These subtasks are also supported by various cognitive, perceptual, and/or verbal actions. Figure 62 shows, for example, actions involved with the subtask set-mcp-alt. Besides the manual dial-mcp-alt, the decomposition could include a cognitive action to determine the altitude that should be set, verbal confirmation actions from both the PF and PNF, and similar confirmations that the altitude was indeed set correctly. Many such actions were omitted from the GT-CATS OFM-ACM for parsimony. Others were included to demonstrate that GT-CATS can effectively expect these actions. The complete GT-CATS OFM-ACM for the 757/767 glass cockpit is illustrated in Appendix B.

State space

The GT-CATS state space is a collection of 757/767 state variables. The GT-CATS state space represents general system state variables (aircraft state), control automation state variables (Autoflight System (AFS) state, i.e., MCP-selected target values and engaged/armed modes), and internal automation variables (FMS state). These elements of the GT-CATS state space are depicted in figure 63.
The GT-CATS state space is constructed by instantiating state variables for each parameter.

Figure 5-16 shows each of the 757/767 state variables used in GT-CATS. Seven state variables represent aircraft state: hdg (heading), msl-alt (mean sea level altitude), agl-alt (above ground level altitude), spd (airspeed), vs (vertical speed), lat (latitude), and long (longitude).

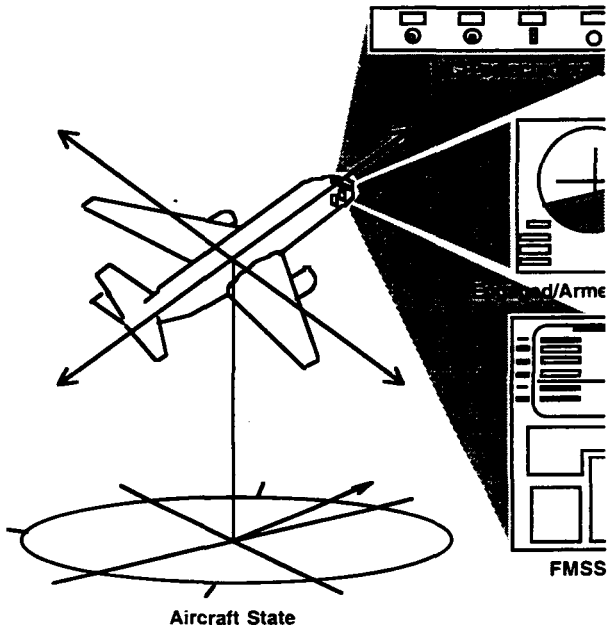


Figure 63. State space elements represented in GT-CATS' 757/767 state space.

Eleven AFS state variables represent the state of the 757 control automation (see figure 64). Five of these represent the armed/engaged modes: roll-engd (engaged roll mode), roll-armed (armed roll mode), pitch-engd (engaged pitch mode), pitch-armed (armed pitch mode), auto-thr-engd (engaged

autothrottle mode). Two others represent autopilot and autothrottle status: cmd-mode (autopilot status—F/D or CMD mode) and tsp (autothrottle thrust limit selected—CLB (climb thrust) or not). The remaining four AFS state variables reflect values selected on the MCP: mcp-hdg (MCP-selected heading), mcp-alt (MCP-selected altitude), mcp-spd (MCP-selected airspeed), and mcp-vs (MCP-selected vertical speed).

Ten additional state variables represent internal FMS variables. The first two (see figure 64) are vnav-tgt-alt (VNAV target altitude) and vnav-tgt-spd (VNAV target speed). These state variables represent the next altitude and speed that the FMS is programmed to attain in VNAV mode. These target values may be associated with a waypoint crossing restriction or the programmed cruise altitude and cruise speed. The vnav-spd-int (VNAV speed intervention submode) state variable indicates whether the VNAV speed intervention submode is in use. As discussed earlier in this chapter, when VNAV speed intervention is in use, the MCP-selected airspeed (represented by the mcp-spd state variable) overrides the FMS-programmed airspeed (represented by the vnav-tgt-spd variable) as the speed that VNAV tracks.

The next three FMS state variables help to identify the aircraft's phase of flight. These are toc-passed (top-of-climb passed or not), tod-passed (top-of-descent passed or not), and vnav-event-dist (VNAV event distance). The vnav-event-dist variable indicates the distance to a VNAV "event," defined as when the aircraft passes a VNAV-computed point (i.e., top-of-climb, top-of-descent, or end-of-descent). Thus, vnav-event-dist indicates, for example, the distance to the top-of-descent, if the the top-of-climb point has already been passed.

Four additional FMS state variables represent important information about the aircraft's position relative to the programmed FMS route.

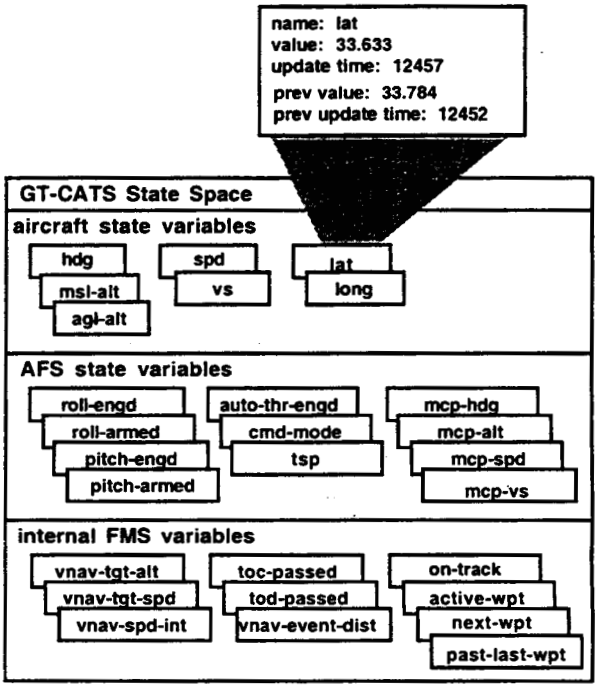


Figure 64. State variables in the GT-CATS state space.

These are on-track (on the LNAV track programmed in the FMS or not), active-wpt (the FMS active waypoint), next-wpt (the FMS waypoint following the active waypoint), and past-last-wpt (past the last FMS waypoint, signaling a route discontinuity, i.e., a condition where there are no waypoints programmed between the aircraft's current location and the start of the approach, or not). The on-track variable indicates whether LNAV is tracking the programmed route, and the remaining variables reflect the location of the aircraft along the programmed route.

Each of the GT-CATS state variables are updated when GT-CATS receives time-stamped update data. As shown in figure 64, the previous values and update times are recorded with each new update. The variables comprising the GT-CATS state space, together with the LOE, play a crucial role in generating context specifiers. The GT-CATS LOE is discussed in the next subsection.

Limiting Operating Envelope

The GT-CATS LOE represents constraints imposed by the 757/767 operating

environment. Constraints stem from three sources: ATC, the route the aircraft is scheduled to fly, and guidelines for general aircraft operation (figure 65). During normal operation these sources constrain typically operation more than the operating capabilities of the airplane itself, so they define the limiting operating envelope.

The programmed route represents the constraints from the flight plan (e.g., depart from airport KLAX, cross VTU, etc.). In GT-CATS, the flight plan route is assumed to be programmed correctly in the FMS; without this assumption, a copy of the actual programmed route would need to be included in the FMS state space to be compared against the flight plan in the LOE. This assumption allows for a parsimonious representation of the programmed route information.

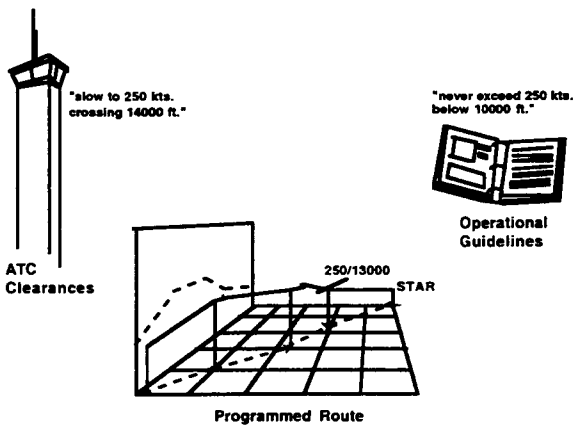


Figure 65. Sources of constraints represented in the GT-CATS LOE.

Another important assumption concerns the operational guidelines for flying the 757/767. In GT-CATS, operational guidelines, such as the 250 knot/10,000 feet speed/altitude restriction shown in figure 65, are assumed to be correctly programmed in the FMS. This is generally the case; thus, the assumption permits GT-CATS' LOE to omit an explicit representation of operational guidelines, because the guidelines are part of the programmed route.

Figure 66 shows GT-CATS LOE for the 757/767 domain. The LOE consists of two elements: ATC limits, and the programmed route. ATC limits are the "short-term" limit states of the LOE; the programmed route is the sequence of "long-term" limit states. The ATC limits include any binding values for cleared heading, cleared altitude, cleared speed, cleared vertical speed (although vertical speed clearances are unusual), and whether the aircraft is to intercept the programmed route. In determining the binding constraints represented by the LOE, ATC limits override the long-term "active-limit-state."

The programmed route has three components, as shown in figure 66: airports (the origin and destination airports for the flight), a waypoint list (a list of the planned waypoints along the flight path), and speed/altitude restrictions. A speed/altitude restriction can either be associated with a waypoint (e.g., cross RMG at 250 knots/8,000 feet), or not (e.g., do not exceed 250 knots below 10,000 feet). The waypoint structure in GT-CATS therefore includes a slot for an associated crossing restriction. GT-CATS determines the next speed/altitude restriction by using the "phase" slot of the particular speed/altitude restriction in conjunction with the limit altitude at the restriction.

The GT-CATS LOE is updated on each processing cycle. An update seeks to determine if the currently active limit state has been passed. If so, GT-CATS uses the "passed" slot to indicate that the limit state no longer constrains the flight. As figure 66 shows, to implement the LOE for both a lateral and vertical profile, the concept of active limit state is extended so that the active limit state has a lateral component (i.e., the active waypoint), and a vertical component (i.e., the binding speed/altitude restriction). These two components are both considered in the update procedure. As an example, the speed/altitude restriction in figure 66 that is not associated with a waypoint is 250 knots/10,000 feet during climb—do not exceed 250 knots below 10,000 feet. If the waypoint crossing restriction shown in figure 66 is 240 knots/8,000 feet during descent, then the climb restriction will become part of the active limit state first.

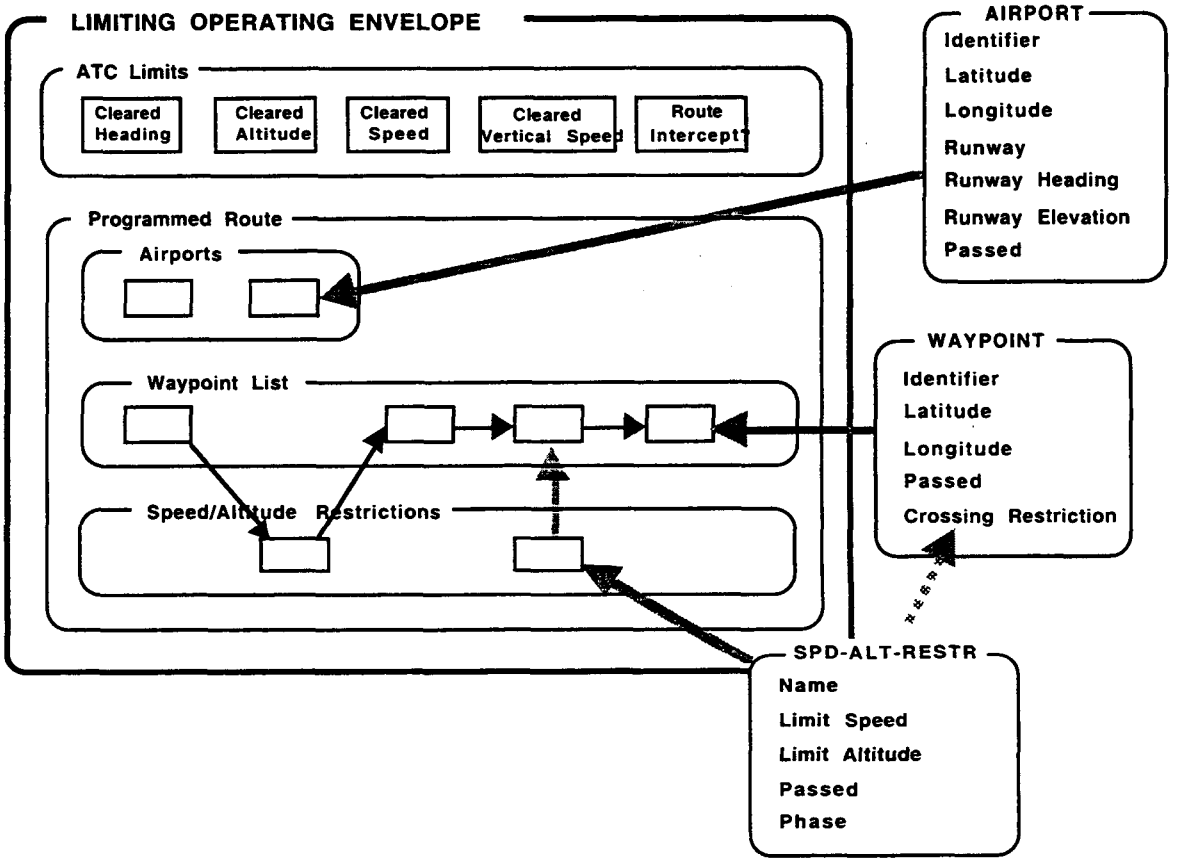


Figure 66. The GT-CATS LOE.

The active limit state, which includes a waypoint, and the next applicable speed/altitude restriction, represents the current goals for the flight—unless ATC intervenes. ATC may issue a clearance for the aircraft to fly a heading that takes it off the LNAV route programmed in the FMS; ATC may also issue a clearance for the aircraft to hold at an intermediate altitude. These ATC directives are encapsulated in the ATC limits portion of the LOE, and always override their corresponding portion of the active limit state when the LOE is used to generate context specifiers.

Context Specifiers

Context specifiers are an important component of the GT-CATS activity tracking process because they form the link between the state space, LOE, and the conditions from the OFM-ACM instantiated in DUO. Context specifiers are generated, in most cases, by comparing information from the state space to information from the LOE. These same context specifiers are used as conditions for activating nodes in DUO to generate expectations and explanations. The following subsections describe the context specifiers used in GT-CATS.

Context Specifiers activated using aircraft state variables

GT-CATS uses the following four context specifiers associated with the aircraft state variable msl-alt (mean sea level altitude):

(acrft-state alt above-limits)
(acrft-state alt within-limits)
(acrft-state alt below-limits)
(acrft-state alt outside-limits).

They are important for determining when the functions corresponding to climbing, descending, and holding altitude are active. Altitude is "within-limits" when, in the case where a crossing restriction at a downpath waypoint is the binding constraint on altitude, altitude is no more than 60 feet above or 50 feet below the limit altitude. When no such crossing restriction is binding, altitude is "within-limits" when the altitude is no more than 60 feet above or 50 feet below the cleared altitude. Altitude is "outside-limits" and either "above-limits" or "below-limits" at all other times.

GT-CATS also activates the following two additional context specifiers from the aircraft state variable msl-alt. They are used as a heuristic for expecting a V/S mode-selection, based on the notion that V/S mode is good for small adjustments in altitude, and smoothing altitude acquistions. GT-CATS compares the variable msl-alt to the cleared altitude to generate one of these:

(acrft-state alt more-than-2000-ft-from-tgt)
(acrft-state alt less-than-2000-ft-from-tgt).

GT-CATS activates the next five context specifiers from the aircraft state variable agl-alt (above ground level altitude):

(acrft-state abs-alt above-origin-apt)
(acrft-state abs-alt at-or-above-1000)
(acrft-state abs-alt at-or-above-3000)
(acrft-state abs-alt at-or-below-1000)
(acrft-state abs-alt at-or-below-3000).

These context specifiers are important for determining when a particular subphase is active. Because the value of the agl-alt state variable is dependent on the terrain, these context specifiers are subject to some variation.

GT-CATS activates two important context specifiers from the aircraft state variable hdg (heading):

(acrft-state hdg within-limits)
(acrft-state hdg outside-limits).

These context specifiers are used for activating either "turn-onto-heading" or "hold-heading" functions. GT-CATS uses two different methods to activate one of these, depending on whether there is a cleared heading specified in the ATC limits portion of the LOE or not. If a cleared heading is given, then heading is "within-limits" when the aircraft heading is within plus-or-minus 0.5 degrees of the cleared heading. If a route intercept on a heading left to pilot discretion is required, GT-CATS uses predictive functions to produce "within-limits" context specifiers when the heading intercepts the programmed route, or the heading matches the course to the next waypoint.

GT-CATS activates two context specifiers from aircraft state variable spd (airspeed):

(acrft-state spd within-limits)
(acrft-state spd outside-limits).

These context specifiers summarize the relation between the aircraft airspeed and the cleared speed, or if there is no cleared speed specified, the VNAV target speed. They are useful for determining when speed adjustments are expected in DUO. Note that the VNAV target speed variable is just a shortcut to accessing the speed/altitude restriction that is part of the LOE's active limit statein the LOE.

Context Specifiers activated using autoflight system state variables

Context specifiers activated from autoflight system (AFS) state variables express the possible mode configurations. At a given time, the context specifier that corresponds to the state of the variable is activated, along with the appropriate "not-" context specifiers, since the possible engaged and armed modes are mutually exclusive. These context specifiers are important for expecting mode selection and setup. GT-CATS does not require the LOE to activate these context specifiers.

GT-CATS can activate eight context specifiers from the AFS state variable roll-engd (engaged roll mode). These context specifiers summarize the engaged roll mode conditions

("to" means "Takeoff mode," a specialized mode used during takeoff):

(afs-state roll-engd to)
(afs-state roll-engd not-to)
(afs-state roll-engd hdg-sel)
(afs-state roll-engd not-hdg-sel)
(afs-state roll-engd lnav)
(afs-state roll-engd not-lnav)
(afs-state roll-engd hdg-hold).

As an example of how most AFS state context specifiers work, consider the following: if roll-engd is HDG SEL, GT-CATS activates the (afs-state roll-engd hdg-sel) context specifier, along with (afs-state roll-engd not-lnav), (afs-state roll-engd not-hdg-hold), and (afs-state roll-engd not-to). Thus, GT-CATS activates context specifiers that reflect which mode is engaged, along with context specifiers that reflect the fact that all mutually exclusive modes are not engaged.
GT-CATS uses the AFS state variable roll-armed (armed roll mode) in a similar manner. These context specifiers summarize the armed roll mode conditions:

(afs-state roll-armed to)
(afs-state roll-armed not-to)
(afs-state roll-armed lnav)
(afs-state roll-armed not-lnav).

From the AFS state variable pitch-engd (engaged pitch mode), GT-CATS activates the following context specifiers to summarize the engaged pitch mode conditions (again, "to" means takeoff mode; "vs" means vertical speed):

(afs-state pitch-engd to)
(afs-state pitch-engd not-to)
(afs-state pitch-engd vnav-path)
(afs-state pitch-engd not-vnav-path)
(afs-state pitch-engd vnav-spd)
(afs-state pitch-engd not-vnav-spd)
(afs-state pitch-engd vnav)
(afs-state pitch-engd not-vnav)
(afs-state pitch-engd vs)
(afs-state pitch-engd not-vs)
(afs-state pitch-engd alt-hold)
(afs-state pitch-engd not-alt-hold)
(afs-state pitch-engd spd)
(afs-state pitch-engd not-spd)
(afs-state pitch-engd alt-cap)
(afs-state pitch-engd not-alt-cap)

GT-CATS activates the following context specifiers to predict capture of the cleared altitude given in the LOE; thus, these context specifiers involve the use of both an AFS state variable and the LOE. The first indicates that the aircraft is capturing the altitude required according to the LOE; the second indicates that the aircraft is capturing a different altitude:

(afs-state pitch-engd alt-cap-rqd-alt)
(afs-state pitch-engd not-alt-cap-rqd-alt).

Two context specifiers are activated using the AFS state variable pitch-armed (armed pitch mode). These context specifiers summarize the armed pitch mode conditions:

(afs-state pitch-armed vnav)
(afs-state pitch-armed not-vnav).

Ten context specifiers are activated from AFS state variable auto-thr-engd (engaged autothrottle mode). These context specifiers summarize the conditions relating to the engaged autothrottle mode:
(afs-state athr-engd n1)
(afs-state athr-engd not-n1)
(afs-state athr-engd thr-hold)
(afs-state athr-engd not-thr-hold)
(afs-state athr-engd fl-ch)
(afs-state athr-engd not-fl-ch)
(afs-state athr-engd idle)
(afs-state athr-engd not-idle)
(afs-state athr-engd spd)
(afs-state athr-engd not-spd).

GT-CATS activates four context specifiers from the AFS state variable cmd-mode (autopilot command mode). These context specifiers indicate the current command mode of the autopilot:

(afs-state cmd-mode fd)

(afs-state cmd-mode not-fd)
(afs-state cmd-mode cmd)
(afs-state cmd-mode not-cmd).

Even though the flight director is normally always on, GT-CATS considers flight director and autopilot CMD mode to be competing modes because the flight director is not required to use CMD mode.
The last set of context specifiers GT-CATS produces to reflect the status of 757/767 autoflight modes are activated from AFS state variable tsp (thrust select panel). These context specifiers indicate whether the pilot has selected climb thrust or not:

(afs-state tsp clb)
(afs-state tsp not-clb).

The remaining context specifiers activated from AFS state variables reflect the MCP-selected values of heading, altitude, airspeed, and vertical speed, relative to cleared values from the LOE, rather than the status of modes. These context specifiers activated from the AFS state variable mcp-spd (MCP-selected airspeed) summarize whether the MCP speed reflects a target speed specified in the LOE:

(afs-state mcp-spd within-limits)
(afs-state mcp-spd outside-limits).

Similarly, two context specifiers summarize whether the MCP heading matches the cleared heading specified in the LOE. GT-CATS uses the AFS state variable mcp-hdg (MCP-selected heading) to activate these:

(afs-state mcp-hdg within-limits)
(afs-state mcp-hdg outside-limits).

GT-CATS uses the AFS state variable mcp-alt (MCP altitude) to activate these context specifiers to summarize whether the MCP altitude reflects the cleared altitude specified in the LOE:

(afs-state mcp-alt within-limits)
(afs-state mcp-alt outside-limits).

Also, GT-CATS uses mcp-vs (MCP vertical speed) to activate context specifiers that summarize whether the MCP vertical speed is properly set:

(afs-state mcp-vs within-limits)
(afs-state mcp-vs outside-limits).

These vertical speed context specifiers are rarely needed, because ATC seldom issues clearances that specify a particular vertical speed.

Context Specifiers activated to summarize FMS state

GT-CATS uses a number of context specifiers to summarize the operating context in light of FMS state variables. The first set of these are activated from the FMS state variable vnav-tgt-alt (VNAV target altitude). These context specifiers indicate whether the programmed vertical profile corresponds to the current desired altitude, as expressed by either a restriction in the active limit state of the LOE, or the cleared altitude:

(fms-state vert-profile progrmd)
(fms-state vert-profile not-progrmd).

These context specifiers are important as conditions for using VNAV, because VNAV cannot be used if a valid vertical profile is not programmed.
GT-CATS generates the following two context specifiers from the FMS state variable vnav-tgt-spd (VNAV target speed). These context specifiers summarize whether the current FMS target speed is consistent with the current cleared speed, or the active limit state speed:

(fms-state tgt-spd within-limits)
(fms-state tgt-spd outside-limits).

The remaining FMS-related context specifiers concern the capability of the FMS, as programmed, to fly the desired lateral profile. GT-CATS activates them using predictive functions that compute whether the programmed route is consistent with the current

clearance and whether the current heading intercepts the programmed route:

(fms-state lat-profile progrmd)
(fms-state lat-profile not-progrmd)
(fms-state lat-profile-intcpt progrmd)
(fms-state lat-profile-intcpt not-progrmd)

These context specifiers are useful as conditions for using LNAV rather than HDG SEL mode to fly the lateral profile.

Context Specifiers activated to summarize phase of flight

GT-CATS requires one more set of context specifiers to use as conditions for activating a particular phase and subphase of flight. These context specifiers stem from the aircraft state variables msl-alt (Mean Sea Level Altitude) and agl-alt (Above ground level altitude), as well as the FMS state variable vnav-event-dist (VNAV event distance):

(current-phase climb in-progress)
(current-phase cruise in-progress)
(current-phase descent in-progress)
(aircraft-position less-than-5-miles-to top-of-descent)
(aircraft-position more-than-5-miles-to top-of-descent)
(aircraft-position less-than-5-miles-to end-of-descent)
(aircraft-position more-than-5-miles-to end-of-descent)

Overall, this set of context specifiers is sufficient for representing the conditions used in the GT-CATS OFM-ACM for the 757/767. In the next subsection, GT-CATS' DUO, constructed from the 757/767 OFM-ACM, is described.

Dynamically Updated OFM-ACM (DUO)

DUO provides GT-CATS' representation of current pilot activities. DUO is instantiated from the OFM-ACM as described in the previous chapter. The OFM-ACM file structures used to instantiate DUO are collected in Appendix C.

DUO is updated on each processing cycle to reflect the current state of pilot-automation interaction. The first step in GT-CATS' DUO-updating process is to activate a complete of set of context specifiers using the current state space and LOE. GT-CATS then searches DUO to determine which nodes in DUO should attain active (or obsolete) status.

Action manager

The GT-CATS action manager handles detected pilot actions. It works as described in the previous chapter, with one exception. In implementing GT-CATS for the Boeing 757/767, it became necessary to track two distinct types of actions. The GT-CATS action manager handles the first type (mode switch presses) in the general manner. The Boeing 757/767, however, also has the knobs used to select values on the MCP. GT-CATS' action manager was therefore extended to also detect when these actions were performed as expected, but an incorrect value was set.

In GT-CATS, these so-called "wrong-setting actions" are handled through an initial check of the MCP-selected value against the limiting operating envelope. When a "dial-" action is detected, an active instance of the action is first located in DUO (as with all actions). The MCP-selected value is checked and, if correct, the action is explained in the usual way. If the value is not correct, the action is assigned the status "explained-wrong-setting" to indicate that the action was expected, and could be explained as correct if not for the wrong value being set. The effect of the new status designator is to allow the action to be re-activated on the next update of DUO, so that GT-CATS expects a correction of the MCP-selected value in question.

Examples of GT-CATS operation

This section presents examples of GT-CATS operation using data exerpted from the empirical evaluation described in the next

chapter. Figure 67 shows the GT-CATS interface; the buttons at the upper left of the interface enable the user to raise or lower the windows that display the state space, the limiting operating envelope, activity tracking output, and DUO. In figure 68, the DUO display window has been brought to the foreground. The simulation time is displayed at the extreme upper left. The ATC clearance that is currently reflected in GT-CATS' LOE is displayed below the interface controls. At the bottom left of the interface is a thumbnail view of the subphase currently active in DUO that can be used to adjust the portion of DUO shown in the large DUO window.

The situation depicted in figures 67 and 68 is now examined more closely. Figure 69 shows the most recent GT-CATS output. Time values in the output window are displayed using GT-CATS' internal system time (expressed in seconds since midnight). GT-CATS prints the active ATC clearance in red. Green lettering indicates manual pilot actions GT-CATS expects; black lettering indicates expectations for undetectable actions.
Blue lettering denotes an explanation. In figure 69, GT-CATS has detected and explained two of the three actions it expected pilots to perform to comply with the recent clearance (i.e., dial-mcp-hdg and push-vnav-sw). Black lettering is used to display a notice that the third action (i.e., dial-mcp-alt) has not yet been detected.

Figure 70 shows the state space at the time shown in figure 69; State variables are grouped into aircraft state variables, AFS state variables, and FMS state variables, from top to bottom. Figure 71 shows the limiting operating envelope (LOE). Values at the top comprise the short-term limit state; values at the bottom are the long-term limit states. Check marks indicate that a limit state has been passed. Table 2 shows the context specifiers activated according to the values expressed in the state space and limiting operating envelope.

In addition to generating context specifiers at time 35113, GT-CATS' action manager also generates an explanation for the push-vnav-sw action detected at that time. Table 3 shows output printed in the Lisp environment indicating that the action manager has explained the action. Other output shown in table 3 shows the action manager event to check that all the expectations have been met.

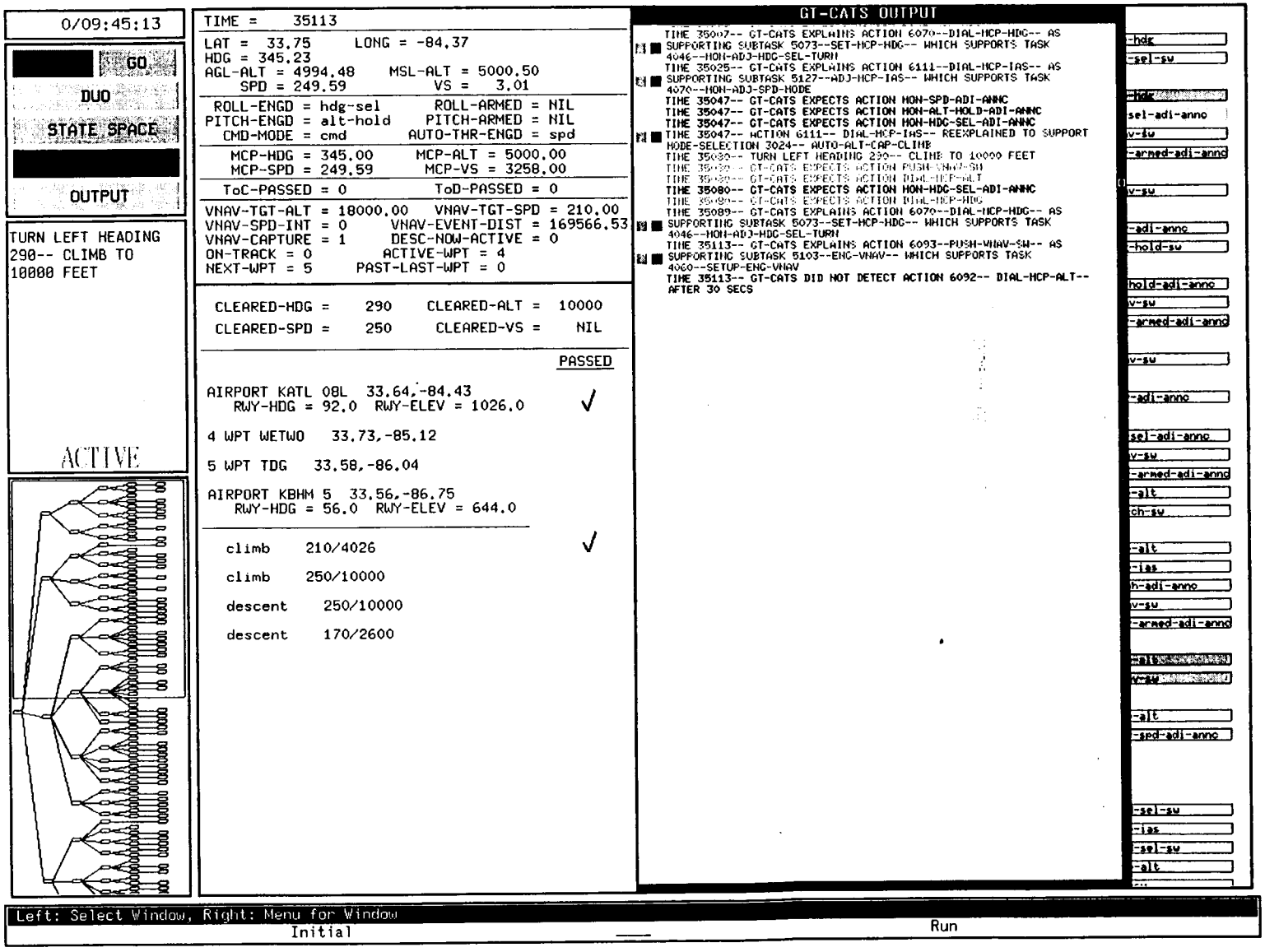


Figure 67. GT-CATS interface.

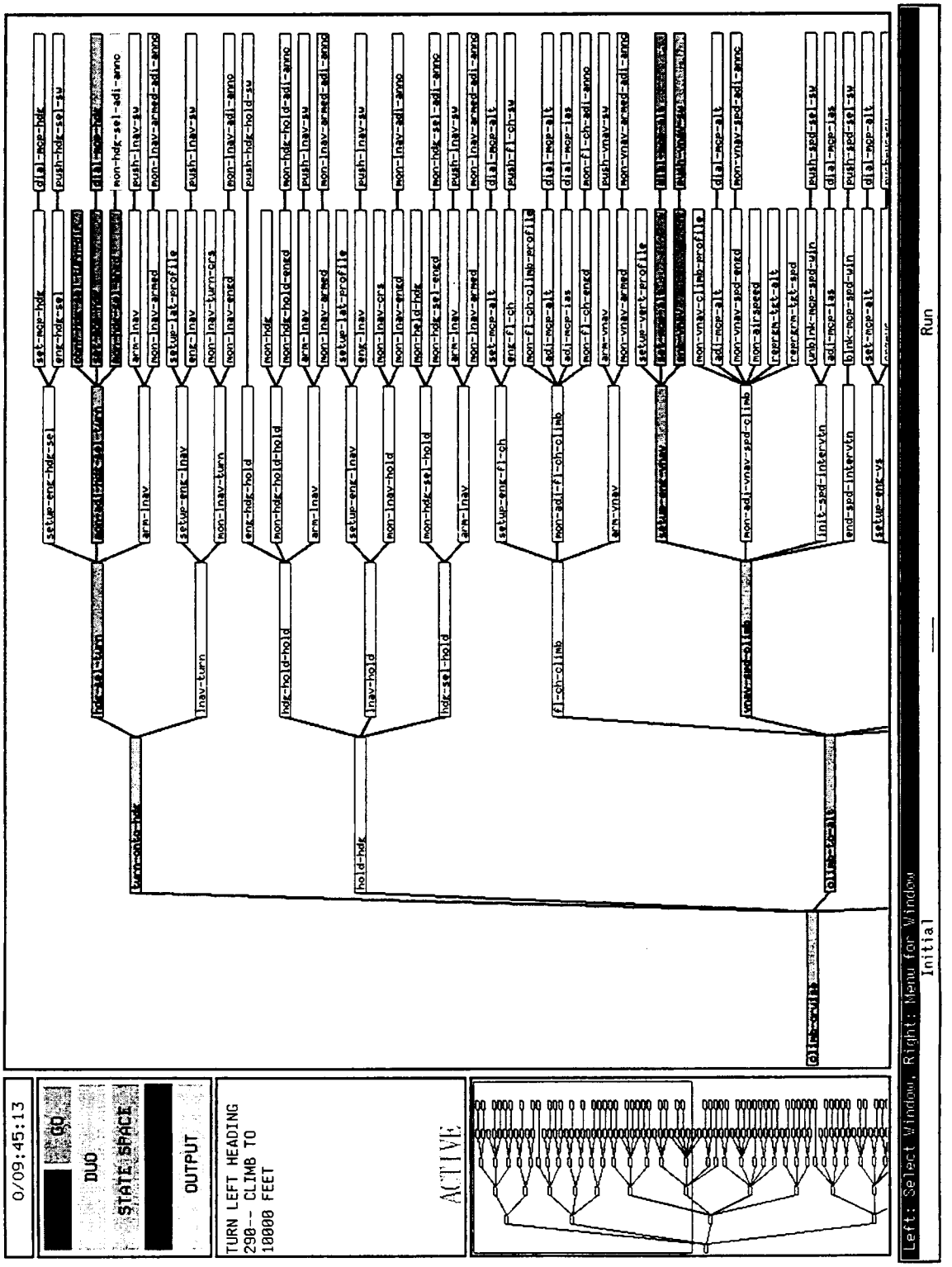


Figure 68. GT-CATS interface with DUO window exposed.

```
TIME 35080-- TURN LEFT HEADING 290-- CLIMB TO 10000 FEET
TIME 35080-- GT-CATS EXPECTS ACTION PUSH-VNAV-SW
TIME 35080-- GT-CATS EXPECTS ACTION DIAL-MCP-ALT
TIME 35080-- GT-CATS EXPECTS ACTION MON-HDG-SEL-ADI-ANNC
TIME 35080-- GT-CATS EXPECTS ACTION DIAL-MCP-HDG
TIME 35089-- GT-CATS EXPLAINS ACTION 6070--DIAL-MCP-HDG-- AS
SUPPORTING SUBTASK 5073--SET-MCP-HDG-- WHICH SUPPORTS TASK
4046--MON-ADJ-HDG-SEL-TURN
TIME 35113-- GT-CATS EXPLAINS ACTION 6093--PUSH-VNAV-SW-- AS
SUPPORTING SUBTASK 5103--ENG-VNAV-- WHICH SUPPORTS TASK
4060--SETUP-ENG-VNAV
TIME 35113-- GT-CATS DID NOT DETECT ACTION 6092-- DIAL-MCP-ALT--
AFTER 30 SECS
```

Figure 69. Enlarged view of GT-CATS output.

```
TIME =    35113
LAT =  33.75      LONG = -84.37
HDG = 345.23
AGL-ALT = 4994.48     MSL-ALT = 5000.50
    SPD = 249.59           VS =   3.01
 ROLL-ENGD = hdg-sel       ROLL-ARMED = NIL
PITCH-ENGD = alt-hold     PITCH-ARMED = NIL
  CMD-MODE = cmd        AUTO-THR-ENGD = spd
   MCP-HDG = 345.00      MCP-ALT = 5000.00
   MCP-SPD = 249.59       MCP-VS = 3258.00
  ToC-PASSED = 0           ToD-PASSED = 0
VNAV-TGT-ALT = 18000.00   VNAV-TGT-SPD = 210.00
VNAV-SPD-INT = 0      VNAV-EVENT-DIST = 169566.53
VNAV-CAPTURE = 1      DESC-NOW-ACTIVE = 0
ON-TRACK = 0         ACTIVE-WPT = 4
NEXT-WPT = 5      PAST-LAST-WPT = 0
```

Figure 70. Enlarged view of GT-CATS' state space window.

```
CLEARED-HDG =    290    CLEARED-ALT =  10000
CLEARED-SPD =    250     CLEARED-VS =    NIL
                                        PASSED
AIRPORT KATL 08L  33.64,-84.43
   RWY-HDG = 92.0  RWY-ELEV = 1026.0      √
4 WPT WETWO   33.73,-85.12
5 WPT TDG   33.58,-86.04
AIRPORT KBHM 5  33.56,-86.75
   RWY-HDG = 56.0  RWY-ELEV = 644.0
  climb    210/4026                       √
  climb    250/10000
  descent    250/10000
  descent    170/2600
```

Figure 71. Limiting Operating Envelope contents.

Table 2. Context specifers at time 35113.

```
CONTROLLED-SYSTEM time is 35113
Context Specifiers at time 35113:
      (ACRFT-STATE VS WITHIN-LIMITS)
      (ACRFT-STATE SPD WITHIN-LIMITS)
      (ACRFT-STATE ABS-ALT AT-OR-ABOVE-3000)
      (ACRFT-STATE ALT MORE-THAN-2000-FROM-TGT)
      (ACRFT-STATE ALT BELOW-LIMITS)
      (ACRFT-STATE ALT OUTSIDE-LIMITS)
      (ACRFT-STATE HDG OUTSIDE-LIMITS)
      (AFS-STATE TSP CLB)
      (AFS-STATE MCP-HDG OUTSIDE-LIMITS)
      (AFS-STATE MCP-ALT OUTSIDE-LIMITS)
      (AFS-STATE MCP-SPD WITHIN-LIMITS)
      (AFS-STATE CMD-MODE NOT-FD)
      (AFS-STATE CMD-MODE CMD)
      (AFS-STATE ATHR-ENGD NOT-IDLE)
      (AFS-STATE ATHR-ENGD SPD)
      (AFS-STATE ATHR-ENGD NOT-FL-CH)
      (AFS-STATE ATHR-ENGD NOT-THR-HOLD)
      (AFS-STATE ATHR-ENGD NOT-N1)
      (AFS-STATE PITCH-ARMED NOT-VNAV)
      (AFS-STATE PITCH-ENGD NOT-VNAV-PATH)
      (AFS-STATE PITCH-ENGD NOT-ALT-CAP-RQD-ALT)
      (AFS-STATE PITCH-ENGD NOT-ALT-CAP)
      (AFS-STATE PITCH-ENGD NOT-SPD)
      (AFS-STATE PITCH-ENGD NOT-VNAV-SPD)
      (AFS-STATE PITCH-ENGD NOT-VNAV)
      (AFS-STATE PITCH-ENGD NOT-VS)
      (AFS-STATE PITCH-ENGD ALT-HOLD)
      (AFS-STATE ROLL-ARMED NOT-TO)
      (AFS-STATE ROLL-ARMED NOT-LNAV)
      (AFS-STATE ROLL-ENGD NOT-TO)
      (AFS-STATE ROLL-ENGD NOT-HDG-HOLD)
      (AFS-STATE ROLL-ENGD NOT-LNAV)
      (AFS-STATE ROLL-ENGD HDG-SEL)
      (FMS-STATE VNAV-SPD-INT OFF)
      (FMS-STATE TGT-SPD OUTSIDE-LIMITS)
      (FMS-STATE VERT-PROFILE PROGRMD)
      (FMS-STATE LAT-PROFILE-INTCPT NOT-PROGRMD)
      (FMS-STATE LAT-PROFILE NOT-PROGRMD)
      (FMS-STATE VERT-PROFILE-INTCPT NOT-PROGRMD)
      (CURRENT-PHASE CLIMB IN-PROGRESS)
Changing node statuses......Done
Highlighting......Done
```

Table 3. Action manager output at time 35113.

```
;;; Processing detected action: (VNAV NIL)
************** explaining an action ***************
  GT-CATS explains action 6093, push-vnav-sw,
  as supporting subtask 5103, eng-vnav,
  which supports task 4060, setup-eng-vnav
(6092 6059)
action 6093, push-vnav-sw not on waiting list
(6092 6059)
action 6092, dial-mcp-alt,  not detected after 30 secs
(6092 6059)
action 6070, dial-mcp-hdg not on waiting list
@
```

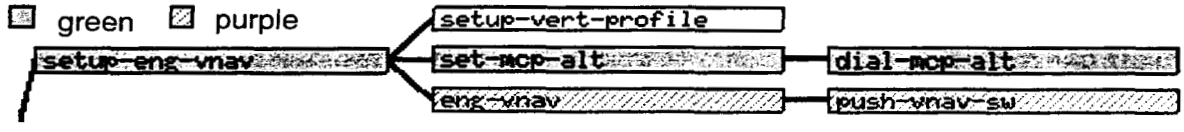


Figure 72. Closeup of DUO explaining push-vnav-sw.

```
TIME 35151-- CLIMB TO CRUISE ALTITUDE FL180-- TURN LEFT HEADING
245 AND PROCEED ON COURSE
TIME 35151-- GT-CATS EXPECTS ACTION DIAL-MCP-ALT
TIME 35151-- GT-CATS EXPECTS ACTION MON-HDG-SEL-ADI-ANNC
TIME 35151-- GT-CATS EXPECTS ACTION DIAL-MCP-HDG
TIME 35160-- GT-CATS EXPLAINS ACTION 6070--DIAL-MCP-HDG-- AS
SUPPORTING SUBTASK 5073--SET-MCP-HDG-- WHICH SUPPORTS TASK
4046--MON-ADJ-HDG-SEL-TURN
TIME 35194-- GT-CATS DETECTS INACTIVE PUSH-VNAV-SW ACTIONS--
(6090 6093 6107 6112 6118 6121 6130)
TIME 35194-- GT-CATS DID NOT DETECT ACTION 6094-- DIAL-MCP-ALT--
AFTER 30 SECS
TIME 35179-- GT-CATS EXPLAINS ACTION 6094--DIAL-MCP-ALT-- AS
SUPPORTING SUBTASK 5105--ADJ-MCP-ALT-- WHICH SUPPORTS TASK
4061--MON-ADJ-VNAV-SPD-CLIMB
TIME 35199-- GT-CATS EXPECTS ACTION PUSH-LNAV-SW
TIME 35199-- GT-CATS EXPLAINS ACTION 6072--PUSH-LNAV-SW-- AS
SUPPORTING SUBTASK 5075--ARM-LNAV-- WHICH SUPPORTS TASK
4047--ARM-LNAV
TIME 35206-- GT-CATS EXPECTS ACTION MON-LNAV-ARMED-ADI-ANNC
TIME 35218-- ACTION 6093-- PUSH-VNAV-SW-- REVISED TO SUPPORT
MODE-SELECTION 3022-- VNAV-SPD-CLIMB
```

Figure 73. Sample output at time 35218.

In the DUO window, the detected push-vnav-sw action is color-coded purple, indicating it has been successfully explained. Figure 72 shows a close-up view of this portion of the DUO window. The green color-coding in figure 72 shows that GT-CATS is still expecting the dial-mcp-alt action.

Figure 73 shows some additional output later in the same data set shown earlier. GT-CATS first expected the dial-mcp-alt and dial-mcp-hdg actions to meet the new clearance (climb to cruise altitude FL180—turn left heading 245 and proceed on course). Because applicable modes for accomplishing this are already engaged (i.e., VNAV and HDG SEL; figure 74), the pilot need only set the new target values for heading and altitude. In this example, the pilot not only sets the new required target values, but also presses the VNAV engagement switch (see figure 73). The push-vnav-sw was not expected because VNAV was already engaged. GT-CATS later revises an explanation the for this unnecessary action, as described below.

Before GT-CATS applies the revision process to the push-vnav-sw action, the aircraft turns onto the required heading, and can intercept the programmed LNAV route as directed in the clearance. At this time, GT-CATS expects the pilot to arm LNAV by pushing the LNAV mode MCP switch (see figure 73). The pilot does perform the push-lnav-sw action, and GT-CATS explains it accordingly.

By this time, GT-CATS is ready to apply the revision process to the unexpected push-vnav-sw action. The revision process finds that the action can support the use of VNAV mode, so it explains the action accordingly. The output from GT-CATS' action manager is shown in table 4 and figure 75.

```
TIME =    35218
LAT =  33.79      LONG = -84.52
HDG = 245.23
AGL-ALT = 10003.49     MSL-ALT = 10009.49
    SPD = 253.16           VS = 3410.56
 ROLL-ENGD = hdg-sel       ROLL-ARMED = lnav
PITCH-ENGD = vnav-spd    PITCH-ARMED = NIL
  CMD-MODE = cmd        AUTO-THR-ENGD = spd
   MCP-HDG = 245.00      MCP-ALT = 18000.00
   MCP-SPD = 251.66       MCP-VS = 4000.00
  ToC-PASSED = 0           ToD-PASSED = 0
VNAV-TGT-ALT = 18000.00   VNAV-TGT-SPD = 250.00
VNAV-SPD-INT = 0      VNAV-EVENT-DIST = 121979.22
VNAV-CAPTURE = 1      DESC-NOW-ACTIVE = 0
ON-TRACK = 0          ACTIVE-WPT = 4
NEXT-WPT = 5      PAST-LAST-WPT = 0
```

Figure 74. State space at time 35218.

Table 4. Action manager output showing successful revision of push-vnav-sw.

***** attempting to revise actions (6090 6093 6107 6112 6118 6121 6130) *****
revising action 6093
action 6093, push-vnav-sw, revised to support mode-selection 3022, vnav-spd-climb
***** revision complete! *****

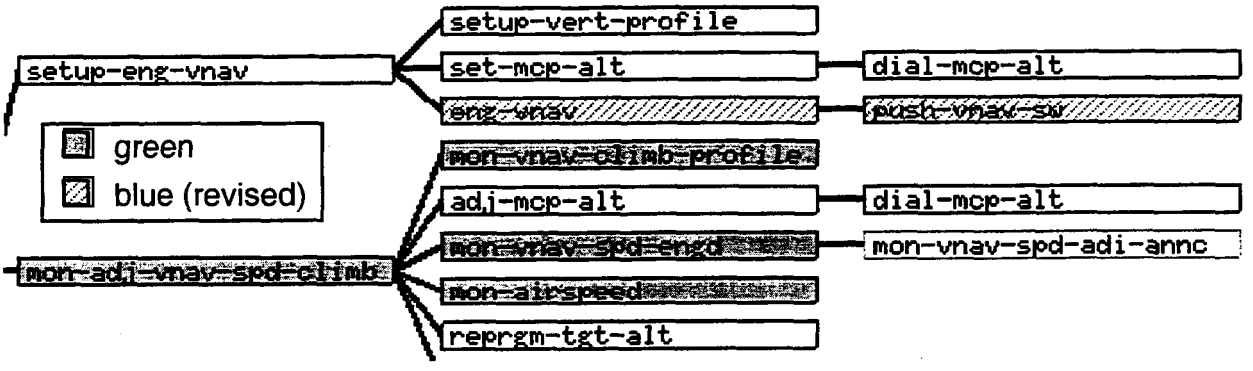


Figure 75. Closeup of DUO showing successful revision of push-vnav-sw action.

```
TIME 35625-- DESCEND TO 8000 FEET-- SLOW TO 250 KNOTS CROSSING
10000 FEET
TIME 35649-- GT-CATS EXPECTS ACTION PUSH-VNAV-SW
TIME 35649-- GT-CATS EXPECTS ACTION DIAL-MCP-ALT
TIME 35649-- GT-CATS EXPECTS ACTION MON-LNAV-ADI-ANNC
TIME 35649-- GT-CATS EXPLAINS ACTION 6344--PUSH-VNAV-SW-- AS
SUPPORTING SUBTASK 5413--ENG-VNAV-- WHICH SUPPORTS TASK
4208--SETUP-ENG-VNAV
TIME 35631-- GT-CATS EXPLAINS ACTION 6343--DIAL-MCP-ALT-- AS
SUPPORTING SUBTASK 5412--SET-MCP-ALT-- WHICH SUPPORTS TASK
4208--SETUP-ENG-VNAV BUT THE VALUE WAS NOT CORRECTLY SET
TIME 35651-- GT-CATS EXPECTS ACTION MON-VNAV-PATH-ADI-ANNC
TIME 35651-- GT-CATS EXPECTS ACTION DIAL-MCP-ALT
TIME 35651-- GT-CATS EXPLAINS ACTION 6345--DIAL-MCP-ALT-- AS
SUPPORTING SUBTASK 5416--ADJ-MCP-ALT-- WHICH SUPPORTS TASK
4210--MON-ADJ-VNAV-PATH-DESC
TIME 35866-- TURN LEFT HEADING 235-- DESCEND TO 4000 FEET-- SLOW
TO 210 KNOTS
TIME 35866-- GT-CATS EXPECTS ACTION PUSH-SPD-SEL-SW
TIME 35866-- GT-CATS EXPECTS ACTION DIAL-MCP-ALT
TIME 35866-- GT-CATS EXPECTS ACTION PUSH-HDG-SEL-SW
TIME 35866-- GT-CATS EXPECTS ACTION DIAL-MCP-HDG
TIME 35884-- GT-CATS EXPLAINS ACTION 6320--PUSH-HDG-SEL-SW-- AS
SUPPORTING SUBTASK 5381--ENG-HDG-SEL-- WHICH SUPPORTS TASK
4193--SETUP-ENG-HDG-SEL
TIME 35874-- GT-CATS EXPLAINS ACTION 6319--DIAL-MCP-HDG-- AS
SUPPORTING SUBTASK 5380--SET-MCP-HDG-- WHICH SUPPORTS TASK
4193--SETUP-ENG-HDG-SEL
TIME 35890-- GT-CATS EXPLAINS ACTION 6345--DIAL-MCP-ALT-- AS
SUPPORTING SUBTASK 5416--ADJ-MCP-ALT-- WHICH SUPPORTS TASK
4210--MON-ADJ-VNAV-PATH-DESC
TIME 35907-- GT-CATS DETECTS INACTIVE PUSH-FL-CH-SW ACTIONS--
(6337)
TIME 35907-- GT-CATS DID NOT DETECT ACTION 6349--
PUSH-SPD-SEL-SW-- AFTER 30 SECS
TIME 35908-- GT-CATS DETECTS INACTIVE DIAL-MCP-IAS ACTIONS--
(6339 6350 6359 6364 6370 6377 6382)
```

Figure 76. GT-CATS output window at time 35908.

Data from later in the same flight exemplifies other important features of the GT-CATS activity tracking process. The output from this segment, showing the responses to two clearances, is shown in figure 76. In response to the first clearance, GT-CATS expects push-vnav-sw and dial-mcp-alt. GT-CATS detects and explains the push-vnav-sw action; it also detects and explains the dial-mcp-alt action, but notes that the altitude set does not match the cleared altitude. GT-CATS therefore expects an adjustment to the set altitude. The pilot performs the action, and GT-CATS again explains it.

The second clearance shown in figure 76 requires changes in heading, altitude, and airspeed. GT-CATS expects that the pilot will continue to use VNAV mode, and that the Speed Intervention submode will be used to adjust the speed (i.e., GT-CATS expects push-spd-sel-sw). GT-CATS also expects a transition to HDG SEL, and a set heading of 235.

As figure 76 shows, the pilot does engage HDG SEL and enter the new heading; GT-CATS explains these actions accordingly. The pilot also enters the required altitude, and GT-CATS explains the dial-mcp-alt action as supporting the continued use of VNAV mode. The pilot, however, transitions to FL CH by pressing the FL CH mode engagement switch. In this case, the dial-mcp-alt action actually supports the use of FL CH. (In the evaluation described in Chapters VI and VII, GT-CATS' explanation is logged as incorrect, even though the action could have supported the continued use of VNAV mode.)

In support of FL CH mode, the pilot performs the dial-mcp-ias action. Because GT-CATS did not expect the pilot to transition to FL CH mode, the dial-mcp-ias action is also unexpected. Figure 77 shows how GT-CATS indicates unexpected actions in yellow. Note that the dial-mcp-ias speed-adjustment action also supports the init-spd-intervtn task; GT-CATS' revision process is charged with disambiguating which instance of the dial-mcp-ias action can best explain the action. Also, GT-CATS flags the push-spd-sel switch as late because it has not been detected.

The last portion of this example shows what happens when GT-CATS attempts to revise the unexpected dial-mcp-ias and push-fl-ch-sw actions. As shown in figure 78, GT-CATS successfully revises these actions to support the use of FL CH to perform the descent. Table 5 shows the action manager output in the Lisp environment, also indicating that the actions are successfully explained by the revision process.

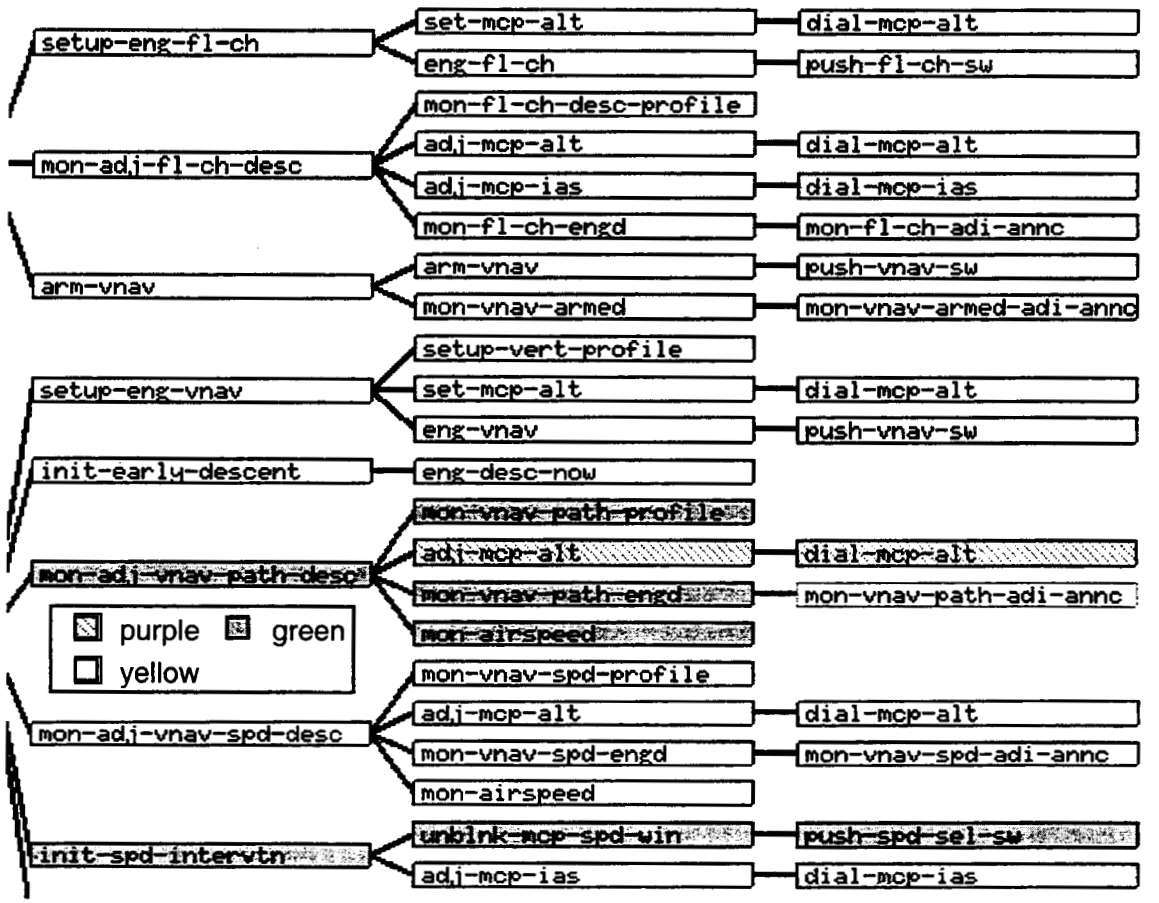


Figure 77. Closeup of GT-CATS' DUO window, showing unexpected actions highlighted in yellow.

```
TIME 35866-- TURN LEFT HEADING 235-- DESCEND TO 4000 FEET-- SLOW
TO 210 KNOTS
TIME 35866-- GT-CATS EXPECTS ACTION PUSH-SPD-SEL-SW
TIME 35866-- GT-CATS EXPECTS ACTION DIAL-MCP-ALT
TIME 35866-- GT-CATS EXPECTS ACTION PUSH-HDG-SEL-SW
TIME 35866-- GT-CATS EXPECTS ACTION DIAL-MCP-HDG
TIME 35884-- GT-CATS EXPLAINS ACTION 6320--PUSH-HDG-SEL-SW-- AS
SUPPORTING SUBTASK 5381--ENG-HDG-SEL-- WHICH SUPPORTS TASK
4193--SETUP-ENG-HDG-SEL
TIME 35874-- GT-CATS EXPLAINS ACTION 6319--DIAL-MCP-HDG-- AS
SUPPORTING SUBTASK 5380--SET-MCP-HDG-- WHICH SUPPORTS TASK
4193--SETUP-ENG-HDG-SEL
TIME 35890-- GT-CATS EXPLAINS ACTION 6345--DIAL-MCP-ALT-- AS
SUPPORTING SUBTASK 5416--ADJ-MCP-ALT-- WHICH SUPPORTS TASK
4210--MON-ADJ-VNAV-PATH-DESC
TIME 35907-- GT-CATS DETECTS INACTIVE PUSH-FL-CH-SW ACTIONS--
(6337)
TIME 35907-- GT-CATS DID NOT DETECT ACTION 6349--
PUSH-SPD-SEL-SW-- AFTER 30 SECS
TIME 35908-- GT-CATS DETECTS INACTIVE DIAL-MCP-IAS ACTIONS--
(6339 6350 6359 6364 6370 6377 6382)
TIME 35931-- GT-CATS EXPECTS ACTION MON-FL-CH-ADI-ANNC
TIME 35931-- GT-CATS EXPECTS ACTION MON-HDG-SEL-ADI-ANNC
TIME 35931-- ACTION 6337-- PUSH-FL-CH-SW-- REVISED TO SUPPORT
MODE-SELECTION 3065-- FL-CH-DESCENT
TIME 35931-- ACTION 6339-- DIAL-MCP-IAS-- REVISED TO SUPPORT
MODE-SELECTION 3065-- FL-CH-DESCENT
```

Figure 78. GT-CATS output showing successful application of the revision process to explain the push-fl-ch-sw and dial-mcp-ias actions.

Table 5. Action manager output from the revision process.

***** attempting to revise actions (6337) *****
revising action 6337
action 6337, push-fl-ch-sw, revised to support mode-selection 3065, fl-ch-descent
***** revision complete! *****
***** attempting to revise actions (6339 6350 6359 6364 6370 6377 6382) *****
revising action 6339
action 6339, dial-mcp-ias, revised to support mode-selection 3065, fl-ch-descent
***** revision complete! *****

Summary

This chapter described an implementation of GT-CATS to track the activities of pilots using modes of automation to navigate. It first presented the OFM-ACM developed for the B757/767. It then described the state space and limiting operating envelope, and DUO—an instantiation of the OFM-ACM that is annotated in real time to track pilot activities. The chapter next showed how the state space and limiting operating envelope are used to activate context specifiers that enable GT-CATS to predict pilot mode usage activities. Pilot actions detected by GT-CATS are processed by the action manager. The action manager explains expected actions based on its expectations. Unexpected actions are either explained by the revision process as supporting an alternative mode selection applicable in the current situation, or identified as possible errors. Expectations not met by pilot actions are also flagged. Finally, the chapter presents examples of GT-CATS operation exerpted from empirical evaluation data. The next chapter describes the evaluation procedure and experimental materials.

6. Empirical Evaluation

Introduction

GT-CATS was implemented to demonstrate and evaluate the effectiveness of the GT-CATS methodology for explaining how 757/767 pilots use complex flight deck automation for navigation. This chapter first gives some background on the evaluation methods used by other researchers to evaluate intent inferencing and aiding systems. It then describes the GT-CATS evaluation study, its aims, methods and expected results.

Background

Before presenting the GT-CATS validation plan, some background on evaluations performed on other knowledge-based systems is provided. By and large, such systems are evaluated in an *ad hoc* fashion. Often a system is termed "valid" merely because the overall performance of the system is in some sense "similar" to that of a domain expert. Jones, Mitchell, and Rubin (ref. 90) found this to be a prevalent approach. In their review of validation methods, they begin by rejecting Schank and Abelson's (ref. 47) assertion that merely implementing a theory on a computer validates it as effectively characterizing the process it is modeling. They refute the claims of expert systems designers who believe that because their system solves a given problem in a way similar to that of a human expert, the system represents a valid methodology for solving such problems in the given domain.

In systems that attempt to infer the intentions of a human operator, researchers have used the results of studies on the aiding component as an implicit measure of the validity of the understanding component. For example, Funk and Lind (ref. 91) find their Agent-Based Pilot-Vehicle Interface effective because pilots were able to perform better with it than with a conventional interface. They also use expert opinion to bolster support for their methodology.

OPAL, the intent inferencing system used to understand pilot's intentions in the Pilot's Associate project, was initially validated in the context of a small process control system (ref. 29). Experimental subjects controlling the system were probed, during the course of the interaction, with messages describing plans and goals from the set of plans and goals inferred by OPAL. Each probe required a yes or no response from the subject. Using a design that also included random probes, the principal hypothesis formulated for the validation experiment was that subjects would produce a reliably greater proportion of yes responses to the OPAL-generated probes than that for the random probes. Experimental data confirmed this hypothesis. The validation procedure included several additional analyses to examine subject effects, learning effects due to the unfamiliar probes used, and configuration effects of the experimental testbed.

Jones et al. (ref. 90) develop a rigorous methodology for statistically evaluating the performance of ACTIN, the understanding component of OFMspert. Their approach to evaluation uses ACTIN's approach to understanding intentions as the basis for judging its validity. ACTIN is said to understand operator actions when it infers support for the same functions, subfunctions, and tasks that a human does. They clarify that the "human" referred to here may be a domain expert performing a *post hoc* analysis, or the operator verbalizing his or her intentions concurrently with actions. Jones et al. therefore use a two-stage approach to validating their system. In the first stage, each operator action was analyzed by a domain expert and compared with ACTIN's interpretation of the same data. In the second stage, concurrent verbal protocols collected from experimental subjects were compared to the interpretations offered by ACTIN. By using this two-stage process, problems with expert comparisons (refs. 92 and 93) and potential deficiencies with verbal protocol analysis (ref. 94) are not, by themselves, allowed to sway the analysis.

Evaluation of GT-CATS

The GT-CATS evaluation study sought to assess the effectiveness of GT-CATS' activity tracking method in the context of a real-time simulation of the Boeing 757/767 autoflight system. Ten type-rated pilots from a major air carrier served as subjects for the study. The study was preceded by a pilot study, which used five type-rated pilots familiar with the goals of the GT-CATS evaluation, to ensure realism and feasibility of the materials and procedures employed in the formal evaluation. The GT-CATS evaluation, like the ACTIN evaluation, uses GT-CATS' approach to tracking operator activities as the basis for judging its validity. GT-CATS predicts and explains actions correctly when it identifies the task and mode selection that the action supports. However, the experimental context of mode usage in the glass cockpit affords the unique opportunity to verify that a pilot action supports a task associated with a hypothesized mode selection by examining the state of the automation: a pilot action is known to support a valid mode selection if the control automation is engaged in that mode. Thus, the mode structure of the automation defines correct and incorrect actions, and minimizes reliance on expert assessments and verbal protocols.

The GT-CATS activity tracking process gives rise to two sets of possible outcomes from which performance measures are derived. One set of outcomes results when an action is expected, the other when an actual operator action is detected. These outcomes, described in detail later in this chapter, serve as the basis for evaluating GT-CATS' performance, rather than the indirect measures of improved operator performance with an aiding system that uses the output of an intent inferencer, or the operator's perceived usefulness of the aid. Furthermore, expert assessment of GT-CATS' activity tracking outcomes is ancillary, because examination of the state of the automation reveals whether GT-CATS' expectations and explanations are valid.

GT-EFIRT

The evaluation was performed using a Boeing 757/767 part-task simulator, called the Georgia Tech Electronic Flight Instrument Research Testbed (GT-EFIRT). GT-EFIRT was developed as an experimental tool for examining pilot interactions with complex flight deck automation, and for studying advanced interfaces (ref. 95). GT-EFIRT includes the components of the Boeing 757/767 flight deck automation and displays that are important for aircraft maneuvering and navigation. Based on the configuration and inputs supplied by the user, GT-EFIRT's flight model accurately describes the resulting real-time behavior of the aircraft. All the components in GT-EFIRT were developed with the full capabilities found on the actual aircraft, with the exception of the FMS Control and Display Unit (CDU). GT-EFIRT's CDU has fully functional display capabilities, but the input processing required for the more complex functions was not developed. Nonetheless, once programmed, GT-EFIRT's FMS acts as the source of information used by the LNAV (Lateral Navigation) and VNAV (Vertical Navigation) autoflight modes, as in the real aircraft. Programmed waypoints are tracked in LNAV mode. Crossing restrictions at waypoints and speed/altitude restrictions are adhered to in VNAV under realistic conditions. If, for example, the information required to use VNAV mode to accomplish a particular flight goal is displayed on the appropriate page of the CDU, VNAV mode can be expected to perform realistically.

The GT-EFIRT display configuration used in the GT-CATS evaluation study is shown in Figure 79. GT-EFIRT runs on a Sun SparcStation10™ computer with three monitors. The left monitor contains reproductions of the primary flight instruments of the 757/767. These are the Attitude Director Indicator (ADI), altimeter, airspeed/mach indicator, and vertical speed indicator. The center monitor has the Mode Control Panel

(MCP), Horizontal Situation Indicator (HSI), HSI range selector, and the FMC CDU. On the right monitor are additional controls for flaps, gear, engine thrust settings, etc. Controls for the simulation are also included on the right monitor, including controls to select the flight scenario to be flown, and a window to display the current ATC clearance.

GT-EFIRT uses a mouse for operator control inputs. Controls with mouse inputs are located on the center and right monitors. MCP windows for setting altitude, airspeed, heading, and vertical speed have virtual knobs that use mouse "hot spots" to simulate turning the knob in a particular direction. Individual mouse clicks increment the set value; holding down the mouse button enables a large change in the set value. This control mechanism may require the operator to deliberately overshoot the set value, then correct it with an appropriate number of mouse clicks in the opposite direction. All other MCP switches operate in a conventional manner: clicking the mouse presses the button.
Overall, GT-EFIRT handles the majority of pilot inputs required to effectively use the automation found on the 757/767. The fidelity and realism of GT-EFIRT were exhaustively reviewed in the pilot-study phase of the GT-CATS evaluation. Early tests identified problems with displays and flight behavior in various modes. These problems were then corrected, and GT-EFIRT was reviewed by other pilots in the course of later tests. These subjects assessed the realism of GT-EFIRT as adequate for exploring mode mangagement behavior.

Subjects

Ten Boeing 757/767 type-rated line pilots from a major carrier volunteered to participate in the study. Each pilot was asked a short series of questions at the outset of the experimental session. The results of this survey are tabulated in table 6. Two pilots were captains, eight were first officers. The mean number of years of reported experience on the 757/767 was 3.2 years (minimum one year; maximum five years). All but one had flown the 757/767 recently. All but two had extensive experience using a mouse as a computer input device (several owned computers).

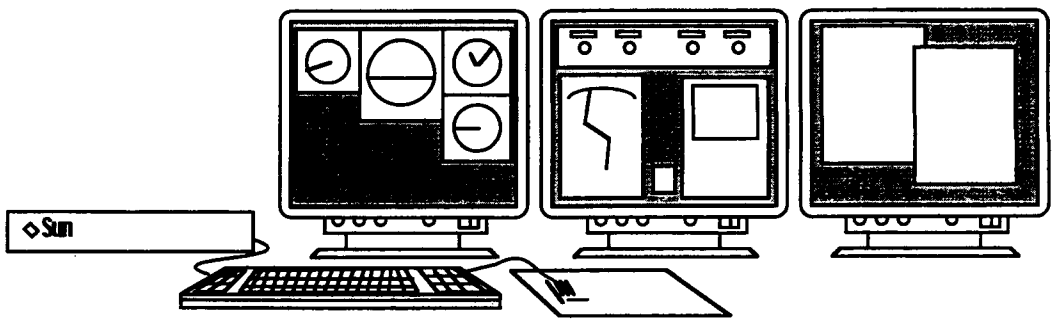


Figure 79. GT-EFIRT simulator hardware configuration.

Table 6. Results of initial subject survey.

Subject Number	Seat	757/767 Experience (years)	Transitioned From	Time Since Flown Type (days)	Computer mouse experience	Notes
1	2	1.0	737	2	some	
2	2	1.5	737	500	yes	now L1011 F/E
3	2	4.0	727	7	limited	
4	1	4.5	727	0	yes	had just come from flying
5	2	3.0	MD-88	1	yes	
6	1	5.0	737	1	yes	
7	2	3.5	727	2	yes	
8	2	3.5	727	45	yes	
9	2	4.0	737	3	yes	
10	2	2.0	MD-88	3	yes	

Experimental procedure

Each subject "flew" five experimental scenarios designed to elicit a range of autoflight system mode usage. Each subject participated in a single experimental session lasting approximately four and one half hours, during which they flew all five scenarios. After the initial survey, each subject was instructed in the operation of GT-EFIRT. As part of the orientation, the experimenter explained the features of GT-EFIRT and led the subjects through a real-time training scenario. The training scenario enabled subject pilots to fly GT-EFIRT in the same manner as they would the five experimental scenarios. The experimenter answered any questions regarding the operation of the GT-EFIRT interface, or the performance characteristics of GT-EFIRT's autoflight modes, during the orientation phase.

Following the orientation, subjects flew each experimental scenario with GT-CATS tracking their activities in real-time. Each scenario was recorded on audio/video tape; pilots were asked to verbalize their activities to the extent necessary to indicate why they performed a particular activity in cases where it was not obvious. After the subjects completed the five scenarios, a post-questionnaire was administered to assess the perceived realism of the scenarios and GT-EFIRT performance.

Experimental scenarios

The GT-CATS evaluation required a set of experimental scenarios. The scenarios are designed to elicit a range of mode usage, in order to provide insight into the ways in which pilots use the available automation to navigate the aircraft. Each scenario is defined by a preflight plan that is preprogrammed into GT-EFIRT's FMS before the flight, and a set of ATC clearances to be issued during the course of flight. Each scenario begins with a clearance issued while on the ground, proceeds through takeoff, climb, cruise, and descent, and ends when the final approach clearance is issued. Each clearance in a scenario is triggered, in order, as the aircraft reaches a particular point in the flight.

The clearances are designed to require the subject pilots to make full use of the available automation. ATC clearances are worded in a standard manner that clearly identifies the flight path required for compliance. Both vertical and lateral clearances utilize a common set of verbs, modifiers, and state variable values

to represent the required flight path (ref. 96). Simple clearances specify modification to a single aspect of the flight path, while complex clearances may dictate changes in several aspects of the flight path. The clearances used in the experimental scenarios combine a subset of the clearances identified by Wagner and Curry. Clearances that require CDU manipulations for compliance are not used. To ensure that the clearances used in the scenarios reflect real-world ATC interventions, the scenarios were exhaustively reviewed by pilots from a major carrier, and by the pilots who participated in the GT-CATS pilot study.

The selection of the origin and destination airports for each scenario, and the flight path prescribed by the clearances, was driven by several factors. One factor concerned a requirement of the GT-EFIRT simulator to have terrain maps of the origin and destination airports, in order to provide realistic radio altimeter readings on the ADI, and to support interface studies (ref. 96). Airports with worthy terrain representations were therefore used to construct the scenarios used in the GT-CATS evaluation. A second factor that impacted the choice of origin and destination airports was the length of the scenarios. It was necessary to create scenarios that could all be flown in a reasonable amount of time. Although the scenarios are designed to "fast-forward" through the period of inactivity in the middle of the cruise phase of flight, a short flight has a lower cruise altitude than a long flight, which in turn yields shorter climb and descent phases. A third factor that affected the choice of scenarios was the importance of including crossing restrictions, in order to elicit a range of mode manipulations. Crossing restrictions are commonly found in published Standard Instrument Departures (SIDs) and Standard Arrival Routes (STARs). The GT-CATS scenarios therefore incorporate SIDs and STARs commonly used for the selected airports. Actual traffic conditions were exaggerated in light traffic areas (such as Birmingham), in order to create a need for ATC interventions.

The following subsections describe the five scenarios developed for evaluating GT-CATS. Each scenario is tabulated to show the conditions that trigger a particular clearance, the clearance itself, and the expected mode(s) the pilot will employ to comply with clearance. Expected modes are based on the conditions for expecting a given mode selection in the OFM-ACM, except in cases noted with an asterisk. Where an asterisk appears, the OFM-ACM is conditioned such that a higher level of automation is expected (e.g., FL CH* indicates that VNAV mode is expected according to the OFM-ACM because the programmed vertical profile is appropriate for complying with the clearance, but there is a strong likelihood of pilots using FL CH due to the low altitude and expected future clearances; FL CH alone indicates that FL CH is the expected mode). Thus, an asterisk indicates cases where GT-CATS is likely to apply the revision process. Each scenario is also depicted graphically to show the flight path prescribed by the scenario clearances.

Scenario 1: KATL-KBHM

Scenario 1 is a flight from Atlanta (KATL) to Birmingham (KBHM). It is tabulated in table 7 and depicted graphically in figure 80. In this scenario pilots GT-CATS expects pilots to use high-level automation (i.e., LNAV/VNAV modes) until the final stages of descent. During climb, however, it is likely that pilots will choose a lower level of automation that GT-CATS must use the revision process to explain. The sixth clearance in the scenario is designed to elicit use of the VNAV speed intervention submode.

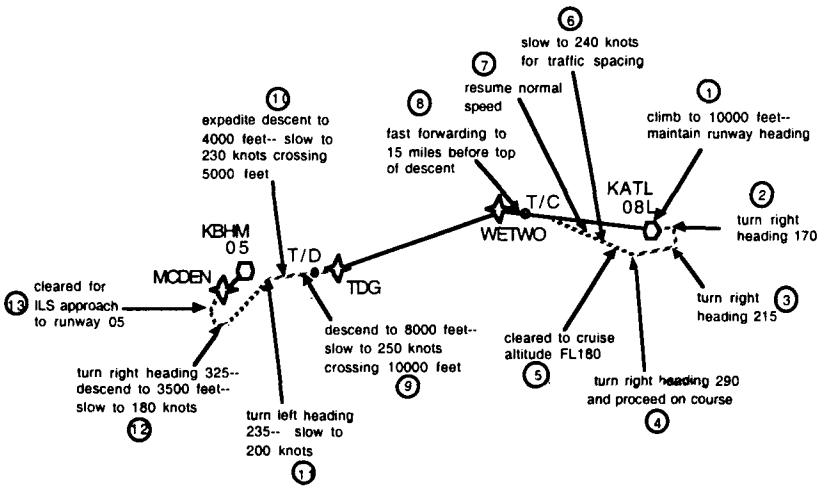


Figure 80. Scenario 1: KATL-KBHM.

Table 7. Scenario 1: KATL-KBHM.

Clearance	Trigger	Expected Mode Usage
1. climb to 10000 feet-- maintain runway heading	on ground	TO, HDG HOLD, TO
2. turn right heading 170	altitude above 2000 feet	FL CH*, HDG SEL, SPD
3. turn right heading 215	altitude above 4200 feet and heading past 168	FL CH*, HDG SEL, SPD
4. turn right heading 290 and proceed on course	altitude above 6900 feet	FL CH*, HDG SEL, SPD
5. cleared to FL180	altitude above 9950 feet	VNAV SPD, LNAV, SPD
6. slow to 240 knots for traffic spacing	altitude above 11300 feet	LNAV, VNAV speed intervention
7. resume normal speed	speed below 241 knots	LNAV, VNAV SPD
8. fast forwarding to 15 miles before top of descent	top of climb passed and on LNAV track	LNAV, VNAV PTH
9. descend to 8000 feet-- slow to 250 knots crossing 10000 feet	top of descent passed	LNAV, VNAV PTH
10. expedite descent to 4000 feet-- slow to 230 knots crossing 5000 feet	altitude below 8500 feet	LNAV, FL CH, SPD
11. turn left heading 235-- slow to 200 knots	altitude below 5000 feet	HDG SEL, FL CH, SPD
12. turn right heading 325-- descend to 3500 feet-- slow to 180 knots	altitude below 4100 feet	HDG SEL, FL CH, SPD
13. cleared for the ILS approach to runway 05	altitude below 4010 feet and speed below 201 knots	NA

Scenario 2: KATL-KBHM1

Scenario 2 is also a flight from Atlanta (KATL) to Birmingham (KBHM). It is tabulated in table 8 and depicted graphically in figure 81. Scenario 2 prescribes a different route, but differs from scenario 1 principally in that the descent is interrupted by clearance to "stop descent for crossing traffic," and the aircraft must leave the LNAV route as dictated by clearance 6. Again the climb phase presents several situations where GT-CATS may apply the revision process, if pilots choose against VNAV mode.

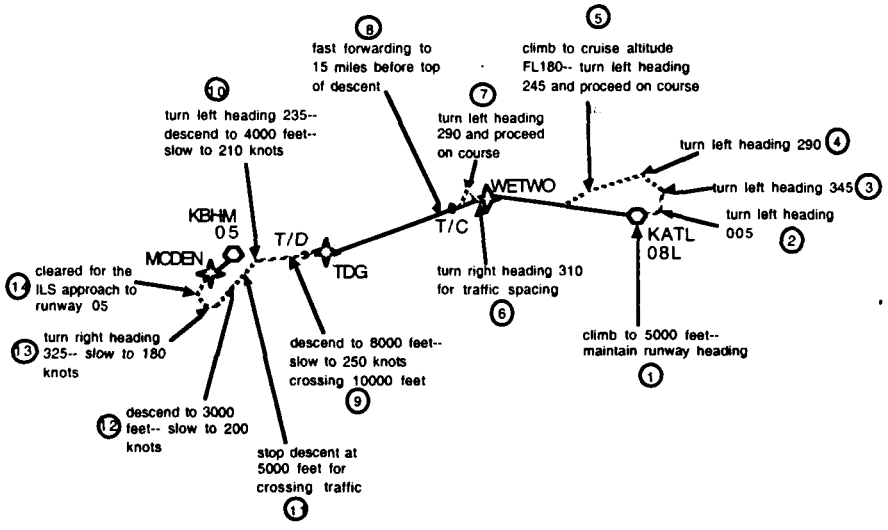


Figure 81. Scenario 2: KATL-KBHM1.

Table 8. Scenario 2: KATL-KBHM1.

Clearance	Trigger	Expected Mode Usage
1. climb to 5000 feet-- maintain runway heading	on ground	TO, HDG HOLD, TO
2. turn left heading 005	altitude above 2200 feet	FL CH*, HDG SEL, SPD
3. turn left heading 345	altitude above 3550 feet	FL CH*, HDG SEL, SPD
4. turn left heading 290-- climb to 10000 feet	altitude above 4050 feet and heading past 344	FL CH*, HDG SEL, SPD
5. climb to cruise altitude FL180-- turn left heading 245 and proceed on course	heading past 292	VNAV SPD, HDG SEL, SPD
6. turn right heading 310 for traffic spacing	altitude above 12000 feet	HDG SEL, VNAV SPD
7. turn left heading 250 and proceed on course	heading past 309	HDG SEL, VNAV SPD
8. fast forwarding to 15 miles before top of descent	top of climb passed and on LNAV track	LNAV, VNAV PTH
9. descend to 8000 feet-- slow to 250 knots crossing 10000 feet	less than 7000 feet to top of descent	LNAV, VNAV PTH
10. turn left heading 235-- to 4000 feet-- slow to 210 knots	altitude below 9700 feet	HDG SEL, FL CH, SPD
11. stop descent at 5000 feet for crossing traffic	altitude below 7700 feet	HDG SEL, FL CH, SPD
12. descend to 3000 feet-- slow to 200 knots	altitude below 5100 feet	HDG SEL, FL CH, SPD
13. turn right heading 325-- slow to 180 knots	altitude below 4200 feet	HDG SEL, FL CH, SPD
14. cleared for approach to runway 05	altitude below 3010 feet	NA

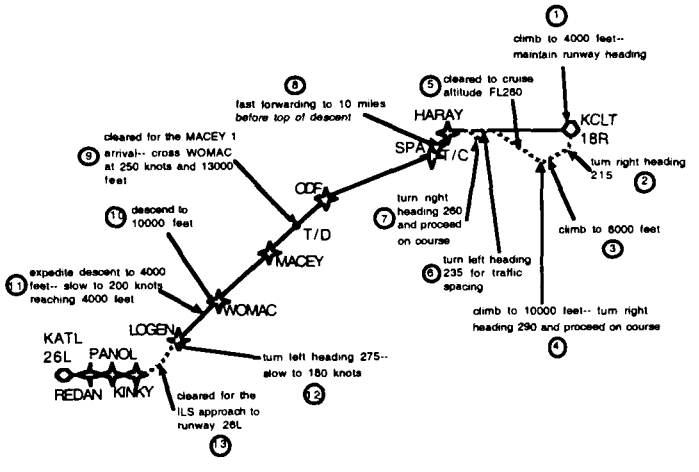


Figure 82. Scenario 3: KCLT-KATL.

Table 9. Scenario 3: KCLT-KATL.

Clearance	Trigger	Expected Mode Usage
1. climb and 4000 feet-- maintain runway heading	on ground	TO, HDG HOLD, TO
2. turn right heading 215	altitude above 2350 feet	FL CH*, HDG SEL, SPD
3. climb to 6000 feet	altitude above 3595 feet and speed above 175 knots	FL CH*, HDG SEL, SPD
4. climb to 10000 feet-- turn right heading 290 and proceed on course	speed above 210 knots	FL CH*, HDG SEL, SPD
5. cleared to cruise altitude FL260	altitude above 9250 feet	LNAV, VNAV SPD, SPD
6. turn left heading 235 for traffic spacing	altitude above 17500 feet	HDG SEL, VNAV SPD, SPD
7. turn right heading 265 and proceed on course	altitude above 20050 feet	HDG SEL, VNAV SPD, SPD
8. fast forwarding to 10 miles before top of descent	altitude above 25995 and past SPA	LNAV, VNAV PTH
9. cleared for the MACEY 1 arrival-- cross WOMAC at 250 knots and 13000 feet	less than 7000 feet to top of descent	LNAV, VNAV PTH
10. descend to 10000 feet	altitude below 13500 feet	FL CH, LNAV, SPD
11. expedite descent to 4000 feet-- slow to 200 knots reaching 4000 feet	altitude below 10500 feet	FL CH, HDG SEL, SPD
12. turn right heading 275-- slow to 180 knots	altitude below 4200 feet	FL CH, HDG SEL, SPD
13. cleared for the approach to runway 26L	altitude below 4050 feet	NA

Scenario 3: KCLT-KATL

Scenario 3 is a flight from Charlotte (KCLT) to Atlanta (KATL). It is tabulated in table 9 and depicted graphically in figure 82. The primary distinguishing features of this scenario are the unusually low 4000 feet clearance on takeoff, and the inclusion of a standard arrival route (STAR), as required by clearance 9. Pilots are expected to use VNAV to comply with this clearance, as the crossing restriction is programmed in the FMS. The clearance to expedite the descent, then slow (clearance 11), is somewhat unusual, albeit not unheard of.

Scenario 4: KCLT-KATL1

Scenario 4 is also a flight from Charlotte (KCLT) to Atlanta (KATL). It is tabulated in table 10 and depicted graphically in figure 83. Scenario 4 includes a crossing restriction that must be adhered to at waypoint GAFFE during the climb phase which, like scenario 3, begins with a clearance to 4000 feet. GT-CATS expects pilots to use VNAV to meet the crossing restriction because the restriction is programmed in the FMS. Scenario 4 also includes the MACEY 1 STAR, and a series of short descents with speed adjustments.

Table 10. Scenario 4: KCLT-KATL1.

Clearance	Trigger	Expected Mode Usage
1. climb and 4000 feet-- maintain runway heading	on ground	TO, HDG HOLD
2. turn right heading 215	altitude above 2350 feet	HDG SEL, FL CH*, SPD
3. turn right heading 280 and proceed on course-- cross GAFFE at 240 knots and 8000 feet	altitude above 3595 feet and speed above 175 knots	HDG SEL, VNAV SPD, SPD
4. cleared to cruise altitude FL260-- resume normal speed	past GAFFE	LNAV, VNAV SPD, SPD
5. turn left heading 235 for traffic spacing	altitude above 8750 feet	HDG SEL, VNAV SPD, SPD
6. turn right heading 255 and proceed on course	altitude above 10000 feet	HDG SEL, VNAV SPD, SPD
7. fast forwarding to 10 miles before top of descent	altitude above 25995 and top of climb passed and on LNAV track	LNAV, VNAV PTH
8. cleared for the MACEY 1 arrival-- cross WOMAC at 13000 feet and 250 knots	less than 7000 feet to top of descent	LNAV, VNAV PTH
9. descend to 6000 feet-- slow to 220 knots reaching 6000 feet	altitude below 13500	LNAV, FL CH, SPD
10. turn left heading 235-- descend to 4000 feet-- slow to 200 knots	speed below 221 knots	HDG SEL, FL CH, SPD
11. cleared for approach to runway 26L	altitude below 4010 feet	NA

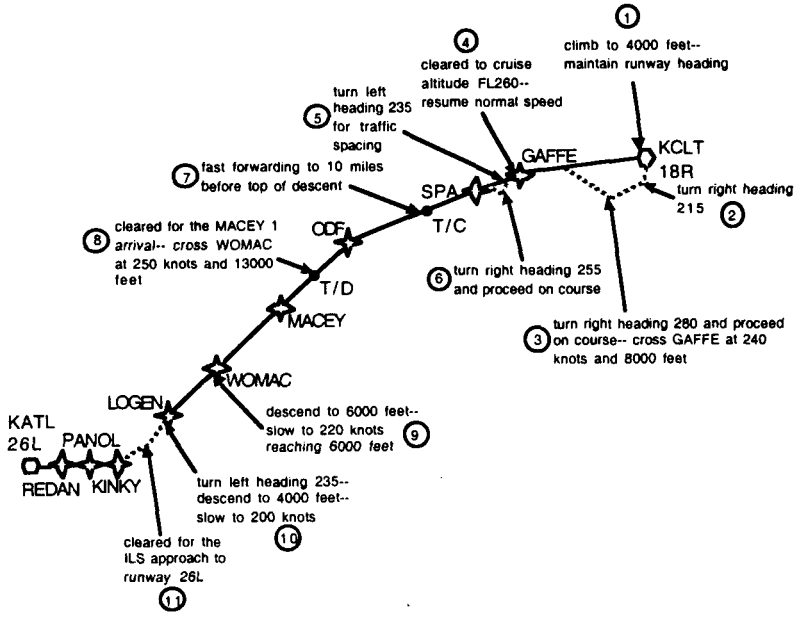


Figure 83. Scenario 4: KCLT-KATL1.

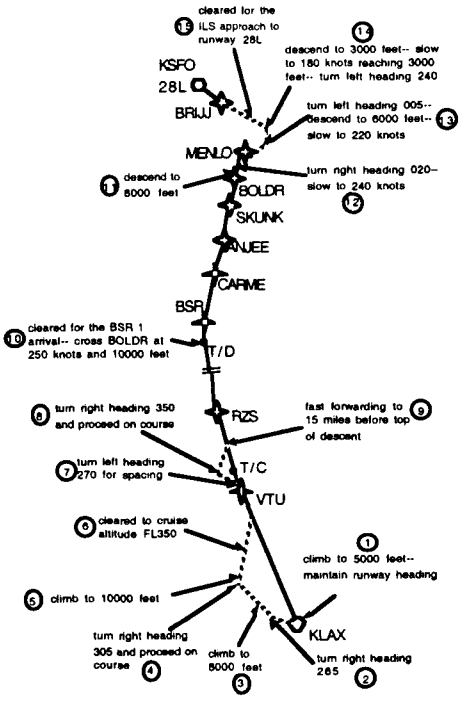


Figure 84. Scenario 5: KLAX-KSFO.

Scenario 5: KLAX-KSFO

Scenario 5 is a flight from Los Angeles (KLAX) to San Francisco (KSFO). It is tabulated in table 11 and depicted graphically in figure 84. This scenario was constructed specifically to mimic a scenario studied in the NASA Ames ACFS flight simulator. It does so to the extent that the mode manipulations required to comply with the ATC clearances used can be successfully executed on GT-EFIRT.

Table 11. Scenario 5: KLAX-KSFO.

Clearance	Trigger	Expected Mode Usage
1. climb to 5000 feet-- maintain runway heading	on ground	TO, HDG HOLD, TO
2. turn right heading 265	altitude above 2000 feet	FL CH*, HDG SEL, SPD
3. climb to 8000 feet	altitude above 4600 feet	FL CH*, HDG SEL, SPD
4. turn right heading 315 and proceed on course	altitude above 6500 feet	FL CH*, HDG SEL, SPD
5. climb to 10000 feet	altitude above 7850 feet	FL CH*, HDG SEL, SPD
6. cleared to cruise altitude FL350	altitude above 9950 feet	LNAV, VNAV SPD, SPD
7. turn left heading 270 for spacing	past VTU and altitude above 20000 feet	HDG SEL, VNAV SPD, SPD
8. turn right heading 350 and proceed on course	altitude above 24500 feet	HDG SEL, VNAV SPD, SPD
9. fast forwarding to 5 miles before top of descent	past RDG	LNAV, VNAV PTH
10. cleared for BSR 1 arrival-- cross BOLDR at 250 knots and 10000 feet	less than 6000 feet to top of descent	LNAV, VNAV PTH
11. descend to 8000 feet	altitude below 10500 feet	FL CH, LNAV, SPD
12. turn right heading 020-- slow to 240 knots	altitude below 8300 feet	FL CH, HDG SEL, SPD
13. turn left heading 005-- descend to 6000 feet-- slow to 220 knots	speed below 241 knots	FL CH, HDG SEL, SPD
14. descend to 3000 feet-- slow to 180 knots reaching 3000 feet-- turn left heading 240	altitude below 6050 feet	FL CH, HDG SEL, SPD
15. cleared for the ILS approach to runway 28L	altitude below 3050 feet	NA

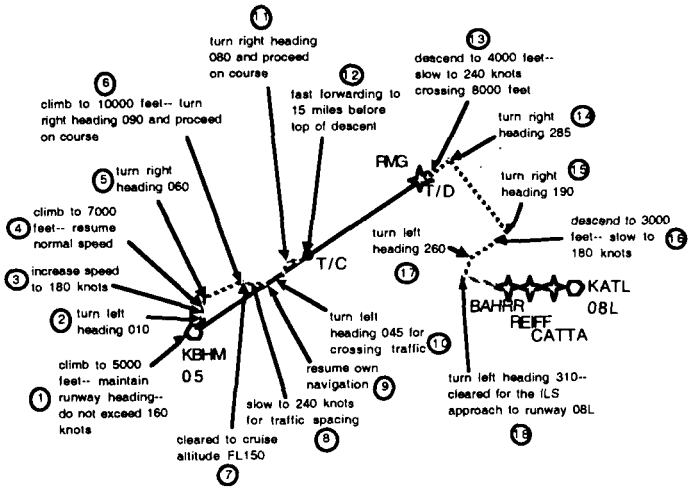


Figure 85. Scenario 6: KBHM-KATL.

Table 12. Scenario 6: KBHM-KATL.

Clearance	Trigger	Expected Mode Usage
1. climb to 5000 feet--maintain runway heading-- do not exceed 160 knots	on ground	TO, HDG HOLD, TO
2. turn left heading 010	altitude above 1500 feet	FL CH*, HDG SEL, SPD
3. increase speed to 180 knots	heading past 012	FL CH*, HDG SEL, SPD
4. climb to 7000 feet-- resume normal speed	speed above 178 knots	FL CH*, HDG SEL, SPD
5. turn right heading 060	altitude above 5500 feet	FL CH*, HDG SEL, SPD
6. climb to 10000 feet--turn right heading 090 and proceed on course	altitude above 6500 feet	FL CH*, HDG SEL, SPD
7. cleared to cruise altitude FL150	altitude above 9200 feet	LNAV, VNAV SPD, SPD
8. slow to 240 knots for traffic spacing	altitude above 10500	LNAV, VNAV speed intervention, SPD
9. resume own navigation	speed less than 241 knots	LNAV, VNAV SPD, SPD
10. turn left heading 045 for crossing traffic	altitude above 13200 feet	HDG SEL, VNAV SPD, SPD
11. turn right heading 080 and proceed on course	heading past 046	HDG SEL, VNAV SPD, SPD
12. fast forwarding to 15 miles before top of descent	top of climb passed and heading between 067 and 065	LNAV, VNAV PTH
13. descend to 4000 feet-- slow to 240 knots crossing 8000 feet	top of descent passed	LNAV, VNAV PTH
14. turn right heading 285	altitude below 5700 feet	HDG SEL, VNAV PTH
15. turn right heading 190	altitude below 4100 feet	HDG SEL, VNAV PTH
16. descend to 3000 feet-- slow to 180 knots	heading past 192	FL CH, HDG SEL, SPD
17. turn left heading 260	altitude below 3100 feet	FL CH, HDG SEL, SPD
18. turn left heading 310-- cleared for the ILS approach to runway 08L	heading past 258	NA

Scenario 6: KBHM-KATL

Scenario 6 is the orientation scenario that each pilot flies prior to the five experimental scenarios. This scenario includes multiple clearances designed to demonstrate GT-EFIRT's flight characteristics. By eliciting a range of mode selections, and requiring several speed adjustments, scenario 6 affords pilots the opportunity to "get a feel" for GT-EFIRT's response to such inputs. Climb rates in various modes, acceleration/deceleration rates, and capture profiles for GT-EFIRT are all readily ascertainable during this scenario.

GT-CATS ATC facility

Because the experimental scenarios depend on the successful execution of each clearance in order to arrive at the condition required to trigger the next, an ATC facility was developed. The ATC facility acts as a stop-gap measure to ensure that pilot errors or timing of mode selections do not stymie data collection during a scenario. In the event that a pilot errs in complying with a scenario clearance, or times compliance such that a triggering event is missed, the ATC facility is used to issue a clearance that redirects the pilot into a position such that downpath clearances are triggered normally.

Any actions performed to comply with a clearance issued through the ATC facility are still interpreted by GT-CATS, because the LOE update necessary for GT-CATS to understand the actions is still performed. This is done by allowing the experimenter to input, first, the required LOE modifications, and second, the text of the clearance.

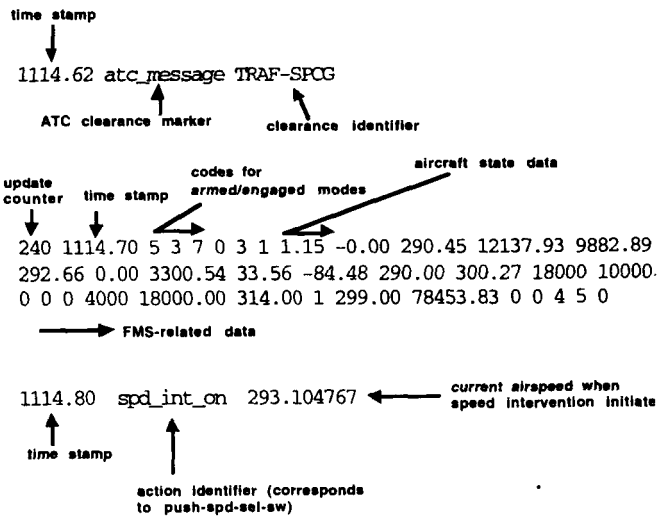


Figure 86. Example GT-EFIRT output data.

Data collection

Data are collected in files output by both GT-EFIRT and GT-CATS. The GT-EFIRT output files include state space data (output every five seconds), time-stamped ATC clearance data, and time-stamped operator actions. Examples of GT-EFIRT output data are shown in figure 86. The first data line in figure 86 shows how ATC clearances are logged. The second data line shows a state-space-data update. The third line shows how pilot actions are logged.

A GT-CATS output file contains time-stamped data for each determination GT-CATS makes. Examples of GT-CATS output file data are annotated in figures 87 and 88. These data show when each expectation was generated, when each action was interpreted, and the results. These data also include the current state of DUO (i.e., all the activities that are active) at the time when GT-CATS interprets an action. The fifteen types of pilot actions that GT-CATS interprets are shown in table 13.

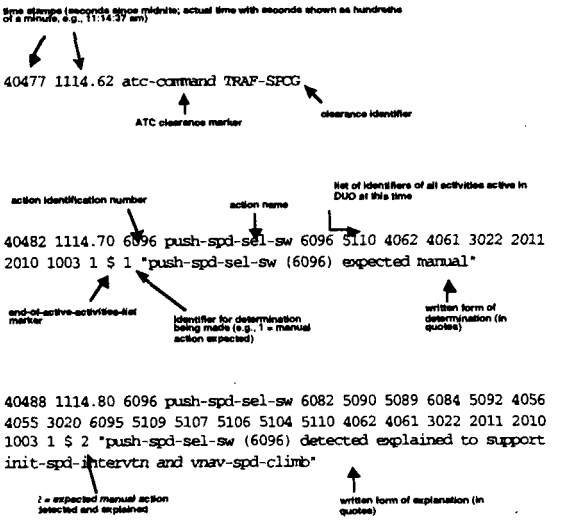


Figure 87. Example GT-CATS output data.

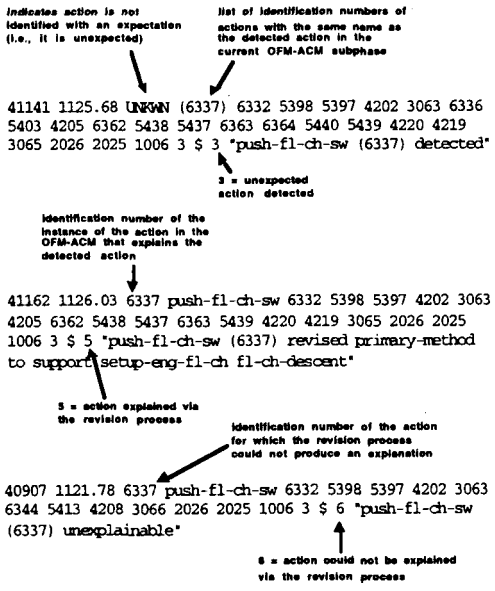


Figure 88. More example GT-CATS output data.

Table 13. Detectable actions in GT-CATS.

1. push-tsp-sw (push thrust select panel switch)
2. push-ap-cmd-mode-sw (push Autopilot Command mode switch)
3. push-alt-hold-sw (push Altitude Hold switch)
4. push-hdg-sel-sw (push Heading Select switch)
5. dial-mcp-hdg (dial MCP heading)
6. push-hdg-hold-sw (push Heading Hold switch)
7. push-lnav-sw (push Lateral Navigation switch)
8. push-vs-sw (push Vertical Speed switch)
9. dial-mcp-vs (dial MCP vertical speed)
10. push-mcp-spd-sw (push MCP Speed switch)
11. push-fl-ch-sw (push Flight Level Change switch)
12. push-vnav-sw (push Vertical Navigation switch)
13. dial-mcp-ias (dial MCP indicated airspeed)
14. push-spd-sel-sw (push Speed select switch)
15. dial-mcp-alt (dial MCP altitude)

Experimental configuration

Figure 89 depicts the experimental configuration used in the evaluation. The experimenter acted as Air Traffic Control, and monitored the operation of GT-CATS and GT-EFIRT. The subject pilot's activities were audio and video recorded. Data were recorded via computer; GT-EFIRT simulator data were recorded on one SparcStation and GT-CATS output data were recorded on the other. As noted above, GT-EFIRT data include the values of the relevant simulator state data recorded and time-stamped every five seconds, along with time-stamped ATC clearances. GT-CATS output data were the hypothesized operator actions, detected actions, and explanations for actions, all time-stamped with the time they were issued. GT-CATS data also included the time-stamped ATC clearances and entries made by the experimenter indicating whether or not explanations produced by GT-CATS were correct, insofar as they accurately described the mode that a given pilot action supported.

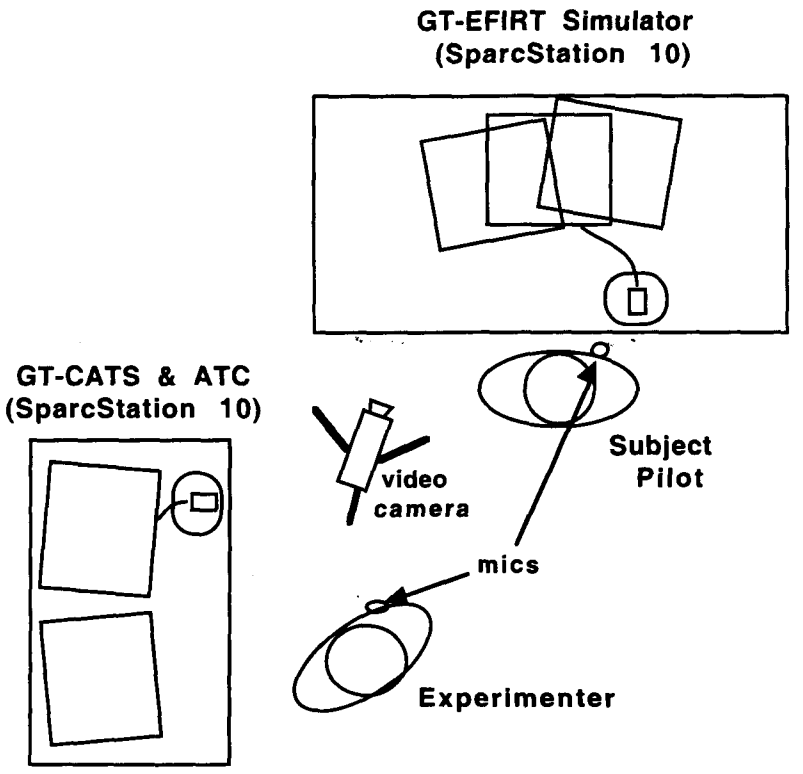


Figure 89. Experimental setup for the GT-CATS evaluation.

Performance measures

The GT-CATS evaluation method seeks to determine the extent to which GT-CATS adequately "tracks" operator actions. Several measures are important for assessing the effectiveness of GT-CATS for understanding the activities of pilots navigating glass cockpit aircraft. Generally, these measures reflect GT-CATS' capability to expect pilot actions, and its capability to explain detected pilot actions. The GT-CATS activity tracking process gives rise to two sets of possible outcomes from which the measures are derived. One set of outcomes results when an action is expected. Another set of outcomes, related to the first, results when an actual operator action is detected.

Two sets of outcomes that define the measures used in the GT-CATS evaluation are shown in figure 90. The top of figure 90 depicts the outcomes that are possible when an action is expected. When the action becomes active in DUO, it is expected. It can then be flagged as late (possibly missed) or not. An action determined to be late can later be detected (if the pilot was indeed slow in performing it), in which case GT-CATS either explains it correctly (letter "A" in figure 90), or explains it incorrectly (letter "B" in figure 90). It can also go undetected if the pilot never performs it, or the situation changes such that it is no longer expected (letter "C"). Actions that are not flagged late can also go undetected if the situation changes such that they are no longer expected (letter "F"); if they are detected, they may be either explained correctly (letter "D") or explained incorrectly (letter "E").

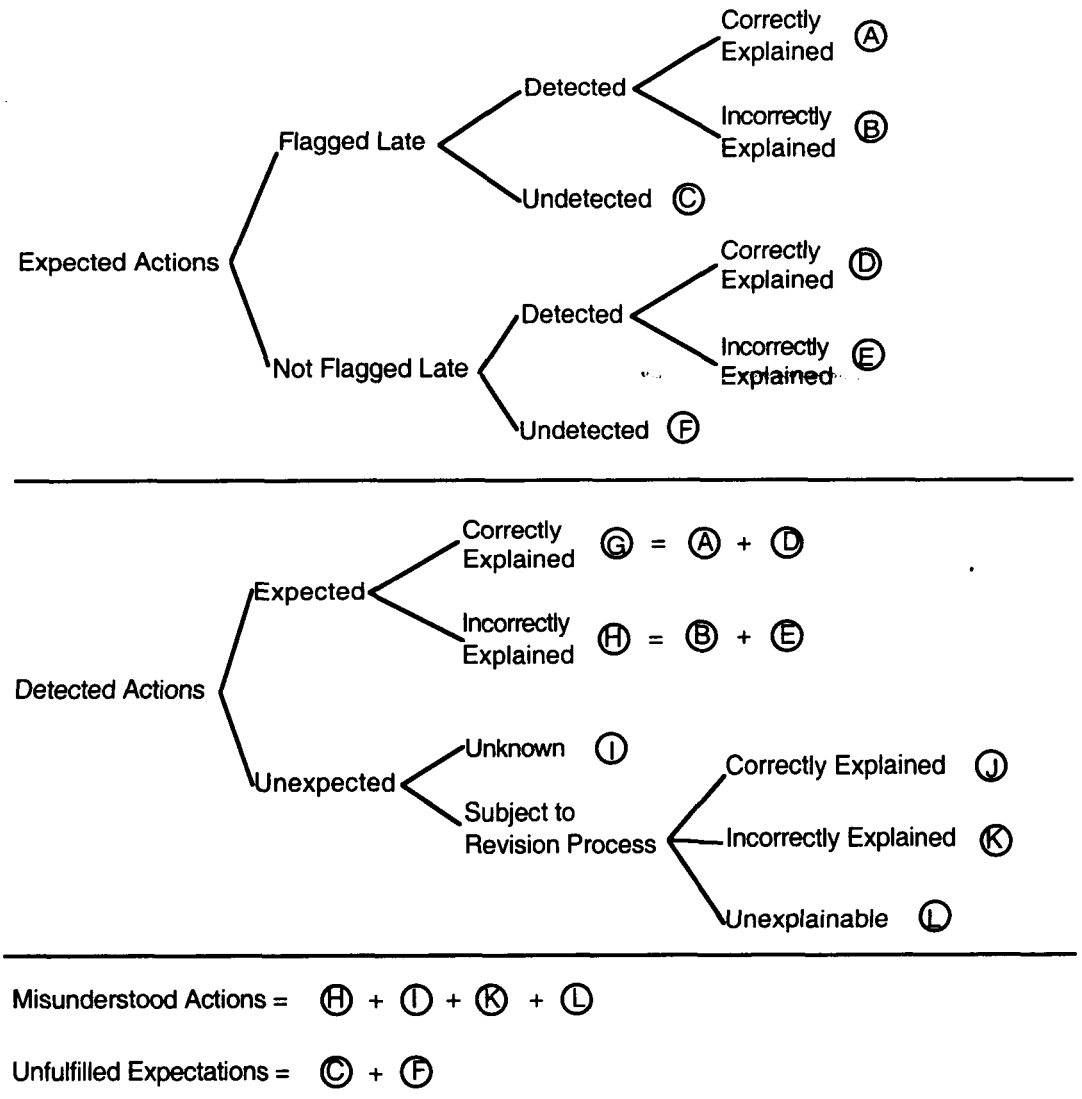


Figure 90. Possible outcomes from the GT-CATS activity tracking process.

The middle portion of figure 90 depicts the set of outcomes possible for a detected action. A detected action may be either expected or unexpected. When an expected action is detected, it may be either correctly explained (letter "G"), or incorrectly explained (letter "H"). As indicated in figure 90, this branch of outcomes essentially ignores whether the action was flagged late or not prior to being detected, so that "G" is the sum of "A" and "D," and "H" is the sum of "B" and "E". The definition of a correct explanation is one in which GT-CATS explains the action to support the task and valid mode selection that the pilot actually selects. Any other explanation is incorrect. In other words, only when GT-CATS explains an action to support a valid mode selection, and only when the valid mode selection is the one that the pilot actually chooses, does GT-CATS correctly explain the action. As an example of an expected action that is incorrectly explained, consider a situation where the pilot is expected to set the MCP altitude to setup/engage VNAV, and the pilot indeed sets the altitude, so that the action is explained to support the expectation. But, then the pilot chooses FL CH mode instead. The fact that FL CH mode was chosen instead of VNAV invalidates the explanation that the altitude was set in support of VNAV mode.

This definition is unambiguous, because the mode that is engaged in the controlled system is inspectable; a difference in the mode selection used as part of the explanation as compared to the valid mode engaged in the aircraft indicates that the explanation is incorrect.

The middle portion of figure 90 also shows the set of outcomes that can arise when GT-CATS does not expect a detected pilot action. If an action was not expected, then GT-CATS either cannot find an instance of it in the currently active subphase, in which case the action is unknown (letter "I" in figure 90), or GT-CATS subjects the action to the revision process. Upon application of the revision process, the action may either be correctly explained (letter "J"), incorrectly explained (letter "K"), or the revision process may fail to generate an explanation for the action (letter "L"). Again, the explanation attained through the revision process is correct only if the action supports a mode selection that is engaged or becomes engaged as a result of the action.

As noted above, figure 90 shows how the outcomes possible when GT-CATS generates an expectation are related to the outcomes possible when an action is detected. When an action is detected, it is either correctly explained or incorrectly explained, regardless of whether GT-CATS flagged it late prior to detecting it. Thus, the sum of correctly explained actions flagged late, and those not flagged late is the number of correctly explained detected actions.

Figure 90 also provides insight into what it means when an action is misunderstood by GT-CATS. As shown at the bottom of figure 90, misunderstood actions are those that are incorrectly explained, unknown with reference to the current subphase of the OFM-ACM active in DUO, incorrectly explained through the revision process, or unexplainable through the revision process ("H" + "I" + "K" + "L"). Note that actions that are not explainable via the revision process ("L") may in fact be operator errors, so GT-CATS is in fact correct in not understanding them. Additional analysis of such actions is discussed in detail in the next chapter. Actions that GT-CATS expects but never detects, whether flagged as late or not, are unfulfilled expectations ("C" + "F"). To find out whether such expectations went unfulfilled as a result of errors of omission, additional analysis (also discussed in the next chapter) is needed.

Collectively, these outcomes serve to define a set of measures for assessing GT-CATS' activity tracking capabilities. By examining data to determine how it breaks down into these categories, the overall effectiveness of GT-CATS can be quantified. In addition to the quantitative measures, subjective indications of the realism, reasonableness, and representativeness of the GT-EFIRT simulator and the experimental scenarios were also collected via a post-questionnaire, as described in the next chapter. These data are important for establishing that GT-CATS activity tracking process was evaluated in a realistic supervisory control environment. Overall, the GT-CATS evaluation sought to show that a majority of pilot actions were understood, and that, although expectations are correct a majority of the time, the GT-CATS revision process makes an important contribution to explaining pilot actions.

Summary

The GT-CATS evaluation sought to assess how well GT-CATS expects and explains pilot actions. Ten type-rated pilots from a major carrier each flew five experimental scenarios. Data from the GT-EFIRT simulator and GT-CATS were collected via computer, and the experimental sessions were audio- and videotaped. In addition to demonstrating GT-CATS' effectiveness in tracking pilot activities, the evaluation was expected to show the effectiveness of the revision process in explaining unexpected pilot actions.

7. Results

Introduction

Effective activity tracking entails successful prediction and explanation of operator activities in real time. The results of the GT-CATS empirical evaluation presented in this chapter describe in detail GT-CATS' activity tracking performance. The evaluation is based on the data and assessments defined in figure 90. GT-CATS "understands" an action when it explains it to support a task that, in turn, supports a pilot's mode selection. GT-CATS either explains actions using prior expectations, or through the revision process.

On the other hand, GT-CATS "misunderstands" actions in four ways. First, the revision process may fail to explain the action (this is the desired outcome if the action is in error). Second, GT-CATS may be unable to locate the action in the currently active subphase of the OFM-ACM. Third, an expectation may lead to an incorrect explanation, and, fourth, the revision process may explain an action incorrectly. GT-CATS explains actions incorrectly if it incorrectly associates a pilot action with a task in DUO that does not support the currently active mode.

This chapter first provides a macro-analysis of the data. The macro-analysis summarizes GT-CATS' predictive and explanatory performance. The above definitions are then used as the basis for a "micro-analysis" of the data. The purpose of the micro-analysis is to categorize each misunderstanding on the part of GT-CATS. For example, one facet of the micro-analysis identifies actions that are in fact errors. These data are in turn useful in identifying enhancements to GT-CATS that address specific classes of misunderstandings. This chapter presents enhancements to the GT-CATS OFM-ACM and processing scheme to remediate several classes of misunderstandings identified in the micro-analysis.
The chapter then describes the results of the questionnaire given pilots to assess the realism, reasonableness, and representiveness of the evaluation scenarios and the GT-EFIRT simulator. Finally, the chapter discusses differences in the number and types of actions pilots performed across scenarios. The overall results presented here are tabulated in Appendix B; data on detected pilot actions are graphed in Appendix C.

Overall results

Figure 91 shows the overall results of the GT-CATS empirical evaluation. The results indicate that, with minor adjustments, GT-CATS can correctly explain 94% of the 2,089 pilot actions observed in the study. GT- CATS expected and correctly explained 51% of pilot actions. It successfully applied the revision process to explain an additional 28% of pilot actions. Pilot 'errors' accounted for 2% of unexplained actions, minor adjustments would enable GT-CATS to explain 13%, and further research is necessary to explain 6%.

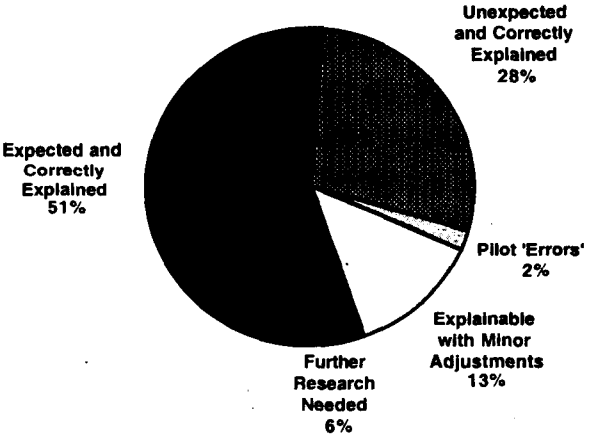


Figure 91. Overall results of the GT-CATS empirical evaluation.

Predictive capabilities of GT-CATS

Although GT-CATS expected more than half of pilot actions, the revision process was nonetheless instrumental in explaining 28% of pilot actions. Figure 92 shows the predictive performance of GT-CATS. Specifically, it shows the numbers of actions that GT-CATS expected and did not expect, and the number of expectations for pilot actions that pilots did not subsequently perform. The results indicate that GT-CATS expects more actions than not, and generates far fewer expectations that are not fulfilled.

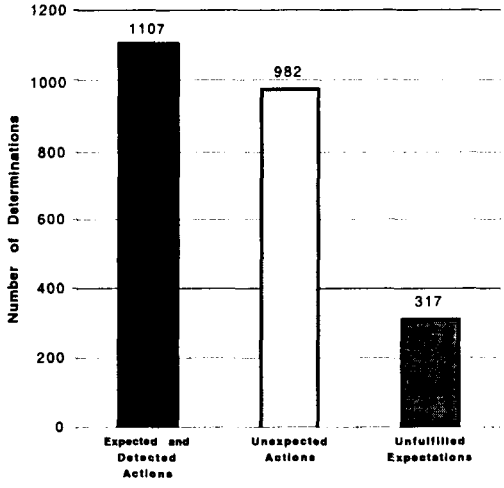


Figure 92. Predictive performance of GT-CATS.

Unfulfilled expectations are the result of either the situation changing, such that current expectations are by replaced by new ones, or pilots choosing a different mode than expected, such that they performed unexpected actions rather than meeting expectations. The results indicate that most of the time unexpected actions are "extra" actions that pilots perform when they transition between modes or adjust target values in situations where these actions are not required. Pilots apparently perform such actions seeking to exploit some perceived advantage of an alternative mode.

As figure 93 shows, GT-CATS' revision process facilitated correct explanations for a majority of action types. For two thirds of the action types, GT-CATS correctly explained at least 75% of pilot actions.

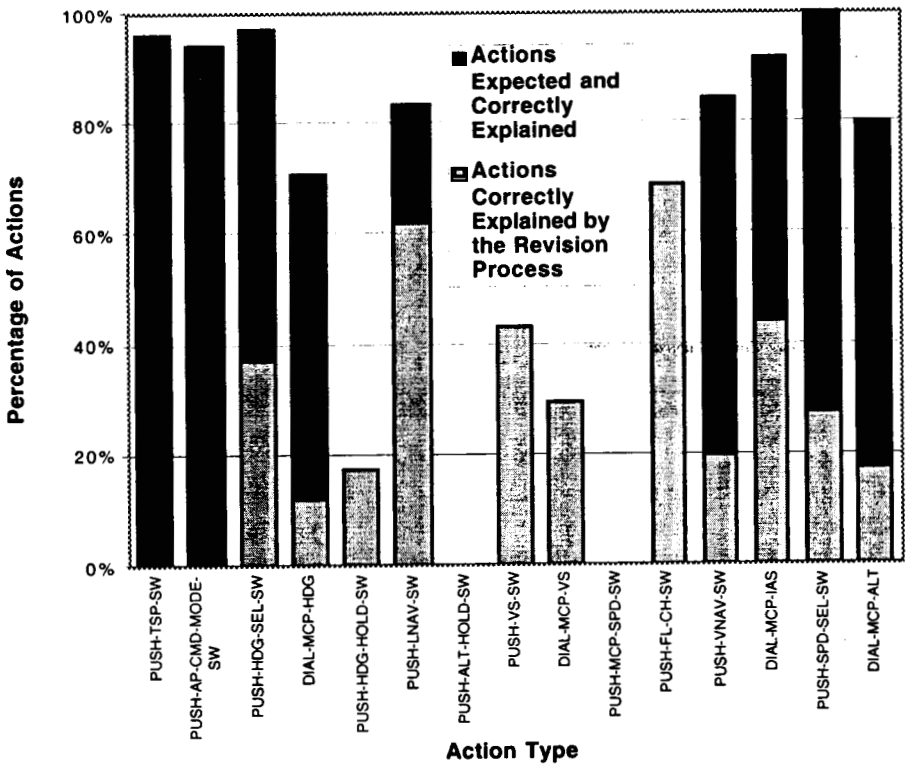


Figure 93. Correct explanations for expected and unexpected actions.

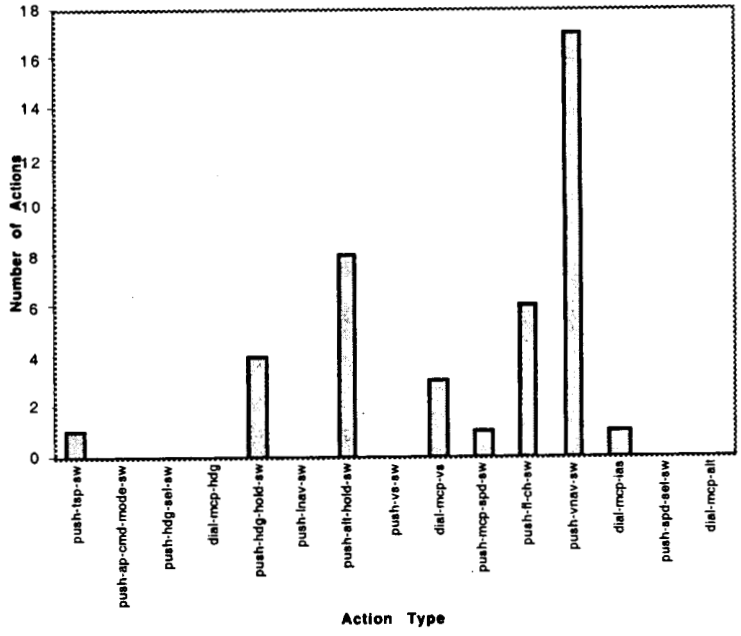


Figure 94. Pilot 'errors.'

Pilot 'errors'

The overall results show that 2% of actions were pilot 'errors.' In making this determination, it is a prerequisite to define what constitutes an error. First, the errors identified through micro-analysis are those actions that are either incorrect because they represent a procedural step performed out of order or an attempt to engage an invalid mode, or because the aircraft already transitioned to a new mode configuration in which the action was unnecessary. Thus, the errors included here are errors of commission. (Only one error of omission was recorded in the study: a failure to engage HDG HOLD before takeoff.) Furthermore, GT-CATS correctly explained 'errors' that involved setting target values on the MCP to have been set with the wrong value. Because these actions were, in fact, explained, they are not included here.

Figure 94 shows those actions identified as pilot 'errors,' although none can be considered threatening to flight safety. The greatest number of errors involved the push-vnav-sw action; the vast majority resulted from pilots attempting to engage VNAV as part of a procedure following autopilot CMD engagement, when in fact the aircraft had already transitioned to capture the cleared altitude. Some pilots did not notice the mode transition to ALT CAP, and attempted to engage VNAV anyway. Another prevalent error involved attempting to engage VNAV or FL CH without first setting a new altitude on the MCP. Other errors were actions inappropriate for the engaged mode, or pressing the wrong mode engagement switch.

Enhancements/adjustments to GT-CATS

Minor adjustments identified by the micro-analysis of outcomes would enable GT-CATS to explain all but 6% of pilot actions. Figure 95 shows the additional percentages of each action type explainable with adjustments.

Access to the next subphase

The first adjustment would enable GT-CATS to explain actions that are represented in subphases of the OFM-ACM other than the currently active subphase. If a pilot performs an action slightly before GT-CATS switches to the subphase in which the action is represented, GT-CATS cannot locate a corresponding action in DUO. Thus, as implemented, GT-CATS cannot explain the action. In the evaluation, pilots sometimes started configuring the autoflight system and adjusting target values before 1,000 feet AGL;these actions are not represented until the climb-to-3000-ft subphase (which becomes active at 1,000 feet AGL). This processing error can be corrected by allowing GT-CATS' action manager access to the subphase immediately following the current subphase.

OFM-ACM enhancements to explain heading adjustments

A second adjustment involves minor additions to the OFM-ACM to correct a model error. The additions enable GT-CATS to explain MCP heading adjustments pilots make during LNAV use, which are devoted to lining up the magenta line on the HSI that shows the MCP-selected heading with the LNAV route. Keeping the heading aligned with the LNAV route helps pilots monitor LNAV operation. These actions constitute a significant fraction (approximately 36%) of actions that GT-CATS did not explain; because no dial-mcp-hdg actions are modeled to support LNAV mode usage, GT-CATS revision process fails to explain these actions.

Figures 96 and 97 show two modifications to the OFM-ACM proposed to enable GT-CATS to explain MCP heading adjustments made during LNAV operation. Figure 96 shows the addition of a dial-mcp-hdg action to support monitoring LNAV turns; figure 97 shows the addition of a dial-mcp-hdg action to support monitoring the programmed LNAV route. Although unverified, these adjustments to the model structure would most likely enable GT-CATS to associate heading adjustments with instances of actions in DUO during LNAV operation.

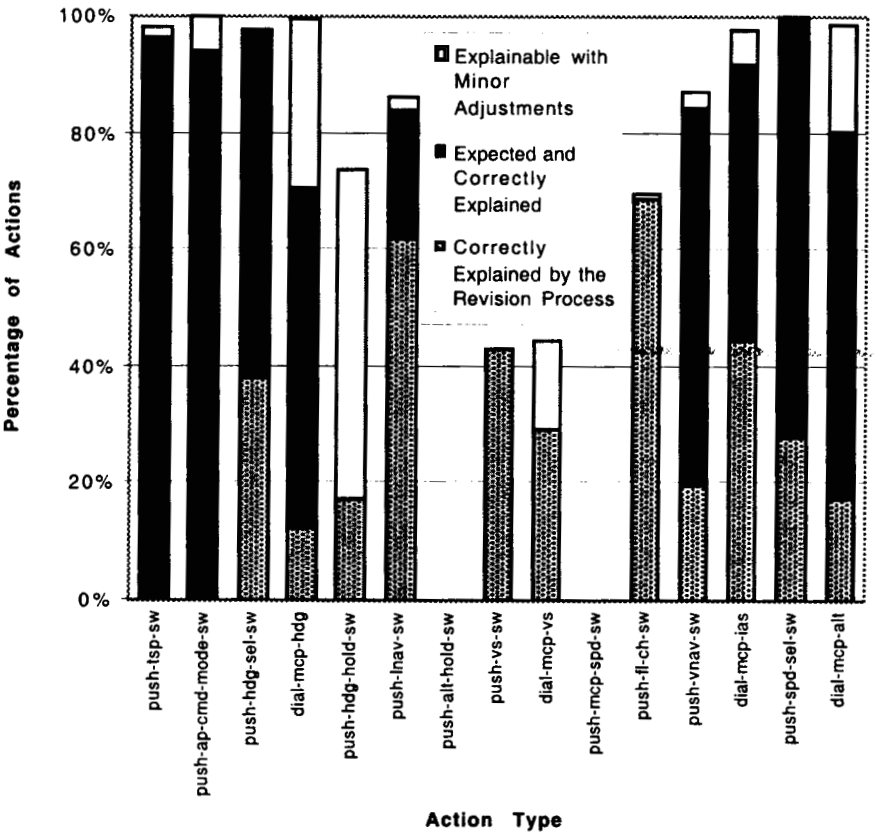


Figure 95. Additional actions explainable with minor adjustments to GT-CATS.

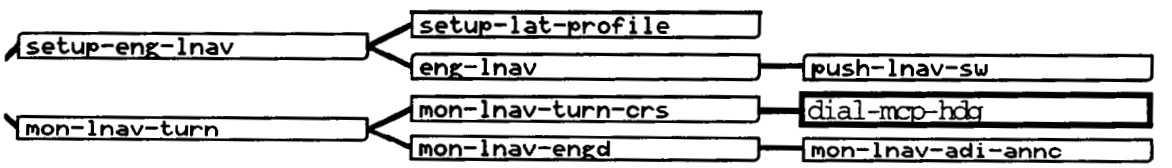


Figure 96. Addition of dial-mcp-hdg action to support the mon-lnav-turn-crs subtask.

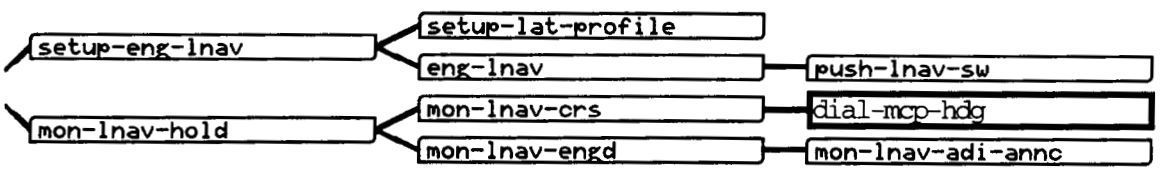


Figure 97. Addition of dial-mcp-hdg action to support the mon-lnav-crs subtask.

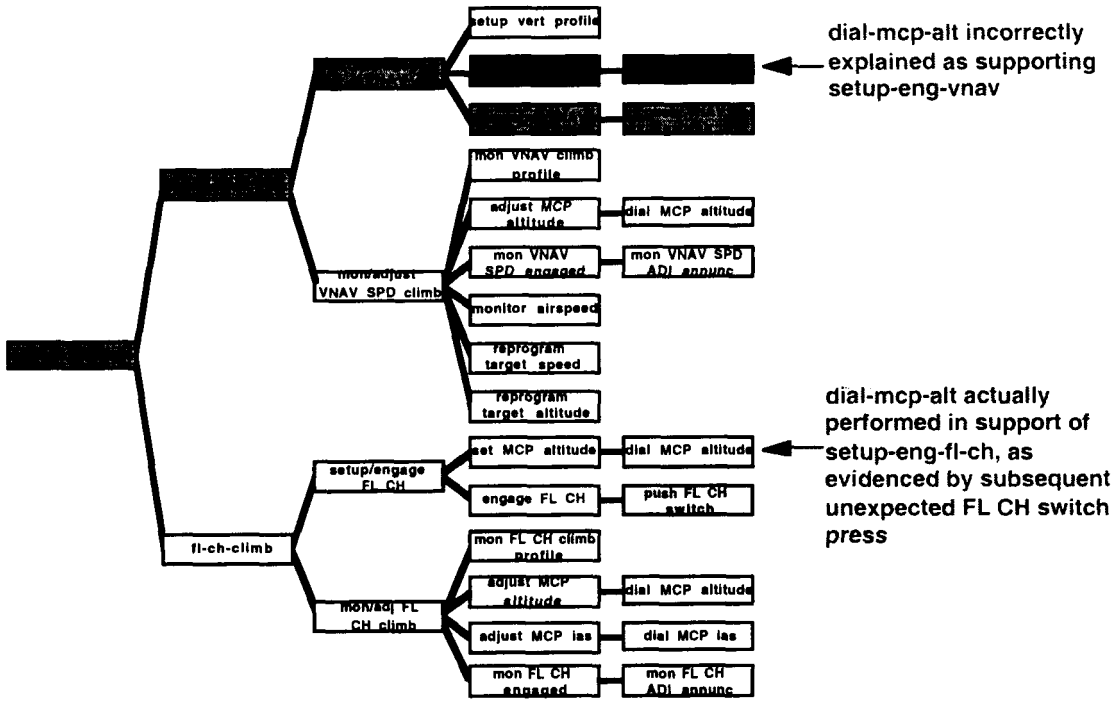


Figure 98. Example of incorrect explanation of dial-mcp-alt action.

OFM-ACM adjustments to explain altitude settings

Another model error that can be corrected with a minor modification to the OFM-ACM would enable GT-CATS to explain nearly all of the dial-mcp-alt actions detected in the study (see figure 95). These actions constitute 15% of actions GT-CATS does not correctly explain as implemented (either following an expectation, or by the revision process). Altitude settings are presently modeled as part of the "setup/engage" task for each vertical mode that pilots can manually engage. Figure 98 shows how an altitude adjustment expected in support of VNAV is incorrectly explained to support VNAV when, in fact, the pilot performed the action in preparation to use FL CH. Because pilots set the MCP altitude whenever a new cleared altitude is issued by ATC, altitude settings may be better modeled as a task separate from the engagement of a particular mode. Figure 99 shows the proposed modification to the OFM-ACM; grey nodes indicate how altitude settings are modeled in GT-CATS' present implementation. The modified OFM-ACM structure, however, removes the ambiguity that leads to incorrect explanations for dial-mcp-alt actions. A similar modification may also enable GT-CATS to better understand speed adjustments; however, speed adjustments are more tightly linked to the use of a particular mode, so such a modification may be unsuitable to faithfully model speed adjustment tasks.

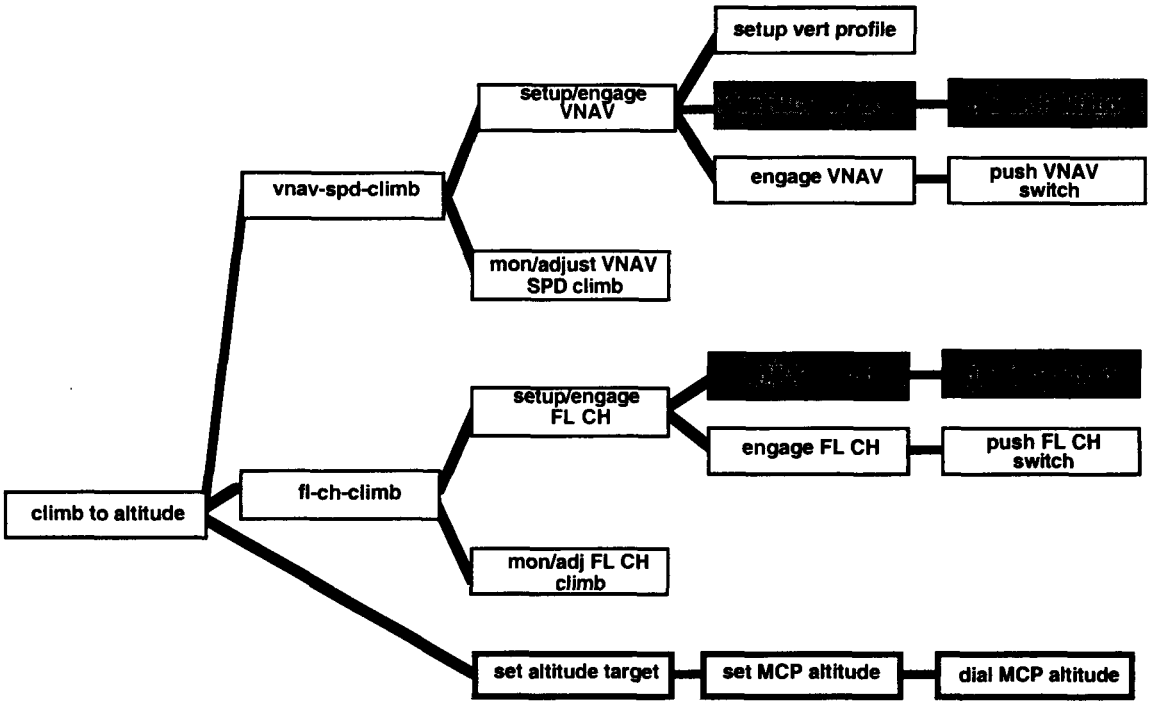


Figure 99. Modeling altitude settings independent of mode.

Other enhancements

The remaining actions that GT-CATS can explain with small modifications result from situations similar to those described above. One pilot who engaged HDG HOLD mode after using HDG SEL to turn onto the cleared heading accounted for the misunderstood push-hdg-hold-sw actions. This usage was not modeled in GT-CATS' OFM-ACM—a model error, although the usage was highly unorthodox.

GT-CATS revised some actions incorrectly due to ambiguities similar to those that caused GT-CATS to explain dial-mcp-alt actions incorrectly. For example, speed may be adjusted using the MCP in both ALT CAP and ALT HOLD modes. Which mode did the action in fact support? A second group of incorrect revisions resulted when pilots attempted to start a descent before the assigned top-of-descent point—a condition for switching from cruise phase to the descent phase. In these cases, GT-CATS incorrectly revised push-vnav-sw actions to support a change of cruise altitude (i.e., a step descent), instead of the actual descent. Again, this processing error could be corrected by allowing GT-CATS access to the next phase.

6% of pilot actions recorded in the GT-CATS evaluation were not explained, and require further research. A variety of additional enhancements are planned as the subject of such research. These enhancements are described in the next chapter.

Results of a post-experimental questionnaire

After completing the five experimental scenarios, each subject pilot was asked a series of thirteen general questions about the reasonableness, realism, and representativeness of the GT-EFIRT simulator and the experimental scenarios (table 14). The questions called for responses on a Likert scale, with 1 being "very bad," "very different," or "very infrequently," and 7 being "very good," "very similar," and "very frequently," depending on the question; 4 was the median response.

Table 14. Post-experimental questionnaire.

1. Rate the training you received.
2. How similar was operating the simulator to the real flying task?
3. Rate the overall reasonableness, realism, and representativeness of the scenarios.
4. Rate the overall reasonableness, realism, and representativeness of the ATC clearances used in the scenarios.
5. Rate the overall reasonableness, realism, and representativeness of the simulator Automated Flight Control System (AFS and FMS).
6. Rate the overall reasonableness, realism, and representativeness of the simulator AFS.
7. Rate the overall reasonableness, realism, and representativeness of the simulator FMS, given its limited functionality.
8. Rate the overall reasonableness, realism, and representativeness of VNAV mode operation.
9. Rate the overall reasonableness, realism, and representativeness of LNAV mode operation.
10. Rate the overall reasonableness, realism, and representativeness of ALT CAP mode operation.
11. Rate the overall speed control performance of the simulator.
12. How easy was it for you to verbalize your intended action?
13. How frequently did you want to use a method not supported by the simulator?
14. General Comments:

The results of the questionnaire indicate that, on the whole, the GT-EFIRT simulator is reasonable, realistic, and representative. General comments also indicated that the GT-EFIRT simulator and experimental scenarios are generally good. Two results, in particular, deserve mention. First, pilots gave good ratings to VNAV mode, as implemented in GT-EFIRT. This is important because poor VNAV performance might have led pilots to select it less often than they otherwise would. Second, pilots indicated that GT-EFIRT supported the navigation methods they would normally use to comply with clearances such as those issued in the study. This is also important because pilots were able to select modes as they would in the actual aircraft.

The averaged responses for the questionnaire are shown in Figure 100. High ratings indicate positive performance for all but question 13, where a low rating is indicative of positive performance. Figure 100 shows that the responses to all questions were, on average, positive. Thus, 757/767 pilots in both the formal evaluation and the "pilot" studies that preceeded it agreed that GT-EFIRT allowed navigation behavior comparable to that found in actual aircraft.

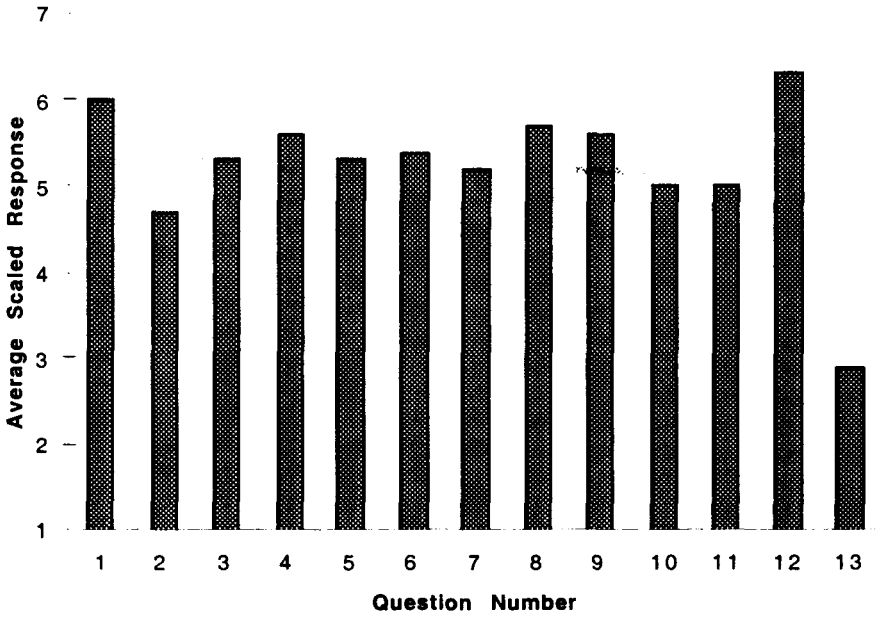


Figure 100. Average responses to the post-experimental questionnaire (n=10).

Pilot mode usage differences

The GT-CATS evaluation revealed variations in the number and type of actions pilots performed. For example, pilots 3, 4, and 7 performed a greater number of actions, and used a wider variety of modes than did pilots 2, 6, and 9. As an example of such differences, figures 101 and 102 depict the explanatory performance of GT- CATS for pilots 3 and 6, respectively. Pilot 3 used more modes than did pilot 6. The results for pilot 3 are representative of results for other pilots who used a "browsing" approach to select modes—selecting one mode, then immediately selecting another instead.

This behavior apparently reflected the desire of some pilots to explore the functionality of the GT-EFIRT simulator in the process of complying with the scenario clearances; at times, however, the behavior resulted from an inappropriate initial mode selection. As with pilot 3, GT-CATS misunderstood more actions of pilots who "browsed" among available modes. For purposes of the evaluation, such actions were not considered errors; only actions meeting the error definitions set forth earlier were investigated as such.

Mode usage differences across scenarios

Differences in performance across scenarios were less pronounced. Such differences primarily reflected the length of the LNAV route, whether clearances called for deviations from the LNAV route, the number and spacing of scenario clearances, and the presence or absence of crossing restrictions. Differences also arose from supplementary clearances issued via the GT-CATS ATC facility, in addition to clearances specified in the design of each scenario.

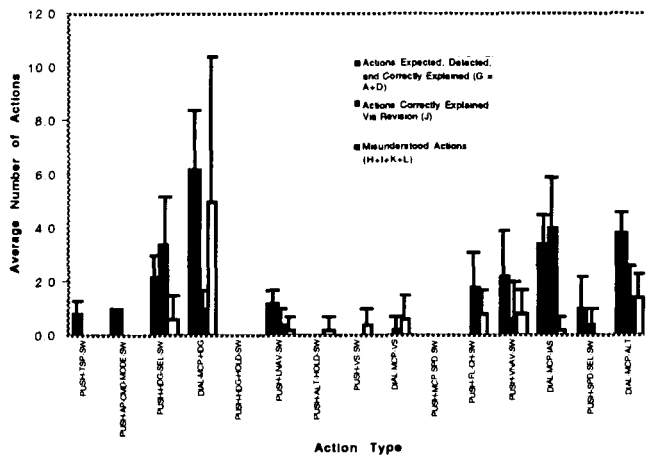


Figure 101. Average numbers of actions performed by subject 3 for all scenarios (error bars indicate one standard deviation).

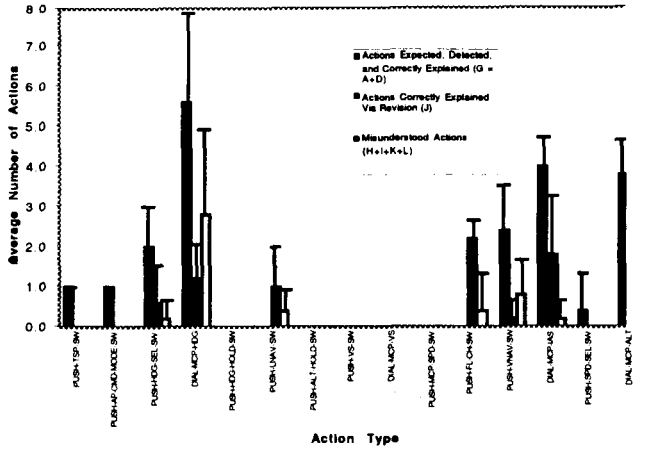


Figure 102. Average numbers of actions performed by subject 6 for all scenarios (error bars indicate one standard deviation).

Supplementary clearances were added during some scenarios to make small modifications to the route, to ensure later clearances were triggered appropriately. Of supplementary clearances, approximately 79% involved modifications to the lateral profile—primarily modifications to heading to allow for a better route intercept heading. Approximately 34% involved modifications to the vertical profile. (12% involved both vertical and lateral profile modifications.) Overall, supplementary ATC clearances accounted for 18% of the total number of ATC clearances issued during the study.

Micro-analysis

A micro-analysis of the data from the GT-CATS evaluation was performed to provide a fine-grained assessment of GT-CATS performance.

The micro-analysis entailed rerunning GT-CATS with the experimental data and identifying the reasons behind individual activity tracking outcomes. The micro-analysis categorizes each unfulfilled expectation, incorrect explanation, unexpected action, incorrect revision, unexplained action, and unknown action, to separate those resulting from implementation-dependent aspects of GT-CATS and GT-EFIRT from those deserving further research. For example, the micro-analysis confirmed that unfulfilled expectations resulted from a change in the situation, an alternative mode selection, or a pilot omission—all cases in which normative expectations are not met by pilot actions. Similarly, the micro-analysis identified misunderstood actions that were actually pilot 'errors.' Thus, the micro-analysis was instrumental in producing the overall view of the evaluation results presented in figure 91.

Summary

The GT-CATS empirical evaluation showed that, overall, GT-CATS predicts and explains pilot activities effectively. GT-CATS successfully predicted and explained over half of the 2,089 actions detected in the study. Furthermore, the GT-CATS revision process was instrumental in explaining approximately 36% of the actions that GT-CATS explained successfully. This result demonstrates the importance of the revision process in tracking the activities of pilots using multiple modes.

8. Conclusions and Further Research

This chapter summarizes the contributions of GT-CATS research. In addition, it posits research areas that would enhance or extend GT-CATS.

Conclusions

GT-CATS extends research on intent inferencing in several ways. First, it provides data on the effectiveness of a methodology for activity tracking in a complex system. Specifically, data are presented for airline pilots performing realistic mode management tasks in the context of realistic flight scenarios. Second, GT-CATS research proposes a set of theoretical knowledge and processing conditions which can track operator activities. Finally, the research suggests how additional knowledge could be captured and utilized within the proposed framework.

GT-CATS contributes four theoretical insights. First, it establishes conditions on the types of knowledge that must be available in a domain in order to build a representation to track operator activities. Detailed information on the state of the controlled system, the state of the control automation, and internal variables used by the automation is required. This information is represented in GT-CATS' state space. Because this knowledge must be available in a form that supports computation, GT-CATS requires that the controlled system is an engineered system from which current state information is directly obtainable.
Second, knowledge about the goals of the operator must also be available in a form that affords comparison of the system state information with current goals. GT-CATS represents knowledge about operator goals in its limiting operating envelope. Again this knowledge must be available in a form that supports computation. In the glass cockpit domain, for example, such information is available via programmed flight plan information contained in the Flight Management System. Amendments to the programmed flight plan required by Air Traffic Control are available via datalink technology.

A third type of knowledge concerns standard operating procedures and methods operators use to meet the systems goals and manage the operation of the controlled system. This knowledge is obtained from operator training curricula and the structure of the control automation. Other engineered domains have comparable well-defined goals and operating procedures to fulfill them (e.g., satellite ground control systems, manufacturing systems, and air traffic control systems).

Given that these types of knowledge are available in a domain, GT-CATS research demonstrates a means of organizing the knowledge to track operator activities in complex systems. Specifically, GT-CATS research shows that an enhanced Operator Function Model—the OFM-ACM—can effectively represent knowledge about an operator's mode management task. At the top of the OFM-ACM hierarchy, knowledge is decomposed into phases and subphases of system operation, and the functions required for each subphase. Below the function level, the OFM-ACM represents knowledge about how control options provided by modes allow the operator to perform various control tasks. These tasks are decomposed into subtasks and, in turn, into operator actions needed to accomplish them.

The third contribution of GT-CATS research specifies how to use domain knowledge to support the inferencing required for activity tracking. Inferencing creates additional requirements on the organization of knowledge about the operator's task. In particular, function-level operator activities represented in the OFM-ACM must be uniquely determinable using knowledge in the state space and limiting operating envelope. Thus, by identifying the functions applicable in the current operating context, the structure of the OFM-ACM identifies viable control options (modes).

Another contribution is the use of context specifiers. Context specifiers serve as a mechanism for transforming knowledge about the state of the controlled system and control automation, along with knowledge about current operator goals as encapsulated in the limiting operating envelope, into activation conditions for nodes in the OFM-ACM. These conditions determine when a particular operator activity is expected.

Context specifiers provide an alternative to script-based inferencing. By acting as activation conditions for nodes in the OFM-ACM, they specify relationships among activities in the OFM-ACM. For example, an operator activity represented in the OFM-ACM (e.g., engage VNAV) may have as a condition a context specifier that results from the effect of a previous activity (e.g., set MCP altitude). Thus, rather than explicitly specifying the ordering of actions using a script, context specifiers provide a mechanism by which actions are flexibly ordered according to the current operating context.

GT-CATS research also proposes an inferencing process. GT-CATS demonstrates that context-specific knowledge can be used to predict operator activities. The revision process allows a prediction to be revised based on updated information when operator activities that support alternative methods (modes) for carrying out the required function are executed.

Finally, GT-CATS demonstrates a method for instantiating, processing, and evaluating the OFM-ACM to predict and interpret operator actions. By evaluating GT-CATS in the context of the glass cockpit, using type-rated pilots, this research establishes activity tracking as a viable approach to interpreting pilot mode usage activities. The GT-CATS empirical evaluation quantifies GT-CATS' activity tracking capabilities in terms of possible outcomes. GT-CATS effectively predicts the mode a pilot will use in the current context, and explains supporting actions as supporting it 51% of the time. GT-CATS' revision process interprets an additional 28% of actions as supporting alternative modes to accomplish the required control functions. Overall, with minor enhancements to the OFM-ACM and GT-CATS' processing scheme, GT-CATS can interpret 94% of pilot actions. Thus, the evaluation establishes the strengths of GT-CATS, and indicates directions for further research.

Enhancements and Suggestions for Further Research

The remainder of this chapter highlights some additional enhancements to GT-CATS that appear promising. Chapter III described how Woods et al.'s (ref. 11) characterization of the factors that impact operators of complex systems applies to modal systems. To effectively choose modes, operators weigh their knowledge of the characteristics of the modes, attentional resources, and strategic factors in the current operating context. The context-specific information GT-CATS uses to predict mode selections reflects the information requirements of the modes (e.g., to use the VNAV mode, a vertical profile must be programmed in the Flight Management System). One area in which in GT-CATS could be usefully enhanced, therefore, is to implement additional state variables and update the state space more frequently.

First, including predictive information in both the state space and limiting operating envelope will improve the ability of GT-CATS to more accurately predict operator activities. For example, the micro-analysis of the current data indicates that if GT-CATS' state space includes predictive information, GT-CATS might better understand operator actions that depend on the capability of the automation to achieve a desired state of the controlled system within a particular time interval. For example, in the glass cockpit domain, predictive information could be used to activate context specifiers that indicate whether a selected altitude and/or airspeed can be achieved in time to

meet a restriction. Such predictive information is often available internally to the automation.

Second, more explicitly modeling operator workload may also improve GT-CATS' predictive ability. The OFM-ACM provides this capability by explicitly modeling cognitive and perceptual actions, such as monitoring the altimeter to ensure that the desired altitude is reached. The micro-analysis in the current experiment showed that as pilots became busier they chose lower levels of automation to reduce their cognitive workload.

Third, temporal factors that underpin the revision process warrant examination. The micro-analysis in the current experiment suggests that the revision process could be improved if better information about the time window during which an action can be reasonably interpreted is available. Information about when to execute the revision process for a particular action might be included in the OFM-ACM.

Beyond improvements to GT-CATS itself, further research should explore applications of activity tracking. First, GT-CATS would be useful for interpreting pilot navigation activities needed to comply with new ATC automation. This application requires that that the OFM-ACM include additional pilot activities (e.g., descents that comply with automation-derived ATC directives, in addition to present descent methods). This would enable researchers and pilots alike to visualize the task or evaluate proposed procedures. Second, GT-CATS can help understand other flight deck areas, such as non-nominal operations. Such applications are expected to result in future enhancements to the underlying architecture of GT-CATS.

Other applications should focus on the use of output from GT-CATS as a source of knowledge for operator's assistants and intelligent tutors. Aiding systems could use GT-CATS to provide intelligent assistance, by offering advice, reminders, or alerts regarding unexpected activities. GT-CATS' OFM-ACM is also suitable as the student and expert models required by an intelligent tutoring systems. Predictions for mode selections provide expert knowledge; GT-CATS' revision process enables alternative mode selections to be explored. The OFM-ACM can also model student knowledge. Such research may involve the development of a "buggy" OFM-ACM that represents common operator errors in using modes of automation, as well as operator activities required in abnormal operating conditions. One such effort is in progress (ref. 15).

In conclusion, GT-CATS research demonstrates a viable methodology for predicting and interpreting operator activities in complex systems. Along with earlier research efforts, GT-CATS serves as the basis for important insights into the application of activity tracking. Ultimately, it is hoped that this research may have a positive impact on the safe and efficient operation of complex systems of the future.

References

1. Mitchell, C. M. (1987). GT-MSOCC: a research domain for modeling human-computer interaction and aiding decision making in supervisory control systems. *IEEE Transactions on Systems, Man, and Cybernetics, 17(4),* 553-570.

2. Banks, S. B., and Lizza, C. S. (1991). Pilot's Associate: a cooperative, knowledge-based system application. *IEEE Expert, 6(3),* 18-29.

3. Geddes, N. D., and Hammer, J. M. (1991). Automatic display management using dynamic plans and events. *Proceedings of the Sixth International Symposium on Aviation Psychology,* Columbus, OH.

4. Rubin, K.S., Jones P. M., and Mitchell, C. M. (1988). OFMspert: Inference of operator intentions in supervisory control using a blackboard architecture. *IEEE Transactions on Systems, Man, and Cybernetics, 18(4),* 618-637.

5. Smith, S. C., Govindaraj, T., and Mitchell, C. M. (1990). Operator Modeling in Civil Aviation. *Proceedings of the 1990 IEEE Conference on Systems, Man, and Cybernetics,* Los Angeles, CA, 512-514.

7. Chu R. W., Jones, P. M., and Mitchell, C. M. (1995). Using the operator function model and OFMspert as the basis for an intelligent tutoring system: Towards a tutor/aid paradigm for operators of supervisory control systems. *IEEE Transactions on Systems, Man, and Cybernetics, 25,* 1054-1075.

8. Billings, C. E. (1991, August). *Human-centered aircraft automation: a concept and guidelines* (NASA Contractor Report 103885). Moffett Field, CA: NASA Ames Research Center.

9. Sheridan, T. B. (1992). *Telerobotics, automation, and human supervisory control.* Cambridge, MA: MIT Press.

10. Bainbridge, 1987

11. Woods, D. D., Johannesen, L. J., Cook, R. I., and Sarter, N. B. (1994). *Behind human error: Cognitive systems, computers, and hindsight.* Dayton, OH: Crew Systems Ergonomic Information and Analysis Center.

12. Kirlik, A. (1995). Requirements for psychological models to support design: Toward ecological task analysis. In J. M. Flach, P. A. Hancock, J. K. Caird, and K. J. Vicente (Eds.), *An ecological approach to human machine systems I: A global perspective.* Hillsdale, NJ: Lawrence Erlbaum.

13. Mitchell, C. M. and Saisi, D. L. (1987). Use of model-based qualitative icons and adaptive windows in workstations for supervisory control systems. *IEEE Transactions on Systems, Man, and Cybernetics, 17(4),* 573-593.

14. Thurman, D. A., and Mitchell, C. M. (1994). Methodology for the design of interactive monitoring interfaces. *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, Piscataway, NJ, 1739-1744.

15. Chappell, A. R., and Mitchell, C. M. (1995). Addressing the trained novice/expert performance gap in complex dynamic systems: A case-based intelligent tutoring system. *Proceedings of the 1994 IEEE Conference on Systems, Man, and Cybernetics,* Vancouver, BC, 4557-4562.

16. Rouse, W. B. (1984). Design and evaluation of computer-based support systems. In G. Salvendy (Ed.), *Advances in Human Factors/Ergonomics, Volume 1:*

Human-Computer Interaction. New York: Elsevier.

17. Rouse, W. B., and Morris, N. M. (1985). Conceptual design of a human error tolerant interface for complex engineering systems. *Proceedings of the Second IFAC/IFIP/IFORS/IEA Conference on Analysis, Design, and Evaluation of Man-Machine Systems*. Varese, Italy.

18. Rouse, W. B., Geddes, N. D., and Curry, R. E. (1987). An architecture for intelligent interfaces: Outline of an approach to supporting operators of complex systems. *Human-Computer Interaction, 3*, 87-122.

19. Palmer, E. A., Hutchins, E. L., Ritter, R. D., and vanCleemput, I. (1991). Altitude deviations: breakdowns of an error tolerant system (NASA Technical Memorandum 108788). Moffett Field, CA: NASA Ames Research Center.

20. Funk, K. H., and Kim (1995) Agent-based aids to facilitate cockpit task management. *Proceedings of the 1995 IEEE Conference on Systems, Man, and Cybernetics*, 1521-1526.

21. Boy, G. A. (1987). Operator assistant systems. *International Journal of Man-Machine Studies, 27*, 541-554.

22. Broadwell, M. M. (1987). Pilot's Associate: challenges and approaches. *Proceedings of the Third Annual Artificial Intelligence and Advanced Computer Technology Conference*, Long Beach, CA, 297-304.

23. Smith, D. M., and Broadwell, M. M. (1989). The Pilot's Associate: an overview. *Proceedings of the Eighth International Workshop on Expert Systems and Their Applications*, Avignon, France, 263-269.

24. Mitchell, C. M. (1995). Human-machine system models: A prerequisite to the design of human-computer interaction in complex dynamic systems. In A. P. Sage and W. B. Rouse (Eds.), *Handbook of systems engineering and management*. New York: John Wiley and Sons, to appear.

25. Callantine, T. J., and Mitchell, C. M. (1994). A methodology and architecture for understanding how operators select and use modes of automation in complex systems. *Proceedings of the 1994 IEEE Conference on Systems, Man, and Cybernetics,* San Antonio, TX, 1751-1756.

26. Hammer, J. M., and Rouse, W. B. (1982). Design of an intelligent computer-aided cockpit. *Proceedings of the 1982 International Conference on Cybernetics and Society*, Seattle, WA, 449-453.

27. Rouse, W. B., Rouse, S., and Hammer, J. M. (1982). Design and evaluation of an onboard computer-based information system for aircraft. *IEEE Transactions on Systems, Man, and Cybernetics, 12(4)*, 451-463.

28. Geddes, N. D. (1985). Intent inferencing using plans and scripts. *Proceedings of the First Annual Aerospace Applications of Artificial Intelligence Conference,* Dayton, OH.

29. Geddes, N. D. (1989). Understanding human operators' intentions in complex systems. Unpublished Ph.D. dissertation. Atlanta, GA: Georgia Institute of Technology.

30. Chambers, A. B., and Nagel, D. C. (1985). Pilots of the future: Human or Computer? *Communications of the ACM, 28(11)*, 1187-1198.

31. Hollnagel, E. (1987). Commentary: Issues in knowledge-based decision support. *International Journal of Man-Machine Studies, 27*, 743-751.

32. Rasmussen, J., and Goodstein, L. P. (1987). Decision support in supervisory control of high-risk industrial systems. *Automatica*, 663-671.

33. Roth, E. M., Bennett, K. B., and Woods, D. D. (1987). Human interaction with an "intelligent" machine. *International Journal of Man-Machine Studies, 27*, 479-525.

34. Woods, D. D. (1987). Commentary: Cognitive engineering in complex and dynamic worlds. *International Journal of Man-Machine Studies, 27*, 571-585.

35. Woods, D. D. (1986). Cognitive technologies: The design of joint human-machine cognitive systems. *AI Magazine*, 86-92.

36. Woods, D. D. and Hollnagel, E. (1987). Mapping cognitive demands in complex problem-solving worlds. *International Journal of Man-Machine Studies, 26*, 257-275.

37. Woods, D. D. and Roth, E. M. (1988). Cognitive systems engineering. In Hollander, M., (Ed.), Handbook of Human-Computer Interaction. New York: Springer-Verlag.

38. Bushman, J. B., Mitchell, C. M., Jones, P. M., and Rubin, K. S. (1993). ALLY: An operator's associate for cooperative supervisory control systems. *IEEE Transactions on Systems, Man, and Cybernetics, 23(1)*, 111-128.

39. Jones, P. M., and Mitchell, C. M. (1994). Human-computer cooperative problem solving: Theory, design, and evaluation of an intelligent associate system. *IEEE Transactions on Systems, Man, and Cybernetics, 25(7)*, 1039-1053.

40. Shalin, V. L., Geddes, N. D., Mikesell, B.G., and Ramamurthy, M. (1993). Evidence for plan-based performance and implications for information management on the commercial aviation flight deck. *Proceedings of the Fourth International Conference on Human-Machine Interaction and Artificial Intelligence in Aerospace*, Toulouse, France.

41. Jones, P. M., and Mitchell, C. M. (1987). Operator modeling: Conceptual and methodological distinctions. *Proceedings of the Human Factors Society 31st Annual Meeting*, 31-35.

42. Rasmussen, J. (1986). *Information processing and human-machine interaction.* Amsterdam: North Holland.

43. Baron, S. (1984). A systems approach to modeling discrete control performance. In W. B. Rouse (Ed.), *Advances in Man-Machine Systems Research*, 1, Greenwhich, CN: JAI Press, Inc, 1-48.

44. Miller, R. A. (1985). A systems approach to modeling discrete control performance. In W. B. Rouse (Ed.), *Advances in man-machine systems research, volume 2.* Greenwich, CN: JAI Press, Inc., 177-248.

45. Wickens, C. D. (1984). *Engineering psychology and human performance.* Columbus, OH: Merrill.

46. Vicente, K. J., and Rasmussen, J. (1992). Ecological interface design: Theoretical foundations. *IEEE Transactions on Systems, Man, and Cybernetics, 22(4)*, 589-607.

47. Shank, R., and Abelson, R. (1977). *Scripts, plans, goals, and understanding.* Hillsdale, NJ: Erlbaum.

48. Wilensky, R. (1983). *Planning and Understanding*. Reading, MA: Addison Wesley.

49. Nii, P. (1986). Blackboard systems. *AI Magazine, 7(2) and 7(3)*, June, 1986.

50. Jones, P. M., Chu, R. W., and Mitchell, C. M. (1994). A methodology for human-machine systems research: knowledge engineering, modeling, and simulation. *IEEE Transactions on Systems, Man, and Cybernetics, 25(7)*, 1025-1038.

51. Colucci, F. (1995). Rotorcraft Pilot's Associate update: The army's largest science and technology program. *Vertiflight, 41(2)*, 16-20.

52. Funk, K. H., and Lind, J. H. (1992). Agent-based pilot-vehicle interfaces. *IEEE Transactions on Systems, Man, and Cybernetics, 22(6)*, 1309-1322.

53. Onken, R., and Prevôt, T. (1995). CASSY—cockpit assistant system for IFR operation. *Proceedings of the 19th ICAS Congress*, Anaheim, CA.

54. Robson, M., Fairbanks, M., and Shorthose, M. (1995). *Final report on a feasibility study into an intelligent flight path monitor*. London: Civil Aviation Authority.

55. Amalberti, R., and Deblon, F. (1992). Cognitive modeling of fighter aircraft process control: a step towards an intelligent on-board assistance system. *International Journal of Man-Machine Studies, 36*, 639-671.

56. Mellor, P. (1994). CAD: Computer-aided disaster. *High Integrity Systems, 1(2)*, 101-156.

57. Hughes, D. and Dornheim, M. A. (1995). Accidents direct focus on cockpit automation. *Aviation Week and Space Technology*, 30 January, 52-54.

58. Eldredge, D., Dodd, R. D., and Mangold, S. J. (1991). *A review and discussion of flight management system incidents reported to the aviation safety reporting system.* Cambridge, MA: Volpe National Transportation Systems Center.

59. Degani, A., Mitchell, C. M., and Chappell, A. R. (1995) Task models to guide analysis: Use of the operator function model to represent mode transitions. *Proceedings of Eighth International Symposium on Aviation Psychology*, Columbus, OH.

60. Funk, K. H., Lyall, B., and Riley, V. (1995). Perceived human factors problems of flightdeck automation (Phase I final report). Corvalis, OR: Oregon State University.

61. Weiner, E. L. and Curry, R. E. (1980). Flight-deck automation: Promises and problems. *Ergonomics, 23(10)*, 995-1011.

62. Boehm-Davis, D. A., Curry, R. E., Wiener, E. L., and Harrison, R. L. (1983). Human factors of flight-deck automation—report on a NASA/industry workshop. *Ergonomics, 26*, 953-961.

63. Curry, R. E. (1985). *The introduction of new cockpit technology: A human factors study* (NASA Technical Memorandum 86659). Moffett Field, CA: NASA Ames Research Center.

64. Wiener, E. L. (1985). *Human factors of cockpit automation: field study of flight crew transition* (NASA Contractor Report 177333). Moffett Field, CA: NASA Ames Research Center.

65. Wiener, E. L. (1988). Cockpit automation. In E. L. Wiener and D. C.

Nagel (Eds.), *Human factors in aviation.* San Diego: Academic Press.

66. Braun, R. and Fadden, D. M. (1987). Flight deck automation today — where do we go from here? *Proceedings of the Conference on Aerospace Behavioral Engineering Technology,* 141-149.

67. Wiener, E. L. (1989). *The human factors of advanced technology ("glass cockpit") transport aircraft* (NASA Contractor Report 177528). Moffett Field, CA: NASA Ames Research Center.

68. Sarter, N. B., and Woods, D. D. (1992). Mode error in supervisory control of automated systems. *Proceedings of the Human Factors Society 36th Annual Meeting*, Atlanta, GA.

69. Sarter, N. B., and Woods, D. D. (1994). Pilot interaction with cockpit automation II: An experimental study of pilots' model and awareness of the Flight Management System (FMS). *International Journal of Aviation Psychology, 4(1),* 1-28.

70. Sarter, N. B., and Woods, D. D. (1995). *"Strong, silent, and 'out-of-the-loop'": Properties of advanced (cockpit) automation and impact on human-automation interaction* (CSEL Report 95-TR-01). Columbus, OH: Cognitive Systems Engineering Laboratory, Ohio State University.

71. Degani, A., Shafto, M., and Kirlik, A. (1995). Mode usage in automated cockpits: Some initial observations. In T. B. Sheridan (Ed.), *Proceedings of the International Federation of Automatic Control; Man-Machine Systems (IFAC-MMS) Conference,* Boston, MA: IFAC.

72. Nievergelt, J., and Weydert, J. (1980). Sites, modes, and trails: Telling the user of an interactive system where he is, what he can do, and how to get to places. In R. M. Baecker and W. A. S. Buxton (Eds.), *Readings in human-computer interaction: A multidisciplinary approach.* Los Altos, CA: Morgan Kaufmann.

73. Tesler, L. (1981). The Smalltalk environment. *Byte*, August, 90-147.

74. Smith, D. C., Irby, C., Kimball, R., Verplank, W., and Harslem, E. (1982). Designing the Star User Interface. In R. M. Baecker and W. A. S. Buxton (Eds.), *Readings in human-computer interaction: A multidisciplinary approach.* Los Altos, CA: Morgan Kaufmann.

75. Norman, D. A. (1981). Categorization of action slips. *Psychological Review, 88(1),* 1-15.

76. Lewis, C., and Norman, D. A. (1986). Designing for error. In D. A. Norman and S. W. A. Draper (Eds.), *User centered system design.* Hillsdale, NJ: Lawrence Erlbaum Associates.

77. Norman, D. A. (1988). *The psychology of everyday things.* New York: Basic Books.

78. Monk, A. (1986). Mode errors: A user-centered analysis and some preventative measures using keying-contingent sound. *International Journal of Man-Machine Studies, 24,* 313-327.

79. Sellen, A. J., Kurtenbach, G. P., and Buxton, W. A. S. (1992). The prevention of mode errors through sensory feedback. *Human-Computer Interaction*, 7, 141-164.

80. Perlman, G. (1985). Making the right choices with menus. In R. M. Baecker and W. A. S. Buxton (Eds.), *Readings in human-computer interaction: A multidisciplinary approach.* Los Altos, CA: Morgan Kaufmann.

81. Vakil, S. S., Hansman, R. J., Jr., Midkiff, A. H., and Vaneck, T. (1995). Mode awareness in advanced autoflight systems. *Proceedings of the 1995 IFAC/IFIP/IFORS/IEA Conference on Analysis, Design, and Evaluation of Man-Machine Systems*. Vancouver, B.C.

82. Sherry, L., Youssefi, D., and Hynes, C. H. (1995). *A formalism for the specification of operationally embedded reactive avionics systems* (Publication C69-5350-001). Phoenix, AZ: Honeywell.

83. Reason, J. T. (1990). *Human error.* Cambridge: Cambridge University Press.

84. Suchman, L. (1987). *Plans and situated actions: The problem of human-machine communication.* New York: Cambridge University Press.

85. Norman, D. A. (1990). The 'problem' of automation: Inappropriate feedback and interaction, not 'over-automation.' *Philosophical Transactions of the Royal Society of London, B 327*, 585-593.

86. Simon, H. A. (1957). *Models of Man*, New York: John Wiley and Sons.

87. Simon, H. A. (1969). *The sciences of the artificial.* Cambridge, MA: MIT Press.

88. Gopal, B. S., and Rao, C. R. S. (1991). *Indian Airlines A320 VT EP: Report of the technical assessors to the court of inquiry.* Indian Government.

89. Thurman, D. A. and Mitchell, C. M. (1995). A methodology for the design of monitoring and control interfaces to highly autonomous systems, *Proceedings of the 6th Annual IFAC/IFORS/IFIP/SEA Symposium on Man-Machine Systems,* Boston, MA.

90. Jones, P. M., Mitchell, C. M. and Rubin, K. S. (1988). Intent inferencing with a model-based operator's associate. *Proceedings of the Sixth Symposium on Empirical Foundations of Information and Software Sciences,* 249 – 258.

91. Funk, K. H., and Lind, J. H. (1992). Agent-based pilot-vehicle interfaces. *IEEE Transactions on Systems, Man, and Cybernetics, 22*(6), 1309-1322.

92. Clancey, W. J. (1987). *Knowledge-based tutoring: The GUIDON program.* Cambridge, MA: MIT Press.

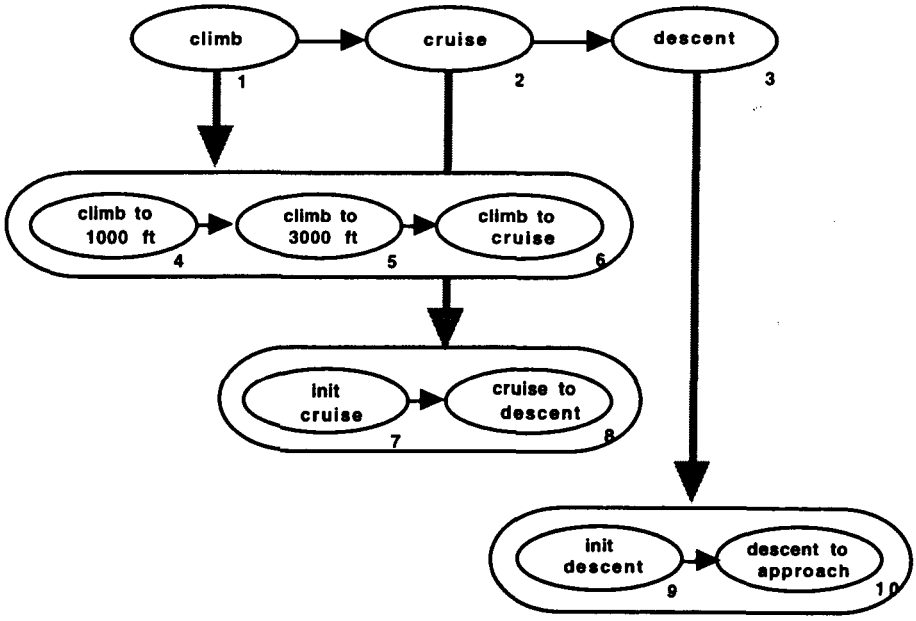
93. Wenger, E. (1987). *Artificial intelligence and tutoring systems: Computational and cognitive approaches to the communication of knowledge.* Los Altos, CA: Morgan Kaufmann.

94. Bainbridge, L. (1979). Verbal reports as evidence of the process operator's knowledge. *International Journal of Man-Machine Studies, 11*, 411-436.

95. Williams, J. A. and Mitchell, C. M. (1993). Effects of integrated flight path and terrain displays on controlled flight into terrain. *Proceedings of the 1993 IEEE Conference on Systems, Man, and Cybernetics,* Le Touquet, France, 709-714.

96. Williams, J. A. and Mitchell, C. M. (1993). Effects of integrated flight path and terrain displays on controlled flight into terrain. *Proceedings of the 1993 IEEE Conference on Systems, Man, and Cybernetics*, Le Touquet, France, 709-714.

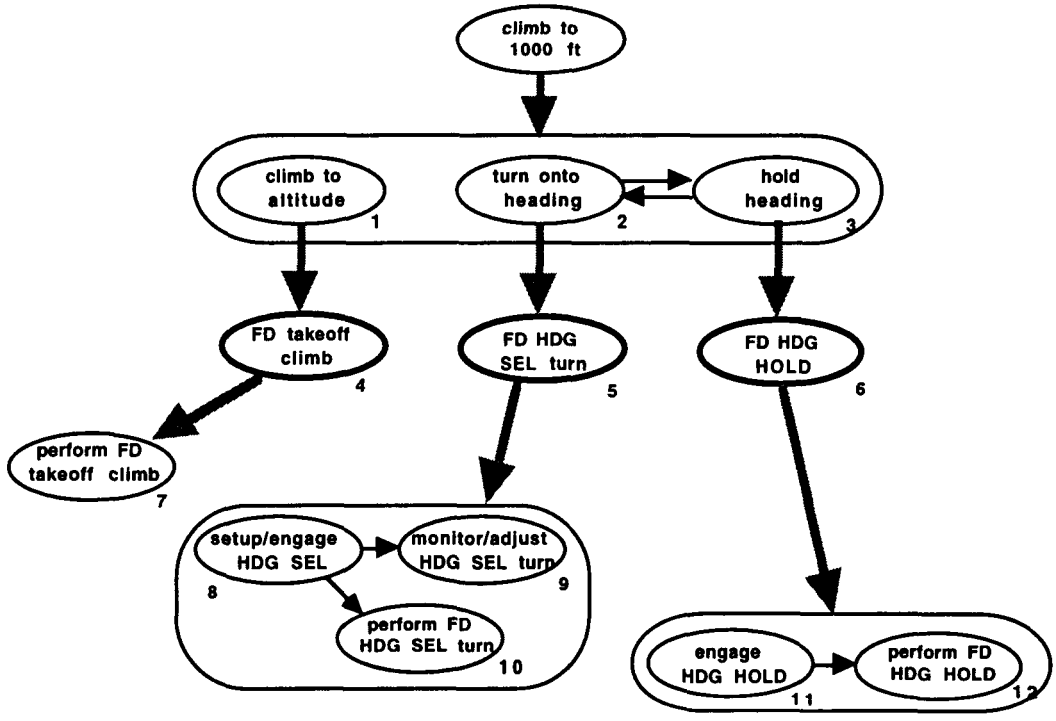
APPENDIX A: GRAPHICAL DEPICTION OF GT-CATS OFM-ACM



Conditions Legend

Phase level:
1. current-phase climb in-progress
2. current-phase cruise in-progress
3. current-phase descent in-progress

Subphase level:
4. acrft-state alt above-origin-apt
5. acrft-state abs-alt at-or-above-1000
6. acrft-state abs-alt at-or-above-3000
7. current-phase cruise in-progress
   & aircraft-position more-than-5-miles-to top-of-descent
8. current-phase cruise in-progress
   & aircraft-position less-than-5-miles-to top-of-descent
9. current-phase descent in-progress
   & aircraft-position more-than-5-miles-to end-of-descent
10. current-phase descent in-progress
   & aircraft-position less-than-5-miles-to end-of-descent

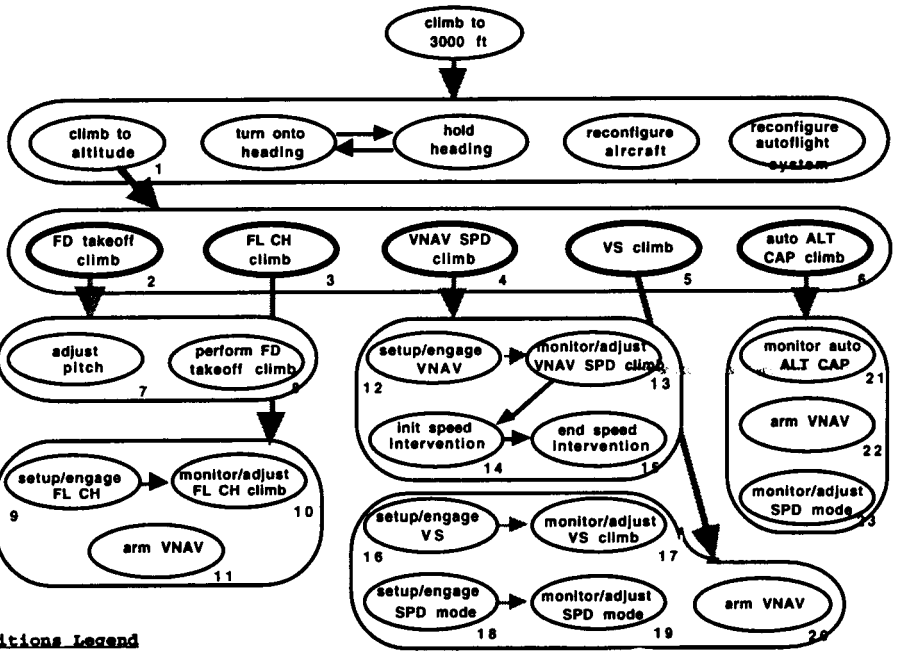


Conditions Legend

Function level:
1. acrft-state alt below-limits
2. acrft-state hdg outside-limits
3. acrft-state hdg within-limits

Mode Selection level:
4. afs-state cmd-mode fd
5. afs-state cmd-mode fd
6. afs-state cmd-mode fd

Task level:
7. afs-state cmd-mode fd
8. afs-state roll-engd not-hdg-sel
9. afs-state roll-engd hdg-sel
10. afs-state roll-engd hdg-sel
11. afs-state roll-engd not-hdg-hold
12. afs-state roll-engd hdg-hold



Conditions Legend

Function level:

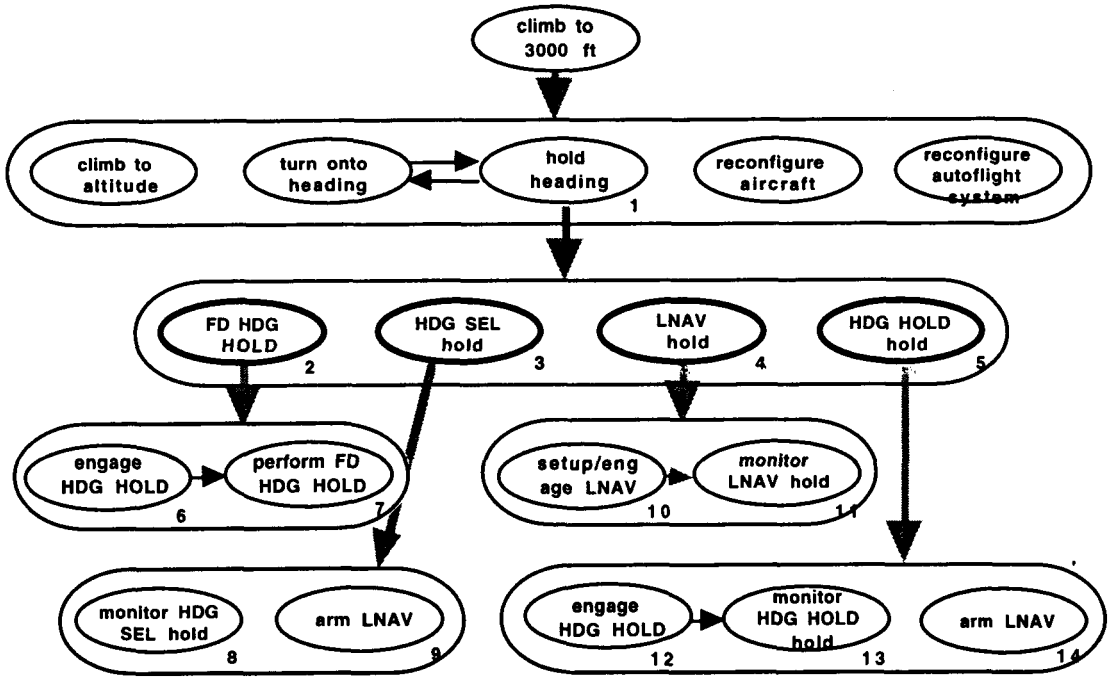
1. acrft-state alt below-limits

Mode Selection level:

2. afs-state cmd-mode fd
3. fms-state vert-profile not-progrmd
   & afs-state cmd-mode cmd
   & acrft-state alt more-than-2000-from-tgt
   & afs-state pitch-engd not-vs
   & afs-state pitch-engd not-alt-cap-rqd-alt
4. fms-state vert-profile progrmd
   & afs-state cmd-mode cmd
   & afs-state tsp clb
   & afs-state pitch-engd not-alt-cap-rqd-alt
5. fms-state vert-profile not-progrmd
   & afs-state cmd-mode cmd
   & acrft-state alt less-than-2000-from-tgt
   & afs-state pitch-engd not-alt-cap-rqd-alt
6. afs-state mcp-alt within-limits
   & afs-state cmd-mode cmd
   & afs-state pitch-engd alt-cap-rqd-alt

Task level:

7. acrft-state abs-alt at-or-above-1000
8. afs-state cmd-mode fd
9. afs-state athr-engd not-fl-ch
10. afs-state athr-engd fl-ch
11. acrft-state abs-alt at-or-above-1000
   & fms-state vert-profile progrmd
   & fms-state vert-profile-intcpt progrmd
  or
   afs-state roll-armed vnav
12. afs-state pitch-engd not-vnav
   & afs-state pitch-armed not-vnav
13. afs-state pitch-engd vnav
14. fms-state vnav-spd-int off
   & fms-state tgt-spd outside-limits
   & afs-state pitch-engd vnav
  or
   fms-state vnav-spd-int on
   & afs-state mcp-spd outside-limits
15. fms-state sched-tgt-spd within-limits
   & fms-state tgt-spd within-limits
   & fms-state vnav-spd-int on
   & afs-state pitch-engd vnav
16. afs-state pitch-engd not-vs
   & afs-state pitch-engd not-alt-cap
17. afs-state pitch-engd vs
18. afs-state athr-engd not-spd
19. afs-state athr-engd spd
20. acrft-state abs-alt at-or-above-1000
   & fms-state vert-profile progrmd
   & fms-state vert-profile-intcpt progrmd
  or
   afs-state roll-armed vnav
21. afs-state pitch-engd alt-cap
22. acrft-state abs-alt at-or-above-1000
   & fms-state vert-profile progrmd
   & fms-state vert-profile-intcpt progrmd
  or
   afs-state roll-armed vnav
23. afs-state athr-engd spd

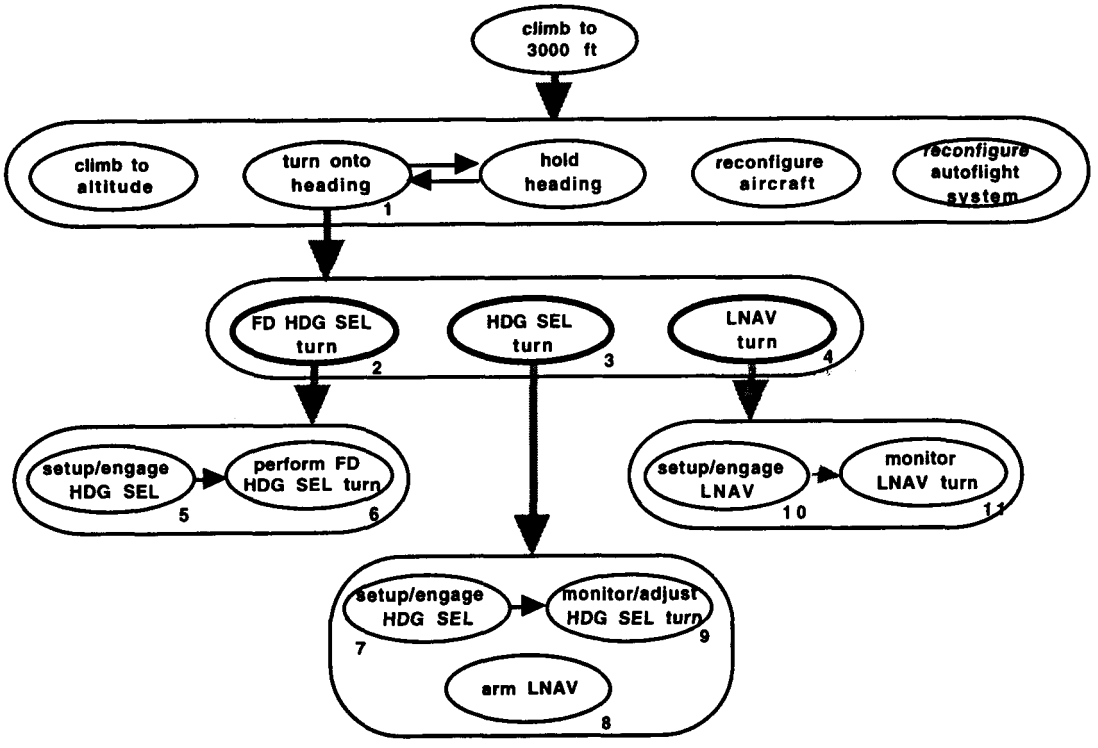


Conditions Legend

Function level:
1. acrft-state hdg within-limits

Mode Selection level:
2. afs-state cmd-mode fd
3. fms-state lat-profile not-progrmd
   & afs-state cmd-mode cmd
   & afs-state roll-engd hdg-sel
   & afs-state mcp-hdg within-limits
4. fms-state lat-profile progrmd
   & afs-state cmd-mode cmd
5. fms-state lat-profile not-progrmd
   & afs-state cmd-mode cmd
   & afs-state roll-engd not-hdg-sel

Task level:
6. afs-state roll-engd not-hdg-hold
7. afs-state afs-state cmd-mode fd
8. afs-state mcp-hdg within-limits
9. fms-state lat-profile-intcpt progrmd
   & afs-state roll-engd not-lnav
   or
   afs-state roll-armed lnav
10. afs-state roll-engd not-lnav
   & afs-state roll-armed not-lnav
11. afs-state roll-engd lnav
12. afs-state roll-engd not-hdg-hold
13. afs-state roll-engd hdg-hold
14. fms-state lat-profile-intcpt progrmd
   & afs-state roll-engd not-lnav
   or
   afs-state roll-armed lnav

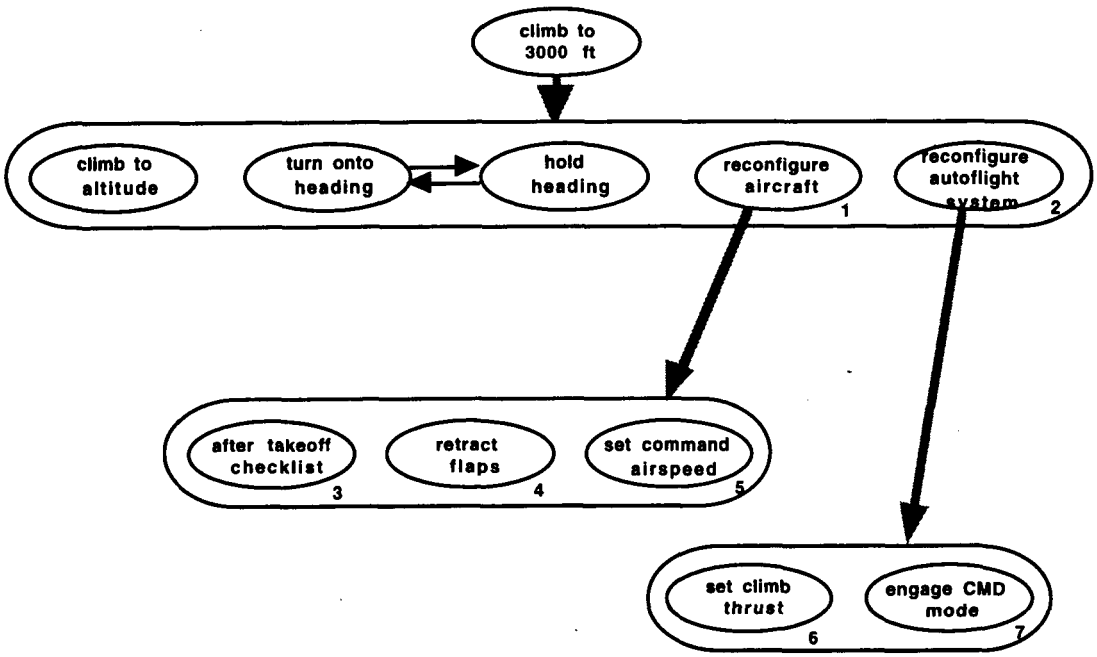


Conditions Legend

Function level:
1. acrft-state hdg outside-limits

Mode Selection level:
2. afs-state cmd-mode fd
3. fms-state lat-profile not-progrmd
   & afs-state cmd-mode cmd
4. fms-state lat-profile progrmd
   & afs-state cmd-mode cmd

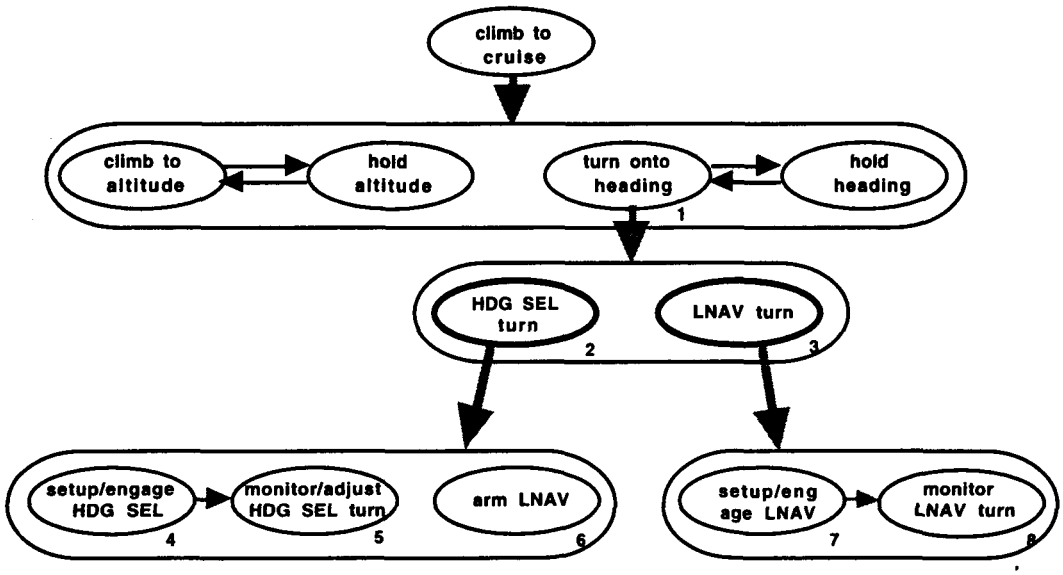
Task level:
5. afs-state roll-engd not-hdg-sel
6. afs-state roll-engd hdg-sel
7. afs-state roll-engd not-hdg-sel
8. fms-state lat-profile-intcpt progrmd
   & afs-state roll-engd not-lnav
   or
   afs-state roll-armed lnav
9. afs-state roll-engd hdg-sel
10. afs-state roll-engd not-lnav
   & afs-state roll-armed not-lnav
11. afs-state roll-engd lnav



Conditions Legend

Function level:
1. acrft-state abs-alt at-or-above-1000
2. acrft-state abs-alt at-or-above-1000

Task level:
3. acrft-state abs-alt at-or-above-1000
4. acrft-state abs-alt at-or-above-1000
5. acrft-state abs-alt at-or-above-1000
   & afs-state mcp-spd outside-limits
   & afs-state pitch-engd not-alt-cap
6. acrft-state abs-alt at-or-above-1000
   & afs-state tsp not-clb
7. afs-state cmd-mode fd
   or
   afs-state cmd-mode cmd

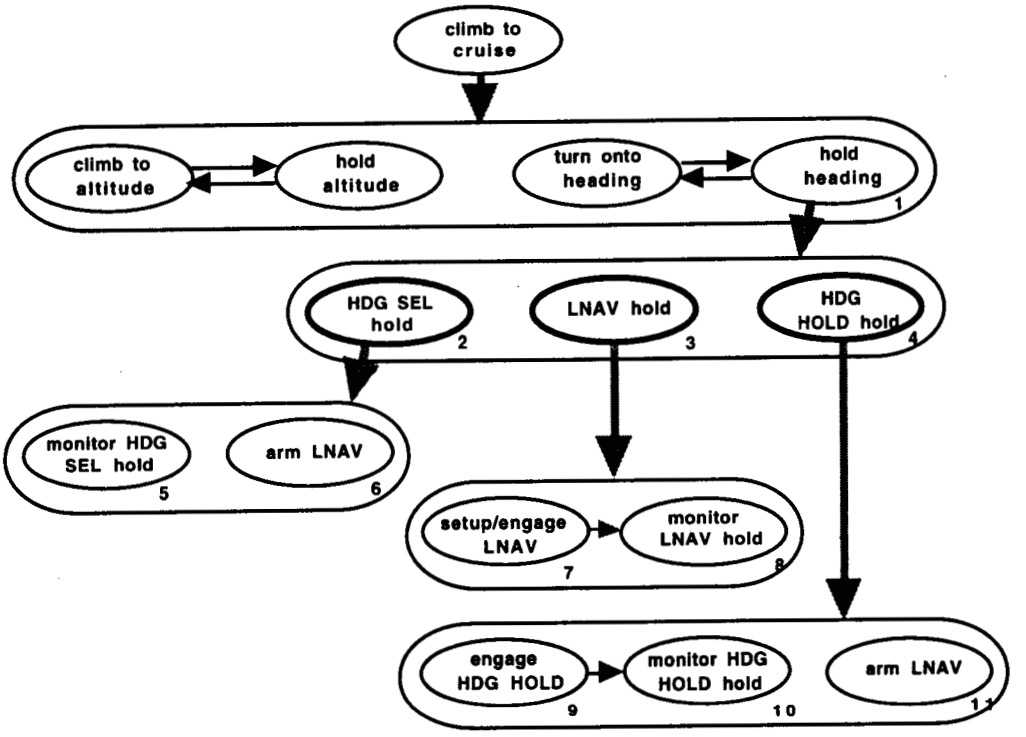


Conditions Legend

Function level:
1. acrft-state hdg outside-limits

Mode Selection level:
2. fms-state lat-profile not-progrmd
   & afs-state cmd-mode cmd
3. fms-state lat-profile progrmd
   & afs-state cmd-mode cmd

Task level:
4. afs-state roll-engd not-hdg-sel
5. afs-state roll-engd hdg-sel
6. fms-state lat-profile-intcpt progrmd
   & afs-state roll-engd not-lnav
   or
   afs-state roll-armed lnav
7. afs-state roll-engd not-lnav
   & afs-state roll-armed not-lnav
8. afs-state roll-engd lnav

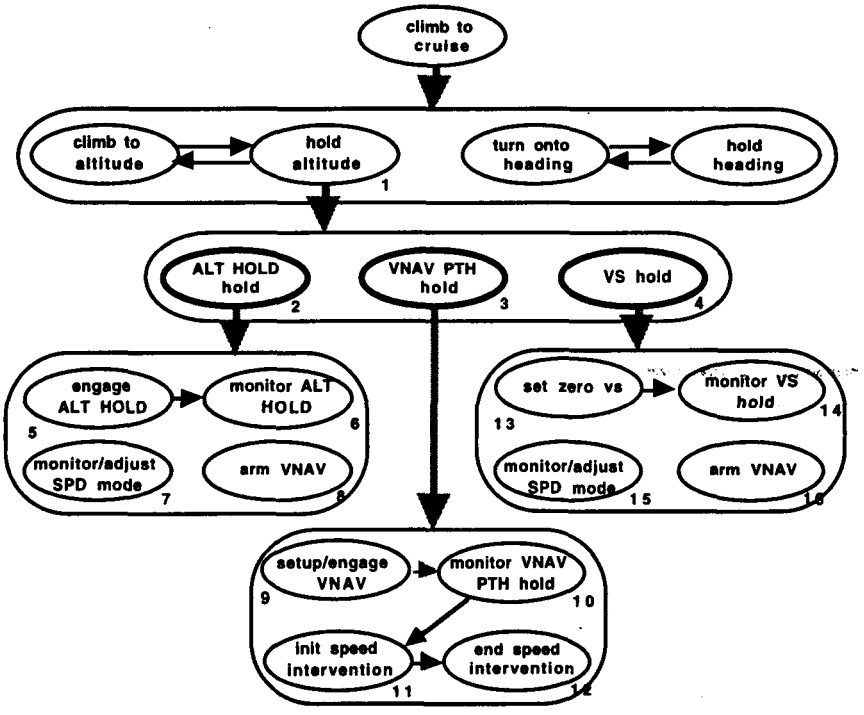


Conditions Legend

Function level:
1. acrft-state hdg within-limits

Mode Selection level:
2. fms-state lat-profile not-progrmd
   & afs-state roll-engd hdg-sel
   & afs-state mcp-hdg within-limits
3. fms-state lat-profile progrmd
   & afs-state cmd-mode cmd
4. fms-state lat-profile not-progrmd
   & afs-state cmd-mode cmd
   & afs-state roll-engd not-hdg-sel

Task level:
5. afs-state mcp-hdg within-limits
6. fms-state lat-profile-intcpt progrmd
   & afs-state roll-engd not-lnav
   or
   afs-state roll-armed lnav
7. afs-state roll-engd not-lnav
   & afs-state roll-armed not-lnav
8. afs-state roll-engd lnav
9. afs-state roll-engd not-hdg-hold
10. afs-state roll-engd hdg-hold
11. fms-state lat-profile-intcpt progrmd
   & afs-state roll-engd not-lnav
   or
   afs-state roll-armed lnav



Conditions Legend

Function level:

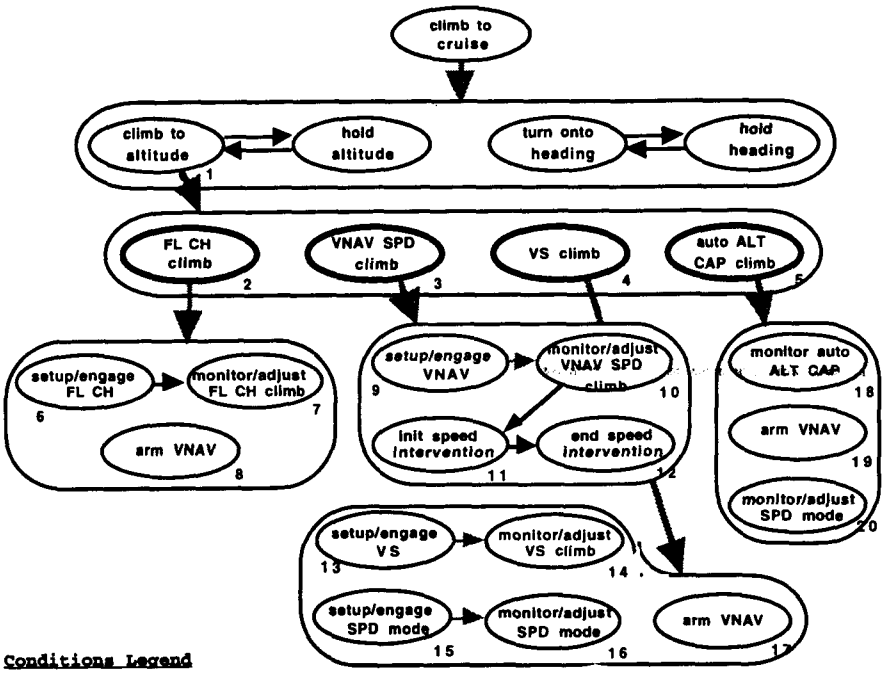
1. acrft-state alt within-limits

Mode Selection level:

2. fms-state vert-profile not-progrmd
   & afs-state mcp-alt outside-limits
   & afs-state cmd-mode cmd
   or
   afs-state mcp-alt within-limits
   & afs-state cmd-mode cmd
   & afs-state pitch-engd alt-hold
3. fms-state vert-profile progrmd
   & afs-state cmd-mode cmd
   & afs-state pitch-engd not-alt-hold
4. fms-state lat-profile not-progrmd
   & afs-state cmd-mode cmd
   & afs-state pitch-engd vs

Task level:

5. afs-state mcp-alt outside-limits
6. afs-state pitch-engd alt-hold
7. afs-state athr-engd spd
8. acrft-state abs-alt at-or-above-1000
   & fms-state vert-profile progrmd
   & fms-state vert-profile-intcpt progrmd
   or
   afs-state roll-armed vnav
9. afs-state pitch-engd not-vnav
   & afs-state pitch-armed not-vnav
10. afs-state pitch-engd vnav-path
11. fms-state vnav-spd-int off
    & fms-state tgt-spd outside-limits
    & afs-state pitch-engd vnav
    or
    fms-state vnav-spd-int on
    & afs-state mcp-spd outside-limits
12. fms-state sched-tgt-spd within-limits
    & fms-state tgt-spd within-limits
    & fms-state vnav-spd-int on
    & afs-state pitch-engd vnav
13. afs-state pitch-engd vs
14. afs-state pitch-engd vs
15. afs-state athr-engd spd
16. acrft-state abs-alt at-or-above-1000
    & fms-state vert-profile progrmd
    & fms-state vert-profile-intcpt progrmd
    or
    afs-state roll-armed vnav



Conditions Legend

Function level:

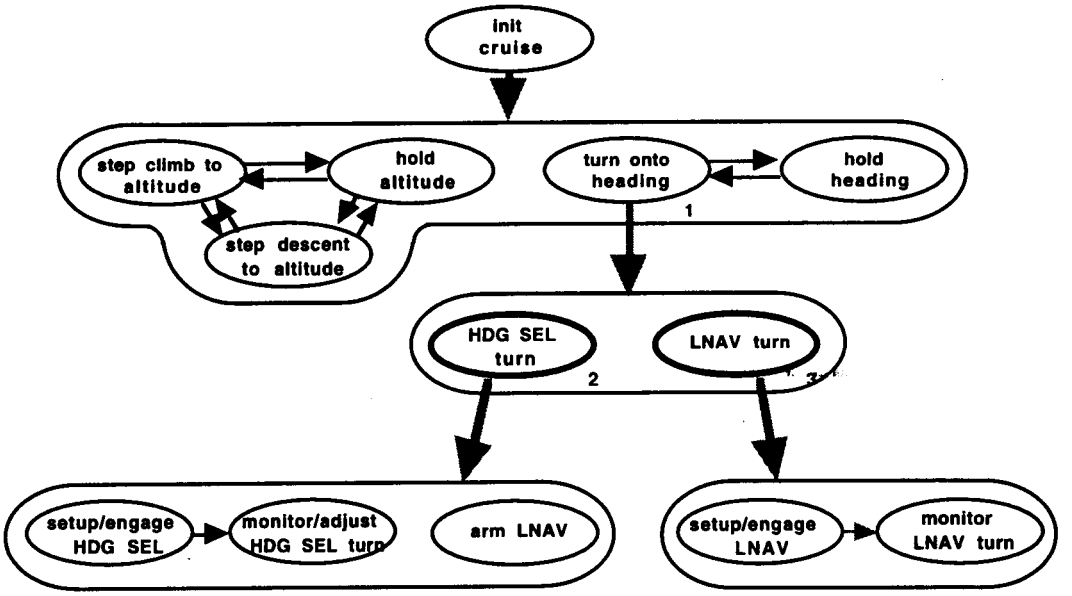
1. acrft-state alt below-limits

Mode Selection level:

2. fms-state vert-profile not-progrmd
   & afs-state cmd-mode cmd
   & acrft-state alt more-than-2000-from-tgt
   & afs-state pitch-engd not-vs
   & afs-state pitch-engd not-alt-cap-rqd-alt
3. fms-state vert-profile progrmd
   & afs-state cmd-mode cmd
   & afs-state tsp clb
   & afs-state pitch-engd not-alt-cap-rqd-alt
4. fms-state vert-profile not-progrmd
   & afs-state cmd-mode cmd
   & acrft-state alt less-than-2000-from-tgt
   & afs-state pitch-engd not-alt-cap-rqd-alt
5. afs-state mcp-alt within-limits
   & afs-state cmd-mode cmd
   & afs-state pitch-engd alt-cap-rqd-alt

Task level:

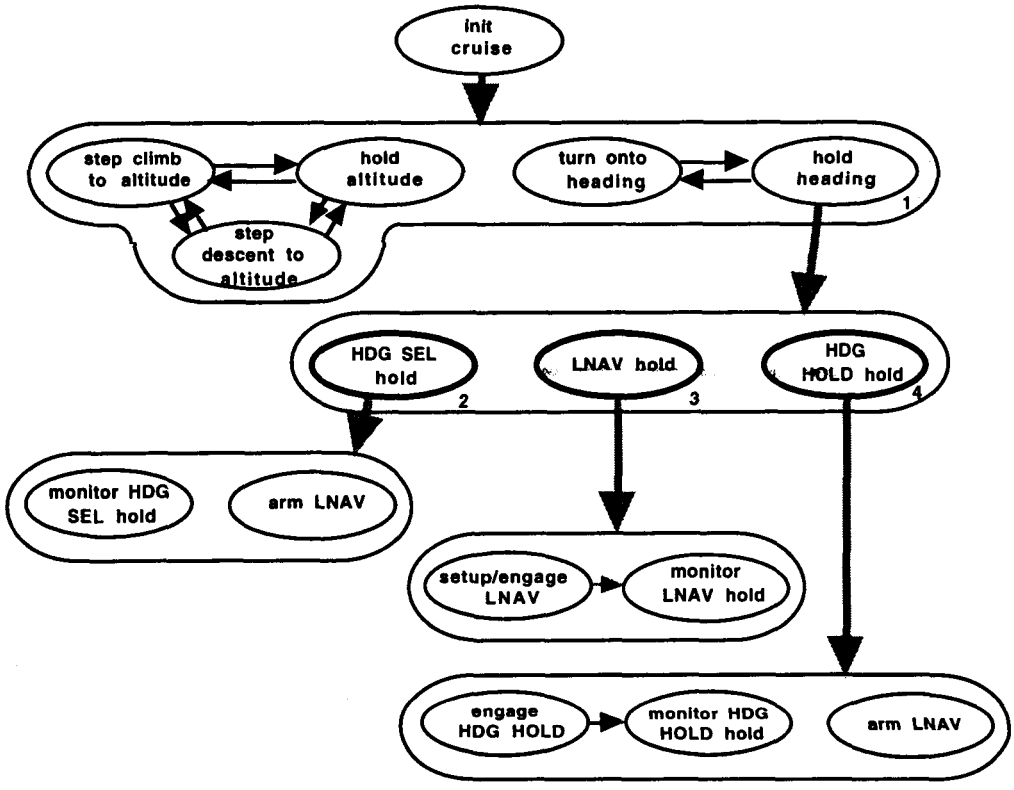
6. afs-state athr-engd not-fl-ch
7. afs-state athr-engd fl-ch
8. acrft-state abs-alt at-or-above-1000
   & fms-state vert-profile progrmd
   & fms-state vert-profile-intcpt progrmd
   or
   afs-state roll-armed vnav
9. afs-state pitch-engd not-vnav
   & afs-state pitch-armed not-vnav
10. afs-state pitch-engd vnav
11. fms-state vnav-spd-int off
    & fms-state tgt-spd outside-limits
    & afs-state pitch-engd vnav
    or
    fms-state vnav-spd-int on
    & afs-state mcp-spd outside-limits
12. fms-state sched-tgt-spd within-limits
    & fms-state tgt-spd within-limits
    & fms-state vnav-spd-int on
    & afs-state pitch-engd vnav
13. afs-state pitch-engd not-vs
    & afs-state pitch-engd not-alt-cap
14. afs-state pitch-engd vs
15. afs-state athr-engd not-spd
16. afs-state athr-engd spd
17. acrft-state abs-alt at-or-above-1000
    & fms-state vert-profile progrmd
    & fms-state vert-profile-intcpt progrmd
    or
    afs-state roll-armed vnav
18. afs-state pitch-engd alt-cap
19. acrft-state abs-alt at-or-above-1000
    & fms-state vert-profile progrmd
    & fms-state vert-profile-intcpt progrmd
    or
    afs-state roll-armed vnav
20. afs-state athr-engd spd



Conditions Legend

Function level:
1. acrft-state hdg outside-limits

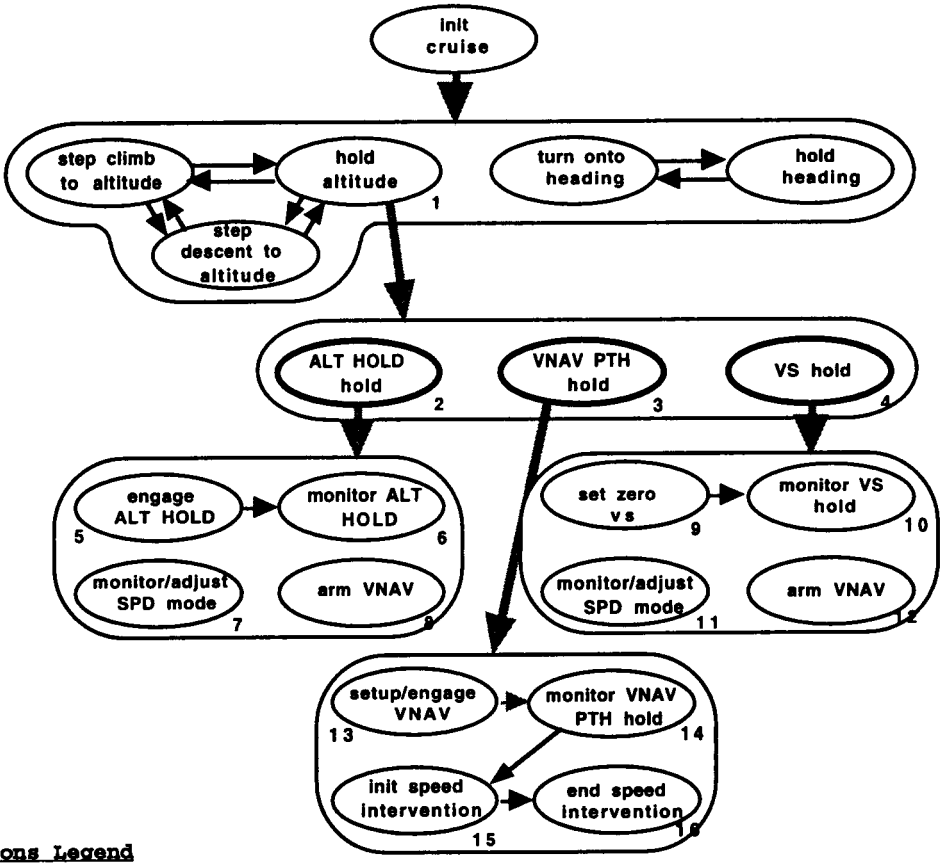
Mode Selection level:
2. fms-state lat-profile not-progrmd
   & afs-state cmd-mode cmd
3. fms-state lat-profile progrmd
   & afs-state cmd-mode cmd



Conditions Legend

Function level:
1. acrft-state hdg within-limits

Mode Selection level:
2. fms-state lat-profile not-progrmd
   & afs-state roll-engd hdg-sel
   & afs-state mcp-hdg within-limits
3. fms-state lat-profile progrmd
   & afs-state cmd-mode cmd
4. fms-state lat-profile not-progrmd
   & afs-state cmd-mode cmd
   & afs-state roll-engd not-hdg-sel



Conditions Legend

Function level:

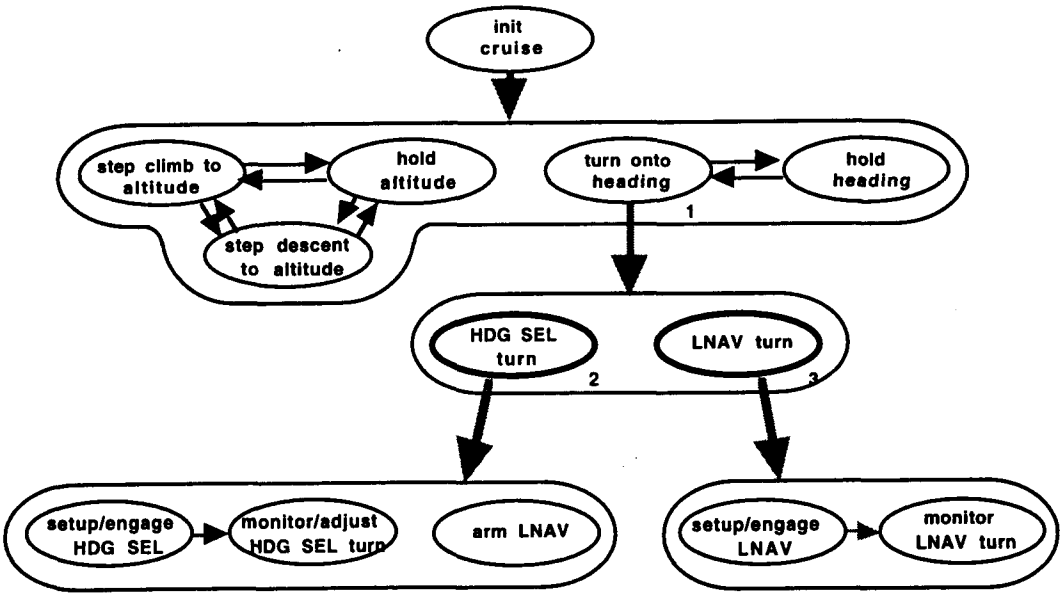
1. acrft-state alt within-limits

Mode Selection level:

2. fms-state vert-profile not-progrmd
   & afs-state mcp-alt outside-limits
   & afs-state cmd-mode cmd
   or
   afs-state mcp-alt within-limits
   & afs-state cmd-mode cmd
   & afs-state pitch-engd alt-hold
3. fms-state vert-profile progrmd
   & afs-state cmd-mode cmd
   & afs-state pitch-engd not-alt-hold
4. fms-state lat-profile not-progrmd
   & afs-state cmd-mode cmd
   & afs-state pitch-engd vs

Task level:

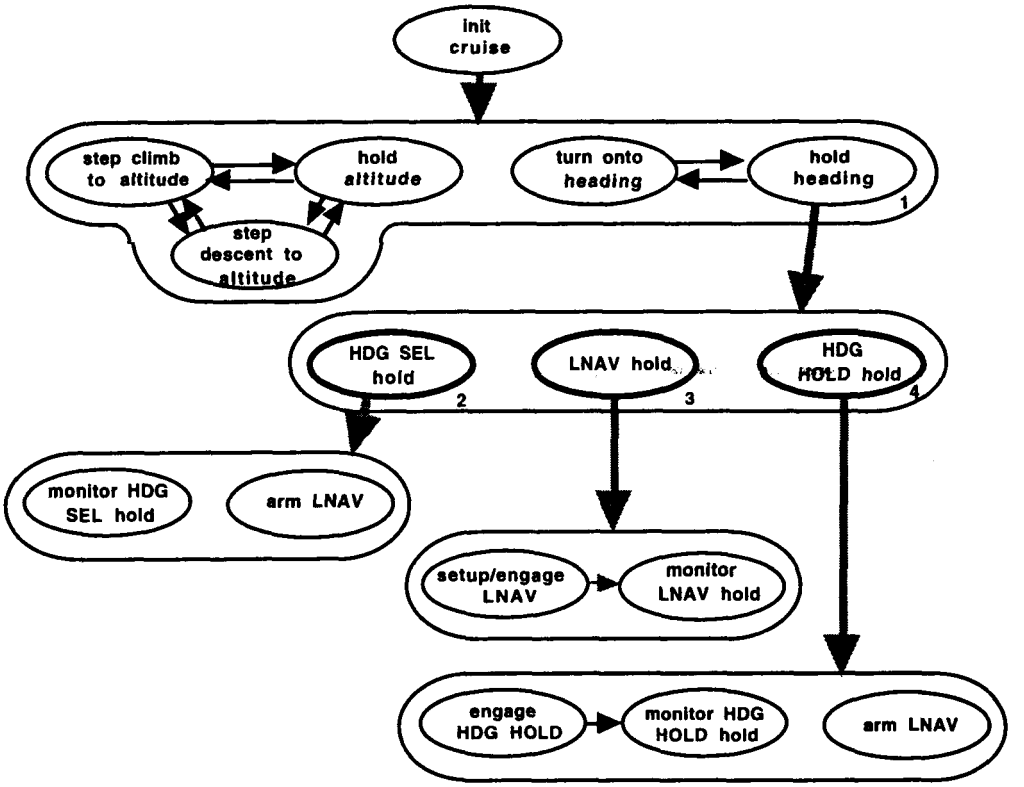
5. afs-state mcp-alt outside-limits
6. afs-state pitch-engd alt-hold
7. afs-state athr-engd spd
8. acrft-state abs-alt at-or-above-1000
   & fms-state vert-profile progrmd
   & fms-state vert-profile-intcpt progrmd
   or
   afs-state roll-armed vnav
9. afs-state pitch-engd vs
10. afs-state pitch-engd vs
11. afs-state athr-engd spd
12. acrft-state abs-alt at-or-above-1000
    & fms-state vert-profile progrmd
    & fms-state vert-profile-intcpt progrmd
    or
    afs-state roll-armed vnav
13. afs-state pitch-engd not-vnav
    & afs-state pitch-armed not-vnav
14. afs-state pitch-engd vnav-path
15. fms-state vnav-spd-int off
    & fms-state tgt-spd outside-limits
    & afs-state pitch-engd vnav
    or
    fms-state vnav-spd-int on
    & afs-state mcp-spd outside-limits
16. fms-state sched-tgt-spd within-limits
    & fms-state tgt-spd within-limits
    & fms-state vnav-spd-int on
    & afs-state pitch-engd vnav



Conditions Legend

Function level:
1. acrft-state hdg outside-limits

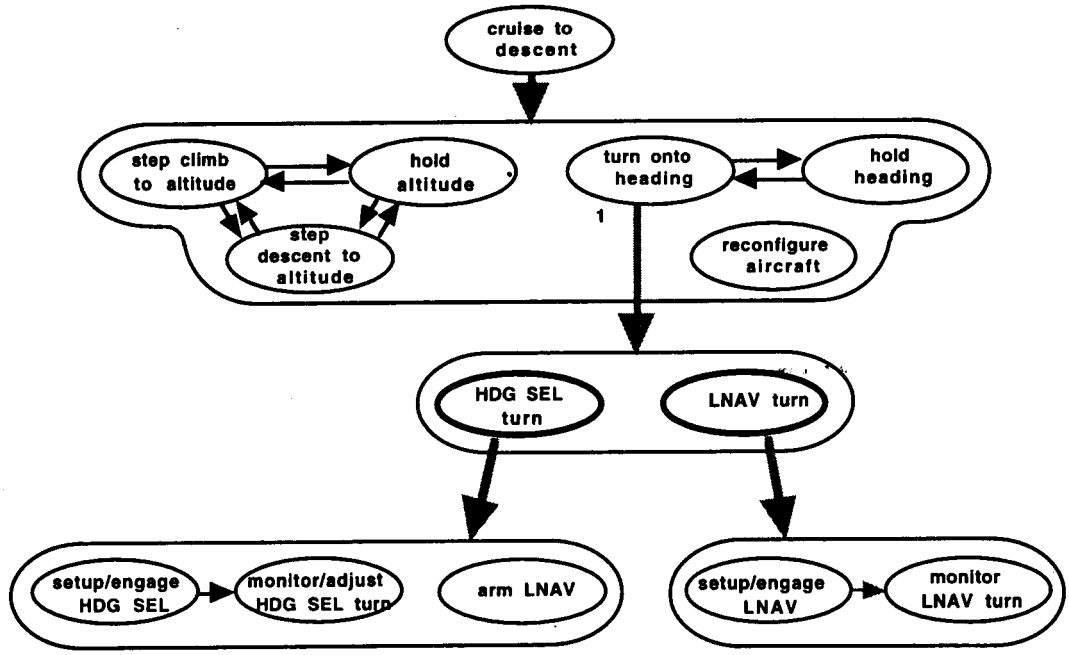
Mode Selection level:
2. fms-state lat-profile not-progrmd
   & afs-state cmd-mode cmd
3. fms-state lat-profile progrmd
   & afs-state cmd-mode cmd



Conditions Legend

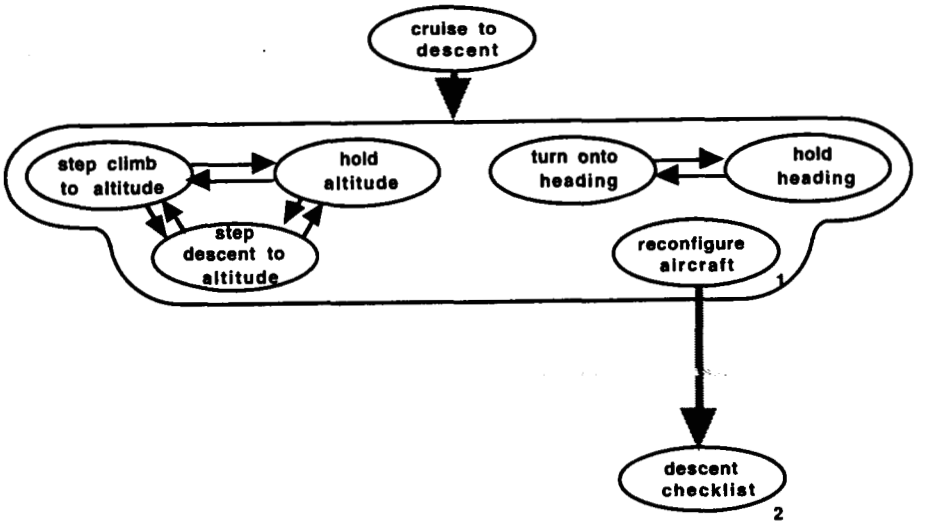
Function level:
1. acrft-state hdg within-limits

Mode Selection level:
2. fms-state lat-profile not-progrmd
   & afs-state roll-engd hdg-sel
   & afs-state mcp-hdg within-limits
3. fms-state lat-profile progrmd
   & afs-state cmd-mode cmd
4. fms-state lat-profile not-progrmd
   & afs-state cmd-mode cmd
   & afs-state roll-engd not-hdg-sel



Conditions Legend

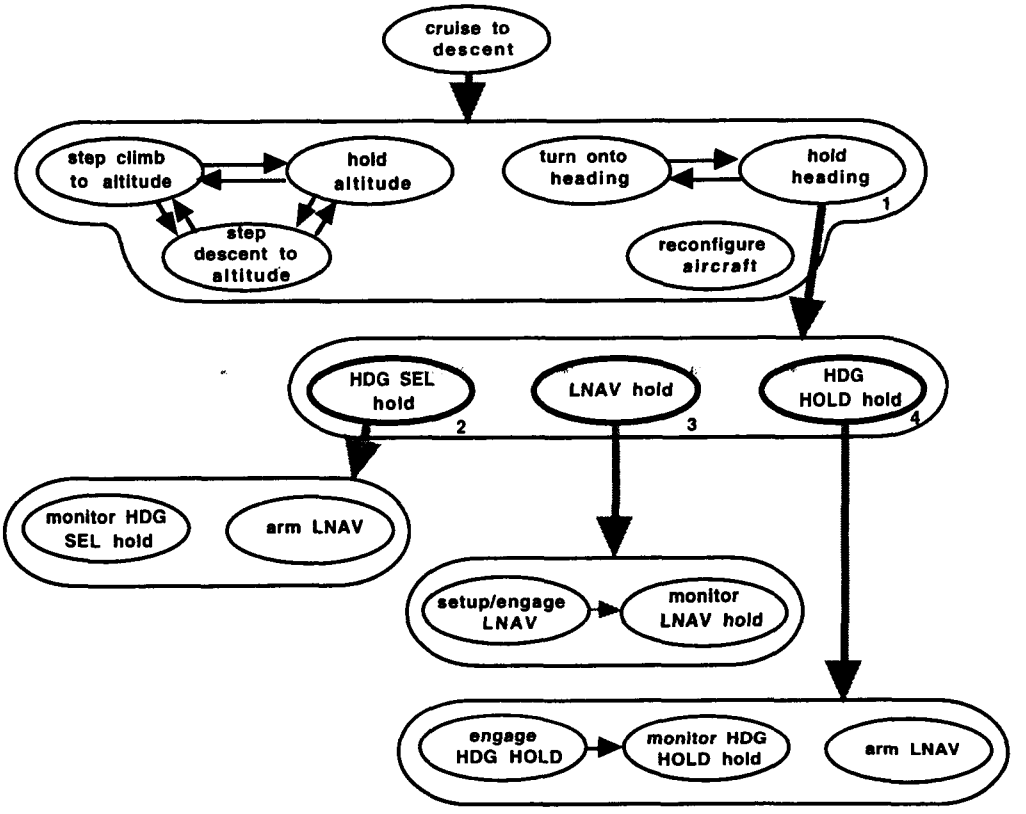
Function level:
1. acrft-state hdg outside-limits



Conditions Legend

Function level:
1. current-phase cruise in-progress

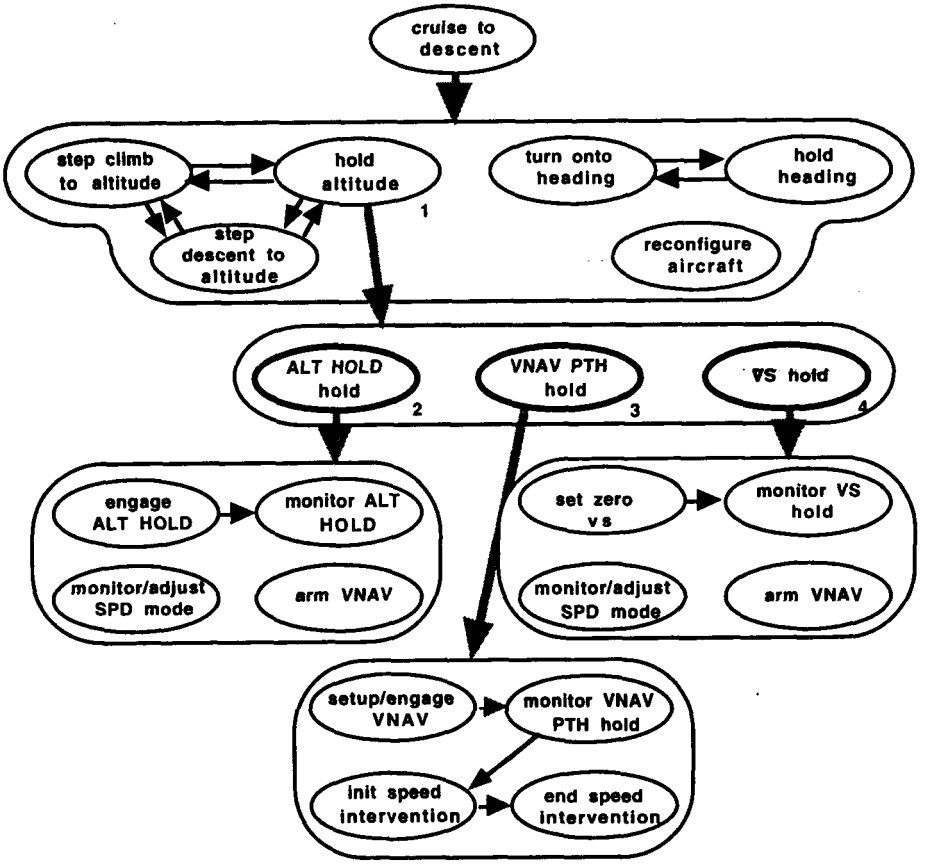
Task level:
2. current-phase cruise in-progress



Conditions Legend

Function level:
1. acrft-state hdg within-limits

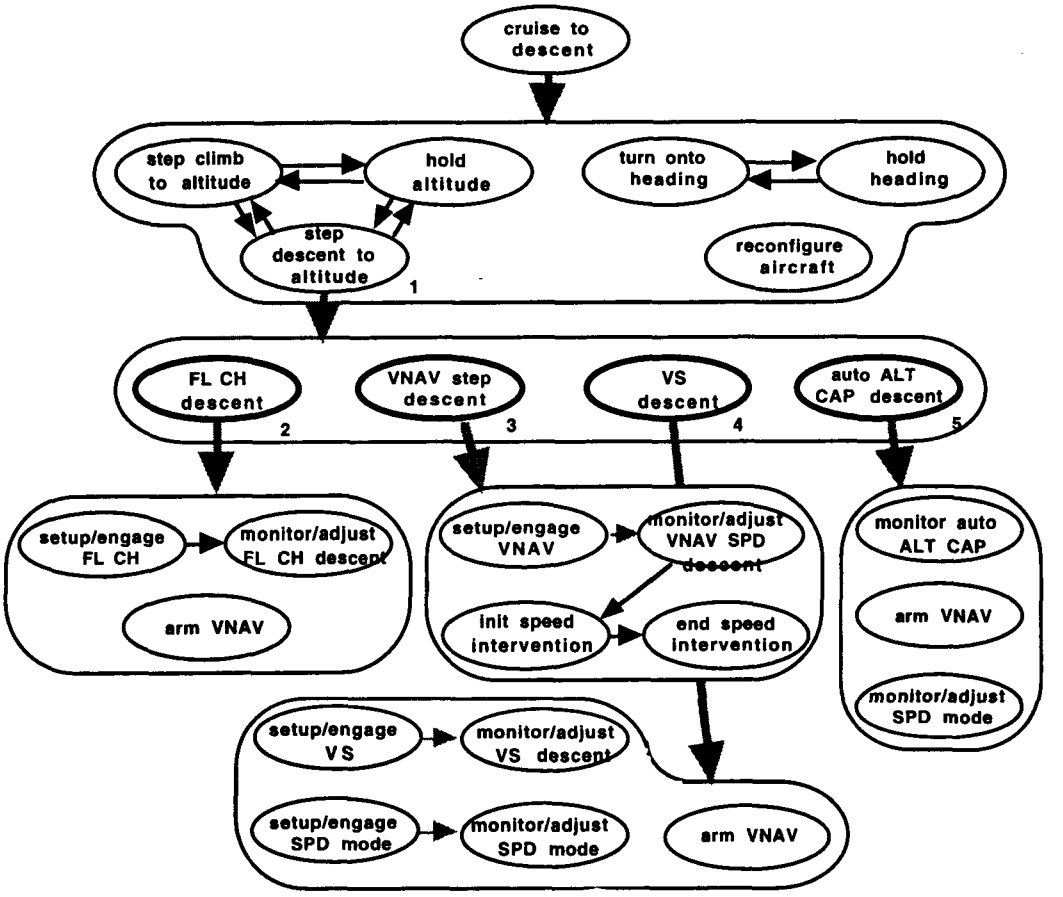
Mode Selection level:
2. fms-state lat-profile not-progrmd
   & afs-state roll-engd hdg-sel
   & afs-state mcp-hdg within-limits
3. fms-state lat-profile progrmd
   & afs-state cmd-mode cmd
4. fms-state lat-profile not-progrmd
   & afs-state cmd-mode cmd
   & afs-state roll-engd not-hdg-sel



Conditions Legend

Function level:
1. acrft-state alt within-limits

Mode Selection level:
2. fms-state vert-profile not-progrmd
   & afs-state mcp-alt outside-limits
   & afs-state cmd-mode cmd
   or
   afs-state mcp-alt within-limits
   & afs-state cmd-mode cmd
   & afs-state pitch-engd alt-hold
3. fms-state vert-profile progrmd
   & afs-state cmd-mode cmd
   & afs-state pitch-engd not-alt-hold
4. fms-state lat-profile not-progrmd
   & afs-state cmd-mode cmd
   & afs-state pitch-engd vs



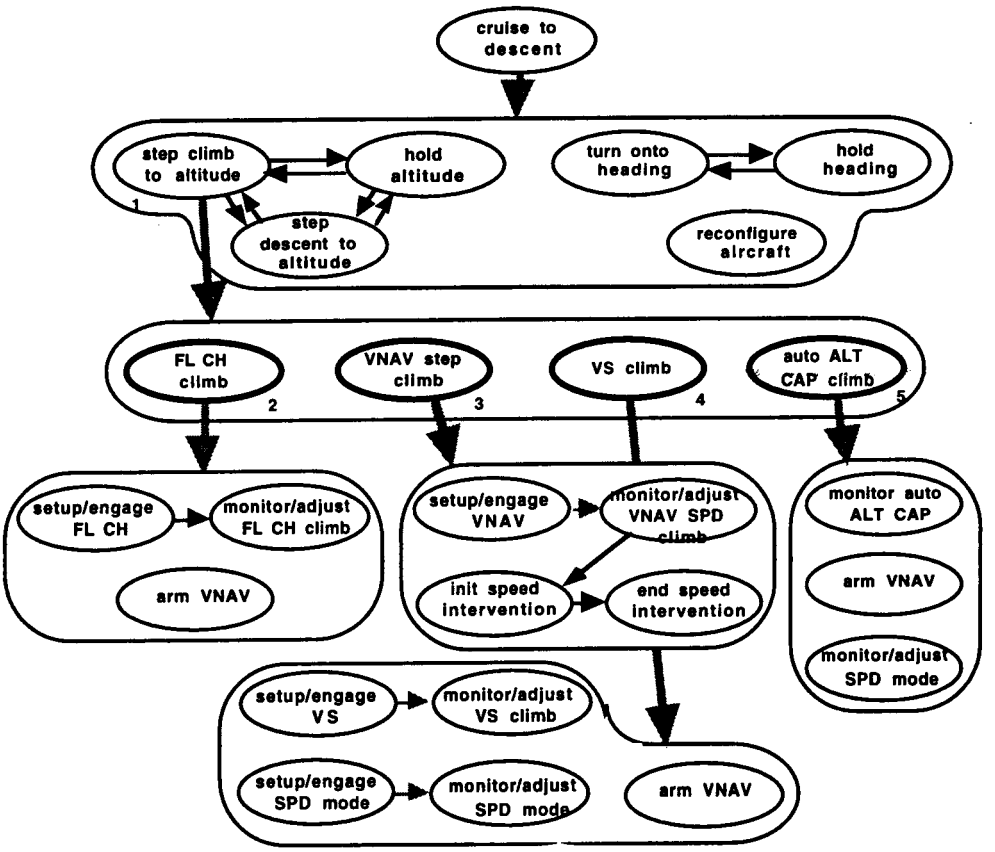
Conditions Legend

Function level:

1. acrft-state alt above-limits

Mode Selection level:

2. fms-state vert-profile not-progrmd
   & afs-state cmd-mode cmd
   & acrft-state alt more-than-2000-from-tgt
   & afs-state pitch-engd not-vs
   & afs-state pitch-engd not-alt-cap-rqd-alt
3. fms-state vert-profile progrmd
   & afs-state cmd-mode cmd
   & afs-state tsp clb
   & afs-state pitch-engd not-alt-cap-rqd-alt
4. fms-state vert-profile not-progrmd
   & afs-state cmd-mode cmd
   & acrft-state alt less-than-2000-from-tgt
   & afs-state pitch-engd not-alt-cap-rqd-alt
5. afs-state mcp-alt within-limits
   & afs-state cmd-mode cmd
   & afs-state pitch-engd alt-cap-rqd-alt



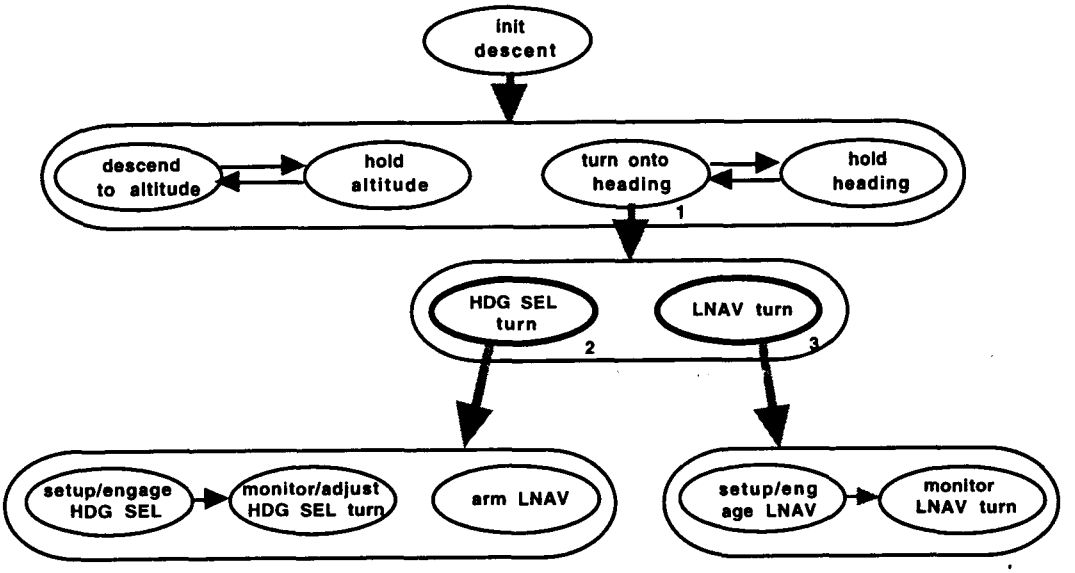
Conditions Legend

Function level:

1. acrft-state alt below-limits

Mode Selection level:

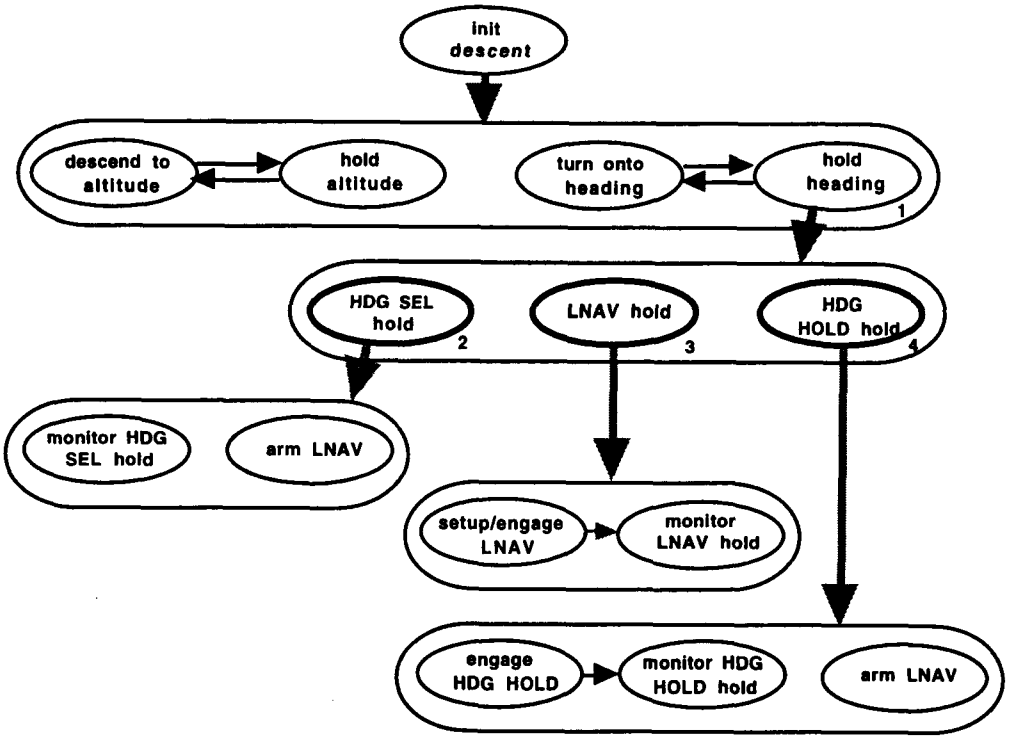
2. fms-state vert-profile not-progrmd
   & afs-state cmd-mode cmd
   & acrft-state alt more-than-2000-from-tgt
   & afs-state pitch-engd not-vs
   & afs-state pitch-engd not-alt-cap-rqd-alt
3. fms-state vert-profile progrmd
   & afs-state cmd-mode cmd
   & afs-state tsp clb
   & afs-state pitch-engd not-alt-cap-rqd-alt
4. fms-state vert-profile not-progrmd
   & afs-state cmd-mode cmd
   & acrft-state alt less-than-2000-from-tgt
   & afs-state pitch-engd not-alt-cap-rqd-alt
5. afs-state mcp-alt within-limits
   & afs-state cmd-mode cmd
   & afs-state pitch-engd alt-cap-rqd-alt



Conditions Legend

Function level:
1. acrft-state hdg outside-limits

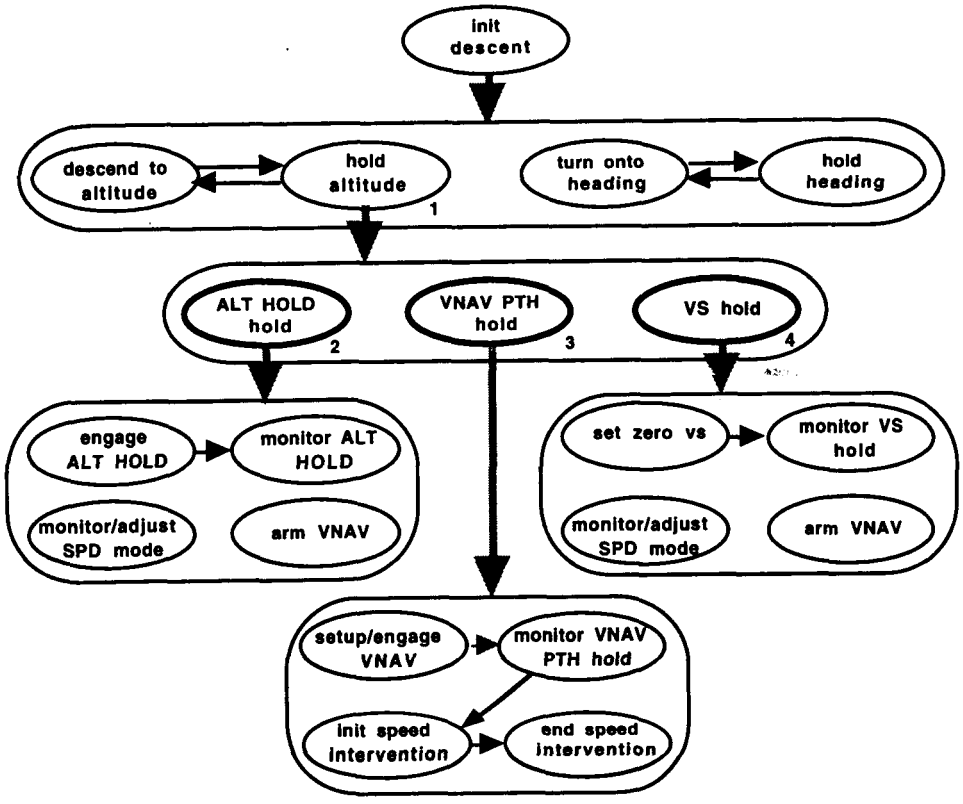
Mode Selection level:
2. fms-state lat-profile not-progrmd
   & afs-state cmd-mode cmd
3. fms-state lat-profile progrmd
   & afs-state cmd-mode cmd



Conditions Legend

Function level:
1. acrft-state hdg within-limits

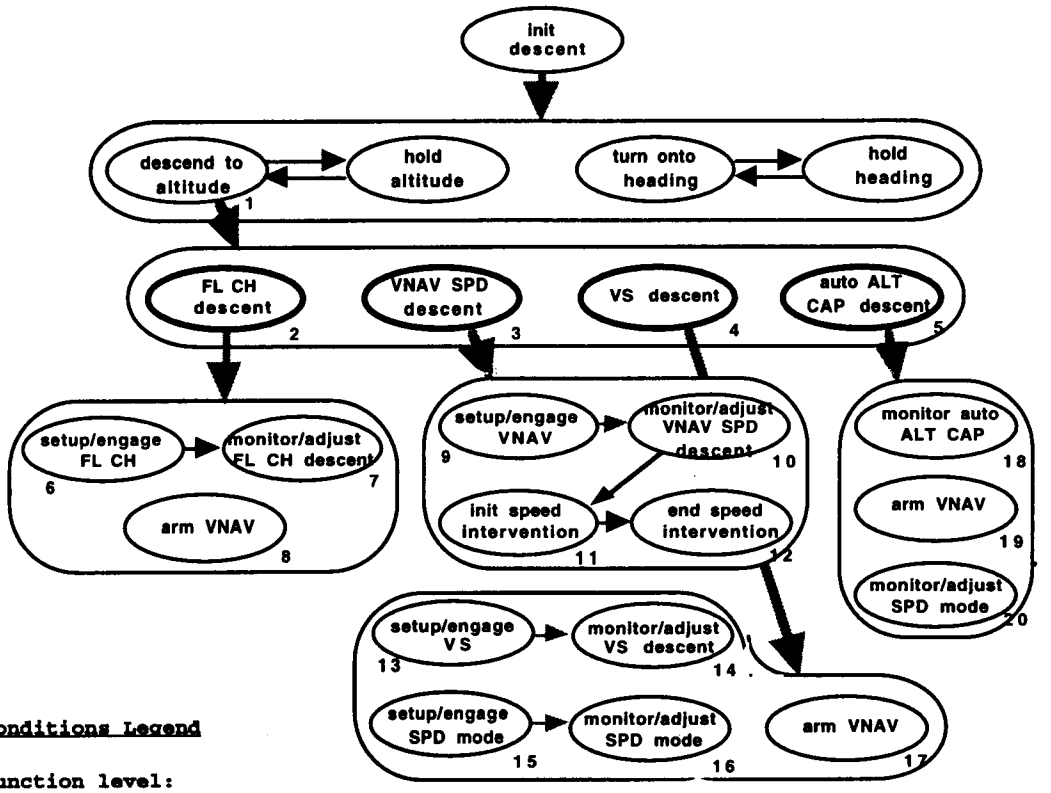
Mode Selection level:
2. fms-state lat-profile not-progrmd
   & afs-state roll-engd hdg-sel
   & afs-state mcp-hdg within-limits
3. fms-state lat-profile progrmd
   & afs-state cmd-mode cmd
4. fms-state lat-profile not-progrmd
   & afs-state cmd-mode cmd
   & afs-state roll-engd not-hdg-sel



Conditions Legend

Function level:
1. acrft-state alt within-limits

Mode Selection level:
2. fms-state vert-profile not-progrmd
   & afs-state mcp-alt outside-limits
   & afs-state cmd-mode cmd
   or
   afs-state mcp-alt within-limits
   & afs-state cmd-mode cmd
   & afs-state pitch-engd alt-hold
3. fms-state vert-profile progrmd
   & afs-state cmd-mode cmd
   & afs-state pitch-engd not-alt-hold
4. fms-state lat-profile not-progrmd
   & afs-state cmd-mode cmd
   & afs-state pitch-engd vs



Conditions Legend

Function level:

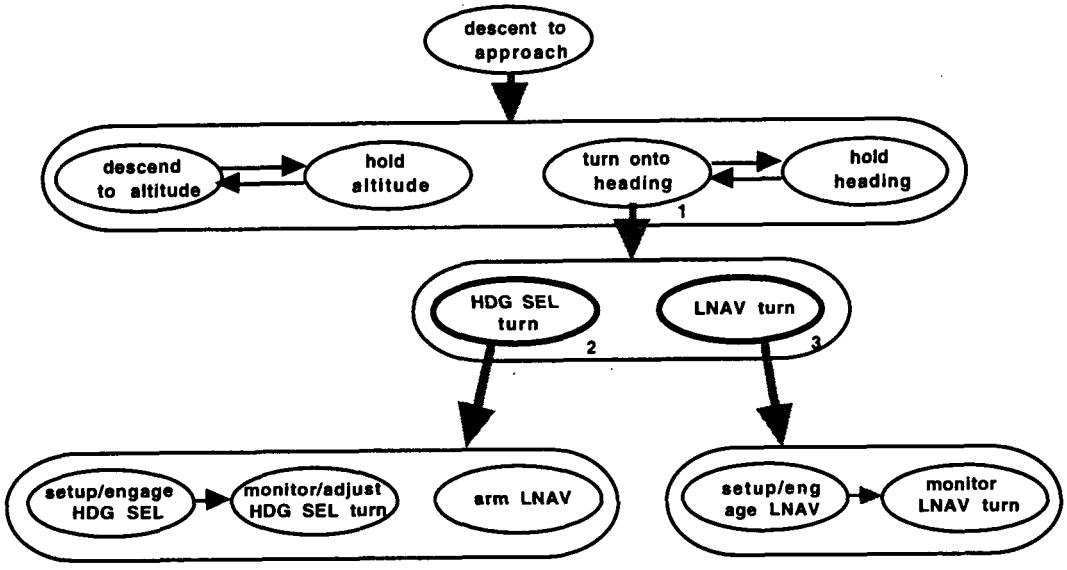
1. acrft-state alt below-limits

Mode Selection level:

2. fms-state vert-profile not-progrmd
   & afs-state cmd-mode cmd
   & acrft-state alt more-than-2000-from-tgt
   & afs-state pitch-engd not-vs
   & afs-state pitch-engd not-alt-cap-rqd-alt
3. fms-state vert-profile progrmd
   & afs-state cmd-mode cmd
   & afs-state tsp clb
   & afs-state pitch-engd not-alt-cap-rqd-alt
4. fms-state vert-profile not-progrmd
   & afs-state cmd-mode cmd
   & acrft-state alt less-than-2000-from-tgt
   & afs-state pitch-engd not-alt-cap-rqd-alt
5. afs-state mcp-alt within-limits
   & afs-state cmd-mode cmd
   & afs-state pitch-engd alt-cap-rqd-alt

Task level:

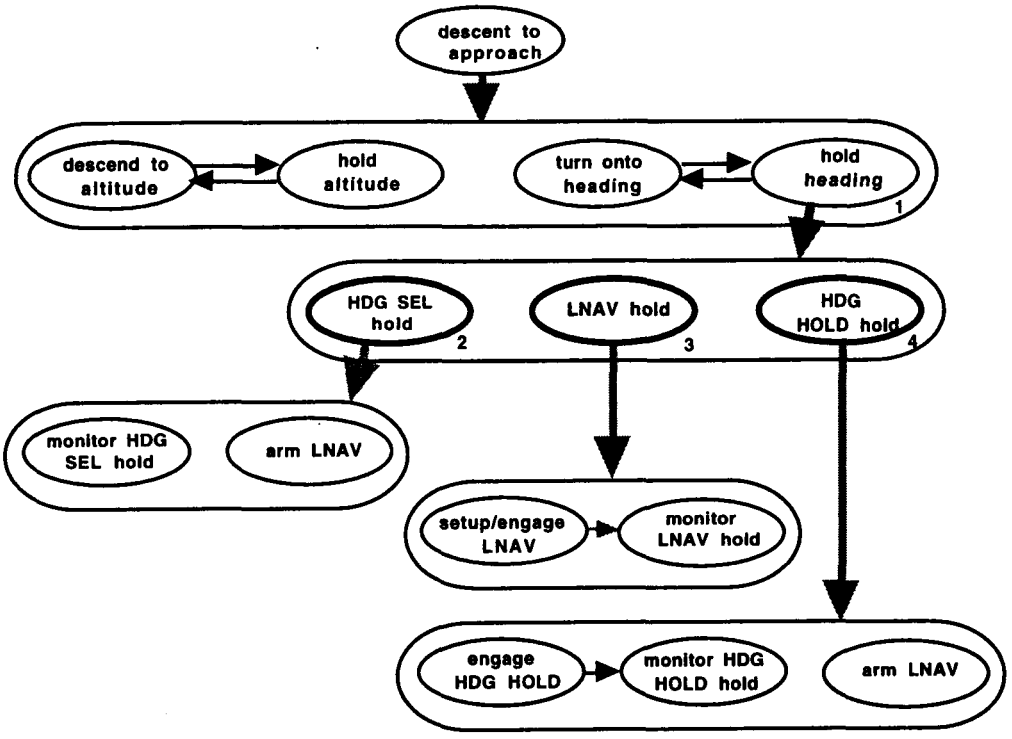
6. afs-state athr-engd not-fl-ch
7. afs-state athr-engd fl-ch
8. acrft-state abs-alt at-or-above-1000
   & fms-state vert-profile progrmd
   & fms-state vert-profile-intcpt progrmd
   or
   afs-state roll-armed vnav
9. afs-state pitch-engd not-vnav
   & afs-state pitch-armed not-vnav
10. afs-state pitch-engd vnav
11. fms-state vnav-spd-int off
    & fms-state tgt-spd outside-limits
    & afs-state pitch-engd vnav
    or
    fms-state vnav-spd-int on
    & afs-state mcp-spd outside-limits
12. fms-state sched-tgt-spd within-limits
    & fms-state tgt-spd within-limits
    & fms-state vnav-spd-int on
    & afs-state pitch-engd vnav
13. afs-state pitch-engd not-vs
    & afs-state pitch-engd not-alt-cap
14. afs-state pitch-engd vs
15. afs-state athr-engd not-spd
16. afs-state athr-engd spd
17. acrft-state abs-alt at-or-above-1000
    & fms-state vert-profile progrmd
    & fms-state vert-profile-intcpt progrmd
    or
    afs-state roll-armed vnav
18. afs-state pitch-engd alt-cap
19. acrft-state abs-alt at-or-above-1000
    & fms-state vert-profile progrmd
    & fms-state vert-profile-intcpt progrmd
    or
    afs-state roll-armed vnav
20. afs-state athr-engd spd



Conditions Legend

Function level:
1. acrft-state hdg outside-limits

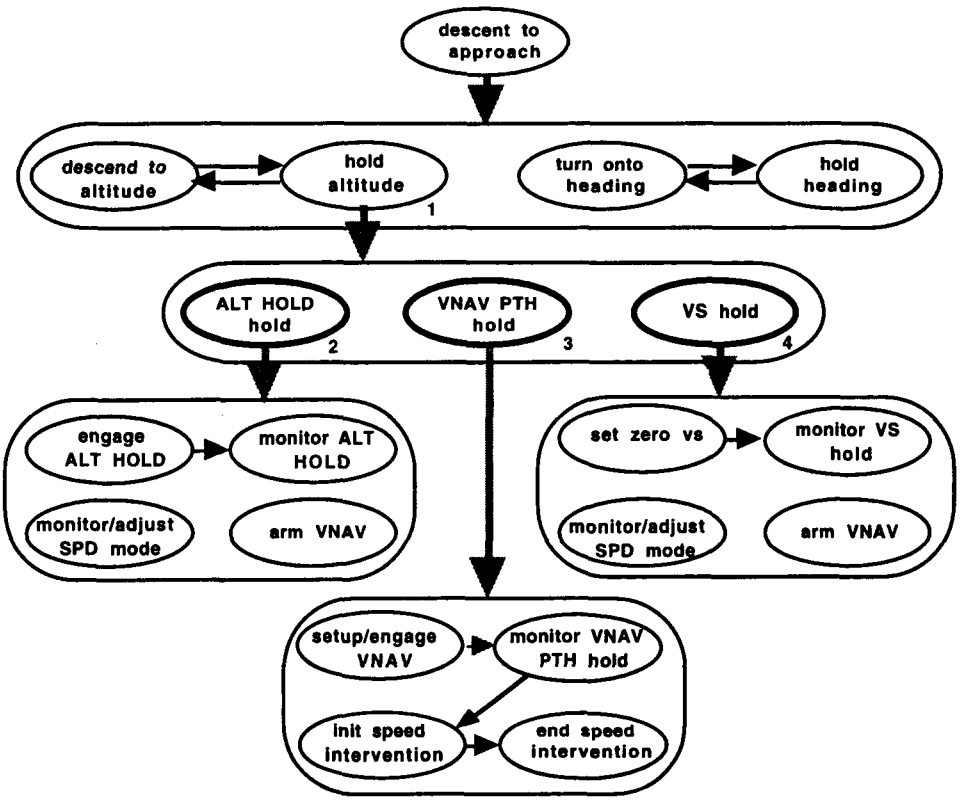
Mode Selection level:
2. fms-state lat-profile not-progrmd
   & afs-state cmd-mode cmd
3. fms-state lat-profile progrmd
   & afs-state cmd-mode cmd



Conditions Legend

Function level:
1. acrft-state hdg within-limits

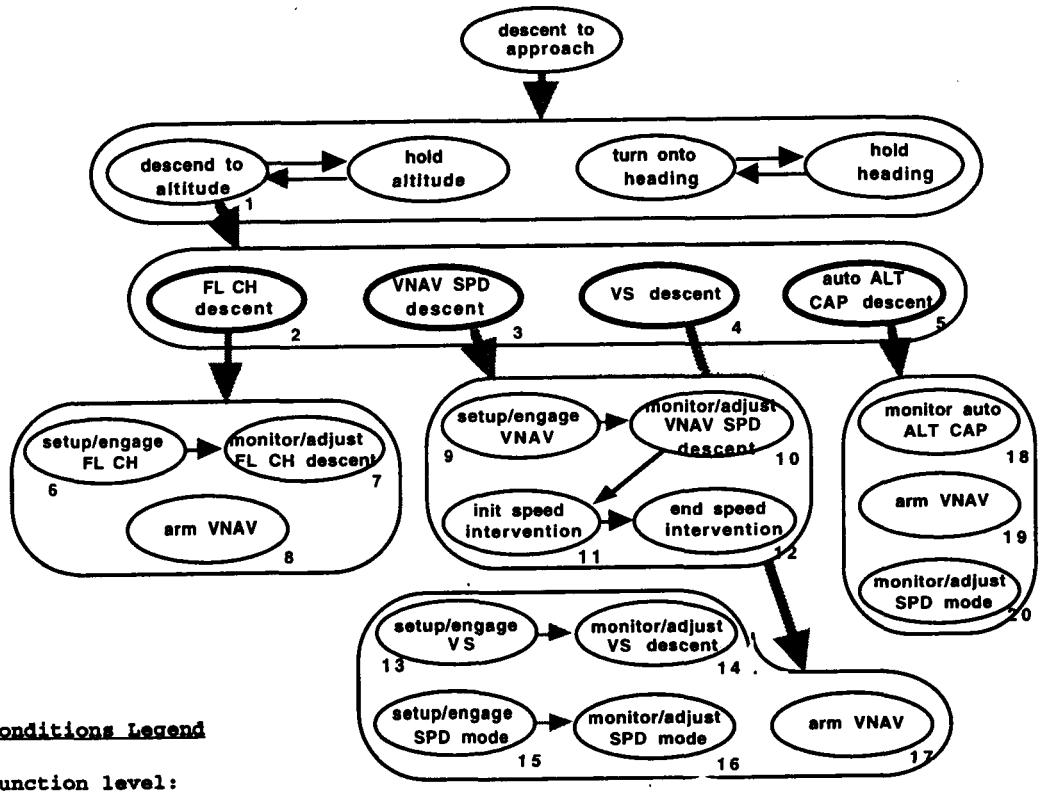
Mode Selection level:
2. fms-state lat-profile not-progrmd
   & afs-state roll-engd hdg-sel
   & afs-state mcp-hdg within-limits
3. fms-state lat-profile progrmd
   & afs-state cmd-mode cmd
4. fms-state lat-profile not-progrmd
   & afs-state cmd-mode cmd
   & afs-state roll-engd not-hdg-sel



Conditions Legend

Function level:
1. acrft-state alt within-limits

Mode Selection level:
2. fms-state vert-profile not-progrmd
   & afs-state mcp-alt outside-limits
   & afs-state cmd-mode cmd
   or
   afs-state mcp-alt within-limits
   & afs-state cmd-mode cmd
   & afs-state pitch-engd alt-hold
3. fms-state vert-profile progrmd
   & afs-state cmd-mode cmd
   & afs-state pitch-engd not-alt-hold
4. fms-state lat-profile not-progrmd
   & afs-state cmd-mode cmd
   & afs-state pitch-engd vs



Conditions Legend

Function level:

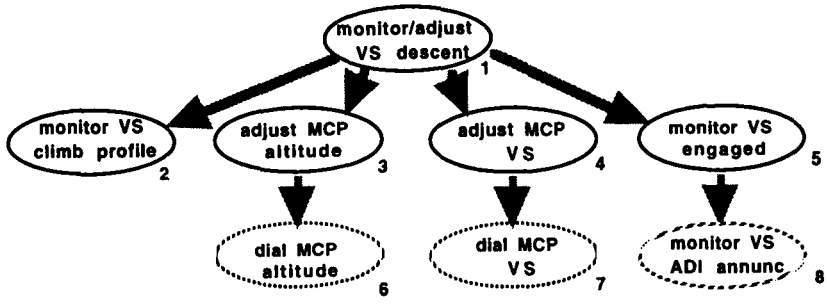
1. acrft-state alt below-limits

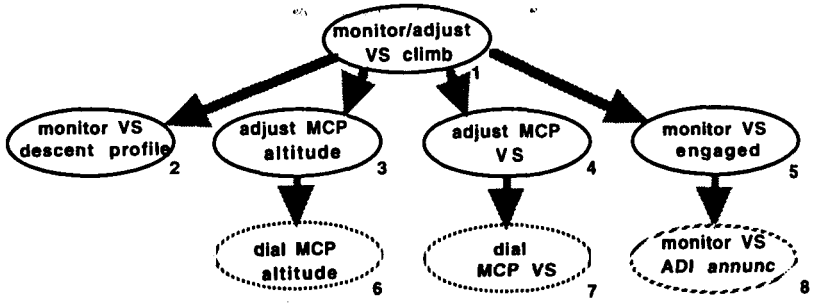
Mode Selection level:

2. fms-state vert-profile not-progrmd
   & afs-state cmd-mode cmd
   & acrft-state alt more-than-2000-from-tgt
   & afs-state pitch-engd not-vs
   & afs-state pitch-engd not-alt-cap-rqd-alt
3. fms-state vert-profile progrmd
   & afs-state cmd-mode cmd
   & afs-state tsp clb
   & afs-state pitch-engd not-alt-cap-rqd-alt
4. fms-state vert-profile not-progrmd
   & afs-state cmd-mode cmd
   & acrft-state alt less-than-2000-from-tgt
   & afs-state pitch-engd not-alt-cap-rqd-alt
5. afs-state mcp-alt within-limits
   & afs-state cmd-mode cmd
   & afs-state pitch-engd alt-cap-rqd-alt

Task level:

6. afs-state athr-engd not-fl-ch
7. afs-state athr-engd fl-ch
8. acrft-state abs-alt at-or-above-1000
   & fms-state vert-profile progrmd
   & fms-state vert-profile-intcpt progrmd
   or
   afs-state roll-armed vnav
9. afs-state pitch-engd not-vnav
   & afs-state pitch-armed not-vnav
10. afs-state pitch-engd vnav
11. fms-state vnav-spd-int off
    & fms-state tgt-spd outside-limits
    & afs-state pitch-engd vnav
    or
    fms-state vnav-spd-int on
    & afs-state mcp-spd outside-limits
12. fms-state sched-tgt-spd within-limits
    & fms-state tgt-spd within-limits
    & fms-state vnav-spd-int on
    & afs-state pitch-engd vnav
13. afs-state pitch-engd not-vs
    & afs-state pitch-engd not-alt-cap
14. afs-state pitch-engd vs
15. afs-state athr-engd not-spd
16. afs-state athr-engd spd
17. acrft-state abs-alt at-or-above-1000
    & fms-state vert-profile progrmd
    & fms-state vert-profile-intcpt progrmd
    or
    afs-state roll-armed vnav
18. afs-state pitch-engd alt-cap
19. acrft-state abs-alt at-or-above-1000
    & fms-state vert-profile progrmd
    & fms-state vert-profile-intcpt progrmd
    or
    afs-state roll-armed vnav
20. afs-state athr-engd spd



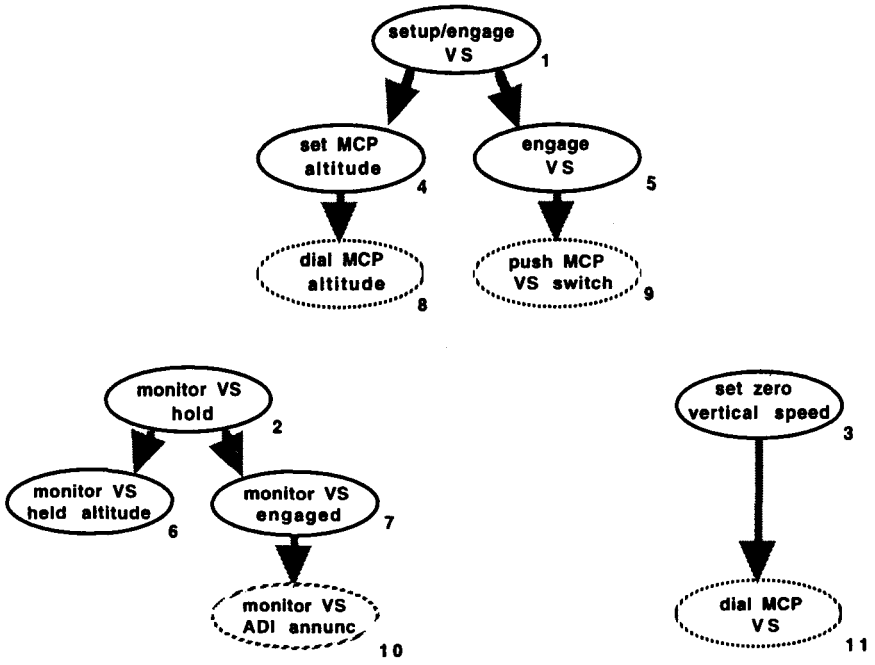


Conditions Legend

Task level:
1. afs-state pitch-engd vs

Subtask level:
2. afs-state pitch-engd vs
3. afs-state mcp-alt outside-limits
4. afs-state mcp-vs outside-limits
5. afs-state pitch-engd vs

Action level:
6. afs-state mcp-alt outside-limits
7. afs-state mcp-vs outside-limits
8. afs-state pitch-engd vs

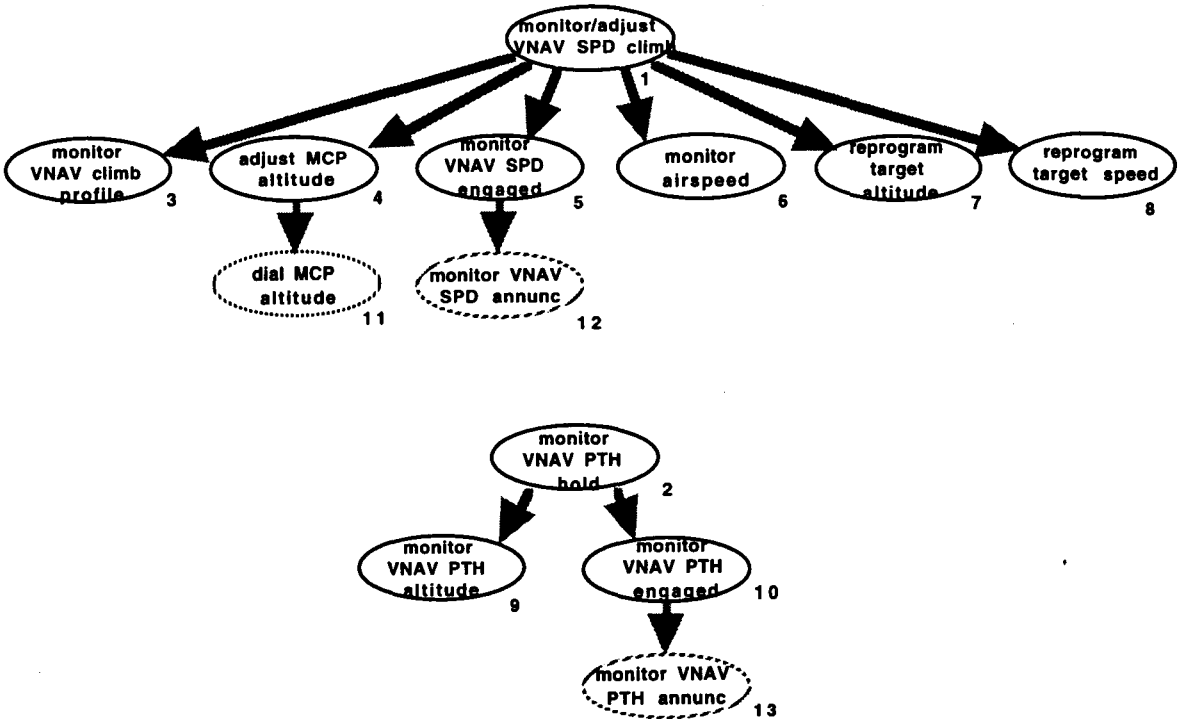


Conditions Legend

Task level:
1. afs-state pitch-engd not-vs
   & afs-state pitch-engd not-alt-cap
2. afs-state pitch-engd vs
3. afs-state pitch-engd vs

Subtask level:
4. afs-state mcp-alt outside-limits
5. afs-state mcp-alt within-limits
6. afs-state pitch-engd vs
7. afs-state pitch-engd vs

Action level:
8. afs-state mcp-alt outside-limits
9. afs-state mcp-alt within-limits
10. afs-state pitch-engd vs
11. afs-state pitch-engd vs

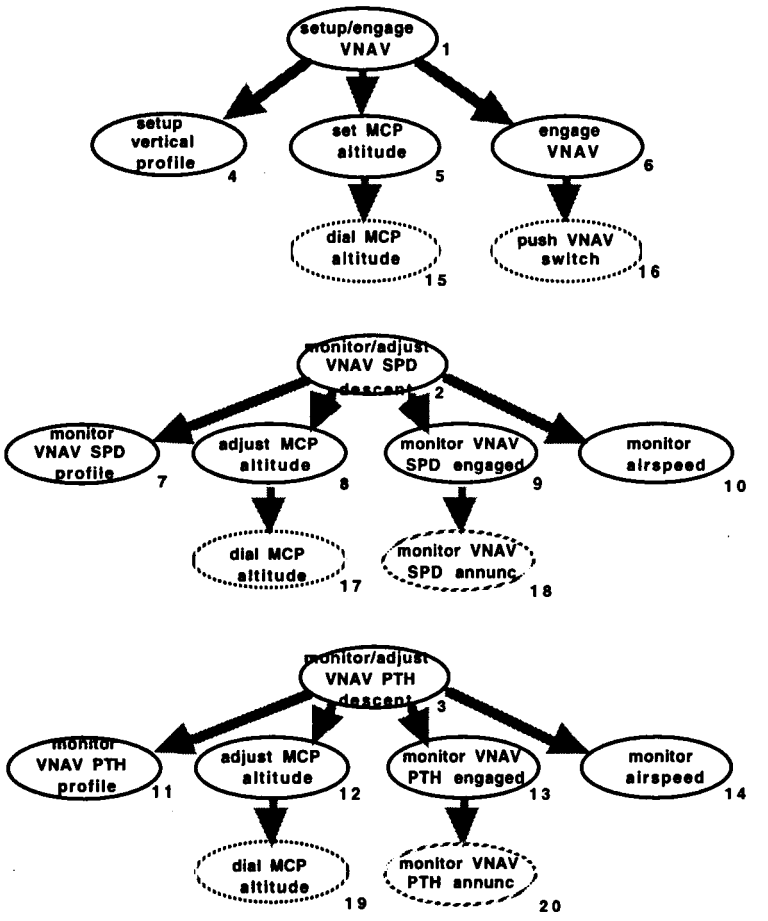


Conditions Legend

Task level:
1. afs-state pitch-engd vnav
2. afs-state pitch-engd vnav-path

Subtask level:
3. afs-state pitch-engd vnav
4. afs-state mcp-alt outside-limits
5. afs-state pitch-engd vnav
6. afs-state pitch-engd vnav
7. fms-state tgt-alt outside-limits
8. fms-state tgt-spd outside-limits
9. afs-state pitch-engd vnav-path
10. afs-state pitch-engd vnav-path

Action level:
11. afs-state mcp-alt outside-limits
12. afs-state pitch-engd vnav
13. afs-state pitch-engd vnav-path

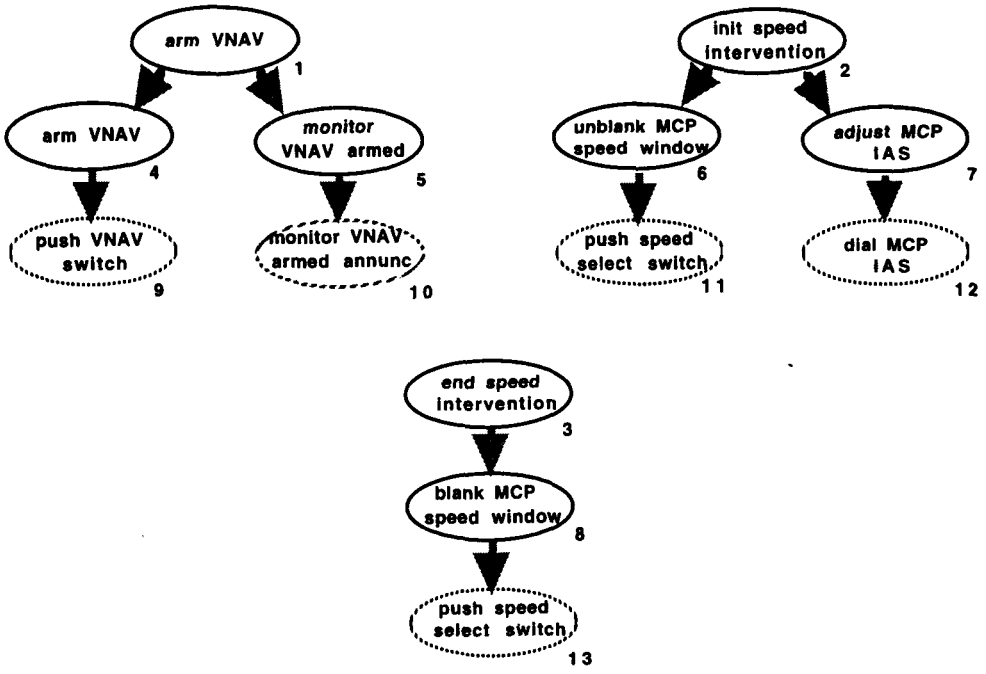


Conditions Legend

Task level:
1. afs-state pitch-engd not-vnav
   & afs-state pitch-armed not-vnav
2. afs-state pitch-engd vnav-spd
3. afs-state pitch-engd vnav-path

Subtask level:
4. fms-state vert-profile not-progrmd
5. afs-state mcp-alt outside-limits
6. fms-state vert-profile progrmd
7. afs-state pitch-engd vnav-spd
8. afs-state mcp-alt outside-limits
9. afs-state pitch-engd vnav-spd
10. afs-state pitch-engd vnav-spd
11. afs-state pitch-engd vnav-path
12. afs-state mcp-alt outside-limits
13. afs-state pitch-engd vnav-path
14. afs-state pitch-engd vnav-path

Action level:
15. afs-state mcp-alt outside-limits
16. fms-state vert-profile progrmd
17. afs-state mcp-alt outside-limits
18. afs-state pitch-engd vnav-spd
19. afs-state mcp-alt outside-limits
20. afs-state pitch-engd vnav-path



Conditions Legend

Task level:

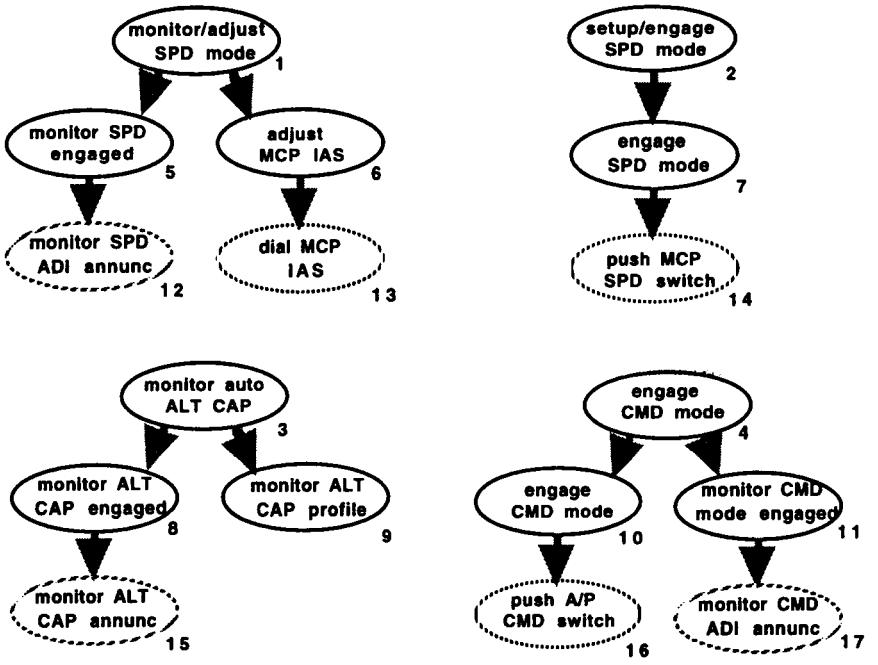
1. fms-state vert-profile progrmd
   & fms-state vert-profile-intcpt progrmd
   or
   afs-state roll-armed vnav
2. fms-state vnav-spd-int off
   & fms-state tgt-spd outside-limits
   & afs-state pitch-engd vnav
   or
   fms-state vnav-spd-int on
   afs-state mcp-spd outside-limits
3. fms-state sched-tgt-spd within-limits
   & fms-state tgt-spd within-limits
   & fms-state vnav-spd-int on
   & afs-state pitch-engd vnav

Subtask level:

4. afs-state roll-armed not-vnav
   & fms-state vert-profile-intcpt progrmd
5. afs-state roll-armed vnav
   & fms-state vert-profile-intcpt progrmd
6. fms-state vnav-spd-int *off*
7. fms-state vnav-spd-int on
   & afs-state mcp-spd outside-limits
8. fms-state sched-tgt-spd within-limits
   & fms-state tgt-spd within-limits

Action level:

9. afs-state roll-armed not-vnav
   & fms-state vert-profile-intcpt progrmd
10. afs-state roll-armed vnav
    & fms-state vert-profile-intcpt progrmd
11. fms-state vnav-spd-int *off*
12. afs-state mcp-spd outside-limits
13. fms-state sched-tgt-spd within-limits
    & fms-state tgt-spd within-limits



Conditions Legend

Task level:

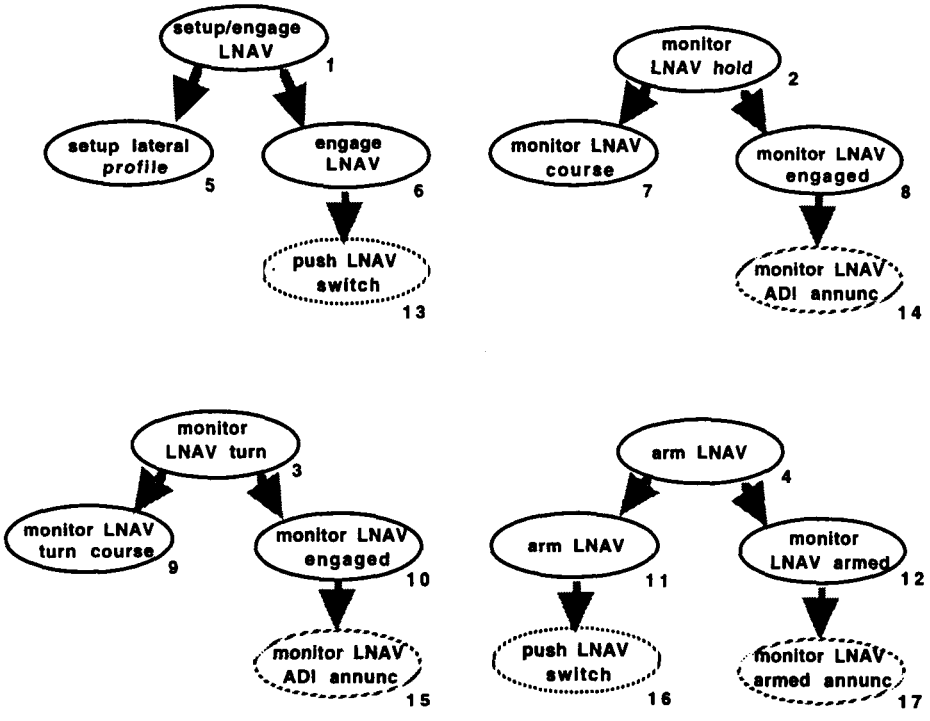
1. afs-state athr-engd spd
2. afs-state athr-engd not-spd
3. afs-state pitch-engd alt-cap
4. afs-state cmd-mode fd
   or
   afs-state cmd-mode cmd

Subtask level:

5. afs-state athr-engd spd
6. afs-state mcp-spd outside-limits
7. afs-state athr-engd not-spd
8. afs-state pitch-engd alt-cap
9. afs-state pitch-engd alt-cap
10. afs-state cmd-mode fd
11. afs-state cmd-mode cmd

Action level:

12. afs-state athr-engd spd
13. afs-state mcp-spd outside-limits
14. afs-state athr-engd not-spd
15. afs-state pitch-engd alt-cap
16. afs-state cmd-mode fd
17. afs-state cmd-mode cmd



Conditions Legend

Task level:

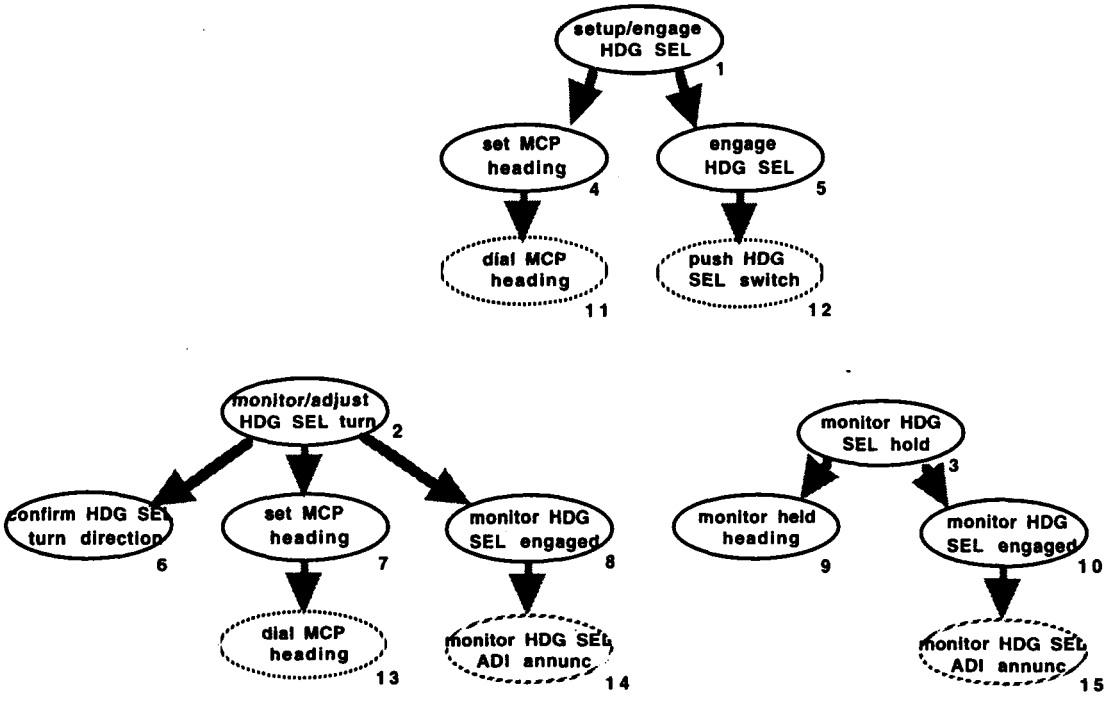
1. afs-state roll-engd not-lnav
   & afs-state roll-armed not-lnav
2. afs-state roll-engd lnav
3. afs-state roll-engd lnav
4. fms-state lat-profile progrmd
   & afs-state roll-engd not-lnav
   or
   afs-state roll-armed lnav

Subtask level:

5. fms-state lat-profile not-progrmd
6. fms-state lat-profile progrmd
7. afs-state roll-engd lnav
8. afs-state roll-engd lnav
9. afs-state roll-engd lnav
10. afs-state roll-engd lnav
11. afs-state roll-armed not-lnav
12. afs-state roll-armed lnav

Action level:

13. fms-state lat-profile progrmd
14. afs-state roll-engd lnav
15. afs-state roll-engd lnav
16. afs-state roll-armed not-lnav
17. afs-state roll-armed lnav

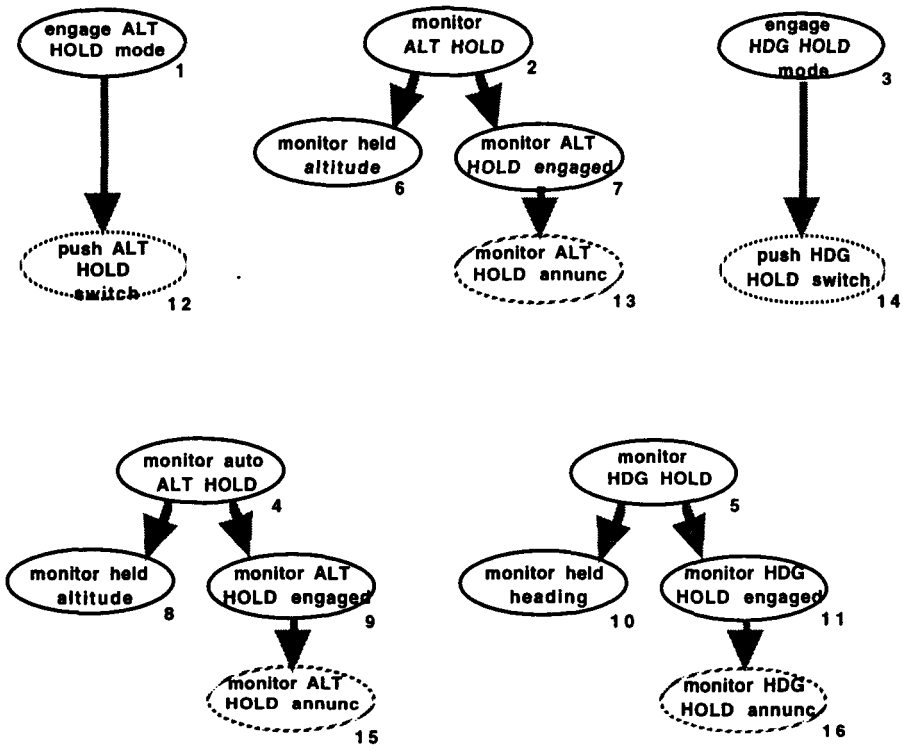


Conditions Legend

Task level:
1. afs-state roll-engd not-hdg-sel
2. afs-state roll-engd hdg-sel
3. afs-state mcp-hdg within-limits

Subtask level:
4. afs-state mcp-hdg outside-limits
5. afs-state roll-engd not-hdg-sel
6. afs-state roll-engd hdg-sel
7. afs-state mcp-hdg outside-limits
8. afs-state roll-engd hdg-sel
9. afs-state roll-engd hdg-sel
10. afs-state roll-engd hdg-sel

Action level:
11. afs-state mcp-hdg outside-limits
12. afs-state mcp-hdg outside-limits
13. afs-state pitch-engd alt-hold
14. afs-state roll-engd hdg-sel
15. afs-state roll-engd hdg-sel



Conditions Legend

Task level:

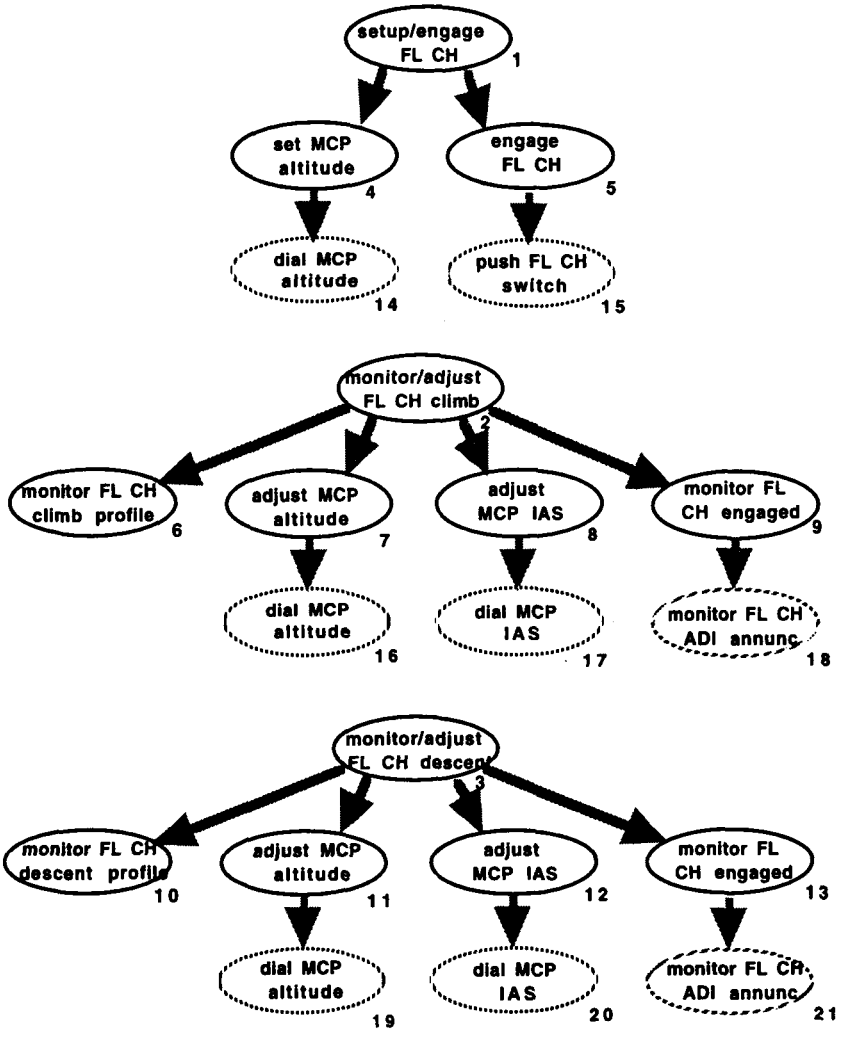
1. afs-state mcp-alt outside-limits
2. afs-state pitch-engd alt-hold
3. afs-state roll-engd not-hdg-hold
4. afs-state pitch-engd alt-hold
5. afs-state roll-engd hdg-hold

Subtask level:

6. afs-state pitch-engd alt-hold
7. afs-state pitch-engd alt-hold
8. afs-state pitch-engd alt-hold
9. afs-state pitch-engd alt-hold
10. afs-state roll-engd hdg-hold
11. afs-state roll-engd hdg-hold

Action level:

12. afs-state mcp-alt outside-limits
13. afs-state pitch-engd alt-hold
14. afs-state roll-engd not-hdg-hold
15. afs-state pitch-engd alt-hold
16. afs-state roll-engd hdg-hold



Conditions Legend

Task level:

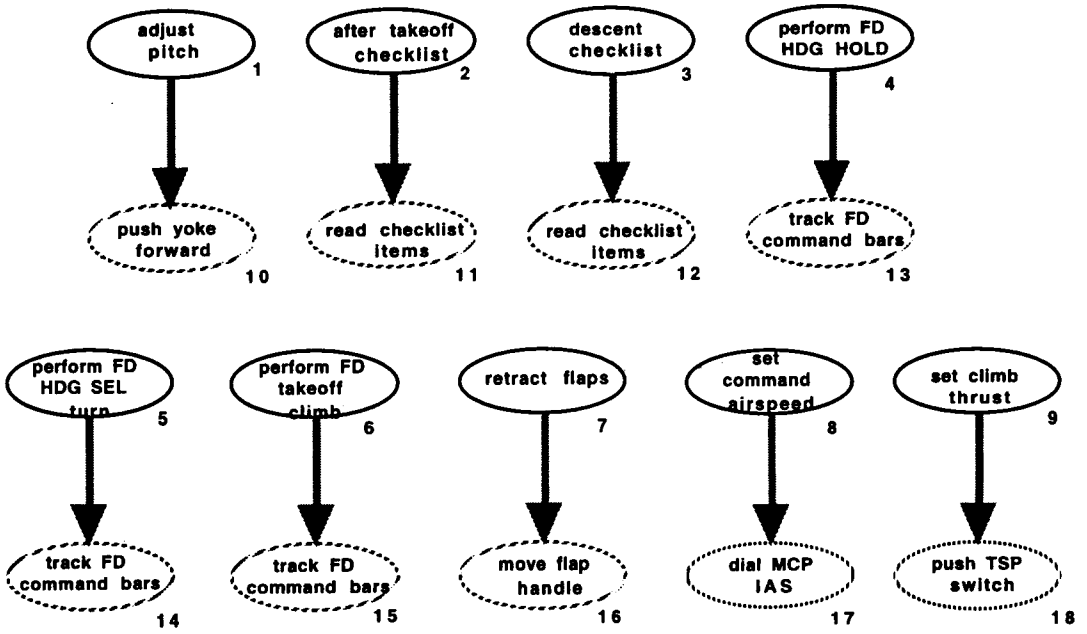
1. afs-state athr-engd not-fl-ch
2. afs-state athr-engd fl-ch
3. afs-state athr-engd fl-ch

Subtask level:

4. afs-state mcp-alt outside-limits
5. afs-state mcp-alt within-limits
6. afs-state athr-engd fl-ch
7. afs-state mcp-alt outside-limits
8. afs-state mcp-spd outside-limits
9. afs-state athr-engd fl-ch
10. afs-state athr-engd fl-ch
11. afs-state mcp-alt outside-limits
12. afs-state mcp-spd outside-limits
13. afs-state athr-engd fl-ch

Action level:

14. afs-state mcp-alt outside-limits
15. afs-state mcp-alt within-limits
16. afs-state mcp-alt outside-limits
17. afs-state mcp-spd outside-limits
18. afs-state athr-engd fl-ch
19. afs-state mcp-alt outside-limits
20. afs-state mcp-spd outside-limits
21. afs-state athr-engd fl-ch



Conditions Legend

Task level:

1. acrft-state abs-alt at-or-above-1000
2. acrft-state abs-alt at-or-above-1000
3. current-phase cruise in-progress
4. afs-state roll-engd hdg-hold
5. afs-state roll-engd hdg-sel
6. afs-state cmd-mode fd
7. acrft-state abs-alt at-or-above-1000
8. acrft-state abs-alt at-or-above-1000
   & afs-state mcp-spd outside-limits
   & afs-state pitch-engd not-alt-cap
9. acrft-state abs-alt at-or-above-1000
   & afs-state tsp not-clb

Action level:

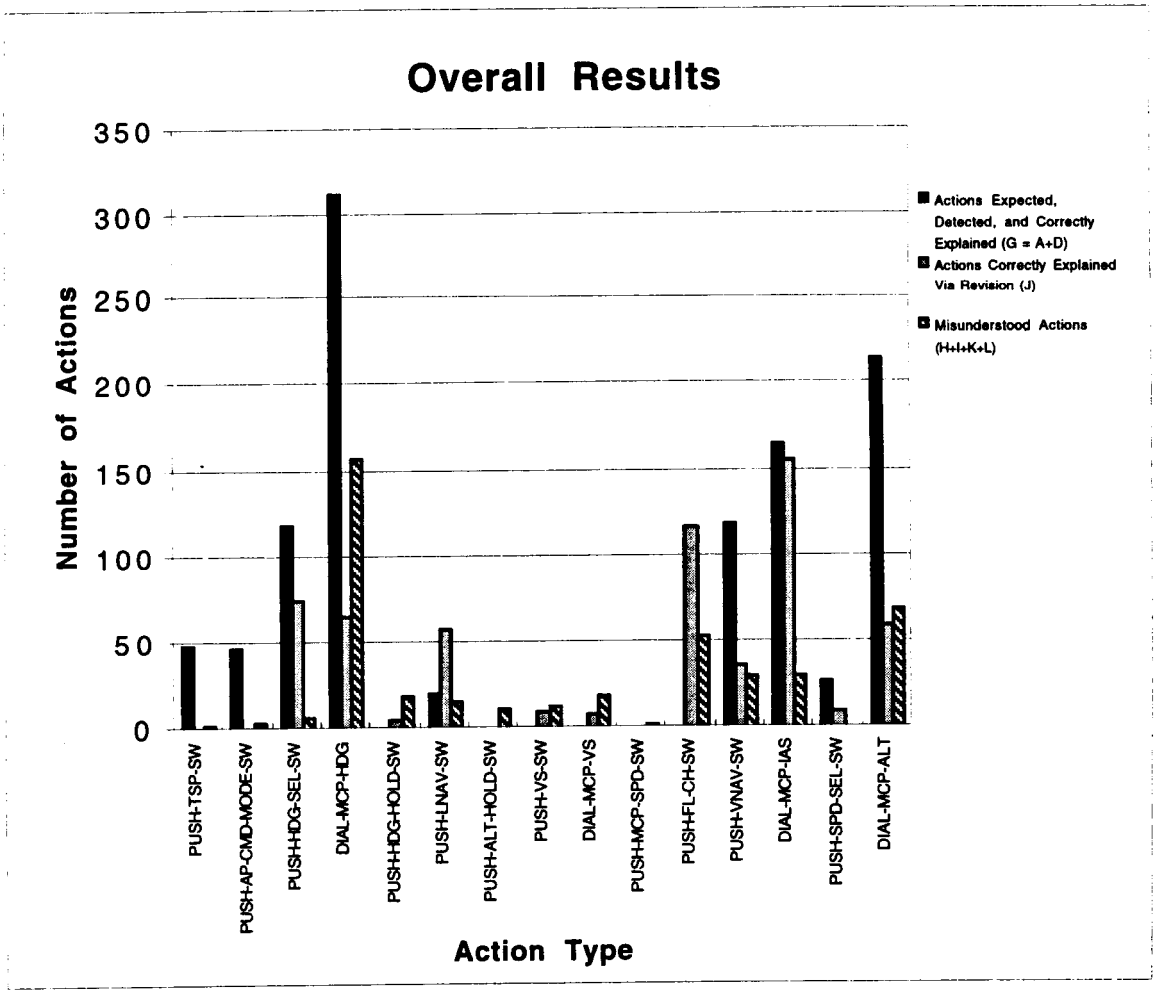
10. acrft-state abs-alt at-or-above-1000
11. acrft-state abs-alt at-or-above-1000
12. current-phase cruise in-progress
13. afs-state roll-engd hdg-hold
14. afs-state roll-engd hdg-sel
15. afs-state cmd-mode fd
16. acrft-state abs-alt at-or-above-1000
17. acrft-state abs-alt at-or-above-1000
    & afs-state mcp-spd outside-limits
18. acrft-state abs-alt at-or-above-1000
    & afs-state tsp not-clb

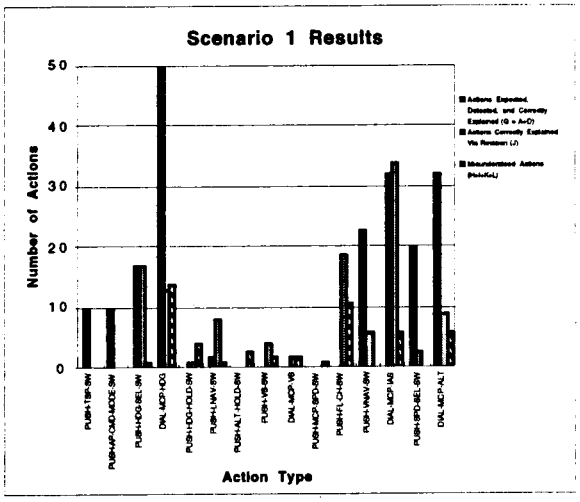
APPENDIX B: EMPIRICAL EVALUATION RESULTS

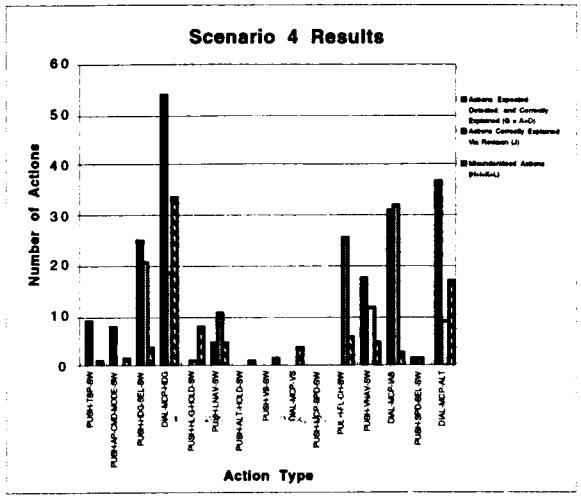
OVERALL RESULTS

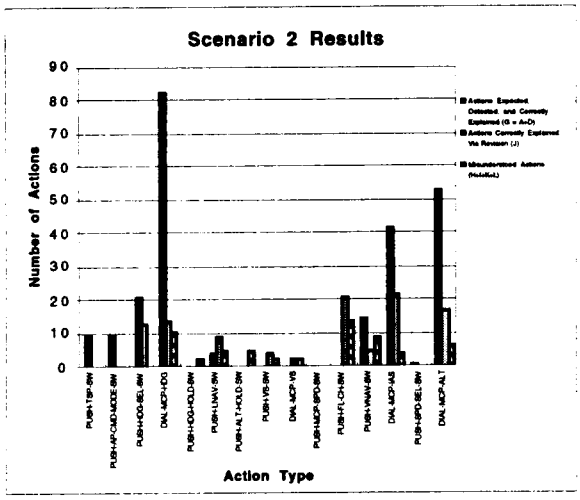
	Total	PUSH-TSP-SW	PUSH-AP-CMD-MODE-SW	PUSH-HDG-SEL-SW	DIAL-MCP-HDG	PUSH-HDG-HOLD-SW	PUSH-LNAV-SW	PUSH-ALT-HOLD-SW	PUSH-VS-SW	DIAL-MCP-VS	PUSH-MCP-SPD-SW	PUSH-FL-CH-SW	PUSH-VNAV-SW	DIAL-MCP-IAS	PUSH-SPD-SEL-SW	DIAL-MCP-ALT
Detected Actions (G+H+I+J+K+L)	**2089**	**50**	**50**	**199**	**535**	**23**	**92**	**11**	**21**	**27**	**2**	**170**	**183**	**349**	**36**	**341**
Expected and Detected Actions (G+H)	**1107**	**48**	**47**	**118**	**314**	**0**	**20**	**0**	**0**	**0**	**0**	**0**	**118**	**165**	**26**	**251**
Actions Expected, Detected, and Correctly Explained (G = A+D)	1069	48	47	118	313	0	20	0	0	0	0	0	118	165	26	214
Actions Expected, Detected, and Incorrectly Explained (H = B+E)	38	0	0	0	1	0	0	0	0	0	0	0	0	0	0	37
Expected Actions Flagged Late (A+B)	53	0	0	7	15	0	1	0	0	0	0	0	4	16	1	9
Unexpected and Detected Actions (I+J+K+L)	**982**	**2**	**3**	**81**	**221**	**23**	**72**	**11**	**21**	**27**	**2**	**170**	**65**	**184**	**10**	**90**
Actions Correctly Explained Via Revision (J)	595	0	0	75	65	4	57	0	9	8	0	117	36	155	10	59
Actions Incorrectly Explained Via Revision (K)	51	0	0	1	0	0	0	0	0	0	0	0	5	19	0	26
Actions Unable to be Explained Via Revision (L)	329	1	0	5	155	18	15	11	12	19	2	53	24	9	0	5
Actions Unknown to Current OFM-ACM Subphase (I)	7	1	3	0	1	1	0	0	0	0	0	0	0	1	0	0
Unfulfilled Expectations (C+F)	317	1	0	8	37	2	7	3	0	0	23	0	52	132	38	14
Unfulfilled Expectations Flagged Late (C)	73	1	0	3	5	0	0	1	0	0	1	0	18	21	19	4
Misunderstood Actions (H+I+K+L)	**425**	**2**	**3**	**6**	**157**	**19**	**15**	**11**	**12**	**19**	**2**	**53**	**29**	**29**	**0**	**68**

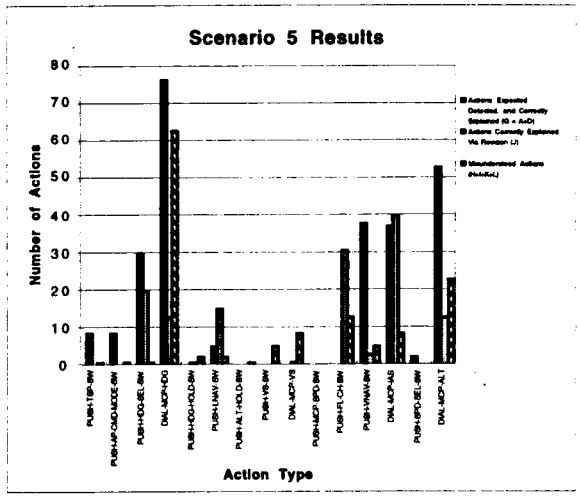
APPENDIX C: GRAPHS OF DETECTED ACTIONS DATA

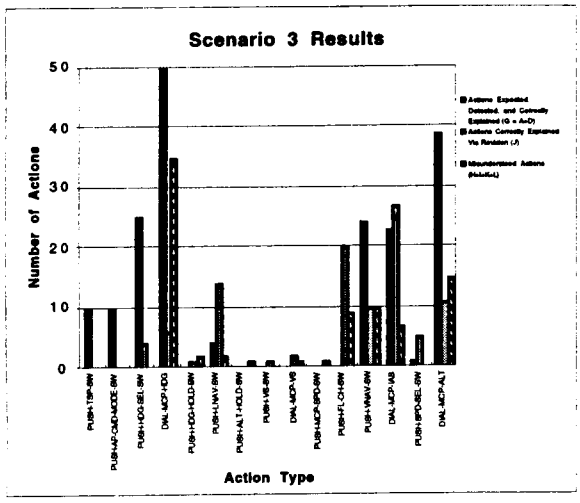


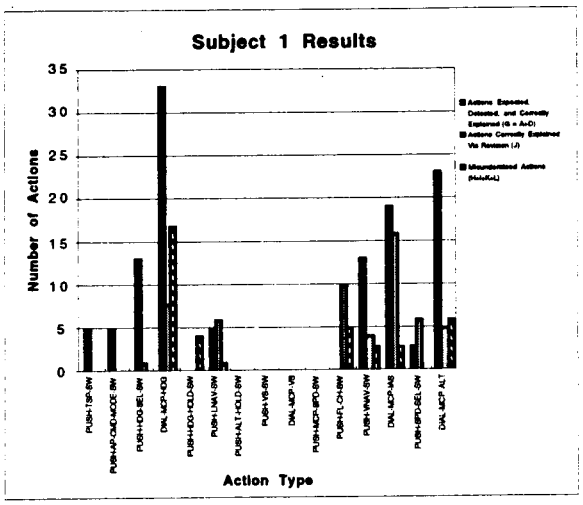

Scenario 1 Results
Number of Actions
Action Type

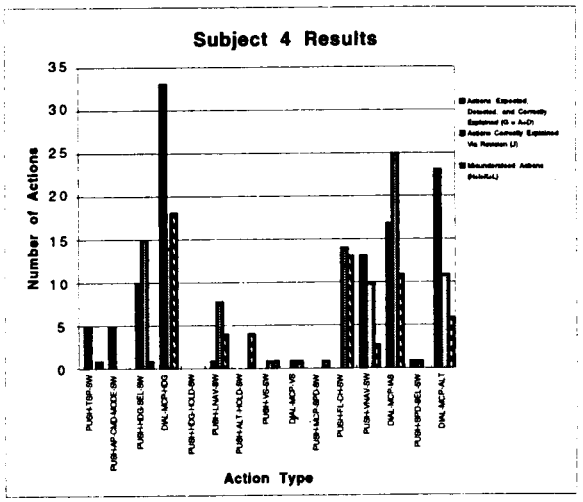

Scenario 4 Results
Number of Actions
Action Type

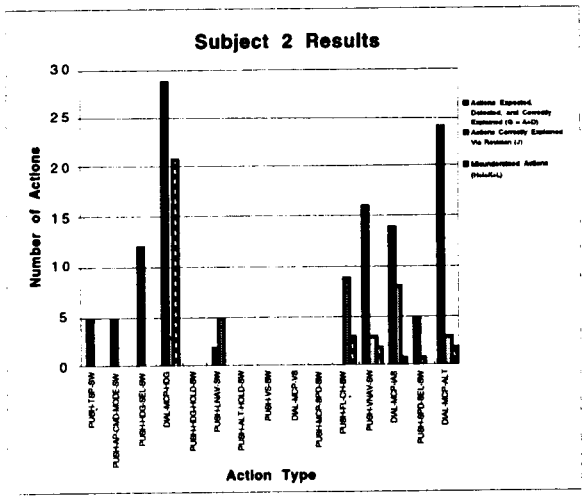

Scenario 2 Results
Number of Actions
Action Type

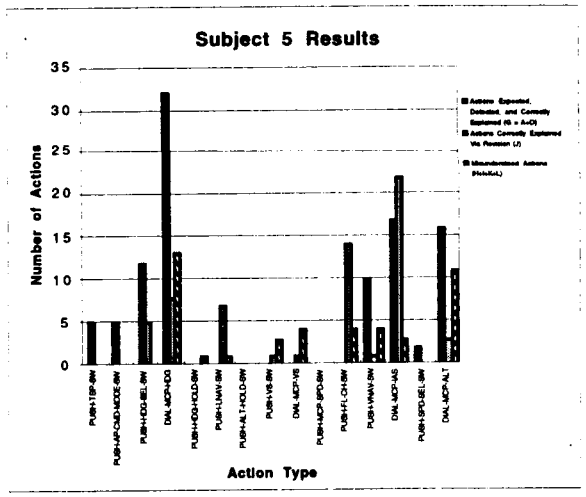

Scenario 5 Results
Number of Actions
Action Type

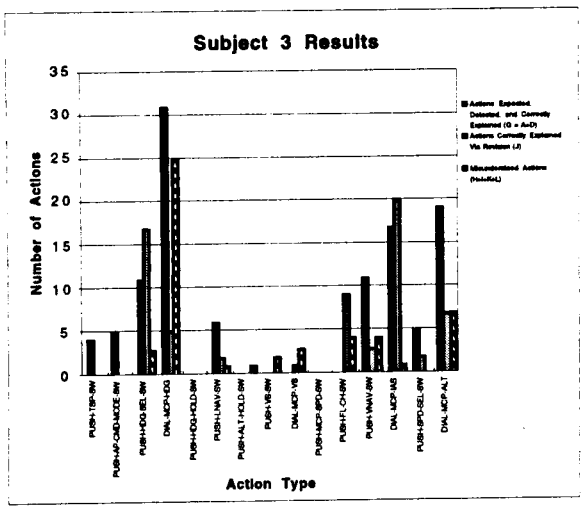

Scenario 3 Results
Number of Actions
Action Type

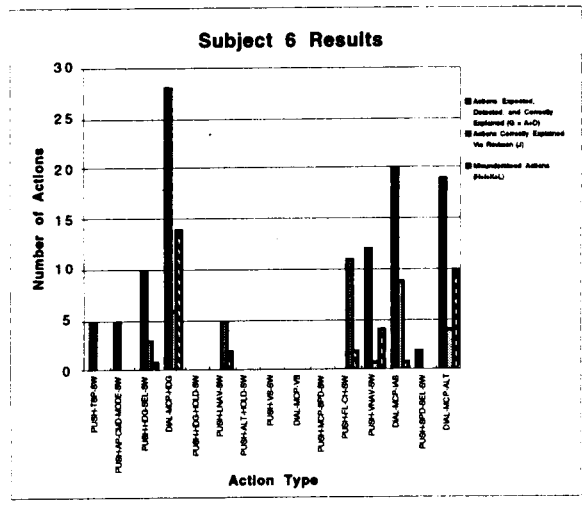

Subject 1 Results
Number of Actions
Action Type

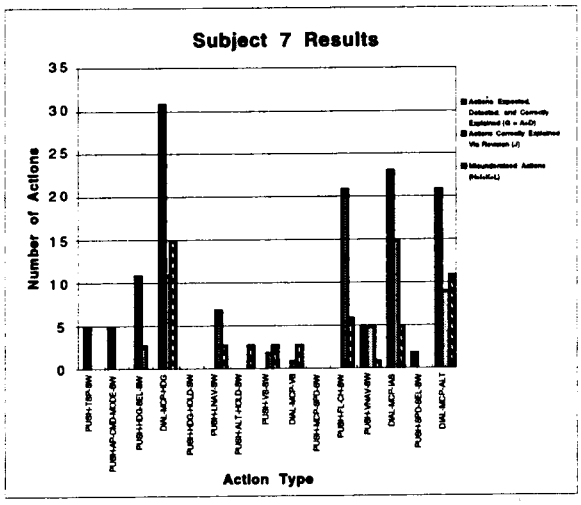

Subject 4 Results
Number of Actions
Action Type

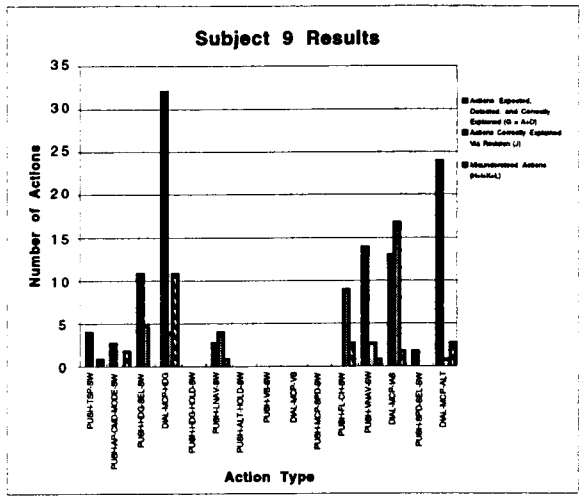

Subject 2 Results
Number of Actions
Action Type

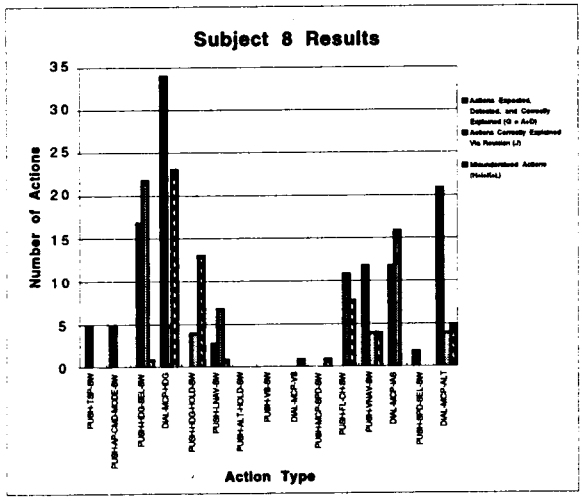

Subject 5 Results
Number of Actions
Action Type

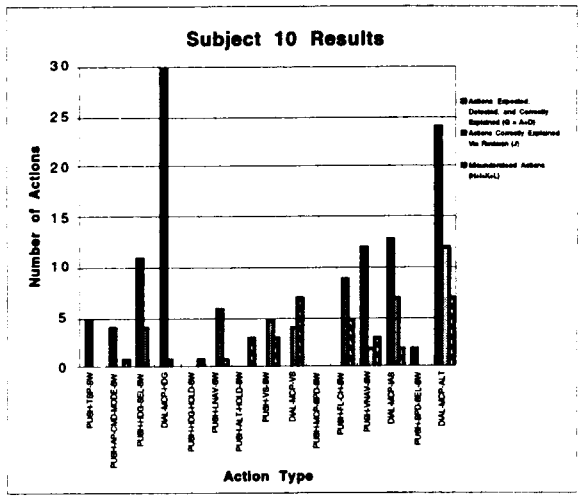

Subject 3 Results
Number of Actions
Action Type

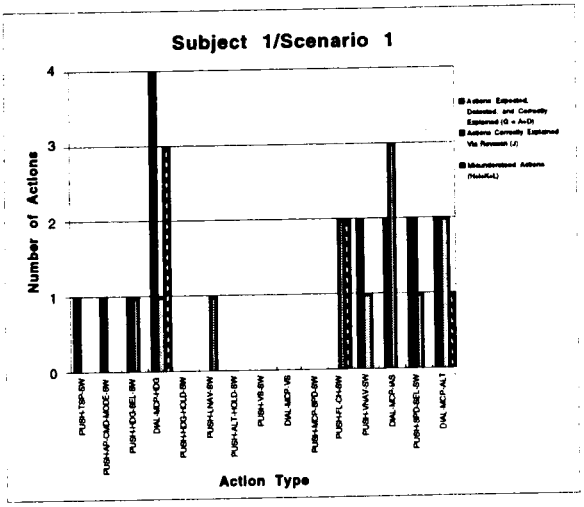

Subject 6 Results
Number of Actions
Action Type

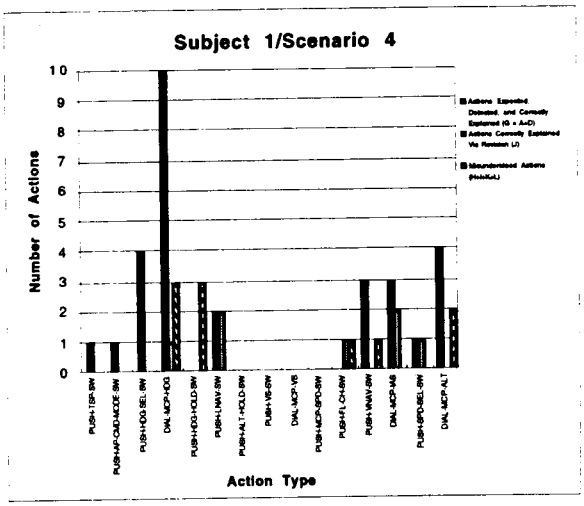

Subject 7 Results
Number of Actions
Action Type

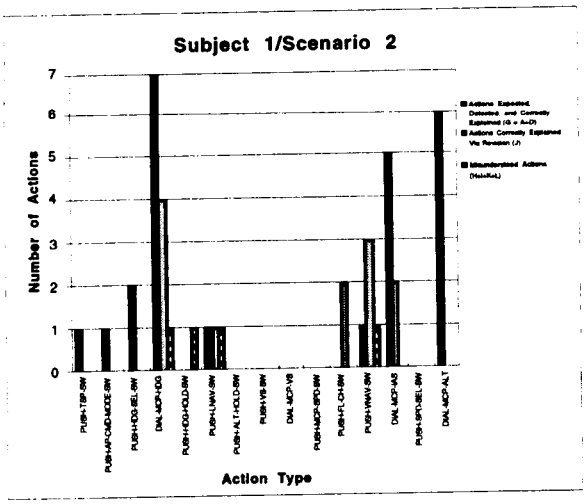

Subject 9 Results
Number of Actions
Action Type

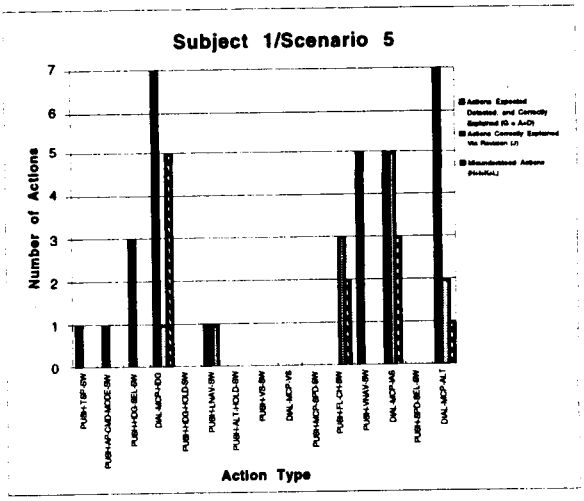

Subject 8 Results
Number of Actions
Action Type

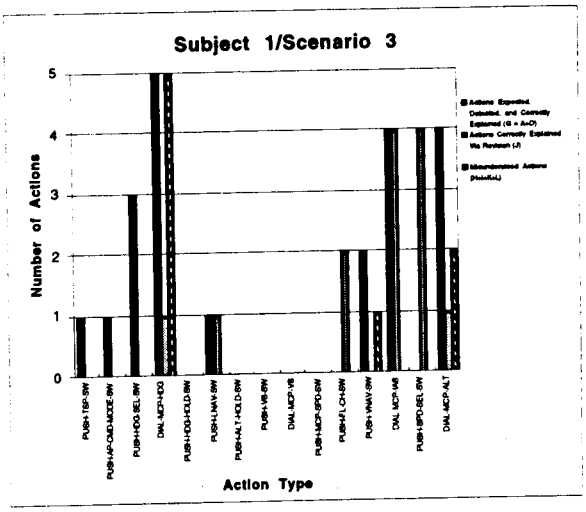

Subject 10 Results
Number of Actions
Action Type

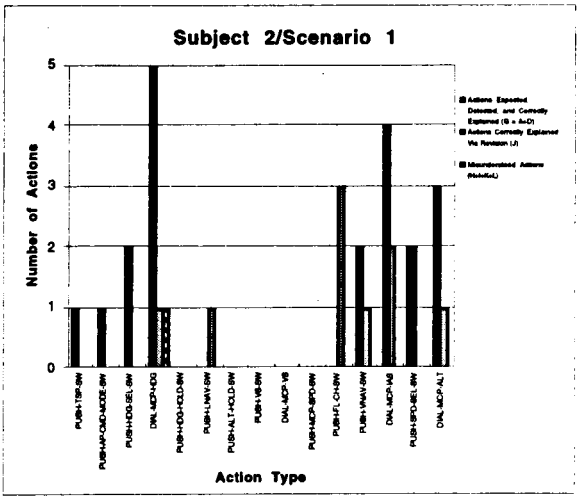

Subject 1/Scenario 1
Number of Actions
Action Type

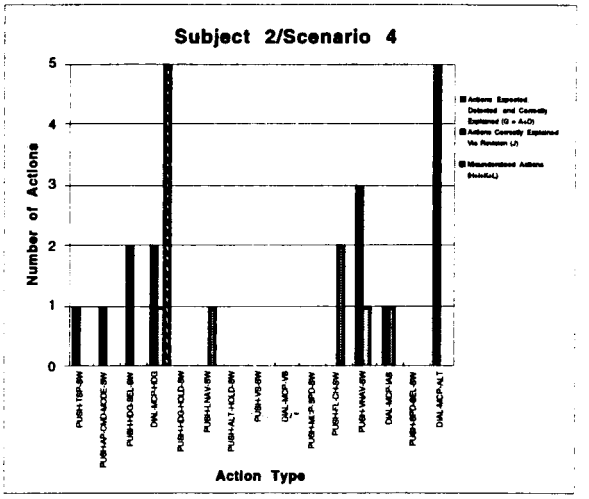

Subject 1/Scenario 4
Number of Actions
Action Type

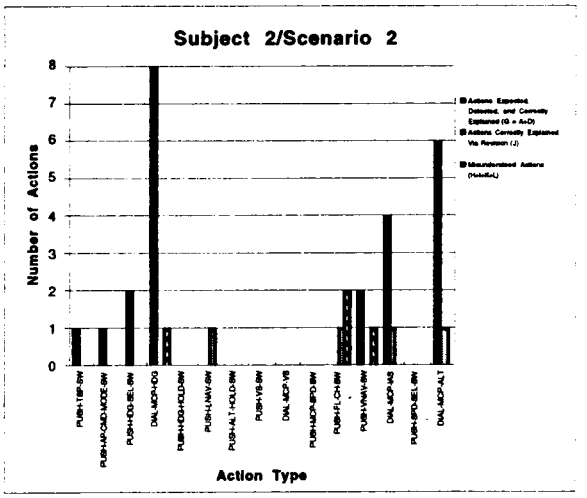

Subject 1/Scenario 2
Number of Actions
Action Type

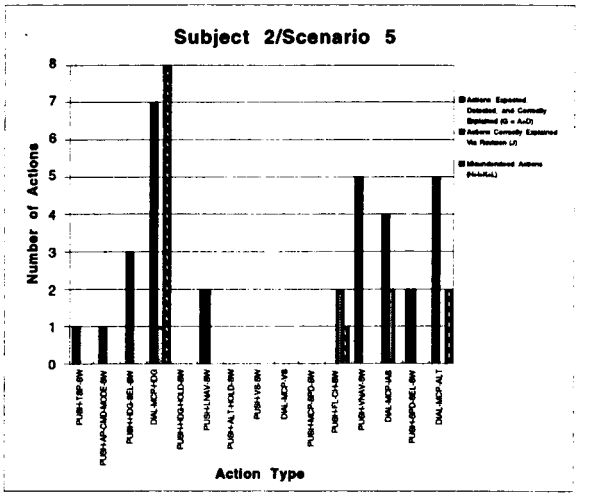

Subject 1/Scenario 5
Number of Actions
Action Type

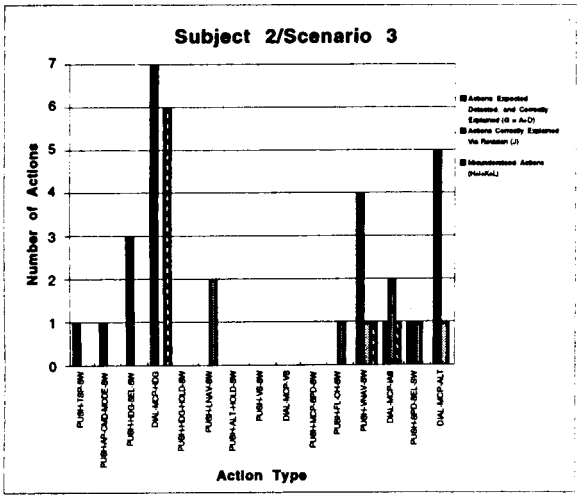

Subject 1/Scenario 3
Number of Actions
Action Type

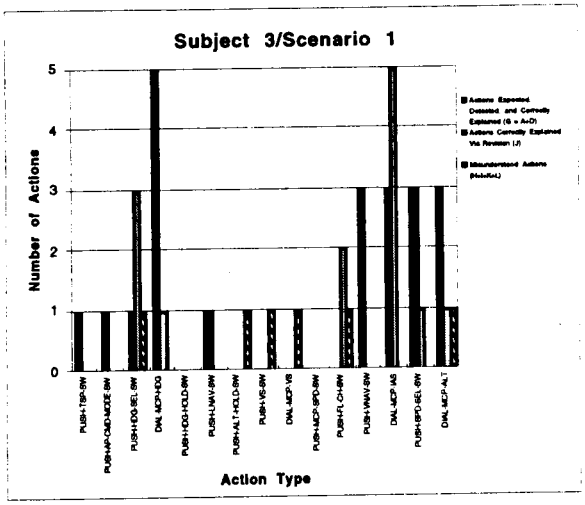

Subject 2/Scenario 1
Number of Actions
Action Type

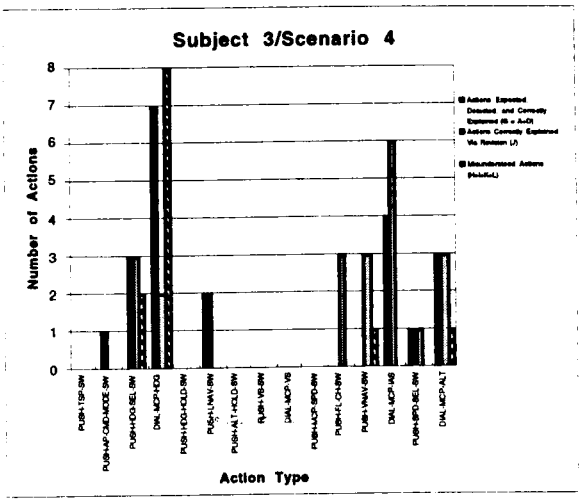

Subject 2/Scenario 4
Number of Actions
Action Type

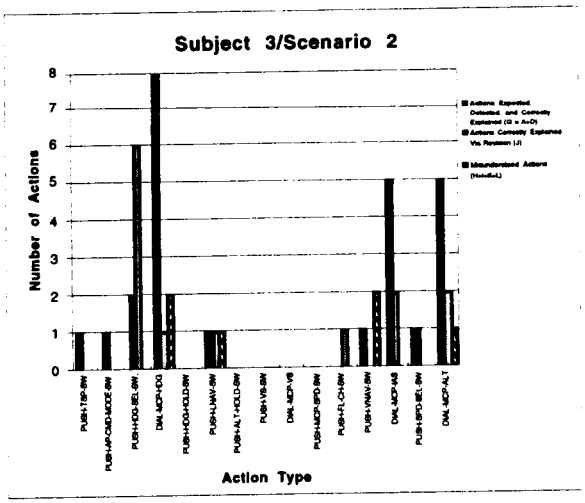

Subject 2/Scenario 2
Number of Actions
Action Type

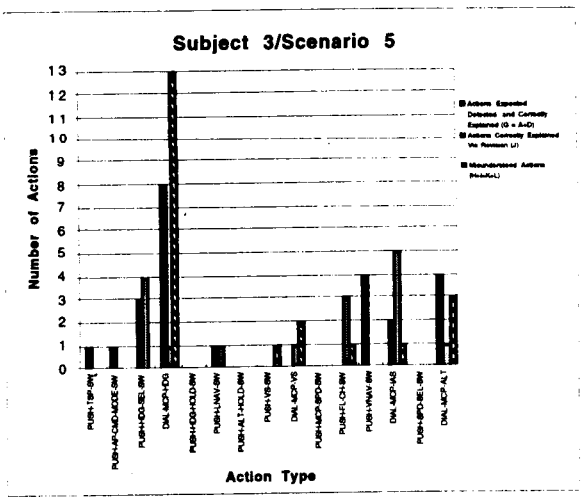

Subject 2/Scenario 5
Number of Actions
Action Type

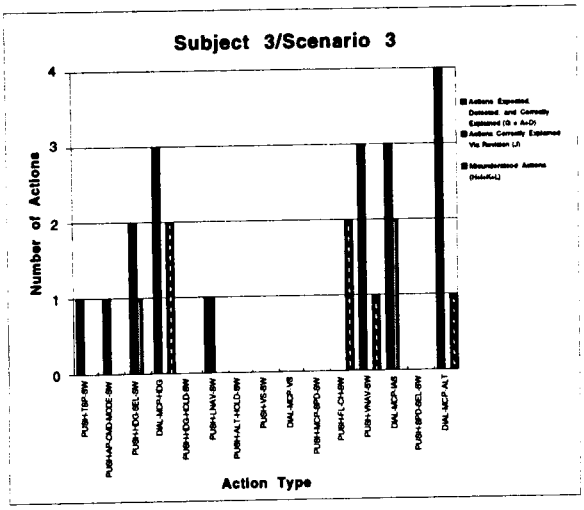

Subject 2/Scenario 3
Number of Actions
Action Type

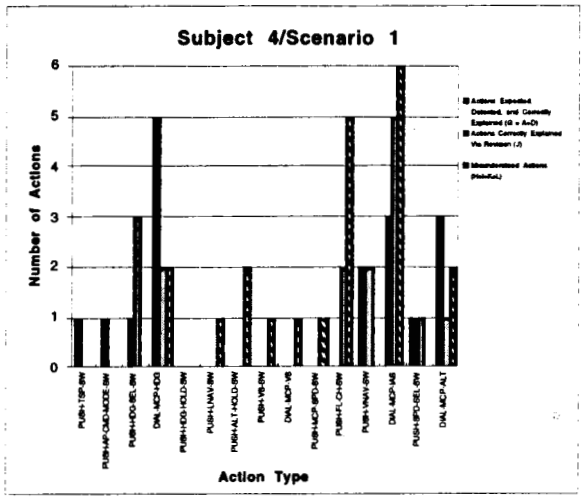

Subject 3/Scenario 1
Number of Actions
Action Type

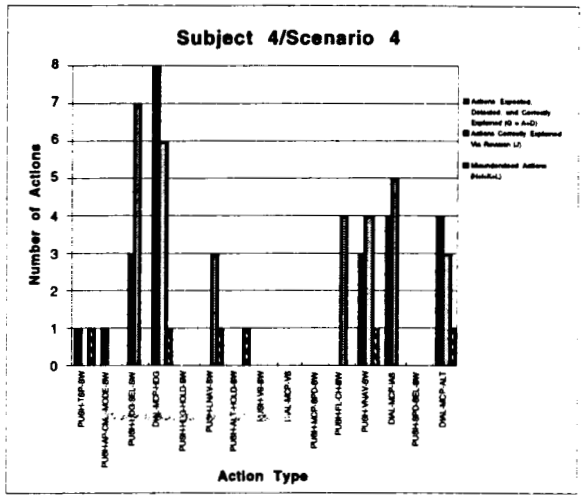

Subject 3/Scenario 4
Number of Actions
Action Type

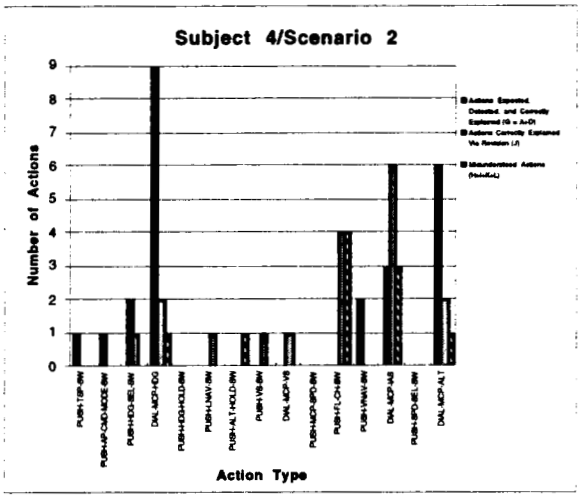

Subject 3/Scenario 2
Number of Actions
Action Type

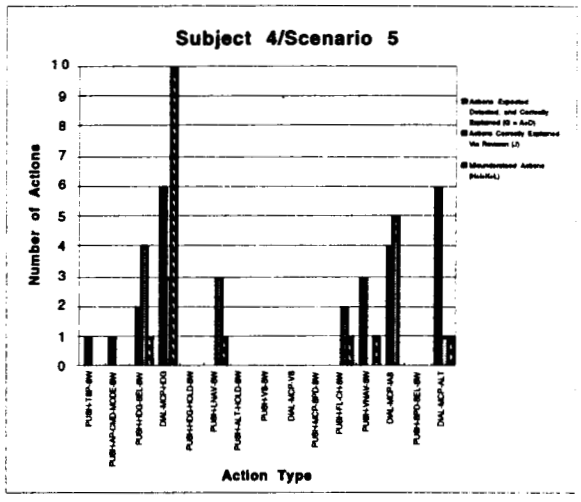

Subject 3/Scenario 5
Number of Actions
Action Type

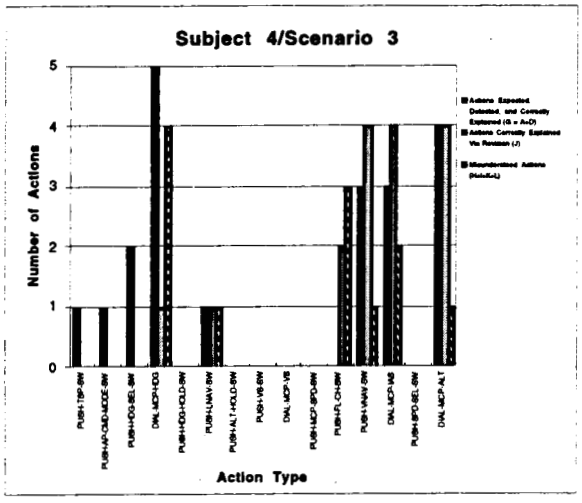

Subject 3/Scenario 3
Number of Actions
Action Type

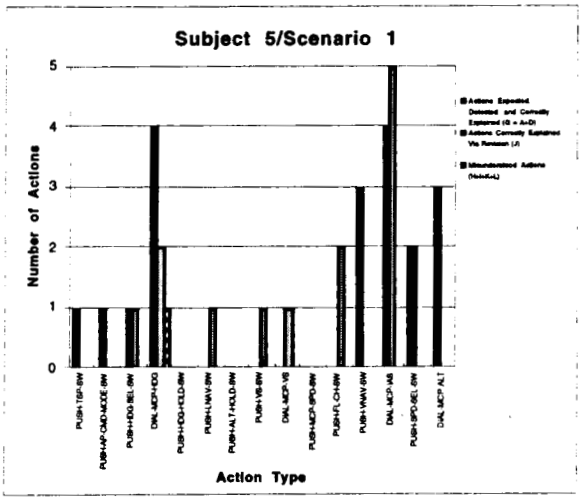

Subject 4/Scenario 1
Number of Actions
Action Type

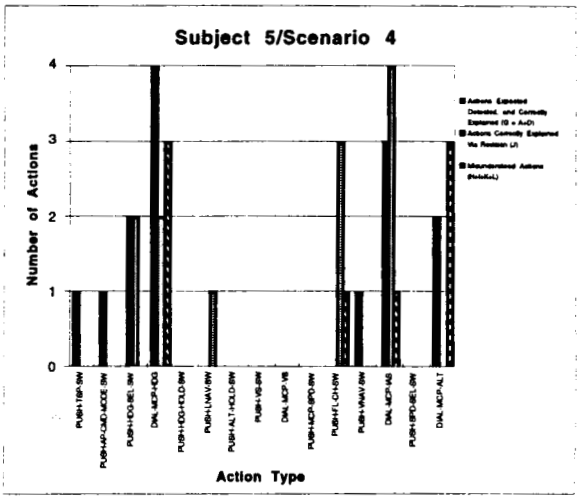

Subject 4/Scenario 4
Number of Actions
Action Type

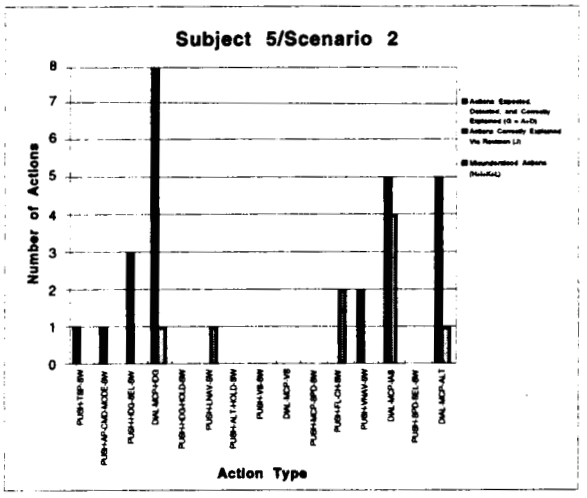

Subject 4/Scenario 2
Number of Actions
Action Type

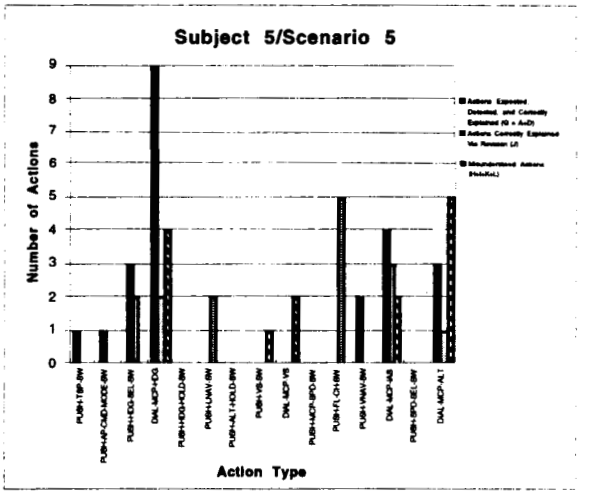

Subject 4/Scenario 5
Number of Actions
Action Type

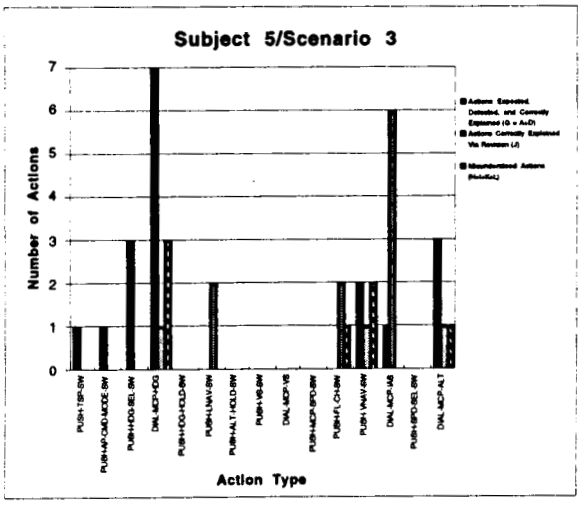

Subject 4/Scenario 3
Number of Actions
Action Type

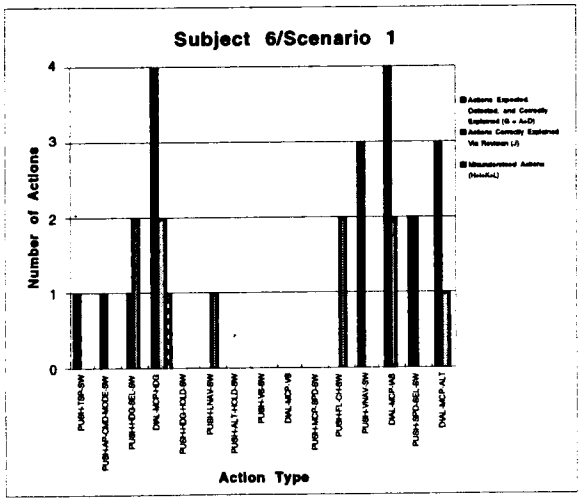

Subject 5/Scenario 1
Number of Actions
Action Type

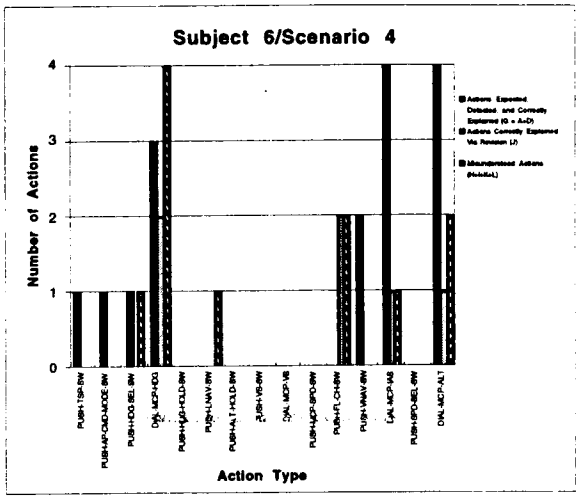

Subject 5/Scenario 4
Number of Actions
Action Type

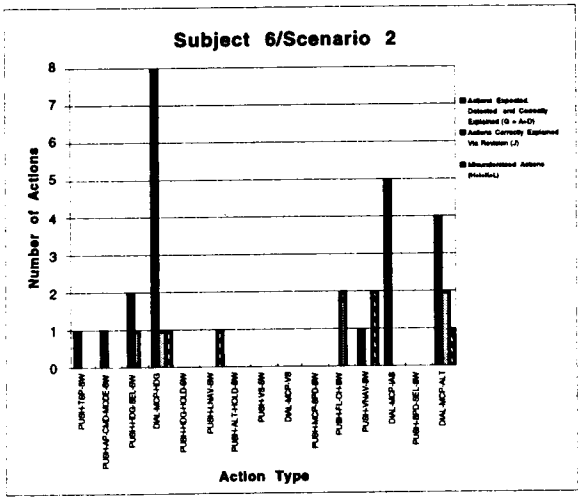

Subject 5/Scenario 2
Number of Actions
Action Type

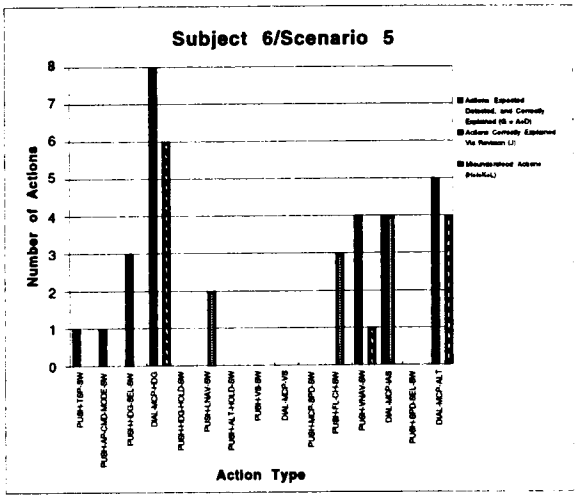

Subject 5/Scenario 5
Number of Actions
Action Type

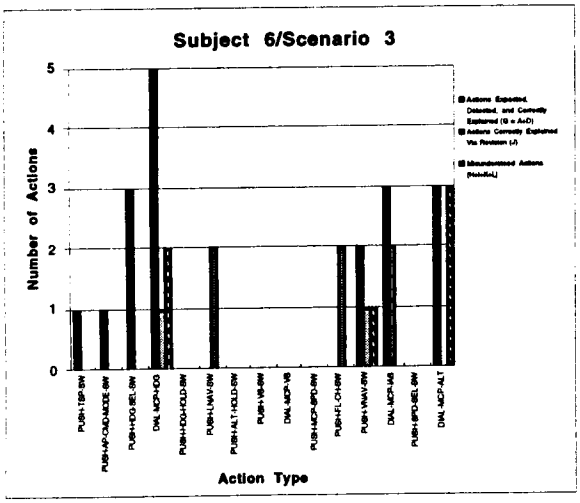

Subject 5/Scenario 3
Number of Actions
Action Type

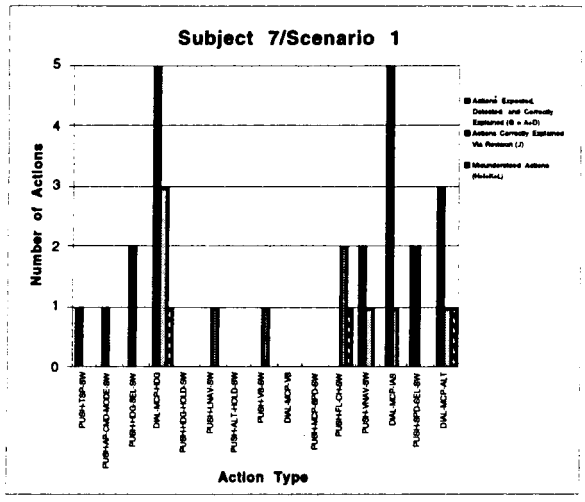

Subject 6/Scenario 1
Number of Actions
Action Type

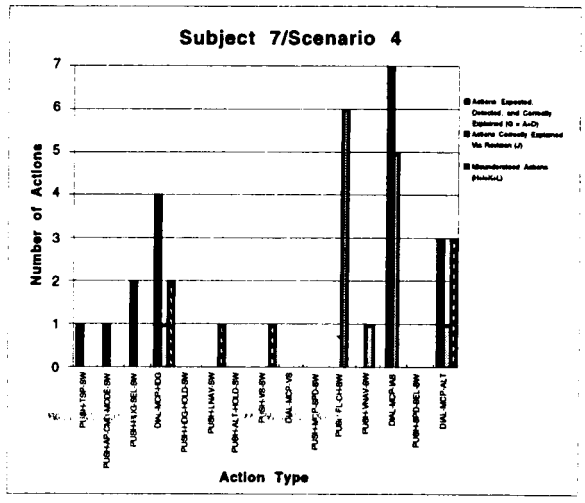

Subject 6/Scenario 4
Number of Actions
Action Type

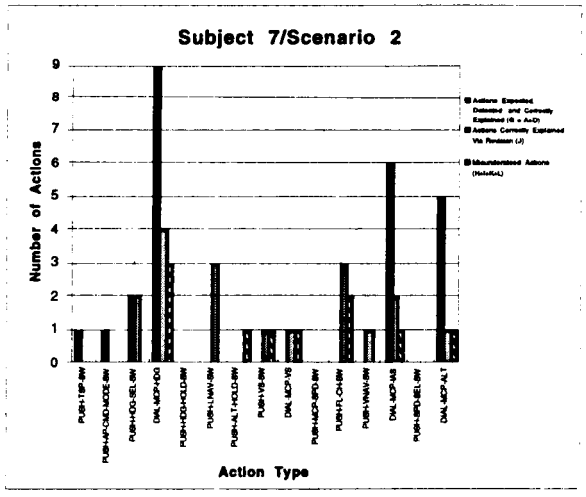

Subject 6/Scenario 2
Number of Actions
Action Type

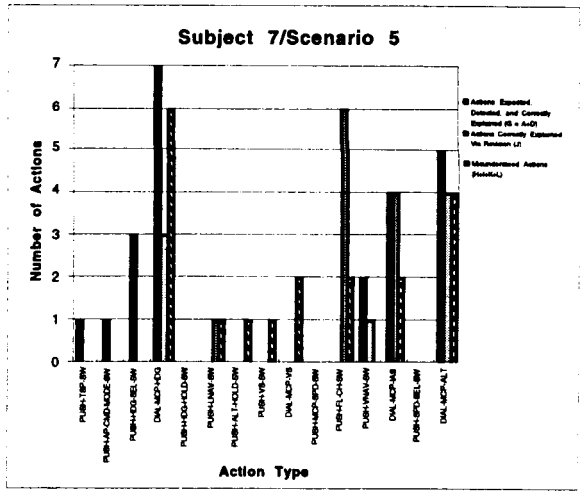

Subject 6/Scenario 5
Number of Actions
Action Type

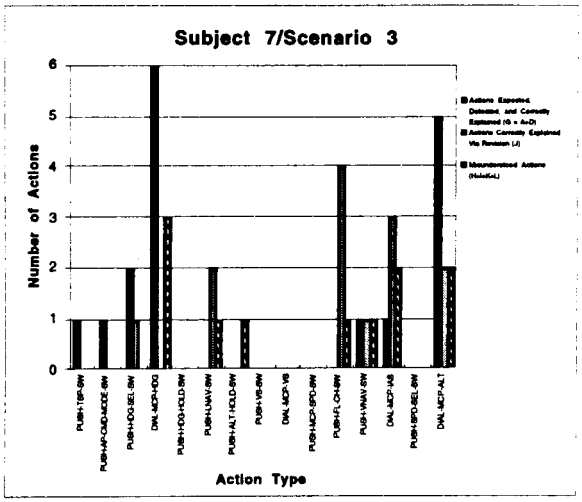

Subject 6/Scenario 3
Number of Actions
Action Type

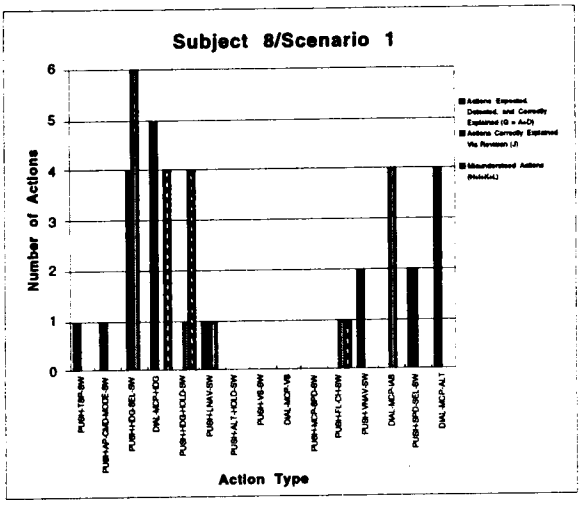

Subject 7/Scenario 1
Number of Actions
Action Type

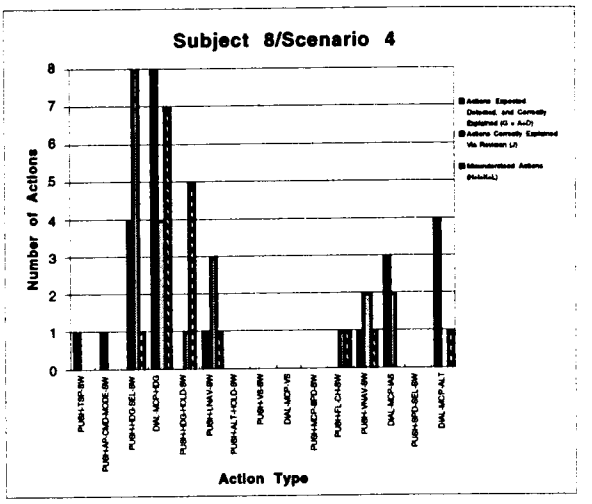

Subject 7/Scenario 4
Number of Actions
Action Type

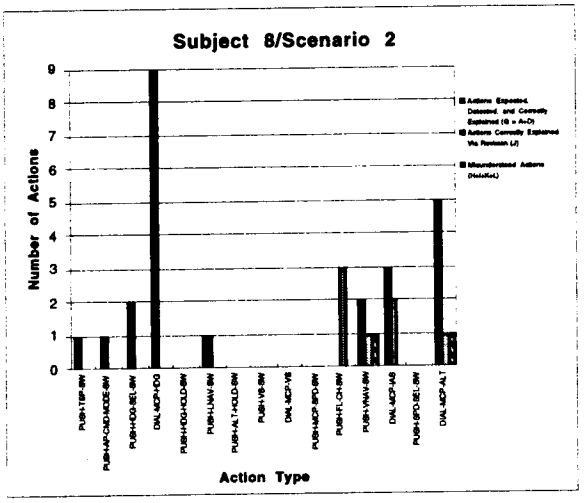

Subject 7/Scenario 2
Number of Actions
Action Type

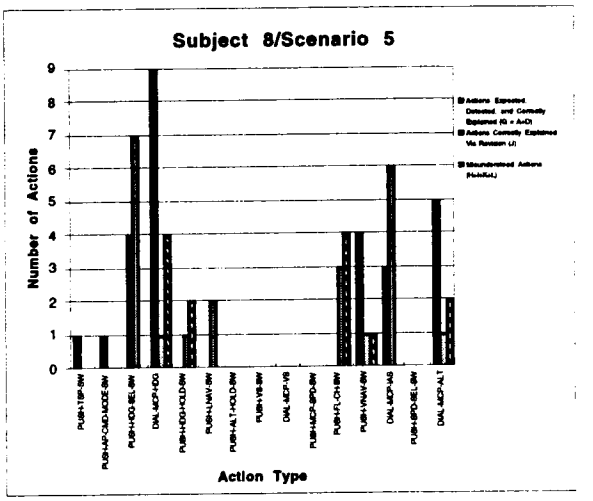

Subject 7/Scenario 5
Number of Actions
Action Type

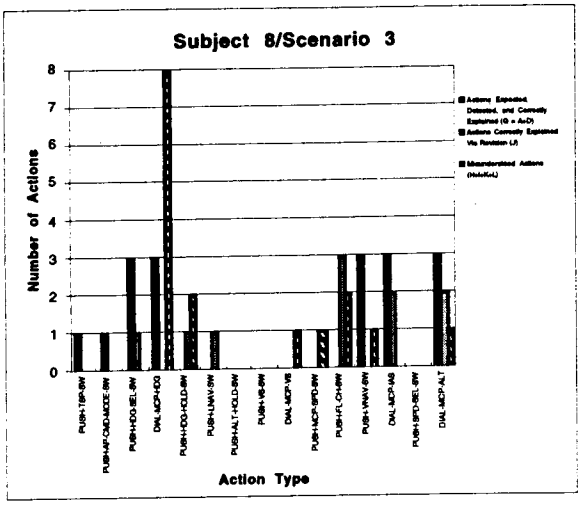

Subject 7/Scenario 3
Number of Actions
Action Type

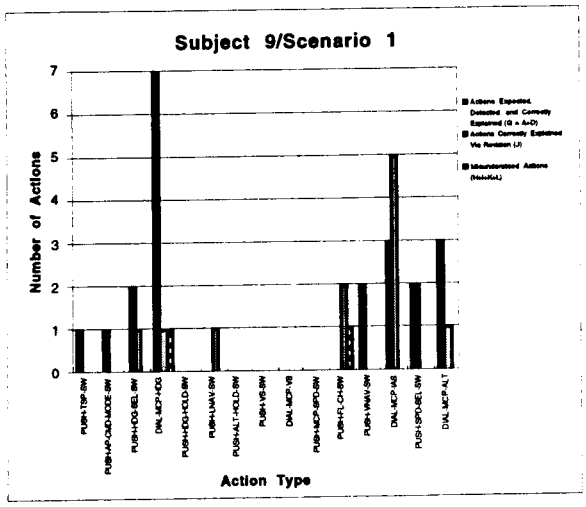

Subject 8/Scenario 1
Number of Actions
Action Type

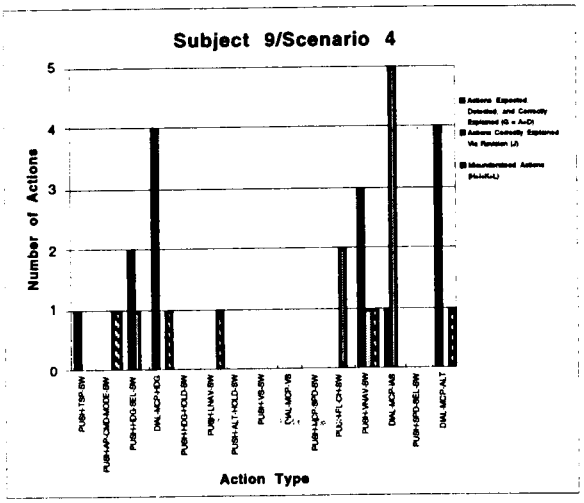

Subject 8/Scenario 4
Number of Actions
Action Type

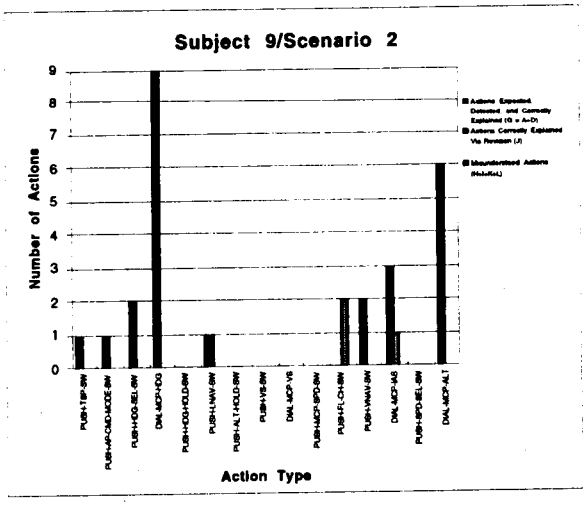

Subject 8/Scenario 2
Number of Actions
Action Type

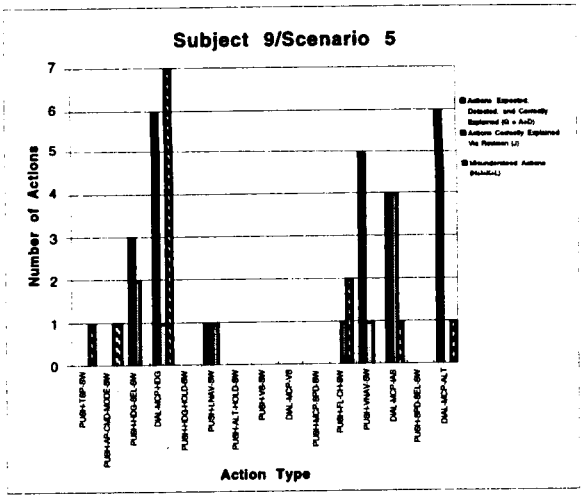

Subject 8/Scenario 5
Number of Actions
Action Type

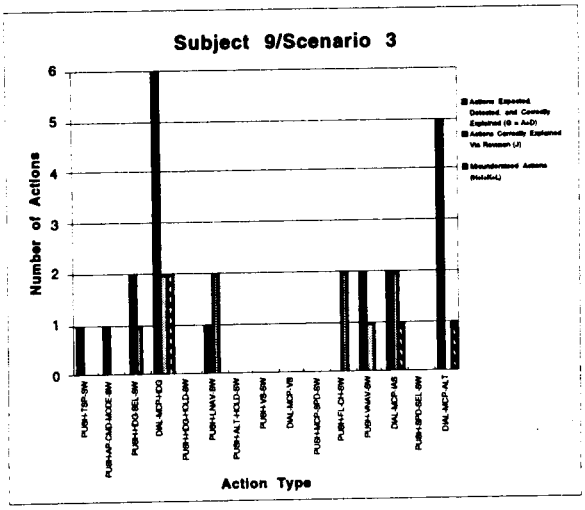

Subject 8/Scenario 3
Number of Actions
Action Type

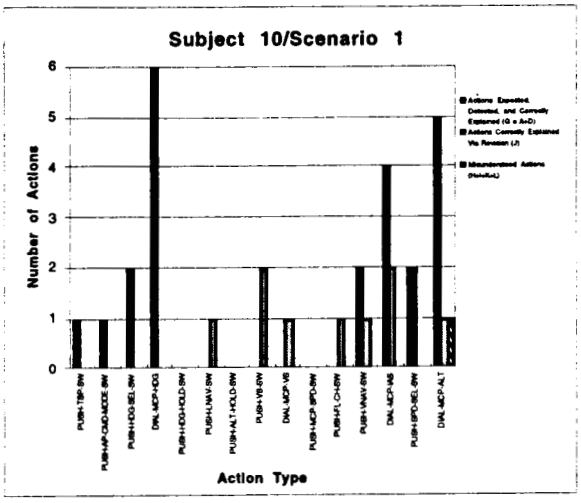

Subject 9/Scenario 1
Number of Actions
Action Type

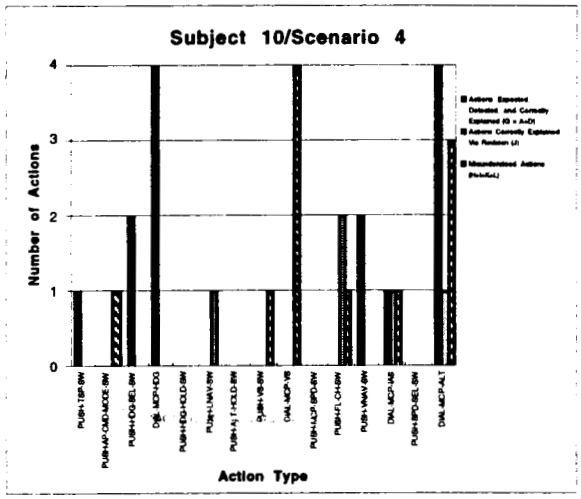

Subject 9/Scenario 4
Number of Actions
Action Type

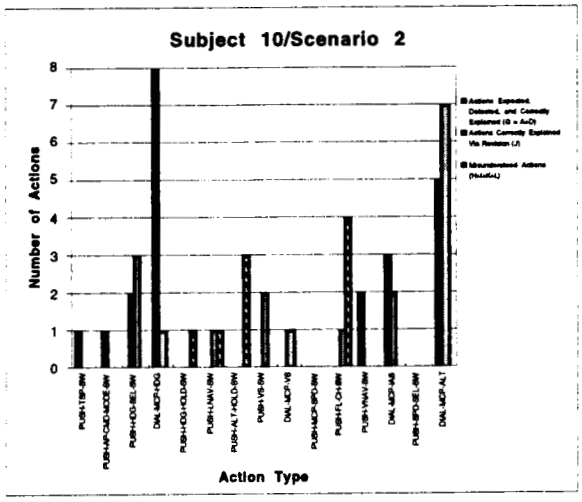

Subject 9/Scenario 2
Number of Actions
Action Type

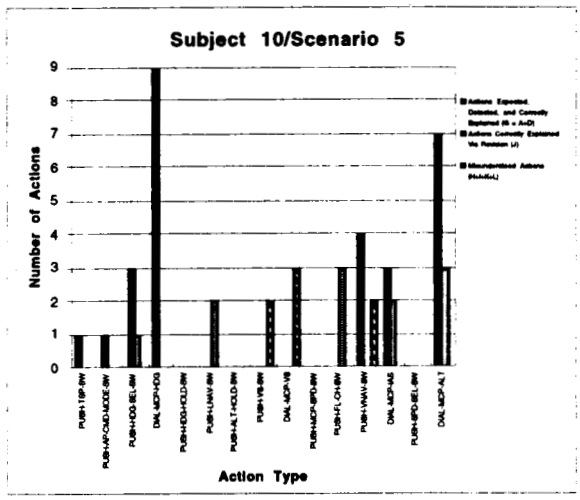

Subject 9/Scenario 5
Number of Actions
Action Type

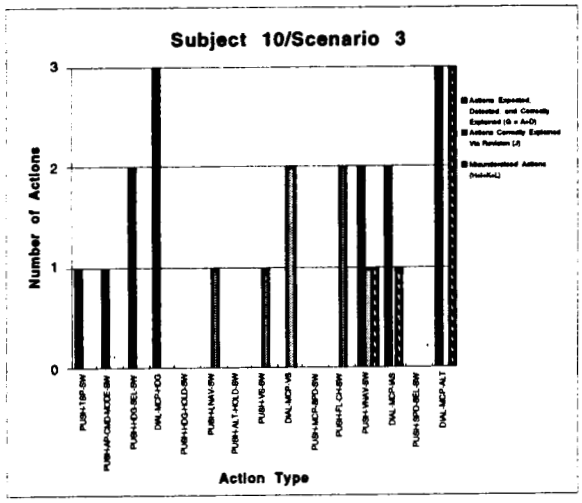

Subject 9/Scenario 3
Number of Actions
Action Type

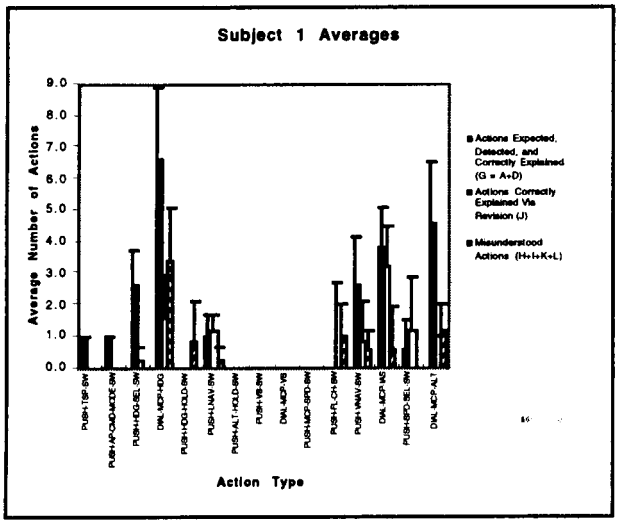

Subject 10/Scenario 1
Number of Actions
Action Type

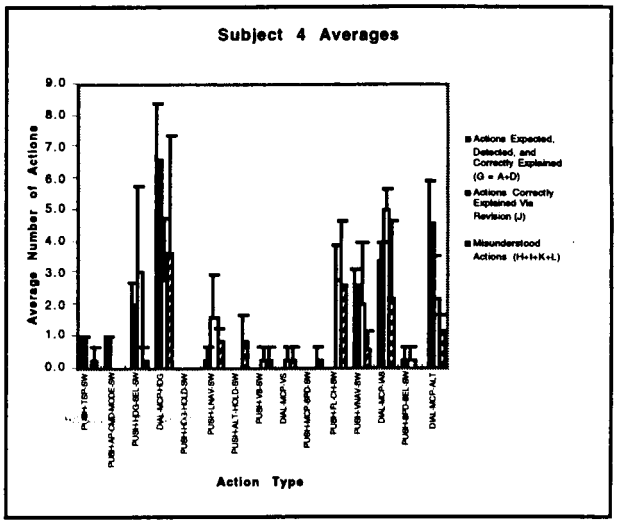

Subject 10/Scenario 4
Number of Actions
Action Type

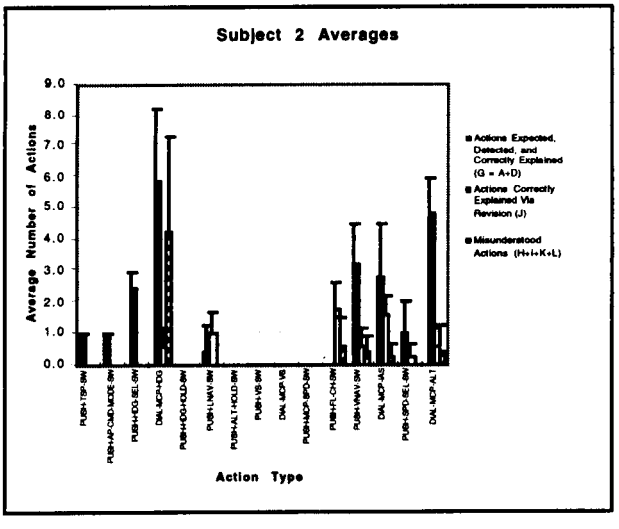

Subject 10/Scenario 2
Number of Actions
Action Type

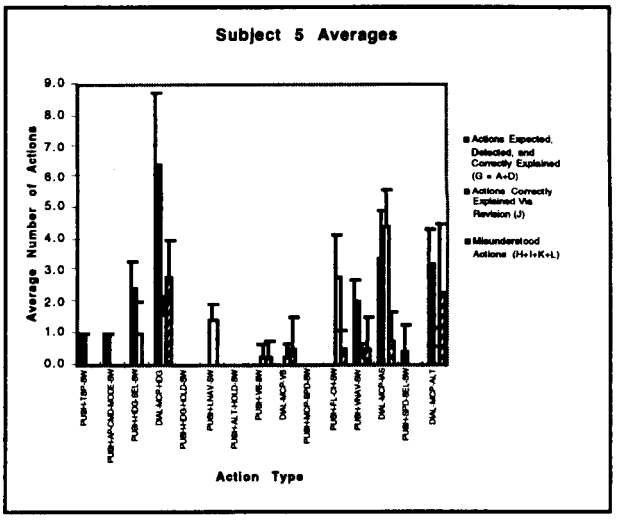

Subject 10/Scenario 5
Number of Actions
Action Type

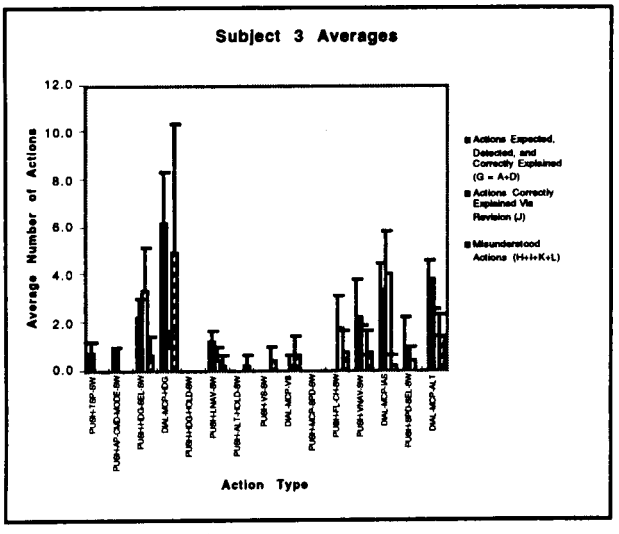

Subject 10/Scenario 3
Number of Actions
Action Type

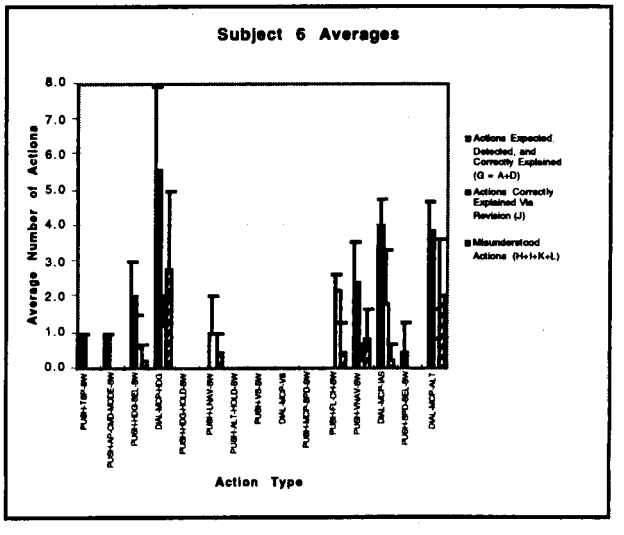

Subject 1 Averages
Average Number of Actions
Action Type
Actions Expected, Detected, and Correctly Explained (G = A+D)
Actions Correctly Explained Via Revision (J)
Misunderstood Actions (H+I+K+L)

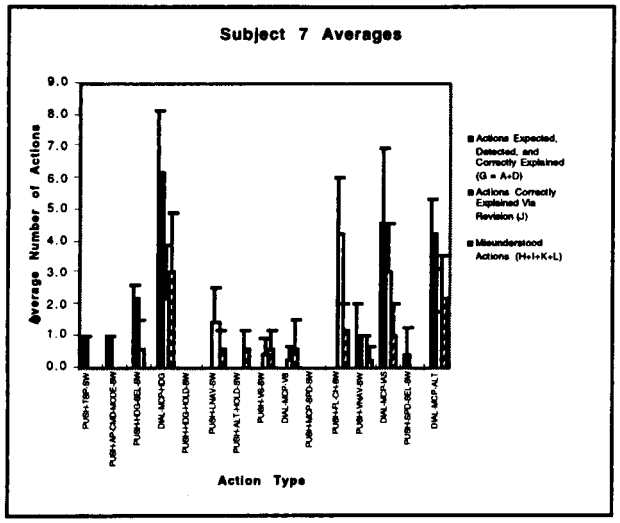

Subject 4 Averages
Average Number of Actions
Action Type
Actions Expected, Detected, and Correctly Explained (G = A+D)
Actions Correctly Explained Via Revision (J)
Misunderstood Actions (H+I+K+L)

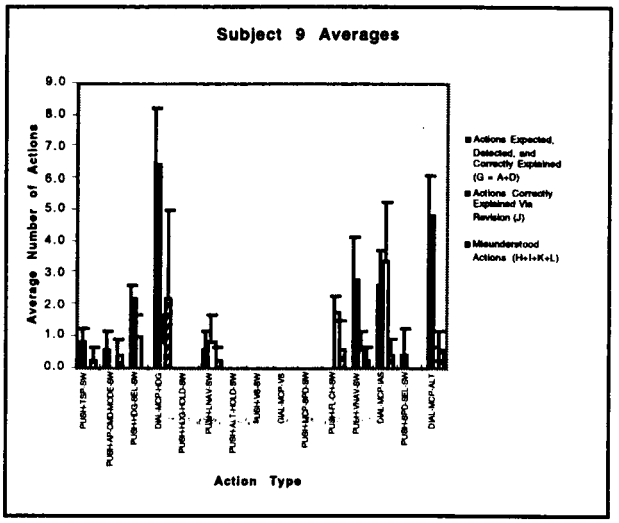

Subject 2 Averages
Average Number of Actions
Action Type
Actions Expected, Detected, and Correctly Explained (G = A+D)
Actions Correctly Explained Via Revision (J)
Misunderstood Actions (H+I+K+L)

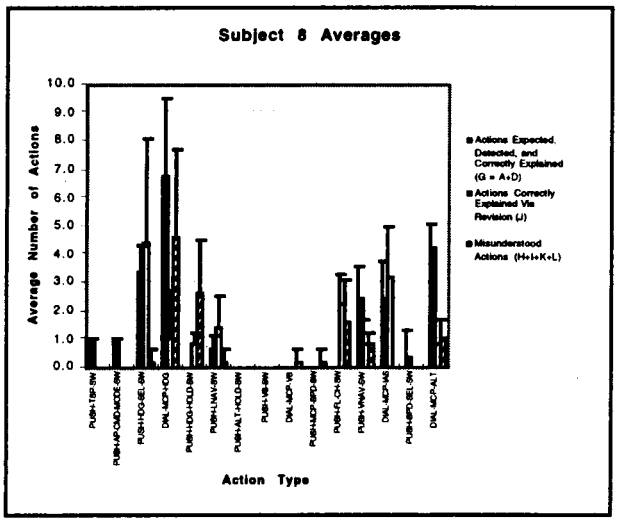

Subject 5 Averages
Average Number of Actions
Action Type
Actions Expected, Detected, and Correctly Explained (G = A+D)
Actions Correctly Explained Via Revision (J)
Misunderstood Actions (H+I+K+L)

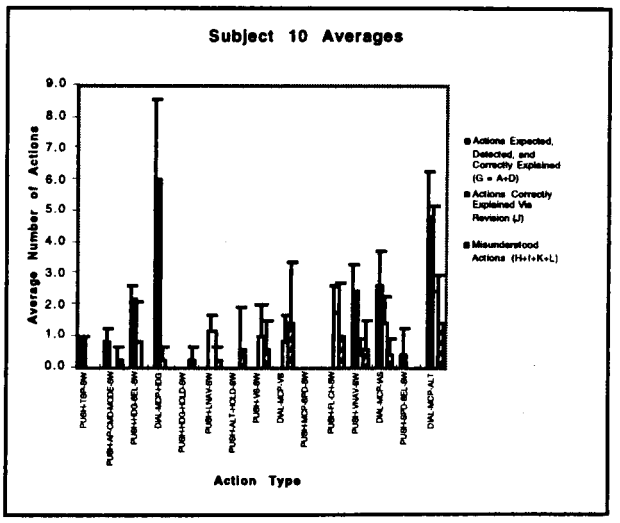

Subject 3 Averages
Average Number of Actions
Action Type
Actions Expected, Detected, and Correctly Explained (G = A+D)
Actions Correctly Explained Via Revision (J)
Misunderstood Actions (H+I+K+L)

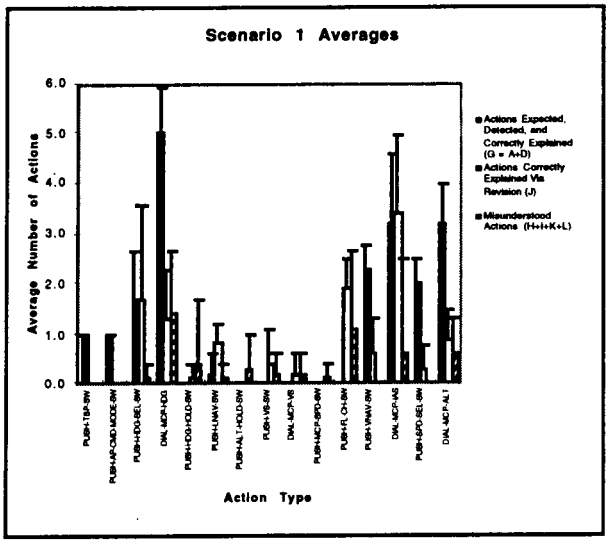

Subject 6 Averages
Average Number of Actions
Action Type
Actions Expected, Detected, and Correctly Explained (G = A+D)
Actions Correctly Explained Via Revision (J)
Misunderstood Actions (H+I+K+L)

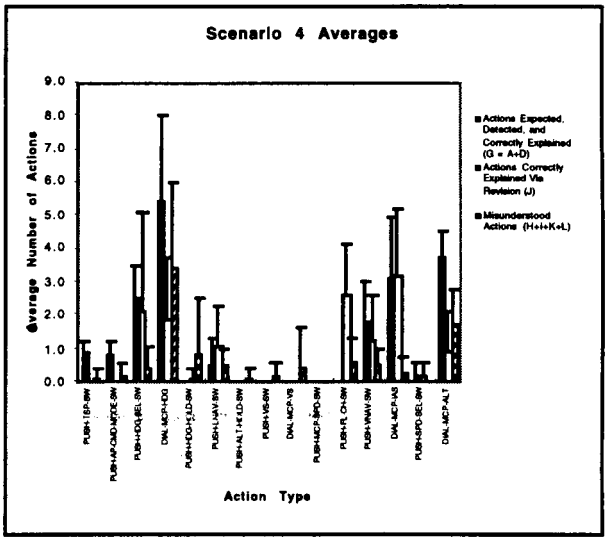

Subject 7 Averages
Average Number of Actions
Action Type
Actions Expected, Detected, and Correctly Explained (G = A+D)
Actions Correctly Explained Via Revision (J)
Misunderstood Actions (H+I+K+L)

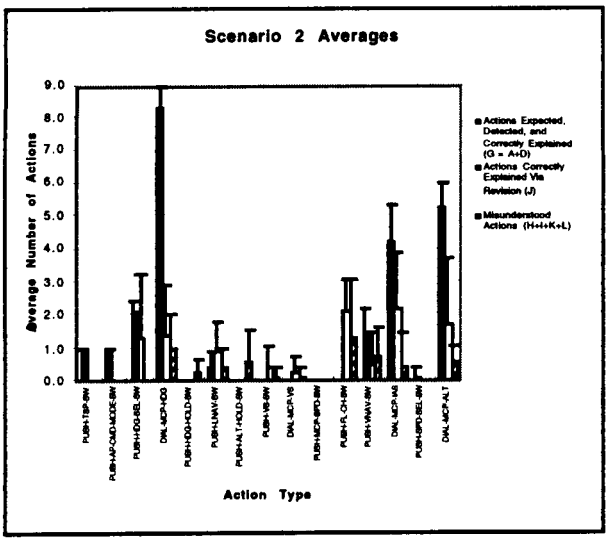

Subject 9 Averages
Average Number of Actions
Action Type
Actions Expected, Detected, and Correctly Explained (G = A+D)
Actions Correctly Explained Via Revision (J)
Misunderstood Actions (H+I+K+L)

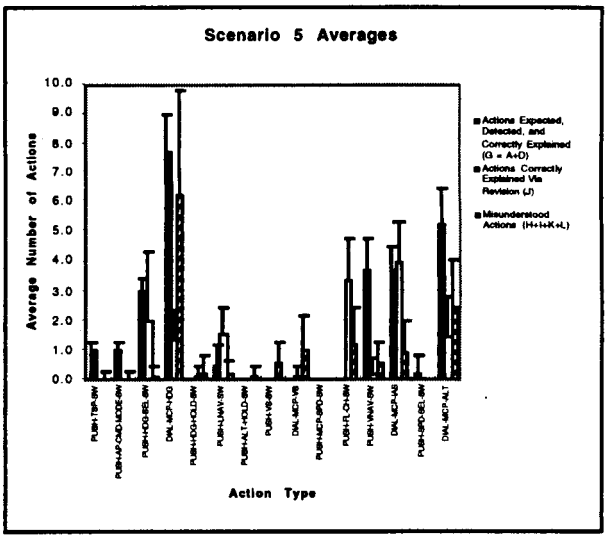

Subject 8 Averages
Average Number of Actions
Action Type
Actions Expected, Detected, and Correctly Explained (G = A+D)
Actions Correctly Explained Via Revision (J)
Misunderstood Actions (H+I+K+L)

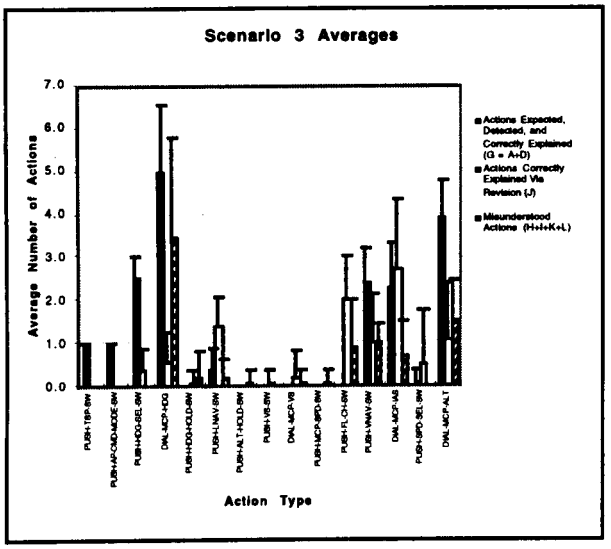

Subject 10 Averages
Average Number of Actions
Action Type
Actions Expected, Detected, and Correctly Explained (G = A+D)
Actions Correctly Explained Via Revision (J)
Misunderstood Actions (H+I+K+L)


Scenario 1 Averages
Average Number of Actions
Action Type
Actions Expected, Detected, and Correctly Explained (G = A+D)
Actions Correctly Explained Via Revision (J)
Misunderstood Actions (H+I+K+L)


Scenario 4 Averages
Average Number of Actions
Action Type
Actions Expected, Detected, and Correctly Explained (G = A+D)
Actions Correctly Explained Via Revision (J)
Misunderstood Actions (H+I+K+L)


Scenario 2 Averages
Average Number of Actions
Action Type
Actions Expected, Detected, and Correctly Explained (G = A+D)
Actions Correctly Explained Via Revision (J)
Misunderstood Actions (H+I+K+L)


Scenario 5 Averages
Average Number of Actions
Action Type
Actions Expected, Detected, and Correctly Explained (G = A+D)
Actions Correctly Explained Via Revision (J)
Misunderstood Actions (H+I+K+L)


Scenario 3 Averages
Average Number of Actions
Action Type
Actions Expected, Detected, and Correctly Explained (G = A+D)
Actions Correctly Explained Via Revision (J)
Misunderstood Actions (H+I+K+L)

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	June 1999	Technical Memorandum

4. TITLE AND SUBTITLE
GT-CATS: Tracking Operator Activities in Complex Systems

5. FUNDING NUMBERS
NCC2-824

6. AUTHOR(S)
Todd J. Callantine, Christine M. Mitchell, and Everett A. Palmer

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Ames Research Center, Moffett Field, California 94035

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
NASA/TM—1999–208788

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Subject Category: 03-01, 63-02 Distribution: Public
Availability: NASA CASI (301) 621-0390

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

Human operators of complex dynamic systems can experience difficulties supervising advanced control automation. One remedy is to develop intelligent aiding systems that can provide operators with context-sensitive advice and reminders. The research reported herein proposes, implements, and evaluates a methodology for activity tracking, a form of intent inferencing that can supply the knowledge required for an intelligent aid by constructing and maintaining a representation of operator activities in real time. The methodology was implemented in the Georgia Tech Crew Activity Tracking System (GT-CATS), which predicts and interprets the actions performed by Boeing 757/767 pilots navigating using autopilot flight modes. This report first describes research on intent inferencing and complex modes of automation. It then provides a detailed description of the GT-CATS methodology, knowledge structures, and processing scheme. The results of an experimental evaluation using airline pilots are given. The results show that GT-CATS was effective in predicting and interpreting pilot actions in real time.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Activity tracking, Intent inferencing, Glass cockpit aircraft, Aiding/tracking systems	192
	16. PRICE CODE A09

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	Unlimited