
**MODELING HUMAN-MACHINE SYSTEMS:
ON MODES, ERROR, AND PATTERNS OF INTERACTION**

Asaf Degani

School of Industrial and Systems Engineering

Georgia Institute of Technology

Atlanta, GA 1996

DEDICATION

In memory of William (Bill) Reynard—a kind man, lawyer, and a pilot—who dedicated his career to the pursuit of aviation safety

ACKNOWLEDGMENTS

Solum Volamus.

Latin: “We fly alone.”

(The motto of a USAF Lockheed U-2 squadron)

But we rarely do. As in any other endeavor, the closure of this research effort masks the many who—in their honest efforts—sustained our wings and allowed us to journey. It is therefore more than appropriate to start by thanking the many people that were there for me—in the beginning, midway, and at the end.

I’m indebted to Alex Kirlik for helping me make the decision to pursue doctoral studies at the Georgia Institute of Technology in Atlanta and for advising, supervising, and supporting the writing of this thesis. Slowly, but surely, he taught me the value of an ecological perspective for describing and understanding human-machine systems. There is much of him in this work. Christine Mitchell provided much support and encouragement along the way, especially at those times when I experienced brief periods of euphoria and long periods of frustration with various modeling approaches. Chris was there to state her belief that “it can be done, and you can do it.” My NASA Ames co-advisor and former boss, Michael Shafto, was there on a daily basis helping brainstorm and providing encouragement. Much of the sophisticated statistical analysis of this research is a product of his knowledge, intuition, and hard labor. Last, my conceptual approach for describing and analyzing human-machine interaction has changed and re-focused following many (and sometimes heated) discussions with Michael Heymann. His approach to formal specification and modeling of interactive automation runs as a thread throughout this thesis.

I owe much in terms of support and help to my colleagues, friends, and members of the research community: Everett Palmer, Kevin Jordan, Dan Fisk, Mark Rosekind, Michael Feary, T. Govindaraj, Todd Callantine, Victor Lebacqz, Corwin Logsdon, Alan Chappell, Peter Polson, Tony Andre, David Gaba, David Harel, Helen Tait, Michael Hayes, Alan Price, Lloyd Sauls, and the people at the Center of Human Machine Systems Research (CHMSR) laboratory at Georgia Tech.

Special thanks are due to Jim Lockhart and Maria Romera of NASA Ames for their hard and dedicated work in modeling all the case studies in the Statemate (a computer-aided software engineering) tool, drafting many of the statistical graphics, and maintaining the database of the field study. Rowena Morrison, from the Aviation Safety Reporting System office, provided much-needed editorial support and a sense of humor to this project. Acknowledgment is also due to the cooperating airline and their pilot union group for approving this research and allowing data collection. Thanks is especially due to some one hundred anonymous airline pilots who allowed me to collect data about their interaction with the Automated Flight Control System, and provided valuable insights into the problem.

Finally, there are personal acknowledgments. I am grateful to Earl Wiener for introducing me to the “exotic” (his word) area of cockpit automation, and the use of field studies for research. Earl has always been there both personally and professionally—he has my deep appreciation for his

guidance and support. I thank my companion in life, Hila, for her many sacrifices throughout this long process. Her understanding and (sometimes therapeutic) advice sustained me throughout this journey. I owe her a great debt, and can now take satisfaction in offering her sustenance, in kind, as she embarks on a similar voyage.

Moffett Field, CA

December, 1996

PREFACE

This research is about the interaction between humans and modern control systems and the ambiguity and confusion that may result from this interplay. Its particular focus is modes. The selection of this research topic is hardly coincidental.

In the early 1980s I was a Weapon Systems Officer responsible for operating large and complex fire control systems onboard missile corvettes. Such systems have several components that operate concurrently, each component with its own set of modes. Understanding the status of the system as a whole is far from trivial (mathematically, it requires a generation of some large-state and mode-configuration vectors). During this period I encountered many mode problems in fire control systems, some of them deadly. For example, an automatic gun had two stages before a shell was ready to fire: “on tray” prior to entering the barrel, and inside the barrel. Most jams would occur when the shell was stuck on the tray. However, no indication was provided as to whether the shell was stuck on the tray or was in the barrel—the display was ambiguous about these two states. Since one could not see the internal mechanism of the gun, the only way to resolve this ambiguity was by pressing the trigger: If a projectile came out, the operator knew it was in the barrel. Otherwise, it was on the tray and jammed. At the time, fellow officers and I would jury-rig improvised designs to modify the interface and avoid such dangerous ambiguity. Running extra wiring between the mechanism and the interface to indicate previously unannounced modes were common “fixes.” These jury-rigged designs were illegal according to military standards and procedures, but were so vital that they were done undercover, regardless.

Another pivotal experience with modes occurred in 1990, when I participated in a special meeting at NASA Ames Research Center regarding human interaction with automated control systems. The subject of the meeting was the crash of an Indian Airlines A-320 in Bangalore. Two senior pilots from Air India—appointed as technical assessors by the Court of Inquiry—visited NASA Ames to discuss the human factors aspects of this accident. The main discussion items were the pilots’ interaction with autoflight modes, and the reaction time for recovery from an unsuitable mode configuration. Specifically, the assessors wanted to know why the pilots had engaged an inappropriate mode configuration for the situation, and why they did not realize its unsuitability as the speed decayed from the managed speed of 132 knots to below 106 knots, given the cues in the cockpit.

The meeting lasted for several hours but went in wide circles. The reason? No actual research information about these problems was available—the topic was hardly studied. Finally, somewhat frustrated and quite emotional, one of pilot-assessors stood up and demanded: “Is there anything, *anything*, you can tell us about why two highly experienced pilots got into, and could not recover from, such a mode configuration?” There was a long silence.

In 1992 I took leave from NASA Ames and left for Atlanta, Georgia to pursue doctoral studies at the Georgia Institute of Technology. The topic I chose was human interaction with modes.

One often reads in research methods books that the investigator must be objective, remote from the research topic, and have no emotional link to it. This work is far from compliance with this research requirement. It represents a personal drive (and at times a struggle) to understand what modes are, how one can represent them, and the extent of their contribution to confusion and harm in everyday devices, machines, and large-scale supervisory systems.

TABLE OF CONTENTS

MODELING HUMAN-MACHINE SYSTEMS:	I
ON MODES, ERROR, AND PATTERNS OF INTERACTION.	I
DEDICATION.	I
ACKNOWLEDGMENTS	II
PREFACE	IV
TABLE OF CONTENTS	VI
LIST OF TABLES	IX
LIST OF FIGURES.	X
SUMMARY	XII
INTRODUCTION.	1
MOTIVATING EXAMPLES	2
SUMMARY	3
RESEARCH OBJECTIVES	3
THESIS OUTLINE.....	3
A REVIEW OF MODES IN HUMAN-MACHINE SYSTEMS	5
MODES IN HUMAN-COMPUTER INTERACTION	6
MODES IN MACHINES AND DEVICES	12
MODES IN SUPERVISORY CONTROL SYSTEMS.....	13
CHAPTER SUMMARY	21
MODELS OF HUMAN-MACHINE SYSTEMS.	23
ABSTRACTION HIERARCHY MODEL.....	23
PROCRU MODEL.....	24
GOMS MODEL.....	25
FINITE STATE MACHINE (FSM) MODEL	27
PETRI-NETS.....	30
DISCRETE CONTROL NETWORK	31
THE OPERATOR FUNCTION MODEL (OFM).....	33
CHAPTER SUMMARY	36

STATECHARTS AND OFAN: A LANGUAGE AND MODELING FRAMEWORK.....	39
STATECHARTS: A LANGUAGE	39
OFAN: A PROPOSED FRAMEWORK.....	45
CHAPTER SUMMARY	47
CASE STUDIES.....	49
CORDLESS PHONE	49
ELECTRONIC CLOCK/RADIO.....	54
CRUISE CONTROL.....	58
AUTOMATIC BLOOD PRESSURE DEVICE	63
MAP DISPLAY	71
CHAPTER SUMMARY	77
CATEGORIZATION OF MODE FEATURES	79
INTRODUCTION	79
TWO PLANT STATES—ONE DISPLAY STATE.....	79
HIDDEN TRANSITION.....	83
CIRCULAR MODE TRANSITION	86
MODE-REFERENCE VALUE INTERACTION	88
MODE-MODE INTERACTION.....	92
CHAPTER SUMMARY	95
MODE USAGE IN A COMPLEX SYSTEM I: FIELD STUDY AND STATISTICAL ANALYSIS.....	97
INTRODUCTION	97
THE COMMERCIAL AVIATION DOMAIN.....	98
APPROACH AND OBJECTIVES	99
METHOD	100
RESULTS	120
MODE USAGE IN A COMPLEX SYSTEM II: OFAN ANALYSIS OF A SCENARIO	153
INTRODUCTION	153
METHOD	154
SPEED MODE SCENARIO.....	154
MODEL	157
ANALYSIS	164
CHAPTER SUMMARY	167

CONCLUSIONS.....	169
CONTRIBUTIONS	170
FUTURE RESEARCH	174
CLOSING REMARKS.....	175
REFERENCES.....	177

LIST OF TABLES

TABLE 2-1. THE EDIT COMMAND	7
TABLE 4-1. SWITCH (A AND B) AND LIGHT-FIXTURE STATE CONFIGURATIONS	43
TABLE 5-1. TABULATION OF STATES AND EVENTS	52
TABLE 5-2. MACHINE AND HUMAN INVOLVEMENT STATES	61
TABLE 5-3. STATE CONFIGURATION	70
TABLE 7-1. PITCH MODES AND DISPLAY FORMAT	106
TABLE 7-2. ROLL MODES AND DISPLAY FORMAT	106
TABLE 7-3. TMC MODES AND DISPLAY FORMAT	107
TABLE 7-4. AFCS SUPERVISORY MODES AND DISPLAY FORMAT	107
TABLE 7-5. VERTICAL CLEARANCES	108
TABLE 7-6. LATERAL CLEARANCES	109
TABLE 7-7. SPEED CLEARANCES	109
TABLE 7-8. DEPENDENT AND INDEPENDENT VARIABLES FOR STATISTICAL ANALYSIS (ASTERISK DENOTES A REFERENCE CATEGORY)	127
TABLE 7-9. REGRESSION RESULTS	132
TABLE 7-10. CORRELATION OF VARIATE PAIRS	147

LIST OF FIGURES

FIGURE 2-1. TRANSFORMATIONS	6
FIGURE 4-1. HOUSEHOLD THERMOSTAT (IN TRANSITION STATE DIAGRAM).....	40
FIGURE 4-2. HOUSEHOLD THERMOSTAT (IN STATECHARTS).....	40
FIGURE 4-3. LIGHT SWITCH AND FIXTURE (IN STATECHARTS)	41
FIGURE 4-4. TWO LIGHT SWITCHES — ONE FIXTURE	43
FIGURE 4-5. SYNTHESIS OF THE FIVE ELEMENTS INTO A FRAMEWORK	45
FIGURE 5-2. MODEL OF THE CONTROL MECHANISM OF THE CORDLESS PHONE.....	51
FIGURE 5-4. MODEL OF ELECTRONIC CLOCK/RADIO	56
FIGURE 5-5. CRUISE CONTROL OF A CAR	59
FIGURE 5-6. CRUISE CONTROL MODEL	60
FIGURE 5-7. LINE DRAWING OF THE AUTOMATIC BLOOD-PRESSURE DEVICE.....	64
FIGURE 5-8. MODEL OF THE BLOOD PRESSURE DEVICE.....	66
FIGURE 5-9. MODEL OF THE CONTROL MECHANISM (BLOOD PRESSURE DEVICE)	68
FIGURE 5-10. MODE- AND RANGE-SELECTOR KNOBS.....	72
FIGURE 5-11. ‘ARC’ AND ‘ROSE’ MAP FORMATS.....	72
FIGURE 5-12. MODEL OF THE MAP DISPLAY	74
FIGURE 6-1. A “UNIVERSAL” REMOTE CONTROL.....	80
FIGURE 6-2. TWO PLANT STATES — ONE DISPLAY STATE (MANUAL TRANSITION).....	81
FIGURE 6-3. TWO PLANT STATES — ONE DISPLAY STATE (AUTOMATIC TRANSITION).....	82
FIGURE 6-4. TWO PLANT STATES — ONE DISPLAY STATE (DEFAULT TRANSITION)	83
FIGURE 6-5. HIDDEN TRANSITION (.OR. CONDITION).....	84
FIGURE 6-6. HIDDEN TRANSITION (.AND. CONDITION).....	85
FIGURE 6-7. CIRCULAR MODE.....	87
FIGURE 6-8. REFERENCE VALUE CAUSES A MODE CHANGE	89
FIGURE 6-9. MODE TRANSITION CAUSES A CHANGE IN REFERENCE VALUE.....	90
FIGURE 6-10. DEFAULT TRANSITIONS IN REFERENCE VALUES AND/OR MODES.....	92
FIGURE 6-11. MODE-MODE INTERACTION (WITH MANUAL TRANSITION)	94

FIGURE 6-12. MODE-MODE INTERACTION (AUTOMATIC TRANSITION)	95
FIGURE 7-1. COMPONENTS OF THE AFCS	103
FIGURE 7-2. INTERFACE TO THE AFCS	104
FIGURE 7-3. LOCATION OF MODE ANNUNCIATIONS IN THE COCKPIT	105
FIGURE 7-4. QUESTIONNAIRE FORM	112
FIGURE 7-5. FLIGHT HOURS IN THE B-757/767 AIRCRAFT TYPE	113
FIGURE 7-6. TOTAL FLYING HOURS IN PART 121 CARRIERS	114
FIGURE 7-7. HOURS IN AIRCRAFT EQUIPPED WITH AN AFCS OR ONLY A FLIGHT MANAGEMENT COMPUTER (L-1011)	115
FIGURE 7-8. DATA SHEET USED BY THE OBSERVER	118
FIGURE 7-9. THE STRUCTURE OF THE DATA FILE	119
FIGURE 7-10. THE MODE CONFIGURATION SPACE (PITCH AND ROLL) OF THE AFCS	122
FIGURE 7-11. MODE TRANSITION DIAGRAM	124
FIGURE 7-12. ORDINAL RANKING OF THE 45 MODE CONFIGURATION OF THE AFCS	130
FIGURE 7-13. PLOT OF EXTERNALLY STUDENTIZED RESIDUALS VS. PREDICTED Y VALUES (AND A NORMAL PROBABILITY PLOT)	134
FIGURE 7-14. THE FIRST VARIATE (STRUCTURE CORRELATION)	140
FIGURE 7-15. THE SECOND VARIATE (STRUCTURE CORRELATION)	142
FIGURE 7-16. THE THIRD VARIATE (STRUCTURE CORRELATION)	144
FIGURE 7-17. THE FOURTH VARIATE (STRUCTURE CORRELATION)	146
FIGURE 8-1. MODE TRANSITION (SCENARIO)	156
FIGURE 8-2. MODE CONTROL PANEL OF A BOEING B-757	158
FIGURE 8-3. ENVIRONMENT, USER TASK, AND INTERFACE MODULES	159
FIGURE 8-4. CONTROL MECHANISM MODULE	162

SUMMARY

In many control systems used today, modes are used as a way of providing several configurations (or functions) via the same machine or device. Modes, therefore, can be defined as the manner of behavior of a machine. A complex machine may have many interacting components, each one with its own set of modes. It appears, however, that human interaction with these mode-based control systems is problematic. Mode ambiguity, a feature of the design that prevents the user from predicting the next mode of the system, may lead to mode confusion. Such confusion has been attributed as a cause in many incidents and several accidents.

In this research we have attacked the mode confusion problem by developing a modeling framework (Ofan) to describe human interaction with mode-based systems. The Ofan framework contains five interacting modules describing the environment, user tasks, interface, control mechanism, and plant. The framework is represented in Statecharts—a modern extension of the Finite State Machine model. Using this framework, we have modeled a variety of mode-based systems and identified some of the design features that may contribute to mode confusion. We have then captured some of these features in a set of abstract categories—all represented in the Statecharts language.

To further understand human interaction with complex and highly automated control systems, we also conducted a field study. The study documented pilots' interaction with the Automated Flight Control System of a Boeing B-757 aircraft. Its focus was on the sequence of actions involved in configuring modes and setting reference values for the machine. Using a variety of statistical techniques, we identified the pilots' patterns of interaction with the system. The results indicate that pilots use only a small subset of all possible mode configuration in operating the machine. In essence, they convert a complex machine to a simple machine.

The analysis identifies the several factors that constrain this conversion process. Our findings point to a very strong relationship between mode selection on part of the pilot and the structure of the operating environment. We discuss in detail the implications of these findings for the use and future design of such automated control systems. Finally, we combine our modeling and statistical resources in analyzing a mode confusion incident that was observed and documented during the field study. Superimposing pilot sequences of actions on top of the Ofan model provided the much sought after view of the human-machine interaction in the context of the environmental demands and the detailed behavior of the machine.

The problem of human interaction with modes is not a phantasm—it is as real and complex as any construct that leads to error. The use of modes and their corresponding architecture in modern control systems will only increase as more functional and automated systems are being developed. In this dissertation we have provided a variety of methodologies for describing, analyzing, and evaluating human interaction with mode-based systems. We hope that these methodologies will allow engineers and designers to understand mode ambiguity, identify it, and provide safeguards against it.

INTRODUCTION

Much attention has recently been devoted to the issue of human interaction with complex, dynamic, and highly automated control systems. In this category we find such platforms as aircraft, ships, and military command and control systems. The human factors research community has learned from case studies and empirical investigations that increased levels of automation are not panaceas (Billings, 1996). In such control mechanisms, the interface to the user may not always provide sufficient information on the status and behavior of various system elements, or there may exist non-trivial relationships between components of the control mechanism. In some cases, such features of the design limit the operator's ability to anticipate the future status of the machine, given an action (e.g., requesting a mode or some reference-value change) or the occurrence of an external event in the environment (e.g., reception of an electronic signal). This ambiguity, with respect to the future behavior of the machine, may lead to operator confusion and subsequent error. Sometimes the error leads to an incident, and in some rare situations to an accident. In many cases, such ambiguity is related to the modes of the control mechanism—a common method for user interaction with the various behaviors of the control system.

Manifestations of mode ambiguity are not confined to complex systems, however. Modes are a characteristic of any system, device, or control mechanism. Every control mechanism has states and modes. The two cardinal superstates, or modes, of almost any device are “On” and “Off”—defining two different ways of behaving.

With technological advances machines have become more functional—i.e., they incorporate more behaviors. Some devices have long been associated with multiple functions. For example, many electronic wrist watches have time, alarm, timer, and chronometer functions. Other machines or products that long afforded only one function now can function (behave) in many different, often mutually exclusive, ways. For example, fax machines no longer just fax—they now copy and print as well. Answering machines no longer have a single mail box, but commonly have three. Yet these multiple modes have been “mapped” to fewer controls (i.e., buttons) and displays. As a consequence of this many-to-one mapping, the potential for mode problems has increased.

Machines hardly ever function independently. They are always used in the context of an *operational environment* (control room, business office, kitchen) and a *user* who interacts with and manipulates them. These three components—environment, user, and machine—define our system of interest. It appears, however, that the increase in a machine's functionality demanded by the operational environment is not paralleled by increased facility on the part of the users. Users of multi-functional machines are often frustrated as a consequence. The common example, and an all-too-familiar target for these frustrations in consumer products, is the Video Cassette Recorder (VCR). Programming the interaction between the user and the VCR is not easy to learn (Mark and Greer, 1995). And indeed, in the vast majority of homes, many functions of the VCR are left unused.

MOTIVATING EXAMPLES

As part of this research, we have considered six human-machine systems of increasing complexity, with known mode problems. In order of their increasing complexity and dynamic characteristics, these human-machine systems are: (1) a cordless telephone, (2) an electronic clock/radio, (3) an automobile cruise control, (4) an automatic blood pressure machine, (5) a “moving map” display on an aircraft, and (6) the Speed Mode in the Automatic Flight Control System of a modern aircraft.

(1) Cordless Telephone

The cordless telephone example is taken from the domain of consumer telecommunication products, and is an example of a simple and relatively static human-machine environment. In this specific model, the user may hang up on the caller when the phone is picked up from its base and the user presses the ‘Talk’ button. We use this example to show how the same sequence of actions may lead to different outcomes. These outcomes, in turn, are dependent on the state configuration of the phone, which is not easily discernible by the user.

(2) Electronic Clock/Radio

The electronic clock/radio is another relatively simple example from the domain of consumer electronic products. Here, if the user accidentally sets the volume low or mistunes the radio station, the “Alarm” mode will engage, yet the user will not perceive this. We use this example to show the important relationship between a mode and its reference value (e.g., volume).

(3) Automobile Cruise Control

This example introduces a control system. It is a dynamic human-machine system consisting of five levels of involvement with respect to speed control. These levels range from fully manual to automatic. In the specific cruise control system described, the user believes that the cruise control is deactivated, when in fact it is still engaged in the background. Lifting one’s foot off the gas pedal can lead to two very different outcomes: the speed falls to zero, or the cruise control re-engages. Without keeping a log of his or her prior actions, the driver cannot resolve between these two, very different, outcomes. The interface is insufficient to resolve this ambiguity.

(4) Automatic Blood Pressure Device

The automatic blood pressure device is an example from the domain of medical control systems. This device has several modes of operation and various reference values (e.g., measurement intervals) that must be set by the user. The mode and reference-value structure of this particular device led to a medical incident in which the anesthesiologist changed only the interval between measurement cycles, believing that the device was still active. Instead, the device went into “limbo” mode—in effect, it stopped measuring blood pressure. Yet the interface gave an indication that the device was still working at the newly set interval.

(5) An Aircraft Moving Map Display

This example is taken from the domain of commercial aviation, yet it focuses on a more generic problem in managing any displays that have modes. The example represents a mode confusion that occurred when the pilot switched between different display formats, or modes, only to find that the range of the display changed. A very unique interaction exists between the range and the mode of the display. The interface does not provide information about this relationship.

(6) B-757 Vertical Navigation Speed Mode

This example describes problems in the interaction between pilots and modes of the Automatic Flight Control System of a modern aircraft. During climb, the crew (at the request of Air Traffic Control) engaged a submode of a fully automatic mode which allowed them to take manual control over the speed of the aircraft. When the plane leveled off at its assigned altitude, the speed entered by the crew was maintained by the machine. Once the crew wanted to climb again to their new assigned altitude, they assumed that the aircraft would still maintain the manually entered speed. Instead, the speed reference value defaulted to an economy speed that provides fuel efficiency, but that was some 50 knots above the speed Air Traffic Control mandated.

SUMMARY

The examples presented here provide a glimpse into the world of human interaction with control systems that employ modes. We begin with a consideration of simple control mechanisms and move to highly automatic ones. Mode ambiguity—the inability of the user to resolve the next mode and reference value configuration of the machine—is a thread that runs throughout these examples. The underlying building blocks that prompt such ambiguity are what this research attempts to uncover. Several methodologies such as modeling, an observational study, and statistical analysis are brought to bear on this effort.

RESEARCH OBJECTIVES

The intent of this research is to better understand the interaction between humans and automated control systems that employ modes, and to identify the contribution of this human-machine interaction to mishaps. The focus of the research is on ergonomic aspects of this interaction. The research, therefore, does not directly deal with issues such as attention, perception, or cognitive process. Instead we set out to investigate the features of the control mechanism and interface that lead to mode ambiguity, and the relationship of these features to the users' knowledge about the system and the specification of the task. Our specific research questions, therefore, are the following:

- (1) What are modes?
- (2) Can we faithfully describe human-machine interaction in mode-based systems? Can this description “point” to specific features of the system that create mode ambiguity? Are there any methods that will allow us to identify these features before the device is built and fielded?
- (3) How do operators in one supervisory control domain—commercial aviation—use modes? Can we describe their patterns of interaction with the system? What factors constrain this interaction?

THESIS OUTLINE

This study is divided into three major parts. The first part (Chapters II–IV) presents a review of the literature. In Chapter II we review the literature on mode-based systems. In Chapter III we discuss some of the modeling approaches that are potentially useful for describing human interaction with modes. In Chapter IV we present the Statecharts modeling formalism and suggest a framework called Ofan.

In the second part of this document (Chapters V–VI), we apply the modeling framework and detail some of the specific features of the machine that lead to mode ambiguity. Chapter V describes the application of the framework to five examples (case studies) of mode-based systems. In Chapter VI we develop a categorization scheme of features that lead to mode ambiguity. We call these unique features the *structural elements* of the system architecture.

In the third part of this document (Chapters VII–IX), we describe a field study and analyze its data. The purpose of the study was to document how users interact with a very complex machine that exhibits many modes, the Automatic Flight Control System of a Boeing B-757 aircraft. In Chapter VII we focus on a statistical methodology for describing the sequence of actions and paths pilots took through the mode configuration space of the system. The chapter ends with a discussion of the research implications for designing new Air Traffic Control and Automatic Flight Control System. In Chapter VIII we detail a mode confusion scenario that was documented during the field study. We apply the Ofan framework to this scenario and employ the categories of structural elements in order to identify the features of the design that led to this incident. Finally, we superimpose the pilots' sequence of actions on the Ofan model to reveal the pattern of human-machine interaction. This study is concluded in Chapter IX with a brief summary of the research process, a discussion of this project's major contributions to the understanding of mode ambiguity and confusion, and recommendations for future research.

A REVIEW OF MODES IN HUMAN-MACHINE SYSTEMS

In this chapter we review the existing literature about human interaction with machines that employ modes. The review is organized historically and according to the domains in which modes are discussed. We start with general definitions and discuss the use of modes in the human-computer interaction (HCI) literature. Next, we briefly discuss modes in consumer electronics and medical devices. Finally, we discuss the use of modes in supervisory control systems such as military fire control systems and aviation. We identify three types of modes that are universally employed in human-machine systems: (1) *interface modes* that specify the behavior of the interface, (2) *functional modes* that specify the behavior of the various functions of a machine, and (3) *supervisory modes* that specify the level of user and machine involvement in supervising the process. We conclude with some operational definitions for the purpose of this research and attempt to identify some of the missing pieces in the literature.

Definitions

The Webster's Dictionary (1994) defines "mode" as follows: "(1) manner of acting or doing; method; way. (2) the natural disposition or the manner of existence or action of anything: form." The English term is based on the Latin word *modus*—meaning measure or manner. A related compound term is "modus operandi," indicating a way of doing or accomplishing something. An organization or a group is said to develop, use, and re-use a modus operandi, or a mode of operation, to achieve its goals or deal with some external demand.

The early use of "mode" in the literature is probably related to sixteenth- and seventeenth-century philosophical texts, where the term is commonly used to denote the properties or attribute of a "substance." In contrast, Spinoza's use of the term denotes an entity (Wolfson, 1934). The relation of a mode to a substance corresponds to that of a species to a genus, where a mode of a substance is one of its many manifestations. If we accept Wolfson's interpretation, a mode is an entity with some characteristics of its source, and not just an attribute of the source. Such modes represent a definitive structure as opposed to just an attribute.

Ashby, in his classical text *An Introduction to Cybernetics* (1956/1964), set forth the construct of a "machine with input." A machine may have several states denoted by *operands* (a, b, c, d). A *transformation* (a set of transitions by a set of operands) can be defined so that, for example, operand *a* undergoes a transition and becomes *c*; operand *b* transitions to *d*, operand *c* to *d*, and operand *d* to *b*. This transformation is defined as transformation M_1 . But there might be another transformation, for example M_2 in which $a \rightarrow b$, $b \rightarrow a$, $c \rightarrow d$, and $d \rightarrow c$. These relationships are illustrated in Figure 2-1.

↓	a	b	c	d
M1	c	d	d	b
M2	b	a	d	c

Figure 2-1. Transformations

The two transitions, M_1 and M_2 , if embodied in the same physical body, would have to correspond to a machine with two different ways of behaving (Ashby, 1956/1964, p. 42). Can a machine have two ways of behaving? It can, if the conditions under which the machine works can be altered, and if the machine has a (mode) switch to alter its behavior. Referring back to Figure 2-1, M_1 corresponds to a mode switch set to position 1, and M_2 to position 2. The change of M 's subscript from 1 to 2 is a transformation from one manner of behaving to another. M 's subscript, or any other symbol whose value determines which transformation will be applied to M 's basic states, is called a *parameter*. A real machine whose behavior can be represented by such a set of closed single-value transformations is defined by Ashby as a machine with input.

Ashby's machine with several modes, each having a different behavior, is one of the early treatments of this topic. It provides us with the theoretical groundwork to proceed on our search.

MODES IN HUMAN-COMPUTER INTERACTION

The issue of human interaction with modes started to appear in the electrical engineering and computer-related literature sometimes in the late 1970s (see Tesler, 1981, for a review). These papers did not specifically discuss human interaction with modes, but instead presented various design approaches for implementing modes in the system code and computer interface. The impetus for this research was that as more and more functions were added to word processors and other applications, the size of the standard keyboard and the number of available function keys stayed constant. The general approach discussed in those early papers was to use the same key to engage several commands. This approach was implemented by providing the user with a button to switch the application from one mode of operation to another. For example, when the early text editors were in "insert" mode, any text entry was inserted after the location of the cursor and existing text was pushed forward; in "replace" mode, the new text replaced (and therefore deleted) the existing text.

A somewhat more colorful example of the potential unwanted consequences of such modes is one early computer application in which the main typing keys were normally interpreted as commands. The "i" key invoked the Insert command, which put the system in "insert mode." In the insert mode, the application interpreted keystrokes as letters. The story goes that a user intended to type the word "edit" into the document, but forgot to enter the insert mode first. The application interpreted "edit" as the following set of commands (Smith, Irby, Kimball, Verplank, and Harslem, 1982, p. 276):

Table 2-1. The Edit Command

Command	Result
E (everything)	Select all words in document
D (delete)	Delete selected words
I (insert)	Enter insert mode
T	Type a “t”

Users, not surprisingly, were not happy with such an implementation—in particular if it wiped out the entire document (with no way to “undo”). Tesler (1981) captured this growing frustration in his influential *Byte* magazine article with this plea: “Don’t mode me in.” By then software developers were starting to recognize the problem, and solutions appeared: the SmallTalk environment (Tesler, 1981), the Xerox STAR (Miller and Johnson, 1995; Smith, et al., 1982), and later the Apple Computer interface (Card, 1995). Nevertheless, modes did not go away. Even applications which at first may seem *modeless*, do indeed have modes. The various application display formats and cursor control functions of the computer mouse are what Tesler calls “modes in sheep’s clothing.” Norman (1983) generalized this point by observing that modes will be with us as long as we wish to be able to do more operations than we have special keys (p. 255).

Mode Problems and Concerns

Tesler (1981) defines a mode in the context of an interactive system as a state of the user interface that lasts for a period of time, is not associated with any particular [display] object, and has no role other than to place an interpretation on operator input. In the same spirit, Poller and Garter (1983; 1984) state that modes refer to the application’s interpretation of the user’s input—there is a mode change each time the interpretation of the same input changes. Sneeringer (1978) notes that computing systems and their interfaces have many modes because different routines can be in control of the application—the mode specifies which routine is currently in control.

Early papers on the topic of human interaction with interactive systems usually mention (although sometimes not explicitly) the topic of modes and possible confusion. Parnas (1969) suggested that if a user does not know what state he or she is in, and what the system interpretation is depending on the state, the system becomes difficult to use (p. 380). Hansen (1971) suggests that an action (pressing a button) should not have more than a few state-dependent meanings. Similarly, Foley and Wallace (1974) call for the reduction of unrecognizable side effects resulting from a user’s action. Such reduction is achieved by making these side effects immediately perceivable to the user (via visual, audio, and kinesthetic channels) so as to reinforce the effects of mode change (p. 466).

Poller and Garter (1984) experimented with ‘moded’ (*vi*) and ‘modeless’ (*emacs*) text editors and linked mode errors to the types of tasks performed by the user (copying text, editing, composing). Their results suggest that moded text editors may be preferable for editing tasks and modeless editors may have some advantages for free composition. They found that participants using the “modeless” editor made responses that fit the definition of mode error: Users clearly intended commands such as <escape>b or <control>b, but instead entered the corresponding letter (b) into the text (p. 168). Kieras and Polson (1982) were concerned about situations in which the user cannot infer the current state of the device by examining the display. In their formulation of user complexity, they term this condition the “hidden state problem” (p. 8).

Newman and Sproull (1979) state that the more modes in a command language, the more likely the user is to make mistakes by forgetting which mode he or she is in (p. 451). Sneeringer (1978) focuses on mode occupancy, stating that mode confusion is more likely to occur when the user spends a long time in a mode other than the usual one, and therefore is not accustomed to its unique behavior (p. 556). Swinehart (1979, cited in Thimbleby, 1982) goes one step further, remarking that interactive systems should not have too many modes, and then uses modes as a measure for evaluating computer applications. Morton, Barnard, Hammond, and Long (1979) describe how the use of modes can lead to situations in which the (computer) system and the user's model of this interaction lose synchronization: the user now attributes incorrect effects to system function.

Mode Error

Donald Norman, in his influential 1981 paper "Categorization of Action Slips," addressed some of the above concerns and coined the term "mode error." Based both on his own experience and on the personal diaries of subjects who documented errors they made during the course of their day, Norman classified different types of errors. In his scheme, mode errors fall into a classification of errors in that involves forming and carrying out an intention—a slip. That is, when a situation is falsely classified, the resulting action may be one that was intended and appropriate for the analysis of the situation, but inappropriate for the actual situation (p. 6).

Mode Ambiguity

One of the first formal treatments of mode confusion was reported by Thimbleby (1982). Somewhat similar to Tesler (1981), he defines a mode as "an interpretation of user input." His analysis is focused on the level of single characters—e.g., the keystroke "d" can be interpreted (by the computer) as either the command "delete" or as literal text, depending on the mode. Therefore, the interpretation of the keystroke "d" depends on the two character modes, i.e., command and text. In this example, the two character modes are said to be overlapping and result in character-mode ambiguity. Thimbleby then develops a criterion for critical and non-critical character mode ambiguity. The distinction is based on the number of keystrokes the user must perform to recover from the unwanted consequence (before any non-visual diagnostic is provided). By focusing his analysis strictly on the features of the computer application, Thimbleby is able to develop a clear-cut measure for evaluating the interface quality of a given computer application, with respect to mode ambiguity.

Monk, a colleague of Thimbleby, remarks that mode ambiguity exists when some action on the part of the user *can* have different effects on the machine depending on its current state (1986, p. 314). He goes one step further by interjecting the notion of user expectations. He states that "mode errors occur when the action of the user leads to an *unexpected* effect, when it could have produced the expected effect, had the machine been in the appropriate state" (p. 314). Therefore, mode ambiguity will only result in mode error when the user has false expectation about the result of some action.

Monk makes a distinction between two types of mode ambiguity: the type that leads to a problem in expectations, and the type that does not. As an example of a feature that leads to mode ambiguity, he describes a time-sharing computer operating system. Here, keystrokes are buffered until the "Return" or "Enter" key is pressed. When the buffer gets full, all subsequent keystrokes are ignored. Since this feature leads to different outcomes as a function of the system state, and this state is unlikely to be expected by the user, it must be considered as mode ambiguity. The user's

action—hitting a key and seeing nothing on the screen (because the buffer is full)—is therefore a “mode error.”

With respect to mode ambiguity that will *not* lead to mode error, Monk distinguishes between two subcases: First, if the user is aware of, or knowledgeable about, the dependency between states, then the correct expectation is formed and no error occurs. Or, if the user has no awareness of the dependency between states, no expectation of the precise effect of the mode will be formed, and thus no “error” will ever be noted. An example of this is the end-of-line algorithm used in many text editors: In this mode the user need not press the “Return” key to end the current line and start the next line—this is done automatically by the end-of-line algorithm. An ambiguity is introduced because the departure point between including the last word on the current line or scrolling to the next line is unknown to the user (e.g., a four-letter word will still stay on the current line but a five-letter word will be scrolled to the next line). As long as the algorithm works reasonably well, the user will not complain. Why? Because he or she has not formed an expectation about which word will stay or scroll down.

To address the problem of mode ambiguity and test countermeasures, Monk conducted an experimental study. He gave his subjects a task that required transitioning between two modes of operation and entering appropriate reference values (depending on the mode). Subjects in the “experimental” condition heard a high tone when entering values in one mode, and a medium tone when entering values in the other. The subjects in the “control” condition were not given any auditory aid in distinguishing between the two modes. The results indicated that subjects in the experimental group performed better, and made less than a third of the errors performed by the control group. Monk suggested that a general increase of mode awareness (aided by different auditory tones) had led to development of better strategies to cope with mode ambiguity.

Monk suggests a three-step approach to understanding and predicting mode errors: (1) identifying the user action that constitutes the error, (2) identifying the user expectations about the effect of the action, and (3) defining the corresponding states (modes) of the system. He argues that it is these components which must be manipulated if the frequency of mode errors is to be reduced (p. 323). Solutions for reducing user errors are possible by (1) increasing the size of the input interface (i.e., more buttons, menus), and (2) making the user more cognizant of the general fact that a machine has several modes of operation.

Mode Feedback

Sellen, Kurtenbach, and Buxton (1992) argue that errors are not the only metric with which to measure users’ problems with mode identification. In some cases, users may diagnose the correct mode, but only after experiencing confusion or uncertainty. Therefore, an appropriate measure is one that reflects the cognitive effort or decision time required to deduce the system state (p. 143). Another metric of mode identification problems is recovery from error, and the relationship of this recovery rate to cognitive load and the efficiency of performing a repetitive task.

Similar to Monk’s work, Sellen *et al.* investigated the possibility of providing the user with salient feedback for identifying the system’s current mode. Sellen and her co-authors summarize the various dimensions along which feedback can be characterized (these dimensions are not necessarily orthogonal):

- (1) *Sensory modality of delivery*: visual, auditory, kinesthetic, etc.

- (2) *Reactive versus proactive feedback*: Does feedback occur when an action is executed? Can one use the feedback to determine the mode prior to taking an action?
- (3) *Transient versus sustained delivery*: Is the feedback sustained throughout a particular mode?
- (4) *Demanding versus avoidable feedback*: Can the user choose not to monitor the feedback?
- (5) *User-maintained versus system-maintained feedback*: Does the user actively maintain the mode?

In a series of experiments, Sellen *et al.* (1992) tested the hypothesis that sensory feedback used for indicating active modes can reduce both mode errors and cognitive load. Another hypothesis was that kinesthetic user-maintained feedback is a more effective method for preventing mode error than (avoidable) system-maintained feedback. Sellen and her associates argue that user-maintained feedback, delivered through muscular tension, reinforces the knowledge about the current state, as opposed to just “visual” feedback. Adding a foot pedal to a Sun workstation, they modified a *vi* text editor to fit their experimental manipulations: Text entry was possible only when the pedal was held down. To issue a command (save, open file, etc.), the user had to release the pedal. In addition, a “visual” condition was added. The user switched mode by hitting a key on the keyboard, and the screen color changed to *dark pink* while the user was in the edit mode.

The results indicated that both pedal and visual feedback were beneficial in reducing mode error, but that pedal feedback was superior. Furthermore, there was an indication that using a foot pedal reduced the time necessary to diagnose the state of the system, suggesting that pedal feedback reduced the cognitive load imposed by the system. Since both types of feedback were sustained (present throughout the mode engagement) and proactive (feedback was provided before action was taken), there had to be some other features that could explain the differences in performance. Sellen *et al.* suggested the following:

- (1) *User-maintained versus system-maintained feedback*: By holding down the foot pedal, the mode was maintained by the user’s kinesthetic system. There was a binding among the kinesthetic action, the mode change, and the primary task (text entry) that provided an elegant continuity (Buxton, 1986).
- (2) *Kinesthetic versus visual feedback*: Visual feedback can be easily ignored; it is far more difficult to ignore kinesthetic feedback.
- (3) *Relationship between mode change and feedback*: In using the foot pedal, the body part that switched between modes was also the body part through which mode status was obtained. In the “visual” condition, one body part switched modes (hands/fingers pressed a key on the keyboard), and another body part obtained the feedback (eyes looked at the display).
- (4) *Distribution of tasks over body parts*: The foot pedal action was independent of the major task (typing). In the visual feedback condition, mode switching was done with the same body part as the major task (the user’s fingers were used for both typing and mode switching). Likewise, the task of monitoring mode state through visual feedback possibly interferes with feedback from the screen about the primary task—text entry.

Sellen and her co-authors remark that mode switching via the keyboard caused more errors, which in turn, resulted in additional recovery periods. This factor, they suggest, may have led to the differences in task completion times. The results indicated that experts were faster than novices under all conditions. Sellen *et al.* (1992) ponder whether experts have less “overhead” involved in

correcting errors. They further remark that expert *vi* users are constantly making such mode errors and are accustomed to recovering from them (see also Poller and Garter, 1984, for similar observations). Possibly because of the low “overhead” associated with such errors, experts users can afford to make more mistakes than novices, thereby trading speed for accuracy.

Sellen and her associates conclude that sustained feedback has led to the superiority of the foot pedal. They suggest that the visual channel is much overused: mode state information delivered through this modality is not salient enough compared to information delivered kinesthetically (even though the visual feedback involved changing the *entire* screen background to pink). Finally, based on the results of their experiments, they argue that mode information can be efficiently incorporated into human-computer interfaces by means of kinesthetic, user-maintained feedback. To some extent, such feedback is already used in existing interfaces: Examples of this are mouse-operated pull-down and pop-up menus, and depressing the mouse button when dragging icons or sweeping out areas on the screen (such as window boundaries). Conversely, the experiments suggest that carrying out these kinds of operations using a mechanism that latches while the user is relieved of the need to supply kinesthetic feedback (e.g., a Caps Lock key) will only tend to produce more mode errors (p.162).

Physical Interaction with Modes

The relationship between sustained feedback and a mode change is important, because in most cases, the desired mode behavior depends on both the mode and its associated reference value. Buxton (1986) states that the design of the syntax (such as the sequence of mode engagement and the entering of reference values) has a major effect on the quality of the user interface of an interactive system. It affects ‘learnability,’ the frequency and nature of users’ errors, the retention of skills, and the speed of task performance. A major problem for users is the cognitive load imposed by remembering the tokens of a command and their sequencing (p. 475).

One approach to this problem is to chunk together several actions into one higher-level task that is “glued” together by user feedback. The task of requesting a mode is a compound task of navigation and selection, but it appears as a single entity. Consider, for example, the task of selecting a menu item from a pull-down menu. The task consists of three subtasks: (1) invoking the menu, (2) navigating to the selection, and (3) making the selection. The sequence of actions appears as one task because it is glued together by the kinesthetic tension of pressing the mouse button throughout the transaction.

By designing the interaction with the interface in this way, the potential for mode errors is suppressed, since the concluding action (articulated by releasing the mouse button) is the unique and natural consequence of the initial action (depressing the mouse button). Moreover, the tension of the finger depressing the mouse button supplies constant feedback that we are in a given mode.

Consider, for example, the task of moving a file to the Trash icon on the “desktop” of the Macintosh Finder. This task is composed of several lower-level actions chunked into a single high-level action. This action, in turn, fits quite well into the user’s “mental model” of the task. By chunking lower-level actions, the user achieves the following:

- (1) The entire transaction—select, drag, and drop—is specified in a single gesture.
- (2) There will never be an error in syntax, since the ordering is implicit in the gesture.
- (3) The operation is specified using existing skills and does not require any new restructuring (p. 476).

Such chunking of both cognitive and kinesthetic efforts can reinforce the function that the user tries to accomplish, and hence reduce mode errors.

Summary

The topic of modes in the human-computer interaction literature affords us with a variety of conceptual tools relevant to this thesis topic. The terms mode error and mode ambiguity are defined, and related concepts such as cognitive load and error recovery are discussed. Specifically, the nature of mode ambiguity is discussed. Mode ambiguity will lead to confusion and possible error if the user's expectations are different than the actual status and behavior of the machine. Mode ambiguity will not lead to error, however, if the user somehow (even if by chance) develops an accurate expectation of mode transitions—or if the user has no expectations at all.

Experimentation has indicated the importance of various forms of user feedback. Performance differences between novices and experts were evaluated, indicating that experts tend to make more mode errors, but recover from them more quickly, thereby avoiding cognitive load and inefficiency. Finally, the notions of sustained feedback and chunking mechanisms were discussed in relation to designing the user interaction with modes and reference values. While these topics are sometimes neglected by researchers and designers of complex systems, much can be learned about methods for dealing with and designing modeless computer interfaces. One problem is that much of this information is considered proprietary, as part of a software company's competitive edge, and hence does not appear in the academic literature. Nevertheless, efficient solutions to common mode problems are already embedded in computer applications that we interact with daily.

MODES IN MACHINES AND DEVICES

Mode discussion in the human-computer interaction literature focuses on input/output relationships between the user and the computer, specifically on data entry and format of the interface. When we survey the use of modes in devices and machines, an additional type of mode is found. This mode refers to the active function of the device. For example, many consumer electronic devices have a variety of embedded functions: An audio system may have a compact disc (CD) player, cassette tape player, and radio. Selection of the appropriate function is conducted via a mode switch, which determines the functions, or modes, currently active. In addition, we may go one step further and say that our tape player has several modes with respect to the motor that turns the tape: "Play," "Fast Forward," and "Reverse." Each mode describes a different function, or more accurately behavior, of the motor. Such *functional modes* exist in almost any modern device or machine (in addition to interface modes that are also found).

Modes in Medical Devices

The medical domain is another setting for the application of human factors in general, and analysis of human interaction with automated devices in particular (Bogner, 1994). Modes have found their way into medical domains as more computer-based medical devices are introduced. These devices improve medical information and take over (automate) routine and manual activities, in some cases making these activities available to out-patient settings (Gaba, 1994; Klatzky, Kober, and Mavor, 1996, p. 17). Along with such automatic devices come modes, and the potential for mode ambiguity and error (Woods, Cook, and Billings, 1995). Indeed, many medical devices are highly moded, using multi-function buttons due to interface limitations (Cook, Woods, Howie, Horrow, and Gaba, 1992, p. 240; Lin et al., 1995, p. 12), as well as having distinct modes of operations (behaviors) that are not efficiently annunciated (Orasovich and Woods, 1995, p. 11–12).

Cook, Potter, Woods, and McDonald (1991) evaluate the human engineering aspect of a heated humidification device and identify ambiguous modes and alarm setting messages that have different meanings depending on the mode (p. 221–223). They define a mode as “a device state which controls or displays functions in a distinct way or has a distinct meaning” (p. 221). With this definition they move one step beyond the HCI definition, which focuses mostly on the distinct mode of the interface, into dealing with devices that employ distinct functions.

MODES IN SUPERVISORY CONTROL SYSTEMS

The term “supervisory control,” originally suggested by Sheridan and Farrell (1974), is commonly used to describe human-machine systems in which the human is supervising (but not necessarily manually controlling) the process. This, of course, requires that the system contain mechanisms capable of processing, acquiring, and disseminating information; exerting control; and making complex decisions (p. 431). Historically, supervisory control has been present in complex and high-risk domains where such monetary and organizational investment was justified (e.g., industrial process control, aviation, space). The literature describing modes in large supervisory control systems has traditionally focused on two domains: the military and aviation.

Modes in Military Control Systems

The military is one domain where the use of modes has proliferated. Even the term “mode arming,” used so commonly in the context of modes to indicate a request for mode engagement (which is predicated on several conditions that must be met before the mode transition occurs), originates from this domain’s jargon (Webster’s, 1994).

To our departure from the meaning of modes in HCI, we now add another type of mode—supervisory—sometimes also referred to as *participatory modes* (Wickens and Kessel, 1979), *control modes* (Modugno, Leveson, Reese, Partridge, and Sandys, 1996), and *modes of control* (Kirlik, 1993). Complex machines that operate in an uncertain environment usually allow the user to specify the level of human-machine involvement in controlling the process. A military fire control system is one example: While the system functional mode may stay the same (e.g., tracking an enemy aircraft), the user has the option of employing automatic, semi-automatic, or fully manual control of the process. The transition between supervisory modes is not trivial, in particular when several functional modes are concurrently active. For example, on June 4, 1996, a U.S. aircraft was shot down during exercises while towing a target for ship-to-air gunnery practice (Proctor, 1996). An anti-aircraft fire control system onboard a naval vessel (the Japanese destroyer *Yugury*) transitioned from manual (optical) tracking mode to automatic. In the process, the target lock shifted to the U.S. aircraft itself instead of the towed target. When the gun opened fire, the aircraft, instead of the towed target, was hit.

Functional and supervisory modes represent two different types of modes. That is, while the functional mode is tracking, the supervisory mode can be either manual (optical only), semi-manual (optical and radar), or fully automatic (radar only).

Miller (1979) studied how operators interact with anti-aircraft artillery (AAA) systems. Such systems have many modes of operations. Selection of the appropriate mode depends on external conditions (the environment), tactics employed, and limitations of the AAA system. The crew of the AAA plant decide which activities to perform and “which mode of operations is to be used at each point in time” (p. 50). The AAA system is responsible for search, target acquisition, tracking, and finally firing. The system has a variety of modes by which the operator specifies the mechanism

that will be used to track the target. The operator can select various interface modes, such as which sight magnification is used for tracking (2x, 6x). He or she can also select and request different function modes, such as whether azimuth tracking control is in “rate” or “position” control. The operator can have the system track the target in azimuth and elevation automatically, or can do this manually (optically)—defining two different supervisory modes. Because such a system has several components, each one with its own set of modes, there can be situations of “mixed” supervisory modes. For example, the computer can track elevation automatically while the operator is tracking azimuth manually. Thus the operator has many combinations of supervisory modes to choose from for controlling the system.

With respect to errors in such modal systems, Miller mentions two types: Procedural errors in which the operator fails or is unable to perform a task, and mode errors. The latter involve situations in which a subsystem state changes while it is inactive, and the new state is not immediately detected as it becomes active. As a consequence, operators have to detect the problem and diagnose the source before they are able to operate the system appropriately. “If the system causing the problem happens to be one whose state is seldom changed, this [diagnosis] could take some time” (p. 83).

The topic of supervisory modes is widely discussed in the adaptive automation literature (Parasuraman, 1987; Rouse, 1988). Parasuraman, Bahri, Deaton, Morrison, and Barnes (1990) argue that it is useful to think of automation as a continuum with various structural levels, rather than as an all-or-nothing attribute. A similar view is also advocated by Billings (1991, p. 8) and Sheridan (1992, p. 25–28). The levels of supervisory modes determine whether the human or automation bounds the task (McDaniel, 1988). In employing such supervisory modes, some concerns regarding the implications of the transitions from one mode to another are discussed in the literature.

McDaniel provides several examples of the lethal consequences of situations in which the operator assumed that a certain subfunction was performed fully automatically, when in fact manual or semi-automatic control was engaged (p. 837). He argues for providing the operator with a means of managing supervisory modes so that it is possible to track aspects controlled by the operator and those controlled by the machine.

With respect to transition from one supervisory mode to another, several studies have been conducted. Notably, Young (1969) has argued that an operator who controls the process manually is more sensitive to malfunctions in control dynamics than when a fully automatic mode is engaged. However, contrary results were published by Ephrath and Curry (1977), who showed that detection of failures (experimentally induced deviations in the controlled process) were detected more reliably and quickly in the fully automatic mode. Evidence leaning toward Young’s conclusions was provided by other researchers (Kessel and Wickens, 1982; Wickens and Kessel, 1979). For the purposes of this work, however, this literature highlights the difficulties in managing and transitioning among supervisory modes.

The literature review from several domains alludes to three different types of modes that are employed in computers, devices, and large control systems. *Interface modes* are used for data entry and changes in display format (insert/command, outline/normal view in a word processor). *Functional modes* allow for different functions (fax, copier) and subfunctions (setup, humidify, idle). Finally, *supervisory modes* are those that specify the level of the interaction or supervision (manual, semi-automatic, and fully automatic). The three subtypes are essentially similar—they all define a manner in which a component of the machine behaves. Armed with knowledge of these three types of modes, we now review their uses in commercial aviation.

Modes in Commercial Aviation

Modes have been fully acknowledged in this domain as both a method of human-machine interaction and a source of potential problems. They are a topic of growing concern to operators, operational management, regulatory agencies, and manufacturers—partly because of several highly publicized accidents and incidents involving highly automated jet aircraft (Abbott, 1996; *Aviation Week and Space Technology*, 1995; Mellor, 1994).

In reviewing the topic of modes in pilots' interaction with Automated Flight Control Systems, one may see a step-wise progression in the study of this problem. First are the early papers establishing the link between automation and human factors issues. Next are studies documenting the problem using field observations, incident reports, and survey methodology. Most recent are the studies using experimental approaches (M. Shafto, personal communication, May 1996).

Automation as a Human Factors Issue

In the 1970s and early 1980s, several authors raised concerns regarding the human-automation interaction in various domains (e.g., aviation, process control). Based on field studies, personal experience, and incident/accident reports, these authors detailed limitations in the structure of automation. Modes (i.e., supervisory modes) were identified as a method of implementing functionality and level of involvement. Likewise, much concern was raised about interfaces to support human interaction and the subsequent impact of automation on operators' performance and motivation (Bainbridge, 1983; Chambers and Nagel, 1985; Edwards, 1976; Edwards and Lees, 1974; Shackel, 1967).

Observational Studies

On the specific topic of modes, Wiener and Curry identified the potential of mode ambiguity in the cockpit in their 1980 paper, "Flight-Deck Automation: Promises and Problems." They were concerned about cases in which the pilot is confronted with ambiguous information about the 'current' mode of a warning system (e.g., Ground Proximity Warning System—GPWS) (Wiener and Curry, 1980, guideline #12).

Curry (1985) studied the introduction of the Boeing B-767, the first of several aircraft with an Automatic Flight Control System (AFCS). His field study incorporated data from cockpit observations and a questionnaire. He reported the potential for mode problems in the relationship between modes and their reference values—that is, when mode transitions occur, the reference values (e.g., speed) change automatically, and this relationship tends to confuse flight crews (p. 15, 24, and 34–35). Another mode feature that leads to confusion is a sequence of automatic mode transitions interrupted by a manual mode change. In cases where the crew is not looking directly at the display (i.e., when they are looking outside the cockpit for other aircraft in the vicinity), this may result in an unexpected mode configuration and subsequent confusion (p. 23). During interviews, pilots voiced concern about their limited knowledge (specification) of the AFCS modes and transitions (p. 19).

Wiener (1985), using the same methodological approach as Curry, studied the introduction of the McDonnell Douglas MD-80, a derivative of the DC-9, equipped with a new AFCS. In the questionnaire responses, the pilots in his study group mentioned several other factors that lead to mode ambiguity: (1) mode-mode relationships in which a mode change in one component leads to a mode change in another (e.g., "overlapping function of the autothrottles and autopilot") (p. 30, 46); (2) difficulty in resolving the next mode configuration, given a mode or reference-value change, because of limitations of the mode display (specifically the Flight Mode Annunciation), or the

pilots' lack of knowledge specification of these relationships (cf. with Palmer's 1995 analysis of this specific problem in the MD-88); (3) not understanding, or forgetting, the defaults of both mode and reference values (p. 30–31); (4) cases in which the same button is used differently depending on the mode, without a clear indication of this relationship (e.g., Mach/Speed selector button—an interface mode problem) (p. 33); and (5) difficulty identifying transitions between modes after a mode has been requested (armed) (p. 62). In addition, pilots voiced concerns regarding the lack of standard operating procedures for using the AFCS (p. 33).

The Curry (1985) and Wiener (1985) field studies focused on the introduction of new Automatic Flight Control Systems into line operation. Several years later, Wiener published another report on pilots' interaction with automated aircraft, namely the Boeing B-757. But by then such aircraft had been in use for several years and experience, both in terms of pilot training and operational doctrine (e.g., policies, procedures), had matured. The objective was to extend the scope of the earlier field studies (Curry, 1985 and Wiener, 1985) to include the full range of crew experience (low-time and experienced) and different approaches for using the aircraft (two airlines were surveyed).

Wiener's report broadens the topic of cockpit automation and pilots' interaction with modes by adding the operational context—the Air Traffic Control (ATC) environment, crew coordination, and workload. The pilots in the survey described problems with mode-reference value interaction, in particular where the source of the value was either the computer or manual entry (p. 107–116). Problems with mode-reference value interaction are detailed with respect to both altitude and speed. These problems arise from the lack of a clear indication which reference value actually constrains mode behavior. Another source of mode confusion occurs in transitions between supervisory modes (e.g., flying with or without the aid of the Flight Director and autopilot). Also, because the AFCS of the Boeing B-757 contains several components (each with its own set of modes), it is no trivial matter for pilots to maintain a vector with all the active modes in their memory. This, however, is crucial for anticipating the next mode configuration in response to an internal or external event. Likewise, mode-mode interaction in such an integrated system is potentially confusing (p. 116). Sequencing of mode selection (p. 117) and reference-value entries (p. 107) within a narrow window of opportunity is another factor that is sharply criticized by pilots.

The report also highlights pilots' strategies for interacting with such mode-based systems. One paradoxical finding is that during high-workload periods, pilots tend to abandon the very modes and devices placed in the cockpit for the purpose of reducing workload, in favor of more manual (supervisory) modes (cf. Bainbridge, 1983). Another insight is the ways pilots work around (“trick”) the possible mode configurations and reference values in order to achieve functionality (system behavior) that is otherwise unavailable to them (p. 170–171).

The report subjectively quantifies some of these concerns about use of automation in general, and modes in particular. Asked whether they always knew what mode they are in, some 70 percent of the pilots agreed. Asked, conversely, if there were still things that surprised them in the B-757 automation, some 60 percent of the pilots agreed with this statement (p. 28). Finally, the relationship between the current ATC environment and pilots' use of cockpit automation and selection of modes was detailed. In many cases reported in the survey, ATC either did not take advantage of AFCS capabilities, or actually increased flight crew workload while the crew was manipulating the AFCS modes to comply with ATC clearances (see Chapter VII).

Incident Studies

In addition to surveys and field observations, another source of information about pilots' interaction with the AFCS is by analysis of incident reports. One source of incident data is NASA's Aviation Safety Reporting System (ASRS) database, which contains some 50,000 reports of aviation safety incidents. An initial attempt to analyze ASRS data conducted by Rogers, Logan, and Boley (1989), indicated that AFCS mode problems can lead to violation of ATC clearances, mostly with respect to airspace constraints. Contributing factors mentioned are inadvertent disengagement of fully automatic modes and wrong entry of the associated reference values (p. 21). A later study (Eldredge, Dodd, and Mangold, 1992) provided additional classifications of mode problems: (1) difficulties in transition among modes in general, and supervisory modes in particular; (2) problems in understanding the logic of the fully automatic mode configuration, in particular the "Vertical Navigation" mode; (3) lack of knowledge about the conditions for mode transition, as well as insufficient information on the mode display.

Vakil, Hansman, Midkiff, and Vaneck (1995) conducted a study to identify cases in which the AFCS fails to execute an action that is anticipated or expected by the pilots. Based on their analysis of 184 incident reports, they identify seven categories of the perceived causes of such problems: database errors, mode transitions, insufficient knowledge, AFCS failures, data entry problems, crew coordination, and unknown causes. Data entry errors were the leading factor identified. Another finding was that problems in the vertical aspect of the flight (including speed) dominate the mode transition category—indicating, perhaps, a unique relationship between the two.

Survey Studies

Several surveys were conducted to address and assess problems with pilots' use of automation and modes. One of these was conducted by Corwin to identify the contribution of display presentation to mode problems (Corwin, 1993; 1995). In most modern autoflight systems, modes are displayed via the Flight Mode Annunciation (FMA) device, a means by which crews can assess and rectify activation of inappropriate modes. Corwin asserts that although the FMA dutifully annunciates modes, its foremost flaw is that mode annunciation occurs in a different location than the mode buttons. Furthermore, he cites a lack of industry standardization in the location, color, and alphanumeric used to annunciate mode requests, engagement, and activation. Based on the results of this survey, he contends that deficiencies in mode displays, coupled with the lack of standardization of mode annunciation, could prove to be the single greatest "growth" area for accident research in automated aircraft (Corwin, 1993, p. 6).

One weakness of ASRS-based studies is that they are limited to situations in which safety was threatened (Degani, Chappell, and Hayes, 1991). In 1992 Sarter and Woods conducted a study incorporating two complementary sources of non-ASRS information: questionnaires of some 135 pilots flying the Boeing B-737-300, and simulator observations of 10 crews which were going through the ground school training for this aircraft. Their survey findings are consistent with Wiener (1989) in regard to whether or not pilots know which mode configuration they are in, and the extent to which they encounter surprises with the AFCS modes. In addition, the survey highlights several other issues: (1) automatic transitions due to boundary conditions (envelope protection) are an event that surprises flight crews; (2) pilots have problems understanding the relationship between modes and their reference values, in particular transitions between modes that use different sources for reference values. A large number of reported problems focused on the use of the fully automatic mode configuration. In particular, the fully automatic vertical mode (VNAV) has a set of submodes that are automatically engaged depending on the situation, resulting in

sometimes unexpected AFCS behaviors. These automatic transitions have led to considerable confusion on part of the flight crews.

Observations of flight crews during training highlighted more issues. One is that the available mode configuration space dynamically changes as a function of the phase of flight and internal/external conditions. When these dynamic mode changes are combined with the lack of an accurate and complete model of the AFCS, the result is mode ambiguity. To manage such complexity, it appears that pilots use only a small repertoire of strategies to interact with the AFCS modes (p. 318-319). This important observation has a corollary in the human-computer interaction domain, where Rosson (1983) found that even experienced users only used a small subset of the possible commands in the system.

In general, the above studies documented some of the problems associated with pilots' interaction with the AFCS through field studies, observations of flight training, questionnaires, and incident reports. Environmental situations and the possible mode configurations of the aircraft that contributed to mode confusion and error were investigated. Based on this information, researchers were able to identify some of the factors that led to such problems. While these studies provide a porthole view into the problem of mode ambiguity, they are limited in the amount of experimental control that researchers can bring to the topic. The next section describes a set of experimental studies in which such control was an important objective.

Experimental Studies

As a follow-on to their survey and observation study, Sarter and Woods conducted an experimental study in a Boeing B-737 part task simulator. They set up an experimental scenario involving specific tasks and events that appear to be problematic in using modes. These tasks and events were triggered during the simulated flight to probe pilots' knowledge of the system, proficiency in conducting tasks, and ability to track and anticipate mode configurations. The study's findings indicated that with respect to the specific tasks in the experiment, many pilots had difficulty recalling the set of conditions that would allow them to engage and disengage a mode. Likewise, most pilots did not know under what conditions a certain mode (e.g., Go-Around) would be enabled by the system. Not all pilots knew the logic and conditions for automatic submode activation of the fully automatic mode. Finally, because the study group comprised both pilots who had past experience with the AFCS (14) and those with no AFCS experience (6), the research allowed for comparison between mode engagement strategies. Pilots with experience tended to select modes depending on operational constraints as well as the likelihood of future constraints. Pilots who had just completed their training (who had no line experience) were much less sensitive to these factors.

Sarter and Woods (1995a) argue that the underlying reason for these deficiencies is the lack of what they call "mode awareness." The term refers to the inability of the supervisor (e.g., pilot, user) to track and anticipate the behavior of the automated systems (p. 7). Two main factors lead to lack of mode awareness: First, users have inadequate or flawed knowledge ("mental models") of the system operation. Second, so called "opaque" interfaces limit the supervisory task of tracking the state and behavior of the system.

Another complicating factor that makes it difficult to maintain mode awareness is that modern Automatic Flight Control Systems allow simultaneous use by multiple pilots (most modern flight decks have a crew of two). Therefore, tracking system status and behavior becomes more difficult when another pilot can alter the AFCS mode configuration and reference values (p. 17). As for the monitoring requirement, pilots not only have to integrate raw data (airspeed, altitude, location in

space) from the displays, but they must also monitor the behavior of the control system. “In other words, the automation (i.e., AFCS) is becoming more of a dynamic process by itself” (Sarter and Woods, 1993, p. 55). Monitoring is additionally hampered because increased capability (pre-programming a fully automatic mode) creates increased time delay between users’ input and system feedback (Sarter and Woods, 1995a, p. 7).

Woods *et al.* (1993) discuss the contribution of “mental models” to understanding mode problems. They argue that mode error tends to occur for two reasons: “Either the user mis-assesses the mode configuration of the system at a given point in time, or misses transitions (and the implications of such transitions) in mode status over time” (p. 118). Lack of knowledge, the authors remark, is another factor that contributes to mode error. This problem comes in two forms: (1) a gap in the operator’s “mental model” of the system, and (2) difficulty in learning how to coordinate and transition among different modes and submodes. These problems usually appear when the system has reached situations that are non-standard and “go beyond textbook cases.”

Finally, Sarter and Woods set out to investigate whether newer and more automated AFCS systems create the potential for other types of errors on the part of flight crews. They focused their efforts on pilots’ interaction with the AFCS of the Airbus A-320—a highly automated system. Similar to the methodology they applied in the preceding studies, they conducted first a survey and then a simulator experiment. Based on the results of the survey, which indicated the type of design features that lead to mode ambiguity and confusion, they designed a full-mission simulator scenario incorporating some of these features as probes.

The survey identified additional design features that may lead to mode confusion: (1) moded phases of flight that require pilots to indicate specifically to the AFCS that they wish to activate a phase (e.g., approach), (2) mode-mode interaction that may lead to unexpected behaviors in relation to the active supervisory mode, and (3) mode-reference value interaction within the Flight Management Computer.

Findings from the full-mission simulator experiment revealed the problems that some pilots have with standard tasks they were asked to perform (p. 47-50). With respect to non-standard tasks, pilots had difficulties conducting tasks in which the AFCS *failed* to make an expected transition because of a condition that is not transparent (p. 52–54). Conversely, pilots had difficulties with tasks in which the AFCS took an action that was unexpected by the crew (p. 51–52).

Since the subject group was split in half with respect to experience in the A-320, the added contribution of the experience factor was also investigated by Sarter and Woods. They did not find any significant differences between the two subgroups in making mode-related errors. “The only trend seems to be that the more experienced pilots are more likely to detect and recover from error” (p. 62). In addition, they discuss the issue of attention allocation in such complex automated systems. They argue that having a mental model of the AFCS is a prerequisite for efficient interaction. A mental model of the automation helps pilots form expectations of the system status and behavior, and also guides their allocation of attention (p. 66).

Sarter and Woods conclude by observing that the newer generation of advanced automation aircraft has not reduced or eliminated human-automation problems. They suggest that these newer aircraft may have introduced an additional type of error. Not only do errors of commission continue to exist, but because of the high autonomy and authority on the part of the AFCS, errors of omission also occur. These errors are associated with the pilots’ failure to detect and intervene with *undesired* system behavior that may occur automatically. But perhaps such errors of commission and

omission are fundamentally similar, varying only in the different type of transition (manual and automatic, respectively) in a system with many interactions between modes and reference values.

Perspectives on Modes in Control Systems

Norman (1990) tries to assess the state of the art of human interaction with automated devices from a global perspective. He asserts that current systems have an intermediate level of intelligence that tends to magnify difficulties. Although the information is accessible and feedback is potentially available, it is not attended to properly. “The task of presenting feedback in an appropriate way is not easy to do. Indeed, we do not yet know how to do it” (p. 9).

He argues that in order to control automation, the user must be provided with an appropriate kind of feedback which requires a higher level of interface sophistication than currently exists (p. 10). Norman identifies two reasons that current systems have poor feedback: One is that the designers are just not sensitive enough to the user’s task, but there is also a perfectly natural reason—“the automation itself doesn’t need it!”

Billings (1996) remarks that aviation automation has provided great social and technological benefits, but these benefits have not come without cost. New types of problems in aircraft have emerged due to failures in the human-machine relationship. He argues that the evidence suggests that the principal problems with today’s aviation automation systems are associated with their complexity, coupling, autonomy, and opacity. These problems are not unique to aviation, argues Billings; they exist in other highly dynamic systems as well. He suggests an approach called “human-centered automation” that focuses on the cooperative relationship in the control and management of aircraft and Air Traffic Control centers.

One important but somewhat less conspicuous aspect of human interaction with complex systems is certification. Because governments regulate the design of aircraft, certification agencies must grapple with the question of technology impact on flight safety (Greene and Richmond, 1993). In that role, certification practitioners must have foresight in predicting the implications of new systems such as the AFCS on flight safety. A certification test pilot expresses this thought succinctly (Newman, 1996, p. 4.3):

The practitioners of the most exact science in aviation, hindsight, frequently provide us with the answer of how to prevent the last accident.... Sometimes those of us who are practitioners of the least exact science, foresight, would dearly like to have the advice of the “hindsighters.”

In the past, engineers and test pilots reviewed a system at the design and certification stages, then tried to envision and analyze all possible conditions and situations that could happen in service. Systems were designed to mitigate any problem areas. “This cannot be done anymore, however, because the systems are too complex” (Scott, 1995, p. 64). Furthermore, with regard to modes and their interfaces, certification officials have no rigorous methodological approach (beyond their subjective evaluations) to assess designs in terms of human-machine interaction. In particular, it is difficult for certification officials to determine whether a design feature is unacceptably bad, or merely not yet optimized. What these test pilots and engineers cry for is a methodology for drawing the metaphorical “line in the sand”; a measure which is absolute, not comparative (Newman, 1996, p. 43). Concerned with the issue of pilots’ interaction with flight-deck automation and their modes, the U.S. Federal Aviation Administration (FAA) launched a study to evaluate this topic. With respect to mode usage, the report lists several general issues (Abbott, 1996, p. 34–38).

- (1) Pilots find it difficult to understand the control algorithms of many modes (i.e., how modes behave).
- (2) Pilots sometimes have a wrong and incomplete model of the system.
- (3) Situations that were unforeseen by the designer lead to mode behaviors that are unexpected.
- (4) Pilots find it difficult to anticipate the next state and behavior of the AFCS when a combination of supervisory modes is used (autopilot “Off” and autothrottle “On”).

As for the factors that prompt mode ambiguity, the report lists several (p. 43–59): One is interface limitations. These may be the result of attention-related characteristics (i.e., performance specifications). Mode symbology is small, cryptic, and transient. Mode ambiguity can also occur because the interface itself is insufficient. That is, the same mode annunciation may yield different mode behaviors during different situations. Second, there is lack of standardization between AFCS systems of different manufacture. The concern is that when flight crews move from one aircraft type to another, AFCS inconsistencies may hinder their use and understanding of the system’s modes. Inconsistencies can be found in mode display formats, symbology, and presentation of mode transitions, as well as in mode behavior. A concern related to lack of AFCS standardization is that the scan patterns for modes are very different than those to which pilots are accustomed for other system indications, thus leading to deficiencies in sampling this information from the interface. Third, too many modes with complex interaction limit the pilots’ ability to have an accurate model of the system—leading to the development of the so called “folk models.” Fourth, there is conflicting information about mode status between the Mode Control Panel and the Flight Mode Annunciation display. Fifth, there are difficulties in the ergonomic design of the interface—the same button is used to request different modes. Finally, there is a lack of feedback via auditory, kinesthetic, and vestibular channels about the behavior of the AFCS and aircraft. In other words, there is overuse of the visual channel.

The report concludes with several recommendations. One of the most important is that the FAA should require systematic evaluation of flight-deck designs to identify design-induced flight crew errors as part of the certification process for new and modified aircraft (p. 95–97).

CHAPTER SUMMARY

The review of the literature provided us with considerable information about the uses of modes in several domains and the type of problems that were encountered.

We have identified three general types of modes that are used: *interface*, *functional*, and *supervisory*. These modes are not mutually exclusive. Depending on the point of view, a certain mode behavior may be classified as two types (e.g., a fully automatic mode is both functional and supervisory). Nevertheless, with the exception of extreme cases, this scheme removes some of the definition inconsistencies in the literature. We find modes (interface modes) that specify the data entry and display format of the interface in the human-computer interaction literature. Modes, however, are used not only for data entry and display format purpose, but also to contain and specify a given function of the machine (function modes). In the supervisory control domain we have identified another type of mode—one that specifies the level of involvement of the user (supervisory modes). Finally, a complex human-machine system (such as aircraft, and military fire and control systems) will have all three types of modes: interface, functional, and supervisory.

In our view, however, these three different type of mode are all similar—they specify the structure and manner of behavior of the system. Interface modes specify the behavior of the display and data entry—the interface either interprets a key stroke as a command or a character key, for example. Functional modes specify the behavior of the control mechanism or plant—for instance, the machine may be used as a compact disc or tape player. Finally, supervisory modes specify the behavior with respect to human-machine involvement. Modes, therefore, are not just an attribute (e.g., interface) of the device. Instead they also specify the hierarchical and functional structure of the machine.

After understanding what modes are, we consider the process of human interaction with them. The user interacts with a moded automated control system through a series of steps. The first step is the *selection* of the most appropriate mode (Callantine, 1996). The second step is the *request* of a mode transition, which involves a physical interaction with the interface. We use the word “request” here because the system may, or may not, honor the request, or it may wait for some conditions to become true before the request is fulfilled. Once these conditions are honored, the system is *engaged*. In most cases, the mode becomes *active* by default, but there are cases in which the user must enter some reference values in order to activate the mode. Finally, there is *disengagement*. Technically speaking, however, the term is incomplete because disengagement always implies engagement of another mode.

The term *mode ambiguity*, as used in this thesis, refers to the inability of the user to anticipate the next mode and reference-value configuration of the machine. Mode ambiguity may lead to *mode confusion*—a discrepancy between user expectations and the actual mode behavior. The existence of mode ambiguity, in itself, does not necessarily translate to mode confusion: If the user has no expectations, no mode confusion will take place. Likewise, if the user formed the accurate expectation, say by chance, no mode confusion will occur even though mode ambiguity does exist.

Now, we can ask what the factors are that independently produce mode ambiguity. Unfortunately, in many of the studies reviewed there is an overlap of many factors. It appears, however, that this list of factors includes the interface limitations, some aspects of the machine’s complexity, the ability of the user to perceive information, and the task at hand and the knowledge of the user about it. Providing insight into this enigma is an important aspect of this research.

Finally, an important observation from our review of the literature is the lack of methods for representing human interaction with modes. The inability to systematically describe mode behaviors limits our understanding of them. Likewise, a language is necessary for describing the interaction between the user and the interface, and the resulting behavior of the machine. A representation is also a prerequisite if one tries to understand the factors that lead to mode ambiguity. In other words, one cannot systematically assess the efficiency of the roads in the city before one has a map of the city. In the next chapter, therefore, we review the literature on models of human-machine interaction and attempt to identify a modeling approach that will allow us to describe mode behavior and human interaction with machines that employ modes.

MODELS OF HUMAN-MACHINE SYSTEMS

There are a variety of human-machine systems and human-computer interaction models in the literature. The models presented in this chapter were selected on the basis of two key criteria: (1) the model is potentially useful for describing a mode-based system and identifying problems with mode information and mode ambiguity; (2) the model has actually been used in the context of human interaction with systems that employ modes.

Models of human-machine and computer interaction can be categorized along several dimensions (Jones and Mitchell, 1987). First, they can be categorized according to their purpose: A model can be descriptive (i.e., describe human cognitive and manual behavior) or normative (i.e., describe what an operator *should* do). Second, they can be categorized according to their structure: The model can be computational, mathematical, or conceptual. Finally, they can be categorized by what they specify: A model can specify the device or the task.

Several models are described in the sections that follow. We start at one extreme with the Abstraction Hierarchy, a conceptual model, and then move to the other extreme with PROCRU—a computational and simulation-based model. GOMS and cognitive complexity theory are normative and computational approaches that provide a representation of the users' tasks in motor, perceptual, and cognitive terms. Finite State Machine theory and its graphical notation (state transition diagrams) provide a more refined system description in terms of the system's finite set of mode and state configurations. Petri-Nets are extensions to the Finite State Machine that add the concept of concurrency and a graphical notation. The Discrete Control Network and Operator Function Model (OFM) are also modern extensions of Finite State Machine models, providing both concurrency and hierarchy.

ABSTRACTION HIERARCHY MODEL

The first model described, the abstraction hierarchy, is a conceptual model developed for the purpose of designing and troubleshooting process control systems. It describes the functionality of a given system in a multi-level representation that can be depicted as a pyramid (Rasmussen, 1985; 1986, chap. 4). At the pyramid's top, the system's functional purpose—its objective and process—are listed. Descending from the top of the pyramid, each level becomes more concrete in specifying the process and types of mechanism used for control. The lowest level of the pyramid describes the actual physical component (e.g., the aircraft's elevator). Links between the pyramid levels are provided via a “means-end” relationship (Newell and Simon, 1972, chap. 8), the means (actions and mechanisms) that are necessary to get to the end (goal).

In the same manner that we can descend along the pyramid, we can also climb and see how the components described at the lowest level are represented according to their physical function (e.g., control pitch), and relationship to other physical components. Successively higher levels are represented in terms of their generalized function (e.g., guidance) and abstract function (e.g., flight management and aerodynamics).

Thus the abstraction hierarchy can be used to observe the system from two different viewpoints: In a top-to-bottom view, we identify the breakdown of the system from functional purpose to physical components. In a bottom-to-top view, we identify how the components are contributing to the function and purpose of the system. In addition, the model decomposes the system along two dimensions: the abstract-concrete and the whole-part. The first dimension relates to the concepts used for description, and the second relates to the level of detail used for description. Both are helpful in describing the relationship between components and goals in the system (Bisantz and Vicente, 1994).

Applicability to Mode-Based Systems

The abstraction hierarchy contains no event-driven or task representations. This is both a weakness and a strength of the model. By not specifying events, the model cannot represent the behavior of the system and it has no dynamic property. But because it contains no event-driven representations, the model does not constrain the operators or provide a single path for controlling the system.

The abstraction hierarchy can be used for describing a mode-based system. It can show how the goal of the human-machine system is translated to the component, mode, and physical unit levels. It can also show the relationship between the physical principles governing the system and the various modes that manage them. In this way, one can identify redundant modes, as well as the modes to use during a failure. Finally, because of its conceptual structure, the model may be able to present some of the interactions between modes and components in a supervisory control system.

Summary

The abstraction hierarchy is a potentially useful model for understanding the relationships between the system's functional purpose and the modes of the system. However, the model is quite limited in identifying the behavioral aspects of the system that cause mode transitions and ambiguity.

PROCRU MODEL

A model that does focus on both the continuous and discrete dynamics of the system is PROCURU (Procedure Oriented Crew Model). PROCURU is a normative and computational model of operators that perform both continuous and discrete control and monitoring tasks. Designed to simulate operators' (with emphasis on the plural) interaction with a supervisory control task, the model may be used for evaluating procedures, tasks, and displays (Baron, Muralidharan, Lancraft, and Zacharias, 1980).

PROCRU specifies the task in terms of procedures, as well as measuring the performance of the simulated crew (Baron, 1984). The model was initially implemented for describing flight crew tasks during simulated landings. A similar model was implemented for simulating operators' interaction with an anti-aircraft artillery unit—a system with many modes (cf. Miller's work on a similar system described later on).

The PROCURU model can account for vehicle dynamics, environmental disturbances, and crew information processing activities such as decision making, control, and communication. Information about the vehicle's (continuous) states and component (discrete) states is provided to the crew via displays. In addition, external information from the operational environment is also provided. With respect to information processing, PROCURU utilizes the approach used in the Optimal Control Model—a model of total system performance and operator response in continuous manual control tasks (Baron and Levinson, 1980). Added to the information processor is the discrete event detector,

which complements the continuous state estimator. Based on these two types of information, the simulated crew executes procedures and tasks, selecting those which are expected to maximize gain.

Applicability to Mode-Based Systems

The PROCURU model can be enhanced to describe mode-based systems. The anti-aircraft artillery unit mentioned earlier is one example of such an application. To deal with mode-based systems, a situation assessment module was added to the information processor. This module was then used for several decisions: the appropriateness of the current operating mode, the gains to be achieved by switching modes, firing strategies, and communication of mode selection to other crew members.

The model output depends heavily on the procedures and operator tasks redefined by the modeler. Although procedural information is modeled essentially error-free, other types of errors do occur. Errors in execution of procedures and tasks may occur because of improper decisions that result from lack of information (quantity and quality) due to perceptual, procedural, and workload situations (Baron, 1984, p. 21). The model therefore can identify problems in the displays, the specification of the task, and scheduling of tasks. PROCURU also generates timelines of crew activities and system response that depend on the situation (disturbances, previous actions); these results may vary. In addition, manipulation of environmental factors, and human-system parameters will yield different results, allowing the modeler to evaluate the human-machine system in a variety of situations.

Summary

A PROCURU model can be useful for designing new moded displays and assessing related procedures. It may be also useful in identifying the contribution of information quality and quantity to the operator's process of selecting and activating system modes. However, PROCURU has to be modified extensively to account for the dynamics of mode transitions. Finally, there is an empirical question as to whether the model can be used for understanding the underlying difficulties users have with mode-based systems, such as mode ambiguity.

GOMS MODEL

While PROCURU is based on theoretical control modeling concepts, GOMS is a model based on psychology and cognitive science concepts. The model was developed by Card, Moran, and Newell (1983, chap. 5) and stems from the early work of Newell and Simon (1972). Its purpose is to provide a modeling framework for analyzing human interaction with computers. The model is normative and computational and provides a representation of the users' motor, perceptual, and cognitive tasks.

GOMS stands for Goals, Operators, Methods, and Selection of rules, which is descriptive of the way the model works. Usually working in a text editor, the user starts with some *Goal* in mind (e.g., moving a piece of text from one place and inserting it in another). *Operators* are the converse of goals: they are the elementary acts which the user performs. *Methods* are the established means of achieving a Goal in terms of subgoals or elementary Operators. When there is more than one possible Method for the achievement of a certain Goal, a *Selection* rule is used to choose between the competing Methods. The Selection rule control structure is a production system that is used to represent the procedural knowledge of the user (Kieras, 1988).

The acts of the user in performing the Methods in order to achieve the Goals are assumed to be sequential. Like any model, the level of description depends upon the purpose of the analysis. The

model supports both fine actions (e.g., hitting a key—Key Stroke Level model) and more coarse actions (e.g., cut-and-paste).

Developing a GOMS model involves a detailed analysis of the user's tasks (Attwood, Gray, and Jones, 1995). The task analysis (and GOMS) must faithfully represent the “how to do it” knowledge that is required to achieve the intended goals. Once an accurate and detailed task analysis is conducted and represented in a GOMS framework, the model can be successfully used to predict human-computer interaction tasks. The reasonableness of these predictions suggest that there may be some degree of correspondence between the GOMS structure and human cognition (Simon and Mayes, 1991). However, as Simon and Mayes warn, this correspondence may exist only “for text editing or relatively similar tasks” (p. 151).

GOMS models have been applied in a variety of domains. One example is the application of the model for analyzing the interaction between telephone operators and their new workstations (Gray, John, Stuart, Lawrence, and Attwood, 1990). Using the model, Gray *et al.* (1990) identified some of the limitations in this interaction and predicted operator work time. They demonstrated that the GOMS model can be used for developing procedures and predicting their overall effect on human-machine interaction.

Another example of GOMS model application is for analyzing pilot interaction with the Flight Management Computer of an Automated Flight Control System of a Boeing B-737-300. Three hierarchical levels of methods are used in the task of managing the airliner's flight via a computer. The top level models the structure of the environment, the tasks, and the selection of methods appropriate for the task. The second level includes all the methods and their associated goals, which are dependent on the corresponding Flight Management Computer functions (e.g., enter the route of flight, modify route, etc.). Finally, the lowest level describes how each one of the methods is actually executed by means of motor and cognitive operations (Irving, Polson, and Irving, 1994; Polson, Irving, and Irving, 1994).

The model highlighted some important characteristics of the interaction between the pilot and the Flight Management Computer. One is that almost all tasks are carried out by a set of three intermediate-level methods (access, designate, and insert). Another characteristic, however, is that these methods are heterogeneous, complex, make significant demands of memory, and therefore result in lengthy training times. Another finding is the relationship between ATC clearance formats and Flight Management Computer formats. The lack of direct “mapping” between these formats requires much compensatory effort on part of the pilot. While these findings are not necessarily new, they are significant because they were quantified using a formal method.

Applicability to Mode-Based Systems

It appears that GOMS can provide insight about the process of human interaction with mode-based systems—specifically, mode selection, request, and insertion of reference values. Such analysis may reveal the common characteristic of the task and possible inconsistencies in the task (Polson and Degani, 1995).

GOMS, as mentioned earlier, is useful for analyzing tasks that are tightly constrained by the interface. These tasks have a well-defined structure (e.g., initialization of the Flight Management Computer). It appears, however, that tasks which are conducted in parallel (concurrent tasks) are quite problematic to model. In fact, the authors who developed the GOMS explicitly rule out modeling parallelism in actions (Card, et al., 1983). The main limitation for modeling mode-based systems is that GOMS does not explicitly describe the behavior of the device. It does so only

implicitly by describing user actions. And indeed, in most office automation applications this suffices. However, in the supervisory control domain where systems are reacting to both internal and external stimuli, transitioning automatically between mode configurations and thereby changing their behavior, a GOMS representation may miss some of the features that lead to mode ambiguity and confusion.

Summary

There are several limitations to the GOMS methodology. First, GOMS can predict only the procedural aspect of the interaction—the amount, consistency, and efficiency of the procedures the user must follow. Since in most human-computer interaction tasks the interface tightly constrains the user’s methods of interaction, a GOMS model can provide reasonable predictions in such tasks. Second, GOMS cannot efficiently model low-level perceptual aspects of the interface, such as type faces on the display, or the user’s conceptual model of the device (Kieras, Wood, Abotol, and Hornof, 1995). Third, GOMS assumes an error-free execution of methods and operators. With these limitations in mind, GOMS models can nonetheless be quite useful for describing the interaction of humans with automation, in particular data entry aspects. However, it appears that with respect to identifying mode ambiguity problems, the framework is somewhat limited.

FINITE STATE MACHINE (FSM) MODEL

In executing a GOMS model, the process of execution travels through a path of Goals, Methods, and Selection rules. The possible paths were predetermined earlier. A similar approach is the application of Finite State Machine models to represent human tasks and procedures in the context of a device such as a computer. The Finite State Machine models and their numerous extensions have their roots in Turing’s computing machines and neural networks (McCulloch and Pitts, 1943; Turing, 1936). By definition, such models are normative and computational.

A Finite State Machine model is a way of describing a system (e.g., a machine) with finite, well-defined, possible configurations—the machine can be in any one of a finite number of internal configurations, or “states.” The model has only finite input and output sets—it can respond only to the specified set of stimuli and produce only the specified set of behaviors (Turing, 1936, p. 232-235). The general structure of such a machine is described by state transitions in the form: when event ‘alpha’ occurs in state A, the system transfers to state B.

The control mechanism of a building elevator is a good example of a finite state control structure. The mechanism does not remember all previous requests for service, but only the current floor, the direction of motion (up or down), and the collection of not-yet-satisfied requests for service (Hopcroft and Ullman, 1979, p.13). The current configuration (the current floor, direction of motion) of the system (elevator controller) summarizes the information concerning past inputs that is needed to determine the configuration of the system (next floor to stop at) on subsequent inputs (requests).

A machine that is modeled using the finite state machine theory cannot do anything beyond what was specified by its designer. The Finite State Machine (FSM) specification does not allow for any evaluation of alternatives and self-control—it merely follows a set of rules and changes its (finite) configuration accordingly. The FSM theory has been applied mostly in Computer Science, but it has also been used to model animal behavior as well as human tasks and procedures (McFarland and Bossler, 1993). The latter is the focus of this section.

Applicability to Mode-Based Systems

The Finite State Machine theory captures, in a very precise way, the behavioral aspect of a system. The corresponding graphical notation—the state transition diagram—presents this behavioral information (states and transition) in a clear perceptual code of nodes and arcs. The combination of the theoretical and graphical format is quite appealing for representing human interaction with computer-based systems. Parnas (1969) used Finite State Machine models to describe user interaction with a computer terminal. Using this formalism, he was able to illustrate several design errors such as “almost-alike” states, inconstant ways to reach a state, and data entry problems. He concluded by arguing that Finite State Machine models and their corresponding state transition diagrams can be used for design specification, coordination among design teams, and documentation; and for formal algorithmic checks for the above design errors. Foley and Wallace (1974) also used this notation to describe their concept of a language of interaction between the human and the computer. Their state transition diagram describes the actions necessary to make the transition from one state to another, as well as the system response during the transition.

Jacob (1983) described the use of several variants of the Finite State Machine, as well as other formalisms such as the Backus-Naur Form (BNF) (Woods, 1970) for the purpose of designing specifications for human-computer interaction with a communication system. He showed how such formalisms can be applied to a system with many commands. In a later study he proposed a modified version of the state transition diagram for specifying direct manipulation of user interfaces (Jacob, 1986). He observed that direct manipulation interfaces have a concurrent structure and a highly *moded* dialogue with the user—every object on the screen has a different range of acceptable inputs with very different interpretation of these user inputs (p. 287). To deal with a structure of multiple display objects (e.g., windows) that are suspended and resumed (typically when the user moves the mouse in and out of the window), he specifies each object as a single state diagram. The objects are then combined to form the user interfaces as a set of multiple diagrams governed by a high-level executive, instead of a large and cumbersome state transition diagram. He uses the augmented recursive transition network (a variant of Finite State Machine models) to deal with such a nested hierarchy (p. 262). In this approach, a transition in one state diagram can call up another (nested) diagram—a feature that can introduce recursion (Woods, 1970). Others researchers, such as Wasserman (1985), also successfully use augmented recursive transition networks to describe different types of human-computer interface.

Kieras and Polson (1985) used a similar formalism to describe the relationship between the user’s tasks and the device. Their goal was to quantify the complexity in the interaction between the user and the computer application. To do this, they represented the user’s tasks as well as the device using the same formalism. The user’s tasks were initially subject to a GOMS analysis, and were then translated into graph notation. Similarly, the device behavior, initially described in a variant of the augmented recursive transition network, was also translated into graph notation. Now, both the user’s tasks and the device were represented in the same formalism (an .AND.-.OR. graph notation which allows for limited concurrency and hierarchy). This allowed Kieras and Polson to identify a case in which the user’s task specification structure did not correspond with the device’s structure. Two solutions were investigated. One was redesign of the device. The other was redefinition of the user’s task specification (i.e., by specifically explaining the general rule for the task in the user manual). Kieras and Polson also developed a computer simulation of the model to further investigate and quantify factors contributing to complexity of the interaction between the user and the device (p. 390–392). They reported two advantages of such simulations: Building and running a simulation allowed them to identify errors in their own specifications of the user’s task. Also,

complexity issues can be investigated systematically without having the actual device (or a prototype) available.

Bosser and Melchoir (1990) employ a similar approach of representing both the user's task and the device behavior in their analysis of human-computer interaction. They consider the tasks, the user knowledge, and the device functionality as three separate entities. A framework called SANE (for Skill Acquisition NEtwork) represents the tasks as goal states, the user knowledge specification as states and procedures, and the visible device functionality as a state diagram (p. 590). The source of tasks is a detailed task analysis, with emphasis on the device-independent goal of the user. There is a device-independent start state and an end state for accomplishment of a task (McFarland and Bosser, 1993, chap. 4). User knowledge specification consists of the detailed procedure to achieve the goals. Procedures are syntactic in nature, while goals are more semantic—they are separate types of representation. Bosser and Melchoir argue that approaches like Jacob's (1983) augmented recursive transition network are limited, because they contain only the syntactic information (i.e., what sequence of events causes transitions), and not the semantic information of the user's task. Formalisms like GOMS that have both syntactic and semantic representations are also limited because there is no explicit separation of procedures and goals. Finally, the device in the SANE framework is represented using any variant of Finite State Machines. The source for this description is the engineering design specifications.

The SANE framework combines the procedural knowledge and the model of the device to form the representation of the system's functions, while the task, or goal-related information, is represented separately. Using this framework and simulation tool, a precise representation of the user's procedural knowledge is generated, comprising command states, variables, operation, and procedures which the user must know in order to perform the task. The SANE framework and its computational/simulation tool kit allow for evaluation of two aspects of the user-device interaction. The first aspect is a determination of whether the general device-independent tasks can be performed with the device. This is tested by means of a reachability graph that proves the task's end-state can be indeed reached. Then, given the information needed to operate the device, a variety of evaluation indices—such as time to perform function, memory load, and complexity of commands—are generated (Bosser and Melchoir, 1990, p.4).

Summary

While the Finite State Machine theory has been successfully used to describe the behavioral aspects of the human-machine interaction, there are several limitations to this approach: The simple transition diagram cannot represent an hierarchical structure in an efficient (e.g., modular) and clear way. Trying to overcome this shortcoming by a brute force approach leads to an exponential explosion of state configurations. Hierarchy, it appears, is a common characteristic in man-made systems and is the way we, as humans, understand complex systems (cf. GOMS; Simon, 1969/1981, chap. 7; Bainbridge, 1993). One manifestation of this is the hierarchical structure of modes in most control systems. Although solutions such as the augmented recursive transition network which allow nested subroutines—a form of hierarchy—have been introduced, they still cannot describe global events mainly because they are processes-driven and not event-driven (Harel, 1987, p. 267-268).

Another characteristic of control systems is concurrency. That is, processes are going on in parallel within various components that make up the entire system. Such representation is very hard to describe in traditional Finite State Machine models. Some solutions to this difficulty in representing concurrent processes, such as the use of a higher-level executive, have been attempted (Jacob,

1982). However, such approaches do not capture this concurrency in a clear way, and are limited in describing synchronization between concurrent processes. Concurrency and synchronization, nevertheless, are common features of modern control systems with modes.

PETRI-NETS

Finite State Machine models and their various variants still have considerable limitations in describing concurrent processes. In fact, this limitation was a critical issue in electrical engineering and computer science in the 1950s and 1960s. Petri-Nets, a formalism based on Finite State Machines, was developed in the early 1960s by Carl Petri in Germany (Peterson, 1977). The main enhancement of Petri-Nets over the Finite State Machine theory is their ability to model concurrency, as well as the interaction between concurrent processes.

Petri-Nets are used to model a system. For the purpose of modeling human-machine interaction, a system may contain the task, user, and machine (or computer). In such an arrangement, the model is normative—only what is explicitly specified can occur. Petri-Nets have a formal mathematical definition which can be translated to a graph structure. The complete structure describes the system as a whole.

A Petri-Net is composed of two types of nodes—places and transitions—that are connected by a one-directional arc. Transitions occur between input places and output places when conditions are met. Tokens are used in the Petri-Net representation to signal the existence of a condition and the current state of the net. The behavioral aspects of the model are denoted by the existence and dynamic assignment of tokens to places; that is, the location (called *marking*) of tokens changes during execution of a Petri-Net. A transition is enabled only when each of the input places contains tokens. When such a transition fires, all enabling tokens are removed from the input places and tokens are deposited in each of the output places.

Several modifications to the original Petri-Nets were developed to deal with some of their shortcomings. Colored Petri-Nets utilize colored tokens to indicate dedicated conditions. While this approach reduces the complexity of the model without sacrificing information (Jensen, 1992), it does require special handling of these colored tokens. Other modifications include adding timing constraints and objects. Colored Petri-Nets have been used to model complex systems such as telephone networks, electronic fund transfers, and semiconductor designs.

Applicability to Mode-Based Systems

In the area of human-machine systems, colored nets have been used for modeling the command-post of a complex radar surveillance system. The responsibility of the command post is the classical supervisory control task: Decision-making based on an assessment of the rapidly changing status of the surveillance network, defensive weapons, and air traffic information. To do this, the individual operator must interact with displays, communicate with other crew members, and get and send information to other control entities. The goal of the modeling effort was to develop an executable model of the command post, simulate it, and identify its limitations in order to improve its effectiveness (Jensen, 1992, p. 213).

Petri-Nets have also been used for safety analyses. In particular, where potential faults are identified and modeled beforehand, the human-machine system can be evaluated for its response to these faults. These responses can take three basic forms: fault-tolerant (system still provides full functionality); fault-soft (system provides only degraded functionality); and fail-safe (system attempts to limit amount of damage). Faults can be the result of human failure (to adhere to a

warning indication, for example), as well as the result of software and hardware failures. Various analysis techniques are provided for the detection of system responses to such failures (Leveson and Stolzy, 1987)

But not just human faults can be modeled—one can also model the human tasks in parallel with the machine's (or computer's) behavior. Palanque and Bastide (1996) present such an approach for modeling a light switch with an internal timer. They build a model of the task (a procedure) and a model of the light switch, then combine the two models to form one large model of the interaction. Using this combined model, they perform formal analysis of such factors as the absence of deadlock. Other analyses, such as the number of steps necessary to reach a given goal and time, can also provide quantitative information about the efficiency of the human-machine system.

Using a combined model (or just a task model), one can also formally describe the various paths the user can take, or has actually taken. Several researchers advocate such an approach for accident analysis (Johnson, McCarthy, and Wright, 1995). They use Petri-Nets to describe the sequence of events and decision making that contributed to a recent airline accident (Department of Transport, 1990). They argue that by using a formal method such as Petri-Nets to describe human interaction with complex systems, one can realize several advantages: The formalism can represent sequences and concurrent events that unfold in time, present the interaction between events, and show alternative courses of events. Most important of all, such analysis is systematic.

Summary

There are some concerns, however, that Petri-Nets suffer some of the limitations of Finite State Machine models. One aspect which is not treated prominently in Net theory is the structural properties of systems composed from many interacting parts (Milner, 1989). In particular, given a complex system with many components and interactions among them, there is an explosion in the number of places and transitions, to the point they become unmanageable. The root of this problem is that Petri-Nets are flat, i.e., they do not have an hierarchical structure. As a consequence, global events that are applicable to many lower-level places are not naturally represented. With respect to the notion of concurrency, the lack of hierarchy allows for only one level of concurrency in describing the system. This limitation is only amplified when we wish to combine a human task model with the machine model and represent this interaction in a clear and concise way (cf. Palanque and Bastide, 1996, Figure 11).

Petri-Nets can be used for modeling human interaction with mode-based systems. By combining a task model with a machine model, one can identify conditions for mode transitions, how this information is presented to the user, and the corresponding procedures and tasks on the part of the user. Furthermore, interaction in the machine model can be described, as well as concurrency. However, it is still an open question whether the architecture of machine, interface, and human tasks represented in a Petri-Net model does not overwhelm with its representational complexity.

DISCRETE CONTROL NETWORK

Another variant of Finite State Machine theory is the use of discrete control networks to model human behavior. Here the emphasis is on the decomposition of the human-machine system according to its hierarchical structure and the decision-making aspects of the interaction (i.e., possible actions). In terms of classification, the Discrete Control Network is a normative and computational model of the user's interaction. It represents the system and its various options solely from the point of view of the operator. It focuses on task representation and its organization.

Discrete control networks have been implemented in systems in which the operators has available only a finite set of actions and decision alternatives, and those are used to control the system. Although the operator may have to conduct other tasks (e.g., continuous ones), the goal of the modeling effort is to describe those tasks by which the system is configured, the mode of operation that is selected, and the procedures used for coordination with other operators (Miller, 1979). The model describes how control actions are determined, given (1) information about the control system, (2) the environment, or external inputs, and (3) the context of the situation. The objective is to provide a plausible and empirically testable accounting and explanation of control behavior (Mitchell and Miller, 1986; p. 345).

Such modeling has been used to simulate several human-machine systems, such as the interaction of operators with an artillery fire and control system (cf. with a similar system modeled in PROCURU), and a routing task in an industrial plant (Miller, 1985; Mitchell and Miller, 1986). Miller (1979) states that there are three steps in the process of building a Discrete Control Network. The first step is to determine all the discrete outputs which the system is required to specify. This task involves all the specific system alternatives that the operators can manipulate, such as modes, commands, and actions. The next step is to identify all the external inputs that drive the system behavior. These might include the trajectories of a target being tracked, demands that are imposed on the operators, and command information from other sources. The final step requires that the modeler specify the states and transitions in each component of the system. The Discrete Control Network shows the relationship between the decision alternatives and the possible behavior of the system.

Based on data collected during simulation with the anti-aircraft artillery system (AAA), Miller constructed a model of the system. He developed a five-layer hierarchical classification of the human-AAA system (cf. with the levels in the Abstraction Hierarchy model): The first level (primitive component) describes the low-level primitive states corresponding to the switch setting in the machine. The second level (system component) describes the physical components of the machine (e.g., computer, sight selector), as well as some pseudo components made up of primitive components that are manipulated together (e.g., gun servo enabling network, which defines how tracking information is provided to the guns). The third level (functional) is used to abstract the functions that the system must perform in order to meet the objectives (track, aim guns, fire). The fourth level (coordination/communications) describes the flow of information about how the system is gradually (sequentially) progressing toward its final objective (search, manual track, locked). The fifth and highest level is the management/command level. This level describes the high level modes of the system, which correspond to the tactics that the commander is employing in order to efficiently perform the task (i.e., supervisory modes). This level determines how the activities in the functional components of the system (Level 3) will progress once engagement sequence (Level 4) conditions are met.

Using this classification method, Miller identified some 39 critical components in the system. This, he argues, “organizes the available knowledge about the AAA system and the discrete tasks required for its operation” (p. 74). He then depicted these 39 components as nodes and added links between the nodes to portray the communication channels through which the state of the originating node is made known to the receiving node (p. 75). In other words, a transition into the next node is predicated upon events in the previous node. The resulting network is termed a “control/coordination network” and describes the hierarchical structure, information flow, and hence the general behavior of the human-machine system from the point of view of the operator. It includes the operator’s activities, the sequencing of mode selection, and the type of information

required to manage the system during the various phases of the process (p. 81). The information provided by this network is related to levels one to three in the classification scheme—states, components, and functions of the system.

In addition to this control/coordination network, Miller constructed another network that focused specifically on the strategy alternatives faced by the operator. While the previous network captured the important state and mode configurations of the system as well as the operator's procedural knowledge, what is left open to the operator are the alternatives (and their sources) not specified by the system makeup or the operating procedures. This network is called the "decision conditioning" network and describes the conditions which prompt the operator to make alternative decisions about engagement. The information provided by this network is related to levels four and five in the classification scheme—communications, coordination, and mode management.

Applicability to Mode-Based Systems

While the discrete control network provides a meaningful insight into the operation of the human-machine system in terms of actions, fulfillment of objectives, and decision alternatives and their sources, it has several limitations. First, the state of the machine and the task of the operator are entangled in the network coordination. This does not allow us to identify the relationship between operator tasks and the active states and modes of the machine. Second, to construct the network one needs to conduct tests (or simulations) and collect empirical data. Finally, although the model describes concurrency, it does not deal explicitly with issues of interaction between concurrent processes (e.g., side effects).

With respect to building such a model for describing a mode-based system, there are several clear advantages to the formalism. One is that we will be able to describe the system in terms of its state transitions. Another outcome from this effort is that the constraints on current and future mode configurations will be described. Further, the various time- and event-critical factors that affect the system behavior have to be specified. This is particularly critical for automatic mode transitions. On the other hand, the model will not be able to provide us with insight about the problem of mode ambiguity and confusion due to the entanglement of both operator tasks and system states.

Summary

The discrete control network provides a general description of the human tasks given the constraints of the machine. Behavior is defined and decision alternatives (and their sources) are provided. This work has allowed researchers to identify several attributes of modal systems: (1) Such systems are hierarchical in nature. That is, when a given component is active, the various subsystems which define it are active too, and states of these subsystems are manipulated to accomplish the task; (2) such complex systems are concurrent. That is, to achieve a task (for example, tracking), information must be passed to a computer from several concurrently active subsystems. Hierarchy and concurrency, it appears, are critical building blocks for describing such human-machine systems.

THE OPERATOR FUNCTION MODEL (OFM)

This model, which is based on the discrete control network research, similarly focuses on the operator activities, given the procedures and the machine states. The Operator Function Model (OFM) is a normative and computational representation for specification of the user's tasks and procedures. The model defines, at various levels of hierarchy, the functions that the user must conduct in order to operate the system for the purpose of achieving his or her goal.

The basis of the OFM is a task analysis framework. The framework is made of *nodes* (states) and *arcs* (transitions). It attempts to represent in graph form how a user might decompose control functions into simpler activities and coordinate those activities in order to supervise a complex system. Nodes at the top level represent major operator functions (e.g., control of the process). Each function is decomposed into a collection of subfunctions (e.g., operating a machine). Each subfunction, in turn, is composed of a number of tasks (e.g., select and engage a mode), and each task is accomplished by one or more operator actions (e.g., enter a reference value). In this manner, the OFM handles the knowledge, information flow, and the selection of functions, tasks, and actions by the user (Jones, Chu, and Mitchell, 1995). Actions in the OFM representation can be either manual (e.g., push a button) or cognitive (e.g., compare two parameters).

While functions, tasks, and actions are represented as nodes, triggering events and completion of actions are represented as arcs. Similar to the Discrete Control Network, the arcs represent the flow of temporal activity in the human-machine system. The formalism allows for concurrency—that is, several nodes can be active simultaneously. Furthermore, there is a global-interrupt mechanism: any transition out of a function, for example, necessitates an immediate transition out of all its nested subfunctions, tasks, and so on.

The application and power of the OFM to represent human-machine interaction is founded on three important observations: First, modern (supervisory) control systems are largely event-driven (Harel, 1987; Sherry and Ward, 1995). Second, many of the task specifications in high-risk domains are based on standard operating procedures (Degani and Wiener, 1994). Therefore, if one is able to capture these procedures, a comprehensive description of the task can follow. Third, many of the important and critical actions taken by operators in modern control systems are discrete rather than continuous (Miller, 1979; Sherry and Polson, 1995).

The OFM has been used to model and aid operators in controlling near-earth satellites (Mitchell, 1987), specify the requirements and design of visual displays (Mitchell and Saisi, 1987), troubleshoot printed circuit board assembly (Cohen, Mitchell, and Govindaraj, 1992), navigate ships (Lee and Sanquist, 1993), and pilot a Boeing B-727 type aircraft (Smith, Govindaraj, and Mitchell, 1990). It has also been used in the development of a blackboard architecture for intent inferencing of operators' activities (Rubin, Jones, and Mitchell, 1988). The use of the model allows for several insights: understanding the behavior of the human-machine system; developing interfaces to better support user's tasks, given this understanding; evaluating procedures and tasks; and developing mission support (and aiding) systems.

Applicability to Mode-Based Systems

The characteristics of the OFM as a method to describe the interaction of operators with a complex system in a discrete fashion make it a valuable method for representing user interaction with a modern control system that employs modes. Indeed, mode engagements and setup are inherently discrete actions. The Operator Function Model can provide related information: It will show the progression of functions, subfunctions, tasks, and actions that the user must perform in order to activate and monitor modes. Therefore, the various sources of information required to progress from one function to the other will be specified, whether these are dependent on the machine state or on the operating environment. Furthermore, the source of this information and how the operator is expected to obtain it (e.g., by moving from one display to another) can be described in the Operator Function Model, thereby providing insight into the cost the operator must pay for obtaining relevant information.

GT-CATS: An Extension of the OFM

Callantine (1996) developed a computer-based tool called the Crew Activity Tracking System (CATS) to predict and explain operator activities in complex control systems. The ultimate goal of this tool is to provide computer support to operators in such domains, with an eye toward both error detection and training. He used an Operator Function Model (OFM) to represent pilot interaction with an Automated Flight Control System of a Boeing 757 aircraft. He augmented the basic OFM representation by explicitly representing automatic control modes. The OFM for systems with Automatic Control Modes (OFM-ACM) deals with situations in which the operator can select among a variety of modes to perform the same function (e.g., there are five modes available to perform a climb-to-altitude function).

On each inferencing cycle, CATS compares the current state of the machine to the current demands imposed by the operating environment, in order to activate so-called ‘context specifiers.’ Thus, the set of active context specifiers summarizes the relationship between the current state of the machine and the current environmental demands. The OFM-ACM uses context specifiers as conditions to define when CATS should expect some pilot activity. CATS filters the active set of context specifiers through the OFM-ACM, predicting user interaction that contains a subset of the active context specifiers. When pilots perform an action (e.g., select a climb-to-altitude mode), CATS compares the action to current predictions. If the action was predicted, CATS explains the action accordingly. If not, CATS uses the structure of the OFM-ACM to determine whether the action can be explained as supporting an alternative valid mode, or if it is a prediction error.

In modern control systems where the user has a variety of modes to choose from, the mode selected by the operator is of utmost importance. In some situations, selecting a “legal” mode may not necessarily provide the most efficient system operation. The OFM-ACM model not only specifies the legal alternatives, but attempts to identify which mode is the most suitable, given the current system state (including the configuration of the automation) and environmental demands. This is done by embedding additional knowledge about preferred modes in a production system structure of context specifiers.

CATS was evaluated using B-757 line pilots flying a part-task simulator. The results indicate that a majority of pilot actions can be predicted and explained by the tool. Pilot errors were also detected. Thus, the CATS research identifies critical aspects of pilot interaction with automated control systems with modes: First, it identifies the types of knowledge specification (procedural, operational/environmental, and machine behavior) needed to operate a complex system. Second, it posits a structure for organizing this knowledge. In doing this, CATS research provides insight into a critical question: Given a complex moded system with a set of task specifications that includes safety and mission objectives, what minimal knowledge specifications must be provided to the user in order to operate it?

The CATS model contains both task and knowledge specifications. And because the model (which contains a predictive capability) has been evaluated and compared with human (pilot) behavior, the findings provide empirical evidence of the importance of the relationship between task specifications and knowledge specifications in understanding human-automation interaction—an issue which is somewhat neglected in the mode-related literature surveyed previously.

OFM for Explaining Mode Usage

Another evaluation of the OFM methodology and the ACM extension for describing human interaction with modes was conducted by Degani, Mitchell, and Chappell (1995). They used the

OFM framework to analyze several mode transitions obtained from cockpit observations in a Boeing B-757 aircraft. Using the Operator Function Model, they represented the progression of functions, tasks, and actions needed to comply with the demands of the operating environment. The model highlighted the relationship among the *environmental demands and states* (e.g., ATC clearances, thunderstorms along the flight path), the *functions and tasks* to be accomplished (e.g., Modify Vertical Path), and the *physical and cognitive actions* the crew must undertake.

The analysis identified a lack of correspondence between the environmental demands placed on the crew and the AFCS mode behaviors: One example showed how the crew avoided mismatches between the environmental demands and a fully automatic mode (VNAV) by transitioning to a semi-automatic mode, and revealed the resulting tradeoffs in fuel efficiency and ease of monitoring. Another example represented the crew's difficulty in understanding the relationship between fully automatic mode configurations and the resulting trajectory of the aircraft. Analysis of yet another example described a possible mismatch between the crew's goal of complying with an ATC clearance and the non-intuitive action sequences necessary to achieve it.

The analysis allowed the researchers to identify some of the limitations and capabilities of this modeling approach for describing mode transitions. One limitation of the OFM, and of many human-machine models, is that it does not explicitly model the behavior of the machine. However, modeling this behavior is critical for describing operator interaction with a control system that initiates automatic mode transitions. Another limitation is the difficulty in formally describing the interaction between nodes, specifically when these nodes are depicted on different branches in the tree-like decomposition. This is particularly important if two or more concurrent processes are coupled, such as temperature and pressure in a boiler or pitch attitude and speed in aircraft (cf. Smith, et al., 1990).

Summary

The many applications of the OFM demonstrate the usefulness of this approach for modeling the behavior of humans in the context of the environment, machine, and procedures. The formalism allows for a concise and clear way to structure the user's goals, functions, tasks, and procedural knowledge, based on internal or external triggering events in the system. The formalism has limitations in describing human interaction with a system that has automatic transitions and heavy interaction between components. Nevertheless, the marked advantage of OFM modeling efforts is that they are based on standard operating procedures and the capability to *formally* describe many aspects of human-automation interaction. Another advantage is that such models are *systematic*—in building the model, the modeler can be complete.

CHAPTER SUMMARY

In this chapter we reviewed several models and evaluated their applicability to modeling human interaction with mode-based systems. Seven such models were reviewed, ranging from conceptual models to computational and behavioral models. Before proceeding, it is important to mention a common, yet sometimes ignored, caveat of any modeling representation. Such representation is, in some sense, a mapping of derivatives from the phenomena to be described onto a formal matrix of coordinates. Each receiving matrix has its formal characteristics which will, *in principle*, be distortive of the phenomena to be mapped onto it (Bateson, 1979, xviii; von Bertalanffy, 1968, p. 200). In other words, the use of any model, representation, or language, filters the observed phenomena. Putting the argument on its head, this limitation has some advantages, as the underlying

matrix and initial assumptions of the modeling approach must be put forth explicitly. By doing this, a formal model is a more faithful representation than a subjective, ad-hoc evaluation of human-machine interaction.

The review on models of human-machine interaction identified several candidate models, as well as their limitations and advantages for modeling modes. The focus was on models that represent the user's task and not the user's psychological aspect. As such, a common feature to all these models is that a very extensive analysis of the user task is required (Kieras, 1983). Any model succeeds or fails depending on the quality of this initial process. Another common characteristic is that many of the models were derived from work in computer science and human-computer interaction. As such, they were constructed in the context of office automation and control systems in which humans determine, via their actions, the state and mode configuration of the system.

In contrast, automated control systems operate in real time and are event-driven, reacting to both external and internal stimuli. While incorporating many of the features of office automation with respect to data entry, such systems have considerable autonomy. This, in the context of this research, translates to automatic mode transitions—a characteristic which contributes to mode ambiguity. Another characteristic of large control systems is interaction (coupling) between concurrent components and their respective modes, leading again to the potential of unexpected mode configurations. Nevertheless, most of the above models do not explicitly describe such reactions.

The results of this review suggest that none of these languages, alone, is sufficient for representing the human, the machine, and the operating environment. One possible solution to this problem was to combine a machine model (e.g., Finite State Machine) with a model of human functions and tasks (e.g., OFM). Once we realized that a Finite State Machine model could serve to model both, the question became which FSM variant to use.

Further investigation has led us to believe that Statecharts, a modern extension to the Finite State Machine model, can be used as a language for a framework containing the machine, the task, and the environment. The next chapter reviews the Statecharts language and describes some of its features using two examples.

STATECHARTS AND OFAN: A LANGUAGE AND MODELING FRAMEWORK

STATECHARTS: A LANGUAGE

Statecharts was conceived as a specification language for computer-driven systems, initially avionics systems (Harel, 1987). It was developed for modeling large, complex, reactive systems. A reactive system, in contrast to a transformational system, is characterized by being event-driven (Harel and Pnueli, 1985). It must constantly react to stimuli from external (e.g., environmental) and internal (e.g., time-out) events. Harel (1987) includes in his examples of such systems telephones, computer operating systems, telephone networks, and the human-machine interfaces of many kinds of ordinary software (p. 231).

Traditionally, Finite State Machine models were used to specify such reactive systems. However, as the complexity of reactive systems increases, the number of corresponding states grows exponentially and becomes quite difficult to comprehend. Statecharts extend the FSM notion, producing modular, hierarchical, and structured diagrams (Coleman, Hayes, and Bear, 1992). These are achieved by employing three basic extensions: hierarchy and default entries, concurrency, and broadcast communication.

Hierarchy and Default Entries: The Household Thermostat

We introduce the use of the Statecharts language through two examples: a household thermostat and a light switch. The thermostat has two buttons—‘on/off’ and ‘mode.’ When the user presses the ‘on/off’ button, the thermostat is turned “On” and the “Fan” mode is active. Any subsequent press on the mode button scrolls the thermostat modes to “Heat” and “Air Condition.” One way to represent such a system is shown in Figure 4-1. Here we have an initial *IDLE* state and all other thermostat modes also drawn as states. The change (or transition) among modes occurs when the user presses the mode button. Pressing the button is an external event (done by the user). As a consequence, an internal event—the system transitioning from *IDLE* to *FAN*—takes place along the marked transition (arrow).

Hierarchy

The abundance of transitions with the *off* label in Figure 4-1 indicates that any time during the operation, pressing the ‘off’ button will send the thermostat back to its idle state. One way to minimize the number of such transitions is to cluster all the operational modes into some newly created superstate, which we will call *OPERATIONAL* (Figure 4-2). Now we can replace the four arrows by a single one. This arrow emanates from the boundary of superstate *OPERATIONAL*, meaning that it is a global transition. When the event *off* occurs, the transition from *OPERATIONAL* to *IDLE* takes place, regardless which of the operational modes is active.

We have therefore created a hierarchy. We have a superstate and several substates embedded inside it. As mentioned earlier, many systems are understood and built in a hierarchical manner, and therefore such a modeling structure is quite appealing.

Thermostat

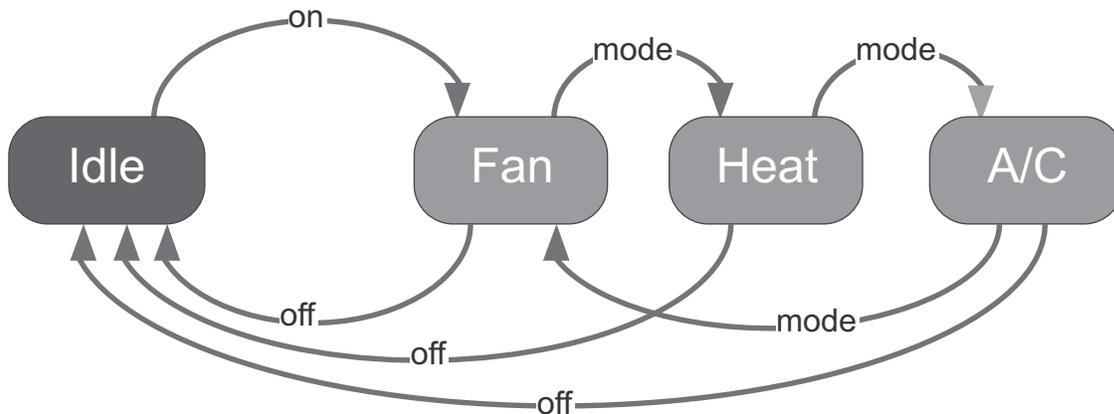


Figure 4-1. Household thermostat (in transition state diagram).

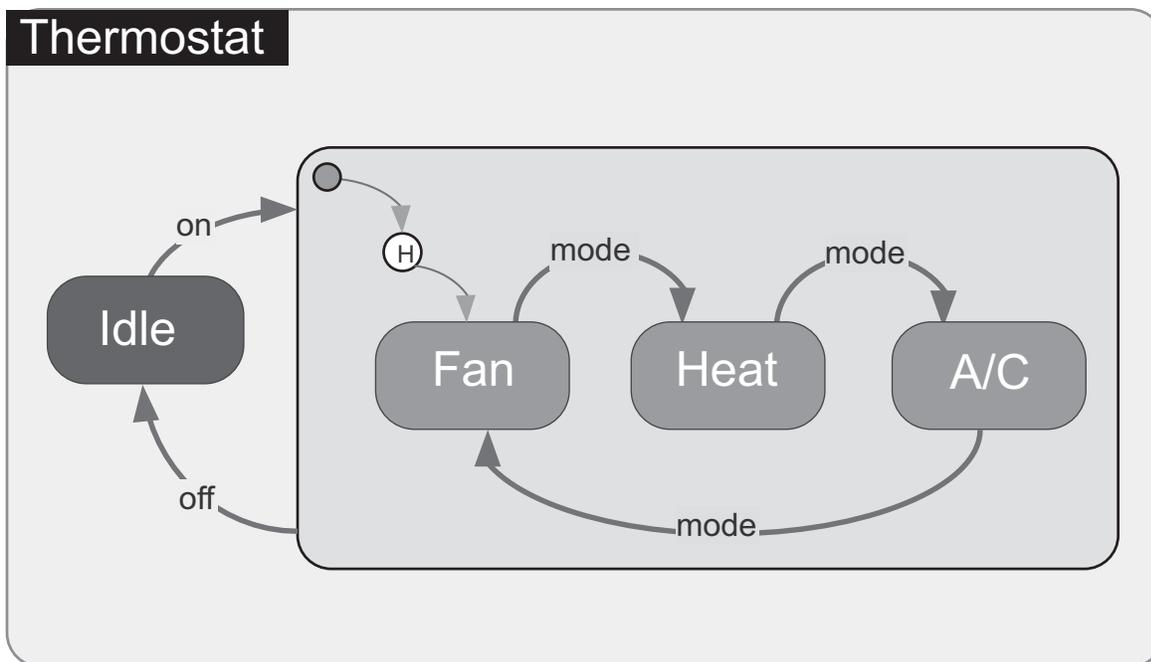


Figure 4-2. Household thermostat (in Statecharts).

Mathematically, the superstate *OPERATIONAL* is an exclusive .OR. (.XOR.): While within *OPERATIONAL* one must be in either *FAN*, *HEAT*, or *AIR CONDITION*.

Default

Default states quickly become an important element when using hierarchical representation. When a superstate is entered, the default specifies which state is initially entered. In Figure 4-2, when we leave *IDLE* and enter *OPERATIONAL*, the default state is *FAN*. Sometimes, however, the default state may change. For example, the thermostat logic (behavior) has some memory: When we re-enter *OPERATIONAL*, we go back to the mode most recently visited. The Statecharts formalism

uses the symbol (H) to represent such a history entry. This type of history is applied only at the level in which it appears (Harel, 1987, p. 238).

Concurrency and Broadcast: The Light Switch

The thermostat described earlier was actually installed in a small pool house. The pool house has a main corridor leading to a central room. We can represent the pool house as a large rounded rectangle. Inside the pool house there are two components—a corridor and the central room (Figure 4-3). Activities occurring in the corridor can occur in parallel with activities occurring in the central room. The activities are said to be occurring concurrently. We represent the notion of such concurrency by means of a broken line which splits the pool house into two components, corridor and central room. Within each component there are additional subcomponents. The central room contains the thermostat and a large light fixture; the corridor has an ordinary light switch.

Inside the small pool house we created a concurrency. The processes in the control room run concurrently with the process in the main corridor. Again, many complex systems have such a feature (e.g., a network of two computers). Mathematically, concurrency is represented as an .AND. product. That is, one must be in all subcomponents concurrently. Being in *POOL-HOUSE* entails being in an .AND. combination of *MAIN CORRIDOR* and *CENTRAL ROOM*.

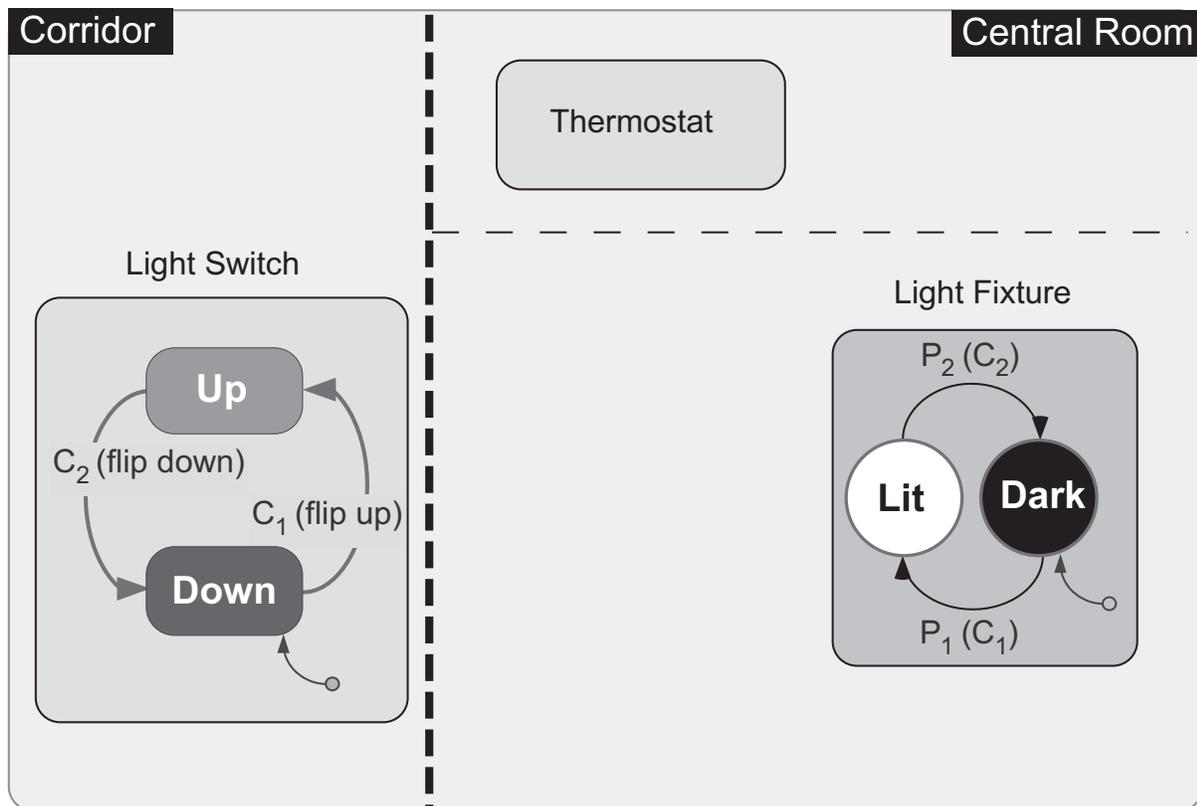


Figure 4-3. Light switch and fixture (in Statecharts)

The light fixture on the ceiling has two states—Dark and Lit—which represent its behavior. We can also represent the behavior of the switches Down or Up. But what causes this change of behavior? The answer is that a transition from Down to Up takes place because some external event (a person)

flips the switch. We represent this transition as an arc with an arrow at the end, pointing in the direction of the transition. We can also describe the event that causes the transition to occur in the first place. We write this event within a parenthesis. In this example, therefore, the triggering event is that “a person flips the switch.” When this external event takes place, several things happen: An internal event C_1 takes place, a transition fires, and the state of the light switch changes from Down to Up.

This sequence of events illustrates another feature of Statecharts: its broadcasting mechanism. By *broadcasting* is meant that every event, or entry and exit of a state that occurs in the system, is communicated to the entire network. Using such a mechanism one can specify side effects: An event in one part of the network can affect another. The general syntax of an expression labeling a transition in a Statecharts is of the form: “ $e(c)/a$ ”, where e is an internal event that causes the transition (\rightarrow) to take place, c is a condition that guards the transition from being taken unless it is true when e occurs, and a is an action that is carried out if and when the transition is taken (Harel and Naamad, 1995, p. 3).

This model of the pool house allowed us to identify some of the characteristics of this system: its hierarchical and concurrent structures, its initial state configuration, its external events, its internal events and transitions, and its new state configuration. The Statecharts model hence provided a behavioral description of this system.

Light Switch B

Our definition of the modeled system can grow as far as we wish: We can add as many components as we like, or provide more detail (depth) to existing components. For the purpose of this discussion, let us say that in addition to the light switch in the main corridor, we find another one inside the central room. We now add another concurrent component in the central room and call it light switch B. Both switches control the light fixture. Figure 4-4 now depicts this extended model.

Analysis of the Model

The figure shows the initial state configuration is that both light switches are in the down position and the light fixture is dark. We first flip switch A to Up, and the fixture is lit. We then flip switch B to Up. Now both switches are Up, and the room is dark. If we place *both* switches down the room is dark again. We now note that there are two configurations possible: (1) both switches in the same position (Up-Up, or Down-Down)—which results in darkness or (2) one switch is up and the other is down (Up-Down or Down Up)—which results in the fixture being lit. Table 4-1 shows this state and event relationship.

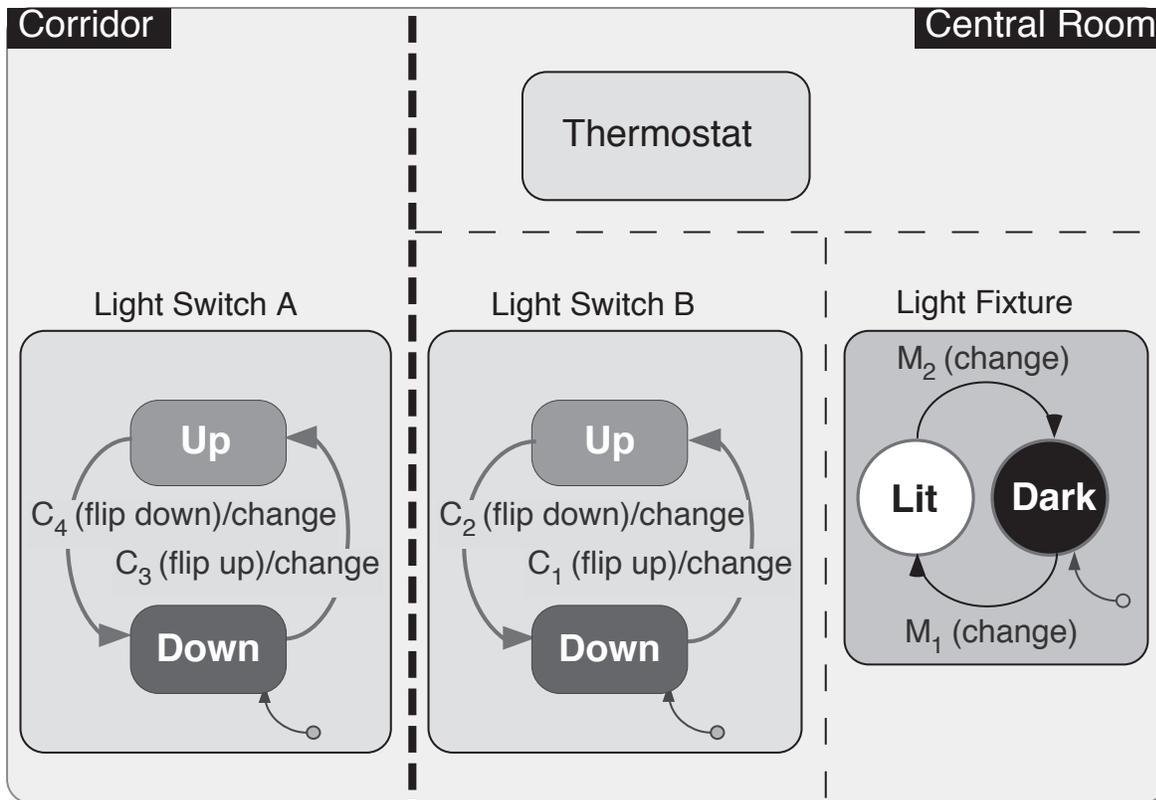


Figure 4-4. Two light switches—one fixture

Table 4-1. Switch (A and B) and Light-Fixture State Configurations

Switch A	Switch B	Light Fixture
DOWN	DOWN	DARK
UP	DOWN	LIT
UP	UP	DARK
DOWN	UP	LIT

An alternative way to look at the behavior of this system is by noting that any flip (Up or Down) of any switch (A or B) will change the current state of the light fixture. Why are there two unique configurations (Down-Down or Up-Up, and Up-Down or Down-Up)? The answer is the initial set-up of the light fixture (*DARK*) and the light switches (Down-Down): Once this default arrangement is set by the electrician during installation, any flip of a switch will result in these two types of configurations. The same applies for the single light switch in Figure 4-3—it just looks as if a unique event and state causes the light to come on. In actuality, all that matters is the default configuration (Down and Dark) and sensing of a flip. With these two attributes, the system can be represented completely. Interestingly, an informal survey of our lab members yielded various (conspiracy) theories: One is that there is some central controller that controls the behavior of the system. A somewhat related alternative theory is that there is a master-slave relationship between the switches (a theory which the present author religiously held until building this model).

Population Stereotypes

Many people are confused by such an arrangement of two light switches and one fixture. One reason is the lack of consistent mapping: one time flipping the switch down will turn the light off, yet another time it will turn the light on (Wickens, 1992, chap. 8). Another factor is that the latter (flip-down results in light-on) is a violation of a “population stereotype” in North America. This term was first coined by Paul Fitts (1951), who asserted that displays should provide a “natural” indication of control action, and that these relationships, in some cases, are so uniform as to constitute a population stereotype (p. 1306). Several studies seem to indicate that in many cases, population stereotypes define mappings that are more related to custom—and hence experience—than to some psychological norm with respect to selection of action (Smith, 1968; N. Moray, personal communication, August, 22, 1996). And indeed, users in Europe will not find this flip-down resulting in light-on mapping problematic—that arrangement is the norm in Europe. Population stereotypes are of concern in designing mode-based systems, because it appears that discrete actions (flipping switches) are more prone to these customs and hence habitual responses than continuous actions (manipulating a joystick) (Loveless, 1962).

The light switch example will be relevant to both North America and European populations, because the arrangement contains *both* stereotypes. The lack of consistent mapping occurs because of state dependency between the two switches (resulting in the two aforementioned configurations), facilitated by the default arrangement of switches and light. This inconsistency will change depending on the other switch position (cf. Grudin, 1989). Our (humble) advice to the reader who encounters such an arrangement is to try to forget all about switch positions and population stereotypes—instead use the switch as means of changing the current state of the fixture. Nevertheless, as Loveless shows in his review of the related literature, regression to the (old) habit may occur in circumstances generally described as ‘stressful.’

Summary

We described in this section the basic elements of Statecharts—a visual formalism for representing the behavior of complex, real-time, reactive systems. We explained the behavior of a common household thermostat and light switches. Furthermore, using the model, we identified the operational logic behind a double light switch and why it sometimes confuses people.

Statecharts have many other features of applicability to complex systems, as well as several possible semantics. We described here some the language features that will be used in this thesis. A more detailed description of the Statecharts and their many features can be found in Harel (1987). Discussion of the various semantics that were applied to the Statecharts language can be found in the following list of papers (Harel and Naamad, 1995; Harel, Pnueli, Schmidt, and Sherman, 1987; Huizing and de Roever, 1991; Leveson, Heimdahl, Hildreth, and Reese, 1994).

In modeling human-machine interaction, Statecharts are less conspicuous than some other formalisms. Statecharts are used in computer science and control theory disciplines as specifications for modeling the behavior of a given machine or a piece of code. Less often, they are used for modeling displays and controls—and this is done mostly by commercial (e.g., avionics) companies (Blake and Goeddel, 1993). Likewise, we failed to find any literature pertaining to modeling of user’s task by means of Statecharts.

Nevertheless, based on the above literature review, we believe that the features of Statecharts are well suited for describing human interaction with a machine and its interfaces that exhibit modes. There are several reasons for this: (1) any Finite State Machine approach maps quite well to modes

description (Jacob, 1986); (2) hierarchy allows us to deal with the organization of modes and submodes within a system; (3) concurrency allows us to describe a multi-component system in which several modes are simultaneously active, and (4) broadcasting allows us to represent interaction between concurrent process. Generally speaking, the Statecharts formalism represents the control behavior of the system—what is responsible for making the time-dependent decisions that influence the system’s entire behavior (Harel, 1988). In the next section we describe how we combine the description of the human tasks, machine behavior, and environment states into a single framework, called Ofan.

OFAN: A PROPOSED FRAMEWORK

We have previously argued that problems of human-mode interaction cannot be understood without some kind of a language and framework to describe them. In the previous section we described a language that appears suitable for the job. We shall now describe a modeling framework.

Components

The Ofan framework uses the Statecharts language to describe the environment-human-machine system as five concurrently active modules: (1) the *Environment*, (2) the *User Tasks*, (3) the *Interface*, (4) the *Control Mechanism*, and (5) the *Plant*. Figure 4-5 presents the synthesis of the five elements into a framework.

Environment

Our literature review identified the importance of the demands of the operating environment on mode-based systems (Callantine, 1996; Casner, 1994; Miller, 1979; Wiener, 1989). The Ofan modeling framework contains a description of the environment, its structure, its active states, and the demands (constraints) it places on the user and the machine. The *Environment* module describes the events in the operational environment as a collection of concurrent states. Although we realize that this view of the environment is somewhat simple, it serves well for feeding triggering events into the *User Tasks* module as well as others (cf. Bråthen, Nordø, and Veum, 1994, for a similar approach).

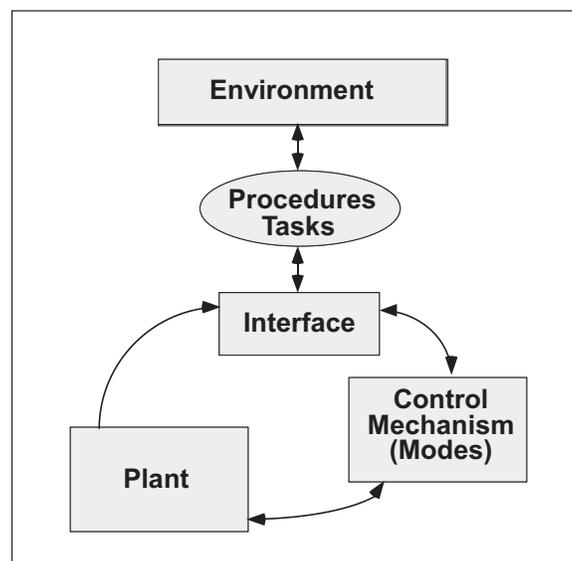


Figure 4-5. Synthesis of the five elements into a framework

User Tasks

Our approach for modeling humans is to represent the user tasks, instead of a more cognitive model of human behavior. This we argue, allows us to easily integrate this module with the representation of the environment and the machine. We resisted many suggestions to “beef up” the User Tasks model into a more psychologically relevant representation for two reasons: (1) such representation (at least drawing from those available in the literature) would have required some ‘filter’ and an interface to the machine module; (2) we wanted to keep the framework as simple as possible so that an engineer with no training in cognitive psychology could use it. We believe, after Hoare (1980), that by using a complex language to describe a complex system, “the language itself becomes part of the problem rather than part of the solution” (p.156).

Here, as in the Operator Function Model, we are not modeling cognition, perception, or decision making. Instead, our focus is on low-level manifestations of human cognition, the *observable* actions that a well-trained and motivated operator had to perform (Bossler, 1986; Bossler and Melchoir, 1990). What we are modeling in the User Tasks module is task specification in terms of the start- and end-states of the task, and the procedures to bridge this gap. This module describes how the operator executes tasks (modeled as states) and the events that trigger them.

The last component is the machine itself. The reason we describe the machine behavior is argued earlier, and is quite straightforward: mode transitions cannot be detailed without an accurate description of the machine’s behavior. In describing the machine, we speak of three subcomponents: the interface, the control mechanism, and the plant.

Interface

The interface represents the medium between the human and the machine, and is further broken into two components—the display and the controls. The *Controls* submodule describes the machine manipulators (e.g., buttons, dial knobs). The *Displays* submodule describes the status of the interface element through which feedback is provided to the operator.

Control Mechanism

This module presents the control system—its components, states, inputs, and transitions. It describes the structure and logic behind the control of the machine.

Plant

The Plant module describes the physical components of the machine—its actuators, pumps, etc.

The Ofan Framework as a System

The Ofan framework represents the complete environment-human-machine ensemble as a set of concurrently active modules—all interconnected, via triggering events, to form a whole. Each module contains a process and broadcasts its active states and events to other modules in the network. Depending upon this communication, the other modules can change state. The system representation is somewhat analogous to a set of cogwheels: an event in one wheel affects the adjacent wheel and so on, in perpetuum (and hence the name *Ofan*—Hebrew for a set of perpetuating wheels).

Matching Representations

The notion of modeling elements that are very distinct in nature, yet have input/output relationships by means of a common symbology, is very powerful. In order to note the relationships among the

elements (or modules) of a system, some representational common denominator is required. Several researchers have called for modeling the human and machine components of the overall human-machine system using the same formalism. The concept, however, is not new—it was employed extensively in the manual control research. The benefits are obvious: (1) the two processes can be merged to obtain a more complete picture of the interaction (Palanque and Bastide, 1996); (2) the relationship between the human tasks and the machine's mode configuration can be evaluated (Kieras and Polson, 1985); and (3) it is possible to identify synchronization and resulting mismatches, as well as determining the effects of a failure or fault in one component on another (Leveson, 1987).

Finally, perhaps the most important benefit of using the same formalism to model distinct elements is the ability to quantify the relationship between the two processes. Corker and Bejaczky (1984), in modeling human interaction with a remote manipulator (of a spacecraft), modeled both the human's limbs and the manipulator arm, a mass spring system, using a common symbology. They note:

It is of value to man/machine interface design that neuromotor control models be formulated in the same terms as control system descriptions. ... If human and teleoperator controls are expressed with a similar nomenclature and found to obey similar control laws, interactions can be described, and compensation for machine or human limitations can be more easily accommodated in the design for optimal system function.

The similar nomenclature and control laws allowed Corker and Bejaczky to apply gravity compensation for weightlessness conditions. Furthermore, instability due to external disturbances (e.g., motion of the spacecraft) could also be compensated for, in some cases.

Interrelationships

From the research point of view, the five elements, or modules, of the framework are regarded as a system. The glue that binds the five modules into a whole system is the interaction among the modules (von Bertalanffy, 1968, chap. 3). These interactions are represented as events and conditions that restrict and enable transitions across modules. When an event occurs in one module, the information is broadcast and may fire another event somewhere else. This also allows for synchronization of these events—a common feature of large and complex systems (e.g., telephone networks). Finally, since all the modules of our system are represented using the same formalism, we can represent all the interactions in the human-machine system—the focus of any human factors analysis.

CHAPTER SUMMARY

In the preceding chapters we reviewed modes and models of human-machine systems. In Chapter II, we reviewed the literature on modes and their usage in a variety of domains. We identified three subtypes of modes (interface, function, and supervisory) and discussed their differences with an eye toward identifying their structural features in the human-machine system. In Chapter III we reviewed some of the literature on models of human-machine systems, starting with conceptual models, moving to simulation and continuous control models, and finally settling on behavioral and discrete models of device behavior and user procedures. We identified some of the analysis abilities of these models in modeling human interaction with modes. In Chapter IV, we have described in detail the Statecharts language and proposed our modeling framework—Ofan.

In the context of human-machine systems there are several major uses of models. Developing a mathematically based model requires a precise and organized understanding of the system being modeled; this process is iterative, as results are may be compared to the actual system and more insight gained. Other uses are for explaining data, guiding the design of future experiments, and providing quantitative information on how the modeled system (for example, a new machine) will behave (Rouse, 1986, chap. 1). In addition, we argue that if one wants to use any representation for analysis purposes, it is not enough just to model the system. The development of the approach and modeling framework must be such that the patterns of interest (e.g., problematic features of the design, failure points) will be highlighted and brought to the surface by the modeling approach. Specifically, modeling templates to bring these problematic features to the surface must be constructed in a systematic (and not ad hoc) way.

In the next section of this document, consisting of Chapters V-VI, we represent five examples (case studies) of human-machine systems. These examples were briefly described in the introduction section (Chapter I), and the mode problem for each example was *informally* noted. Using the Statecharts language and our Ofan framework we now describe them formally. In addition, we develop modeling templates to deal with common architectures of mode-based systems. In Chapter VI, we identify and categorize some of the features that contribute to mode confusion.

CASE STUDIES

In this chapter, five examples of human-machine systems, all modeled using the Ofan framework, will be presented. We will use the model to illuminate some mode ambiguity problems. Combining the Ofan representation of the system and a corresponding analysis adds a representational power to our discussion of modes and ambiguity. The human-machine systems that we model and analyze here—which we briefly mentioned in the Introduction—have known mode problems. In one sense, this is a post-mortem investigation whose goal is to test the modeling framework, identify the actual feature that led to the resulting confusion, and search for some commonality among these features. We start with very simple systems and move to more complex ones.

CORDLESS PHONE

We first model, analyze, and discuss the mode problem associated with a popular cordless telephone. The problem has to do with the sequence of actions the user takes in picking up the phone and talking to the caller. Depending on whether the phone is on its base or “off-base,” there exists the potential of disconnecting an incoming call, when in fact all the user wanted to do was to pick up and answer.

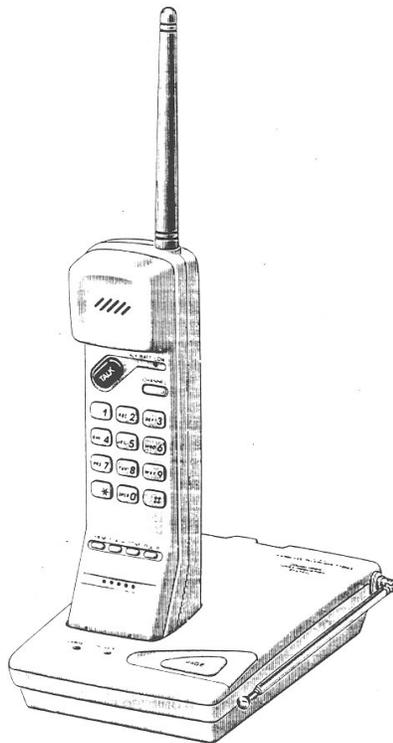


Figure 5-1. Line drawing of the cordless phone

Description

The cordless phone has two integral components: a base (cradle) and a handset (phone). The handset can reside in either of two places: it can be left on the base (and be charged) or it can be left off-base. Figure 5-1 is a line drawing of the phone.

In most phones on the market today, the user presses the ‘talk’ button to activate the phone (either for answering incoming calls or dialing an outgoing call). This particular phone also works in this way, with one exception: when the handset is on-base and an incoming phone arrives, the user lifts up the handset from the base and the handset is already activated (the line is open and there is no need to press the ‘talk’ button). When the handset is off-base, on the other hand, he or she must press the ‘talk’ button.

The problem occurs when the user picks up the phone from the base and (without due consideration) presses the ‘talk’ button. The rather embarrassing consequence is that the user hangs up on the caller! Now, not knowing who called, the user can’t remedy the situation—no recovery is possible. The problem, at least anecdotally, is rather common among users of this cordless phone type.

Model

The model of the control mechanism includes two superstates: *ON-BASE* and *OFF-BASE*. In *ON-BASE*, the default state is *IDLE* (Figure 5-2). Once an incoming call arrives, the transition to *EXTERNAL REQUEST* takes place. If the caller decides to hang up before our user can open a channel, the control mechanism transitions back to *IDLE*.

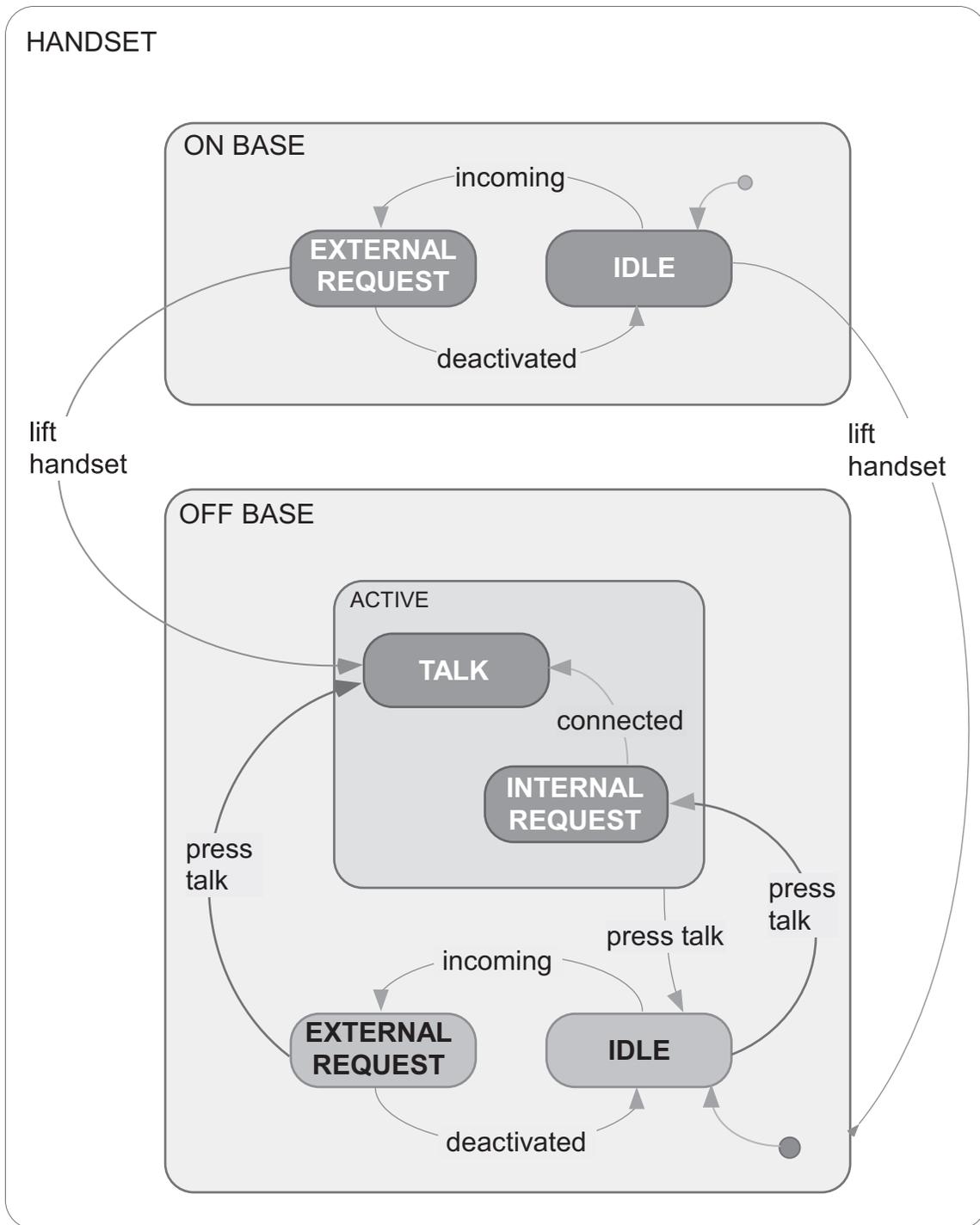


Figure 5-2. Model of the control mechanism of the cordless phone

The transition to the *OFF-BASE* superstate occurs when the user lifts up the phone from the base (event: *lift hand-set*). As long as the phone is not ringing, the transition is into *OFF-BASE* and an immediate default transition to *IDLE*. When the user decides to initiate a phone call, he or she presses the ‘talk’ button which activates the handset. In this (*ACTIVE*) superstate, the user may dial a number and when connected, transition into *TALK*. While the handset is active, the user may press ‘talk’ again to deactivate the handset (and subsequently hang up or stop the dialing process). If an incoming phone call arrives (*EXTERNAL REQUEST*) while the handset is off-base, pressing the

‘talk’ button will make the transition into the *ACTIVE* superstate and its *TALK* substate. There is, however, another way to get into the *TALK* substate. If the phone rings (*EXTERNAL REQUEST*) while in *ON-BASE*, lifting the phone will cause an (automatic) transition to *TALK*.

There are two paths, therefore, to get into *TALK*. One is by pressing the ‘talk’ button while the handset is off-base. The other is simply by lifting the handset from its base. The selection of path is dependent upon the previous state of the handset.

Analysis

As Figure 5-2 shows, the two transitions out of *ON-BASE* are prompted by the same event (*lift hand-set*). The transitions, however, start at two different substates: *IDLE* or *EXTERNAL REQUEST*. As a consequence they also end at two different substates: *IDLE* or *TALK*. The destination of the transition is dependent upon the previous state of the device. This structure and behavior of the device is not unique or more unusual than other consumer electronic devices. The potential problem emerges when the user is not cognizant of the difference between these two initial states.

The user’s sequence of actions, once a phone rings, is to lift the handset to the ear. Once the handset is already in the hand of the user, he or she is at a decision checkpoint—to press or not to press the ‘talk’ button—because, again, this action can either activate the phone line or hang up on the caller.

Table 5-1 describes the sequence of actions on part of the user and the corresponding system states.

Table 5-1. Tabulation of States and Events

Environment	Initial State	User Actions (events)	End State
‘Incoming’	OFF-BASE	Lift handset, press ‘talk’	TALK
‘Incoming’	ON-BASE	Lift handset, press ‘talk’	IDLE (after being in TALK momentarily)

To operate the device, the user must remember, when holding the handset, whether the previous state was on-base or off-base in order to resolve the next state configuration. Two sources of information are there to aid the user: The first source is a small indicator, about 1/8th of an inch in diameter, which lights up to indicate that the line is open when the user lifts the phone from the base. The second source of information is whether the handset was on-base or off-base when it was picked up.

While the above state dependency may lead some users to make an error from time to time, the problem is further compounded because of the characteristics of the ‘talk’ button. The ‘talk’ button has a circular state-transition architecture. That is, pressing the ‘talk’ button while the phone is *IDLE* will activate the handset; pressing it while in *ACTIVE* will deactivate the handset (and hang up on the caller). Pressing on the ‘talk’ leads to two different outcomes.

The state-dependency structure of the phone behavior, with respect to picking up the phone and pressing the ‘talk’ button, leads to mode ambiguity. That is, some action on the part of the user *can* have different effects on the machine depending on its current state (Monk, 1986, p. 314). The resulting mode confusion can, of course, lead to mode error.

Discussion

In the literature review presented earlier, we noted that several researchers assert that one way to curtail mode ambiguity is to provide additional feedback to the user. If we look at this particular device, we can see that all the pertinent mode information is provided: First, the ‘talk’ button has a red light to indicate whether or not the line is open. Next, the environment (external to the display) provides additional information—whether the handset is on-base or off-base—that allows the user to resolve what the next state configuration of the phone will be. All the necessary information is there, and in a very redundant environmental and display form (Kirluk, Kossack, and Shively, 1994). Nevertheless, from anecdotal information it appears that even users with an accurate and complete knowledge of the system still fall into the trap of inadvertently disconnecting incoming calls. Why does this happen?

It appears that there is another factor here at play—a psychological one. The sequence of actions that leads to two different outcomes is sometimes referred to as “capture error” (Norman, 1988, p. 107; Reason, 1990, p. 68-71). A common analogy is driving—“I intended to drive to Place X, but then I ‘woke up’ to find that I was on the road to Place Y” (p. 73). This error type was initially described by Reason (1979, p. 85) as the branching error type—“where an initial sequence of actions is common to two different outcomes, and the wrong ‘route’ is taken” (p. 73). Branching errors stem from a failure to verify the progress of an action sequence at key “checkpoints,” with the result that actions proceed toward a different outcome (p. 72). The key checkpoints, or critical decision points, occur when actions or situations are common to two or more motor programs (p. 76). In this special case the action is lifting the handset—a common action to both paths. At this point two avenues are open: pressing the ‘talk’ button and answering the caller, or simply talking to the caller.

Reason (1979) remarks that such errors are “almost invariably drawn from the individual’s own repertoire of highly skilled or habitual action sequences, and reveal the existence of systematic bias toward the activation of long-established motor programs.” Indeed, most phones nowadays (e.g., cellular phones) require pushing the ‘talk’ button before speaking.

Reason (1979) further discuss the issue of prediction. He argues that such errors will occur when an habitual process coincides with a critical decision point and where the strength of the motor programs beyond that point are remarkably different. “It is also predicted that when such an error will occur, it will involve the unintended activation of the strongest motor point beyond the node” (p. 79).

Having a formal model of the phone behavior allowed us to go one step further in our understanding of such architecture. The label *capture error* denotes only the outcome of the situation; it does not say much about the features of the machine (or environment) that led to this so-called error. Both in Reason’s and Norman’s definitions, the problem is internalized—they do not describe the external conditions of the environment and machine that have interacted to produce the error. One advantage of our modeling representation with respect to human-machine interaction is that one can ‘see’ these possible avenues that are available after a given decision checkpoint. The model can be used to illuminate these paths.

Summary

The phone example is used to show how a formal model can illuminate the behavior of the device and its relationship to a certain psychological phenomenon. The transitions to *TALK* or *IDLE* are state-dependent. And while the user interacts with these modes, the potential for capture error exists

because of habitual and therefore dominant sequences of action. Finally, the circular structure of the ‘talk’ button makes the device prone to errors that allow for no recovery. In some respect, the phone and the light switch examples are similar. In these two cases psychological factors (i.e., capture error and population stereotypes) interact with the behavior of the device to produce the potential for mode error.

In the phone example, we have modeled the behavior of the control mechanism of the device. No additional parameters such as reference values were present. Furthermore, we modeled the user interaction as external events to the system. In the next example, we describe an electronic clock/radio. This device has modes and reference values. We show a template that was developed to deal with the relationship between the two. We also introduce the interface module of the Ofan framework and explain how the interface may be modeled from the point of view of the user.

ELECTRONIC CLOCK/RADIO

The device is a common household clock/radio. Preparing the device for waking up the user (at the desired time) requires the setting of modes and reference values. As mentioned earlier in the Introduction chapter, mode ambiguity may occur here because of the relationship between the alarm (mode) and the volume (reference value) of the radio: If the volume is set below the hearing threshold, the alarm will not sound even though the alarm mode is engaged. This particular situation, of turning down the radio volume the night before and forgetting the consequences with respect to the alarm feature, has occurred to the author many (late) mornings.



Figure 5-3. Picture of an electronic clock/radio

Description

The clock/radio device has two integral components: an electronic clock and an AM/FM radio. The configuration of the device is a combination of the modes of these two components, and their associated reference values (as well as several other components not relevant for this discussion—such as a night light). The interface includes a ‘mode’ switch with three positions: ‘on,’ ‘off,’ and ‘alarm.’ “On” and “Off” are the two modes of the radio. The “Alarm” mode is a request (arm) for operating the radio at the alarm set-time. Figure 5-3 is a picture of the device.

Model

The Ofan model described here contains the Interface and the Control Mechanism modules. Figure 5-4 is a partial Ofan model of this device.

Interface

The sliding switch used for selecting the mode of the clock/radio is described as a three-states element. The user can select among the three mode settings—“On” (the default), “Off,” and “Alarm”—by sliding the switch back and forth. The interface has two additional (concurrent) components—visual and audio. On the visual display, we describe the indication on the clock that the alarm is armed—a red dot appears. The transition between no alarm (blank) and arming the alarm (red dot) is predicated upon the switch being set in the “Alarm” position.

The audio interface has also two states—*OFF* and *ON*. The default state, *OFF*, indicates that no alarm is heard by the user. *ON* indicates that the alarm has sounded and the volume is above the user’s hearing threshold. Conditions for the transition are predicated on the states of the control mechanism, and will be discussed later. The main point here is that the interface is modeled from the user’s perception, and not from that of the machine outputs.

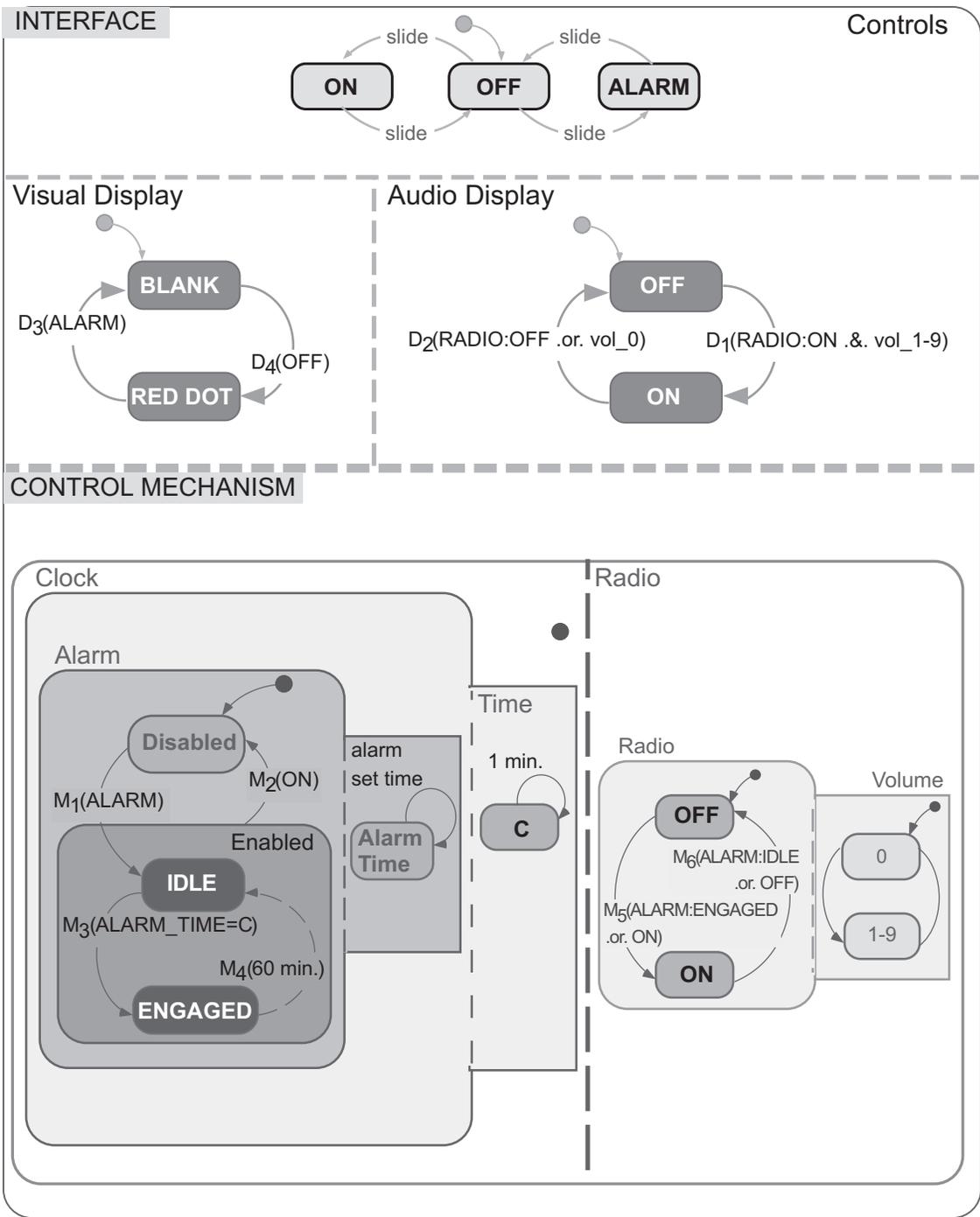


Figure 5-4. Model of electronic clock/radio

Control Mechanism

As mentioned earlier, the clock and a radio component make up this device. *CLOCK* is a superstate, and time is its reference value. Time can be described as a single state with a one-minute update cycle. We depict this relationship as a superstate (*CLOCK*) with a concurrent state (*TIME*) “glued” to it like a handle to a pan. Encapsulated within *CLOCK*, we find another superstate—*ALARM*—with *ALARM TIME* as its reference value. *ALARM* has two substates: *DISABLED* is the default state; transition to the adjacent state, *ENABLED*, occurs when the user

slides the switch to the “Alarm” mode. *ENABLED* defaults to *IDLE*—in which case the alarm mechanism is inactive. Once *clock_time* is equivalent to *alarm_time*, the alarm is engaged. Conversely, an internal (automatic) transition back to *IDLE* ensues when the alarm is engaged for sixty minutes. Such timed transition out of the *ENGAGED* state is called a “timeout.”

The radio is also a superstate and *VOLUME* is its reference value. *VOLUME* has ten graduations, zero to nine (0–9). In state zero, the audio output is below the user’s hearing threshold; in states 1–9 the audio is within the human hearing threshold. *RADIO* has two states: *OFF* and *ON*. The default is *OFF* and the transition to *ON* occurs when either one of two conditions is met: the user places the sliding mode switch to ‘on,’ or the alarm is *ENGAGED*.

The two components, clock and radio, work in parallel, yet there are some links between them. We use the Statecharts broadcasting mechanism to link these two components. The particular condition is shown as guards—that is, once a transition is broadcast to the net, the condition becomes true and a guard is lifted on another transition. The transition from *RADIO: OFF* to *RADIO: ON* is conditional upon the clock being in *ALARM: ENGAGED* or the mode switch being moved to the ‘on’ position.

Returning to the Interface module, we now revisit the audio display. Two states are depicted here—*OFF* and *ON*. The transition from *OFF* to *ON* is predicated upon two conditions being true at the same time: (1) the radio is *ON* (alarm is engaged) and (2) the volume setting is between 1-9. The transition back to *OFF* is predicated upon either the radio being *OFF*, or the volume being set to 0.

Analysis and Discussion

The transition from ‘OFF’ to ‘ON’ in the audio display is the crux of the problem. Even though the alarm has engaged, the device will not sound an alarm if the user, directly or indirectly, has set the volume to 0. That is, although the mode has indeed engaged, it is not perceived as active by the user. We make the distinction here between the machine’s output (engaged) and the user’s input (active). The example also shows the general form of a mode behavior: an exclusive mode and its associated reference value. Only the combination of the two marks the mode behavior of the machine. As we can see, there is a fundamental interrelationship (coupling) between the two which can cause a breakdown if not attended to.

Mode ambiguity is always in the context of the task specification. In this case the (implicit) task specification is that the user should be able to differentiate (while asleep or not attending the display) between idle and alarm modes. The alarm clock example contains mode ambiguity because the user, based on the display, believes and acts as if the clock is in one mode (and continues sleeping), while it is actually in another. In contrast to the previous two examples of the phone and light switch, here the interface (at a certain setting) is insufficient to resolve between the two modes.

Summary

The electronic alarm clock example shows the unique relationship, with respect to mode behavior, between the engaged mode and its reference value, and how this is perceived by the user. We model the display from the point of view of the user, and not just based on the machine’s output (engaged). In this respect, our modeling approach for describing the display is different than some of the modeling formalisms that were reviewed earlier. Finally, since the interplay between the exclusive mode and its reference value is the general form of any mode behavior, we have developed a format, or template, to describe it. This interplay is represented by means of a structure that

includes a mode element with its attached and concurrent reference values. We use this structure in many of the models described in this thesis.

The examples until now have focused on systems that operate in a relatively static and low-risk environment (but tell that to the person who woke up late and missed his flight). In the next section we describe a more dynamic system, the cruise control of a car. The focus will be on levels of supervisory modes.

CRUISE CONTROL

In the mode literature review we described supervisory modes as those behaviors that determine the level of involvement, in operating a system, between the human and machine. In this example of a cruise control in a modern automobile, we create a special template for describing such modes.

Cruise controls are devices employed to alleviate driver workload in maintaining speed control. Commonly, cruise controls are used while driving on highways in which the cruise speed is constant for an extended period of time. The device allows the driver to set the desired speed and engage the cruise control. The device will then maintain this speed and the driver can take his or her foot off the gas pedal. The driver can accelerate and override the speed setting to pass another car or disengage the cruise control device.

The following is description of an incident involving cruise control, that occurred to the author in Atlanta, Georgia, several years ago:

*Driving on highway 85 North during a rainy night, the traffic was slow at about 40 miles-per-hour because of the road conditions. Bored and tired, the driver engaged the cruise control by turning it ON and pressing the set-speed button. The cruise control engaged, and the car **cruised at 40 mph**. Several minutes later the rain stopped and the traffic speed increased; subsequently, the driver depressed the gas pedal to manually override the cruise control and increase the speed to 60 mph. He drove in this configuration for some 10 miles until coming to his planned exit from the highway. At this point he had completely forgotten that the cruise control was previously engaged (there was no indication in this type of car that the cruise control was on and engaged).*

*The exit ramp (North Druid Hills) initially sloped downhill and then extended uphill, ending with a curve into a busy intersection. Aware of this landscape, the driver planned to release the gas pedal and let the car glide downhill (lifting his foot from the gas pedal) and maintain a slow speed during the turn into the intersection. Initially it all worked as planned. However, once the car reached a speed of just **below 40 mph** the cruise control “kicked in.” Not expecting such a jolt, the driver lost control of the car as it sped into the intersection. Luckily, no other cars were present at this late-night hour.*

Description

The cruise control described here is installed on a popular mid-size 1985 sedan. The cruise control mechanism is operated via a pneumatic valve that feeds from the engine pressure. The cruise control interface is located on the turn signal lever (Figure 5-5).

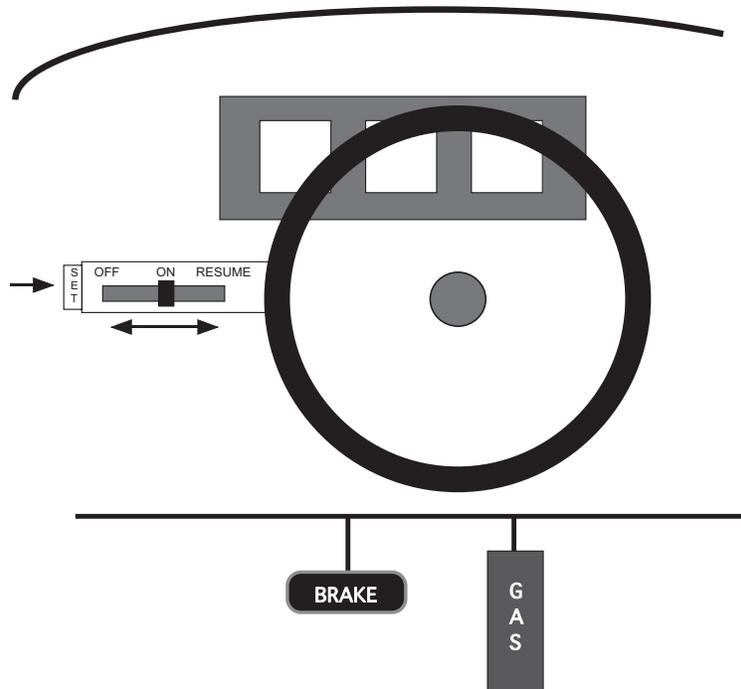


Figure 5-5. Cruise control of a car

There are two controls for this automatic device: a mode selector and a set-speed push button. The mode selector has three positions: ‘off,’ ‘on,’ and ‘resume.’ When the selector is ‘off,’ the cruise control is inactive. When it is ‘on,’ the cruise control is armed. When the selector is in ‘resume,’ the cruise control is reactivated. There are no display indications on the instrument panel about the status of the cruise control.

Model

The Ofan model contains two modules: the interface and the control mechanism (Figure 5-6). Similar to the mode switch on the clock/radio, we describe the modes of the cruise control (‘off,’ ‘on,’ and ‘resume’) via a three-state element. The ‘resume’ state is unique because the ‘resume’ detent is spring-loaded. That is, once the user selects ‘resume’ and lifts his or her finger, the switch returns to ‘on’ using its own spring-loaded power. We describe this internal transition as a broken line.

Control Mechanism

Five levels of human-machine involvement are depicted in Figure 5-6. Each level of involvement is defined as a combination of machine and human states. The supervisory levels range from fully manual to fully automatic. The levels of involvement are represented via hierarchical encapsulation of levels (substates) within larger levels (superstates).

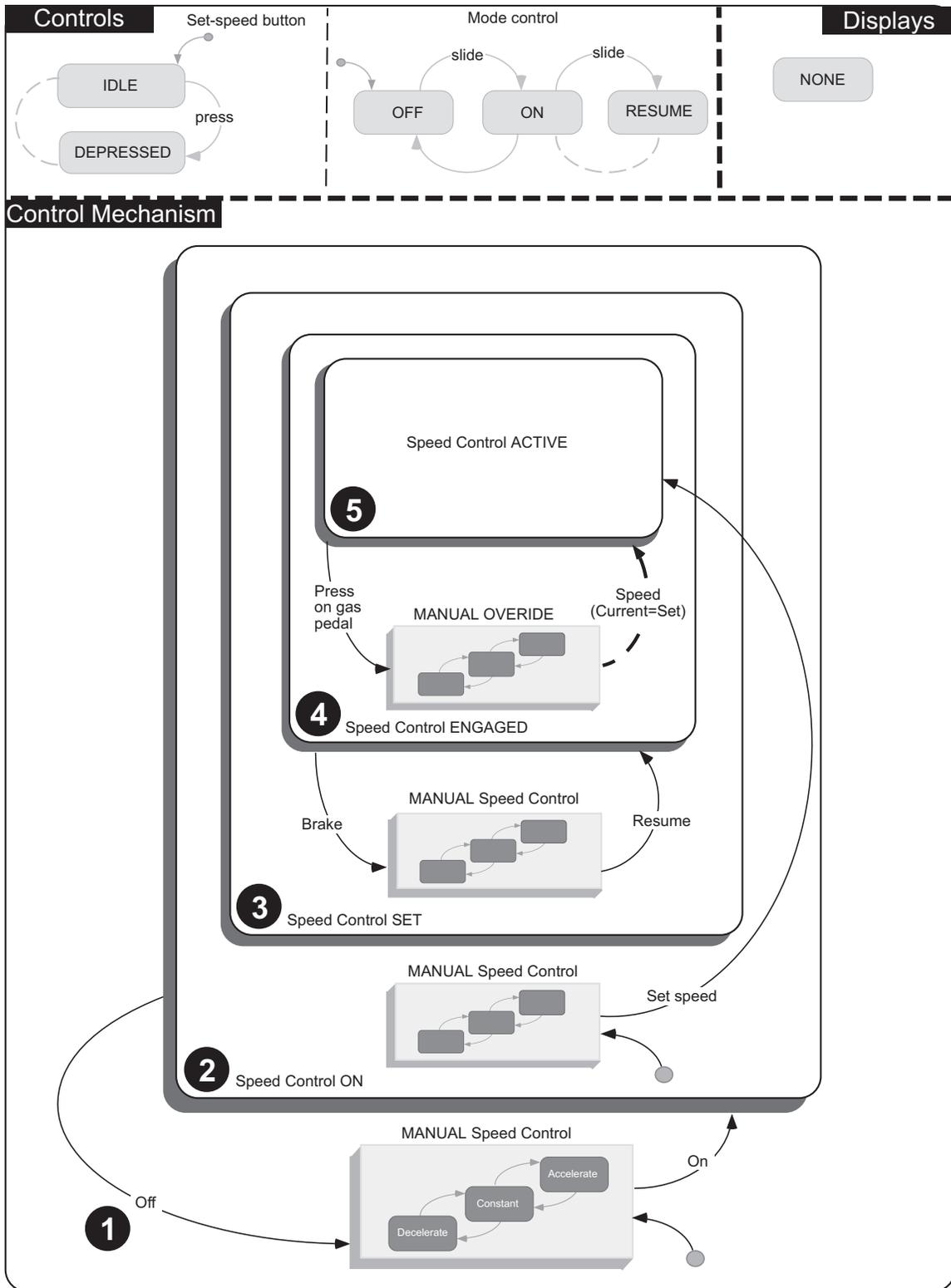


Figure 5-6. Cruise control model

Table 5-2 lists the five supervisory modes of this system. At level one ❶, the cruise control is *OFF* and the car's speed control is completely manual (*OFF / MANUAL*). At level two ❷, the cruise control is turned 'on,' but speed is still controlled manually by the driver. Note, however, that the configuration of the human-machine system has changed: It is cruise control *ON* with respect to machine state, and *MANUAL* with respect to human state (*ON / MANUAL*). In essence, the configuration of the human-machine system is a vector of these two components. Once the driver presses the 'speed-set' button, the current speed (e.g., 40 mph) becomes the set speed, the machine engages, and we jump immediately to level five ❺. At this level the speed is actively controlled by the machine and the human is idle (*ON: SPEED SET: ENGAGED: ACTIVE / IDLE*).

Table 5-2. Machine and Human Involvement States

Level	Machine	Human
❶	OFF	MANUAL
❷	ON	MANUAL
❸	ON: SPEED SET	MANUAL
❹	ON: SPEED SET: ENGAGED	MANUAL-OVERRIDE
❺	ON: SPEED SET: ENGAGED: ACTIVE	IDLE

If the driver now decides to increase speed, he or she depresses the gas pedal and manually overrides the fully automatic speed control, as depicted in Level ❹ (*ON: SPEED SET: ENGAGED / MANUAL-OVERRIDE*). Note that the cruise control is still engaged, but the speed of the vehicle is no longer actively controlled by the machine. Nevertheless, there is a lower bound on the speed (40 mph in the case of aforementioned incident) at which, when the current speed is equal to the set speed, an internal transition to the fully automatic *ACTIVE* state will occur *automatically* (depicted as a broken arc in Figure 5-6). When this happens we are back to Level ❺.

Finally, the transition to Level ❸ will occur if the brake pedal is depressed while the machine is engaged. In this case, the speed-set value is still stored in memory and the machine is *ON*, but the driver is in a completely manual mode (*ON: SPEED SET / MANUAL*). Once the 'resume' button is pressed, the machine will be re-*ENGAGED* and transition to the fully automatic *ACTIVE* state again (Level ❺).

Analysis

These five levels represent, in discrete and hierarchical steps, the continuum from manual to fully-automatic control with respect to the speed control of the car. The hierarchical structure of the system and the Statecharts formalism provide that an exit from any superstate will cause an exit from every one of its substates. For example, notice that turning the cruise control 'off' causes an immediate transition into fully manual, regardless of which level we were previously at. This hierarchical approach to describing such human-machine systems is quite different from the standard engineering approach for modeling similar systems (cf. Smith and Gerhart, 1988).

Generally speaking, in order to anticipate the next behavior of a machine, one requires three sets of information: 1) a complete and accurate model of the system in terms of states and transitions, 2) sensing of both internal and external stimuli that fire transitions between states, and 3) the ability to keep track of (or log) the system's states. In this analysis we impose a constraint that keeping track

of the system's states should be done by reference to feedback provided by a display; no memory of the system's past behavior should be required.

In attempting to understand the mode confusion and resulting error that was described in the incident, we focus on the automatic transition from *ENGAGED / MANUAL-OVERRIDE* (Level ④) to *ACTIVE / IDLE* (Level ⑤) after releasing the gas pedal and reaching the set speed. This is the only automatic transition in this system. All other transitions are initiated by neuromotor actions (e.g., depressing the brake, pressing the 'on' button).

Discussion

Again, the implicit task specifications are laid out so that the driver must know in which supervisory mode he or she will be during the next transition. When we analyze the system according to this specification, we discover that this system is ambiguous. More accurately, under the assumption that our driver has no memory of the last transition (action) taken, we find that this human-machine system is non-deterministic. That is, we can't resolve which of two different transitions may occur after the driver releases the gas pedal: 1) The cruise control system engages and activates when the car's speed reaches the set speed (40 mph), or 2) the cruise control system never engages and the car's speed continues to decline (as was expected by our driver).

We consider this situation analogous to a blind person standing on a spring board. The only way he or she can resolve the question of whether or not there is water in the pool is by jumping into it. This non-determinism, or our inability to differentiate between behaviors of the system and predict its next configuration, is one of the factors that produces mode ambiguity (Dix, 1991, chap. 6). In this case, similar to the clock/radio example, the ambiguity is due to insufficient display. The result is confusion and subsequent *mode error* —“doing the operation appropriate for one mode when in fact you are in another” (1983, p. 255). Note that the position of the control switches does not aid the driver in resolving this ambiguity. Moreover, this holds true even if he or she has an absolutely complete and accurate model of the system (i.e., complete knowledge specification)! This is because the settings of the cruise control interface are the same regardless of whether the system is at Level ② (*ON / MANUAL*) or Level ④ (*ON: SPEED SET: ENGAGED / MANUAL-OVERRIDE*). Even if the driver is looking at the control switches like a hawk, he or she will still fail to resolve between these two configurations. Given the entry assumption of no past memory, the display is insufficient for accomplishing the task specification.

Summary

Using this example, we have defined a factor that leads to mode ambiguity. In addition, we have suggested a template that will allow us to describe the various levels of human-machine involvement, and to capture the hierarchical and discrete nature of supervisory modes. We identify non-determinism as a factor that can define an insufficient display, and show that such definition is always in the context of the task specification. In addition, we also show how one can degrade the performance (specification) of the user and make some worst-case yet quite reasonable assumptions. In this particular case, we have listed the task specification, assumed the user has all the required knowledge specification about the device, but degraded the user performance specification. We have assumed the user cannot recall the last action taken (e.g., 30 minutes ago). The combination of all three specifications has allowed us to determine that because of insufficient display capability, the mode information provided to the driver is inadequate for the task (specification).

Our next example, an automatic blood pressure machine, exemplifies a human-machine system that operates in a dynamic and high-risk environment. Several levels of supervisory modes are presented, as well as some envelope protection modes. The example focuses on the relationship between modes and reference values. In particular, it shows how a reference value can cause a mode transition without the user noticing it.

AUTOMATIC BLOOD PRESSURE DEVICE

In this example we will add another module to our Ofan representation. Specifically, we add the Plant module which describes the operation of the physical components—in this case the pump of an automatic blood pressure device.

The blood pressure device is used in medical settings—both in operating rooms and wards. The device, as its name implies, is used for measuring the patient’s blood pressure. Automation is employed to alleviate the labor-intensive, time-consuming measurement and recording of the patient’s vital signs that otherwise would require the attention of medical personnel.

The manual procedure for measuring blood pressure involves pumping up a cuff, monitoring the peak vascular blood pressure (systolic), deflating the cuff, and monitoring the lowest pressure (diastolic) (see Ganong, 1979, chap. 29). This procedure takes 30 seconds to a minute, “but that is time that may well be needed to do other tasks” (Gaba, 1994). This automatic device relieves the medical provider from this procedure by automatically conducting the procedure at given pre-set intervals and recording the results. However, the process and type of measurement performed by the device is non-trivial (Ramsey, 1991), and some measurement uncertainty may occur (Weinger, Scanlon, and Miller, 1992).

Like many medical devices, the non-invasive blood pressure device has several modes of operation and reference values (e.g., interval between measurements) that need to be set by the user. This particular device, however, has a unique mode and reference value relationship that taxes the user knowledge specification and was the impetus for a recent incident. The incident occurred during an operation and is quoted from Gaba (1994):

*The patient’s blood pressure read high (“hypertension”) and the surgeon complained of profuse bleeding in the surgical field. The anesthesiologists set the device to read more frequently and became quite busy trying to lower the blood pressure to help the surgeon control the bleeding. Periodically the anesthesiologists glanced back at the blood pressure device, only to see that it still **showed an elevated blood pressure** [emphasis added]. They did not realize that it showed the same blood pressure as before and had never made another measurement.... So, for 45 minutes (sad to say) they aggressively treated the hypertension with powerful drugs that dilated blood vessels. Incidentally, the bleeding continued to be a problem until the surgeon got control of the bleeding sites. Finally it was discovered that there had not been a recent measurement of blood pressure, which was in fact **found to be very low** [emphasis added]. Things got sorted out, and fortunately there was no lasting harm to the patient.*

Description

This particular automatic blood pressure device has several components, each one with its own set of modes. For describing this particular incident, we focus on the normal/abnormal modes and the timer interval. Figure 5-7 is a line drawing of the device.

The machine has two basic modes of operation with respect to the measurement cycle: “Manual” (the symbol ‘OFF’ is displayed), in which the user presses a button and the machine makes one cycle and then goes idle; and “Active,” in which the machine sequences its measurement cycles according to what the user sets in the timer interval component.

The timer interval component has three modes: “Off,” “Continuous,” and “Interval.” The “Off” mode (displayed ‘OFF’ on the screen) is the default. To activate the measurement cycle in this mode, the user must press the ‘start’ button every time he or she wants a reading. “Continuous” commands one measurement immediately after the other for a maximum period of five minutes. In “Interval” mode, nine different interval values can be entered (1, 2.5, 5, 10, 15, 20, 30, 45, 60, and 120 minutes). The machine automatically schedules measurement cycles accordingly. These time intervals are discrete reference values of the fully automatic (“Active”) mode.

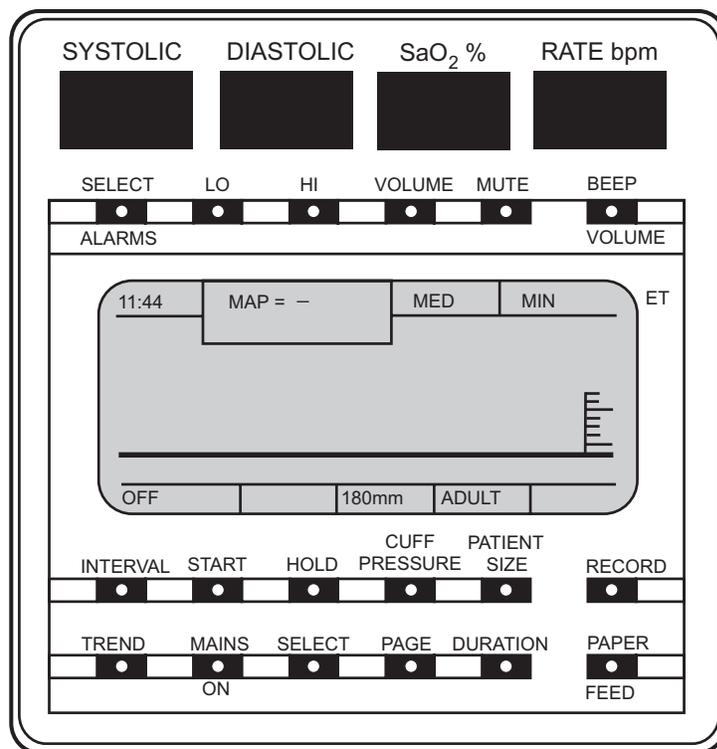


Figure 5-7. Line drawing of the automatic blood pressure device

In addition to the basic modes (“Manual” and “Active”), two other modes are pertinent to our discussion: “Hold” and “Retry.” “Hold” is used to terminate a measurement cycle (the cuff is deflated) and suspends any scheduled (interval) cycles. “Retry” is a type of protection mode that will automatically initiate a measurement cycle when the current cycle is determined to be inadequate. Three such retries will be attempted before the device will abort and an ‘UNABLE TO MEASURE’ message will be displayed on the screen. Two other indications of system state are also provided through the interface: ‘ELAPSED TIME’ and ‘TIME.’ Elapsed time indicates the

(relative) elapsed time since the completion of the last measurement. The (absolute) time indicator is updated each minute.

Model

The model describes three modules: interface, control mechanism, and the plant. The last module, plant, describes the physical components of the device. In this case we detail the behavior of the pump assembly and sensors. Figures 5-8 and 5-9 depict the model.

Interface

Two display elements of the automatic blood pressure machine are represented in Figure 5-8: the systolic/diastolic value indicator, and the mode display. Systolic/diastolic values are presented on the top portion of the device. The display gets updated following a measurement cycle. It retains the last value and has no feature to indicate the 'age' of the measurement. The mode display indicates both the engaged mode (not necessarily the active mode, as will be discussed shortly) and reference values selected while in the "Interval" mode. Three control buttons are modeled here: these are 'interval,' 'start,' and 'hold.' These are push buttons and are all spring-loaded. Once the user depresses the button, this event is broadcast and fires a transition in the control mechanism module.

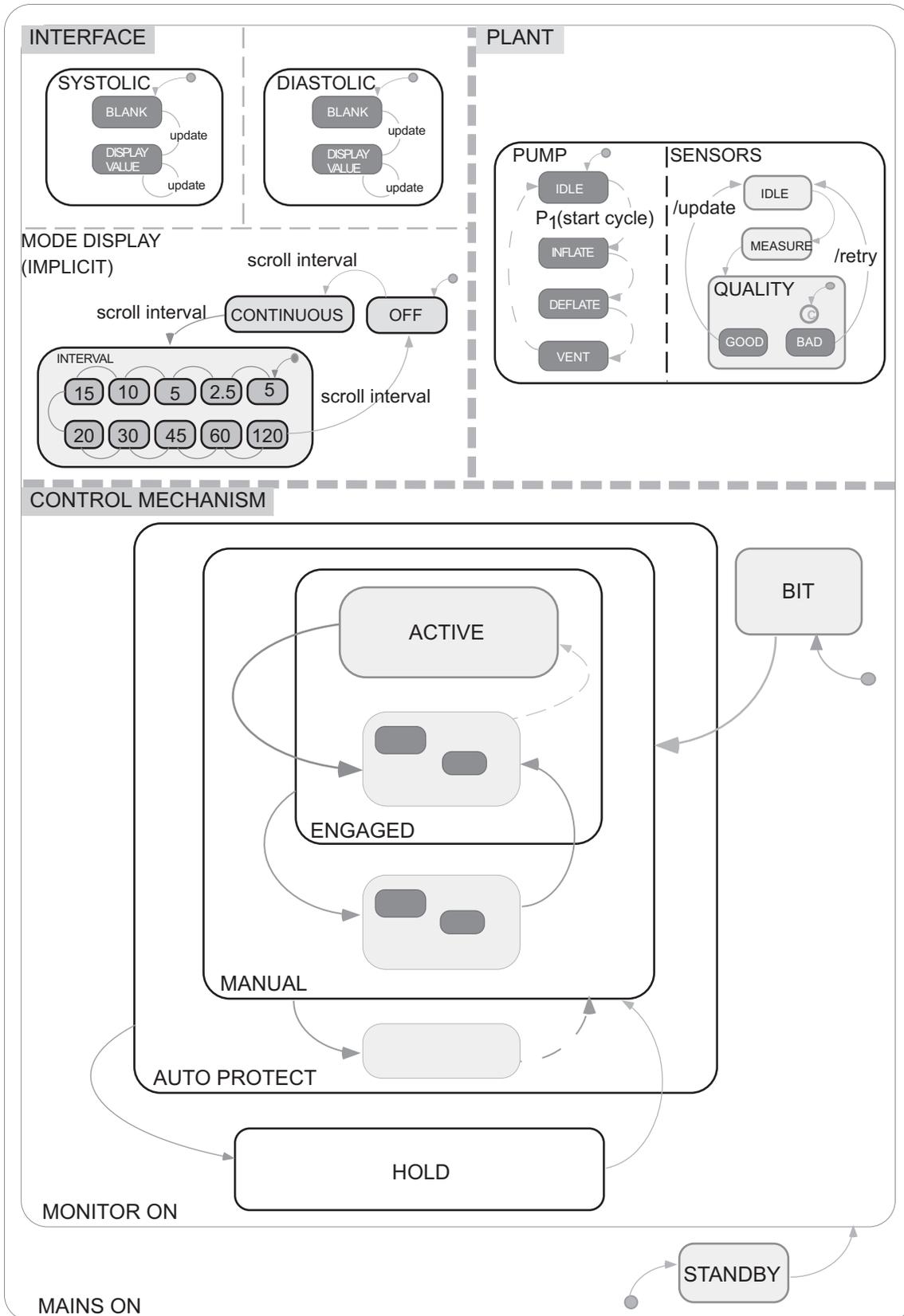


Figure 5-8. Model of the blood pressure device

Control Mechanism

Two levels of supervisory modes are represented in Figure 5-9: *ON* and *ACTIVE*. Two other states, *AUTO PROTECT* and *HOLD*, are external to normal operations and will be discussed later. With respect to the sequence of operation, the device initially transitions to *STANDBY*, and after the monitor is turned ‘on,’ a built-in-test (BIT) is performed. Once the device passes the test, the *MONITOR ON* superstate is active and the default state is *IDLE*.

Once the device is in *IDLE* and ready to be used, pressing the ‘start’ button creates a transition to *START CYCLE*. This event triggers the beginning of the measurement cycle. Pressing the ‘interval’ (mode) button and subsequently selecting an interval value (1–120 minutes) fires the transition to the *ENGAGED* superstate.

Initially, the *MANUAL OVERRIDE* and its *IDLE* substate are occupied. To reach the fully automatic *ACTIVE* mode, the user must hit the ‘start’ button, which initiates a measurement cycle. Once this is done an automatic transition to the *ACTIVE* state occurs. In this state the measurement cycles are initiated and scheduled according to the timed interval value (1–120) and all is well. The user may request an immediate measurement by hitting the ‘start’ button again (this feature is somewhat similar to the manual override in the cruise control example). This will initiate a start cycle and once done, the automatic transition back to *ACTIVE* will take place (note that the same ‘start’ button fires three different transitions, depending on the mode configuration).

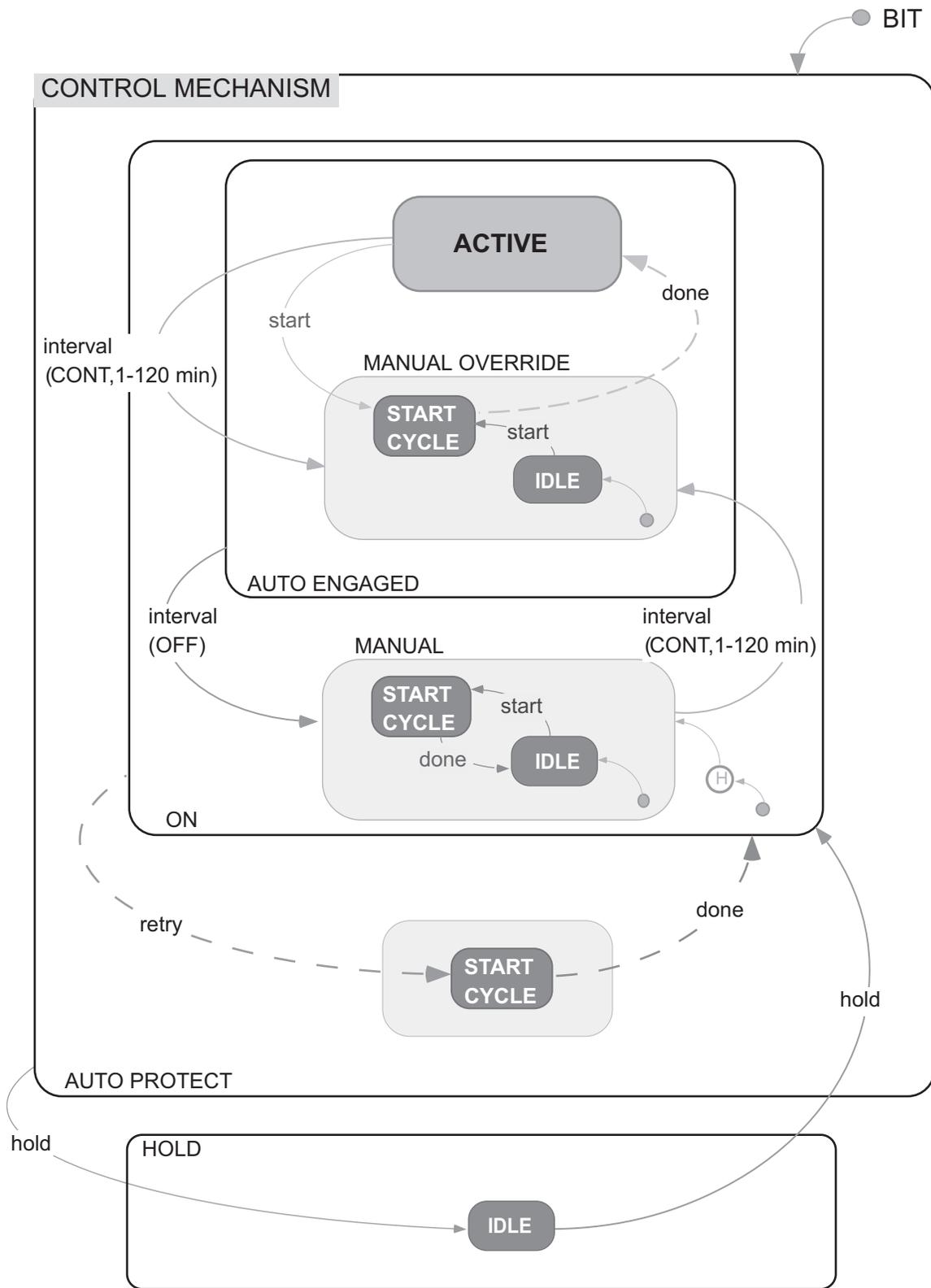


Figure 5-9. Model of the control mechanism (blood pressure device)

Now, if the user wishes to change the interval (reference value) setting (say from five-minute intervals to one-minute intervals), pressing the ‘interval’ button will scroll among the possible reference values. This event (changing the reference value) will cause a transition out of *ACTIVE* into *MANUAL OVERRIDE*, with an entry into *IDLE*. Only a subsequent press on the ‘start’ button will activate the device. The display, however, indicates that a reference value is set and the user may assume the “Active” mode is engaged, when in fact the device is in *IDLE*.

The *AUTO PROTECT* superstate encapsulates all the existing normal states of the device. Once a measurement is inadequate, a transition into “Retry” mode takes place automatically. This mode is an automatic envelop protection that attempts another cycle. Up to three such cycles can occur successively. Once “Retry” is disengaged (automatically), the previous mode is engaged. The history (H) feature of the Statecharts formalism allows for recall of past states—*ON* or *ENGAGE*. Similarly, if and when the user hits the ‘hold’ button, the “Hold” mode is engaged and the device is held at *IDLE*. Subsequent pressing on this button will disengage “hold” and engage the previous mode.

Plant

In the preceding example, we did not model the actual physical component of the device. There was no need to do so, as our interest was in the control mechanism and its behavior. In this example, we broaden our representation and add the Plant module. The blood pressure has two concurrent processes that are of interest here: the pump assembly and the sensors. The pump has four state of operation (*IDLE*, *INFLATE*, *DEFLATE*, *VENT*), and the transition out of *IDLE* is predicated upon the control mechanism being in a *START CYCLE* state. The sensors measure the quality of the cycle. If the quality is good, then an update of the interface (e.g., systolic and diastolic indicators) occurs. If the measurement quality is inadequate, the device automatically goes into “Retry” mode.

Analysis

The breakdown between the anesthesiologists and the device occurred in two steps. First, the anesthesiologists changed the interval (reference) value and either forgot, or did not know, that a subsequent press on ‘start’ was necessary to activate this mode. Then, after the initial breakdown, the interface did not allude to the actual status of the device. Several design features contributed to the occurrence of this incident:

The first breakdown occurred because of the unique behavior of the control mechanism. As can be seen in Figure 5-9, changes in reference values cause the device to exit the *ACTIVE* state and re-enter the *MANUAL OVERRIDE* state and then land in *IDLE*. In other words, a change in reference value causes a significant change in mode.

The second breakdown occurred when the anesthesiology team did not realize, for quite a long time, that the device was in *IDLE*. The interface, indeed, does not portray the actual mode in this situation. What it portrayed instead was the interval value re-entered by the team (following the hypertension and the profuse bleeding). So there are two very different configurations of the device with the same display status (Table 5-3).

Table 5-3. State Configuration

Initial State	User Actions (events)	Display	End State
MANUAL	Set interval, press 'start'	Interval value	ACTIVE
ACTIVE	Set interval	Interval value	IDLE

The only (indirect) indication that could cue the team the device was indeed in *IDLE* configuration was the 'TIMER INTERVAL' indicator on the display. The timer shows the time elapsed since the last measurement. And in this particular case, the elapsed time would increase without being reset to 0 as a consequence of a new cycle. But again, this is only an indirect cue that requires some additional mental processing (Time Elapsed < Interval Set value). Furthermore, the systolic and diastolic values present on the screen may have cued the team that the device was working.

Discussion

Three main factors contributed to this mode ambiguity: knowledge specification, task specification, and insufficient display.

With respect to knowledge specification, the interactions between the reference values and modes are far from trivial. This is particularly true of the engagement of "Active" mode and its reference value, and of the updating of intervals. The (user's) task specifications are twofold: (1) Changing a mode (and an initial reference value setting); (2) changing only a reference value. Although the tasks' specifications are very different, the actions the user must take are identical. This, again, is because the mode and reference value settings are combined: i.e., initially setting a reference value implies both a mode change and reference value change, while updating a setting does not. Yet in both cases, the user has to set a reference value and also press the 'start' button. It appears that this unique interrelationship is not easy to maintain in both factual and procedural memory.

In the light switch example discussed in Chapter IV, we mentioned the topic of population stereotypes. While the topic has been investigated with respect to control levers and rotary switches, not much is known about how population stereotypes affect user interaction with computerized devices, such as consumer electronics and electronic devices. It appears, however, based on observations of college students developing human-machine interfaces, that such stereotypes may exist (A. Kirlik, personal communication, June, 1996). One such stereotype involves modes and associated reference values. The user may assume that hitting the 'start' button implies a mode change, and that setting the interval (while in automatic mode) implies only a reference value change. Many consumer electronic devices such as microwaves, electronic watches, and even complex Flight Management Systems work that way.

The discrepancy between the user's expectation ("Active" mode and one-minute intervals) and the device's actual configuration (idle) is further compounded by a lack of mode information. Based on the task specification, we know the interface should provide both mode and reference value indications. The interface, however, supplies only reference value information. As a consequence, the same display (one-minute interval) may imply two (radically) different mode configurations—the device is working or the device is in idle. Unless the user knows what the sequence of actions was, he or she cannot immediately resolve between the two configurations. In this respect, the insufficiency of this display is similar to the cruise control example.

Summary

In this particular example we introduce, in a more complex device and environment, the relationship between the two elements of mode behavior: modes and reference values. While these two elements exist in any machine, how this relationship is interrelated and displayed is not a trivial matter. Later in the thesis (Chapters VI and VII) we discuss this relationship in detail. In this specific example, the combination of a unique mode and reference-value relationship, violation of a common stereotype, and insufficient display contributed to a life-threatening error. For any device, state dependency and mode-reference value relationships are far from trivial and tax the user's knowledge. Henceforth, a display that shows this relationship should be in place. In particular, a salient display is a necessity when these two features are 'fighting' against a population stereotype.

The example also elaborates on the structure of modes. In addition to the hierarchy of supervisory (manual, automatic, manual override) modes, we also represented 'envelope' modes (the "Retry" and "Hold" modes). The former is external to the supervisory modes and engages automatically when a measurement cycle is inadequate. The latter is a global interrupt that will stop any measurement cycle when the user requests it (e.g., due to an emergency).

In this and the preceding examples, we focused on the interface, the control mechanism, and the physical plant. In the next example we broaden the Ofan framework to include the user task module, which describes the task specification, and a module describing the operating environment. The Ofan framework will now be presented in its entirety.

MAP DISPLAY

In this example we describe a model of an interface mode and its associated reference values. In particular, we model the display of a moving map used for navigation of an aircraft. We use this example to add another modeling template—one which we employ for representing the user tasks. We present and discuss the theoretical concepts behind this template in the model section of this example.

The domain from which we take this example is commercial aviation. In this domain there is much emphasis on use of navigation aids for piloting the aircraft along its route and toward its destination airport. Modern aircraft cockpits incorporate a map display which shows the location of the aircraft in space, and superimposes on the display other information such as airport location, navigational waypoints, and weather information. The aircraft symbol on the map display stays in one place, and the scenery moves as the aircraft progresses along its intended route of flight—hence the name "moving map display" (Wickens, 1992, chap. 4).

This example was documented during cockpit observations. The pilots complained that there was "something weird" about changing the range and format of the moving map display. During an approach to an airport, this display is monitored by the pilots in order to assess the relationship between the aircraft ("ownship") and the location of other objects within the environment, such as airports and navigation waypoints. According to standard operating procedures, the crew should also monitor this display to assess the relationship between ownship location and electronic navigation aids such as the ILS (Instrument Landing System) and VOR (Very High Frequency Omnidirectional Radio). After an examination of the controls and the display, it was evident that the source of the problem arose from an unexpected interaction between the interface mode of the map display and the range (a reference value).

Description

The map mode and range are controlled by two rotary knobs located in front of the pilot (Figure 5-10). The range-selector knob allows the pilot to select among six range options (320 nautical miles [nm], 160 nm, 80 nm, 40 nm, 20 nm, and 10 nm). The (interface) mode-selector knob allows the pilot to switch among five different map formats: 'PLAN,' 'ARC,' 'Navigation' ('NAV'), 'VOR,' and 'ILS.' The 'PLAN' format is mainly used for reviewing the flight plan. The 'ARC' format presents a 90-degree arc view emanating from the ownship symbol at the bottom of the screen and is the most frequently used interface mode. The Navigation, VOR, and ILS modes all present a 360-degree (compass rose) view around the ownship symbol (now located at the center of the display); these formats are normally used during an approach.

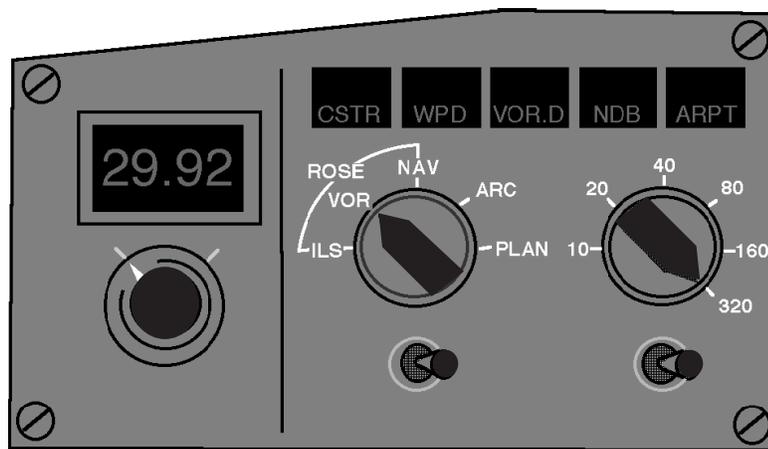


Figure 5-10. Mode- and range-selector knobs

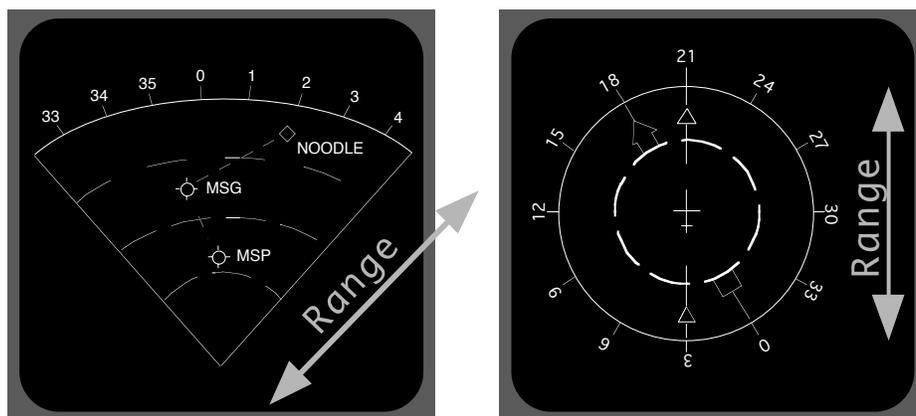


Figure 5-11. 'ARC' and 'ROSE' map formats

When in ‘ARC’ format, the range selected with the range-selector knob corresponds to the distance from the ownship symbol to the edge of the arc. When in any of the three (compass) ‘ROSE’ modes (NAV, VOR, and ILS), the selected range is not from the ownship symbol to the edge—it is from the upper edge of the arc to the lower edge (Figure 5-11). Therefore, when the pilot switches from ‘ARC’ to any of the three ‘ROSE’ formats, objects at the top of the display disappear (those that are more than half-way distant from ownship).

For example, if the active format is ‘ARC,’ the range selected is 20 nm, the destination airport is 15 miles, and the pilot switches to ‘NAV’ format—the destination airport symbol disappears! Manipulation of either the range-selector switch (to 40 nm) or display format switch (back to ‘ARC’ format) will bring the airport symbol back to view. Two additional factors augment this problem: The transition from ‘ARC’ to any of the ‘ROSE’ formats usually occurs during one of the busiest periods in the flight—the approach to the airport. Also, after switching the range or format, it takes 1–2 seconds for the display to update itself (in the meantime it is blank). There is, however, one advantage to this design—the display resolution stays constant regardless of format changes.

Model

Figure 5-12 depicts the Ofan representation of this system. It contains four modules: environment, user task, interface, and the mode switching control mechanism.

Environment

The *Environment* module (top) has three concurrent components. The environment in which the aircraft is operating can be air (aircraft flying) or on the ground. Navigation aids such as ILS or VOR can be received, or not, by the aircraft sensors. The distance between the aircraft and the airport dynamically changes as the flight progresses.

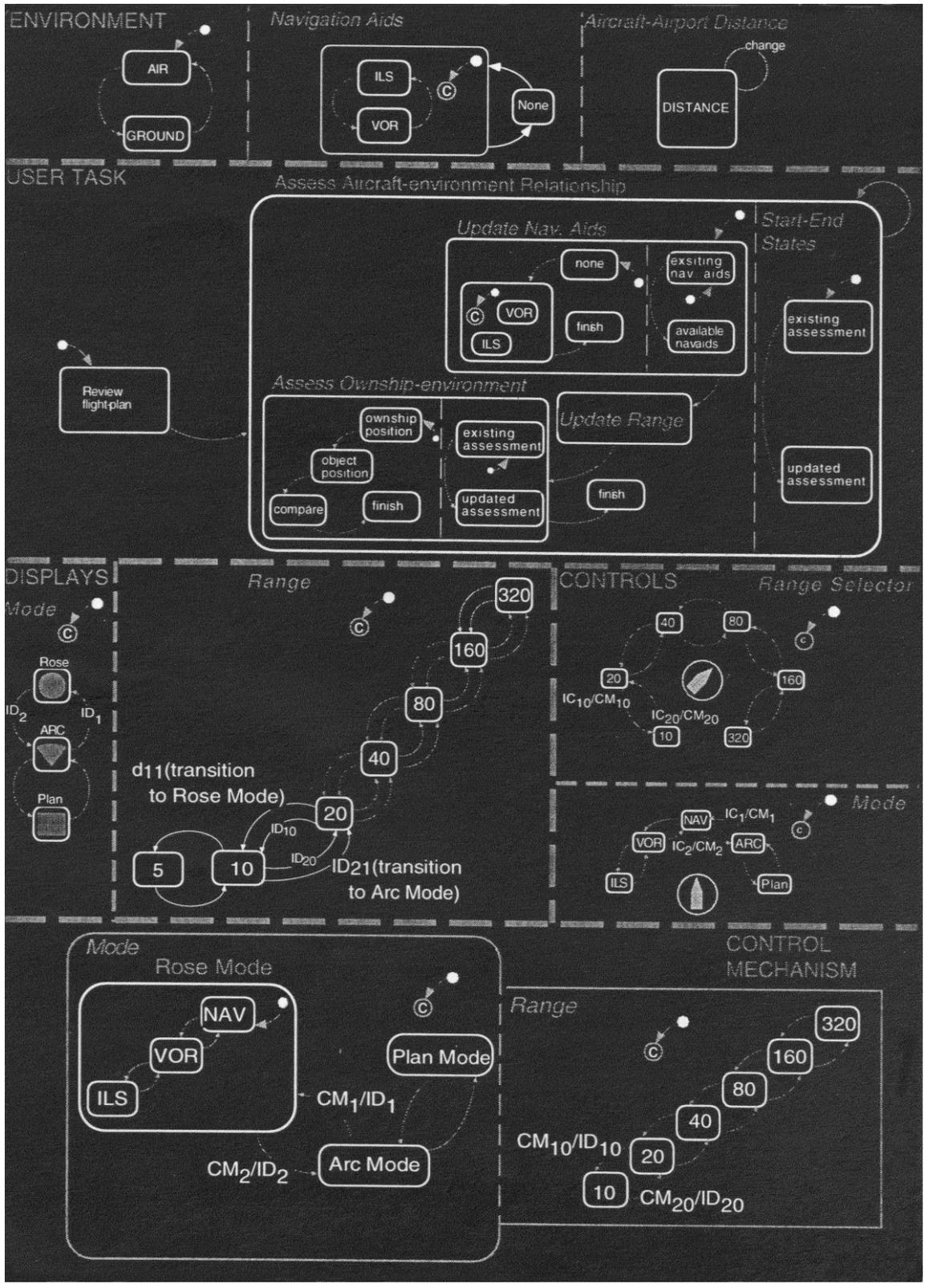


Figure 5-12. Model of the map display

User Task

In this subsection we first introduce the theoretical concepts behind the representation of user tasks and then describe the specific module of this example.

The user task model defines the user task specification. A task, according to Drury, Paramore, Van Cott, Grey, and Corlett (1987) is a set of human actions that contribute to a specific functional objective and ultimately to the output goal of the system (p. 373). To model tasks we developed a format, or template, that can be hierarchically and consistently employed for procedural types of tasks. The template is made up of a superstate and two concurrent processes: the start and end states of the task and the necessary procedural steps.

Drury, et al. (1987) list several important characteristics of a task: Each task has a starting point that can be identified as a stimulus or cue for task initiation. Each task has a stopping point that occurs when information or feedback is received that the objective of the task has been completed (p. 374). Similarly, McFarland and Bossert (1993, chap. 8) use the same concept of start and end states in their approach for modeling human (or robot) tasks. In their approach, a task is accomplished when a particular state is reached as the result of a given behavior. How the task is accomplished is a question that should be distinct from questions about the definition of a task. Tools are often seen as aids in accomplishing tasks, but we must be careful to define tools independently of the tasks to which they relate (p. 191).

Once the start and end states are defined, selection of tools and procedures to achieve the end state can take place. This is an important advantage for design as it decouples the tools from the task during the design process. In addition, such a structure of start-end states and an associated procedure is amenable to a formal test such as reachability analysis.

Our template, therefore, contains the definition of the start and end state of the task. Initially (by default) the start state is active. Concurrently we model the procedural steps necessary to achieve the end state (Bainbridge, 1993). Once the steps of the procedure (defined as states) are completed, a transition to an arbitrary finish state takes place. When the transition occurs, it fires a transition to the end state. This, in turn, fires a global transition out of the task superstate to another.

Using this template we describe the user tasks in our Ofan framework. Since each task is made up of several procedural steps (subtasks), and each of its subtasks can also be defined with a start-end state and a related procedure, the hierarchical decomposition can continue to whatever level deemed necessary. With these theoretical concepts in mind, we shall now describe the user task module of this example.

The *User Tasks* module in Figure 5-12 describes two (super) tasks performed by the crew—(1) review of the planned route, and (2) assessment of the aircraft-environment relationship. Both activities are depicted as states. The assessment task has a start state—*EXISTING ASSESSMENT*—and an end state—*UPDATED ASSESSMENT*. The end state signals the closure on this task. The procedure to fulfill the task specification involves three subtasks: The first subtask, *UPDATE NAVIGATIONAL AIDS*, has also start and end states. The procedure here involves switching between the various aids. The second subtask, *UPDATE RANGE*, starts in state *INAPPROPRIATE RANGE* and is brought to an end when *APPROPRIATE RANGE* is active (not depicted). This is achieved by assessing the current aircraft-airport range and then updating the range selector accordingly. The third subtask, *ASSESS OWNERSHIP-ENVIRONMENT*, has a start state—*EXISTING ASSESSMENT*—and an end state—*UPDATED ASSESSMENT*. The end state is achieved after the three-step procedure is complete. The procedure involves identifying ownship

position, identifying other objects' positions, and then comparing between the two. At this point, the assessment of the aircraft-environment relationship task has ended and a transition to the *FINISH* state takes place. As a consequence the transition from the start state—*EXISTING ASSESSMENT*—and an end state—*UPDATED ASSESSMENT* takes place. Once this task is completed, or several minutes afterward, the pilot will conduct another assessment, and so on.

The module provides us with three types of information. One is the task specification—what the user is supposed to do (specified by start and end states). The next is the procedure for achieving this task specification. (Once the procedure is specified, that in turn becomes the more detailed task specification, and this embedding can go on and on.) Third is the type of information required to go through the procedure.

Interface

The *Controls* submodule in Figure 5-12 describes each knob (range and format) as concurrent superstates, each one with its own set of substates. Transitions from one state to another are initiated by rotating the knob. The triggering events for these transitions come from the *User Tasks* module (*UPDATE RANGE* subtask). The *Interface* submodule also contains the display element—this, however, will be described later.

Control Mechanism

The *Control Mechanism* module describes the interface modes and the map ranges. We use the same template for mode and reference value as was described in the alarm clock example. Transitions between substates are triggered by events (rotating the knob) in the *Controls* module. For example, switching the range-selector knob from 20 nm to 10 nm (depicted as transition IC_{10}) triggers an event in the *Control Mechanism* module (CM_{10})—this relationship is depicted as IC_{10}/CM_{10} .

Closing the information flow loop is the *Displays* element of the *Interface* module. Again the same structure is repeated: two concurrent superstates, and transitions that are triggered by transitions in the *Control Mechanism* module. For example, event CM_{10} in the *Control Mechanism* module triggers a transition (ID_{10}) that switches the display range from 20 nm to 10 nm—this relationship is depicted as CM_{10}/ID_{10} . Another element in specifying transitions is a parenthesized condition or a guard, e.g., $ID_{11}(\textit{transition to Rose Mode})$ —indicating that transition ID_{11} will occur only when the transition to *Rose Mode* state has fired.

Analysis

The model shows the progression of events in the environment, user task, interface, and control mechanism. Events within a given module (e.g., the *Control Mechanism*) are mostly dependent upon events occurring in other modules (e.g., *Interface*). That is, any selection of the mode and range in the *Interface* module directly affects the *Control Mechanism* in the computer element that generates the map and its symbols. The analysis described here focuses, in particular, on the interface.

Similar to the alarm clock example, we model here the interface as it is perceived by the user. A requirement for such an approach is a well-prescribed task and knowledge specification. Task specification, as mentioned earlier, includes two elements: the start and end states (mission objectives), and the required procedural steps to reach the end.

Based on a task analysis, the pertinent tasks and information requirements were listed and modeled in the User Tasks module. Specifically, we focused on the assessment of the ownship-environment relationship. When this was done, a unique structure of the mode and reference value was revealed. In comparison to the Control Mechanism module, the Interface module is somewhat different—there are two arcs for transitioning from one range to another.

One arc corresponds to a transition in the Control Mechanism module which was earlier initiated by rotating the range knob. This transition is obvious and consistent with similar transitions in other modules. The other arc—which is unique only to the Interface module—is contingent on transitioning to the ‘Rose’ format. This dependency, however, cannot be intuitively grasped from the control panel layout. Another interesting aspect of this dependency is the ‘birth’ of a new range—5 nm. This range, which is not specified on the range-selector knob, arises from the interaction between the 10-nm range and ‘Rose’ format. This ‘new’ range appears, however, only when we model the display from the point of view of the pilot’s task specifications.

Discussion

The pilot’s task (as modeled in the User Tasks module) is to assess the relation between the ownship and objects in the environment (e.g., airport). That is one specification of the pilot’s monitoring task. However, during the transition between ‘ARC’ and (any of the) ‘Rose’ modes, the distance between the ownship and objects in the environment changes without the pilot requesting such a change. All the pilot wants to do is to change the display format—not the range.

The problem is detected only when we model the interface from the point of view of the user. When this is done, we realize there is a complex interaction between modes (in particular ‘Rose’) and their reference (range) values. Similar to the blood pressure device, there are two pieces of information that contribute to the mode ambiguity. First, the information about this interaction taxes the user’s knowledge specification. (Although this relationship is actually mentioned in the Flight Manual, none of the pilots remembered this piece of information). Also, the display does not allude in any way to this unique interaction between modes and reference values.

Summary

Using this example we introduced two additional modules (User Tasks and Environment) of the Ofan framework. We noted how the modules are all interrelated to form the complete environment-human-machine system. Events in the environment prompt events on the part of the user, who in turn manipulates the machine. The example also alluded to the important relationship between the task specification (as modeled in the User Tasks module) and the information presented on the display. We were able to show that changes in display format affect the range information (between ownship and object) required by the pilot, under a certain circumstance. This complex interaction between display format and display range translates into mode ambiguity. The mode ambiguity existed because the pilots were not aware of this unique interrelationship between modes and range (i.e., a limitation on knowledge specification). Moreover, the layout of the interface controls did not aid them in noting this relationship. In our Ofan framework this problematic design feature is detected by noting the two transitions to the same state: one transition is consistent with the layout of the interface, while the other is not.

CHAPTER SUMMARY

In this chapter we introduced the ‘Ofan’ modeling framework by applying it to five case studies. The case studies ranged from devices that have a simple interface and behavior, and operate in a

relatively static environment (cordless phone, electronic clock/radio), to devices that operate in a dynamic environment (cruise control) and have more modes and ‘envelope protection’ modes (automatic blood pressure device). We find that mode problems exist in a variety of systems from different domains, and are not just unique to high-technology systems (Andre and Degani, 1996). The examples also allowed us to develop three modeling templates to describe modes and human interaction with them. We provided one template to deal with mode and reference value interaction, and another to deal with supervisory modes. Finally, we provided a template for describing task and procedural knowledge specification. This third template is founded on concepts from task analysis. The templates are based on our theoretical understanding of the general form of modes, levels of human-machine involvement, and modern approaches of task analysis.

The Ofan framework and the Statecharts language allowed us to visualize the behavior of the machine—its interface, control mechanism, and plant. We could finally ‘see’ the structure of modes within the device. In addition, the focus on examples in which known mode ambiguity exists allowed us to identify (in a post-hoc fashion) the types of design features that lead to mode ambiguity. In doing this, we have attained our first objective, which was the development of a representation to describe human-interaction with modes. We then use the framework it to identify the features that contribute to mode ambiguity.

CATEGORIZATION OF MODE FEATURES

INTRODUCTION

In this chapter we launch an attempt into further understanding the features that contribute to mode ambiguity. It is also our intention to be proactive about this problem—to provide the designer with several design specific features that contribute to mode ambiguity.

Based on some twenty human-machine systems that we have modeled in the course of this research, we have identified several common features that contribute to mode ambiguity. We attempt here to abstract and classify these features. We call these features *structural elements*, in the same sense this term is used in architecture to describe a unique feature of a building that requires special attention. The classification scheme suggested here is not complete—it is based only on features that have been already observed. It is a set of heuristics in lieu of a more complete theory of mode ambiguity, and provides only an interim step in our understanding of the problem. Five different categories of structural elements are described below:

- Two Plant States—One Display State
- Hidden Transition
- Circular Mode Transition
- Mode-Reference Value Interaction
- Mode-Mode Interaction

Most of the structural elements described here have a set of subcategories. Interestingly, some of the subcategories are similar. Specifically, the subcategories relate to the type of transition that takes place in a given structural element—i.e., manual, automatic, and default.

In the following sections we introduce the five structural elements and their related subcategories. For each subcategory we describe an example and provide an abstract representation (model) of it.

TWO PLANT STATES—ONE DISPLAY STATE

This structural element exemplifies a rather common situation in which the interface is under-specified. That is, given the task specification, the interface does not provide all the information about the next state configuration. The structural element has three subcategories, depending on the type of transition between states: (a) manual, (b) automatic, and (c) default.

Manual Transition

In this subcategory, the transition between the modes (states) is manual. That is, the user manually (external to the machine) requests the transition by pressing a button, setting a switch, or using some other method of initiating a transition.

Example

The remote control illustrated in Figure 6-1 is an example of a system that harbors this structural element (Andre and Degani, 1996). This is a typical “universal” remote control that can operate any of four entertainment systems: the TV, the stereo (audio), the VCR, and the cable TV box. Imagine that you want to watch a movie on your VCR. You press the ‘VCR’ button, then press ‘play’ to start the tape. While watching the movie, you get an important phone call and rush to mute the TV volume using the remote. You press the ‘mute’ button, but nothing happens. Why? Since you are still in the “VCR” mode, the remote control does not respond to “TV” commands, Mute being one of them. In order to mute the volume, you must first press the “TV” mode button and then press the ‘mute’ button.

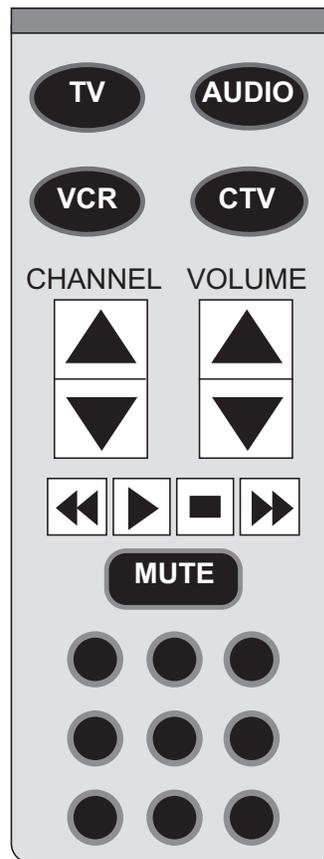


Figure 6-1. A “universal” remote control

But how can the current mode be determined? The remote control device doesn’t provide the answer since the buttons do not change position to reflect their state, nor is there a display to indicate this. In essence, the display is blank. Instead, the user has to keep a mental log of past mode button presses. If we assume the user doesn’t remember the mode last pressed (a worst-case but common scenario), the user cannot infer the current state of the system. Therefore, he or she will not be able to predict the effect of pressing the ‘mute’ button.

Representation

The following example of the VCR feature is mapped here to an abstract representation (Figure 6-2). The system has two modes: A and B.

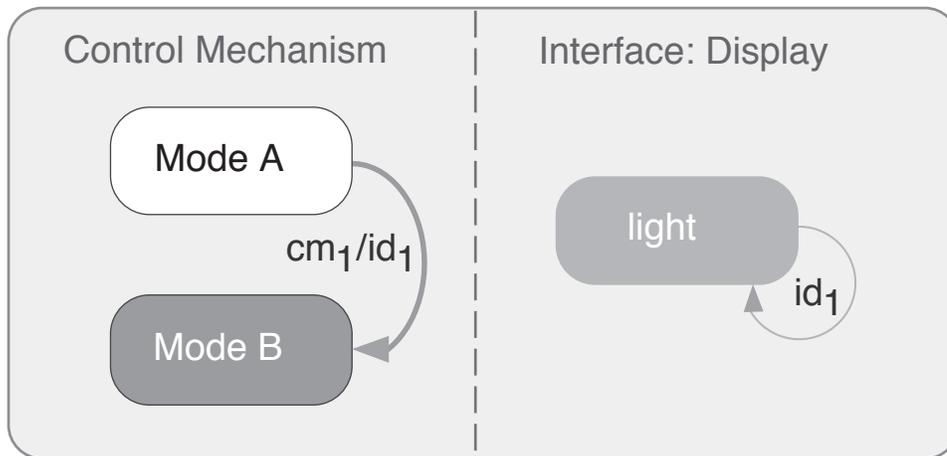


Figure 6-2. Two plant states—one display state (manual transition)

Initially the system is in “Mode A.” The user *manually* requests a transition from mode A to B, for example. When this event, CM_1 , occurs, the control mechanism transitions to “Mode B.” Event CM_1 also fires event ID_1 , but ID_1 is a loop transition. As a consequence, the display does not change its state to annunciate this mode transition. Feedback of the current mode (B) status is identical to the previous mode (A) status. By just looking at the display, the user cannot resolve whether the system is in mode A or B. However, the ambiguity can be resolved if the user remembers or keeps a log of his or her (past) actions. Note that the D_1 transition loop is a general case. There could be a special (private) case with no transition loop at all, in which case the same problem will still arise.

Automatic Transition

In this subcategory, the transition between the modes is automatic. That is, the plant automatically (internally) triggers the transition between the modes. The events that trigger the transition are fully specified by the designer, yet are predicated upon some internal events in the machine.

Example

This subcategory is one way to represent the cruise control example described earlier. There are two problems here: current and next-state transition. The user (driver in this case) cannot resolve, given the state of the controls and the display, whether the cruise control is engaged in the background, or the cruise control is just armed (but not engaged). This ambiguity becomes critical when we consider the next-state configuration of the cruise control. That is, performing a similar action will lead (transition) to two different state configurations. Specifically, lifting the driver’s foot off the pedal can either: (1) initiate an automatic transition (when the current speed is equivalent to the set speed), or (2) reduce speed to zero.

Representation

The system has two modes, A and B (Figure 6-3). Initially the system is in mode A.

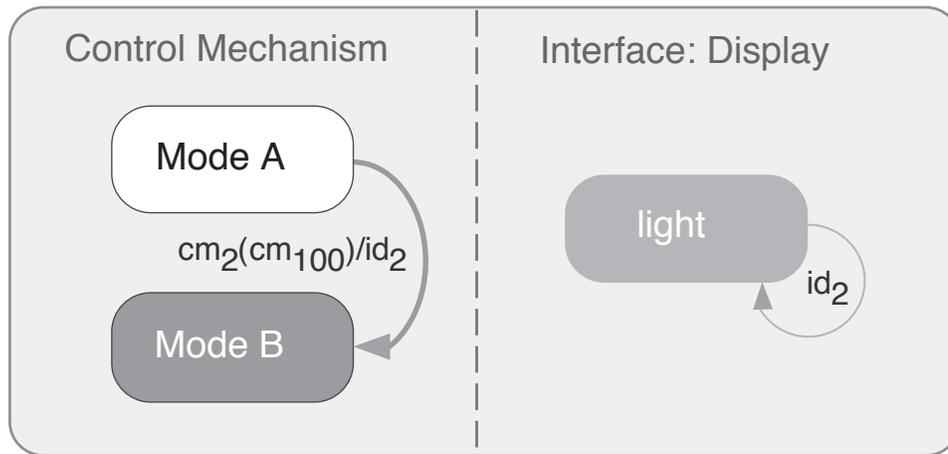


Figure 6-3. Two plant states—one display state (automatic transition)

When event CM_{100} has taken place somewhere in the network (a guard), event CM_2 is fired and the plant automatically transitions from mode A to B. Event CM_2 also fires event ID_2 , but ID_2 is again a loop transition. As a consequence, the display does not change its state to announce that B is the current mode. By just looking at the display, the operator cannot resolve whether the system is in mode A or B. This subcategory (automatic transition) is more problematic than the previous one (manual), because the mode change occurred automatically. In order to anticipate this transition, the user must (1) have a complete factual knowledge specification of the device's behavior (all the transition and conditions), and (2) independently sense the occurrence of event CM_{100} which initiates this step.

Default Transition

In this subcategory, the transition between the modes is a default one. Default entries exist in situations where the designer wants to set the parameters/states in a certain way, given an entry into some (super) state. Default transitions are automatic in the sense that the user has no direct control over their occurrence (firing)—they are internal transitions of the machine. Although existing in many so-called “static” systems (e.g., word processors, electronic watches), default transitions can be viewed as dynamic; the system does not wait for the user to make a transition—it makes the transition on its own.

Example

For an example of a default transition that gives the impression of being dynamic (and indeed is), consider the transition between the (interface) modes in ‘OUTLINE’ view in the Microsoft Word processing application. When there is any change in outline levels (or modes of the screen), the documents scrolls to a page that contains the current location of the cursor. So while a user may be looking at page 20, and the cursor is located in page 1, once he or she transitions to a different mode, the screen displays page 1. This is particularly annoying when the user has just opened the document, and has not done any editing. In the latter case, the cursor location is at the top of the document. Once the user scrolls down to view other parts of the document, a change in mode will take him or her back to the first page. This represents a dynamic process in a very static system.

Default transitions are somewhat similar to automatic ones. And they are problematic in the same way that automatic transitions are. The only difference is that default transitions occur immediately—there is no dependency upon some other condition becoming true. They can be found in many consumer electronic as well as computer-driven interfaces.

Representation

The system has two modes, A and B (see Figure 6-4). Initially the system is in mode A. Then, some event (CM_3) occurs (either manually or automatically), and the system is initialized. Upon reentry, default transition CM_4 ensues. Again, the display does not change its state to announce that B is the current mode. This subcategory (default) is more problematic than manual transitions because the mode changes without the user’s direct control. In order to anticipate this transition, the user must have (1) complete knowledge specification about the transition and the defaults, and (2) sensing of event CM_3 .

Summary

This structural element involves a situation in which the interface under-specifies the behavior of the machine. That is, given the task specification of the user, the behavior of the machine with respect to its states (two plant states—one display state) is such that the user cannot tell which mode is actually engaged. This structural element is rather common and can be found in many everyday products.

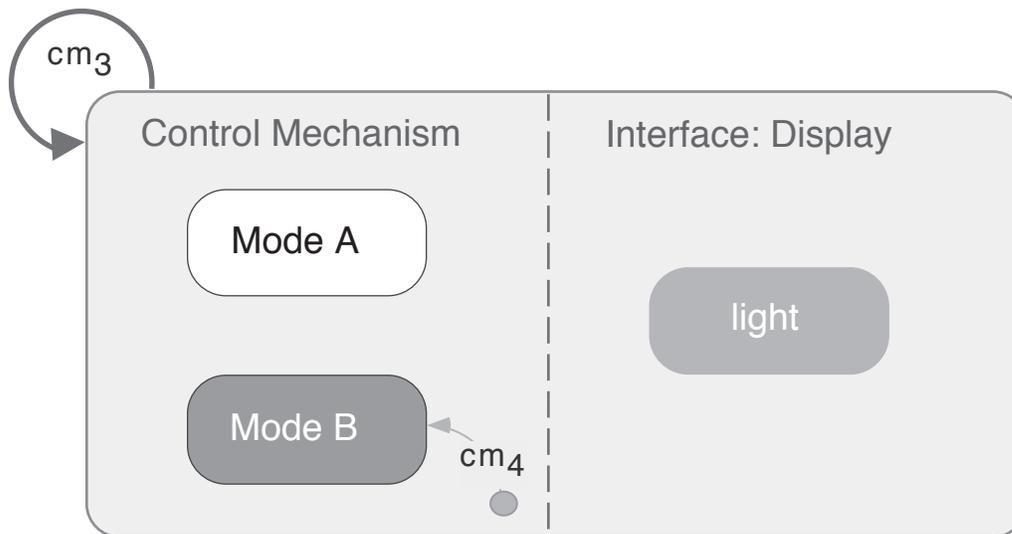


Figure 6-4. Two plant states—one display state (default transition)

Given this unique structural element, the *type* of transitions that change the configuration of the device are described. Three subcategories are characterized: manual, automatic, and default. Each type of transition may lead to mode ambiguity, depending on the user knowledge specification and interface sufficiency.

HIDDEN TRANSITION

In the previous structural element we focused on the display and control-mechanism relationships. In this structural element we focus on the controls and control-mechanism relationship. There is much in common between the two. We expand on this topic in order to make it easier for designers to identify such structural elements.

This structural element characterizes an architecture in which there is more than one way to engage a mode. This, however, is not necessarily a problem. It only becomes so in the context of an interface that masks this information. When the user is told, via the interface, that there is only one given way to change a mode and it turns out there is another (hidden) way to do it, the user may get confused. That is, he or she expects that only one action (moving a switch) will engage a mode, and then unexpectedly, the machine changes into another state.

The type of transition described here is manual (although the same problems may occur in an automatic transition). There are two subcategories, depending on the type of condition (.OR., .AND.) that triggers the transition.

.OR. Condition

In this subcategory, the hidden transition is manual. That is, the user manually fires this transition from somewhere else in the interface. The interface, however, does not depict this action.

Examples

One example of this structural element is the map display system described earlier. The transition from one range to another can be made in one of two ways: (1) by changing the display range via the range selector knob, or (2) by selecting different interface modes ('ARC' or 'ROSE) via the interface mode selector switch. While the former has a one-to-one mapping between physically rotating the range knob and the desired effect (change in map display range), the latter (mode selector knob) has no such clear linkage to the map display range.

Representation

Figure 6-5 depicts this structural element. Initially the plant is in mode A. The user *manually* requests a transition from mode A to B by throwing a switch. The switch label corresponds one to one to the machine modes. When the switch is thrown (or a button pushed), the event ic_5 fires another (internal) event cm_5 . As a consequence, the transition from mode A to B takes place. To the user, who is guided by the interface, this is the only (perceived) way of making the plant transition to mode B. However, there is another way to transition to mode B. If event cm_{110} is sensed, the control mechanism will also transition to mode B. The user may know about this (by means of an instruction book), or not. However, the interface does not hint of this possibility.

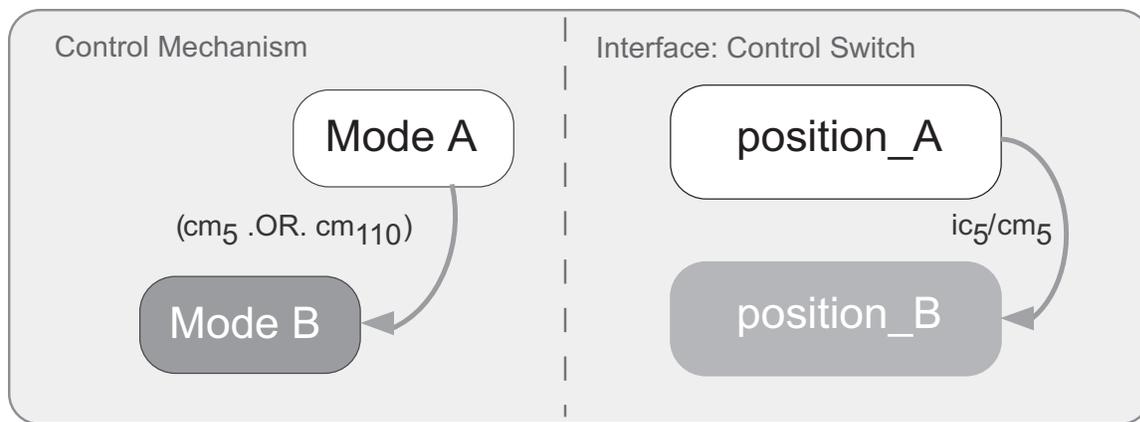


Figure 6-5. Hidden transition (.OR. condition)

.AND. Condition

In this subcategory there is an .AND. condition, or guard, on the transition between the modes. That is, until the guarding event is TRUE the transition will not take place. The ambiguity occurs because the switch is in the desired position, yet the machine is not in the corresponding state.

Examples

The example for this structural element comes from a military fire control system. The fire control system has two main modes: “Operational” and “Simulation.” In “Operational” mode the system sends output directly to the firing units (guns, missiles). In “Simulation” mode (used for training), the output is directed to a data file (to be used later in debriefing). Due to the design of the mechanism that switches between the modes, another condition has to be TRUE before the transition takes place. Yet, both the control position and the display indicator are dependent only upon the switching request (and not upon the control mechanism state). A similar feature was a contributor in the Three Mile Island nuclear reactor accident (Rubinstein, 1979).

An incident occurred in which the switch was thrown to “Simulation.” However, since the (hidden) condition was not TRUE the system stayed in “Operational” mode. Nevertheless, the control/display indicated “Simulation”! The user then started what he thought was a simulated missile launch process. Luckily, the process was aborted early on.

Representation

In this case two conditions must be both TRUE before the transition takes place. The user throws a switch (ic_6) which fires an internal event (cm_6) (Figure 6-6). The transition between mode A and B is predicated upon two events or conditions, cm_6 and cm_{120} , being TRUE at the same time. In other words, unless the event cm_{120} is TRUE, the transition will not be completed, even if the user has already thrown the switch.

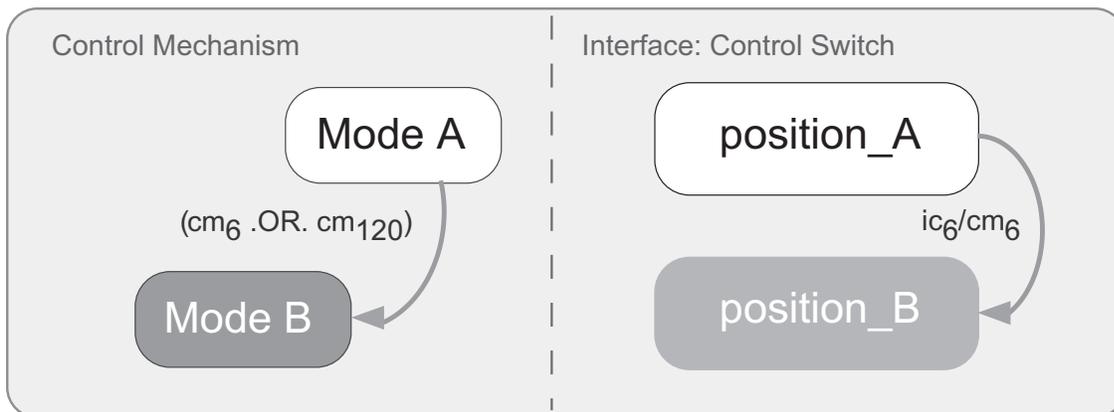


Figure 6-6. Hidden transition (.AND. condition)

In this type of situation, the only way a user can resolve the situation is by having (1) a complete and accurate model of the system, and (2) the capability to sense whether the “hidden” event (cm_{120}) took place or not. Interestingly, one immediate solution that was devised after the fire control incident was for the operator to place an ear on the side of the control panel. It turned out that the transition made a perceivable noise (“thump”) when it occurred. This technique, in a very crude way, is what we have argued earlier—the capability to independently sense events that cause transitions. Only by having this sensing capability and knowledge of the conditions, can the user resolve the next mode configuration of the machine.

Summary

The structural element described here is a heuristic. It categorizes an architecture that may lead to mode ambiguity, and is further broken down to two subcategories—*.OR.* and *.AND.* conditions—that are reciprocals of one another. In the case of the *.OR.* condition, the system may transition to a new mode without any change in the interface. In the case of the *.AND.* condition, even though the user requests a transition, the system may not switch modes—yet the interface shows that it did. In both cases, however, the problem is observed when there is asynchronization between the state of the interface and the state of the control mechanism (cf. Leveson, 1994, p. 366).

Two missing pieces of information will lead to mode ambiguity. One is that the user does not remember the knowledge specification or did not have it in the first place. The other is that the user has no ability to sense the hidden event. Note that without the sensing capability, no procedure would have aided in this situation. The user, by way of a poor system design, is set up for a mode confusion.

The issue of hidden transitions and resulting unexpected system behavior and confusion on the part of users, is of much concern in modern control systems, particularly high-risk ones. Because such systems may be large and complex, not all information can be provided to the user in an instructional book, or directly taught. Users, in the meantime, tend to develop their own explanations for such unexpected behaviors. The inability to confirm the reasons for the behavior of the machine may result in what is sometimes termed “folk models” in the mental-model literature (Gentner and Stevens, 1983; Wilson and Rutherford, 1989). These folk models, in the context of complex systems such as aircraft, are symptoms of crews trying to develop conceptual understanding of the behavior and layout of the system (Hutchins, 1992, p. 6; Sarter and Woods, 1995b).

CIRCULAR MODE TRANSITION

Circular states (or modes) are a common architecture for switching between modes in modern-day devices. There is nothing inherently wrong with this implementation. However, coupled with other factors such as lack of feedback and hidden transitions, this architecture may lead to mode ambiguity.

In this architecture, the physical switch has an idle state and only one other state. In contrast to a static switch that stays in its set position, this particular switch always springs back to its initial state. An example is the set-speed switch on most cruise controls. Here, the state of the control does not indicate the desired setting. The only feedback about the state of the machine comes from the display (if there is one).

Now if there is a hidden transition that changes the state of the machine, mode ambiguity may occur. Specifically, this occurs when the user, unaware of a hidden transition that took place, hits the switch and gets into an unexpected mode. In some systems (e.g., electronic watches, VCRs) we observe situations in which the user, frustrated, hits the switch several times trying fruitlessly to transition into the desired mode, but instead finds himself or herself in a succession of undesired modes. The situation sometimes leads to a parody of mistakes.

Example

A specific example of this structural element is the cordless phone described earlier. In that example, the ‘Talk’ button had a circular mode transition. This feature augmented the user’s possible confusion after lifting the handset. Pressing on the ‘Talk’ button after lifting the handset

from the base initiated an unwanted result—hanging up on the caller. Why? Because the ‘Talk’ button is state dependent: While the phone is idle, hitting the ‘Talk’ button activates the phone. While the phone is active, hitting the same button puts it back to idle.

Representation

The spring-loaded switch in Figure 6-7 has two states: idle and depressed. Once the switch is depressed (event ic_8), a request for mode change (cm_8) is broadcast. Once this information is received by the control mechanism module, the mode transitions from A to B. Back to the interface module, once the user lifts his or her finger off the button, an immediate transition back to idle takes place. This internal transition, caused by a spring-loaded mechanism, is depicted as a broken line in Figure 6-7.

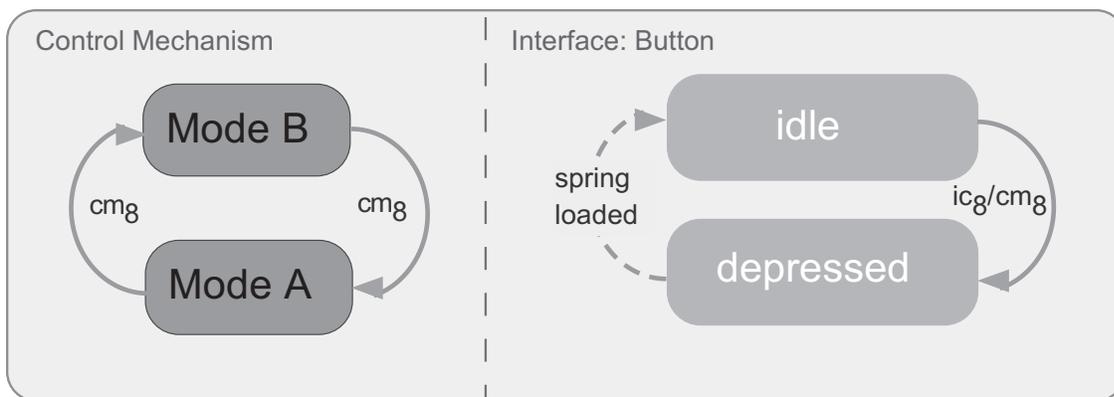


Figure 6-7. Circular mode

A closer look at the control mechanism module will reveal that the same event (cm_8) transitions the control mechanism between the modes (it is the label of both transitions). The new configuration of the system, therefore, is always dependent upon the previous state configuration of the machine. The only indication for this is through a display (if there is one). If the user is not aware that the control mechanism has already transitioned to mode A and decides to change mode, he or she will experience an unexpected mode configuration.

In the case of a hidden transition, the only way a user can resolve the situation (with no display) is by having (1) a complete and accurate model of the system, and (2) the capability to sense the event that triggers the hidden transition.

Summary

This structural element describes an architecture in which the user action is mode dependent. Unless salient feedback is added, the interface itself no longer acts as a differentiating indication of the mode configuration of the system. Two issues must be clarified, however. We are not arguing that circular mode architecture is undesirable. We are suggesting only that in some contexts (e.g., the cordless phone) this architecture may lead to mode ambiguity and confusion. We also are not suggesting that simply adding some type of feedback (e.g., a light switch) on the button is the way to solve the problem. In the case of the cordless phone, there is an indication (a red light) to signal whether the phone is active, yet this does not prevent the user from making an error. The reason is that there is a learned sequence of action that bypasses this indication (see the discussion section in the phone case study). Interestingly, one solution to this problem, seen in later models of this

phone, is to add another button. Now there is one button for activating the phone and another for deactivating it.

Examples of the contribution of circular mode transitions are not confined to consumer products. Aircraft Flight Management Systems also have a variety of circular buttons and modes. Coupled with hidden transitions, this structural element may lead to mode ambiguity. It appears that if circular mode architecture is used in conjunction with automatic mode transitions, one has to build a very powerful display to combat the likelihood of mode confusion and subsequent error.

MODE-REFERENCE VALUE INTERACTION

In many systems, mostly those that control a dynamic process, there is a relationship between the engaged mode and the associated reference value. The reference value serves as a qualifier on the behavior of the mode—it constrains the mode behavior. This relationship has several different architectures, some simple and others more complex. The most complex are found in high-risk systems where built-in “envelope” protections are provided for safety (e.g., the “Retry” mode in the blood pressure device, also cf. Sarter and Woods, 1995a, p. 12).

The modes are described using the same format as in the previous structural elements, while the reference values are described as concurrent processes which run in parallel to the modes. Reference values can be either manually set by the user, automatically set by the machine, or sensed from the state of the plant (e.g., speed).

In this particular structural element, we describe three subcategories: (a) reference value causes a mode change, (b) mode transition causes a change in reference value, and (c) default transitions in reference values and/or modes.

Reference Value Causes a Mode Change

In this subcategory a change in reference value, entered by the user, causes a mode transition. The mode transition is automatic and therefore may confuse the operator who may be unaware of (or forget) the unique interaction between the reference value and the mode.

Changes in reference values are not necessarily entered only by the user—they can also be sensed by the machine. This kind of architecture is usually in place to protect the system from boundary conditions (“envelope protection”). That is, when the control mechanism senses that either the user requested, or the plant (and process) is at some value which is defined as a critical threshold, the control mechanism will automatically change modes to avoid this boundary condition. Such examples are found in the control mechanism of medical devices and modern aircraft.

Example

Consider, for example, the architecture of the blood pressure device discussed earlier. The device allows the user to set an interval between measurements cycles, and then automatically conducts the measurement. When the device is operating in fully automatic mode, and the user enters some new interval, the device transitions into “Idle” mode. In other words, a reference value change caused a mode transition.

Representation

Because of a change in reference value, the machine transitions from one mode to another. The control mechanism initially is in “Mode A” (Figure 6-8). The user makes an adjustment to the reference value that is beyond some system threshold. This event (rv_3) fires an event (cm_3) which

causes the transition from “Mode A” to “Mode B.” In this case, in order to predict the next mode and reference value configuration, the user must have a complete and accurate model of the system. Another subcategory of this structural feature is when the transition rv_3 is not made directly (manually) by the user—instead, it is adjusted by some other component. This situation is more confusing, of course, because the user must possess the capability to sense the event that triggered transition rv_3 .

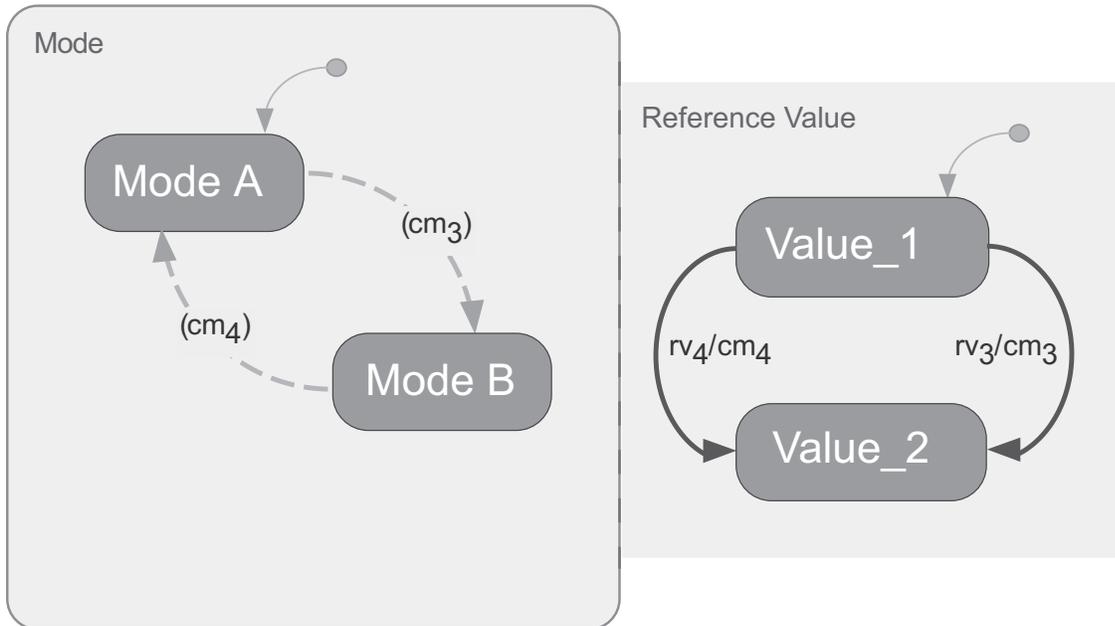


Figure 6-8. Reference value causes a mode change.

In the above situations, the transition between “Value_1” and “Value_2” depends on a reference value entered manually by the user or automatically by the machine. There are architectures in which the transition between reference values and subsequent mode changes depends on values sensed by the machine. The latter situation, again, taxes the user because he or she must be able to sense this value in addition to having a complete and accurate model of the system.

Mode Transition Causes a Change in Reference Value

This subcategory is common during transitions among supervisory modes. At the highest level, i.e., fully automatic, the reference value is adjusted by the machine in order to achieve fine tuning and (cost) efficient operation. The user may step in and change the machine’s mode. This event now opens the reference value to the control of the user.

Example

An example is a pilot who transitions (because of a request or ATC procedure) between a fully automatic and semi-automatic mode while the machine is trying to adjust speed. Once the semi-automatic mode is active, the speed reference value defaults to the current speed at the time of the transition. This speed, of course, may be different than the ATC-requested speed.

In order to anticipate the new status of the system, the user must first have a complete and accurate model of the system. In the case where the transition between “Mode 0” and “Mode 1” is automatic, it is much more difficult for the user to anticipate the next mode configuration of the

machine. In addition to full knowledge specification, the user must also have the capability to sense the event that triggered the transition.

Representation

Figure 6-9 depicts this kind of architecture. The initial state configuration is “Mode A” whose reference value is provided automatically by the machine. The system is fully automatic—it adjusts the reference value in accordance with some control algorithm. The user now engages “Mode 1” (event cm_1). As a consequence, event rv_1 is fired and the reference value is now manual—at the whim of the user.

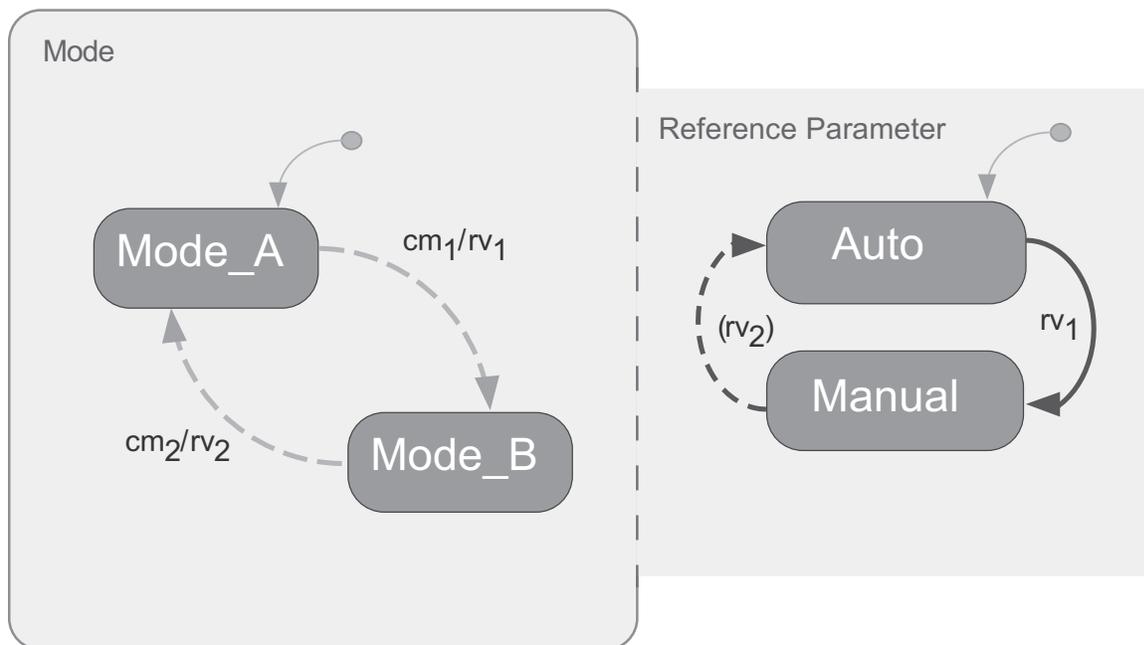


Figure 6-9. Mode transition causes a change in reference value.

In most machines, the manual reference value defaults to the current value at the time when the transition between modes occurred. This also can be problematic in cases where a gross adjustment in reference value is taking place automatically, and the user decides to transition to some semi-automatic mode (that simply captures the current value at time of transition).

Default Transitions in Reference Values and/or Modes

This subcategory deals with transitions that cause superstates to cycle and hence default to some preset values. As mentioned earlier, default entries have some similarities to automatic transitions, as they are not directly prompted by the user. Instead, they cause transitions into configurations that were selected by the designer. One of the concerns in this type of implementations is whether the user is aware of (1) what causes a transition that fires default entries, and (2) what the new default states are.

In complex, dynamic, and high-risk systems, default transitions are necessary because the process cannot wait until the user decides to initiate an adjustment. The machine, therefore, selects the new state configuration by default. Other systems have default states as initial values when the machine or component is powered or re-activated. Designing good defaults is far from being a trivial matter.

The designer must predict the situation in advance and choose a default that will be within the expectations of the user and still maintain the machine in a safe configuration.

Examples

In the aviation domain there have been several accidents in which unexpected default configuration, unanticipated by the crew, was one of the contributing factors of the accident. Default states were also a factor in several documented incidents (Degani, et al., 1995, scenario 3) and were used as a scenario element in a recent simulation study about pilot interaction with mode-based systems. In the simulation study, only about half the subjects anticipated the default; the other half reacted to the situation belatedly (Sarter and Woods, 1995b, p. 51).

Representation

Figure 6-10 depicts a machine with two modes and a set of reference values. The initial mode and reference value configuration is “Mode A” and (reference) “Value 2.” The machine allows for changes in reference values, and various other values can be selected either manually or automatically. However, once the user transitions to “Mode B,” event cm_5 fires event rv_0 . On the reference value side, event rv_0 initializes the superstate and default transition to “Value_2” takes place.

A more involved scenario takes place when event e_1 takes place. When this is sensed, transition cm_6 is enabled and the entire superstate (including the modes and their reference values) gets initialized. Regardless of the previous configuration of modes and reference values, the machine defaults to “Mode A” and (reference) “Value_2.” In order to anticipate the new (default) configuration of the machine, the user must have (1) a complete and accurate model of the system—with all its default transitions, and (2) the capability to sense the event (in this case, e_1) that triggered this cascade.

Summary

A mode and its reference value compose the general form of mode behavior. While mode and reference-reference value interaction exist in almost every machine, they are critical in dynamic control systems. It appears that when these interactions violate some population stereotype, the potential for mode ambiguity increases.

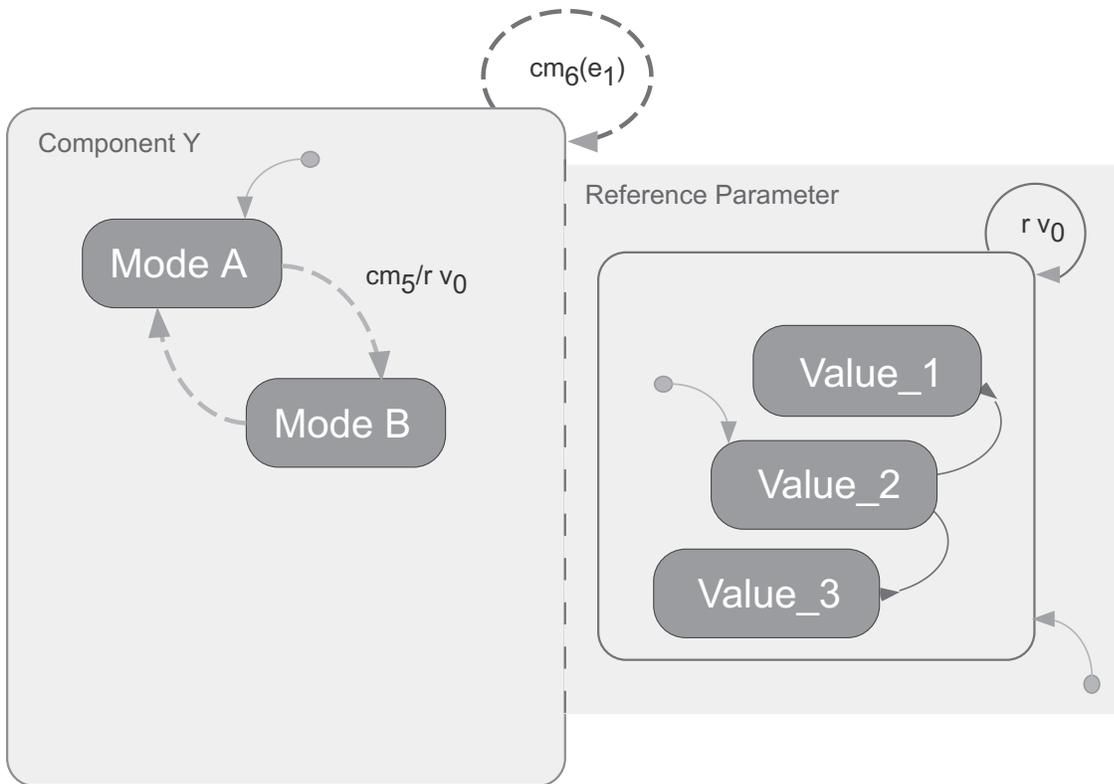


Figure 6-10. Default transitions in reference values and/or modes

To support such complex interaction, the user must have a detailed knowledge specification and/or a good interface that provides cues about the mode and reference value interaction. And perhaps dealing with a system that has several elements with non-trivial interaction is simply inherently difficult for humans (Simon, 1969/1981, p. 219). In designing such system, this issue must be evaluated and tested.

Default configurations following the reset or exit from modes and levels of automation is a complicated topic, given the difficulty for the designer in anticipating the situation in which this will take place. This difficulty is only magnified in high-risk systems. It is a human factors area that is far from being completely researched—yet it extracts a very painful toll.

MODE-MODE INTERACTION.

Complex systems have several concurrently active components. Each of these components has a variety of modes. For example, the blood pressure machine has a mechanism to deal with the timing of a cycle as well as a concurrent mechanism to deal with the pressure of the cuff. Each component is equipped with its own set of modes, and the mode configuration of the system is an (.AND.) combination of these modes.

There are situations and corresponding design features that require interaction between modes of each component. In this case, the output of one component is the input of another (Ashby, 1956/1964)—the components are unilaterally or bilaterally (in cases of a two-way link) coupled. In the same way that any concurrent process with interaction poses problems to the human supervisor, mode-mode interaction may also lead to mode ambiguity. Again, state (mode) dependency—the inherent feature of any complex system—is the foundation of such ambiguity.

In this particular structural element, we describe two subcategories: (a) manual transition and (b) automatic transition.

Manual Transition

In this subcategory, the transition that couples one component to another is manual. That is, the user changes the mode in one component, and that event automatically changes the corresponding mode in another component. A mode-mode interaction is a common feature in many devices that we operate on daily basis. In chapters VII and VIII we shall discuss such interaction in detail.

Example

Consider, for example, the device that controls temperature, humidity, and vent in a modern car. Commonly what one sees is a push-button air conditioner button ('A/C') and a moving selector for air distribution (e.g., 'FLOOR', 'MIX', 'PANEL', and 'DEFROST').

The characteristics of the device are such that there is an interrelationship between selecting the defroster and air conditioning. This is because efficient defrosting of the windshield requires not only that air be blown on the window, but also that dry (conditioned) air be used. Dry air absorbs the moisture and helps defrost the windshield. Interestingly, this information is not presented in many car manuals and is not necessarily depicted on the interfaces. The interrelationship is so designed that there is a unilateral coupling between the defroster and air conditioner. Moving the selector to 'DEFROST' automatically activates the air-conditioning compressor. In many cars, however, activation of the compressor by way of the defroster is not annunciated; that is, the (green) light on the 'A/C' button is not lit. The combination of such unilateral coupling (so that the defroster activates the compressor) with limitations in knowledge specification (users don't know this fact), may lead to mode ambiguity. And finally, the lack of mode indications ('A/C' light is off) hampers the user's ability to recover from being in the wrong mode (and usually leads to a parody of errors). Imagine driving up a steep hill on a foggy day while the person by your side "helps" you by selecting 'DEFROST'!

Representation

The machine described here has two parallel components, Y and Z (Figure 6-11). Component Y has two modes (A and B) and component Z has two (α , and β). When the user requests a transition (cm_1) from "Mode A" to "Mode B," the transition fires an event (cm_2). As a consequence, the transition between "Mode α " and "Mode β " in component Z takes place. To anticipate the new configuration of the bi-component machine, the user must have a complete and accurate model of the system.

Automatic Transition

While the previous subcategory presented manual transition, this category describes events generated internally by the machine. The machine senses an event somewhere in the network and fires a transition, which, in turn, fires another transition. This category bears similarity to structural element 2 (hidden transition). However, our focus here is on the interaction between two mode components, while in hidden transitions the focus is on the interaction between the control mechanism and interface. This mode-mode interaction is quite difficult to track without the appropriate knowledge specification and sensing capability, and therefore may lead to mode ambiguity. Such mode-mode interaction architecture exists in complex systems such as aircraft Flight Management Systems and military fire control systems. In chapter VII we shall discuss such automatic transitions in detail.

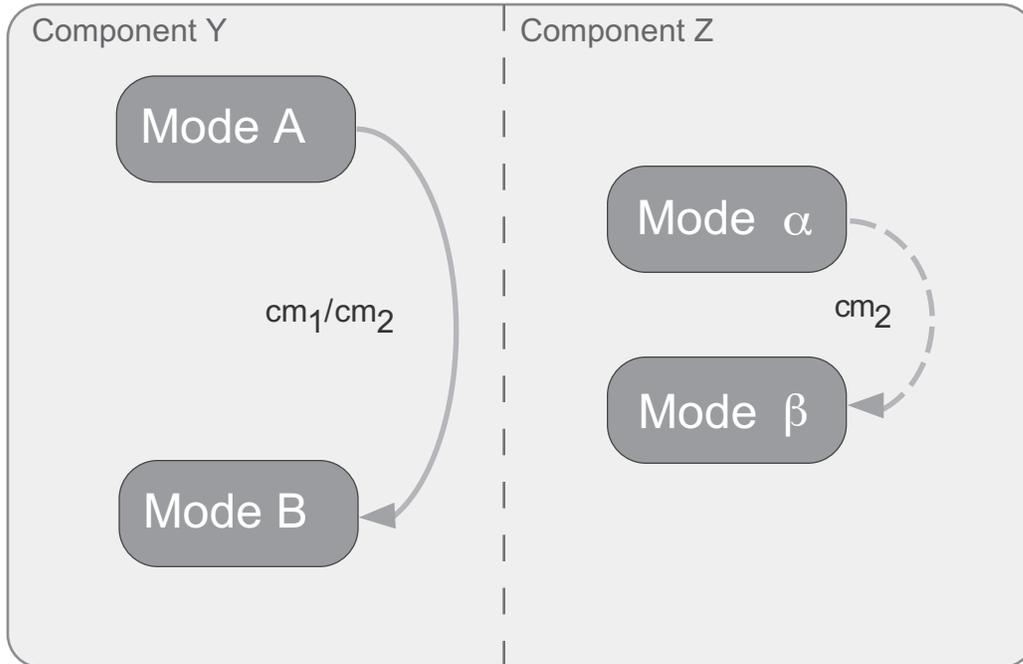


Figure 6-11. Mode-mode interaction (with manual transition)

Representation

When the machine senses event cm_{130} it transitions to from “Mode A” to “Mode B,” and the transition fires event cm_4 (Figure 6-12). As a consequence, the transition between “Mode α ” and “Mode β ” in component Z occurs. To anticipate the new configuration of the bi-component machine, the user must have (1) a complete and accurate model of the system, and (2) the ability to sense events in the network.

In addition to the two subcategories described here (manual and automatic), another category—default—can be found. In this subcategory, there is an event that initializes the superstates that contain the two components. After this event, the default modes in each component become the active modes. Again, the user must have a complete and accurate model of the machine, as well as the ability to sense the initialization event, in order to predict the new configuration of the machine.

Summary of Mode-Mode Interaction

The mode-mode coupling is a common structural element in complex system. In these systems, each component has a variety of modes. In many situations, there is independence between components’ modes. This allows for many possible mode configurations and hence much flexibility in using the device. However, during initialization, as well as situations in which only one mode combination must be used, there is synchronization between modes. From a human factors perspective, such mode-mode coupling requires much attention in the design process.

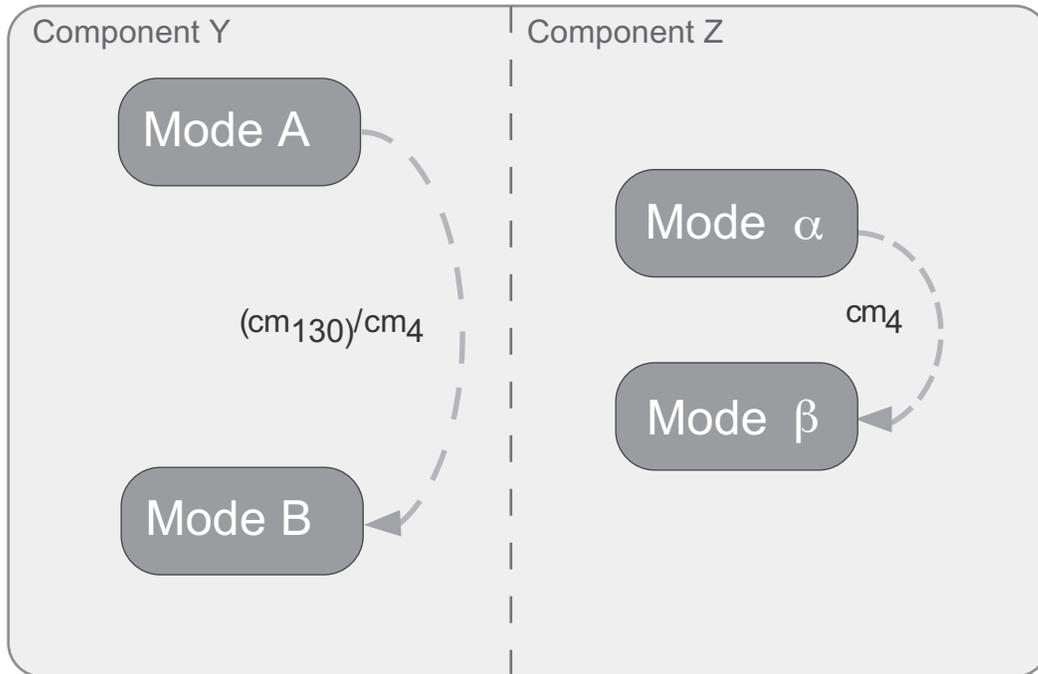


Figure 6-12. Mode-mode interaction (automatic transition)

CHAPTER SUMMARY

In this chapter we place some of the design features that contribute to mode confusion into five categories. Each category has several subcategories, mostly related to the type of transition—i.e., manual, automatic, default. For each category and subcategory we present an example and provide an abstract representation. In addition, we discuss the type of information required in order to anticipate the next mode configuration. We then continue to discuss the implication of such features on the design.

From a theoretical point of view, the structural-elements categorization scheme proposed here has several limitations: (1) Because they are heuristic in nature, a systematic (e.g., formal and automatic) check of the specification cannot be done; (2) because they are based on past experience, we can never be sure if we are over-fitting—i.e., making a structural element of a unique problem that may never occur again; (3) finally, we can never be sure whether additional undocumented structural elements exist but have not been identified. The structural elements are design heuristics. They are a set of specific design features that, in combination with other factors, may lead to mode ambiguity. Theoretically, they can only be viewed as an initial step toward a theory of mode ambiguity.

The main reason they are described here is because mode error is so profound and potentially lethal in supervisory control systems. Efforts to address these problems are currently ongoing in many design endeavors, specifically those involving human interaction with supervisory control systems (R. C. Graeber, personal communications, June, 1996). Any information, even of a preliminary nature, may provide guidance to designers who grapple with these issues on daily basis. It is our duty, as researchers in this field, to support these designers.

Such representation of human-machine interaction is portrayed in a language which is commonly used by control engineers (Leveson, Heimdahl, Hildreth, and Reese, 1994). Furthermore, when the machine's specification is already described in a Finite-State-Machine formalism such as

Statecharts, the costly overhead associated with building a human-machine model from scratch is much reduced.

The structural-elements categorization scheme is a toolkit that may help designers in several ways. The elements may provide designers with understanding of the unique features that lead to mode ambiguity. Further, because they are abstract, the designer can use them to search a given design specification and identify whether or not some of these structural elements exist. Once this is done, a red flag is hoisted, and then a design decision about how to solve the potential problem can take place. The solution may involve the control mechanism, the interface, training, or a combination of several of these factors.

Finally, structural elements are static—they are a snapshot of the structure of the machine behavior. They do not take into account the real-world action sequences or the paths taken by the user in operating the device. In the next chapter, we attempt to address this issue by collecting data on how pilots use the modes and reference values of the Automatic Flight Control System (AFCS) of a modern aircraft. We describe in detail the sequence of mode configurations pilots use in operating this system, the various factors that prompt mode transitions, and the patterns of interaction between the pilots and the AFCS.

MODE USAGE IN A COMPLEX SYSTEM I: FIELD STUDY AND STATISTICAL ANALYSIS

INTRODUCTION

We now proceed to conduct an analysis of mode ambiguity in a complex system, namely the Automatic Flight Control System (AFCS) of a modern aircraft. Such a complex system, with several concurrently active components and many functional and supervisory modes, is quite challenging to study mode usage. We must first ask ourselves whether we have enough conceptual resources (i.e., models and templates) to analyze mode usage and mode ambiguity in this highly complex system.

The previous review of the literature of modes in aviation hinted that our Ofan model alone may not be sufficient for understanding the interaction between users and a complex system. In our previous models, we assumed the mode configurations and paths users take in operating the device. In the cordless phone example, there was no urgent need to model the user's mode selection sequence because the task specification was simple and implicit in the model. In a complex system that has a huge mode configuration space, however, it is simply not feasible to assume the user's sequence of mode selection. Other forms of analysis must therefore be investigated in order to further understand the interaction between users and a highly complex system, and to potentially discover the sources of mode ambiguity.

In the research to be presented in this chapter, we have employed a variety of statistical methods to understand mode usage—how operators and users actually use modes, which modes they select and occupy, the mode configurations to which they transition, and the path users take through the possible mode configuration space of the system. We believe that users' selection of mode trajectory paths is not arbitrary. We have therefore set out to explore the factors that constrain the selection of paths.

To perform such an analysis, one needs data—and a lot of it. We obtained the data for this large-scale analysis through an observational study. Observational studies (or field studies, as they are sometimes called) can provide detailed insight into how users actually interact with a system. In this chapter we present an overview of the field study as a research tool, and discuss the vehicle, design, participants, procedures, and type of data collected. The results of the analysis are presented in the latter portion of this chapter.

We were also concerned with whether our Ofan framework could indeed be scaled to such a complex system. Could the templates and the structural elements that we developed earlier be used here, too? To address these issues we describe a mode confusion incident, or scenario, that was documented during the field study, and then model it using the Ofan framework. This analysis is conducted in Chapter VIII.

THE COMMERCIAL AVIATION DOMAIN

In this section we introduce the topic of modes in commercial aviation by first focusing on the use of automation in aircraft systems. Next, we briefly discuss some of the well-documented mode-related incidents and accidents in commercial aviation.

Automation in Aviation Systems

Increased levels of automation commonly parallel the extent to which technology is available (Zuboff, 1991)—and commercial aviation is no exception. Automation existed in pre-1980 airline cockpits in the form of autopilots, navigation systems, and other control systems. Nevertheless, more sophisticated automation systems in general, and the Automatic Flight Control System (AFCS) in particular, were introduced in the early 1980s because the efficient computers and sophisticated logic to process information were available (Billings, 1996).

When the AFCS was first introduced in the Boeing B-767, McDonnell Douglas MD-80 series, and Airbus A-320, it represented a revolutionary advance in the management of flight. The AFCS reduced workload significantly and contributed to the reduction of flight crew complement from three to two (McLucas, Drinkwater, and Leaf, 1981). It also reduced cost by optimizing flight profiles to minimize fuel and time costs (Lidén, 1994). Other advantages were economy of cockpit space, increased reliability, and enhanced navigation precision in fully automatic modes (Chambers and Nagel, 1985).

Mode-Related Problems

While there are many advantages to increased automation, there is a price tag for these advantages, as well. The foremost limitation from a human factors standpoint is the parallel increase in system complexity. The complexity of the AFCS modes and behaviors is formidable, and is highlighted by the many studies mentioned in the literature review. The manifestations of such complexity in terms of human interaction can be seen in training, in-flight incidents, and—worst of all—accidents.

Training airline pilots is both necessary and expensive. Currently, flight training departments at various airlines have considerable difficulty in training pilots to fly highly automated aircraft. Explaining the complexity of the AFCS and identifying the level of knowledge (specification, in our terms) that must be provided about the AFCS, without compromising safety, are issues of much concern and debate (*Aviation Week and Space Technology*, 1996b). There is a consensus among aviation safety professionals that incidents are usually precursors to accidents (Reynard, Billings, Cheaney, and Hardy, 1986). There are hundreds of mode-related incidents, some of them nearly identical in situation to accidents that have occurred, documented by organizations such as NASA's Aviation Safety Reporting System (ASRS) in the U.S. (Chappell, 1994) and similar organizations world-wide. As an example, consider the following report submitted to the NASA Aviation Safety Reporting System:

...Area arrival progressed smoothly and we were cleared for the approach to the runway. When changing radio frequency from Approach to Tower (head down), the First Officer selected "open descent" to 400 feet. The autopilot was off, both flight directors were engaged, and autothrust was on. While contacting San Francisco Tower, I became aware that we were below the glideslope, that airspeed was decaying, and that we were in an "open descent." I instructed the First Officer to engage the "Vertical Speed" mode in order to stop our descent, restore the speed mode for the autothrust, and continue the approach visually. Company

procedures explicitly prohibit selecting an altitude below 1500 feet for engaging the “Open Descent” mode, since this places the aircraft close to the ground with engines at idle. (ASRS Report No. 149672).

What is portrayed, in very laconic language, is a near-accident in which the aircraft was below the glide path to the runway, with an increasingly limited ability for recovery.

AFCS Accidents

In the last decade or so, there have been several major accidents in which human interaction with the AFCS was cited as a contributor to the accident (*Aviation Week and Space Technology*, 1995). The most notable is a recent accident involving a Boeing B-757 near Cali, Columbia (Aeronautica Civil of the Republic of Colombia, 1996), in which the aircraft, under the guidance of a fully automatic mode, initiated a turn because of an erroneous reference value entered by the crew. Mode-reference value relationships appear also to have contributed to a recent accident involving a test flight in the new Airbus A-330 (Commission of Inquiry, 1995). In another accident that occurred near Strasbourg, France, the (functional) mode configuration of the aircraft was ambiguous, possibly due to interface mode problems and mode-mode interactions (Commission of Inquiry, 1992). Finally, several accidents have been partially attributed to ambiguity of the current (and next) supervisory modes of the AFCS. One of these involved an Indian Airlines A-320, and the other a China Airlines A-300/600 (*Aviation Week and Space Technology*, 1996c; Indian Court of Inquiry, 1992).

Autopilot Accidents

Mode problems in aviation are not confined to the modern AFCS. Pilot interaction with autopilots and other automatic control mechanisms also involves mode ambiguity and resulting errors. The fate of Korean Air Lines flight 007, shot down over Soviet airspace, is a tragic case in point. While several books postulated a variety of conspiracy theories, the culprit appeared to be a navigation error prompted by a non-occurring (because the guard condition was not satisfied) automatic mode transition between a semi-automatic and fully automatic mode (International Civil Aviation Organization, 1993, p. 40-45). Likewise, other accidents have occurred because of confusion regarding autopilot modes.

Summary

In this section we have briefly reviewed the topic of automation in commercial aviation (see Billings, 1996, for a more detailed review). We have discussed some of the benefits and human-automation interaction problems that were documented, and have focused specifically on problems in pilot interaction with automatic flight control system modes. Armed with this background about the commercial aviation domain and its employment of automation, we now describe the objectives of our investigation into pilot interaction with these mode-based systems.

APPROACH AND OBJECTIVES

Pilot interaction with the AFCS is a dynamic process. Most of the mishaps discussed above are the result of a certain sequence of actions taken by the user, on the one hand, and a given (problematic) design feature of the system, on the other. This user-system interaction is bound by the process (flying) and the environment (ATC) in which it must operate.

As we mentioned earlier, we want to know what paths the user will take, given the mode of the interaction. We also need to ask, what are the sequences of action that depend on, and what prompts, such mode transitions? Finally, to understand human interaction with such control

systems, we must have an understanding of the user's operational environment. This environment drives mode transitions and the sequence of actions taken by the pilots.

Approach

Our approach for understanding pilot interaction with the AFCS was to conduct observations in the cockpit during routine revenue flights. We collected detailed data about the current mode configuration of the aircraft as well as aircraft status information. In addition, some 28 incidents of mode confusion that took place during the observation were recorded. Realizing that the demands from the operational environment are critical for understanding this pilot-AFCS interaction, we also collected data on pilot behavior as it is shaped by pressures of ATC, weather constraints, on-time arrival, and fuel efficiency.

Objectives

Our objectives were as follows:

- (1) Describe the various mode configurations that pilots actually occupy.
- (2) Describe the transitions between these mode configurations (i.e., the mode configuration trajectory).
- (3) Identify the factors external to the machine that constrain the pilot in making mode selections, and hence contribute to this trajectory.
- (4) In addition to listing these external factors and their magnitudes, identify their patterns of interaction with pilots' mode selections.

In the following section we describe the methodological aspects of conducting a field study. In addition, we provide a description of the system and the environment in which it operates.

METHOD

This section will briefly survey the available literature about field studies, focusing on their representational methodology and the type of insights they provide. We will then describe the Automatic Flight Control System of a Boeing B-757, and the air traffic control environment in which it operates. Finally, we will detail the experimental design framework, participants, procedures, and data for our field study.

Field Studies: A Research Methodology

The field (observational) study, the empirical tool of choice in many industrial engineering applications, has been neglected by the human factors research community until fairly recently. In the aviation domain, however, several human factors practitioners have used this methodology to produce meaningful research results to which this study is deeply indebted. Curry (1985) and Wiener (1985, 1989) conducted field studies focused on the introduction of very new automated flight control systems into line operation. They identified mode ambiguity and confusing mode transitions as recurring problems for flight crews. Casner (1994) conducted a field study that demonstrated the importance of operating environment demands on mode-based systems. These previous research insights have helped focus our attention on the structure of mode interaction in the human-machine-environment system. They have also pointed the way to our choice of methodology and data sources.

Because field studies are just now “coming into their own” in decision making, human factors, and human-computer-interaction research, a brief overview of their origin and use in a variety of domains may be helpful.

Historical Perspective

Field studies and observations of workers in their job settings are two of the fundamental methods for understanding human-machine interaction. The pioneers of field studies in industrial engineering, notably Frederick Taylor and the Gilbreths (Frank and Lillian), have developed and used observational studies in a variety of settings, including assembly stations and operating rooms, to obtain qualitative and quantitative information (e.g., time and motion studies) (Mandel, 1989). In time, partially due to the influence of experimental psychology on the discipline of human factors, such observational methodologies were discouraged because of concerns about the lack of experimental control (Doherty, 1993). The price of abandoning observational studies, however, was an inability to capture rich and context-dependent human interaction with tools and machines. Furthermore, applied studies have the potential to shed new light and further advance our theoretical understanding of human-machine interaction beyond the laboratory door (Fisk and Kirlik, 1996). However, there are some indications that the tide may be slowly turning back: Developments in human-computer interaction (HCI) and cognitive science have re-focused attention on viewing and studying human actions as situated in a particular environment (Hutchins, 1995; Kirlik, 1995; Klein and Calderwood, 1987; Suchman, 1987). These new developments give a central role to the environment and the capabilities of the machine in organizing user activity and interaction (Agre and Shragar, 1990).

In the following section we review several modern field studies with relation to the domain in which they were conducted.

Process Control Studies

Several descriptive field studies based on anthropological approaches, but applied to the socio-technical environment, have been performed. In process control, observations of operators and their verbal protocols have been conducted in a variety of industrial settings (Beishon, 1974; Edward and Lees, 1974). The issue explored was how operators control a multiplicity of activities and how they sequence and effectively control the process. Hutchins (1995) observed teams performing maritime navigation and described how sailors have adapted their cognitive resources to this task. He then used these observations in combination with his theory of cognition to shed new light on how we view human interaction with the surrounding world. Zuboff (1991) studied the sociological changes as companies and workers adjusted to the introduction of computers.

Some authors have used the field study as the source for developing modeling frameworks. Notably, Rasmussen developed the decision ladder based on his observation of power plant operators (1976). Another modeling and representation framework is the Abstraction Hierarchy (1993) that was reviewed in Chapter III, which is based on a variety of studies, one of them a field study of electronics technicians conducting troubleshooting tasks. In medicine, Xiao (1994) observed how anesthesiologists prepared themselves and their environments for an operation. Based on these observations, he developed a framework of planning and execution. The framework uses the Rasmussen decision ladder to represent planning and user actions.

Aviation Studies

In aviation, Amalberti and Deblon (1992) developed a cognitive model of mission and navigation tasks based on (video) observation of fighter pilots' tasks. The model was then used in the development of intelligent task aids. Several field studies have also provided guidance for training and design. Curry (1985) provided guidance for training pilots in using new cockpit automation (e.g., his call for "turn it [automation] off" training). Wiener, in an ongoing field study, provides guidance to airlines about training for automation (*Aviation Week and Space Technology*, 1996a). Degani and Wiener, in a series of field studies, observed pilots' use of procedures (e.g., checklists) and provided a list of guidelines for design of procedures in general, and procedures for automated cockpits in particular (Degani, 1992; Degani and Wiener, 1990; Degani, and Wiener, 1994).

Obtaining data for statistical analyses and appropriate inferences was the objective of two recent field studies. Lyall (1990; 1992) used an activity analysis methodology to gather and analyze in-flight data. She used the data to assess whether there were performance decrements when pilots switched between two different derivatives of the Boeing B-737 aircraft during the course of their trip—one equipped with an AFCS (-300) and one without (-200). The analysis indicated that there was no significant detrimental effect associated with switching between different models of the Boeing B-737. These findings led the cooperating airline to make an important organizational decision regarding this issue. Casner (1994) also conducted in-flight observations to determine the extent to which measured features of the operating environment influence pilot selection of four primary levels of automation in the Boeing B-737-300 aircraft. His specific purpose was to determine whether the semi-automatic or fully automatic modes were used, and whether the pilots were manually following the Flight Director guidance or turning the autopilot ON to perform this task. The results suggested a learning process in which pilots extract information and use it to devise strategies that are tuned to environmental demands. These strategies are based on learned scenarios that may occur downstream in the progress of the flight.

Summary

Field studies are a valuable method for obtaining data about human-machine interaction. The data can be used for providing insights and guidelines, statistical analysis, and documenting scenarios. Conducting a field study about user interaction within a very complex system forces the investigator to pay careful attention to the three components of any system: the machine, the operational environment, and the people operating the system.

Vehicle

The data for the field study were recorded onboard Boeing B-757-200 and B-767-200 aircraft. Although different in weight and size, these aircraft share an almost identical cockpit and aircraft subsystems (Corning, 1979, p. 14:16). With respect to the focus of the research, the Automatic Flight Control System, the B-757 and B-767 are identical.

The AFCS Overview

The major function of the AFCS is managing the flight path of the aircraft. Three fundamental aspects are involved in doing this: control, guidance, and navigation. Control refers to the problem of making small but continual adjustments necessary to maintain a reference value such as heading, altitude, and/or airspeed. Guidance refers to the problem of deciding what sequence of control activity is necessary to achieve a new altitude, heading, or airspeed reference value. Navigation involves formulating the sequence of heading, altitude, and airspeed that constitute the flight route.

The Automatic Flight Control System (AFCS) on the Boeing B-757/767 consists of three major hardware/software components: (1) *the autopilot and Flight Director system* that provides control (pitch and roll) commands to the aircraft control surfaces (ailerons, horizontal stabilizers, rudder) in order to provide guidance; (2) *the autothrottle system* that provides thrust commands to the engines; and (3) *the Flight Management System (FMS)* that combines flight route information, information from onboard sensors, and performance information stored in its databases to compute an optimum (i.e., fuel-efficient) flight route.

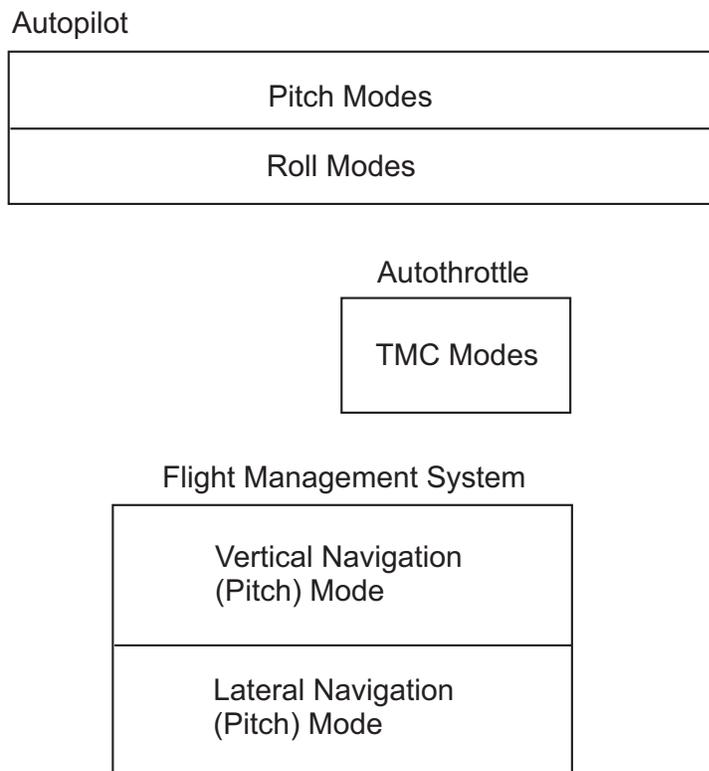


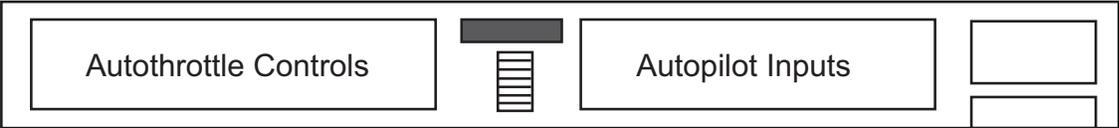
Figure 7-1. Components of the AFCS

Generally speaking, the AFCS functional mode *behavior* controls the flight path along two principal axes: vertical and horizontal. Changes in the vertical flight path result in altitude changes; the pilot requests such changes by transitioning among a variety of “pitch” modes. Changes in the horizontal, or lateral, flight path result in heading changes; the pilot requests such changes by transitioning among “roll” modes. There are ten pitch modes and eight roll modes. In addition, the Thrust Management Computer (TMC) has seven thrust-reference modes. These modes specify the limits on the engine thrust (Figure 7-1).

AFCS Interface: Controls and Display

The pilot interacts with the AFCS system through three control panels: (1) the *Mode Control Panel (MCP)*, which provides mode and reference value inputs to the autopilot and autothrottle systems; (2) the *Thrust Mode Select Panel (TMSP)*, which allows the pilot to engage additional thrust modes; and (3) the *Control Display Unit (CDU)*, which is the interface to the Flight Management System (FMS), and allows the pilot to enter navigation, (e.g., flight plan) performance (e.g., fuel cost), and atmospheric (e.g., wind) information (Figure 7-2).

Mode Control Panel



Thrust Mode Select Panel



Control Display Units

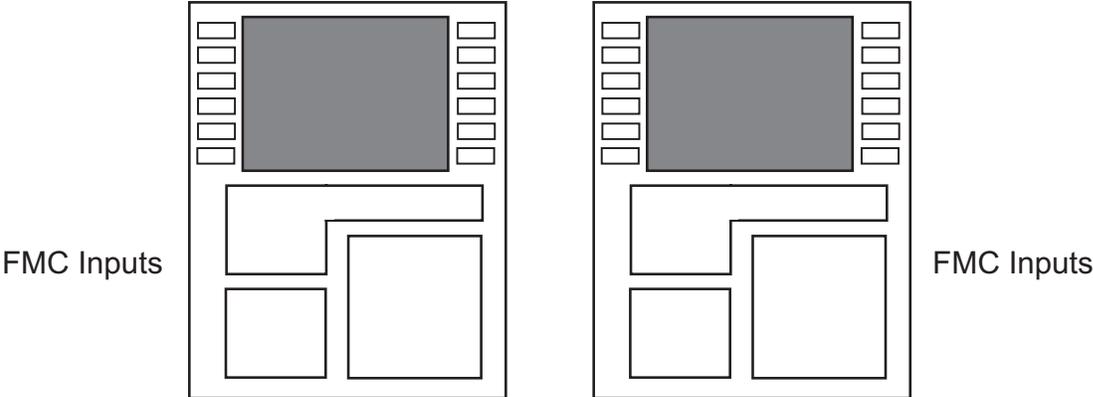


Figure 7-2. Interface to the AFCS

The pilots obtain feedback about the mode configuration through the automated Flight Mode Annunciation (FMA) unit. Pitch and roll modes status is displayed on the sides of the Attitude Display Indicator (ADI) (Corwin, 1993). The mode annunciation format is a truncated or abbreviated name of the mode (e.g., the text ALT HOLD indicates that “Altitude Hold” mode is engaged) (Figure 7-3). The Thrust Management Computer mode annunciations are displayed on the engine status display (EICAS).

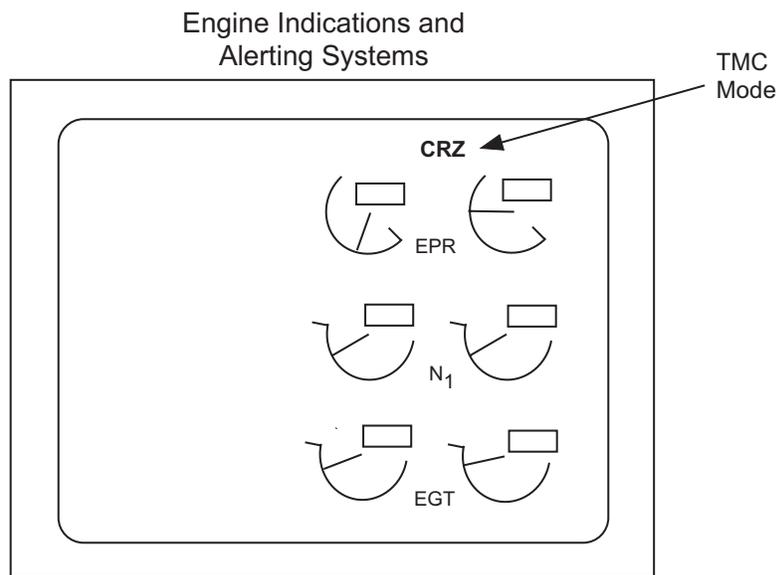
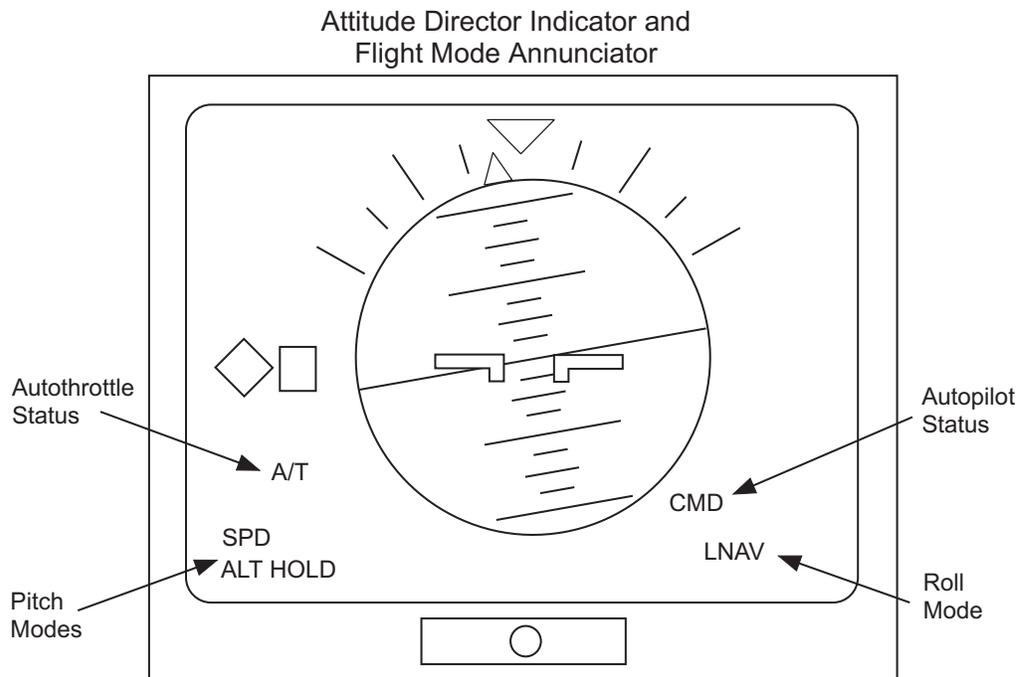


Figure 7-3. Location of mode annunciations in the cockpit

Pitch Modes

A variety of pitch, or vertical, modes exist. These modes are part of the autopilot (guidance) as well as the Flight-Management-Computer (navigation) functions. Table 7-1 describes the pitch modes that were documented in the study, and their corresponding formats on the Flight Mode Annunciator (FMA) display.

Table 7-1. Pitch Modes and Display Format

Pitch Modes	FMA Display Format
Takeoff	TO
Altitude Hold	ALT HOLD
Vertical Speed	VS
Flight Level Change	FLCH
Speed	SPD
Vertical Navigation	VNAV
Go Around	GA
Glide Slope	GS
Arming of “Approach”	GS - in white text

Several additional vertical modes exist. For the purposes of this research, data on the Flare (‘FLARE’) mode were not meaningful because they were associated only with automatic (Category 2 and 3) landings, and such landings were not observed during the field study. With respect to submodes of the “Vertical Navigation” mode, we captured only those submodes that are requested manually (“Descend Now” and “Speed Intervene”). Transient modes such as “Vertical Navigation-Speed” (‘VNAV-SPD’) and “Vertical Navigation-Path” (‘VNAV-PTH’) were not systematically recorded, although they can be inferred from the data in some situations.

Roll Modes

Table 7-2 describes the roll, or lateral modes that were documented in the study, and their corresponding format on the Flight Mode Annunciation (FMA) display.

Table 7-2. Roll Modes and Display Format

Roll Modes	FMA Display Format
Takeoff	TO
Heading Hold	HDG HOLD
Heading Select	HDG SEL
Localizer	LOC
Localizer Back Course	B/CRS
Go Around	GA
Arming of “Approach”	LOC - in white text
Lateral Navigation	LNAV - in green text
Arming of Lateral Navigation	LNAV - in white text

Two additional lateral modes exist: “Rollout” (‘ROLLOUT’) and “Attitude” (‘ATTITUDE’). Data on the first were not collected because the “Rollout” mode is associated only with automatic

(Category 2 and 3) landings. The “Attitude” mode is an abnormal condition. Neither mode was used by pilots observed in the field study.

Thrust Management Computer Modes

The Thrust Management Computer (TMC) computes thrust values and limits for the engines. These thrust values are dependent on the thrust mode selected from the Thrust Mode Computer Select Panel (TMCSP). The Thrust Management Computer also controls the autothrottles.

We recorded the reference thrust modes displayed on the engine parameters display—the Electronic Indication and Crew Alert System, EICAS (Figure 7-3). Table 7-3 describes the thrust management computer modes, and their corresponding format on the display.

Table 7-3. TMC Modes and Display Format

TMC Modes	EICAS Display	Description
Takeoff	TO	Takeoff thrust
Takeoff-1	TO-1	Reduced takeoff
Takeoff-2	TO-2	A more derated thrust
Climb	CLB	Climb thrust
Climb-1	CLB-1	Derated climb thrust
Climb-2	CLB-2	A more derated climb thrust
Cruise	CRZ	In-flight cruise thrust
Continuous	CON	Maximum continuous in-flight thrust
Go Around	GA	Go-around thrust

Autopilot, Autothrottle, and Flight Director Modes.

Table 7-4 describes the Autopilot, Autothrottle, and Flight Director ON/OFF status modes that were documented in the study, and their corresponding format on the Flight Mode Annunciation (FMA) display. These represent some of the supervisory modes of the AFCS.

Table 7-4. AFCS Supervisory Modes and Display Format

Autopilot, Autothrottle, & Flight Director Modes	FMA Display Format
Autopilot ON	CMD
Autopilot OFF	Blank indication on display
Flight Director ON	FD
Flight Director OFF	Blank indication on display
Autothrottle ON	CMD
Autothrottle OFF	Blank indication on display

Summary

In this section we have described the components and modes of the Automatic Flight Control System. Four types of modes were documented in the field study: pitch, roll, thrust, and supervisory (flight director, autopilot, autothrottles). They represent the mode-configuration space for this study.

The Operational Environment

The environment in which the AFCS and its modes are situated involves many components. Three are described here: the Air Traffic Control (ATC) environment, the navigational aspect of the flight, and the mandatory and recommended procedures at the airline.

The ATC Environment

Air Traffic Control (ATC) provides the regulatory boundary in which the aircraft must operate. For example, ATC regulations mandate that the aircraft speed should not exceed 250 knots below an altitude of 10,000 feet. In addition, controllers provide real-time directives (called clearances) to achieve separation from other aircraft and sequence departures and arrivals to the airport.

In general, an ATC clearance can be partitioned into three content elements: vertical, lateral, and speed. Each element contains a limited set of clearances that are used by ATC. Tables 7-5, 7-6, and 7-7 describe the categories of the vertical, lateral, and speed elements in a clearance.

Seven common categories of vertical clearances are listed in Table 7-5. The first and last are the initial and final clearances issued by ATC. The rest include a variety of instructions to the crew on how to proceed with the flight.

Table 7-5. Vertical Clearances

Lateral Clearance	Meaning
Cleared for takeoff	Authorized to depart
Climb to	Climb from the current altitude to another (e.g., climb to 7,000 feet)
Expedite climb	Increase the current rate of climb
Descend to	Descend from the current altitude to another (e.g., descend to 5,000 feet)
Cross at	Cross the specified waypoint at the given altitude (e.g., cross Meridian at 25,000 feet)
Cleared for the approach	Authorized to proceed toward the runway
Cleared to land	Authorized to land

Table 7-6 depicts the seven categories of lateral clearances. Three of the lateral categories are identical with vertical categories (cleared for takeoff, cleared to land, cleared for the approach). Such clearances have both vertical and lateral dimensions.

Table 7-6. Lateral Clearances

Lateral Clearance	Meaning
Cleared for takeoff	Authorized to depart
Turn to	Turn to a heading (e.g., 180 degrees)
Direct to	Fly direct to a waypoint (e.g., Atlanta Airport)
Turn to intercept	Turn to intercept a radial (e.g., the 090 degree radial from a navigational aid)
Turn to [approach]	Turn and intercept the approach course to the airport
Cleared for the approach	Aircraft is cleared (authorized) to proceed toward the runway
Cleared to land	Authorized to land

The last lateral element, speed, has four categories involving commands related to the speed of the aircraft. Table 7-7 depicts these.

Table 7-7. Speed Clearances

Speed Clearance	Meaning
Speed change	Change the current speed to the value assigned by ATC
Speed at fix	Upon reaching the designated waypoint, change the current speed to the value assigned by ATC (e.g., 250 knots at Meridian)
Speed limit	Do not exceed or fly slower than the assigned speed (e.g., do not exceed 280 knots)
Resume normal speed	Speed restriction is lifted by ATC; now fly the normal speed

Air traffic controllers are located in different types of facilities, and are responsible for the different geographical sectors through which a flight progresses. Controllers in airport towers give clearances to take off and land. After takeoff, the aircraft is directed by a controller in a Departure Control facility. Departure controllers are responsible for directing the aircraft beyond the airport terminal area (usually to an altitude of 14,000–17,000 feet). At higher altitudes, control is transferred to an Enroute (Center) controller who supervises the aircraft during cruise. Once close to the airport of destination, control is switched to Approach Control, which directs the aircraft to the terminal area and the approach (to the runway). Finally, tower controllers issue landing clearances and are responsible for separation of aircraft during the final approach. These five different ATC control facilities provide different types of clearances, due to the nature of their control tasks.

The Navigation Environment

The second component in the operational environment is the route of flight and the navigation aids in the airway system. The route of flight is usually predetermined, but can change dynamically due

to weather or schedule constraints. Navigation aids, such as glide slope and localizer transmitters that can direct the aircraft toward the runway, are usually available but sometimes may be inoperable and must be replaced by other navigational aids. The status of the navigation environment has a direct relationship to pilot selection of AFCS modes.

The Airline Procedural Environment

The last component of the operational environment is the airline's mandatory procedures and recommended techniques. Each airline has its own philosophy of operation that is translated into a set of policies and procedures. Specifically, many airlines have developed philosophies of operation for modern AFCS systems (Degani and Wiener, 1994). These philosophies are then translated into a set of procedures and recommended techniques that direct pilots how to operate the AFCS.

In summary, the environment in which the aircraft operates is made up of several components: air traffic control, navigation, and the airline's operational environment. Each component has its own separate structure, but some components (e.g., ATC procedures and airline procedures) overlap and interact. We have attempted here to map and categorize these structures. Later, we will use these structures to quantify and understand the relationships between the operational environment and pilots' interaction with the AFCS.

Design of Experiment

The research design called for a field study in which both quantitative and qualitative data would be collected concerning pilot interaction with, and environmental (external) factors relating to, the AFCS. In addition, data on any mode problems detected either during the flights, or expressed by the pilots during informal discussion, were to be collected. The observations were to be performed in a systematic manner—through repeated observations of the same type of aircraft flying into the same terminal areas during the same time of day. Finally, the data were to be collected during day-to-day normal revenue flights.

The data were collected during two trips.

The Northern trip was a series of three flights:

1. Atlanta, GA to Washington DC
2. Washington DC to Cincinnati, OH
3. Cincinnati, OH to Atlanta, GA

The Southern trip was also a series of three flights:

1. Atlanta, GA and to New Orleans, LA
2. New Orleans, LA to Dallas/Fort-Worth, TX
3. Dallas/Fort-Worth, TX to Atlanta, GA

We wanted to avoid possible environmental biases in the data, and hence one trip (Northern) was conducted in part of what is known as the "Eastern corridor," which is characterized by heavy traffic density. The other trip (Southern) was observed on less traveled routes, with the exception of the arrival into Dallas/Fort-Worth. Each trip was observed (replicated) ten times. Since each trip was composed of three flights, the yield was 30 flights per trip. The two trips (Northern and Southern) totaled 60 observational flights in which qualitative and quantitative data were collected.

The methodological objective of the study was to document pilot interaction with the AFCS in a structured way: Trips and flights were replicated. We hoped this would allow us to reduce unnecessary variance in the data. Scheduling of crew observations was also accomplished in a structured way, so that repeated observations of the same crew could be achieved during the first two flights of a trip.

Participants

Participants were all professional pilots with years of experience flying large and complicated aircraft. They were all current (with recent flight activity) and rated by the Federal Aviation Administration (FAA) to fly the Boeing B-757/767 aircraft.

Fifty-seven airline pilots participated in the study. Selection of flight crews occurred without any experimental intervention—i.e., flight crews did not know in advance that they were going to be observed. According to the airline assignment procedure, pilots selected trips according to their preferences and their seniority rank, but we were not aware of any bias toward flying, or not flying, the specific trips that we observed. To avoid any further bias in the subject pool due to preferred dates (e.g., weekends and holidays—dates on which there is a tendency in the airline to use lower seniority pilots—(National Transportation Safety Board, 1990), all flights were conducted during non-holiday weekdays. Because of the uncontrolled assignment of pilots to the 60 flights, three pilots were observed twice.

Dear Sir/Madam.

Thank you for your participation in this study. This study is focusing on the compatibility between the ATC system and the FMS of "Glass" cockpit aircraft. The study is conducted by Georgia Tech for NASA Ames Research Center. With your permission, the jumpseat observer will (1) record ATC clearances that are transmitted on the radio, (2) record CDU/MCP mode engagement, and (3) administer this questionnaire. The data from this study will be handled according to NASA's confidentiality guidelines. Any discussion that will take place in the cockpit during this study is also considered confidential. **Please do not write your name or ID number on this form.**

Seat [F/O, Captain] : _____

Your age: _____

Number of flying hours at the airline (and other Part 121 airlines): _____

Total flying hours: _____

List number of number of flight hours in a glass or FMS equipped aircraft.

1. B-757/767: _____
2. MD-88: _____
3. A-310: _____
4. L-1011: _____
5. Other Airliners: _____
6. Military aircraft with EFIS (e.g., F-18): _____

Years/Months since completion of 757/767 ground school: _____

List number of hours per week that you use a personal computer. _____

Do you program computers?

During descent into the terminal area, I usually disengage V/LNAV when..... (list several reasons):

During the approach, I disengage the Autopilot (CMD) when

Figure 7-4. Questionnaire Form

The following section presents biographical information about the pilots in the study. Using these data we also tested whether there was any bias in the ‘uncontrolled’ assignment to trips and flights.

Questionnaire Data

We collected biographical information from the crews participating in the study by administering a questionnaire to the pilot flying (PF) on each flight. We had several objectives in administering the questionnaire: (1) to obtain biographical information about the participants, (2) to assess bias in (the uncontrolled) assignment of pilots to trips and flights, and (3) to investigate whether any individual biographical data might relate to the way pilots interact with the AFCS. Figure 7-4 is the questionnaire form.

The questionnaire asked for information on total flight hours, flight hours in large commercial aircraft (Part 121 operation), age, flight hours in glass cockpit aircraft, years/month since initial 757/767 ground school, computer experience, and preferred techniques for disengaging modes and the autopilot. The data from the questionnaire were entered into a Microsoft Excel spreadsheet, verified for data entry and comprehension errors on the part of the crews, and then analyzed.

Figure 7-5 depicts the histogram of flight hours (in B-757/767 type aircraft) of the pilots in the sample. The horizontal axis represents pilots’ flight hours at intervals of 500 hours; the vertical axis represents the count (absolute frequency) of the number of pilots in each interval. The different shades identify the ‘seat’ (captain or first officer) of the subjects.

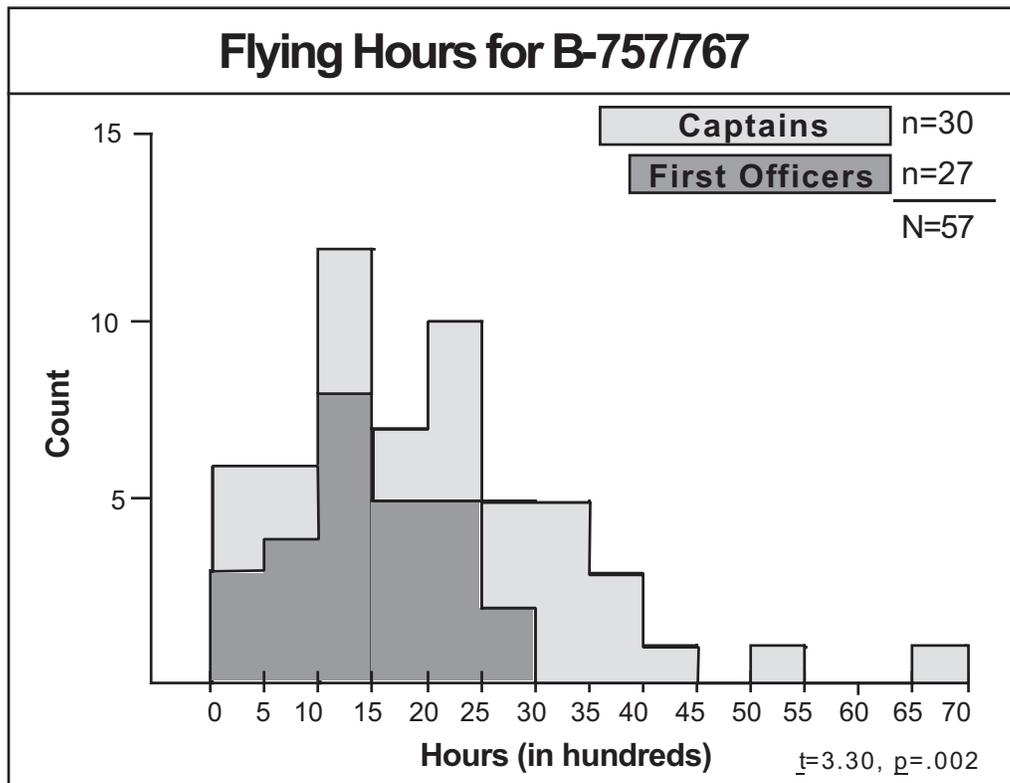


Figure 7-5. Flight hours in the B-757/767 aircraft type

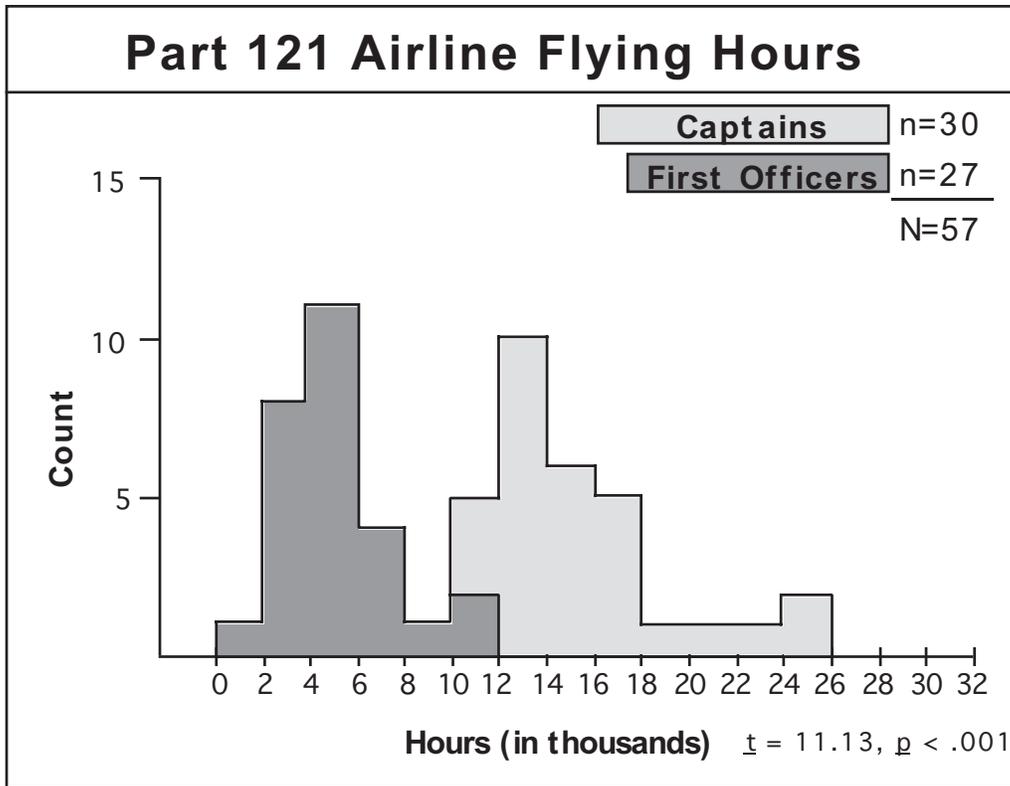


Figure 7-6. Total flying hours in Part 121 carriers

For example, within the range of 1000 to 1500 hours there are 8 first officers and 4 captains—totaling 12 participants. Some ninety percent of the pilots had between 100 and 4000 flight hours. On average, captains had slightly more 757/767 flight hours than the first officers. A student “t” test (two-tailed) was conducted to test for differences between the captains and first officers, with respect to flight hours. The result suggests a significant difference in the mean 757/767 flight hours for captains and first officers: $t(40.7) = 3.40, p = .002$. (Welch-Satterthwaite solution for adjustment of degrees of freedom due to unequal, or heterogeneous, variance). The t-test results are probably heavily influenced by the two outliers, however.

Some ninety percent of the total flight hours in large commercial aircraft (Part 121 airlines) ranged between 2,000 and 25,000 hours (Figure 7-6). With respect to the distribution of captains and first officers in the histogram, a bi-modal distribution was observed ($t(40) = 11.04, p < .001$).

An important datum regarding familiarity with modes is past experience in other aircraft equipped with a full AFCS or Flight Management Computer. In the subject airline there were three other aircraft types that fit into this category: the McDonnell Douglas MD-88 (full AFCS), Airbus A-310 (full AFCS), and Lockheed L-1011 (Flight Management Computer only).

Experience with glass cockpit and FMS-equipped cockpits in other airlines and military aircraft was also analyzed. Figure 7-7 depicts the past experience (aircraft type and hours) of the pilots sampled in other aircraft that have some form of AFCS. The aircraft types are represented on the horizontal axis. On the vertical axis are represented the number of pilots that flew the aircraft, as well as their hours-in-type stacked one on top of the other (the height of the bar indicates the total hours-in-type for the pilots in this sample).

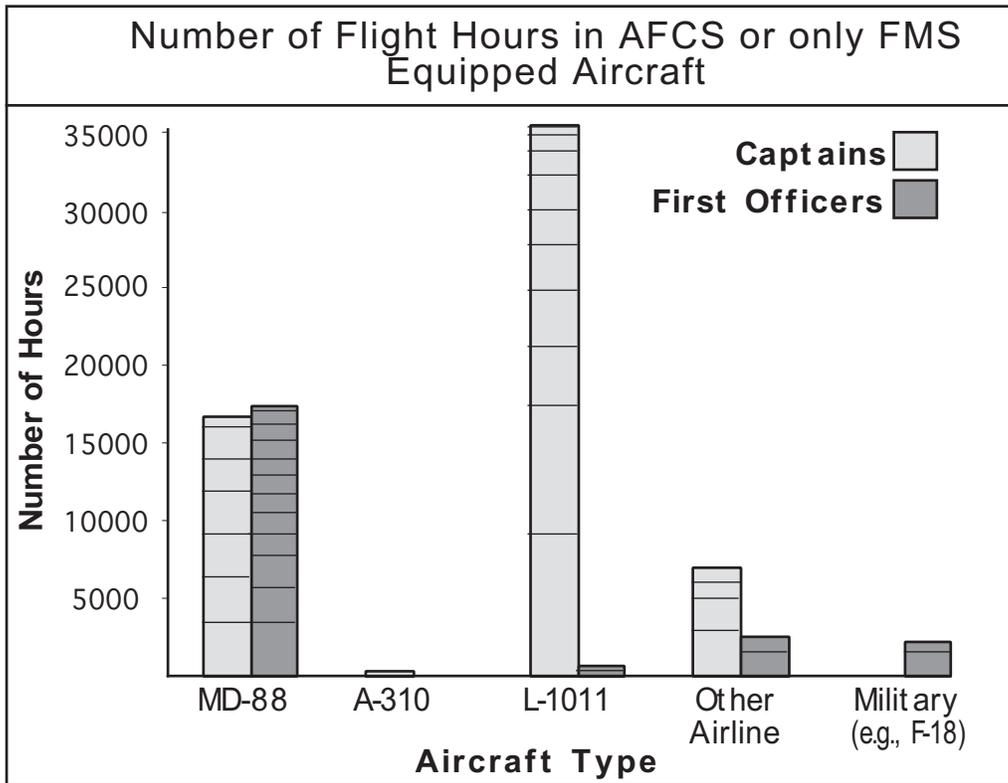


Figure 7-7. Hours in aircraft equipped with an AFCS or only a flight management computer (L-1011)

As mentioned earlier, the assignment of pilots to flights was not experimentally controlled; instead, it was an assignment by the airline and the pilots themselves. The questionnaire data were therefore also used to assess whether biases were present in assignment of pilots to trips and flights.

The two different trips (Northern and Southern) and six different flights were assessed for possible bias using a student t-test and data visualization software, applied to the following variables: age, Part 121 flying hours, total hours, other glass cockpit experience, years since B-767/757 ground school, B-757/767 flight hours, and use of computers. No significant differences were identified.

A more refined assessment of biases was done with regard to glass cockpit experience. Previous research had suggested that about 1,200 hours (roughly a year and a half) of flying experience marks the difference between experienced and less experienced pilots on the AFCS (cf. with Sarter and Woods' partitioning of their data according to this category, 1995b). Analysis of the data according to this dimension (less than, or equal to, and more than 1200 hours of glass cockpit experience) revealed that there were ten pilots in the less-than-1,200-hour category. For the purposes of this study, we operationally defined glass cockpit aircraft as both B-757/767s, and MD-88s. Again, concerned about sampling biases, we checked for differences in the means of this newly derived category. The tests were conducted with respect to the two trips (Northern and Southern) as well as flights (1-6). We did not observe any meaningful differences. In summary, we did not find any indications in the data that suggested systematic bias with respect to assignment of flight crews into the trips and flights.

The third objective of this analysis was to identify any possible factors that might affect pilots' manner of interaction with the AFCS. Based on the above analyses, we focused on two factors: seat (captain or first officer) and level of experience (less or more than 1,200 hours). The captain–first officer distinction appeared to be highly correlated with experience and flight hours. The level of experience, a newly derived variable, was constructed based on earlier findings. We therefore considered these two factors in the multivariate analysis.

Summary

In summary, the analysis of the biographical data provided important information about the pilots in the study. The analysis portrayed a considerable difference between captains and first officers' experience in flying AFCS and Flight Management Computer equipped aircraft. The analysis seemed to indicate that there were no striking sampling biases in the (uncontrolled) assignment to trips and flights. Finally, based on this analysis, we decided that two categories—seat and level of experience—should be considered in the analysis of mode usage data.

Procedure

Before data were collected, several preparatory steps had to be taken. First we had to obtain approval from an airline for conducting in-flight observations. In 1992, we concurrently submitted proposals to an airline and its local pilot's union describing our study's intent and methods, confidentiality provisions, and requirements for implementing the field study. Both proposals were approved.

It was also evident that we would need to augment our technical knowledge of commercial aircraft automated cockpit systems. Conducting meaningful observations of technically skilled individuals interacting with a very complex machine requires in-depth knowledge of the domain under study, as well as effective observational techniques. As is the case in ethnography, the observer must be attuned to detailed interaction between the human and the environment (Agar, 1985), the human and the machine (Christoffersen, Hunter, and Vicente, 1996), and interaction among the crew members (Segal, 1990). In addition to documenting such high-risk and complex systems, the observer should be capable of identifying deviations from normal operations and detecting potential human-machine interface deficiencies (Rochlin, La Porte, and Karlene, 1987). To address these requirements, several further measures were taken.

The observer (also a private pilot) participated in the complete Boeing B-757/767 pilot training course. Next, a preliminary field study was conducted in the winter of 1993 and mode transition data were collected during ten flights. The objectives of this preliminary study were to test the efficiency of the observation procedures and the suitability of one statistical approach (regression analysis). These insights were used in developing the method and analysis techniques for the actual field experiment.

The full field study was conducted during the summer of 1993, and was intended to provide qualitative and quantitative insights into actual pilot interaction with AFCS modes.

The observer met the crew some 30 minutes before the flight and informed them about the study, the type of observation, and data recording required. During this initial meeting he presented to the crew a letter from the Vice President of Flight Operations and the pilot union describing the purpose of the study and authorizing the data collection. The crew had the privilege of declining the presence of the observer (none did, however).

Seated in the jumpseat of a Boeing B-757, located behind the captain and first officer seats, the observer recorded two types of data: (1) pilot interaction with the AFCS; (2) aircraft state and environment data. In addition, the observer documented cases in which the crew engaged a mode in a way that was different from standard techniques and procedures, as well as specific cases of mode ambiguity and resulting confusion. Analysis of one such scenario is described in Chapter VIII.

AFCS Data

The observer collected information on pilot interaction with the AFCS during the climb and descent phases of flight. AFCS modes and some of their associated reference values were recorded from within the cockpit.

Several operational definitions were used to direct the data collection. They represent the level of description (granularity) that we believed was suitable for the objectives of the field study, and were operationally feasible to collect. We focused on the following AFCS elements: (1) modes, (2) transitions, and (3) reference values.

The AFCS provides guidance and navigation commands to the aircraft flight control along two axes: lateral and vertical. Various guidance and navigation modes exist in each axis. In addition, the autothrottles and their Thrust Management Computer (TMC) are also part of the AFCS, having their own set of modes and reference values. In general, the combination (vector) of the three determines the mode configuration of the AFCS. The status of related components—autopilot and Flight Director ON/OFF status— are also part of this vector. We have described the modes of the AFCS that were to be collected in the previous sections (Vehicle).

We operationally defined “mode transition” as the request and/or engagement (but not necessarily activation) of a different mode than the present mode. The only mode transitions we did not record were the transient “Altitude Capture” and “Altitude Hold” modes. However, when “Altitude Hold” mode was the active mode and another mode was requested, the “Altitude Hold” mode was recorded. Similarly, automatic transition of the autothrottles to “Go-around” mode was also recorded. We also recorded requests for modes such as “Approach” and “Lateral Navigation” that transition automatically when guarding conditions are met.

For the purpose of data collection, we defined “reference value” as changes in the parameters of the active mode configuration. While it was impossible as a practical matter to collect every speed, heading, or altitude value that was entered, we tried to capture as many of these data as possible. In particular, we collected reference values associated with the “Vertical Navigation” and “Lateral Navigation” modes because they have a significant effect on system behavior. Some of these reference values are entered via the Control Display Unit into the Flight Management Computer.

Aircraft and Environment Data

A variety of variables external to pilot interaction with the AFCS were also recorded and entered into the data set. With respect to the state of the aircraft, we documented the aircraft altitude and its location in space by recording the bearing and distance from the airport. This was done every time the pilot changed the mode configuration of the AFCS.

Several components of the environment were collected. First, every clearance that was transmitted to the cockpit was recorded. This information provided both the clearance content and the ATC facility from which it was broadcast. Second, we noted the phase of flight during which a mode transition occurred. Third, we recorded any unique weather situation that may have contributed to a specific mode selection.

Several pieces of information regarding the flight itself were also documented. We recorded the trip (Northern, Southern), the flight number (1–6), and the rank (or seat) of the pilot who was manipulating the AFCS (first officer or captain). In addition, the pre-programmed flight route (flight plan), the Airport Terminal Information Service (ATIS) data, and Pre-Departure Clearance (PDC) were recorded for each flight. All of this information (especially the meteorological and route information) has potential bearing on selection of AFCS modes (e.g., takeoff thrust setting).

A considerable amount of information about mode usage was collected during these 60 flights. Two main types of data were recorded—pilot interaction with the AFCS, and the state of the aircraft and the operating environment. In the following sections, Apparatus and Data, we briefly describe how the information was collected and then organized.

Apparatus

For data collection, two types of apparatus were used: a data collection sheet and a tape recorder.

A data sheet was used for manually recording events in the cockpit (Figure 7-8). The data sheet had two sections: a header, and a body of text. The header contained general information regarding the flight such as the pilot flying (captain or first officer), departure and destination airports, and the Airport Terminal Information Service (ATIS). The body of the data sheet was made up of columns and rows. Columns represented the variables of interest (e.g., time, altitude, ATC clearance, autopilot status, pitch mode, roll mode). Rows represent the status of all of these variables immediately after an event (e.g., mode transition) occurred.

FL

Date: **11/3/33** PF: **(P)** Location: **DEA-CUS** A/C#: **650 (757)** Crew:
ATIS ① ^{1500Z} **25000 V9 2100** ^{190/10 617} ²⁰⁰³ **18L/R**

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Time	Altitude	Distance	Clearance	A/P	F/L	T/M/C	A/T	A/T state	Pitch	pitch state	Roll	roll state
2	3:30	390	20 270 HMM	. PP ↓ 350						VNAV		LNAV	
3	4:01	—	106 272 FLN	. → FLN → CUS							DES ABL		
4	02	→ 380	93	. 70 E FLN 240						FLCH	200 CR-PAS		
5	—	—	83	. 15 FLN-70				270		↓	2 SPD BRK		
6	05	250	02 FLN/70	. ↓ 230 FLN 02 12000						V/S	-1,000		
7	10	230	35 211 FLN	. ↓ 230 FLN 02 12000						V/S	-2000		
8				. 123.57 CUS APP									
9	13	170	14 270 FLN	. ↓ 1200 EXP 18L									
10	16	12	21 330 CUS	. ↓ 9000 PP.									
11	17	40	20 —	. ↓ 300 RECORD FROM ANY COM.									
12	18	11.8	14 —	. ↓ PP 6000									
13			h —	. 50R						FLCH/V/S-1900			
14	20	8.5	020W CUS (APP)	. ↓ 4000						FLCH			
15	22	40	1010 BASE 10N	. ↓ 270 180k						HOLD			
16	23	40	1010 FINAL	. ↓ 200 180k THRU FRASE									
17	25	2.7	3 170 FR 220	. 18L CLR TO LAND									
18		1.3											
19		1.0											
20		(1.0)		. LAND 4:28									
21													
22													
23													
24													
25													
26													
27													

Figure 7-8. Data Sheet used by the observer

The tape recorder was a Radio Shack model 14-1154 (costing about \$40) which was connected to the audio panel next to the jumpseat. Radio transmissions on VHF1 —communications with ATC—were recorded from takeoff to cruise and from descent to touchdown. The recordings were used during data entry to verify the accuracy of the ATC clearances that the observer noted during the flight.

The data sheet and audio recording of clearances allowed us to document in detail the many variables associated with mode usage in this complex system. The instruments provided relatively reliable data about what happened in the cockpit. Other devices, such as video recorders, laptop computers, and multi-channel tape recorders, might also have been used (and these options were indeed investigated). However, they had to be ruled out—the first two because of airline policies, and the latter because of cumbersome and complex operation. We had the intuition that a complicated recording device in a complex and demanding environment would hamper, rather than support, collection of meaningful data.

Data

Information about events in the environment-human-machine system, recorded over 60 flights via the data sheet and the tape recording, was coded into a spreadsheet file and represented. The file has 1,665 records. Each record describes a new event in the environment-human-machine system. The event can be a change in the environment state (an ATC clearance), in the user action (a mode or reference value change), or within the machine (an automatic mode transition). Any such event was recorded with all corresponding variables such as the aircraft altitude, location in space, AFCS status, and modes. Figure 7-9 is a small portion of a data file, corresponding to the flight documented in Figure 7-8.

Location	time	Altitude	ATC_Fac.	Bearing	Dist.	Airport	Clearance	A/P	F/D	A/T	TMC	Pitch Mode	Roll Mode
DC-CVG	15:58	39000	enroute	96	142	KDCA	pilot descretion to FL 350	on	on	on	CRZ	VNAV	LNAV
DC-CVG	16:01	39000	enroute	100	130	KDCA	direct Falmouth (FLM) direct Cincinnati	on	on	on	CRZ	VNAV(DES_NOW)	LNAV
DC-CVG	16:02	38000	enroute	100	120	KDCA	descend and maintain FL 230, FLM at 1	on	on	on	CRZ	VNAV(DES_NOW)	LNAV
DC-CVG	16:03	37000	enroute	100	107	KDCA	cross 70 East FLM at FL 240	on	on	on	CRZ	FLCH	LNAV
DC-CVG	16:05	25000	enroute	101	100	KDCA	descend and maimatin FL 230; FLM at 1	on	on	on	CRZ	VS	LNAV
DC-CVG	16:13	17000	approach	127	38	KDCA	descend and maintain 12000 expect 18	on	on	on	CRZ	VS	LNAV
DC-CVG	16:16	12000	approach	150	21	KDCA	descend and maintain 9600 at pilot descretion	on	on	on	CRZ	ALT_HLD	LNAV
DC-CVG	16:17	12000	approach	150	20	KDCA	right turn to 360 degrees, vectors for final	on	on	on	CRZ	ALT_HLD	HDG_SEL
DC-CVG	16:18	11800	approach	150	14	KDCA	descend and maintain 6000 at pilot descretion	on	on	on	CRZ	FLCH	HDG_SEL
DC-CVG	16:18	11700	approach	150	12	KDCA	turn 5 degrees to the left	on	on	on	CRZ	FLCH	HDG_SEL
DC-CVG	16:20	8500	approach	90	10	KDCA	descend and maintain 4000 feet	on	on	on	CRZ	FLCH	HDG_SEL
DC-CVG	16:22	4000	approach	30	12	KDCA	left turn to 270 degrees, 180 knots	on	on	on	GA	ALT_HLD	HDG_SEL
DC-CVG	16:23	4000	approach	15	12	KDCA	left turn 200 degrees, 180 knots until	on	on	on	GA	ALT_HLD	HDG_SEL
DC-CVG	16:24	4000	approach	360	10	KDCA		on	on	on	GA	GS	LOC
DC-CVG	16:25	2700	tower_land	360	7	KDCA	CLEAR TO LAND 18L	on	on	on	GA	GS	LOC

Figure 7-9. The structure of the data file

Two types of data are contained in the data set: (1) pilot interaction with the AFCS; (2) aircraft state and environment data. With 28 original variable fields and 1665 records, some 46,000 pieces of information are contained in the data set.

Field studies present considerable challenges to the process of data collection: In each phase—observations, recording, coding, and organization of the data—there are threats to the integrity of the collected information (Bateson, 1972, xviii). Due to the methodology used in this

study, we had concerns about several aspects of the data: 1) accuracy of observations, 2) accuracy of data entry, and 3) coding. To address these concerns we took the following steps:

- An airline pilot who was rated on the Boeing B-757/767 verified the accuracy of data. He was given both the data sheets (raw data) and printouts of the spreadsheet and asked to do the following: 1) identify any mismatches between the data sheet and the printouts; 2) based on his experience on the B-757/767 (some 1,000 hours), note any mode transitions that appeared different than normal, did not make sense, or any apparent data omissions. Mismatches were discussed, negotiated, and resolved.
- Another experienced pilot, familiar with the AFCS, verified the transformation of text into categorical data. His task was to make sure that the categories were reasonable and that there were no data entry errors in the transformation from text to numerical values.

With these two steps, some of our concerns were relieved. We do not argue here that either the data accuracy or our decisions regarding coding methodology and categorization of information were perfect. Nevertheless, we begin our analysis with confidence about the overall integrity of the data.

RESULTS

A major objective of this analysis was to describe the various mode configurations, and transitions among them, used by line pilots during normal revenue flights. Another key objective was to identify the factors external to the machine that constrain the pilot in making mode selections, and analyze their pattern of interaction. In general, our intent was to understand the features of the machine and the operating environment that influence the selection of mode configuration. The data set described in the previous section allowed us to perform several types of analysis to address these objectives. Two are described here: descriptive analysis and multivariate statistics.

Descriptive Analysis

The modeling framework described in the previous chapters provides a map of the human-machine territory. The field data, on the other hand, provide us with information about the specific paths users have taken in this territory, in terms of mode selections. The combination of the two provides a template for understanding human-machine interaction.

We employ two related representations for describing the paths users take in this system: *mode occupancy* and *mode transition diagrams*. Similar representations have been used in previous studies in human-machine systems to describe trajectories of interactions (Miller, 1985; Miller, Jagacinski, Nalavade, and Johnson, 1982; Moray, Lootsteen, and Pajak, 1986).

Mode Occupancy-All Flights

In this analysis we were interested in identifying which mode configurations of the AFCS were actually occupied. The analysis summarized all the mode configurations in the entire data set.

The primary mode configurations of interest were the pitch and roll modes of the AFCS. There are nine pitch modes and five roll modes. For this analysis, each mode configuration of the AFCS is a vector containing these two elements.

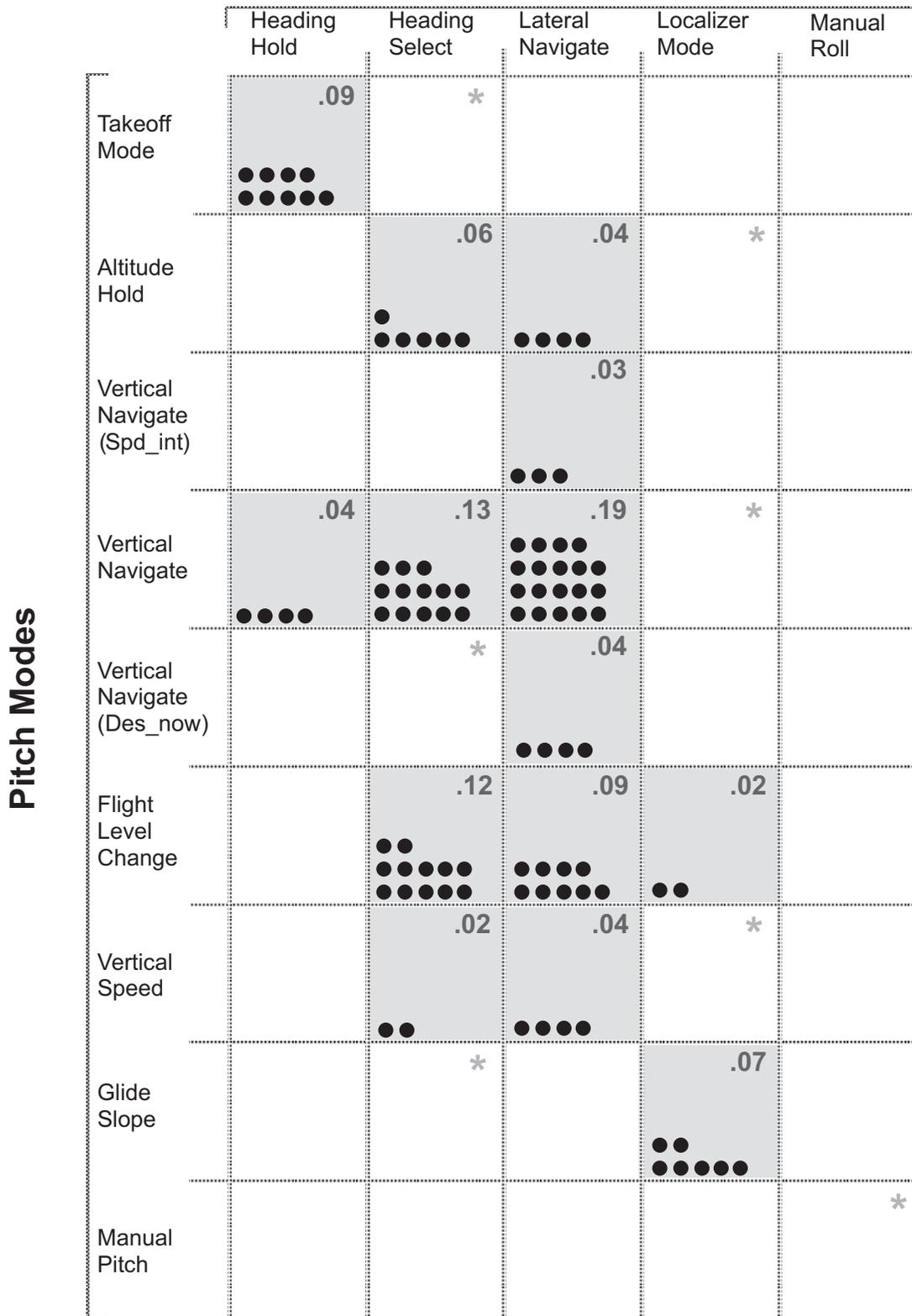
In addition to pitch and roll, other possible dimensions, such as autopilot (on, off), Flight Director (on, off), and autothrottle (on, off) states may be added, but this will lead to a hyperspace representation. In most cases the autopilot, Flight Director, and autothrottles were “on; therefore, representing these functions would not have added much information. To deal with such situations,

we represented occasions when the crew disengaged the autopilot and autothrottles and flew without reference to the Flight Director. We added a category called “Manual Roll” and “Manual Pitch” modes to describe this unique AFCS configuration.

For each mode configuration, we noted the relative frequency of occupying that combination of pitch and roll. To obtain the frequency, we tallied every transition into a given mode configuration and divided it by the total number of transitions in the data set (889). We describe only the cell occupancies that contain a occupancy of 2 percent or more (Casner, 1994; Miller, et al., 1982).

Figure 7-10 provides a two-dimensional description of the mode configuration space of the AFCS. The focus is on the pitch and roll mode combinations. On the horizontal legend (columns) are the roll modes; on the vertical legend (rows) are the pitch modes. Since the status of the AFCS in this analysis is described as a vector of both pitch and roll modes, each cell in the table indicates such a combination. The small (cannon) balls in each cell provide a redundant graphical depiction of this value. An asterisk denotes that the relative occupancy is less than two percent ($0 < * < .02$).

Roll Modes



note: $0 < * < .02$

Figure 7-10. The mode configuration space (pitch and roll) of the AFCS

Taken as a whole, Figure 7-10 provides a map of the AFCS mode-configuration space. Superimposed on this map are the relative frequencies of transitioning to, and occupying, a given mode configuration. This space is a function of the system's behavior and is fixed. Nevertheless, we recognize that there are dynamic restrictions on which portions of this space can be occupied at any time.

The mode configurations in Figure 7-10 describe the 45 functional modes of the AFCS. In addition, the figure also indicates the supervisory modes selected by the crew. At one extreme, the crew may decide to fly the aircraft entirely by manual means (autopilot and autothrottle are disengaged, and the pilot is flying *without* reference to Flight Director guidance). In this case, the "Manual Roll" mode and "Manual Pitch" modes are engaged (lower-right corner of the table). At the other extreme, the crew may decide to fly the aircraft entirely by automatic means (autopilot and autothrottle are engaged, and the aircraft follows the guidance from the flight-management-computer). In the latter case, the "Vertical Navigation" and "Lateral Navigation" mode combinations are engaged. Other mode combinations, such as "Flight Level Change"/"Heading Select" correspond to semi-automatic modes in which the pilot specifies in real time the navigation of the aircraft.

Two observations can be made from this table: (1) only about 50 percent of the pitch/roll mode configurations are occupied (at the 0.02 level), and (2) heavy occupancy is found in three mode configurations: "Takeoff Mode"/"Heading Hold," "Flight Level Change"/"Heading Select," and "Vertical Navigation"/"Lateral Navigation." The first mode configuration is associated with a mandatory procedure in the cooperating airline—takeoffs must be performed by engaging this mode configuration. The second is a semi-automatic mode configuration in which the pilots can respond quickly to the type of air traffic control demands prevalent in the terminal area; it is also a recommended mode configuration (per standard operating procedures) when the aircraft is below 10,000 feet. The third mode configuration is a fully automatic mode combination that provides fuel efficiency and navigation accuracy.

Mode Transitions

The previous analysis of mode occupancy describes both the full mode configuration space of the AFCS and the actual configurations used by pilots in the study. However, the representation format (Figure 7-10) conceals a critical piece of information—the transition between these pitch/roll mode configurations. In the next analysis, we are interested in identifying the pattern of the transitions in the AFCS mode configuration space and the factors that restrict its size.

As we noted in the literature review of Finite State Machine models, the state transition diagram provides a powerful representation of the sequence of interactions between the user and the machine. The description is characterized in terms of the states and the transitions in human-machine interaction. We use this representation to highlight and bring to the surface the pattern of interactions between the crew and the AFCS in terms of mode configurations and transitions.

A transition diagram is made up of two elements: a node and an arc. The mode configurations of the AFCS form the nodes of the mode transition diagram. Based on the previous analysis, we considered only those mode configurations with at least two percent occupancy. The arcs indicate the direction and magnitude of transitions between the nodes. The magnitude of each arc indicates the relative frequency of making the transition. To obtain the relative frequency, we have used the same procedure as for occupancy frequencies.

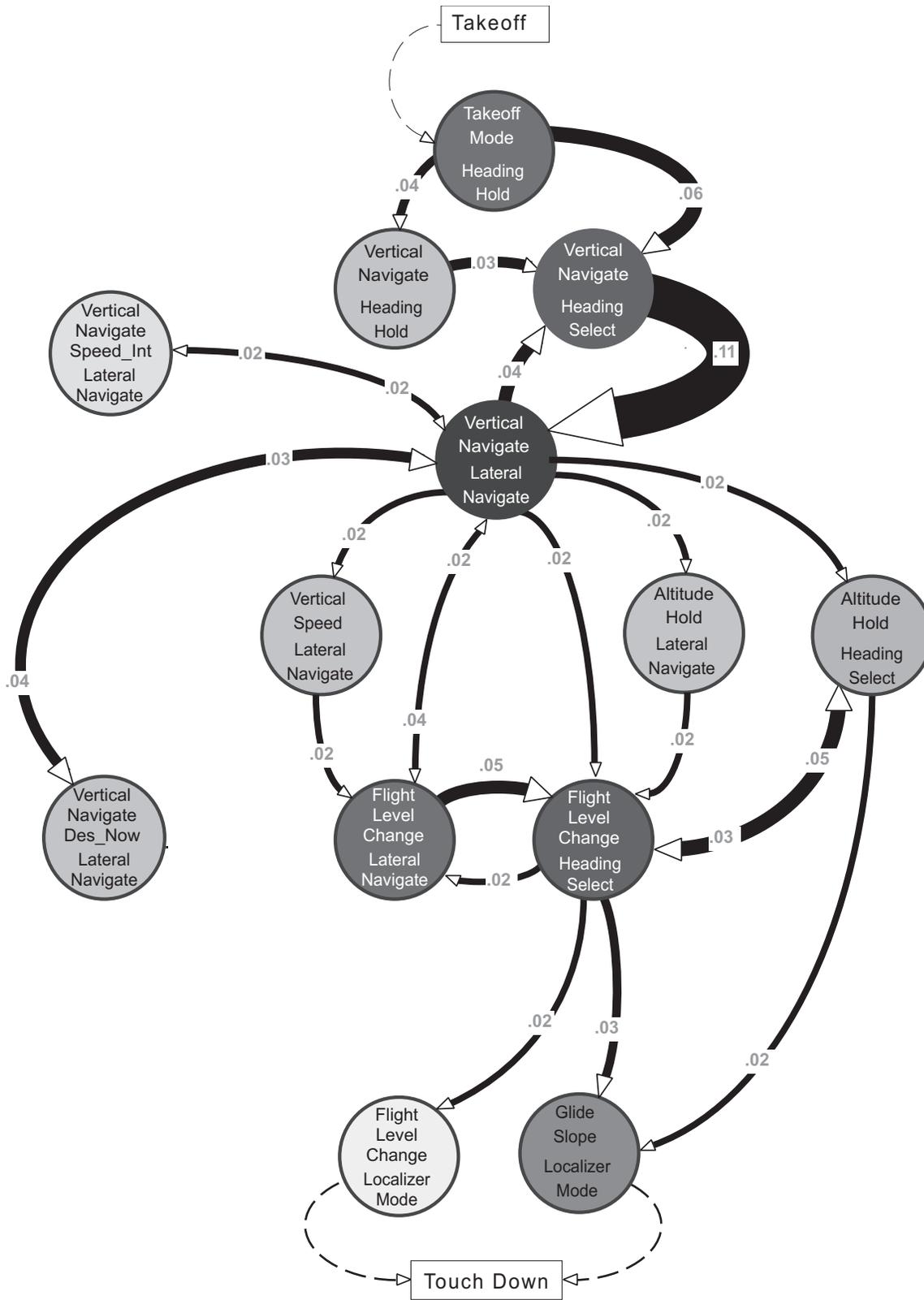


Figure 7-11. Mode transition diagram

Figure 7-11 presents the AFCS mode configurations and the transitions among them. The shading of the nodes corresponds to relative frequencies of occupancy from Figure 7-10. Between the nodes are the transition arcs, which connect the mode configurations and indicate the flow of transitions in the network. The width of the arc provides a redundant graphical depiction of the relative frequency of transition.

When an arc between two nodes has two arrowheads and two values, it indicates a two-way transition, and the values correspond to the closest arrowhead direction. For example, consider the transition between “Vertical Navigation”/“Lateral Navigation” and “Flight Level Change”/“Lateral Navigation.” The arc has two arrowheads and two values are depicted on it. The relative frequency of transition from “Vertical Navigation”/“Lateral Navigation” to “Flight Level Change”/“Lateral Navigation” is 0.04; the reciprocal transition (from “Flight Level Change”/“Lateral Navigation” to “Vertical Navigation”/“Lateral Navigation”) is 0.02. Finally, broken lines shows the initial transition from start of flight to the “Takeoff Mode”/“Heading Hold” mode combination, as well as the final transitions from “Flight Level Change”/“Localizer Mode,” and “Glide Slope”/“Localizer mode” to touchdown.

Figure 7-11 describes the possible transitions from one mode configuration to another—the mode configuration trajectory in the AFCS. Transitions may be initiated by the pilots (manual transitions) or by the control mechanism (automatic transitions). While the mode configuration space is fixed, the set of admissible mode configurations is dynamically changing. For example, the transition to the “Glide Slope” mode can occur only after the glide slope signal is received and processed by the AFCS. Any attempt to manually engage this mode—while in cruise, for example—will be futile. Therefore, we see that during the course of flight the set of admissible mode configurations constantly changes—expanding and contracting as internal and external events take place. This set, nevertheless, is always a subset of the full mode configuration space (Figure 7-10), which is fixed.

The transition diagram combines two features: mode configurations and mode transitions. It shows the possible paths that pilots use to traverse the AFCS mode-configuration space. One objective of this analysis was to identify the constraints on the flow of transitions. We hypothesized that this flow would reveal the factors behind the pattern of interaction with the AFCS.

In graph theory, there is a concept called the “planar graph” (Clark and Holton, 1992). A planar graph is a particular diagram which can be drawn on a plane so that no two arcs intersect geometrically except at nodes at which they are both incident (Nishizeki and Chiba, 1988). Planar graphs describe a property of a diagram and have many practical applications. One recent practical application is the use of planar graphs in the design of electronic chips, so as to minimize the number of layers in a chip.

In constructing Figure 7-11 we have attempted to minimize arc intersections. In fact, the diagram is a planar graph (note, however, that we omitted transitions below 0.02). When this was done, an unexpected organization emerged (Boardman, 1995; Resnick, 1994): The emergent pattern of nodes and arcs that satisfied the planarity rule was organized according to the phases of a flight.

During takeoff the initial mode configuration is “Takeoff”/“Heading Hold.” During the initial climb the mode configuration is “Vertical Navigation” with either “Heading Hold” (remnant of the takeoff procedure) or “Heading Select.” Engagement of “Vertical Navigation” usually occurs at 1,500 feet above the ground and is a highly recommended technique at the airline.

Following initial climb, the climb and cruise phases are characterized by selection of “Lateral Navigation.” Per recommended technique at the airline, flight crews attempt to transition into the

“Lateral Navigation”/“Vertical Navigation” mode configuration as quickly as possible. This is a fully automatic mode configuration that provides economical as well as workload advantages (Casner, 1994).

The “Vertical Navigation”/“Lateral Navigation” mode configuration is the apex in the network and serves as a pivot. Once in the descent phase, the “Flight Level Change”/“Lateral Navigation” mode configuration is heavily used. This is a mode configuration in which horizontal navigation is fully automatic, yet the vertical aspect is controlled semi-automatically. That is, the crew responds immediately to ATC vertical or altitude clearances (e.g., descent to 17,000 feet) while still maintaining the preprogrammed lateral route of flight (expected response time is less than 20 seconds—Casner 1994, p. 585). Once the aircraft is at lower altitudes where immediate heading and altitude clearances are given by ATC, the “Flight Level Change”/“Heading Select” mode configuration is frequently used. In this semi-automatic mode configuration, the crew can immediately respond to ATC heading and altitude clearances in the vicinity of the landing airport. It is a recommended technique at the airline to engage this mode configuration when ATC issues short-term heading and altitude clearances. Similar to the fully automatic mode, the “Flight Level Change”/“Heading Select” mode configuration is also a pivot in this graph.

In the approach phase, a gradual transition into the “Glide Slope”/“Localizer” mode configuration was observed. Nevertheless, other transitions into touchdown were also observed. One transition—via the “Flight Level Change”/“Localizer” mode configuration—is potentially dangerous at low altitudes and has contributed to many reported incidents and one accident (Indian Court of Inquiry, 1992). This specific transition occurred due to the lack of the glide slope navigational aid at one airport’s runway (New Orleans).

In summary, the mode transition diagram shows the possible paths that pilots use from takeoff to touchdown. The structure of the graph reveals that the phase of flight is an important factor in pilot interaction with the mode configuration space of the AFCS. In addition, the phase of flight corresponds with the dynamic change of the available mode configuration space of the AFCS.

Section Summary

The mode occupancy and transition analysis allowed us to describe the flow of transitions among the possible mode configurations of the automated flight control system of the Boeing B-757/767 aircraft. This descriptive analysis provided a view from *outside* the AFCS of how pilots interact with this system. We stress “outside,” because, unlike the Ofan model of the control mechanism, the mode occupancy and transition analysis says very little about the internal transitions and behavior of the AFCS.

Nevertheless, the analysis identified the mode configurations commonly used by flight crews and the flow transitions among these. It revealed a unique pattern that appeared to be constrained by three factors: (1) the possible (legal) mode configurations of the AFCS; (2) the standard operating procedures and recommended techniques at the airline; (3) the phases of flight—takeoff, climb, cruise, descent, approach, and land. As a whole, the analysis revealed how pilots traverse the mode configuration space of the AFCS. It also showed that the set of admissible mode configurations changes dynamically as the flight progress and external events occur.

Multivariate Analysis

In the previous section we identified three factors that constrain pilot interaction with the AFCS. It is quite clear, however, based on the literature review (e.g., Casner, 1994; Wiener, 1989) that these are not the only factors.

In the current section, therefore, we attempt to use other statistical methodologies in our search for factors that prompt mode transitions. In addition, we are interested in identifying the pattern of interaction between these factors and mode selection.

Two types of multivariate statistical tests are employed—regression and canonical correlation. Although both tests are based on the General Linear Model (GLM), there are several differences with respect to their assumptions and the quantification of results. The two statistical tests allowed us to identify the factors that prompt mode transitions. The combination of two statistical tests allowed us to see if there was converging evidence for this. In addition, the canonical correlation highlighted patterns of interaction between these factors and mode selections.

It is important to note that the statistical models were not used for prediction purposes. We used them to highlight the factors that affect mode transitions, as well as those that do not. In other words, the purpose of the statistical analysis was to reduce the large space of possible factors into a smaller set of meaningful factors (Simon and Gregg, 1979).

The type of information in the study data set is not easily amenable to quantitative statistical analysis. The data were collected in the field and not in a controlled laboratory; there are many sources of information; there are several types of data (continuous and categorical); and many factors are not independent of one another. Using such data for quantitative analysis requires several preparatory procedures. We discuss these procedures in the following section (as well as separately for each test).

For the purpose of the multivariate analysis we divided the variables in the data set into two subsets—dependent and independent variables. The independent variables were all the factors external to the AFCS such as altitude, ATC clearance, Phase of Flight, and level of experience in operating the AFCS. The dependent variables were the mode configuration vectors of the AFCS. Table 7-8 provides a list of the dependent and independent variables used in this statistical analysis. For each variable, we list its corresponding states or categories within parentheses. Categories identified with an asterisk denote the reference category for each variable. These reference categories are discussed in the next section.

Table 7-8. Dependent and Independent Variables for Statistical Analysis
(asterisk denotes a reference category)

Dependent Variables	Independent Variables
Flight Director (off [*] , on)	Trip (Northern [*] , Southern)
Autopilot (off [*] , on)	Flight (ATL-DCA [*] , DCA-CVG, CVG-ATL, ATL-MSY, MSY-DFW, DFW-ATL)
Autothrottle (off [*] , on)	Pilot Flying (first-officer [*] , captain)
Pitch Modes (takeoff [*] , manual pitch, altitude hold, vertical speed, flight level change, vertical navigation, VNAV-descend now, VNAV-speed intervene, glide slope)	Experience in AFCS (equal or more than 1200 hours [*] , less than 1200 hours)

Armed Pitch Modes (none [*] , approach, glide slope)	Aircraft Type (B-757 [*] , B-767)
Roll Modes (takeoff [*] , heading hold, manual roll, heading select, lateral navigation, localizer)	Phase of Flight (climb [*] , cruise, descent)
Armed Roll Modes (none [*] , approach, lateral navigation, localizer)	Aircraft Altitude
TMC Modes (takeoff-1-2 [*] , climb, climb-1, climb-2, cruise, continuous, go around)	Distance from Airport
	ATC Facility (tower-takeoff [*] , departure, enroute, approach, tower-land)
	Lateral Clearance (cleared for takeoff [*] , turn to heading, direct to [a waypoint], turn to intercept, cleared for the approach, cleared to land, turn to approach)
	Vertical Clearance (cleared for takeoff [*] , climb to altitude, descend to altitude, cross [a waypoint] at an altitude, cleared for the approach, cleared to land, expedite climb)
	Speed Clearance (speed change [*] , speed at fix, speed limit, resume normal speed)

The data collected during the flights pose some challenges for multivariate analysis, since the values are mostly discrete. Such discrete, or categorical, data require special numerical coding for analysis. For each categorical variable (e.g., phase of flight) a set of indicator variables (sometimes called “dummy” variables) is constructed. This set must be large enough to exhaust the information contained in the original categorical variable. Categorical variables can be dichotomous (e.g., Autopilot is either “On,” or “Off”) or polytomous (having more than two categories). For example, the polytomous variable “Phase of flight” contains three categories: climb, cruise, approach. This variable requires a set of two indicator variables in order to capture all the distributional information contained in the original variable. As a rule, coding a categorical variable with j categories requires a set of $j-1$ indicator variables (Hardy, 1993).

The reason there is always one less indicator variable for each category is due to a requirement of the General Linear Model, specifically, the assumption of imperfect colinearity among independent variables (see Neter, Wasserman, and Kutner, 1990, chap. 10). As a consequence, the category that is omitted becomes the reference category. The results of the analysis (e.g., regression coefficients) will be evaluated relative to this single reference category.

Our approach was to use the initial category (e.g., the “cleared for takeoff” clearance) or default category (e.g., the “Takeoff” pitch mode) as our reference categories for each of the variables (Hardy, 1993, p. 10). In this way there was some consistency in selection of reference categories across all variables. This assisted the inference process, in particular when conducting comparisons between variables. These reference categories are denoted in Table 7-8 with an adjacent asterisk (e.g., first officer).

Regression

Regression is one of the most widely used techniques of quantitative analysis. In the area of decision-making research, many authors have shown that regression models are extremely powerful in accounting for the variance in human judgment (Dawes, 1979; Einhorn, Kleinmuntz, and Kleinmuntz, 1979; Goldberg, 1968). Regression techniques can be used to identify the most important factors among the selected set of candidates that contribute to making the judgment. Analysis of the results can point to the marginal contribution of each significant factor, the inter-correlations among factors, and the overall goodness of fit of the regression model. Somewhat similar to the use of regression in judgment research, we employed this approach to identify the factors, or cues, that were involved in making mode selections.

A typical regression analysis attempts to explain variation in a quantitative dependent variable (Y_i), by mapping the relationship of Y to a specific set of independent variables as an additive, linear function. Regression analysis provides a measure of the relationship between a single dependent variable and a set of independent variables. The resulting statistical model and the corresponding analysis of variance table provide further insight into the significance and contribution of each independent variable to estimating the dependent variable.

When independent variables of interest are categorical, we require a technique that allows us to represent this categorical information in quantitative terms. Defining a set of indicator variables for each categorical variable allows us to capture the information contained in the categorical variables, and then to use this information in a standard regression estimation. In fact, the set of independent variables can include any combination of categorical and quantitative (e.g., altitude) information (Hardy, 1993).

The objective of the analysis was to identify those factors that affect the selection of mode configuration. Specifically, the regression allowed us to identify the relationship between the dependent variable (AFCS mode configuration) and the set of independent variables (the aircraft altitude, the phase of flight, ATC clearance, etc.), such that the unexplained variation in the dependent variable was minimized.

Since the dependent variable of interest, mode configuration, is a vector of several components (Pitch and Roll modes, as well as Autopilot and Autothrottle status), a composite single variable had to be constructed. There are several ways one can define this vector. One alternative is to view this vector as defining the level of automation used in the AFCS. We can then regress all the independent variables on this single dependent (“ Y ”) variable—defining the level of automation—and identify the relationship between them.

The pitch/roll mode configuration and autopilot/autothrottle status, were combined into a single ordinal value (the dependent variable “ Y ”). This aggregation of the raw data into a single composite variable is not without some limitations (see the Results section for further discussion of this issue).

The dependent variable was derived from the 45 possible mode configurations that are identified in Figure 7-10. These mode configurations were subjectively ranked according to the level of automation of each of the 45 possible mode configurations. Two subjective ranking criteria were used: (1) the precision with which the automation follows the predetermined flight path; and (2) the level of human-machine involvement in controlling the flight path. In general, the rank provides a subjective estimate of the level of automation employed.

Roll Modes

		Heading Hold	Heading Select	Lateral Navigate	Localizer Mode	Manual Roll
Pitch Modes	Takeoff Mode	29	27	8	20	44
	Altitude Hold	32	31	18	26	43
	Vertical Navigate (Spd_int)	16	13	2	6	34
	Vertical Navigate	15	12	1	5	33
	Vertical Navigate (Des_now)	17	14	3	7	35
	Flight Level Change	30	25	10	21	39
	Vertical Speed	28	23	9	19	40
	Glide Slope	24	22	4	11	37
	Manual Pitch	42	41	36	38	45

Figure 7-12. Ordinal ranking of the 45 mode configuration of the AFCS

Low values were assigned to a combination of pitch and roll modes that were highly automated (e.g., “Vertical Navigation”/“Lateral Navigation” mode was ranked 1). High values were assigned to a combination of modes that were manual (e.g., “Manual Roll”/“Manual Pitch” mode was ranked 45). Figure 7-12 depicts the rankings that were assigned. Dark shades denote high levels of automation and light shades low levels. A dark cross-like pattern can be seen in the middle of the table—pertaining to the high (dark) and moderate shading around the fully automatic modes (“Vertical Navigation”/“Lateral Navigation”).

For the purpose of building and validating the regression model, the records pertaining to the 60 flights were randomly split into two equal-size data sets, each one containing 30 flights. The first, called the *model-building set*, was used to develop the model. The second data set, called the *validation set*, was used to evaluate the reasonableness and predictive ability of the selected model. This was done by applying the selected model (built on the first data set) to the data in the validation set. If the predictive ability of the selected models are within the reasonable confidence intervals, we have validated the model’s usefulness (Neter, et al., 1990, chap. 12).

The validation procedure outlined above is very useful in identifying the stability of the selected model. In a practical sense, the data splitting and validation procedure provide a guard against overfitting the data. In a general sense, they reveal the robustness of the model. This procedure can be done only when there are enough records in the data set, so that the data splitting does not affect the estimated regression coefficients. Since there are some 450 records in each of our two sets, this concern did not apply here.

The regression analysis of the model-building set identified the significant (independent) factors that contribute to reducing the variance around the predicted Y ($R^2_{adj.} = 0.55$, $p < 0.001$). A test of the model using the validation data set yielded a comparable fit ($R^2_{adj.} = 0.64$, $p < 0.001$).

The validity of the regression model refers to the stability and reasonableness of the regression coefficients, the plausibility and usability of the model, and the ability to generalize inferences drawn from the regression analysis (Neter, et al., 1990, p. 438). Statistically, the results of the validation procedure indicate that the selected model is very stable—its predictive ability is high. For the purposes of this research, the results indicate that the model captured some of the more important characteristics of operational environments that affect pilots’ mode selection. The results allowed us to continue our inference with assurance that the selected model was not a case of overfitting.

Once the selected model was validated, we applied this model to the entire data set. Table 7-9 presents the regression results for the entire data set (60 flights).

Table 7-9. Regression Results

R squared = 59.6%		R squared (adjusted) = 59.0%		
s = 6.625 with 889 - 13 = 876 degrees of freedom				
<i>Source</i>	<i>Sum of Squares</i>	<i>df</i>	<i>Mean Square</i>	<i>F-ratio</i>
Regression	56677.4	12	4723.12	108
Residual	38452.1	876	43.8951	
<i>Variable</i>	<i>Coefficient</i>	<i>Stand-Err.</i>	<i>t-ratio</i>	<i>probability</i>
Constant	24.5188	.7416	33.1	≤ .0001
Altitude	-.000231	.0000	-5.86	≤ .0001
Descent Phase	5.06220	.7389	6.85	≤ .0001
Departure	-15.0313	.9817	-15.3	≤ .0001
En-route	-15.0103	1.306	-11.5	≤ .0001
Approach	-10.8755	1.162	-9.36	≤ .0001
Tower-land	-14.8360	1.385	-10.7	≤ .0001
Turn	7.45981	.6660	11.2	≤ .0001
Direct to	-2.17195	.8159	-2.66	.0079
Cleared for the Approach	8.40385	1.471	5.71	≤ .0001
Descend to	2.81426	.6546	4.30	≤ .0001
Cross at	1.18182	1.256	.941	.3471
Expedite	7.32822	1.815	4.04	≤ .0001

Interpretation of the results suggest that 59% of the variance in the level of automation selected by the pilots can be explained by four main factors: (1) the aircraft altitude, (2) the phase of flight, (3) the type of ATC facility supervising the aircraft, and (4) the type of ATC clearance issued by this facility. Specifically, in the phases of flight the “Descent” category is significant. With respect to ATC facility, “Departure,” “En-route,” “Approach,” and “Tower” (at the landing airport) are highly significant.

Of the three elements in a clearance (vertical, lateral, speed), only the vertical and lateral elements directly relate to mode selection. In the lateral element, three clearance categories were significant: “turn to a heading,” “direct to (a waypoint),” and “cleared for approach.” In the vertical element, we also identified three clearance categories: “descend to an altitude,” “cross (a waypoint) at an altitude, and “expedite your climb.” Each of these clearance categories and possible combinations requires a very specific mode configuration. The regression analysis appears to capture such unique consistencies in the data set. Additionally, the frequency of these clearances (with the exception of “expedite your climb”) in each flight is high.

Two of the most commonly used diagnostic methods in regression are analysis of the residuals and normal probability plots.

Ordinarily, direct diagnostic plots for the dependent variable Y are not very useful in multiple regression because the values of the observations are some function of the independent variables. Instead, diagnostics for the dependent variable are carried out indirectly through examination of the residuals. A residual (e_i) is the difference between the observed value (Y_i) and the predicted value (\hat{Y}_i).

However, raw residuals may not be the best approach when the residuals have different variances. A useful alternative is to standardize the residuals by dividing each by an estimate of its own standard deviation. Externally studentized residuals are one type of such standard residual. (The term *studentized* is after the pseudonym of W.S. Gosset, the originator of the “t” test and distribution).

This method has the advantage that each residual can be interpreted as a “t” statistic with $(n-p-1)$ degrees of freedom) for testing whether the residual is an outlier in the regression. The externally studentized residuals plot is constructed by computing the standardized residual of each case and plotting it against the predicted values. If the model is appropriate for the data at hand, the residuals should reflect the properties assumed for the error.

A second diagnostic method is analysis of normal probability plots. Here we plot the raw residual versus its expected value if the distribution were normal. A plot that is nearly linear suggests agreement with normality assumptions, whereas a plot that departs substantially from linearity suggests that the error distribution is not normal. A good measure of normality is the coefficient of correlation between the residuals and expected (normal) values. Figure 7-13 is a plot of the externally studentized residuals versus the predicted values. Embedded in the upper-right corner is a normal probability plot of the raw residuals.

An immediately evident pattern in the residual plots is the repetition of parallel rows of points. The source of this unique pattern is the aforementioned method of data coding. When each independent variable is coded as a set of indicator variables with either 0 or 1, the additive linear function of the predicted values is not continuous. It is most commonly distributed along constant intervals. Additionally, since the dependent variable (Y) is a ranked integer, the residuals reflect this characteristic of the dependent variable.

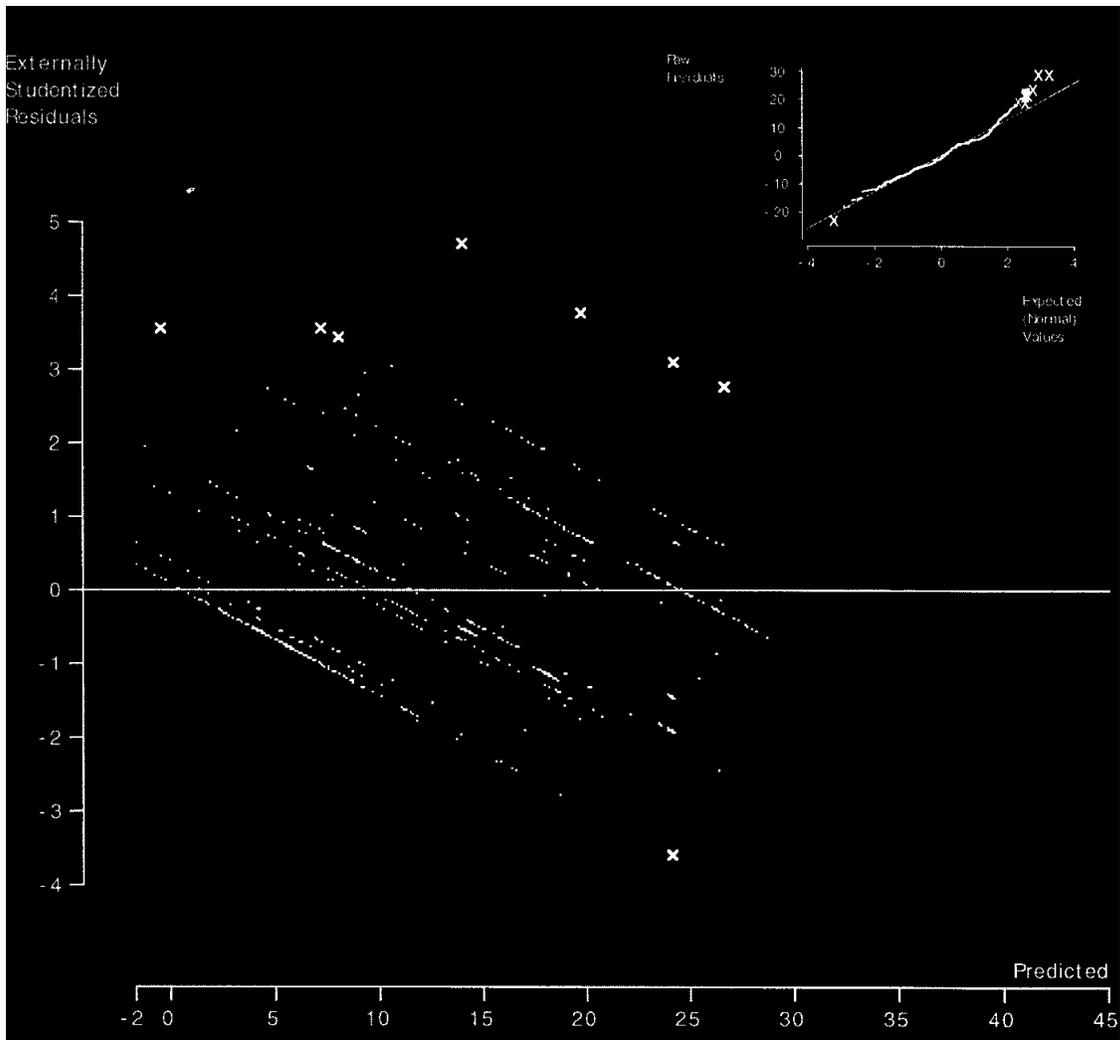


Figure 7-13. Plot of externally studentized residuals vs. predicted Y values (and a normal probability plot)

The distribution of the residuals along a part of the full range (1–45) of the predicted values provides insight about the regression model and the data. The prediction range is between -1 and 30, indicating the range in which the model is effective. The dependent variable, Y, is ranked such that a high level of automation (e.g., “Vertical Navigation”/“Lateral Navigation” mode configuration) is ranked 1, and a low level of automation (e.g., “Manual Roll”/“Manual Pitch”) is ranked 45. We note that in predicting high levels of automation, the distribution of residuals is rather small (about one standard deviation)—indicating high predictive ability. Note the tendency to overfit (due to the ceiling effect) in the 0–5 range of the predicted values. In predicting high to moderate levels of automation (mode configurations such as “Flight Level Change”/“Lateral Navigation”—a rank of 10), the distribution of residuals around the predicted values is large. In predicting moderately low levels of automation (mode configurations such as “Flight Level Change”/“Heading Select”), the distribution of residuals is moderate. The lack of points at the higher ranges of the predicted values (30–45) exist because such levels of automation are rarely used. This can easily be detected by comparing the rankings of cells in Figure 7-12 with the actual cell occupancies in Figure 7-10.

Taken as a whole, the results indicate the model has reasonable predictive abilities at high levels of automation (fully automatic modes) and moderately low levels of automation (semi-automatic modes). The model appears to be less stable at the moderate levels of automation (e.g., the combination of a fully automatic mode and a semi-automatic mode). Indeed, such moderate levels of automation appear to be more dependent on pilots' personalized techniques than on actual demands from the environment. For example, many of the points in this region correspond to pilots' use of the "Speed Intervention" and "Descend Now" submodes of the "Vertical Navigation" mode. Selection of these mode configurations is mostly based on pilots' own preference rather than on external factors.

The normal probability plot of the residuals versus the expected (normal) values provides us with additional information about fit of the model to the data. The coefficient of correlation between the two plotted variables is .985. Controlling the alpha risk at .05, the value obtained indicates that the distribution of residuals (or error terms) does not depart substantially from a normal distribution. This again is another measure for the reasonableness of the model in predicting these levels of automation. However, several outliers on this plot (especially at the top), suggest that these points deserve special attention.

Some authors argue that analysis of the outliers is just as important as the analysis of the model fit (e.g., Tukey, 1977). One way to think about outliers is that their departure from the fit provides additional information about the rest of the data (given no sampling or measurement biases).

In both plots of Figure 7-13, outliers are denoted with an X. Outliers are identified as those externally studentized residuals having a 't' value of more than three standard deviations, and those raw residuals that substantially depart from the regression line in the normal probability plot. Figure 7-13 reveals that given the all possible mode configurations of the AFCS, almost all of the mode selections were at one region, while a very few were located in another. We now can ask why—both with respect to the outliers and the rest of the data.

For example, one extreme outlier represents a situation in which the pilot engaged "Manual Pitch"/"Manual Roll" at 7,300 feet—a very uncommon mode configuration at this altitude. Another represents a case in which the aircraft descended to 3,000 feet without a direct clearance (the aircraft was cleared for the approach at 3,500 feet). Other outliers are associated with using the "Vertical Speed" pitch or "Heading Select" roll modes at very high (cruise) altitudes, or use of a fully automatic mode configuration ("Vertical Navigation"/"Lateral Navigation") at very low altitudes (2,000 feet). Two other outliers are related to a malfunction of the autothrottles that occurred during one flight (they were turned OFF).

These outliers correspond to mode configurations of the AFCS that do not rely heavily on the structure of the environment and the capabilities of the machine. Furthermore, these mode configuration situations are not commonly used by most pilots. It appears, on the contrary, that most of these outlier mode configurations reflect a personalized technique (e.g. use of "Vertical Speed" pitch mode at high altitude to smooth the descent), unique environmental demands (navigating between high altitude thunderstorms), or a pilot's seizing an opportunity to manually hand-fly the aircraft during an undemanding approach.

These observations about pilot's interaction with the AFCS are somewhat contrary to Rosson's (1983) observations about secretaries' interaction with a text editor: Rosson argues that while it may be the case that users tend to develop *predictable* methods, it is not clear to what extent these methods will be most efficient for the task at hand (p. 171). We argue that while these outliers do

indeed exist, it is hard to see them as some consistent *inefficiency* on part of these crews in operating the AFCS.

Finally, we earlier stated that identifying the significant factors was not our only goal in performing this analysis. The process of weeding out the non-significant factors was just as critical. We identified six such non-significant variables: the rank of the pilot flying (captain, first officer), experience in AFCS-equipped aircraft, aircraft type (B-757, B-767), the trip (Northern, Southern), the flight, the distance from airport, and any speed-related clearances.

The regression analysis provided information regarding the factors that affect mode transitions. It allowed us to quantify the relationship between several external factors and the subjective ranking of mode configurations in the AFCS (according to levels of automation). The analysis identified four main factors: the aircraft altitude, the phase of flight, ATC clearance, and the type of ATC facility supervising the aircraft. The predictive ability of the model (measured reduction of variance) was around 60 percent. This moderate value is probably due to the finely grained categorization of the levels of automation in the AFCS mode configuration space (45 levels).

Regression analysis is a flexible and widely used procedure. The advantages of regression analysis are its simplicity and the fact that it is well understood. One disadvantage, however, is the limit on the amount of raw information that enters the model due to using a composite “Y” variable (see Walker and Catrambone, 1993, for a detailed treatment of this issue). Not only is this composite “Y” usually not too easy to interpret, but we also have a genuine interest in each of the variables that makes up this composite. In addition, the use of regression analysis implies the acceptance of the normality assumptions associated with this type of analysis.

With these concerns in mind, we explore in the next section the utility of canonical correlation. Also a GLM multivariate technique, canonical correlation and some of its inference methods address the above limitations directly, and provide additional insight about how pilots interact with the AFCS.

Canonical Correlation

The previous analysis (regression) described the relationship between the ordinal rank of mode configurations based on level of automation, and several aircraft data and environmental factors. However, the analysis format concealed a critical piece of information—namely that there may exist more than one pattern of relationship between the set of external factors and selection of mode configurations. In the canonical correlation analysis, therefore, we were interested in identifying these patterns of interaction between the pilot, the machine, and the operational environment.

Canonical correlation provides a general multivariate methodology. Most other tests, such as the regression analysis employed earlier, are special cases of this methodology. Whereas regression considers only a *single* dependent variable (“Y”), canonical correlation considers a full set of dependent variables (Tatsuoka, 1988).

Canonical correlation is applied when the data can be divided into two subsets for meaningful reasons, such as a set of independent variables and a set of dependent variables. The method finds two vectors of weights (one for each subset), such that the correlation between the linear combinations of these two subsets is high. The correlation provides a measure for the degree of reliable relation between the two subsets.

In computing these two vectors of weights, several solutions are possible (their number is equal to the number of variables in the smallest subset). Each solution provides two vectors of weights—one

for the independent and one for the dependent subsets. This paired sets of weights defines a paired canonical variate. The correlation between the two subsets of the variate is the canonical correlation.

For a given data set, therefore, we may have several canonical variates. Some will have high canonical correlation, indicating a reliable relationship between the subsets, and some will have low correlation indicating less reliable relationships. Each of these paired variates gives us the pattern of relationships between the independent and dependent variables.

Sometimes the pair of variates that provides the largest canonical correlation may provide sufficient information for the purposes of the investigator. However, in many cases we recognize that there may be several other consistent patterns in the data set. These consistent patterns are also important for our understanding of the data. To identify these additional patterns we need to know about the remaining sets of paired variates.

One immediate concern is whether these patterns correlate among themselves. An interesting property of these variates is that all correlations among them are zero, except for the two paired variates. Each paired variate is orthogonal to the others. What this means is that each pattern is completely independent of others. It presents a unique relationship in the data set. And for a given data set, there may be several such unique patterns (Cliff, 1987, chap. 17).

The concept of looking for several patterns in the data set seems counter-intuitive if one is familiar only with approaches such as multiple regression. Regression identifies one pattern of relationship between the independent set of variables and the single dependent variable. But what if other patterns exist? Indeed, they will have lower correlations, but these correlations may still be statistically significant. And finally, in the case of canonical correlation we add more dependent variables—possibly allowing for more patterns to appear.

Our objective in this analysis was to find the most prominent patterns in the data. These patterns between the mode configuration selected by the pilot and all other factors provide very useful information. They tell us about consistent relationships in the data set, and also tell us which variables have no effect on mode selection.

Because of the obvious inapplicability of the normal distribution assumptions to a mostly discrete data set, resampling procedures were employed (Edgington, 1987; Efron and Tibshirani, 1993). Resampling procedures are useful in cases where the sampling distribution is unknown. They are tools for developing *inferential* statistics (e.g., confidence intervals and bias estimators) (Mooney and Duval, 1993, p. 60-62). We used resampling to test the significance level, obtain the standard error, and determine an approximate 95% confidence interval for each canonical correlation.

Resampling methods differ from traditional parametric approaches to inference in that they employ large numbers of repetitive computations to estimate the shape of the statistic's sampling distribution, rather than strong distributional assumptions and analytical formulas. The central idea is that it may sometimes be better to draw conclusions about the characteristics of a population strictly from the sample at hand, rather than making perhaps unrealistic assumptions about the population.

In summary, both resampling- and parametric-based inferencing have the same underlying purpose: Using limited information, to estimate the sampling distribution of a statistic in order to make inferences about a population parameter. The key difference between these inferential approaches is how they obtain this distribution. Whereas traditional parametric inferences relies on *a priori* assumptions about the population distribution (i.e., the central limit theorem), resampling estimates

the entire sampling distribution by relying on the analogy between the sample and the population (Mooney and Duval, 1993, chap. 1).

There are several steps involved in employing resampling techniques for inferencing. In general, the procedure we have used to estimate the significance values, standard errors, and compute the confidence intervals, involves three steps:

1. Define the available sample of data as the “universe” for resampling purposes. The term “universe” refers to the system that is relevant for a single simple event.
2. Specify the procedure that resamples the universe. In the case of significance testing, resampling was done by re-pairing X variables and Y variables at random (the “randomization” procedure). In the case of estimating the standard error and 95% confidence intervals, resampling was done by deleting one case at a time from the original sample (the “jackknife” procedure, originally proposed by Tukey).
3. If some computation must be performed on the resampled data, and if the computation was not described in the Step B procedure, then describe it. In the present case, the canonical correlation had to be calculated for each data set produced by resampling. One thousand sets of canonical correlations were produced by randomization in order to establish the reference distribution for the significance tests; 897 jackknife samples were used to estimate the standard errors and 95% confidence intervals (Peterson, 1991; Simon and Bruce, 1991).

The substantive interpretation of our significance levels and confidence intervals is the same as in traditional parametric inferences. For example, if a canonical correlation is found to be significant at the 0.001 level, this means that, using a data set like ours, random X-Y pairings would produce a correlation that large less than 1 time in 1000. Similarly, if the 95% confidence interval is [0.71, 0.80], this means that if we “re-did” our study using a similar sample of pilots, aircraft, routes, and other conditions, we would estimate the probability as 0.95 that the correlation would fall between 0.71 and 0.80.

We represent the resulting canonical correlation in a graph plate, with one graph for each paired variate. We depict each of the variables in the independent subset and each of the variables in the dependent subset. For each variable in a subset, we indicate its correlation with the corresponding paired variate. This measure is derived from the vector of weight and is more easily interpreted. This situation occurs because the raw weights themselves, like regression weights, may be deceptive. In addition, any inter-correlation between variables is filtered out. In identifying patterns, we indeed are interested in knowing that two (albeit correlated) variables have equal impact on the relationship.

The format of the center (sun-like) diagram is somewhat similar to a snowflake graph. The rays on the left represent all the dependent variables (the Y’s). The rays on the right represent each of the independent variables (the X’s). The positive correlation of each variable is plotted as a dark line protruding outward. The negative correlations are plotted as white lines pointing toward the center. Taken as a whole, the graph depicts the pattern of relationship between the dependent and independent variables. We call these sun-like graphs heliographs.

In the upper-left corner of the plate we depict a small scatter plot of the canonical correlation. The slope of the line is proportional to the bi-variate (canonical) correlation between the paired variates. This correlation value, as well as its 95% confidence interval, is depicted at the lower right corner of the plate. The significance (p) value of the canonical correlation, obtained from the randomization test, is also depicted here. Figures 7-14 to 7-17 are the plates of the first to fourth paired variates, respectively.

The canonical correlation analysis provided four pairs of variates. Each pair of variates provided us with a different pattern that exists in the data set. These patterns, as mentioned earlier, are unique—each one is independent of the others. We consider here only the first four pairs of variates. Their corresponding canonical correlation ranges from 0.95 for the first pair to 0.72 for the fourth pair. Since the variance is a square of the correlation, the fourth pair explains some 50% of the variance, and for this reason we considered it as the cut-off.

Figure 7-14 is a plate of the first pair of variates. The canonical correlation between the independent and dependent variate is .95, with a 95% confidence interval between .94 - .96. The correlation is highly significant ($p < 0.001$). The most consistent pattern in the data set indicates the following:

When

- altitude is slightly above the average (of 13,000 feet),
- the phase of flight is “Descent,”
- the ATC facility is “Approach,”
- and the vertical clearance is “Descend to altitude,”

Then the mode configuration of the aircraft is likely to be:

- autopilot “On,”
- pitch mode in “Flight Level Change,”
- and TMC mode in “Cruise.”

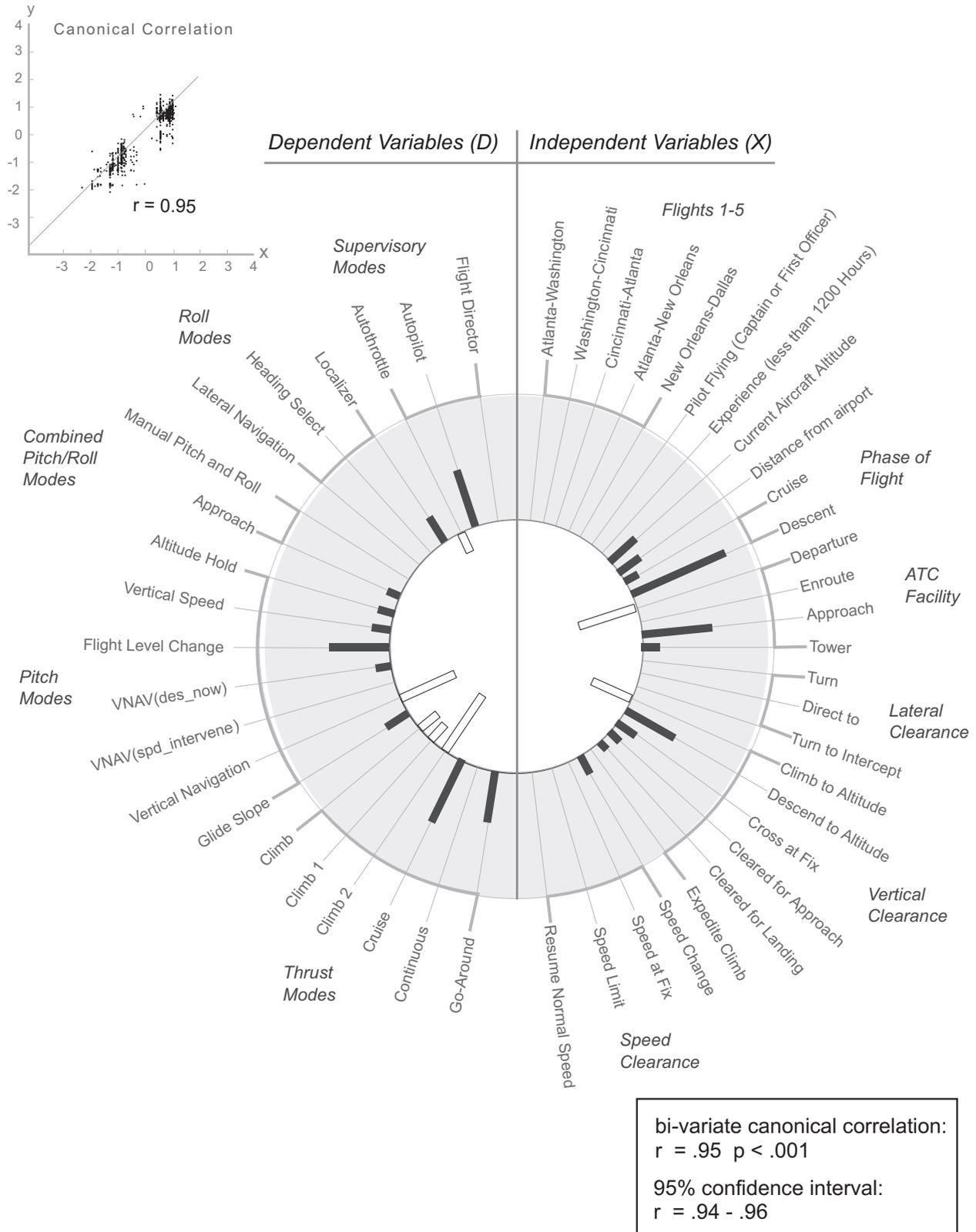


Figure 7-14. The first variate (structure correlation)

The pattern that emerges as the most consistent and highly reliable in the data occurs during the descent to the terminal area. When Approach Control is providing descent instructions to the crew, they most commonly reply by engaging the semi-automatic “Flight Level Change” mode. In this case, they most likely use the autopilot, as opposed to hand-flying the aircraft. This observed pattern for dealing with ATC clearances mirrors a highly recommended technique at the airline. The technique calls for using “Flight Level Change” modes and not “Vertical Navigation” mode when ATC (usually Approach Control) provides constant changes in cleared altitudes.

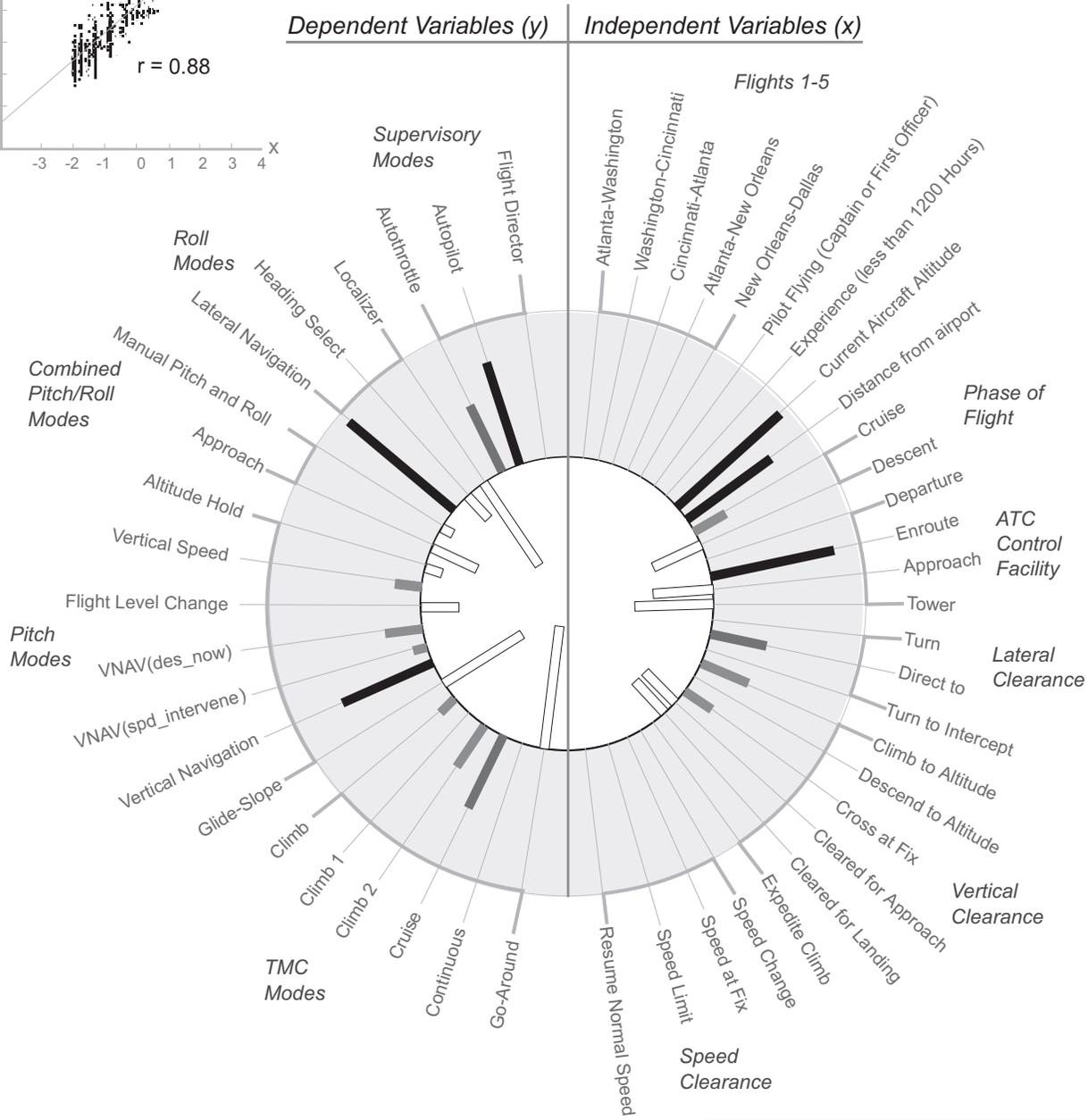
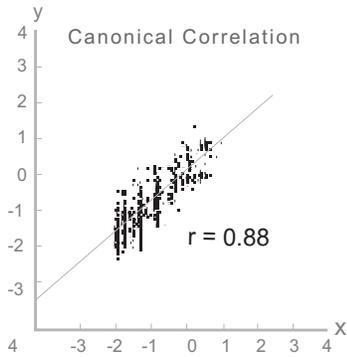
Figure 7-15 is a plate of the second highest pair of variates, in terms of the canonical correlation ($r = .88$, confidence interval between $.87 - .90$, $p < 0.001$). This pattern in the data set, which is also highly consistent, indicates the following:

When

- the altitude is high,
- the distance from airport is large,
- the ATC facility is “Enroute,”
- and the lateral clearance is mostly “Direct to,”

Then the mode configuration of the aircraft is likely to be:

- autopilot “On,”
- autothrottle “On,”
- pitch mode in “Vertical Navigation,”
- roll mode in “Lateral Navigation,”
- and the TMC mode in “Cruise.”



bi-variate canonical correlation:
 $r = .88$ $p < .001$
 95% confidence interval:
 $r = .87 - .90$

Figure 7-15. The second variate (structure correlation)

The pattern that emerges occurs during high-altitude cruise. Here the crew engages the “Vertical Navigation” and “Lateral Navigation” mode configurations. The autopilot and autothrottles are both engaged and the crew attempts to get the most efficient flight profile by letting the automation work. Air Traffic Control usually provides strategic clearances at these altitudes (e.g., “Direct to”), and this fully automatic mode configuration is very compatible with these type of clearances (Casner, 1994).

Figure 7-16 is a plate of the third highest pair of variates, in terms of the canonical correlation ($r = .81$, confidence interval between $.78 - .86$, $p < 0.001$).

The individual (structured) correlations are relatively small, indicating that the independent and dependent variables in this pattern are only weakly correlated. The pattern appears to be an amalgamation of two subpatterns: one associated with the initial climb, the other associated with the final approach. While the variate values are mathematically accurate and the correlation statistically significant, this combined pattern itself is not interesting and is probably operationally insignificant.

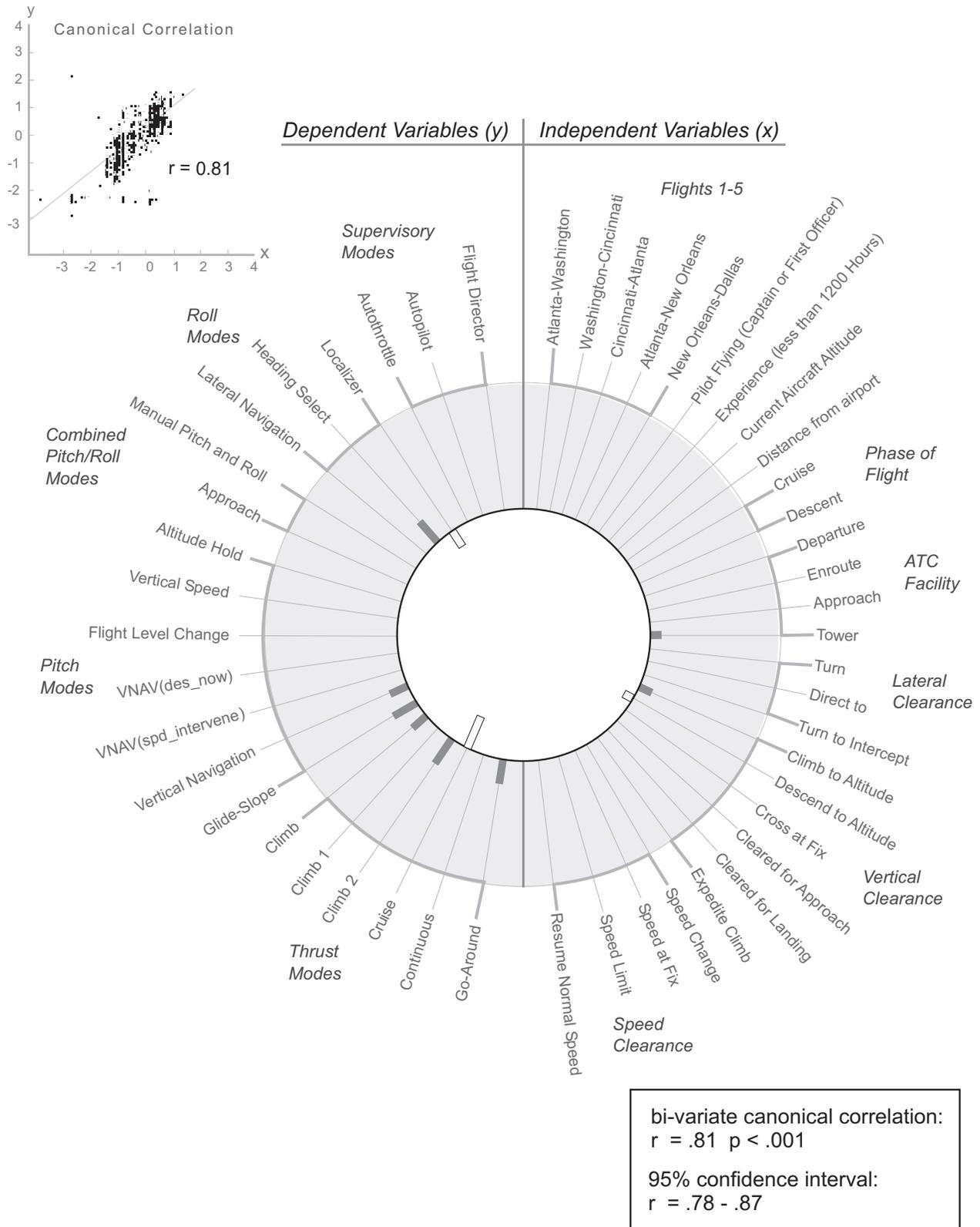


Figure 7-16. The third variate (structure correlation)

The fourth paired variate has a canonical correlation of ($r = .72$). Figure 7-17 is a plate of this last analysis. This pattern in the data set, which is still highly reliable, indicates the following:

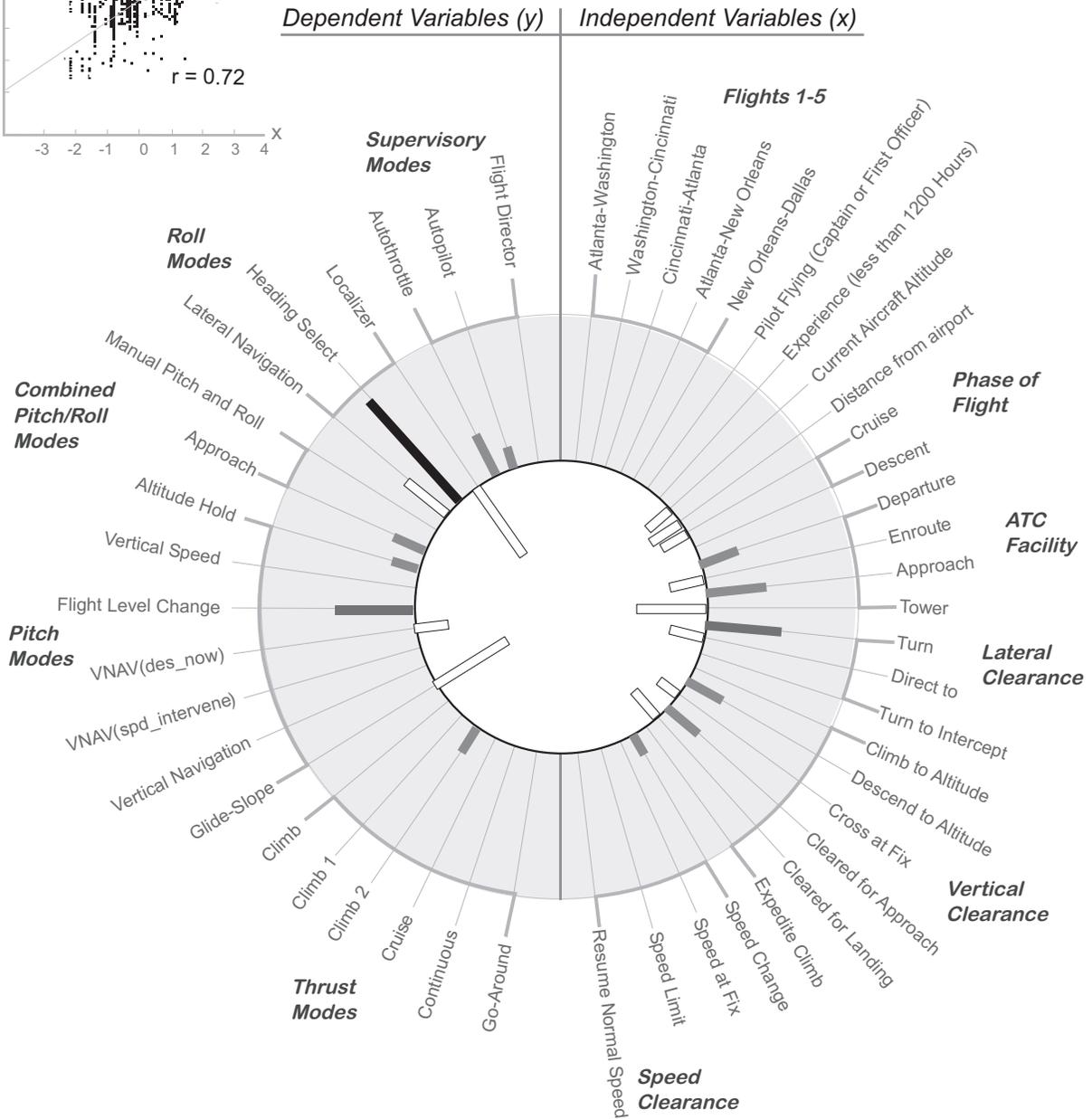
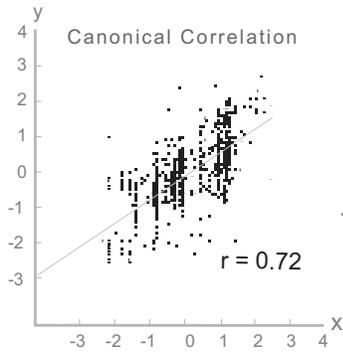
When

- the ATC facility is “Approach,”
- and the lateral clearance is mostly “Turn” to a heading,

Then the mode configuration of the aircraft is likely to be:

- autothrottle “On,”
- pitch mode in “Flight Level Change,”
- and the roll mode in “Heading Select.”

The pattern that emerges is situated in the same flight phase as the first variate—the descent to the airport. When Approach Control is providing turn instructions to the crew, the crew most commonly replies by engaging the semi-automatic “Heading Select” mode. In most cases, when this type of clearance is issued the pitch mode is already in “Flight Level Change,” as was identified in the first variate.



bi-variate canonical correlation:
 $r = .72$ $p < .001$
 95% confidence interval:
 $r = .71 - .79$

Figure 7-17. The fourth variate (structure correlation)

The four pairs of variates described in this analysis are highly correlated. They range from very high ($r = .95$) to moderately high with the last variate (0.72). The generally high correlation supports the notion that there are four separate patterns in the data, and that these patterns are independent of one another (Table 7-10).

Table 7-10. Correlation of Variate Pairs

	First “X” Variate	Second “X” Variate	Third “X” Variate	Fourth “X” Variate
First “Y” variate	.95	.00	.00	.00
Second “Y” variate	.00	.88	.00	.00
Third “Y” variate	.00	0.00	.81	.00
Fourth “Y” variate	.00	0.00	.00	.72

The canonical correlation captures the most *consistent* relationships between the independent and dependent variable subsets. We stress the word “consistent” because the procedure capitalizes on a set of variables that repeats itself in the data set.

The procedure reveals that there are very strong relationships between certain environmental factors and mode engagements. There are four such patterns of relationship in our data set. Three of these four statistically significant patterns also appear to be operationally significant.

The first pattern is associated with ATC clearances during descent into the terminal area. Many such clearances are provided to the crew as Approach Control attempts to gradually descend the aircraft toward landing. The crew reacts to these tactical clearances by using semi-automatic modes such as “Flight Level Change.” The second pattern describes the state of affairs when the aircraft is in cruise. At cruise altitudes ATC provides mostly strategic clearances, such as “Direct to” a waypoint, instead of headings. Crews tend to engage the fully automatic modes in this regime of flight. The third pattern is associated with landing. Here, crews employ the landing aids that will direct them toward the runway. The last pattern complements the first, and highlights the fact that heading clearances during the descent phase mostly prompt the pilot to use “Heading Select,” a semi-automatic mode.

In summary, the canonical correlation analysis identified the most reliable and consistent patterns in our mode transition data set. In addition, we were also interested in the magnitude of the relationship between the dependent and independent variables. The canonical correlation highlighted once again that ATC clearances and aircraft phases of flight have significant influence on mode selection and engagement. In doing so, it also de-emphasized those factors, in our independent subset, that appeared to have no impact on mode transitions. Among these are the leg (1–6), the rank of the pilot flying (captain, first officer), and the extent of his/her AFCS experience, as well as any speed clearances.

Chapter Summary

The purpose of the field study analysis was twofold. First, we wanted to describe the various mode configurations that pilots actually occupy, and present the transitions between these mode configurations (i.e., the mode configuration trajectory). This would allow us to understand and quantify how pilots interact with the modes of a complex system in order to achieve task goals.

The second purpose was to identify some of the factors in the environment that constrain the pilot in making mode selections, and hence contribute to this trajectory. In addition to listing these factors and their magnitudes, we were interested in their patterns of interaction with pilots' mode selections.

The analysis discussed here is the result of an observational study. This methodology poses some limitations for identifying cause-effect relationships, mainly that the factors are not directly manipulated by the experimenter (Cook and Campbell, 1979). Bearing in mind this limitation, the results presented here suggest the following conclusions.

Findings—Mode Occupancy and Transitions

Within the possible mode space there are certain mode combinations that are frequently used. Pilots use several preferred paths for transitioning among modes during the progress of the flight. It appears that one of the strategies flight crews use to combat the complexity of the system (e.g., its large and interacting structure and behaviors) and still get their job done is to use a limited subset of all possible modes.

In the domain of human-computer interaction (HCI), Rosson (1983) found that both experienced and less-experienced users used a very basic set of (text editor) commands. There were no unique and idiosyncratic sets for each individual. With experience, users simply added new commands to the basic set. Yet the set of frequently used commands employed by experienced users still fell short of the possible command space. Even the most experienced user employed only about one-fourth of the possible command space (p. 172). Similar findings were also brought forward by Sarter and Woods (1994, p.26) in their study of pilot interaction with the Boeing B-737/300 aircraft.

It appears that when confronted by a modern aircraft with enormous complexity, pilots convert it into a simpler machine by employing only a few of the many possible mode configurations.

Findings—Multivariate Analysis

The next logical question concerns the factors that shape the mode structure of this simplified machine. That is, why do pilots select one subset of all possible mode configurations and not another? To address this question we employed two types of statistical analyses—multiple regression and canonical correlation.

In conducting these multivariate statistical tests, we divided the data set into a set of dependent (mode configuration) variables and a set of independent variables. The latter consisted of environmental factors and pre-conditions (e.g., ATC clearances, pilot experience) *external* to the AFCS. This set of external variables was used for predicting the next mode configuration selected by the pilot. We completely ignored the current mode configuration of the aircraft in order to predict the next (cf. Callantine, 1996). Instead, we focused exclusively on the contribution of these external, or environmental, factors on pilots' selections of modes. Our ability to explain much of the variance using only these external factors is indicative of their highly significant contribution to mode selection.

The analysis indicated that mode selections in the AFCS are influenced by six key external factors:

- (1) The possible mode configurations of the AFCS
- (2) The standard operating procedures and recommended techniques at the airline

- (3) The phase of flight
- (4) The altitude
- (5) The type of ATC clearance—both lateral and vertical
- (6) The type of ATC facility controlling the flight

We offer several possible explanations for the influence of these factors:

The possible *mode configurations* naturally constrain the modes that can be selected by the crew. But unlike a text editor, this mode-configuration space is highly dynamic—as a function of several other factors, it dynamically changes its size. Depending on the state of the environment (e.g., receipt of navigation aids), some mode configurations are selectively disabled and enabled. Some mode configurations are internally disabled by the AFCS itself (e.g., “Lateral navigation” during takeoff), while other are simply inappropriate for the task specification (e.g., “Vertical Navigation-descend now” and “Localizer”). In addition, some of the mode-mode interaction in the AFCS further limits this space. For example, after the pilot arms the “Approach” mode, the system will attempt to automatically engage both the “Glide Slope” (pitch) and “Localizer” (roll) modes.

The second factor that constrains mode selection is the airline’s *operating procedures and recommended techniques* for using the AFCS. Each airline has a set of such mandatory procedures that the crew is expected to follow “by the book.” Naturally, these sets of procedures limit the set of possible mode configurations used by pilots at the airline. Other guidelines, such as recommended techniques, further limit this set.

While some airlines allow the pilot the freedom to engage any mode configuration and supervisory modes deemed necessary for the efficiency of the flight (Wiener *et al.* 1991), there are several well-founded techniques for using the AFCS shared by most pilots across all airlines. One such technique is the avoidance of what is sometimes called *mixed modes*: engaging fully automatic mode configurations such as “Vertical Navigation” and “Lateral Navigation,” and disengaging the autothrottles (Sarter and Woods, 1995b). Finally, pilots have their own set of personalized techniques for how they prefer to use the AFCS (Degani and Wiener, 1994). While these personal techniques normally do not differ from the airline’s mandatory and recommended set, they do further constrain the airline’s basic set.

The *phase of flight* is another important factor which constrains the mode configurations used by pilots. This relates to what we mentioned earlier as modes that are inappropriate for the task specifications at hand. For example, during the climb phase (a task specification), the pitch mode “Vertical Navigation-descend now” is inappropriate; it commands a descent and not a climb. For the climb phase there is a set of task-appropriate mode configurations. Likewise, for the descent phase there is another set. These sets, by the way, are not mutually exclusive. Some mode configurations (e.g., “Vertical Navigation” and “Lateral Navigation”) are at the intersection of these two sets.

Altitude is a primary and highly correlated factor with almost any task specification on the flight deck. For example, low altitudes are associated with takeoff or preparation for landing; high altitudes are correlated with cruise. As such, altitude is highly correlated with phase-specific tasks and therefore, directly or indirectly, influences mode selections.

ATC clearances also prompt mode transitions. This comes as no surprise, since modes can be viewed as a set of methods for executing the tasks directed by ATC. Flight crews use very simple cues to guide their mode selection. For example, “cleared for the approach” is a cue to engage the

“Approach” mode. Following a clearance that has a “direct to” element, pilots usually selected a “Lateral Navigation” mode, since it is the most efficient mode to comply with this directive. The set of seven categories of ATC clearances, both in the lateral and vertical component, seems to have a strong relationship with mode configurations. This relationship is clearly evident in the patterns highlighted by the canonical correlation.

The structure of the ATC environment is such that clearances are constrained by the *type of ATC facility* from which they are issued. ATC facilities vary in the type and rate of clearance issuance (Lee and Lozito, 1989). For example, ATC controllers in an Approach Control facility issue mostly tactical clearances (e.g., maintain heading of 280 degrees, descend to 6000 feet) at a high rate, while demanding a quick response. In contrast, ATC controllers in an En Route (Center) Control facility issue mostly strategic clearances (e.g., a complete route of flight between several waypoints). Evidence for the influence of both ATC facility and clearance type on pilots’ mode engagement was also found by Casner (1994), who studied the structure of this environment in detail.

Finally, in addition to the structure of the ATC clearances, there is also high consistency in the phraseology of clearances issued. This was easy to detect, as we documented the same flights over and over. Analysis of the tapes revealed that the same clearances were issued at the same locations during flights. Again, this consistency is a feature of the ATC that pilots capitalize on.

Furthermore, even within the same flight, aircraft coming from the same directions tend to receive similar clearances. Naturally, the controllers identified a set of clearances that best suited their own constraints. For example, at a busy airport like Dallas-Fort Worth (DFW), a pilot can easily anticipate what the clearance will be and get ready for it by monitoring the ATC radio transmissions to aircraft ahead. Analysis of the tapes identified many such replications of clearances in almost all airports in the study. Pilots capitalized on this information and knowledge of the ATC environment to prepare themselves and manipulate the AFCS accordingly.

Implications of Analysis

The findings of this study provide evidence of the relationship between the AFCS mode-configuration space and the structure of the operating environment. In fact, this relationship is so strong that one can predict, with a high degree of reliability, the next mode configuration just by noting a few environmental factors. There is no need for any information about the current state of the AFCS. These findings have important implications for improving current, and designing future, ground (ATC) and aircraft interaction.

ATC clearances and pilot selection of AFCS modes are tightly coupled. The coupling, however, is only unilateral (ATC → AFCS) at the current level of technology. Nevertheless, it is clear that in order to maintain efficient air-ground integration, ATC procedures (observed as clearances by the crew) and the AFCS must be compatible.

During the field study, however, we noted several cases in which such ATC practices (and possibly procedures) were *not* compatible with the behavior of the AFCS. One such incompatibility that occurred three times during flight to New Orleans, is described below:

On the flight route from Atlanta to New Orleans, there is a waypoint called Meridian (MEI). Commonly, ATC would direct the aircraft to cross 50 miles south of Meridian at a given altitude as the beginning of the descent toward New Orleans. However, this clearance was usually broadcast once the aircraft was either over or past Meridian. The problem? Once the aircraft passed Meridian (MEI), MEI was no longer in the active waypoint list, and therefore the pilot could not directly

specify a new fix based on it (50 south of MEI). There are several work-around solutions to this problem, but they require time and effort (calculate the location of the new fix from the active waypoint, go to the fix pages in the CDU, and bring it up).

But there is also the reverse side of such incompatibility. There are cases in which the AFCS does not fit very well to the status of the operating environment. We earlier discussed a situation in which flight crews used the “Flight Level Change” mode during the final approach to the runway. Here, because of the lack of glide slope navigation aid, the crew *cannot* engage the “Glide Slope” mode and instead must use other means to descend to the runway. There are general difficulties in using the AFCS during non-precision approaches (e.g., those with no glide slope aid). This is a well-known limitation of the current AFCS. Some airlines address this problem by specifying clear procedures on how to use, or not use, certain features of the AFCS during critical phases such as landings (for a discussion of this issue see Degani and Wiener, 1994, section 5.6).

The very strong coupling between the types of ATC clearances and pilot selection of AFCS modes also has implications for the design of new ground-aircraft systems. We shall briefly discuss some of these implications in the context of ATC modernization, design of new AFCS systems, and their effect on both.

Much research is currently underway to identify the issues involved in the modernization of the ATC system in the U.S. It is clear that ATC modes of operation and procedures will change dramatically. Our findings about the strong relationship between ATC clearances and mode selection on the flight deck suggests two preliminary conclusions: (1) changes in ATC clearances must be evaluated against the current types of AFCS, as these aircraft (e.g., B-757) will operate beyond the next decade; (2) any modifications to the AFCS must be in accordance to the types of task demands imposed by ATC. The relationship between ATC clearances and demands, and AFCS mode structure and design is critical for efficient ground-air interaction. In the 1980s, when AFCS-equipped aircraft were introduced into the ATC environment, many pilots complained that ATC did not take advantage of, or sometimes even diminished, the capabilities of these new machines (Wiener, 1989). It is crucial that this history not be repeated.

Similarly, design of new AFCS systems must take into account the type of ATC environment in which they will operate. Simply designing very sophisticated modes or employing traditional modes is not very efficient. In particular, designers must anticipate new ATC procedures and design flight decks accordingly. This is not an easy task, especially when the design span is lengthy. The problem may be compounded if ATC environments differ among various continents and countries. AFCS designers will then be faced with a very complicated problem.

Finally, with respect to both ATC and AFCS developments, there are several open questions. In the light of much more automated ATC-aircraft interaction, what type of mode usage patterns will exist? Is it worthwhile for designers to invest in the same modes, or others? What will be the impact of automatic interaction (e.g., data transfer and negotiations between AFCS and ground-based computers) on the likelihood of mode errors? While these questions are open, we hope that some of the methodologies explored in this chapter will provide both support and benchmarks for addressing these challenges.

In conclusion, the field study and analysis performed in this chapter allowed us to assess some aspects of the relationship between the structure of AFCS modes and the structure of the operating environment. The results of this relationship—spelled out in procedures, techniques, and common practices of mode selection—constitute the mode trajectories of this system.

These trajectories, however, are external to the machine's internal behavior. They must be set apart from an important aspect of this human-machine system, which is the corresponding internal behavior of the machine in response to these trajectories. The consideration of mode trajectories and corresponding machine behavior is the topic of the next chapter.

MODE USAGE IN A COMPLEX SYSTEM II: OFAN ANALYSIS OF A SCENARIO

INTRODUCTION

The field study described above allowed us to identify the various factors that affect mode transitions as well as some patterns of interaction between the pilots and the AFCS, given the state of the environment. In this chapter, we attempt to combine such sequences of interaction with our approach for modeling human-machine systems—the Ofan framework. Earlier we argued that the analysis of the field study data provided information about the interaction external to the machine behavior. Now we add the machine's (internal) behavior in order to better understand this human-environment-machine system.

Conceptual Approach

Bridging these external and internal machine interactions is not a trivial task. In modeling human interaction with a very complex system such as the AFCS, we need to augment our conceptual approach. Since all of the previous devices were rather simple, it was simple to infer the sequence of actions a user would take in operating these devices, such as mode selections and reference value manipulations. In fact, there was almost no practical need to model the user's sequence of actions to highlight the mode ambiguity. Based on the results of the field study, we can no longer simply infer what actions the user will take on the basis of interface design and user goals.

Instead, we must augment our methods of modeling to incorporate the complexity of the machine and the paths users take in managing this system. We have to model the machine, describe the sequence of actions, and then trace what prompted these actions. Only then can we identify what went wrong in this system. It is clear that the user's sequence of actions, and the trajectories of these actions, will indeed be an important factor.

In performing such an analysis, resources we have already developed come to our aid. With respect to modeling, we draw on the Ofan framework, the templates, and the structural elements. As for the sequences of actions and the environmental factors that prompted them, we draw on the methodologies and findings from the field study analysis. The combination of these two resources is the topic of this chapter.

Objective

The main goal of this chapter is to describe and analyze a mode confusion incident in a very complex system. In attempting to do so, we face a critical question: Is the Ofan framework adequate for describing human interaction with multi-component and multi-modal systems such as the autoflight control system of a modern aircraft? We must ask whether the templates that were developed in Chapter V and the categorization of structural elements can indeed be scaled to a complex system such as the automatic flight control system (AFCS). If not, then perhaps modes in complex systems are indeed different from modes in interfaces and simple devices, as suggested by

some authors. Finally, a related objective is to view the sequence of operators' actions (in this case, pilots) in the context of the environmental demands and the detailed behavior of the machine.

METHOD

To address the above objectives we conducted an analysis of the complete pilot-machine interaction process. The data include pilot behavior (in terms of sequences of actions) and machine behavior (in terms of an Ofan representation).

The source of data was the field study. In addition to collecting data about the AFCS mode configurations and external variables, the observer documented information about pilots' deviations from procedures, use of recommended techniques, and common practices in manipulating the AFCS. Of particular interest were cases of observed mode ambiguity and resulting confusion. We call these cases *mode scenarios*.

Twenty-eight scenarios involving clear deviation from procedures and mode confusion were observed and documented. Some of these occurred more than once. In an earlier study we described three of these scenarios (Degani, et al., 1995). Here, we model another such scenario using the Ofan framework. This particular incident involved confusion regarding the speed of the aircraft (a Boeing B-757) while the crew was transitioning from one vertical mode to another.

Documentation and analysis of scenarios is sometimes termed *behavioral protocol analysis*. The technique has been developed for studying human interaction with engineering processes (Woods, O'Brien and Hanes, 1987). The human role is to manage this process in the face of disturbances. The source of information includes operator actions, machine states, verbal reports made following the event (scenarios), and expert commentaries (Woods, 1993). The basic target to be achieved in a behavioral protocol analysis is tracing and understanding the evolution of the state of the underlying process or device, *in parallel with* the human agent's state of understanding, intentions, and activities in managing the process (p. 237).

Seated in the cockpit, the observer documented these incidents as they occurred. The log of previous mode configurations, state of the aircraft, and ATC clearances provided collaborative information about the dynamics of this scenario. In addition, following the scenario the crew made verbal reports about what went wrong. All this information was documented and later used for the analysis. During the process of analyzing the scenario, another pilot with considerable experience in the B-757 provided input to the process.

In addition to analyzing behavioral protocols, we also built an Ofan model of the underlying system. The data for building the model was based on information obtained from AFCS manuals. A secondary source of data was the computer code from an AFCS simulator.

In the following sections we describe the scenario and the underlying system, then proceed to model the system. We then continue with the analysis and discussion.

SPEED MODE SCENARIO

In this section we will first describe the mode confusion incident as it occurred, and then further describe the underlying features of the AFCS associated with this scenario.

Scenario

The aircraft was climbing to 11,000 feet per Air Traffic Control (ATC) instructions, and a fully automatic vertical mode called “Vertical Navigation” was active. In this mode the speed target-value is obtained from the Flight Management Computer, which computes the most economical speed (about 300 knots in this case). During the climb (at about 10,500 feet), ATC instructed the crew to reduce speed to 240 knots. The first officer engaged a unique feature of “Vertical Navigation” called “speed intervene,” which allowed him to enter the speed specified by ATC via the Mode Control Panel, as a new reference value.

As the aircraft neared 11,000 feet, it started the level-off maneuver by automatically disengaging “Vertical Navigation” and engaging “Altitude Capture” mode. Once at 11,000 feet, the “Altitude Capture” mode disengaged and the “Altitude Hold” mode was engaged automatically. During and after this maneuver, the speed was kept at 240 knots. Shortly after—ATC instructed the crew to climb to 14,000 feet. The first officer reached up to the Mode Control Panel and engaged the “Vertical Navigation” mode in order to initiate the climb. However, instead of climbing at a speed of 240 knots (as was still required by ATC), the aircraft speed defaulted to 300 knots!

The crew violated the ATC speed restriction because they assumed that the “Vertical Navigation” mode would ‘remember’ the speed reference value entered previously into the Mode Control Panel. However, the “Vertical Navigation” mode, once re-engaged, defaults to economy speed.

Mode Transition Diagram

We asked ourselves what resources we could bring to bear to describe the sequence of events that unfolded in this scenario. One approach is to describe the mode trajectory that occurred during this incident. For such an analysis, we draw upon the mode transition diagram that we previously used—but with some exceptions. First, instead of describing the relative frequency of making a transition, we portray the absolute frequency of the transition. Second, instead of summarizing 60 flights, we summarize only one flight. But more important, we are also interested in representing the reference values (e.g., commanded, or target speeds) associated with these mode transitions.

In addition to making mode transitions, pilots also change reference values in response to ATC demands. For example, when the aircraft is in “Flight Level Change” mode and ATC says “descend to 5,000 feet,” the pilot changes the selected altitude from the previously cleared altitude to the new assigned altitude (5,000 feet). Although no mode transition occurred, the mode is revisited. One way to represent this is by a loop transition.

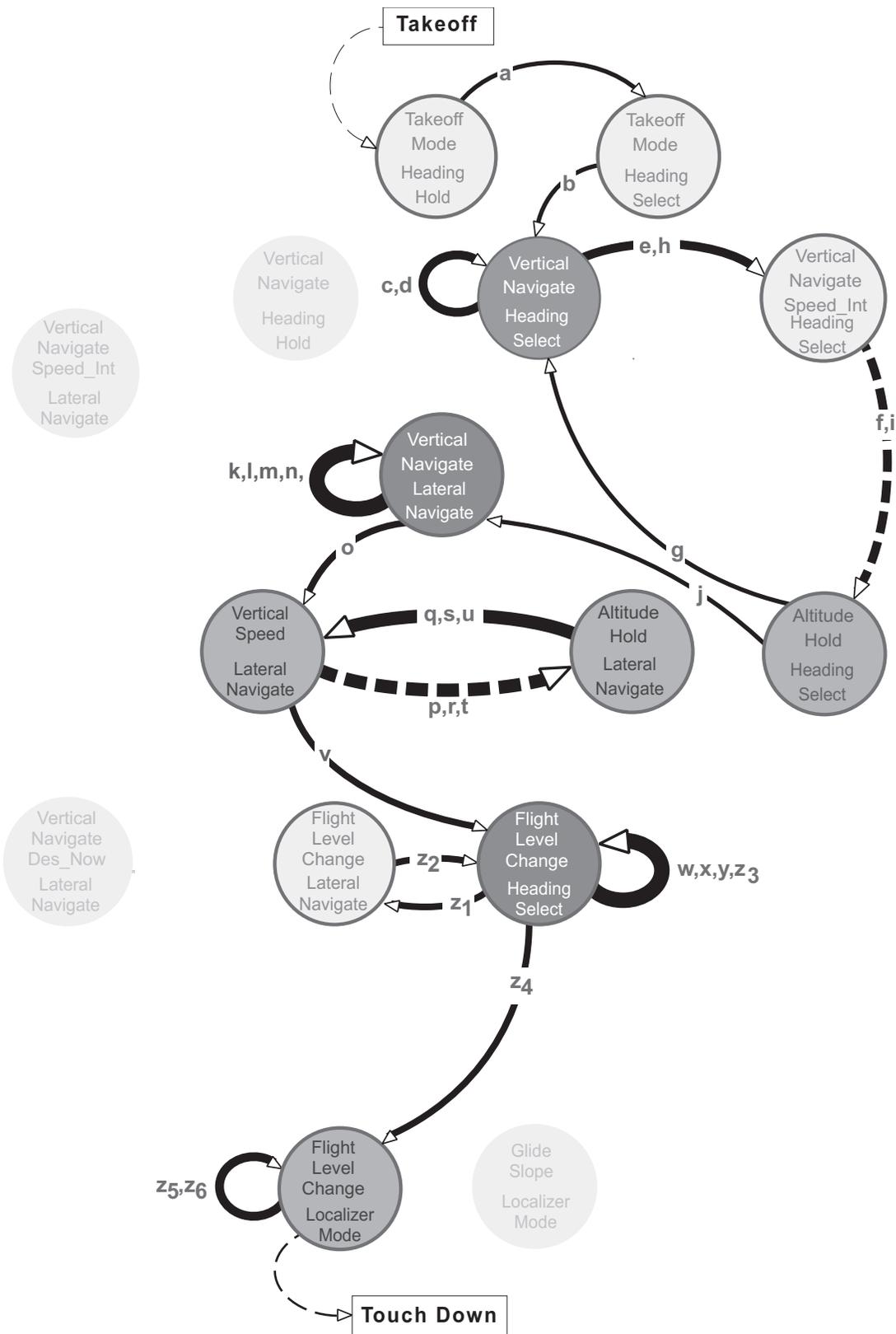


Figure 8-1. Mode transition (scenario)

Therefore, we have added such transitions when the crew manipulated the reference value but never exited this mode configuration.

Similar to the mode transition diagram used in the previous chapter (Figure 7-11), we depict mode configurations as nodes and transitions as arcs. In contrast to the previous diagram, we note here the (absolute) frequency, or count, of making the transition. The transitions can be between modes or a loop transition (reference value change only). We depict manual transitions as full lines and automatic transitions as broken lines. In addition, we describe the sequence of mode transitions and reference values according to the sequence of letters in the alphabet (a–z). Figure 8-1 represents the sequence of transitions, or paths, the pilots made during the entire flight.

The transitions described in the upper right portion of the diagram correspond to the several transitions in this scenario. We observe how the crew made a transition from one mode to another: After they reach the “Vertical Navigation”/“Heading Select” mode configuration, ATC requests a speed change to 240 knots (“maintain 240 knots”). As a consequence, the crew decides to transition to the “Vertical Navigation-speed intervene”/“Heading Select” mode configuration (this transition is labeled ‘e’ in the mode transition diagram in Figure 8-1). Once the aircraft completes the level-off maneuver at 11,000 feet, the transition to “Altitude Hold”/“Heading Select” takes place automatically (f).

The mode confusion occurs several minutes later: ATC clears the aircraft to climb to 14,000 feet (“climb and maintain 14,000 feet”). At this point, the crew re-engages “Vertical Navigation” (g) and the new mode configuration is “Vertical Navigation”/“Heading Select.” The aircraft is now accelerating past 240 knots because the new target value is economy speed. The crew realizes that something went wrong and they disengage the autothrottles and manually set the throttles. They press the speed knob, set 240 knots, and re-engage the autothrottles. As a consequence, transition (e) takes place, and now the new mode configuration is “Vertical Navigation-speed intervene”/“Heading Select.”

MODEL

The Automatic Flight Control System of this aircraft has eight functional modes to control the vertical aspect of the flight path. Three modes are discussed here: “Altitude Capture,” “Altitude Hold”—‘ALT HOLD’, and “Vertical Navigation”—‘VNAV.’ With respect to the speed reference value, the “Altitude Capture” and “Altitude Hold” modes obtain this parameter from the Mode Control Panel (Figure 8-2), which displays the existing speed at the beginning of the altitude capture maneuver. By default, the “Vertical Navigation” speed reference value is obtained from the Flight Management Computer, which computes the most economical speed for the particular flight regime. Yet another option, called “Speed Intervene,” allows the pilot to override the Flight Management Computer speed and manually enter a different speed. This is achieved by pressing the ‘speed’ knob and then dialing in the desired speed to the Mode Control Panel.

In modeling this system we draw on three types of modeling templates that were developed in Chapter V. We employ a template for describing task specification in the User Tasks module. The two other templates—the mode and reference-value template, and the supervisory mode template—are used in the Control Mechanism module.

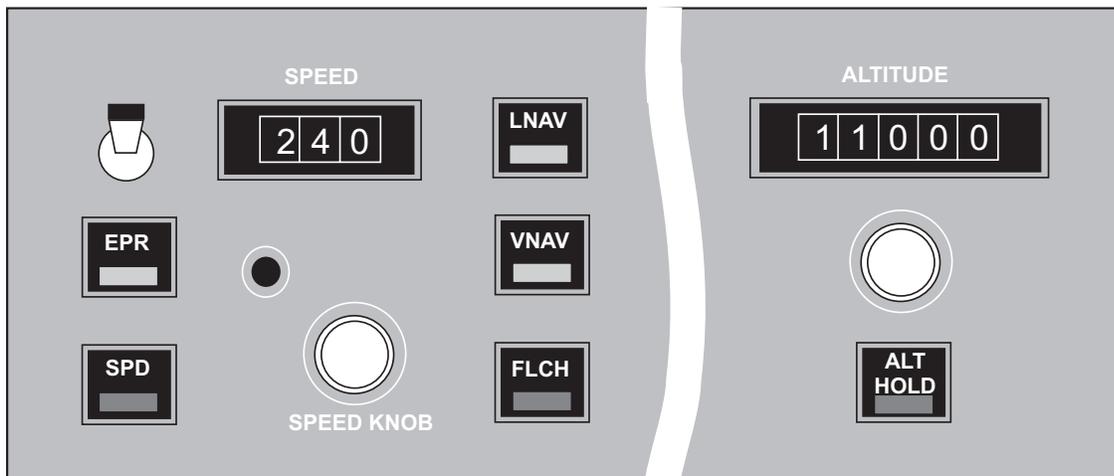


Figure 8-2. Mode Control Panel of a Boeing B-757

Figure 8-3 depicts part of the Ofan representation of this example. It contains three modules: Environment, User Tasks, and Interface. A detailed description of the Control Mechanism module is presented in Figure 8-4.

Environment

The *Environment* module describes the demands (in this case ATC clearances) on the crew. Similar to the way we earlier coded a clearance as a vector of several elements, we model in Figure 8-3 a clearance as two concurrent processes — vertical position and speed. Specifically, speed restrictions can be either given directly by the air traffic controller, or mandated by a given procedure (e.g., a Standard Instrument Departure).

User Tasks

In this module we employ the template for modeling the user task specification that was developed earlier. We shall now briefly present this template (but see the map display case study in Chapter V for a more detailed and theoretical discussion).

The basic template structure is a superstate with two concurrent processes. One process defines the start- and end-state definitions of the task. Concurrently, we have another process which describes the user's procedural steps (defined as states). These steps provide the necessary tasks to proceed from the initial start state and fulfill the end-state condition. Each step, or subtask, can be further broken down using the basic structure of two concurrent processes — start-end states and corresponding procedural tasks. In this way we can build a hierarchy to the level of detail deemed necessary. Portrayed as a reachability graph, the structure has a start state, and a fanning tree which closes up in a single end state.

The *User Tasks* module in Figure 8-3 encompasses the two primary tasks performed by the crew: (1) modify speed clearance, and (2) monitor constant speed. Both are activities which are depicted as superstates. We use here the same structure as was used in the map display example.

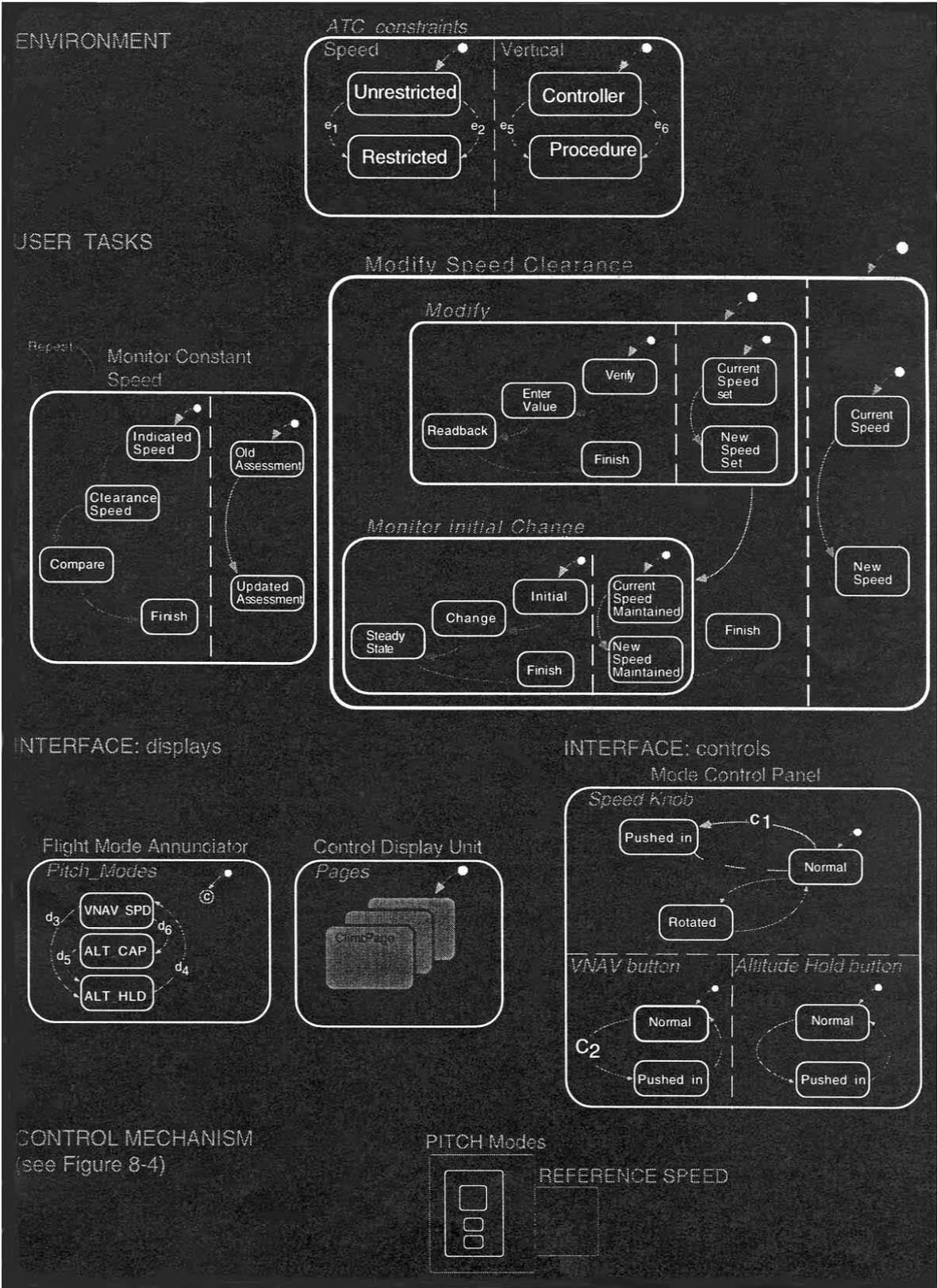


Figure 8-3. Environment, user task, and interface modules

The modify speed clearance state has a start state—*CURRENT SPEED*—and an end state—*NEW SPEED*. The end state signals the closure on this task. The procedure to fulfill the task specification involves two subtasks. The first subtask, *MODIFY*, also has start- and end-states. The procedure here involves verifying the speed clearance value, inserting the value into the Mode Control Panel, and reading back the new clearance to ATC. The subtask is complete when the new speed value is set correctly.

The second subtask in Figure 8-3, *MONITOR INITIAL CHANGE*, starts when the aircraft maintains the current speed and is brought to an end when the new speed is finally maintained. This is achieved by monitoring for the initial change in aircraft speed, the acceleration or deceleration of the aircraft, and finally reaching steady state at the new speed. Now, the new speed is set and maintained properly by the aircraft and we also have reached the end state of the overall (super) task (*MODIFY SPEED CLEARANCE*).

As a consequence of reaching the end state (*NEW SPEED*) we transition to the next task—*MONITOR CONSTANT SPEED*. Note that this new task is outside of the large superstate—*MODIFY SPEED CLEARANCE*. The monitoring task has a start state—*OLD ASSESSMENT*— and an end state—*UPDATED ASSESSMENT*. The end state is achieved after the three-step procedure is complete. The procedure involves identifying indicated speed and recall of clearance speed, and then comparing between the two. When this procedure has been completed (*FINISH*), a transition from start state (*CURRENT ASSESSMENT*) to end state (*UPDATED ASSESSMENT*) occurs. At a certain time interval after this task is completed, the pilot will again sample the speed value, and so on. This repeated entry to the *MONITOR SPEED* task is depicted as a loop transition.

Similar to our depiction of the user task in the map display example, the model provides us with three types of information. One is the task specification—the set of outcomes and task objectives. The next is the step-by-step procedure for achieving this objective. Third is the type of information and interaction required to fulfill each state in the procedure. A critical aspect of this concept is that task specification is *independent* of the procedure. A procedure is only one way to achieve the task objective. And hence other procedures can also be evaluated (McFarland and Bossler, 1993, chap. 8).

Interface

The *Interface* module in Figure 8-3 contains two elements: Controls and Displays. The Control element describes the Mode Control Panel and its various knobs and buttons

The initial state of the speed knob is NORMAL. The switch itself is multi-functional: when rotated, it is used for entering speed reference values into the Mode Control Panel. When momentarily pushed, it engages or disengages the “speed intervene” submode of the “Vertical Navigation” mode. We depict these two functions of the switch as states in Figure 8-3. The outcome of pushing the switch is dependent on the current state of the submode. However, this is not the only way to disengage this submode, as both manual and automatic mode transitions can also affect this submode.

Two other mode buttons are depicted in Figure 8-3—“Vertical Navigation” (VNAV) and “Altitude Hold” (ALT HOLD). These buttons are used for manually engaging the two modes. These controls, however, provide an indication of the status of their corresponding modes, as they are lit when the modes are active, and can be disengaged.

The second element of this module describes two displays. The Flight Mode Annunciation (FMA) provides indications about the current mode configuration of the AFCS, as well as the armed modes. The FMA is updated according to the state of the Control Mechanism. The other display that we describe is the Control Display Unit (CDU) of the Flight Management Computer. One of its many pages is the Climb page. One of the line items on the Climb page is the value of the economy speed.

Control Mechanism

In describing the *Control Mechanism* of the AFCS we draw on two other modeling templates that were developed in Chapter V: the mode and reference-value template, and the supervisory mode template. Figure 8-4 is a refined view of the Control Mechanism module.

On the mode side of Figure 8-4 we describe the “Vertical Navigation” (VNAV) mode, “Vertical Speed,” “Altitude Hold,” and “Altitude Capture” mode. All are modes in the vertical aspect of the flight. On the reference value side we depict the source of the speed reference value—either from the Flight Management Computer (FMC) or from the Mode Control Panel (MCP). In describing the model we first discuss modes and then the reference values.

Modes

The mode structure of the AFCS contains both supervisory modes and functional modes. We shall first describe the supervisory structure (or template) and then describe some of the functional modes.

The modes of the AFCS are organized in an hierarchical structure which corresponds to the supervisory mode levels in the AFCS. We capitalize on this fact and employ here a supervisory mode template that was developed earlier (cf. the cruise control and blood pressure device).

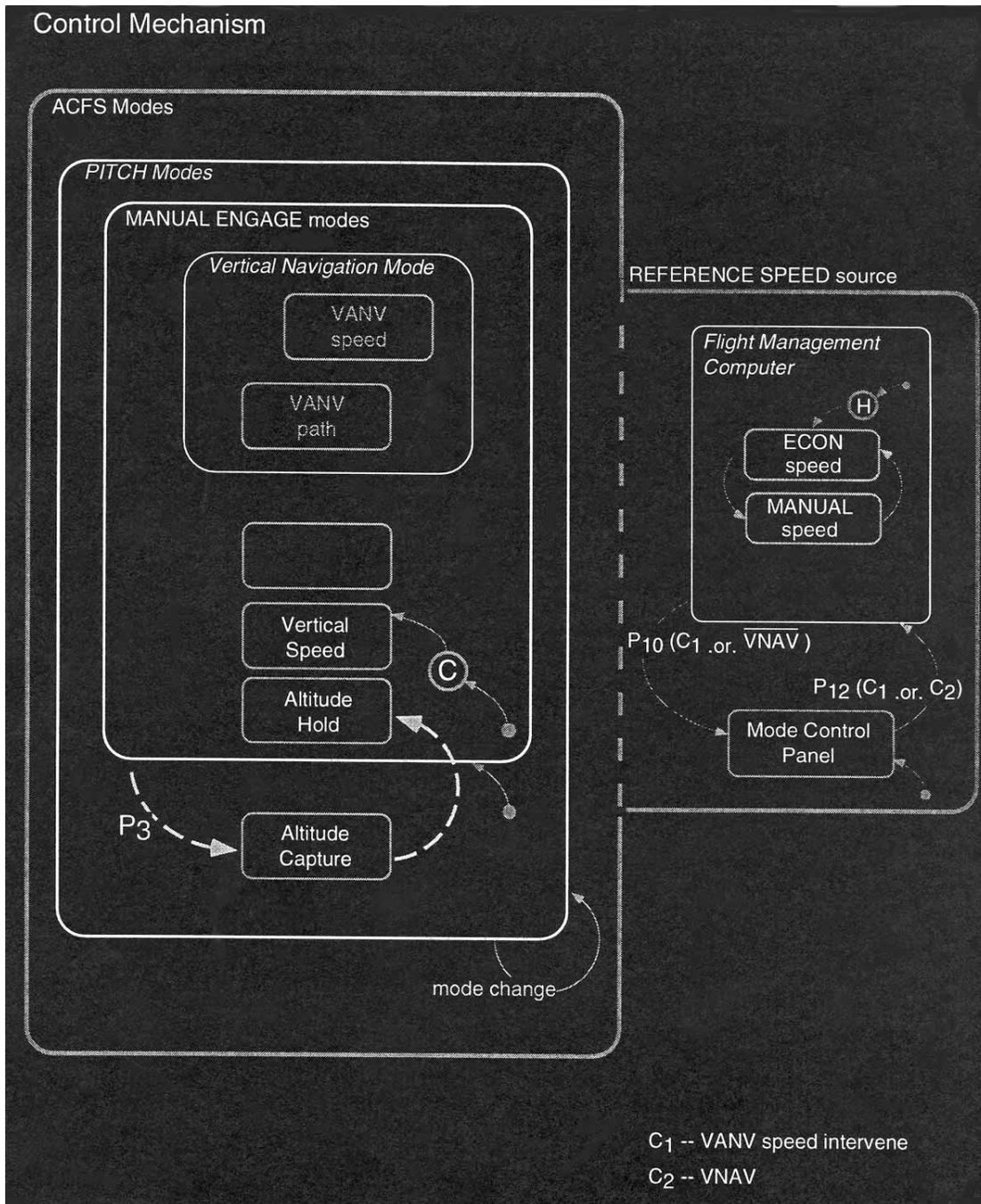


Figure 8-4. Control Mechanism module

The highest level of automation in this partial description is the VNAV mode. This mode is depicted as a state at the top of the mode pyramid in Figure 8-4. This mode is both functional and supervisory. It is functional in that it has a unique mode behavior in terms of control algorithms and reference values. It is also a distinct supervisory mode (fully automatic) as the crew can pre-program it in advance and let the computer completely manage the navigation of the flight. In contrast, the “Vertical Speed” and “Altitude Hold” modes are both part of a set of semi-automatic modes. The crew must manipulate them in order to navigate the aircraft.

We encapsulate Vertical Navigation (VNAV) and the set of semi-automatic modes under one superstate called “MANUAL ENGAGE MODES” to indicate that all these modes can be manually selected by the crew (in addition, these modes can also be engaged automatically—but we shall discuss this point later.)

Other modes in the AFCS can *only* be engaged automatically—no direct manual engagement is possible. One such example is the “Altitude Capture” mode. Similar to the “Retry” mode in the blood pressure device, this mode engages automatically only when the aircraft is climbing or descending to an altitude. When the aircraft is close to the selected (or pre-programmed) altitude, an automatic transition from the current mode (e.g., “Vertical navigation”) to “Altitude Capture” takes place.

Since the pilot *cannot* directly engage this mode, we depict it outside of the set of the manually engaged modes encapsulated by the “MANUAL ENGAGE MODES” superstate. By referencing this feature of the supervisory mode template, it is evident that transition to the “Altitude Capture” mode can occur from any active manual mode. This is depicted as a global (and also automatic) transition P_3 . Other only-automatic modes, such as stall protection modes which are activated when the aircraft is operated in marginal (envelope) conditions, may be depicted here.

In this section we describe some of the features of the functional modes of the AFCS. Specifically, we pick up from a point that was mentioned, but not elaborated earlier—automatic transitions. Although most of the functional modes in the AFCS are engaged manually, some can also be engaged automatically. One example is the “Altitude Hold” mode.

In the capture maneuver described earlier, once the aircraft comes within several hundred feet of the target altitude, an automatic transition to “Altitude Hold” mode takes place. This transition re-enters the superstate that encapsulated all the modes. This transition is automatic and illustrates that some of the functional modes are located in the intersection of both manual and automatic mode sets.

Finally, the loop transition ‘mode change’ outside the “PITCH MODE” superstate indicates that any mode selection and request by the pilot may “break” the currently engaged mode. For example, while the aircraft is in “Altitude Capture” mode, requesting “Vertical Speed” will cause immediate engagement of this mode (and disengagement of “Altitude Capture” mode). We use this loop transition as a means of engaging the selected mode.

Reference Value

In this section we describe the reference value aspect of mode behavior. As we discussed at the beginning of this study, modes and their associated reference values are what produces mode behavior.

The template of a mode and its reference value describes this form of mode behavior. It is depicted as a superstate divided into two concurrent elements—a mode process and a reference value process. We have used this template all along—starting with the alarm clock example and moving all the way to the map display example in Chapter V. We employ the same structure in this model, too.

On the right side of Figure 8-4, we note the reference-value element of the pitch mode. It contains two superstates: *FLIGHT MANAGEMENT COMPUTER* and *MODE CONTROL PANEL*. Each superstate describes a different source for the speed reference value: The speed can be obtained either from the Mode Control Panel or the Flight Management Computer. Initially, the source of the

speed parameter is from the Mode Control Panel. Engagement of “Vertical Navigation” (C_2) via the interface (Figure 8-3) will cause a transition to the Flight Management Computer as the source of speed values (P_{12}). By default, the speed reference value is the computed economy (*ECON*) speed. The pilot, however, can manually enter a different speed reference value into the computer. This is described by the second state (*MANUAL SPEED*) within the *FLIGHT MANAGEMENT COMPUTER* superstate. Any engagement of one of the semi-automatic modes (such as “Altitude Capture”) causes a transition to the Mode Control Panel as a source for the speed reference value.

In addition to transitioning between “Vertical Navigation” (VNAV) and the set of semi-automatic modes, the crew has another option for manually adjusting the speed. A submode of “Vertical Navigation”—namely “speed intervene”—allows the pilot to enter speed values via the Mode Control Panel without manually entering the desired speed via the Flight Management Computer interface (i.e., by bringing up the Climb page and entering a new value). Such submodes become handy when the crew engages “Vertical Navigation” during climb (to maximize efficiency) and ATC is changing speed for separation purposes.

Engagement of this submode is accomplished by pressing the speed knob on the Mode Control Panel (see the Interface module in Figure 8-3). This event, C_1 , is a guard on the transition between the Flight Management Computer and the Mode Control Panel as a source for speed reference values. A closer look at the two transitions between the *FLIGHT MANAGEMENT COMPUTER* superstate and the *MODE CONTROL PANEL* state, reveals an interesting architecture: the guard on both these transitions is C_1 . Later in the Analysis section we shall discuss in detail the significance of this type of engagement architecture (circular mode transition).

To conclude, we describe in this model a portion of the interaction between the pilot and the AFCS pitch modes. In describing the pilot tasks, we rely on the template of task specification that was also developed previously. In modeling the control mechanism, our focus is on the (pitch) mode and speed reference-value relationship. To represent this intricate relationship we have employed two of the modeling templates that were developed earlier. We have used the supervisory mode template to describe the hierarchy of pitch modes in the AFCS. This has allowed us to describe the types of mode transitions in the AFCS. We have then used the mode and reference-value template to capture the relationship between pitch modes and speed values in the AFCS.

ANALYSIS

In this section we combine the two main elements of this scenario analysis—the mode transition diagram and the model. The mode transition diagram provides information about the pilots’ sequence of actions. This behavioral protocol provides a view, from *outside* the AFCS, of how pilots interact with this system. The Ofan model, on the other hand, provide a view from *inside* the AFCS about the behavior of the machine.

In the following analysis, we combine the two views by superimposing the mode transition diagram on the Ofan model of the AFCS. Using the Ofan model as a map of the territory, we trace the mode transitions selected by the crew. As such, we describe the actual mode selection paths the crew took in the context of the intricate behavior of this complex control system.

Tracing the Mode Transitions

The scenario begins as the “Vertical Navigation”/“Heading Select” mode configuration is active (Figure 8-1) and the aircraft is cleared to 11,000 feet. This implies that the vertical mode is the fully automatic “Vertical Navigation” and the source for the reference values is *FLIGHT*

MANAGEMENT COMPUTER: ECON SPEED. ATC requests a restricted speed of 240 knots and the crew responds by pressing the speed knob of the Mode Control Panel. This action fires event C_1 which is broadcast to the network (Figure 8-3). As a consequence, event P_{10} takes place (C_1 is the guard on this transition). Now the speed source is the Mode Control Panel, and the “speed intervene” submode is active. Again referencing Figure 8-1, we note the corresponding transition to the “Vertical Navigation-speed intervene”/“Heading Select” node (transition ‘e’).

As the aircraft nears the target altitude of 11,000 feet, event P_3 takes place and an automatic transition to “Altitude Capture” ensues (Figure 8-4). The Mode Control Panel captures the current speed of 240 knots and the aircraft continues to maintain this speed. When the aircraft is just below 11,000 feet, another automatic transition, this time to “Altitude Hold,” takes place. The speed is still 240 knots and the source for this value is the Mode Control Panel. On the mode transition diagram, a corresponding transition to “Altitude Hold”/“Heading Select” is depicted (transition ‘h’).

Shortly thereafter, ATC clears the aircraft to 14,000 feet. In Figure 8-3 we note that the crew pushed the VNAV button on the Mode Control Panel (Interface module). This fired event C_2 , which is now broadcast to the entire network. In Figure 8-4, event P_{12} takes place because C_2 is one of its guard conditions. As a consequence, the speed reference source reverts to the Flight Management Computer, and ECON speed is the active value by default. Back to the mode transition diagram (Figure 8-1), we note the corresponding transition ‘g.’

At this point, the crew assumes that this mode selection will maintain the ATC speed restriction of 240 knots. Instead, however, this mode selection transitions the system to the “Vertical Navigation”/“Heading Select” mode configuration, with all its speed implications. As a consequence, the aircraft starts to accelerate to a new reference speed of 300 knots. Several seconds later, the crew realizes the mismatch between the aircraft current speed (now about 280 knots) and the ATC restricted speed (of 240 knots) and intervenes by disengaging the autothrottles and retarding the throttles manually.

During three transitions—from “Vertical Navigation” to “Vertical Navigation-speed intervene,” to “Altitude Capture” and finally to “Altitude Hold”—the reference value trajectory was the same. That is, the 240 knots speed was maintained via the Mode Control Panel. Only during the last transition to “Vertical Navigation” was this trajectory broken. The crew forgot about the default values of “Vertical Navigation” and assumed that the reference-reference value of the past three transitions (240 knots) would still be maintained. The mode confusion occurred because of an inaccurate expectation on the part of the crew. There was a difference between the expected behavior (maintenance of the existing reference value trajectory) versus the actual behavior (a new reference value). The consequence was an ATC clearance violation on the part of the crew.

Structural Elements

We may wonder what were the underlying features of the system that led to this mode confusion. Two structural elements that we developed earlier allow us to shed some light on this scenario. The first structural element contained in this architecture is the “circular mode transition.” In this structural element, the switching between the modes is circular. That is, the same event is used to switch between two modes. As a consequence, the user’s identification of the next mode configuration is dependent on his or her knowledge of the current state. In most cases, this is not a problem if the user is well aware of the current mode configuration of the system (e.g., via a display).

Circular Mode Transition

Nevertheless, two factors can make this architecture confusing: (1) a hidden transition can take place and (2) the display can be inadequate. As mentioned earlier, the speed knob is not the only way to disengage the “speed intervene” submode. When an automatic transition out of “Vertical Navigation” takes place (such as to “Altitude Capture” and “Altitude hold”), the submode is disengaged automatically.

Second, the display status of the “speed intervene” submode is not explicit. When the submode is active, the only indication on the Mode Control Panel is by observing the speed window. If the speed window displays a value, the submode is active; if the speed window is blank, the submode is not active. The other source of information is the Flight Management Computer interface, located some two feet below the Mode Control Panel. But during climb in the terminal area, pilots’ attention is focused mostly outside of the cockpit, searching for conflicting traffic, not inside. Perhaps this inability to quickly determine that the “speed intervene” submode was active may have contributed to the pilots’ confusion about the status of the AFCS.

Mode Reference-Value Interaction

But it appears that the circular mode transition may not be the only structural element at play in this scenario. Another structural element—namely the mode reference-value interaction—probably also contributed to the mode confusion. In this structural element, a change in mode carries with it a change in the source (manual, automatic) of the reference value. In order to anticipate the new status of the system, the pilot must have a complete and accurate model of the system. In this case, illustrated by the scenario where the mode transition is automatic, it is much more difficult for the user to anticipate the next mode configuration of the machine. In addition to requiring an accurate system model, the user must have the capability to sense the event that triggered the transition.

In this specific example, the pilots did in fact know that the automatic transition to “Altitude Hold” had occurred. What appeared to be missing was the ability to quickly recall the mode reference-value interaction information. Following the incident, the crew mentioned that although they knew about this potential problem, they had difficulty in recalling it as the situation was evolving.

The Dynamics of Reference Value Transitions

So now we wonder what it was that lulled the crew into believing that the AFCS would indeed maintain the previous reference-value trajectory. We focus here on the transitions from one reference value to another. The transition trajectory, occurring almost completely automatically, may be a factor here.

In the case of the cordless phone, we had two factors working together. One was a well-rehearsed sequence of actions (pressing the ‘Talk’ button to activate the phone). The other was a configuration of the machine that most of the time was in agreement with the user’s expectations, but sometimes not. Although our users were aware of both the logic of the phone and of the potential trap, they fell in to it regularly.

In the AFCS speed mode problem, we also have two factors working together. One is a sequence of automatic reference-value transitions. The second is a configuration of the AFCS that most of the time is in agreement with the user’s expectations about this trajectory, but sometimes not. Again, although our users were aware of the potential trap, they were led into it by the dynamics of events. And perhaps this dynamic of *automatic* mode and reference value interaction that must be

monitored and understood by the user is indeed unique, at least for now, only to complex supervisory control systems (Sarter and Woods, 1995b).

Asynchronization

As we mentioned earlier, one way to think about mode confusion is as an asynchronization between the user-expected states (depicted in the User Tasks module) and the behavior of the machine (depicted in the Control Mechanism module). In this case, the problem is detected when we note a mismatch between the states of the Environment module (speed: *RESTRICTED*), the User Tasks module (*MONITOR CONSTANT SPEED*), and the state of the Control Mechanism module (transition to *ECON SPEED*). While the Environment and User Tasks modules did not change their corresponding states (speed is still restricted by Air Traffic Control, and the crew is only required to monitor constant speed), the Control Mechanism did. The system as a whole lost the necessary synchronization between its modules, and mode confusion resulted.

Discussion

This specific mode problem, we argue, is a combination of several factors. Two of these factors can be traced to our set of structural elements: the “circular mode transition” and the “mode-reference value interaction.” Another factor is probably related to a well-rehearsed and expected sequences of reference value transitions.

Specifically, the machine is inconsistent in maintaining a global reference value. While transitions among several modes all maintain a global reference parameter (speed), one transition does not. After transitioning into the “Vertical Navigation” mode, a default entry changes the value of the global reference parameter into a local one (economy speed). The crew did not expect this transition to ECON speed and mode confusion occurred. The display, in particular the relationship of the speed knob to the speed reference value, does not aid the crew in resolving or recovering from the situation.

CHAPTER SUMMARY

The main objective of this final chapter has been to describe and analyze a mode confusion scenario. The analysis allowed us to identify the factors that contributed to this mode confusion. In addition, two other objectives were set forth in the introduction to this chapter. One was to assess whether the Ofan model can be used to represent a complex system and identify some of the features that contribute to mode confusion. The other was assess how to combine and use information on mode trajectories and sequences of actions within the structure of the Ofan representation.

With respect to the scaling-up issues, we have shown that the same templates, developed earlier in more simple systems, can indeed be brought to bear on this complex system. We have used here both the mode-reference values template and the supervisory mode template. The two templates allowed us to describe the behavior of this complex machine in a organized form. Furthermore, they highlighted the problematic interaction between modes and reference values—which is at the heart of this scenario.

In addition, we have demonstrated that two of our structural elements which were developed earlier and based on the previous set of models, are applicable. We have identified a circular mode transition problem in the speed knob of the Mode Control Panel. The problem resembled the ‘Talk’ button transition in the cordless phone example. The second structural element that applied

here was the mode-reference value interaction, which was also noted in the cruise control and blood pressure device examples.

Nevertheless, it appears that another factor also contributed to this scenario, namely, the dynamics of the mode and reference-value interaction in light of automatic transitions. Although bearing some resemblance to the sequences of actions in the cordless phone, this factor does represent a new feature. It is unique because it is associated with *monitoring* a set of automatic trajectories as opposed to manually performing a set of actions. This feature has much in common with the distinction Sarter and Woods (1995b) make between errors of commission (manual) and errors of omission (monitoring) in their study of pilot interaction in the Airbus A-320.

The last objective of this analysis was to assess the usefulness of combining two representations of human interaction with mode-based systems. These two representations are the model of the machine and the paths users take through this machine. Our approach was to combine the Ofan model of the machine with the mode transition diagram representing users' paths. Because these approaches are essentially based on the same modeling theory—the Finite State Machine—the mating was painless. We were able to combine two conceptual resources that we have developed and used throughout this thesis. The superimposing of the mode transition diagram on top of the Ofan model provided the much-sought-after view of the human-machine interaction in the context of the environmental demands and the detailed behavior of the machine.

CONCLUSIONS

We began this thesis with several examples of mode-related problems. They all involved confusion on the part of the users regarding the next mode (or state) configuration of the device. We asked ourselves why these problems occur, and what the features of these devices and systems are that lead to such confusion, in conjunction with human input. We then surveyed the literature and found that mode problems are well recognized. In HCI, the problem has been a concern for at least two decades, and has been discussed, researched, and modeled. Many researchers and software developers have converged on similar solutions. In human-machine systems, the problem recognition span has been shorter. Nevertheless, recent research has certainly addressed the problem forcefully. Both the research and operational communities have recognized the problem, and several insightful experiments have been conducted. Recent research has focused on attributes such as complexity, coupling, autonomy, and opacity, as factors that create mode and automation problems in cockpit automation, as well as in other highly dynamic and complex system systems.

We pressed on with our inquiry into the intrinsic mechanisms—both on the part of the user and the machine—that combine to produce these problems. We learned from the aviation literature that the mode confusion problem has been attributed as a cause in many incidents and several accidents—hardly a trivial matter. Yet the design and regulatory communities lack the necessary methods to quantify and systematically evaluate either the general contribution of automation, or the particular contribution of mode usage, to aircraft safety.

We then realized that some formal description may be useful for identifying and quantifying the potential for mode confusion and error in these systems. Our next step, therefore, was to survey the literature on models of human-machine systems and identify available representations. We concluded that no single existing model can capture the behavior of the human, the machine, and the interaction between them. However, we found that certain features of the Statecharts specification language were well suited for representing the behavior of mode-based system. These features included Statecharts' ability to represent hierarchy and concurrency.

Using Statecharts—a modern extension to the Finite State Machine—as our common modeling language, we developed the Ofan modeling framework. The framework maps the environment, human, and machine processes as separate modules. All these processes are then interconnected, via transitional links, to represent the interactions in this human-machine system.

Using this framework we began to model one system after another. We found that many mode-based systems share particular types of structure. This structure appears, albeit with varying levels of complexity and in different guises, in each of the mode-based systems we modeled. We developed two modeling templates to capture and capitalize on this structure. We later recognized that we must also evaluate the way that users obtain information about the state of the machine, in order to categorize some of these mode problems. This information-gathering process is directly linked to the user's task specifications. We then devised our theoretical approach for modeling task specifications, and proceeded to apply it to a concrete example—the map display of a modern 'glass cockpit' aircraft.

We also found that there are recurring features in these machines that, in combination with the user's process of interacting and monitoring, may lead to mode confusion. We captured some of these features in a set of abstract categories that we term the structural elements of a design.

To address operator interaction with modes in highly dynamic and complex systems, we analyzed data from a field study and developed methodologies to understand pilots' mode selections. The results provided us with a better understanding of the factors that prompt mode selection, and of how pilots constrain their interaction with the Automatic Flight Control System as a consequence of these factors.

Armed with this understanding, we next analyzed a mode-related incident that occurred during the field study. To describe the machine, we modeled it in our Ofan framework. We then applied some of the methodologies and conceptual resources described above in our analysis of the features that contribute to mode confusion, and identified several such structural elements in this design. Since we could no longer assume the user's sequences of actions simply by knowing the task specifications (e.g., user goals), we realized that we would have to obtain this information from other sources. To finally achieve the goals of the analysis, we superimposed our methods for capturing the user's sequence of actions on the Ofan model. This effort provided a much sought-after view of the user's sequence of action in the context of the demands from the environment and the detailed behavior of the machine.

CONTRIBUTIONS

It is now appropriate to ask: What is the added value of this effort to the research, design, and operational communities? We believe that the contributions of this work to our understanding of human interaction with machines that employ modes lie in three primary areas: *models*, *modes*, and *patterns of interactions*.

Models

This research effort has demonstrated that in analyzing mode-based systems with automatic and default transitions, a model of either the human alone or the machine alone is insufficient to capture the interaction. In 'static' human-machine systems, the state of the user usually mirrors the state of the device. If the user presses a button, the machine will stay in the corresponding mode —until the user acts again. Therefore, one can explicitly model the states of the user and implicitly assume the corresponding state of the machine. Such modeling provides much insight into the process of the interaction and allows the modeler to identify problems in the interface or human procedures, tasks, and goals. Conversely, if we model only the states of the machine, and represent the user actions as events that fire transitions, we can also obtain much insight into the human-machine interaction. In this approach, one models the machine explicitly, and the user's tasks implicitly.

However, when we march into highly dynamic domains and sophisticated control systems there are limitations to the above modeling approach. These systems are complex, having multiple interacting components and much autonomy. Automatic and default transitions are employed throughout these systems. In addition, global transitions, many of which are manual but some of which are also automatic, cause the system to change the configuration of many of its components as a result of a single event. In such control systems, this split, or dichotomy, between the model of the human and model of the machine, can no longer prevail. The machine takes action and changes states and modes based on internal and external events that are not easily perceived by the user. No longer can

one who models only the state of the user implicitly assume the corresponding state of the machine, or vice versa.

But just combining any two models will not allow us to model human interaction with such automatic control systems. We note that most mode problems emerge as cases of asynchronization between the user's state and the machine's mode configuration. To capture and quantify these cases of asynchronization between the human component and the machine component, a representational common denominator is required between the two models.

Finally, we address the issue of modeling the operational environment. For the purpose of analyzing human interaction with consumer electronics and similar devices, we can implicitly assume the current state of the environment. We can reasonably represent the processes in the environment as events in both the human model or the machine model. In that way, the demands and effects of the environment are represented implicitly. For most simple systems, this is indeed sufficient.

Nevertheless, this modeling assumption collapses in the face of a supervisory control system. Our findings from the field study demonstrate that such control system are event driven and strongly coupled to the operating environment. The processes and states of the environment directly affect the operator's mode selection. Furthermore, the state of the machine (e.g., its set of admissible mode configurations) dynamically changes as a function of external events coming from the environment. And again, for these types of systems we can no longer simply assume the corresponding state of the environment. Instead we have to model it explicitly.

The Ofan framework contains five modules: *Environment*, *User Tasks*, *Interface*, *Control Mechanism*, and *Plant*, all interconnected to represent the complete system. Each module in the framework represents a separate process. All the processes are represented as states and transitions. Using Statecharts as our modeling language, we are able to denote the hierarchical structure of most mode-based systems. Concurrency, another feature of Statecharts, allows us to view each individual module and the entire framework as one big concurrent process. Finally, the glue that combines all these concurrent processes into a single system is the Statecharts broadcasting mechanism. Every state transition is broadcast to the entire network, causing transitions in separate processes and modules to take place. And thus the system is all interconnected to represent the interactions in a complete environment-human-machine system.

Underlying the structure of the Ofan framework are the concepts of independence and dependence. The five modules are independent because they are all active, separately, at the same time. Yet they are also dependent, because events within a given module are contingent upon the events in another. At every "step" (initiated with the arrival of an event), the entire model resonates until no more internal events and transitions are fired—and the system settles back into a steady state. In summary, the representation allows us to map the territory and model some of the pertinent interactions in the environment-human-machine system.

In surveying the topic of models, we have also realized that selection of the modeling approach depends on what is to be described and revealed (Rumbaugh, Blaha, Premerlani, Eddy, and Lorenzen, 1991). The model must be structured so as to capture the types of problems that one is looking for. To model a complex system made of multiple components with modes and reference values, one must develop a modeling template. Otherwise each model becomes an ad hoc solution. These templates must be based upon a theoretical foundation of the behavior of the machine, given what we want the model to highlight and bring to the surface. We believe that such modeling

templates are a prerequisite before one can march into a modeling effort of a complicated system. In this thesis we have developed three such templates: *mode and reference value*, *supervisory mode hierarchy*, and *user task templates*.

The integration of the environment, user task, and machine processes into a single framework allows us to map the entire territory. Using this map we draw on it the interrelationships that bound these processes into a system. By applying the modeling templates, we are able to model a variety of different human-machine systems that employ modes in a consistent manner. In addition, the modeling templates allow us to highlight and bring to the surface some of features that contribute to mode ambiguity and error in such systems.

Modes

Using the model, we are finally able to describe the underlying structure of mode-based systems. We note how these system behave and identify constructs in which mode ambiguity exists. *Mode ambiguity*—a term that was initially proposed by Monk (1986) and Thimbleby (1982)—is employed here to describe the user's inability to resolve the next state of machine. Whether mode ambiguity will indeed lead to confusion depends on the user's task specification. If this resolution is *not* part of the task specification, no confusion will occur—the user simply does not care. If this resolution is indeed part of the user's goals, the user will get confused when the next mode configuration of the machine does not coincide with expectations.

This research has also allowed us to pinpoint some of the unique features that may contribute to mode ambiguity, confusion, and possible error. Several of these features are discussed below:

One striking feature is the case in which the display does not provide enough information to the user. In such system architecture the display is underspecified: Given the user's task specification, he or she cannot resolve the next mode configuration of the machine. At this point, no perceptual or cognitive arrangement of the display can solve the problem. The battle is lost before the war begins—the user is essentially set up for a mode confusion situation. We have identified several insufficient displays in simple devices and also in high-risk supervisory control systems. The results—mode ambiguity and potential for confusion and error—exist in both.

Another feature that leads to differences between user expectations and actual system state involves sequences of actions. We find that some very well-rehearsed sequences of action, some of them deeply rooted in our experience with devices and machines, can bypass all conscious efforts and produce mode confusion—even though the display is indeed sufficient and the user has complete factual knowledge about the working of the device. In cases where the sequence of actions involved in interacting with the device is very different than these well-rehearsed actions, confusion and error will sometimes occur. The problem is only compounded when the device can work differently according to the selected mode. If, for example, the device works in concert with these well-rehearsed actions most of the time, but then changes the rules (due to a mode change) in a way that is not harmonious with these deeply rooted experiences, mode confusion will indeed occur. Populations stereotypes and capture error are some of the psychological terms for this type of confusion.

In complex systems in which there is interaction between components, or separate processes, there may be more than one way to engage a given mode. This, however, is not necessarily a problem. It only becomes so in the context of a user who is not aware of the hidden transition—he or she cannot recall or does not have the knowledge about the existence of the transition, and has no way of sensing the event that triggers it. An interface that provides both types of information is the only

rescue, here. However, when the user is ‘told,’ via the interface, that there is only one given way to change a mode and it turns out there is another (hidden) way to do it—the user may get confused.

Another important characteristic of any mode-based system is the relationship between the engaged mode and the associated reference value. The reference value serves as a qualifier on the behavior of the mode—it constrains the mode behavior. In complex and dynamic systems, both modes and reference values can be changed automatically, manually, or by default. Such interdependency between modes and reference values appears to be a consistent source of mode ambiguity in complex systems. In order to overcome mode ambiguity, users must have an accurate knowledge of the interaction between the mode component and the reference value component, as well as the ability to sense events that trigger the transitions. Displays that portray this interaction, and thereby relieve the user of exercising one or more of these capabilities, are challenging to design.

When we mix the architecture of modes and reference values with well-rehearsed sequences of actions, we obtain, again, a very interesting feature of mode-based systems. The user not only has to be concerned with the mode transition, but has to be cognizant of the trailing reference-value trajectory. We have modeled several cases in which the reference value is maintained globally—i.e., independent of mode transitions. But there are also many cases in which the reference value is globally maintained in all modes, with one or two exceptions. Such system architecture may lead to mode confusion if the operators forget this relationship and the display does not provide much support—especially if the global values are maintained most of the time, but then change in one unique case.

Another common characteristic of a complex system is that there are several components, each with its own set of modes. In most cases the design is such that independence exists between the neighboring modes. Nevertheless, there are more than a few cases in which mode transitions in one component trigger a transition in a neighboring component. Several authors, notably Herbert Simon, argue that humans have difficulty understanding systems that have unique interactions among individual elements of each component. We are much better at understanding systems that are hierarchical in nature, or have interactions at a more global level. Understanding large concurrent processes in which interactions are far from trivial poses considerable difficulties for the human supervisor. Individual interaction between modes of several concurrent components may burden the user’s ability to recall this behavior at the right time. Finally, mode-mode interaction triggered by an automatic transition makes it more difficult for the user to keep track of the current mode and correctly predict the next mode configuration of the machine.

We have summarized here some of the design features of mode-based systems that induce mode ambiguity and may lead to confusion and error. We note that state (mode) dependency—the inherent feature of any complex system—is a thread that runs through many of these features. In Chapter VI we categorize these features in the Statecharts language. We call them the structural elements of a design, in the same way that the term is used in architecture to denote a unique (and perhaps problematic) element of the building. They are a set of design heuristics for identifying the potential for mode problems early in the design phase.

It is not our point to argue here that control systems should be made state *independent*. That is probably impractical. Instead, we argue that the above features should be made known to designers of these systems. Engineers and evaluation teams may use them to search for and identify similar structural elements in a given design specification. Once such structural elements are recognized, a design decision on how to solve the potential problem can take place.

Patterns of Interaction

After developing the Ofan framework, modeling many systems, and identifying some of the structural elements that produce mode ambiguity, we realized that we were still missing a piece of the interaction. In our models of simple human-machine systems, we had assumed the path users take in operating the device. In a large control systems that have many mode configurations and a variety of modes to perform a task, such assumptions do not hold anymore. Instead, we employed and developed other forms of describing human-machine interaction.

We employed an observation methodology to obtain rich and very detailed quantitative information about pilot interaction with modes of an Automatic Flight Control System. We showed how to categorize the states and modes of the machine. One can similarly categorize the operating environment and its demands on the user. The combination of the two types of categories that detail the structure of the machine and the structure of the environment allow us to investigate their mutual relationship quantitatively. To present mode usage we employ several representation schemes. Namely, we use the mode occupancy diagram, and mode transition diagram. Finally, we show how to add reference values and describe sequences of actions in mode transition diagrams.

The application of canonical correlation allowed us to highlight patterns of interaction between the state of the machine and the status of the operating environment. Such patterns of interaction, running criss-cross in the way users interact with a system, are critical pieces of information for designers and developers, as well as for researchers who can employ them for analysis of studies. In general, we apply a set of quantitative methodologies and analysis schemes in places where traditionally the only type of information has been mostly qualitative and anecdotal. We believe that the categorization of system and environment states, and as well as the variety of representation and analysis tools, can be employed for studying similar systems in this and other domains. Perhaps these methods can also provide some contribution to the developing field of usability testing.

The data indicate that pilots use only a subset of all possible mode configurations in operating the machine. With respect to the ones that are admissible, they convert a complex machine to a simple machine. The analysis identifies the factors that constrain this conversion process. Our findings point to a very strong relationship between mode selection on part of the pilot and the structure of the operating environment. So strong is this relationship that one can predict, with a high degree of reliability, the next mode configuration just by noting a few environmental factors. This finding has fundamental implications for the design of these control systems: (1) the operating environment must be well characterized, in terms of its structure and behavior, long before the plans for the design are laid out; (2) designers must pay extremely close attention to the relationship between this environment and the functionality of the machine, recognizing the strong effects of environment on user selection of machine functions; (3) the identification of task specification and the limitations on users in terms of performance (specification) and knowledge (specification) must be a critical part of laying out the interface design.

FUTURE RESEARCH

The scientific process embodied by this thesis started with observations and moved into descriptions (models). Based on these two phases, a categorization of the features that contribute to mode ambiguity has been proposed. Future research should focus on developing formal theories, and analysis techniques based upon them, that allow one to evaluate a system design in terms of these error-inducing features.

Finally, the closing of the loop of the scientific process must take place. That is, models and categorization scheme must be used to predict mode problems. In other words, we need to test the goodness-of-fit of the our ability to predict, at the early design phase, where these mode problems will indeed occur.

CLOSING REMARKS

We started with very confusing human-machine interaction topic and ended by identifying a specific problem that can be faithfully described and formally analyzed. The focus of attention was mode confusion and error—previously a poorly understood and ill-defined issue. We have cast a light on the three faces of modes: the good, the bad, and the ugly. On the good side, modes allow great flexibility for the user of fully and partially automated devices. Yet, they can also have a bad effect on the user, who is lost in the maze of the mode configuration space and the consequences of his or her actions on mode behavior. This may lead to the ugly—confusion, errors, and resulting incidents and accidents. The problem is not just academic: It raises its ugly head out of the earth ever so often and actually kills people.

We have tried to show, throughout this thesis, that mode ambiguity is independent of domain and operator expertise—it is not unique to the aviation domain, nor to complex systems in general, but instead, plays an important role in our interaction with any control system. Solutions to mode problems in one domain can certainly benefit the other domains. Furthermore, we note that the problem of human interaction with automation in general, and with modes in particular, is located in the seam between software engineering, control theory, and computer science, on the one hand, and our understanding of human-machine interaction, on the other. Such a problem therefore requires a multi-component and multi-disciplinary approach: No single discipline will be able to carry this problem to the finish line.

Modes are going to be with us for the long haul. Most human-machine systems include modes, and their employment will only increase as systems become more automated and have more concurrently running processes. The problem of mode ambiguity and resulting confusion is not a phantasm—it is complex and real like any construct that leads to human error. Yet this construct can be described, analyzed, and reasonably managed. We do not believe that any one approach, methodology, or algorithm can ever allow a designer to completely prune mode ambiguity from a given device or machine. Indeed, only a combination of methodologies, some from very different disciplines, can provide us with the conceptual resources to address this problem. We hope that some of the concepts and tools provided here will allow engineers and designers to understand such mode ambiguity, identify it, and obtain insights that will result in providing guards against it.

REFERENCES

- Abbott, K. (1996). The interface between flightcrews and modern flight deck systems. Washington, DC: Federal Aviation Administration.
- Aeronautica Civil of the Republic of Colombia. (1996). *American Airlines Flight 965. Boeing B-757-223, N651AA. Controlled flight into terrain. Near Cali, Colombia, December, 20, 1995*. Santa Fe De Bogota: Author.
- Agar, M. H. (1985). *Speaking of ethnography*. Beverly Hills, CA: Sage.
- Agre, P. E., and Shrager, J. (1990). Routine evolution as the microgenetic basis of skill acquisition. *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society* (pp. 694-701).
- Amalberti, R., and Deblon, F. (1992). Cognitive modelling of fighter aircraft process control: a step towards an intelligent on-board assistance system. *International Journal of Man-Machine Studies*, 36, 639-671.
- Andre, A., and Degani, A. (1996). Do you know what mode you're in? An analysis of mode error in everyday things. *Proceedings of the 2nd Automation Technology and Human Performance Conference*. Daytona Beach, FL: University of Central Florida.
- Ashby, R. W. (1956/1964). *An introduction to cybernetics*. New York, NY: Methuen.
- Attwood, M. E., Gray, W. D., and Jones, B. E. (1995). Project Ernestine: Analytical and empirical methods applied to a real world CHI problem. In M. Rudisill, C. Lewis, P. Polson and T. D. McKay (Eds.), *Human-computer interface design: success stories, emerging methods, and real world context* (pp. 101-121). San Francisco, CA: Morgan Kaufmann.
- Aviation Week and Space Technology*. (1995). Automated cockpits: who's in charge? Part 1 and 2. 142(5 and 6).
- Aviation Week and Space Technology*. (1996a). Airlines improve automation training. 145(10), 128-131.
- Aviation Week and Space Technology*. (1996b). The changing face of training. 145(10), 90-167.
- Aviation Week and Space Technology*. (1996c). Pilots, A-300 systems cited in Nagoya crash. 145, 36-37.
- Bainbridge, L. (1983). Ironies of automation. *Automatica*, 19(6), 775-779.
- Bainbridge, L. (1993). Types of hierarchy imply types of model. *Ergonomics*, 36(11), 1399-1412.

Baron, S. (1984). A control theoretic approach to modelling human supervisory control of dynamic systems. In W. B. Rouse (Ed.), *Advances in Man-Machine Systems Research* (pp. 1-47). Greenwich, Connecticut: JAI Press, Inc.

Baron, S., Muralidharan, R., Lancraft, R., and Zacharias, G. (1980). *PROCRU: A model for analyzing flight crew procedures in approach to landing* (NASA Contractor Report #152397). Moffett Field, CA: NASA Ames Research Center.

Baron, S., and Levinson, W. H. (1980). The optimal control model: Status and future directions. *Proceedings of the IEEE Conference on Cybernetics and Society* (pp. 90-100). Cambridge, MA: IEEE.

Bateson, G. (1972). *Steps to the ecology of mind*. New York: Chandler Publishing Company.

Bateson, G. (1979). *Mind and nature: A necessary unity*. New York: Bantam Books, Inc.

Billings, C. E. (1991). *Human centered aircraft automation: A concept and guidelines* (NASA Technical Memorandum 103885). Moffett Field, CA: NASA Ames Research Center.

Billings, C. E. (1996). *Human-centered aviation automation: Principles and guidelines* (NASA Technical Memorandum 110381). Moffett Field, CA: NASA Ames Research Center.

Bisantz, A. M., and Vicente, K. J. (1994). Making the abstraction hierarchy concrete. *International Journal of Man-Machine Studies*, 40, 83-117.

Blake, A. A., and Goeddel, W. C. (1993). *Using Statemate for early requirement validation*. Collins Commercial Avionics.

Boardman, J. T. (1995). Wholes and parts—A systems approach. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-25(7), 1150-1160.

Bogner, S. (1994). *Human error in medicine*. Hillsdale, NJ: Lawrence Erlbaum.

Bosser, T. (1986). Modeling skilled behavior and learning. *Proceedings of the 1986 IEEE International Conference on Systems, Man, and Cybernetics* (pp. 272-276). Atlanta, GA.

Bosser, T., and Melchoir, E. M. (1990). The SANE toolkit for user centered design, prototyping, and cognitive modelling. *Proceedings of the Annual ESPRIT Conference*. Brussels, Belgium: London, U.K: Kluwer Academic Publishers.

Bråthen, K., Nordø, E., and Veum, K. (1994). An integrated framework for task analysis and system Engineering: approach, example and experience. *International Journal of Human-Computer Studies*, 41, 509-526.

Buxton, W. A. (1986). Chunking and phrasing and the design of human computer dialogues. In H. J. Kugler (Ed.), *Proceedings of the IFIP 10th World Computer Congress Conference* (pp. 475-480). Amsterdam: North Holland.

Callantine, T. J. (1996). *GT-CATS: Tracking operator activities in complex systems* (NASA Contractor Report). Moffett Field, CA: NASA Ames Research Center.

Callantine, T. J., and Mitchell, C. M. (1994). A methodology for understanding how operators select and use modes of automation to control complex dynamic systems. *Proceedings of the 1994 IEEE International Conference on Systems, Man, and Cybernetics* (pp. 1751-1756). San Antonio, Texas.

Card, S. K. (1995). Pioneers and settlers: Methods used in successful user interface design. In M. Rudisill, C. Lewis, P. Polson and T. D. McKay (Eds.), *Human-computer interface design: success stories, emerging methods, and real world context* (pp. 122-172). San Francisco, CA: Morgan Kaufmann.

Card, S. K., Moran, T. P., and Newell, A. (1983). *The psychology of human computer interaction*. Hillsdale, NJ: Lawrence Erlbaum.

Casner, S. M. (1994). Understanding the determinants of problem-solving behavior in a complex environment. *Human Factors*, 36(4), 580-598.

Chambers, A. B., and Nagel, D. C. (1985). Pilots of the future: Human or computer? *Communications of the ACM*, 28, 1187-1199.

Chappell, S. L. (1994). Using voluntary incident reports for human factors evaluations. In N. Johnson, N. McDonald and R. Fuller (Eds.), *Aviation Psychology in practice* (pp. 149-172). Avebury Technical.

Christoffersen, K., Hunter, N. C., and Vicente, K. J. (1996). A longitudinal study of the effects of ecological interface design on skill acquisition. *Human Factors*, 38(3), 523-541.

Clark, J., and Holton, D. A. (1992). *A first look at graph theory*. Singapore: World Scientific.

Cliff, N. (1987). *Analyzing multivariate data*. San Diego: Harcourt Brace Jovanovich.

Cohen, S. M., Mitchell, C. M., and Govindaraj, T. (1992). Analysis and aiding the human operator in electronic assembly. In M. Helander and M. Nagamachi (Eds.), *Design for manufacturability: A systems approach to concurrent engineering and ergonomics* (pp. 361-376). London: Taylor & Francis.

Coleman, D., Hayes, F., and Bear, S. (1992). Introducing objectcharts or how to use Statecharts in object-oriented design. *IEEE Transactions in Software Engineering*, 18(1), 9-18.

Commission of inquiry. (1992). *Air Inter Airbus A-320, F-GGED, Strasbourg-Entzheim airport, January 20, 1992* (edited excerpts reprinted in *Aviation Week and Space Technology*, March 13, 1995—vol. 142, number 9). French Ministry of Planning, Housing, Transport and Maritime Affairs.

Commission of inquiry. (1995). *Airbus Industrie A-330, production #42, Toulouse-Blagnac airport, France, June 30, 1994* (edited excerpts reprinted in *Aviation Week and Space Technology*,

May 22, 1995—vol. 142, number 21). French Ministry of Planning, Housing, Transport and Maritime Affairs.

Cook, R. I., Potter, S. S., Woods, D. D., and McDonald, J. S. (1991). Evaluating the human engineering of microprocessor-controlled operating room devices. *Clinical Monitoring*, 7, 217-226.

Cook, R. I., Woods, D. D., Howie, M. B., Horrow, J. C., and Gaba, D. M. (1992). Unintentional delivery of vasoactive drugs with an electromechanical infusion device. *Cardiothoracic and Vascular Anesthesia*, 6(2), 238-244.

Cook, T. D., and Campbell, D. T. (1979). *Quasi-experimentation: Design and analysis issues for field settings*. Boston: Houghton Mifflin Company.

Corker, K., and Bejaczky, A. (1984). The effect of part-simulation of weightlessness on human control of bilateral teleoperation: Neuromotor considerations. *Proceedings of the Annual Conference on Manual Control*. Sunnyvale, CA: NASA.

Corning, G. (1979). *Supersonic and subsonic, CTOL and VTOL, airplane design* (4th ed.). College Park: University of Maryland.

Corwin, W. H. (1993). Autoflight mode annunciation: Complex codes for complex modes. In R. S. Jensen (Ed.), *Proceedings of the Seventh International Symposium on Aviation Psychology*. Columbus, OH: The Ohio State University.

Corwin, W. H. (1995). *Investigation of operational problems associated with flight mode annunciation*. Minneapolis, MN: Honeywell Technology Center.

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.

Dawes, R. M. (1979). The robust beauty of improper linear models in decision making. *American Psychologist*, 34, 571-582.

Degani, A. (1992). *On the typography of flight-deck documentation* (NASA Contractor Report 177605). Moffett Field, CA: NASA Ames Research Center.

Degani, A., Chappell, S. L., and Hayes, M. S. (1991). What saved the day: A comparison of traditional and glass cockpits. In R. S. Jensen (Ed.), *Proceedings of the Sixth International Symposium on Aviation Psychology Symposium* (pp. 227-234). Columbus, OH: The Ohio State University.

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. In R. S. Jensen (Ed.), *Proceedings of the Eighth International Aviation Psychology Symposium* (pp. 210-215). Columbus, OH: The Ohio State University.

Degani, A., and Wiener, E. L. (1990). *The human factors of flight-deck checklists: The normal checklist* (NASA Contractor Report 177549). Moffett Field, CA: NASA Ames Research Center.

Degani, A., and Wiener, E. L. (1994). *On the design of flight-deck procedures* (NASA Contractor Report 177642). Moffett Field, CA: NASA Ames Research Center.

Department of Transport. (1990). *Report on the Accident to Boeing 737-400 G-OBME near Kegworth, Leicestershire on 8 January 1989* (Aircraft Accident Report 4/90). London: Air Accident Investigation Branch.

Dix, A. J. (1991). *Formal methods for interactive systems*. London: Academic Press.

Drury, C. G., Paramore, B., Van Cott, H. P., Grey, S. M., and Corlett, E. N. (1987). Task Analysis. In G. Salvendy (Ed.), *Handbook of human factors* (pp. 370-401). New York: John Wiley.

Edgington, E. S. (1987). *Randomization tests* (2 ed.). New York: Marcel Dekker.

Edwards, E. (1976). Some aspects of automation in civil transport aircraft. In T. B. Sheridan and G. Johannsen (Eds.), *Monitoring behavior and supervisory control* (pp. 13-23). New York: Plenum Press.

Edwards, E., and Lees, F. P. (1974). *The Human Operator in Process Control*. London: Taylor and Francis.

Efron, B., and Tibshirani, R. J. (1993). *An introduction to the bootstrap*. New York: Chapman and Hall.

Einhorn, H. J., Kleinmuntz, D. N., and Kleinmuntz, B. (1979). Linear regression and process-tracing models of judgment. *36*(5), 465-485.

Eldredge, D., Dodd, R. D., and Mangold, S. J. (1992). *A review and discussion of flight management system incidents reported to the aviation safety reporting system*. Washington, DC: Federal Aviation Administration (FAA).

Fisk, A. D., and Kirlik, A. (1996). Practical relevance and age related research: Can theory advance without application? In W. A. Rogers, A. D. Fisk, and N. Walker (Eds.), *Aging and skilled performance: Advances in theory and applications*. Hillsdale, NJ: Erlbaum.

Fitts, P. M. (1951). Engineering psychology and equipment design. In S. S. Stevens (Ed.), *Handbook of experimental psychology*. New York, NY: Wiley.

Foley, J. D., and Wallace, V. L. (1974). The art of natural graphic man-machine conversation. *Proceedings of the IEEE*, *62*(4), 462-471.

Gaba, D. M. (1994). Automation in anesthesiology. In M. Mouloua and R. Parasuraman (Eds.), *Human Performance in Automated Systems: Current Research and Trends* (pp. 57-63). Hillsdale, NJ: Lawrence Erlbaum.

Ganong, W. F. (1979). *Review of medical physiology* (9th ed.). Los Altos, CA: Lange Medical Publications.

- Gentner, D., and Stevens, A. L. (1983). *Mental models*. Hillsdale, NJ: Lawrence Erlbaum.
- Goldberg, L. R. (1968). Simple models or simple processes? Some research on clinical judgment. 23, 483-496.
- Gray, W. D., John, B. E., Stuart, R., Lawrance, D., and Attwood, M. (1990). GOMS meets the phone company: Analytic model applied to real world problems. *Proceedings of the Human Computer Interaction Conference (INTERACT '90)*: Elsevier.
- Greene, B., and Richmond, J. (1993). Human factors in workload certification. *Proceedings of the Aerotech '93 Conference*. : Society of Automotive Engineers.
- Grudin, J. (1989). The case against user interface consistency. *Communications of the ACM*, 32(10), 1164-1173.
- Hansen, W. J. (1971). User engineering principles for interactive systems. *Proceedings of the AFIPS Conference. Volume 32*. (pp. 523-532).
- Hardy, M. A. (1993). *Regression with dummy variables*. Newbury Park, CA: Sage Publications.
- Harel, D. (1987). Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8, 231-274.
- Harel, D., and Naamad, A. (1995). *The Statechart semantics of Statecharts* (CS-95-31). Rehovot, Israel: Weizmann Institute of Science.
- Harel, D., and Pnueli, A. (1985). On the development of reactive systems. In K. R. Apt (Ed.), *Logics and models of concurrent systems* (pp. 477-498). Berlin: Springer-Verlag.
- Harel, D., Pnueli, A., Schmidt, J. P., and Sherman, R. (1987). On the formal semiotics of Statecharts. *IEEE*, ,
- Hoare, C. A. R. (1980). The emperor's old clothes. *ACM Turing award lectures: The first twenty years* (pp. 143-162). Reading, MA: Addison-Wesley.
- Hopcroft, J. E., and Ullman, I. (1979). *Introduction to automata theory, languages, and computation*. Reading, MA: Addison-Wesley.
- Huizing, C., and de Roever, W. P. (1991). Introduction to design choices in the semantics of Statecharts. *Information Processing Letters*, 37, 205-213.
- Hutchins, E. L. (1992). *An autoflight synoptic* (Third Ames AS/A Investigators Meeting). Moffett Field: NASA Ames.
- Hutchins, E. L. (1995). *Cognition in the Wild*. Cambridge, MA: The MIT Press.
- Indian Court of Inquiry. (1992). *Report on accident to Indian Airlines Airbus A-320 aircraft VT-EPN at Bangalore on 14th February, 1990*. Indian Government.

International Civil Aviation Organization. (1993). *Destruction of Korean Air Lines Boeing 747 (Flight KE 007) on August 31, 1983* (report number LE 4/19.4 - 93/68). Montreal: Author.

Irving, S., Polson, P. G., and Irving, J. E. (1994). A GOMS analysis of the advanced automation cockpit. *Proceedings of CHI'94: Human Factors in Computing Systems Conference* (pp. 344-350). Boston, MA.

Jacob, R. J. K. (1983). Using formal specifications in the design of human-computer interfaces. *Communications of the ACM*, 26(4), 259-264.

Jacob, R. J. K. (1986). A specification language for direct-manipulation user interface. *ACM Transactions on Graphics*, 5(4), 283-317.

Jensen, K. (1992). *Coloured Petri nets*. Berlin, Germany: Springer-Verlag.

Johnson, C. W., McCarthy, J. C., and Wright, P. C. (1995). Using a formal language to support natural language accident report. *Ergonomics*, 38(6), 1264-1282.

Jones, P. M., Chu, R. W., and Mitchell, C. M. (1995). A methodology for human-machine systems research: Knowledge engineering, modeling, and simulation. *IEEE Transactions on System, Man, and Cybernetics*, SMC-25(7), 1025-1038.

Jones, P. M., and Mitchell, C. M. (1987). Operator modeling: Conceptual and methodological distinctions. *Proceedings of the Human Factors Society 31st Annual Meeting Conference* (pp. 31-40). New York: Human Factors Society.

Kessel, C. J., and Wickens, C. D. (1982). The transfer of failure-detection skills between monitoring and controlling dynamic systems. *Human Factors*, 24, 49-60.

Kieras, D. E. (1988). Toward a practical GOMS model methodology for user interface design. In M. Helander (Ed.), *Handbook of human-computer interaction* (pp. 135-157). Amsterdam: Elsevier.

Kieras, D. E., and Polson, P. G. (1982). *An outline of a theory of the user complexity of devices and systems* (Working Paper No. 1). University of Arizona and University of Colorado.

Kieras, D. E., and Polson, P. G. (1985). An approach to the formal analysis of user complexity. *International Journal of Man-Machine Studies*, 22, 365-394.

Kieras, D. E., Wood, S. D., Abotel, K., and Hornof, A. (1995). GLEAN: A computer-Based tool for rapid GOMS model usability evaluation of user interface design. *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST'95)*.

Kirlik, A. (1993). Modeling strategic behavior in human-automation interaction: Why an "aid" can (and should) go unused. *Human Factors*, 35(2), 221-242.

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.

Kirlik, A., Kossack, M. F., and Shively, R. J. (1994). *Ecological task analysis: Supporting skill acquisition in dynamic interaction* (Unpublished manuscript). Center for Human-Machine Systems Research, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA.

Klatzky, R. L., Kober, N., and Mavor, A. (1996). *Safe, comfortable, attractive, and easy to use: Improving the usability of home medical devices*. Washington, D.C: National Research Council.

Klein, G. A., and Calderwood, R. (1987). Decision models: Some lessons from the field. *Proceedings of the 1987 IEEE International Conference on Systems, Man, and Cybernetics* (pp. 247-251). Washington, DC.

Lee, A. T., and Lozito, S. (1989). Air ground communications in the National Airspace System. In R. S. Jensen (Ed.), *Proceedings of the Fifth International Symposium on Aviation Psychology*. Columbus, OH: The Ohio State University.

Lee, J. D., and Sanquist, T. F. (1993). Systematic evaluation of technological innovation: a case study of ship navigation. *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics* (pp. 102-107). Le Touquet, France.

Leveson, N. G., Heimdahl, M. P. E., Hildreth, H., and Reese, J. D. (1994). Requirements specification for process-control systems. *IEEE Transactions on Software Engineering*, 20(9), 684-707.

Leveson, N. G., and Stolzy, J. L. (1987). Safety analysis using Petri nets. *IEEE Transactions on Software Engineering*, SE-13(3), 386-397.

Lidén, S. (1994). *The evolution of flight management systems*. Phoenix, AZ: Honeywell Air Transport Systems.

Lin, L., Isla, R., Doniz, K., Harkness, H., Vicente, K. J., and Doyle, D. J. (1995). *Applying human factors to the design of medical equipment: Patient controlled analgesia* (CEL 95-06). Cognitive Engineering Laboratory, University of Toronto.

Loveless, N. E. (1962). Direction-of-motion stereotypes: A review. *Ergonomics*, 5(2), 357-383.

Lyall, E. A. (1990). *The effects of mixed fleet flying of B-737-200 and B-737-300*. Unpublished doctoral dissertation, Arizona State University.

Lyall, E. A. (1992). The effects of mixed fleet flying of the 737-200 and -300. *Proceedings of the Human Factors Society 36th Annual Meeting Conference* (pp. 35-39). Atlanta, GA: Santa Monica, CA: Human Factors Society.

Mandel, M. (1989). *Making good time: Scientific management, the Gilbreths, photography and motion, futurism*. Santa Cruz, CA: California Museum of Photography.

Mark, M. A., and Greer, J. E. (1995). The VCR tutor: Effective instruction for device operation. *Learning Sciences*, 4(2), 209-246.

McDaniel, J. W. (1988). Rules for fighter cockpit automation. *Proceedings of the IEEE National Aerospace and Electronics Conference* (pp. 831-838). New York: IEEE.

McFarland, D., and Bossert, T. (1993). *Intelligent behavior in animal and robots*. Cambridge, MA: MIT Press.

McLucas, J. L., Drinkwater, F. J., and Leaf, H. W. (1981). *Report of the president's task force on aircraft crew complement*. Washington, DC: U.S. Government Printing Office.

McCulloch, W., and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5, 115-133.

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

Miller, L. H., and Johnson, J. (1995). the Xerox Star: An influential user interface design. In M. Rudisill, C. Lewis, P. Polson and T. D. McKay (Eds.), *Human-computer interface design: success stories, emerging methods, and real world context* (pp. 70-100). San Francisco, CA: Morgan Kaufmann.

Miller, R. A. (1979). *Identification of finite state models of human operations* (Dept. of Industrial and systems Engineering project # 784556 AFOSR #77-3152). Columbus, OH: The Ohio State University.

Miller, R. A. (1985). A systems approach to modeling discrete control performance. In W. B. Rouse (Ed.), *Advances in Man-Machine Systems Research* (pp. 177-248). Greenwich, CT; London: JAI Press Inc.

Miller, R. A., Jagacinski, R. J., Nalavade, R. B., and Johnson, W. W. (1982). A finite-state description of coordination in a two-handed target acquisition task. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-12(4), 529-538.

Milner, R. (1989). *Communication and concurrency*. New York: Prentice Hall.

Mitchell, C. M. (1987). GT-MSOCC: A domain for research on human-computer interaction and decision aiding in supervisory control systems. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-17(4), 553-572.

Mitchell, C. M., and Miller, R. A. (1986). A discrete model of operator function: A methodology for information display design. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-16(3), 343-357.

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*, SMC-17(4), 573-593.

Modugno, F., Leveson, N. G., Reese, J. D., Partridge, K., and Sandys, S. D. (1996). Integrated safety analysis of requirements specifications. *Proceedings of the Third IEEE International Symposium on Requirements Engineering*. Annapolis, MD.

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.

Mooney, C. Z., and Duval, R. D. (1993). *Bootstrapping: A nonparametric approach to statistical inference*. Newbury Park, CA: Sage.

Moray, N., Lootsteen, P., and Pajak, J. (1986). Acquisition of process control skills. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-16(4), 497-504.

Morton, J., Barnard, P., Hammond, N., and Long, J. B. (1979). Interacting with a computer: A framework. In E. J. Boutmu and A. Danthine (Eds.), *Proceedings of the IFIP Teleinformatics '79 Conference* (pp. 201-208). Paris.

National Transportation Safety Board. (1990). *USAir, Inc., Boeing 737-400, N416US. LaGuardia Airport. Flushing, New York. September 20, 1989* (Aircraft Accident Report, NTSB/AAR-90/03). Washington, DC: Author.

Neter, J., Wasserman, W., and Kutner, M. H. (1990). *Applied linear statistical models* (3rd ed.). Homewood, IL: Irwin.

Newell, A., and Simon, H. A. (1972). *Human problem solving*. Englewood Cliffs, NJ: Prentice-Hall.

Newman, T. P. (1996). Human factors problems in “glass cockpit” aircraft—a review. *Proceedings of the Royal Aeronautical Society “Future Flight Deck” Conference* (pp. 4.1-4.7). London, UK.

Nishizeki, T., and Chiba, N. (1988). *Planar graph: Theory and applications*. Amsterdam: North Holland.

Norman, D. A. (1983). Design rules based on analysis of human error. *Communications of the ACM*, 26(4), 254-258.

Norman, D. A. (1988). *The psychology of everyday things*. Basics Books.

Norman, D. A. (1990). The ‘problem’ with automation: inappropriate feedback and interaction, not ‘over-automation’. *Phil. Trans. Research Society London, B* 327, 585-593.

Orasovich, J. H., and Woods, D. D. (1995). *Users as designers: how people cope with poor HCI design in computer-based medical systems*. Ohio State University: Cognitive Systems Engineering Laboratory.

Palanque, P., and Bastide, R. (1996). A design life-cycle for the formal design of user interface. *Proceedings of the BCS-FACS Workshop on the Formal Aspects of Human Computer Interaction*. Sheffield, U.K.

Palmer, E. A. (1995). "Oops, it didn't arm"—A case study of two automation surprises. In R. S. Jensen (Ed.), *Proceedings of the Eighth International Aviation Psychology Symposium*. Columbus, OH: The Ohio State University.

Parnas, D. (1969). On the use of transition diagrams in the design of a user interface for an interactive computer system. *Proceedings of the 24th Annual ACM Conference* (pp. 379-385).

Peterson, I. (1991). Pick a sample. *Science News* (July 27).

Peterson, J. L. (1977). Petri nets. *Computing Surveys*, 9, 223-252.

Poller, M. F., and Garter, S. K. (1983). A comparative study of moded and modeless text editing by experienced editor users. *Proceedings of the CHI'83: Human Factors in Computing Systems Conference*. Boston, MA: New York: Association for Computing Machinery (ACM).

Poller, M. F., and Garter, S. K. (1984). The effects of modes on text editing by experienced editor users. *Human Factors*, 26(4), 449-462.

Polson, P. G., and Degani, A. (1995). *Flight mode annunciation: Solution or problem*. Presentation to the FAA human factors task force on automated cockpits (Abbott, K. and Slotte, S., Co-Chairpersons).

Polson, P. G., Irving, S., and Irving, J. E. (1994). *Application of formal models of human-computer interaction to training and use of the control display unit* (Contract No. DTFA01-91-00028). Washington, DC: Federal Aviation Administration.

Proctor, P. (1996). Friendly-fire probe continues. *Aviation Week and Space Technology*, 145(1), 13.

Ramsey, M. I. (1991). Blood pressure monitoring: Automated oscillometric devices. *Clinical Monitoring*, 7(56-67).

Rasmussen, J. (1985). The role of hierarchical knowledge representation in decisionmaking and system management. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-15, 234-243.

Rasmussen, J. (1986). *Information processing and human machine interaction: An approach to cognitive engineering*. New York: Elsevier.

Reason, J. (1990). *Human Error*. Cambridge, England: Cambridge University Press.

Reason, J. (1979). Actions not as planned. In G. Underwood and R. Stevens (Eds.), *Aspects of consciousness*. London: Academic Press.

Resnick, M. (1994). *Turtles, termites, and traffic jams: Explorations in massively parallel microworlds*. Cambridge, MA: MIT Press.

Reynard, W. D., Billings, C. E., Cheaney, E. S., and Hardy, R. (1986) *The Development of the NASA Aviation Safety Reporting System* (NASA Reference Publication 1114). Moffett Field, CA: NASA Ames Research Center.

Rochlin, G. I., La Porte, T. D., and Karlene, H. R. (1987). The self-designing high-reliability organization: Aircraft carrier flight operations at sea. *Naval War College Review*, 78-90.

Rogers, W. H., Logan, A. L., and Boley, G. D. (1989). *Classification and reduction of pilot error* (NASA Contractor Report 181867). Moffett Field, CA: NASA Ames Research Center.

Rosson, M. B. (1983). Patterns of experience in text editing. *Proceedings of the CHI'83: Human Factors in Computing Systems Conference*. 171-175: New York: ACM.

Rouse, W. B. (1986). *Systems Engineering Models of Human-Machine Interaction*. New York; Oxford: North Holland.

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

Rubinstein, E. (1979). The accident that shouldn't have happened. *IEEE Spectrum*, 16(11), 33-42.

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., and Lorenzen, W. (1991). *Object oriented modeling and design*. Englewood Cliffs, NJ: Prentice-Hall.

Sarter, N. B., and Woods, D. D. (1993). *Cognitive engineering in aerospace applications: Pilot interaction with cockpit automation* (NASA Contractor Report 177617). Moffett Field, CA: NASA Ames Research Center.

Sarter, N. B., and Woods, D. D. (1994). Pilot interaction with cockpit automation II: An experimental study of pilot's mental model and awareness of the flight management and guidance system. *International Journal of Aviation Psychology*, 4(1), 1-28.

Sarter, N. B., and Woods, D. D. (1995a). How in the world did we ever get into that mode? Mode error and awareness in supervisory control. *Human Factors*, 37(1), 5-20.

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

Scott, W. B. (1995). Certification officials grapple with flight deck complexity. *Aviation Week and Space Technology*, 142(5), 64-65.

- Segal, L. D. (1990). Effects of Aircraft Cockpit Design on Crew Communication. In E. J. Lovesey (Ed.), *Contemporary Ergonomics 1990* (pp. 247-252). London: Taylor & Francis.
- Shackel, B. (1967). Ergonomics research in automation. *Ergonomics*, 10(627-632).
- Sheridan, T. B. (1992). *Telerobotics, automation, and human supervisory control*. Cambridge, MA: MIT Press.
- Sheridan, T. B., and Farrell, W. R. (1974). *Man-Machine Systems: Information, Control, and Decision Models of Human Performance*. Cambridge, MA: The MIT Press.
- Sherry, L., and Polson, P. G. (1995). *A new conceptual model for avionics annunciation* (ICS Technical Report #95-08). Boulder, CO: Institute of Cognitive Science.
- Sherry, L., and Ward, J. (1995). A formalism for the specification of operationally embedded reactive systems. *Proceedings of the 14th AIAA/IEEE Digital Avionics Systems Conference*. Cambridge, MA.
- Simon, G., and Mayes, T. (1991). Cognitive task analysis? In G. R. S. Weir and J. L. Alty (Eds.), *Human computer interaction in complex systems*. London: Academic Press.
- Simon, J. L., and Bruce, P. (1991). *Probability and statistics the resampling way*. Resampling Project: College of Business and Management, University of Maryland, College Park.
- Simon, H. A. (1969/1981). *Sciences of the Artificial* (2nd ed.). Cambridge, MA: The MIT Press.
- Simon, H. A., and Gregg, L. W. (1979). Process models and stochastic theories of simple concept formation. In H. A. Simon (Ed.), *Models of thought*. New Haven: Yale University Press.
- Smith, D. C., Irby, C., Kimball, R., Verplank, W., and Harslem, E. (1982). Designing the STAR user interface. *Byte*, 7, 242-282.
- Smith, E. E. (1968). Choice reaction time: An analysis of the major theoretical positions. *Psychological Bulletin*, 69(2), 77-110.
- Smith, S. C., Govindaraj, T., and Mitchell, C. M. (1990). Operator modeling in civil aviation. *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics* (pp. 512-514). Los Angeles, California.
- Sneeringer, J. (1978). User-interface design for text editing: A case study. *Software-Practices and Experience*, 8, 543-557.
- Newman, W. M., and Sproull, R. F. (1979). *Principles of interactive computer graphics*. New York: McGraw-Hill.
- Suchman, L. (1987). *Plans and Situated Actions: The Problem of Human-Machine Communication*. Cambridge: Cambridge University Press.

- Tatsuoka, M. M. (1988). *Multivariate analysis: Techniques for educational and psychological research* (2nd ed.). New York: Macmillan Publishing Company.
- Tesler, L. (1981). The SmallTalk environment. *Byte*, 6(8), 90-147.
- Thimbleby, H. (1982). Character level ambiguity: consequences for user interface design. *International Journal of Man-Machine Studies*, 16, 211-225.
- Tukey, J. W. (1977). *Exploratory data analysis*. Reading, MA: Addison-Wesley.
- Turing, A. M. (1936). On computable numbers with an application to the Entscheidungsproblem. *Proceedings of London Mathematical Society*, 42, 230-365.
- Vakil, S., Hansman, R. J., Midkiff, A. H., and Vaneck, T. (1995). Mode awareness in advanced autoflight systems. In T. B. Sheridan (Ed.), *Proceedings of the International Federation of Automatic Control; Man-Machine Systems (IFAC-MMS) Conference*. Boston, MA: IFAC.
- von Bertalanffy, L. (1968). *General system theory: Foundations, developments, applications*. New York: George Braziller.
- Walker, N., and Catrambone, R. (1993). Aggregation bias and the use of regression in evaluating models of performance. *Human Factors*, 35(3), 397-411.
- Wasserman, A. I. (1985). Extending state transition diagrams for the specification of human-computer interaction. *IEEE transactions on Software Engineering*, SE-11(8), 699-713.
- Webster's. (1994). *Encyclopedic Unabridged Dictionary of the English Language*. New York, NY: Random House.
- Weinger, M. B., Scanlon, T. S., and Miller, L. (1992). A widely unappreciated cause of failure of an automatic noninvasive blood pressure monitor. *Clinical Monitoring*, 8, 291-294.
- Wickens, C. D. (1992). *Engineering psychology and human performance* (2nd ed.). New York: HarperCollins Publishers.
- Wickens, C. D., and Kessel, C. (1979). The effect of participatory mode and task workload on the detection of dynamic system failures. *SMC* 9(1), 24-34.
- Wiener, E. L. (1985). *Human factors of cockpit automation: A field study of flight crew transition* (NASA Contractor Report 177333). Moffett Field, CA: NASA Ames Research Center.
- 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.
- Wiener, E. L., Chidester, T. R., Kanki, B. G., Palmer, E. A., Curry, R. E., and Gregorich, S. E. (1991). *The impact of cockpit automation on crew coordination and communication: I. Overview, LOFT evaluations, error severity, and questionnaire data* (NASA Contractor Report 177587). Moffett Field, CA: NASA Ames Research Center.

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

Wilson, J. R., and Rutherford, A. (1989). Mental models: Theory and application in human factors. *Human Factors*, 31(6), 617-634.

Wolfson, H. A. (1934). *The philosophy of Spinoza*. Cambridge, MA: Harvard University Press.

Woods, D. D. (1993). Process tracing methods for the study of cognition outside of the experimental psychology laboratory. In G. A. Klein, J. Orasanu, R. Calderwood and C. E. Zsombok (Eds.), *Decision Making in Action: Models and Methods* (pp. 228-251). Norwood, New Jersey: Ablex.

Woods, D. D., Cook, R. I., and Billings, C. E. (1995). The impact of technology on physician cognition and performance. *Journal of Clinical Monitoring*, 11(1), 9-13.

Woods, D. D., O'Brien, J. F., and Hanes, L. F. (1987). Human factors challenges in process control: The case of nuclear power plants. In G. Salvendy (Ed.), *Handbook of human factors* (pp. 1724-1770). New York: Wiley.

Woods, W. A. (1970). Transition network grammars for natural language analysis. *Communications of the ACM*, 13, 591-606.

Xiao, Y. (1994). *Interacting with complex work environments: A field study and a planning model*. Unpublished Doctoral Dissertation. University of Toronto, Canada.

Zuboff, S. (1991). *In the age of the smart machine—The future of work and power*. New York: Basic Books.