2000

An approach to concurrently evaluate human task goals and machine behavior

David James Austin
San Jose State University

Follow this and additional works at: https://scholarworks.sjsu.edu/etd_theses

Recommended Citation

Austin, David James, "An approach to concurrently evaluate human task goals and machine behavior" (2000). *Master's Theses*. 2036.
DOI: <https://doi.org/10.31979/etd.f4hy-eyf5>
https://scholarworks.sjsu.edu/etd_theses/2036

This Thesis is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Theses by an authorized administrator of SJSU ScholarWorks. For more information, please contact scholarworks@sjsu.edu.

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA

UMI[®]
800-521-0600

**AN APPROACH TO CONCURRENTLY EVALUATE
HUMAN TASK GOALS AND MACHINE BEHAVIOR**

**A Thesis
Presented to
The Faculty of the Department of
Human Factors and Ergonomics
San José State University**

**In Partial Fulfillment
Of the Requirement for the Degree
Master of Science**

**By
David James Austin**

August, 2000

UMI Number: 1400649

Copyright 2000 by
Austin, David James

All rights reserved.

UMI[®]

UMI Microform 1400649

Copyright 2000 by Bell & Howell Information and Learning Company.

All rights reserved. This microform edition is protected against
unauthorized copying under Title 17, United States Code.

Bell & Howell Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

© 2000

David James Austin

ALL RIGHTS RESERVED

APPROVED FOR THE DEPARTMENT OF HUMAN
FACTORS AND ERGONOMICS

Dr. Kevin Jordan

Dr. Asaf Degani

Dr. Kevin Corker

APPROVED FOR THE UNIVERSITY

ABSTRACT

AN APPROACH TO CONCURRENTLY EVALUATE HUMAN TASK GOALS AND MACHINE BEHAVIOR

By David James Austin

Differences between descriptions of human tasks and machine behaviors make examining concurrent human-machine systems difficult. Most task analysis methods describe human operators textually, while engineering languages describe machine systems mathematically or graphically. A systematic methodology for describing human tasks, machine behaviors, and the resulting interaction is lacking because media and “language” incompatibilities exist between system designers, evaluators, and users.

The proposed methodology embraces both operator tasks and machine behaviors, allowing simultaneous evaluation of human-machine systems. Operator tasks are described as target states within a device's Finite-State-Machine description. The “game” of Russian Roulette illustrates the methodology's principles. Evaluations of a videocassette recorder and a commercial airliner's Automated Flight Control System are case studies illustrating the proposed methodology.

The proposed methodology can identify potential machine-induced errors and can pinpoint when the interface lacks all the information necessary for a user to operate a device correctly and efficiently.

Acknowledgements

I owe a debt of gratitude to Dr. Asaf Degani for his patience with me in this extended effort. His encouragement and understanding kept me going when I became frustrated with the project's apparently slow progress. Asaf's vision of the modeling shaped my work and his continued encouragement allowed me to use my experience and talents to persevere in this research which at times seemed impossible. Special thanks to Dr. Kevin Jordan who encouraged me to undertake my continued education and who supported and advised me on this current work. Thanks to Dr. Kevin Corker who brought new insights to bear on the modeling effort. I owe thanks to the Boeing Aircraft Corporation and, in particular, Dr. Michael Moodi and Dr. Randy Mumaw who generously gave of their time and extended effort with so little in return. Without their involvement this effort could never have been completed. Rowena Morrison, from the ASRS office, provided much needed editorial support to this project. Thanks to my fellow students Nancy Johnson and Maria Romera whose parallel efforts brought continuity to and will extend the current effort.

Special thanks to my companion in life, Linda, for her many sacrifices throughout this long process. Her understanding and (sometimes therapeutic) advice sustained me when the emotional weight of the work slowed my desire to continue.

TABLE OF CONTENTS

CHAPTER 1. INTRODUCTION.....	1
1.1 PROBLEM.....	1
1.2 APPROACH.....	1
1.3 OBJECTIVES.....	2
1.4 SCOPE.....	3
CHAPTER 2. FINITE STATE MACHINE LANGUAGES.....	4
2.1 MACHINE DESCRIPTIVE LANGUAGES.....	4
2.1.1 <i>State Transition Diagrams</i>	5
2.1.2 <i>Modes</i>	8
2.1.3 <i>Statecharts</i>	10
2.2 TASK ANALYSIS TOOLS.....	12
2.3 ERRORS.....	24
CHAPTER 3. CASE STUDY ONE: RUSSIAN ROULETTE.....	29
3.1 MACHINE BEHAVIOR AND USER INTERACTION.....	29
3.1.1 <i>Task Goal and Task Specification</i>	33
3.1.2 <i>Redesign</i>	34
3.2 SUMMARY.....	35
CHAPTER 4: CASE STUDY TWO: CONSUMER VCR.....	35
4.1 DESCRIPTION.....	36
4.1.1 <i>TV/VCR</i>	36
4.1.2 <i>Submenus</i>	40
4.1.3 <i>On-Screen</i>	41
4.2 EVALUATION.....	43
4.3 TV/VCR PROBLEM ONE: PROGRAMMING THE MENU DISPLAY.....	43
4.3.1 <i>Task Goals and Task Specifications</i>	43
4.3.2 <i>Analysis</i>	45
4.4 MACHINE BEHAVIOR AND USER INTERACTION.....	47
4.4.1 <i>Task Goal and Task Specification</i>	48
4.4.2 <i>Analysis</i>	49
4.5 POPULATION STEREOTYPE.....	51
4.6 USER MANUAL.....	52
4.7 DISCUSSION.....	54
4.8 SUMMARY.....	55
CHAPTER 5. CASE STUDY THREE: COMMERCIAL AIRLINER AUTOFLIGHT FUNCTION.....	56
5.1 DESCRIPTION: THE FMC COMPONENTS AND PILOT INTERFACE.....	57
5.2 DELETE NEXT ALTITUDE FUNCTION.....	61
5.3 EVALUATION.....	62
5.3.1 <i>Scenario of FMS and DNA in Flight</i>	63
5.4 DNA GENERAL MODELS.....	73

5.4.1 General Climb Model.....	74
5.4.2 General Cruise Model.....	76
5.4.3 General Descent Model.....	78
5.5 FMC EXAMINATION WITH TWO TASK GOALS AND TASK SPECIFICATIONS	79
5.5.1 Task Goal One: Mode Control Panel Altitude Set Changed While in VNAV(ALT CAP)	79
5.5.1.1 Analysis.....	83
5.5.2 Task Goal Two: Mode Control Panel Altitude Set Not Advanced to Next Constraint.....	84
5.5.2.1 Task Goal and Task Specification.....	87
5.5.2.2 Analysis.....	89
5.6 SUMMARY.....	90
CHAPTER SIX: DISCUSSION AND CONCLUSIONS.....	92
6.1 OBJECTIVES AND APPROACH	92
6.2 METHOD.....	93
6.3 LIMITATIONS.....	99
6.4 BENEFITS	100
6.5 FUTURE RESEARCH.....	101
REFERENCES.....	103
APPENDIX.....	106
1. ALTITUDE INTERVENE (DELETE NEXT ALTITUDE) FUNCTION CHART.....	106
2. STANDARD DEPARTURE AND STANDARD ARRIVAL ROUTING.	106
3. DELETE NEXT ALTITUDE STATE TRANSITION DIAGRAM.....	106

CHAPTER 1. INTRODUCTION

1.1 Problem

Human factors research into complex systems exposes many problematic features of human-machine interactions (Billings, 1996; Sarter & Woods, 1995). Languages exist to describe machine behaviors isolated from the actions of their human operators. Conversely, task analysis methodologies exist to describe only the human operator's tasks in the human-machine interaction. Synthesis of the interaction between an operator's tasks and the machine's behaviors remains difficult because of the descriptive differences of each methodology. The value of examining such interactions is particularly important in human-machine systems that have considerable autonomy, because the human supervisor must be aware of a machine's behavior in order to assume control, if necessary.

In some cases, the user sees an ambiguous interface; he or she cannot determine the current machine configuration or anticipate the next configuration from the available information. When the desired machine configuration is not achieved, an error may occur which may be potentially hazardous to the operator or system.

1.2 Approach

The approach proposed in this study uses two Finite State Machine representation languages (State Transition Diagrams and Statecharts) to model machines graphically.

Simple systems with few interactions, as well as complex systems with multiple subsystem interactions, may be described and examined with these languages.

The proposed methodology examines the human element in terms of a user's operational goal, or as it is labeled in this text --Task Goal. In addition, constraints or specifications are imposed on how a user is to achieve the specified Task Goal. For example, "drive the system to state A, unambiguously" is both a Task Goal and a Task Specification. The specification "unambiguously" means that there should never be a case where the user tries to drive the system to state A, but instead lands in state B due to lack of information on the interface. The interaction between the user and the machine is then examined to determine whether or not the operator can accomplish the task, given the Task Specification(s).

1.3 Objectives

The primary interest of the proposed methodology is to evaluate human-machine interfaces in general, and human-automation interfaces, in particular. The first objective is to develop a systematic and formal method for performing human-machine interaction evaluations. The proposed approach involves the synthesis of a machine description and its operator's task description to identify inconsistent or unexpected behaviors of the machine.

The second objective is to apply the proposed methodology to both simple and complex human-machine systems. Description of a simple machine is relatively

straightforward because a machine's behavior progresses in a linear fashion. However, a complex system has relationships among elements that complicate its description. Languages describing simple and complex systems are necessarily different, but the proposed methodology will be applied to both types of system's descriptions.

The third objective is to validate the proposed methodology. To this end, an existing complex system will be examined and design deficiencies described that were not identified by other evaluation methodologies.

1.4 Scope

This work focuses on a machine and its behavior, a set of operator tasks, and the method of accomplishing these tasks within the machine system. A machine's description will be in terms of its states and transitions toward system configuration goals. An operator's description will be in terms of Task Goals and Specifications toward the system goals. Exploration and explanation of human cognitive and perceptual capabilities are beyond the scope of this work.

In Chapter Two, two Finite State Machine languages will be examined, and a survey of human task analysis techniques presented. The purpose of the task analysis survey is to establish that current techniques do not fulfill the current goal of evaluating a machine system and a human operator simultaneously. In order to identify errors in the human-machine interaction, a concise definition of will be established.

The “game” of Russian Roulette will be presented in Chapter Three to describe the proposed methodology. Chapters Four and Five are case studies of a consumer videocassette recorder and a commercial airliner autoflight guidance system. Both devices are modeled and evaluated using this methodology.

The final chapter presents the conclusions, discusses methodology limitations, and makes suggestions for further research and development.

CHAPTER 2. FINITE STATE MACHINE LANGUAGES

2.1 Machine Descriptive Languages

The description of machine behavior begins with a discussion of states. Webster (1994) defines “state” as, “mode or condition of being; nature; condition.” The current state of a machine is a summary of all its past machine inputs; because the number of states in a particular machine is finite, that machine can be fully defined (Hopcroft & Ulman, 1979). A state change caused by an input event from either a human or from another machine is called a *transition*, and is considered to occur instantaneously, that is without any temporal delay. Like states, input events causing transitions are finite, in that for a given system, only certain input events can trigger transitions to defined states.

Machine descriptive languages based on the Finite State Machine concept are well developed. A.M. Turing, a mathematician, published the first definitive explanation of what has become known as a Finite State Machine theory (FSM). Turing (1936) theorized a machine capable of reading symbols from a tape, sequentially adding and

subtracting the symbol values and at the tape's end, producing a value equal to the sum of the values read. The keystone revelation in Turing's analysis is the process of sequential changes to the current state following an input. The FSM model Turing described has become an accepted methodology for machine description, and the basis for the design of computers.

2.1.1 State Transition Diagrams

After Turing, technological developments begged for languages richer in descriptive capabilities. A corresponding graphical notation was developed – the State Transition Diagram – to present the machine's behavioral information (states and transition) as a network of nodes and arcs (Hopcroft & Ulman 1979). State Transition Diagrams use this uni-dimensional graphical language to describe machine behavior as input/output relational events, i.e., state, event, state. A simple example of this concept is turning a room light on. If an operator assumes the initial light *state* is “OFF” (room dark), a transition *event* occurs when a person entering the room moves the switch up and the light changes to its “ON” *state* (room lighted). Additional extensions of this concept include Petri nets (Peterson, 1977) and Statecharts (Harel, 1987).

A State Transition Diagram representing a particular machine may be explained in terms of a finite set of states: simple machines have few interactions, while complex machines are characterized by multiple interactions between system components. Progression between states occurs when an initial machine condition or state receives an

input causing a transition to take place. Since the number of states and transitions for a given machine is finite, a complete description of a machine is possible in terms of initial states, transitions or inputs, and resulting states. Figure 2-1 depicts a simple State Transition Diagram.

Figure 2-1. State Transition Diagram.

Figure 2-1 describes a hypothetical machine with four states: A, B, C, and D. Labeled circles in the diagram represent discrete machine states, while directed lines or arcs with input symbols define transitions. Input events α , β , and χ trigger transitions between states A, B, C, and D. An additional transition, C to B, is triggered by γ . In all cases, a State Transition description is of the type: when event α occurs in state A, the system transitions to state B.

An important advantage of using State Transition Diagrams is that machine states and transition triggers are defined, and both are finite for a given machine. Machine behavior is thus expected to be consistent and predictable, a quality termed determinism, or deterministic behavior. The language describing the machine is deterministic finite automata (DFA). Figure 2-1 illustrates these qualities. Conversely, Figures 2-2(A) and 2-

2(B) depict non-determinism, or non-deterministic behavior, defined as two, or more outcomes to an undefined state(s) from the same input (Hopcroft & Ulman 1979). Such a condition is termed non-deterministic finite automata (N DFA). Unlike the DFA in Figure 2-1, an input α in state A may cause transitions to either states B or C (2-2(A)). Figure 2-2(B) is also an N DFA, since input α in state A may cause transitions (loop transition) to either state A or B.

Figure 2-2(A). Non-deterministic Behavior. Figure 2-2(B). Non-deterministic Behavior.

The issue here is that of predicting the next machine state from the current state. Accurate portrayal of the current machine state and all the information necessary to anticipate the next state, given a specific event (e.g., α), is deterministic behavior. Some human-machine interfaces, and in particular those that involve complex and automated functions, contain non-deterministic behavior (Degani, Heymann, Meyer, & Shafto, 2000). That is, sometimes the user will get what he or she asked for, and sometimes not.

The concept of determinism is critical to the understanding of error assessment. Machine behavior will appear non-deterministic to a user if an unexpected state results from an apparently correct input. In most cases, the machine itself is perfectly

deterministic, but the interface to the user renders the machine non-deterministic from the user's perspective. Such problems are commonly found in machines that employ modes.

2.1.2 Modes

“Mode” is defined in the Webster Dictionary (1994) as “(1) manner of acting or doing; method, way. (2) the natural disposition or the manner of existence or action of anything: form.” Degani (1999) characterizes a mode as a single machine behaving in different ways. Simply stated, modes represent the different configurations and corresponding behaviors of a single machine. In a simple machine capable of two functions, interpreting the interface provides a small potential for error because of feedback and complete system model. For example, a household light fixture with ON (lighted) and OFF (unlighted) states allows for little ambiguity between states (assuming a functional light bulb is installed).

The issue of modes is important because the likelihood for errors increases with mode proliferation. A simplistic illustration is a common three-way light bulb with 30-watt, 60-watt, and 90-watt configurations or modes (Figure 2-3). The transition from OFF to ON, 30 watts, is readily apparent. The 30-watt to 60-watt transition is somewhat discernable, while the 60-watt to 90-watt transition is less so. Perceived illumination is the normal interface, because the lamp conveys no information about the lumens produced, and the four switch positions are indiscriminated.

Figure 2-3. Three-Way Light Bulb.

Most people cannot easily discriminate among 30, 60, and 90 watts when walking into a room. The non-determinism of the lamp lies in the user's inability to discriminate between 30, 60, and 90 watts, and also the fact that the same action, which increases illumination, may also turn the lamp OFF. This leads to errors. Most people turn the light OFF and start counting the clicks to the desired light level.

Error probability can increase as modes increase. One measure of Human Error Probability (HEP) estimation for discrete tasks is the number of errors divided by the total number of opportunities for the error (Park, 1997). If the mode representation is not clearly displayed and understood, as with the light fixture example above, the probability for error may increase. On the other hand, well-designed machines, machines that "are easy to interpret and understand" and that "contain visible clues to their operation" (Norman, 1988, p. 2) reduce the total number of errors while the opportunity for error decreases and would therefore have low error probability.

The State Transition Diagram language described above can graphically depict discrete systems. Progressive expansion of complex systems frequently creates or requires extensive submode interactions. Complexity, in the current context, means the

interdependence of elements so that a change in one element will trigger changes in many other elements (Dorner, 1997). When more than one mode is involved, relationships between submodes may create a very large state boundary (state explosion) and potentially, a machine system description that cannot be interpreted in the State Transition Diagram formalism (Peterson, 1977). David Harel suggested one solution to this problem in 1987. This formalism is called Statecharts.

2.1.3 Statecharts

Statecharts is a visual formalism for modeling complex reactive machines. Reactive refers to a system that responds to internal and external inputs. The Statecharts formalism allows for depiction of superstates that embody a cluster of related substates. Degani (1996) develops a thermostat model to illustrate the Statecharts methodology. The thermostat in this heating system has two buttons –“on/off” and “mode.” Figure 2-4 illustrates a State Transition Diagram of the thermostat. A user at the control panel always initiates transitions between states. In this instance, transitioning from IDLE using the ON/OFF button by default starts the Fan. This is displayed on the direction line from IDLE to FAN. Sequential button selections change the thermostat modes from FAN to HEAT and to A/C and back to FAN. The system can be turned OFF from any of the modes (directed lines labeled “off”).

Figure 2-4. Thermostat in State Transition Diagram (from Degani 1996).

The same operational representation of the thermostat is given in Figure 2-5.

Figure 2-5. Thermostat in Statecharts (from Degani 1996).

The concept of hierarchy is represented in Figure 2-5 by the consolidation of modes into superstates when inputs are common to those modes. The superstate in this case embodies FAN, HEAT, and A/C with ON/OFF being common commands to each mode. Our representation now says that pressing “off” in any mode within the superstate will turn the fan OFF. Entry into the superstate with the “on” button from IDLE, will default to the FAN mode (indicated by the small arrow).

The two representations, State Transition Diagrams and Statecharts, are the means selected for the current work to describe simple and complex machines and perform analyses (such as the existence of non-determinism). Next, the human involvement in

system functions will be considered and representations will be identified that will allow description of the user's tasks.

2.2 Task Analysis Tools

A Guide to Task Analysis (Kirwan & Ainsworth, 1992) reviews 25 task analysis techniques. Kirwan and Ainsworth define a task analysis as “the study of what an operator (or team of operators) is required to do, in terms of actions and/or cognitive processes, to achieve a system goal.” The various analytic methodologies facilitate information collection and categorization intended to “blueprint” human involvement in a machine system.

The concern in this thesis is specifically concurrent evaluation of humans and machines. A review of the summary literature (Kirwan & Ainsworth, 1992; Salvendy, 1997) reveals few task analysis methodologies useful as both machine and system descriptive tools. Many methods describe the human within a system without necessarily describing the system, *per se*. For the purpose of this work, task analysis approaches are segregated into two types of presentation forms: textual and graphical. Graphical analysis is of special interest here because of its compatibility with State Transition Diagrams and Statecharts. User task methods may be further categorized as cognitive or behavioral. User's cognitive processes are implicit in the proposed methodology. Therefore the review of current task analysis methodologies is narrowed to *behavioral and graphical*.

Kirwan and Ainsworth (1992) in the Chapter titled “Charting and Network Techniques,” summarize a class of task analysis methodologies that graphically represent human tasks to complete a goal.

Gilbreth and Gilbreth (cited in Kirwan & Ainsworth, p.84) developed the first graphical charting technique (Process Chart) to decompose human-machine processes in top-down flow diagrams. Six action symbols tie operator behavioral elements to process elements. The six basic process chart symbols are shown below in Figure 2-6(A). An operation describes a human operator’s physical activity directly involving the product or process. Time, distance, and motion components of a process are conveyed through the transportation symbol. When a product is set aside for an unspecified period and will occupy space but be returned to the process later, a storage component is described. A time and motion component required by an inspection is described with the inspection symbol. A delay involving a process stoppage is described by the delay symbol. Finally, a combined operation, such as an inspection involving a physical motion component, is described as a combined operation. The six symbols described here are the accepted basic set, although there are a number of other formal symbols.

Figure 2-6(A). Six Process Control Symbols.

The example in Figure 2-6(B) describes a small portion of a hypothetical machine bolt production line process in which an operator first collects bolts from a production line container and fills an inspection tray. The bolts in excess of the tray container limits are then removed. After a partial product inspection, topside only, the tray is set aside to be retrieved later for further processing.

Figure 2-6(B). Example of Process Chart.

The method has a strong time element for physical distance and motion/time accounting. System operation, while important to this methodology, is secondary to operator performance within the system. The result is that only operator tasks in manual operations can be described. The process control methodology therefore does not adequately describe the human in a complex system.

Task analysis methods can include engineering languages such as Petri Nets. (Kirwan & Ainsworth, p. 91-92). van Biljon (1987) proposes a Petri Nets (PN) user

interface model. Petri Nets, an extension of the Finite State Machine theory, are graphs with nodes composed of *places* and *transitions*. A place usually represents a system state, while transitions represent events that affect a system state change. A marker or token is contained in the current system place (state) and moves to an output place when a transition from the marked place fires.

User actions are transitions in van Biljon's Petri Nets, and system outputs (places) are Task Goals. Figures 2-7(A) and 2-7(B) represent a computer-aided design (CAD) box drawing task. The application is shown in idle (token marker) in Figure 2-7(A). A user action, menu selection, is shown in Figure 2-7(B) by the solid bar with the action (menu, box) attached. User action moves the token marker to the state attached to the transition. In this example, the user elects to begin the box drawing in the upper right corner. An alternative construction is to begin the box drawing in the lower left corner and end in the upper right corner.

Figures 2-7(A) and 2-7(B). Petri Net Example of User Task in CAD Drawing.

The user elects to begin drawing the box in the upper right corner, dragging the image to the lower left corner. The marker moves appropriately to indicate this action (Figure 2-7(C)). The final action completes the task by moving the token to the end (Figure 2-7(D)).

Figures 2-7(C) and 2-7(D). Petri Net Example of User Task in CAD Drawing.

This methodology is attractive because it is formal and has a graphical representation. An operator's action towards a specific goal is represented from the starting place of a task to the final place, or completion, of a task. Petri nets do not allow depiction of hierarchy, and therefore a large or complex system's representation would pose the same problem as in a State Transition Diagram: state explosion.

Baber and Stanton (1994) propose a methodology – TAFEI (Task Analysis for Error Identification) – that links State Space Diagrams (SSDs) with Hierarchical Task Analysis (HTA). The SSD methodology is based on Finite State Machine theory. This integrated process relates human action to machine behavior, demonstrates sequences of user actions required to achieve a goal, and illuminates potential illegal actions. TAFEI's

authors believe that analysis of either a user, or a product in isolation, leads to a restricted assessment of places where error may occur. TAFEI is important because the Hierarchical Task Analysis and the system's descriptions are integrated.

Baber and Stanton develop a VCR programming task to illustrate TAFEI. First, they define system components and materials. Next, they generate a Space State Diagram (SSD) from the manufacturer's user manual describing VCR functions in terms of states (OFF, ON, CASSETTE INSERTED, etc.) and transitions between those states. Users' goals and tasks are described separately in an HTA format. The HTA information is mapped onto the SSD, producing a TAFEI diagram. In a final step, transitions between states are plotted in a matrix and examined for correctness and design conformance.

Figure 2-8. Space State Diagram (SSD) Explanation.

The Space State Diagram symbology in Figure 2-8 will be explained before introducing a full SSD. TAFEI SSD states are identified with numbers that allow correlation to the Hierarchical Task Analysis (HTA) numbering system in the later analysis stages. The state descriptor is a brief textual description of the state. One or more exit conditions may exist for a particular state, but each exit condition accepts specific inputs only from an HTA, or machine. The line from the exit condition is a

transition defined as the “relationship between the product’s current state, the goals of human, and the actions which either the human or product can perform.” The “By-Product” descriptor conveys that a state has a conditional by-product that it is, for example, potentially hazardous.

Figure 2-9 is the initial product of the TAFEI process, presenting an eleven-step system progression to program this VCR for a future time recording. Transition lines indicate transitions between exit conditions and next states. At this point, the input may be either human or machine. So for example, starting at State 1, the system is “awaiting power.” An unspecified input causes the transition from the exit condition, “waiting to be switched on,” to State 2, “VCR on.” The transition from State 2 exit condition, “waiting for cassette,” leads to State 3. The numbered circle following an exit condition e.g. State 3, “waiting to play” → (4.1) indicates the next state (which is described elsewhere). The purpose of the present Space State Diagram is to reach a time-programmed VCR state so the extended functions, such as 4.1 “play a tape” need not be described. However, an extension of this SSD could follow from State 4.1 that would fully explain playing a tape. The extensions could continue until each exit condition transition is fully explained. Note that the SSD description is flat, without hierarchy.

Figure 2-9. State Space Diagram (SSD) of a VCR Programming Task.

The next step in the TAFEI process describes the normative operator behavior expected to complete a task in a Hierarchical Task Analysis (HTA) format. The overall HTA goal “Program VCR for timer recording” is stated at the top of the diagram (Figure 2-10). Tasks are directly related to the overall goal, numbered according to order of completion, and textually defined, e.g. 1 Prepare VCR. Subtasks, representing actions needed to complete the antecedent task, are described numerically with the prefix of the task and assigned a suffix indicating the order of completion. A subtask may be further refined into subtask tasks, e.g. 4 – 4.1 – 4.1.1/4.1.2.

Figure 2-10. HTA for VCR Programming Task.

In Figure 2-10, five tasks with their subtasks are developed describing tasks to program a VCR. A solid bar under a subtask indicates the completion of that action. A series of actions to achieve a goal or subgoal can be grouped into a plan, allowing description of tasks toward a specific goal. Plan 0, for example, describes the HTA goal and requires all tasks be completed without specifying the subtask. Plan 1 explains specific subgoals involving operator actions, e.g. 1.1 Switch VCR on, 1.3 Insert cassette. When the HTA is complete for the specific tasks corresponding to the system SSD, the HTA it mapped onto the SSD.

Figure 2-9 represents states or actions that designers assume a competent user would perform to achieve the goal of recording a program. In contrast, the Hierarchical Task Analysis diagram in Figure 2-10 is the normative error-free performance of an expert. Subgoals from the HTA become the transitions from one state to another in the VCR Space State Diagram. The goal of the HTA is describing the human contribution to achieving a specific SSD product state.

Figure 2-11. Space State Diagram.

Figure 2-11 will be used to illustrate a basic TAFEI mapping. The VCR is “off” in State 1 with an exit condition, “waiting to be switched on.” A by-product or potential problem here is the condition “no power,” which could be the result of a power outage, machine malfunction, or the power simply not being switched “on”. The transition from “waiting to be switched on” is HTA step 1.1 that was defined as “switch VCR on.” After subtask 1.1 is complete, the system is in State 2, VCR ON and exit condition “waiting for cassette.” The transition from State 2 to State 3 is triggered by HTA

subtask 1.3, "insert VCR" (tape). And so the TAFEI continues until the final user goal, program record is reached.

Figure 2-12. TAFEI of VCR programming.

Figure 2-12 is a complete Space State Diagram-Hierarchical Task Analysis map describing the VCR program record sequence. Note particularly the flow of tasks from state to state, and in the case of State 3, from one of several expected exits to another state, i.e., the WAITING TO PLAY transition on subtask 4.1 (not described here). To proceed from one state to the next, appropriate triggers are required to exit the current state for the next expected state. If an inappropriate HTA subgoal is selected in a particular state, the desired transition may not occur; in TAFEI terms, the transition is illegal, the goal

unreachable, and an error occurs. (Arrows are not included on TAFEI transition lines to reduce the clutter.) A transition may also include a plan. Figure 2-10 describes plan 4, which in terms of the TAFEI mapping is the transition from state 6 to state 7.

The final TAFEI product is a matrix (Table 2-1) of Space State Diagram states defining transitions between VCR states that are “legal” (L), “illegal” (I), and unachievable (--).

	To State										
	1	2	3	4,5	5	6	7	8	9	10	11
From State	1	-	L	I	-	-	-	-	-	-	-
2	L	-	L	I	-	-	-	-	-	-	-
3	L	-	-	L	-	-	-	-	-	-	-
4,5	I	-	-	-	L	I	-	-	-	-	-
5	I	-	-	-	-	L	I	I	-	-	-
6	I	-	-	-	-	-	L	I	-	--	-
7	I	-	-	-	--	-	-	L	-	-	-
8	I	-	-	-	-	-	-	-	L	-	-
9	I	-	-	-	-	-	-	-	-	L	-
10	I	-	-	-	-	-	-	-	-	-	L
11	I	-	-	-	-	-	-	-	--	-	-

Table 2-1. TAFEI Matrix

In this matrix, a transition from state 1 (VCR OFF) to state 2 (VCR ON) is legal (L). Transition 1 (OFF) to 3 (insert VCR tape) is illegal because the machine must be ON for a tape to be inserted. A transition from state 4.5 (PROGRAM MODE) to state 5 (PROGRAM NUMBER SELECTED) is legal. However, a transition from state 1 (VCR OFF) to state 5 (PROGRAM NUMBER SELECTED) is unachievable and marked with a --.

By using the TAFEI methodology, designers may identify logic design flaws. The matrix product, in the TAFEI authors' view, may demonstrate actions that the designers had not intended or thought possible. The important point is once problems are identified, designers can consider solutions and make decisions about whether a flaw is consequential and whether or not product changes are needed.

The preceding TAFEI description highlights elements of the process that have value in human-machine descriptions. There are, however, limitations to the applicability of TAFEI to human-machine descriptions. First, TAFEI does not address important interface design problems, such as users knowing precisely what the present state of the machine is, or their inability to anticipate the next machine state. Second, the Space State Diagram element of TAFEI does not model "all possible" machine behaviors. Finally, the TAFEI Hierarchical Task Analysis uses domain experts, a process that may remove a class of errors that occur among new users.

The objectives of most task analysis methods are to evaluate human-machine systems, and to identify potential errors. In order to do this errors must be defined. The next sections will define error types and possible error causes.

2.3 Errors

A definition of human error is necessary in order to identify it in our human-machine interaction investigations. Numerous researchers (Reason, 1990; Norman, 1989; and Dorner, 1997) have studied the ubiquitous manifestations of human error and

assigned to them various terms like mistake, slip, and variability. The underpinning of human error study implies the notion of a performance standard, and deviations from that standard. Implicit in a standard is the willful intention to accomplish a specific act or task. However, execution or resultant outcomes of the act or task may be improper or incomplete, i.e., an error. Norman (1988) and Reason (1990) place error types into two categories: slips and mistakes.

Slips are assumed to be the result of a human's subconscious action. Most errors are slips: people intend one action but do another. Norman (1988) identified six slip categories resulting in errors. A description error occurs when an intended action has much in common with another task, but the incorrect action is accomplished. A capture error results when there is an intention to complete a specific task, but another more frequently accomplished and similar action intervenes. Automatic behavior from sensory inputs (reflex actions) inappropriately activated in a given situation cause data-driven errors. When a person thinks he/she is performing one action but in fact is carrying out another action because internally activated data supplanted our intention, this is termed an associative activation error. Finally, simply forgetting an action in the process of completing that associated task is a loss of activation error or lapse, and represents the failure to carry out an action. This failure then results in an error of omission, whereas previous slips were the result of errors of commission.

Mistakes, on the other hand, result from acts based on inappropriately chosen goals. Errors in this category are deliberate acts, neither mischievous nor malicious, whose intention is based on current perceptual input or memory. According to Norman (p. 114), “A person makes a poor decision, misclassifies a situation, or fails to take all the relevant factors into account.” Mistakes can result from active bias and heuristic shortcomings of perception, memory, or cognition.

The result of both slips and mistakes is the same: an action results in an unintended outcome. In this discussion, an error is defined as *an event that prevents the system from accomplishing its intended mission*. Errors in this context are uncategorized: they are neither minor nor catastrophic; they include acts of commission and omission; and they are primarily a mismatch between intent, action, and outcome. The question this paper will address is how to systematically and reliably identify human-machine error, when the error results from a human operator’s inability to accomplish an intended action given the information provided by the interface.

Human error, as described above, is a causal factor in seventy percent of aviation accidents (Boeing, 1998). The attribution of mishap causation as human error may be misleading, because where machines are involved, some aspect of the machine (e.g., its logic and/or its interface design) may also be contributory.

Reason (1990) drew an important distinction between local triggers and latent failures while examining five major accidents: the Chernobyl meltdown, the NASA

Challenger explosion, the Bhopal, India chemical disaster, the ferry boat Herald of Free Enterprise sinking, and the Three Mile Island incident. Reason argues that latent failures are hidden system design flaws. In each of the five accidents cited, the latent failures were the result of a series of poor design decisions, task mismanagement, and poor maintenance decisions, many of which existed long before the local trigger immediately preceding the accident. The local trigger is the action immediately preceding the event defined as the accident. Collectively, these failures represented an accident waiting to happen. The local trigger was the tip of the iceberg.

Reasons' analytic methodology is a valuable tool for developing and enforcing safety systems in industrial settings. Frequently used in reaction to local triggers, Reason's methodology examines training, maintenance, operating procedures, system goals, and hardware defects in order to isolate and provide preventative measures. While effective in examining a problem system, it is beyond the normal analysis domain of Reason's methodology to undertake proactive scrutiny of an individual system to enhance a more global industrial safety program.

2.4 Summary

In this Chapter, two descriptive languages – State Transition Diagrams and Statecharts – have been examined that are widely employed to describe a machine's behavior (states and transitions) in relation to user interaction. Using examples drawn from a VCR programming task, several graphical task analysis tools were illustrated that are especially compatible with these languages. These tools include process charts, Petri

Nets, and Task Analysis for Error Identification (TAFEI) maps. TAFEI is an integrated analysis process that links Finite-Machine Space State Diagrams (SSDs) with Hierarchical Task Analysis (HTA) techniques. The result of a TAFEI analysis is a visual “map” of human action to machine behavior, the sequence of user actions required to achieve a goal, and potential illegal actions.

Finally in this Chapter, the human error definitions developed by Reason, Norman, and others were reviewed. A definition of human error for the purpose of further analyzing human-machine interactions in the context of this work is now proposed: *An error is an event that prevents the system from accomplishing its intended mission.*

Human error and mistakes were considered separately from the majority of possible machine-induced errors. Machine functions are fixed by hardware or software constraints. Therefore, a design error will generally be consistent; it will always take place given a certain sequence of events because machines behave deterministically. Human action, on the other hand, varies and may at times be either a slip or a mistake. If the operator simply erred in input selection and the output was consistent with the input, the error is considered human error. On the other hand, if user selection of an input was inconsistent with the performance goal due to an interface that did not provide all the necessary information, the error is considered a machine design error. The user, in this case, made a correct action selection based on the available information about a current machine, but drove the system into an unintended configuration.

In Chapter Three, the “game” of Russian Roulette is a case study to illustrate the proposed human-machine analysis method. The methodology combines two elements: (1) a Finite-State-Machine description of a pistol and (2) the user’s Task Goals.

CHAPTER 3. CASE STUDY ONE: RUSSIAN ROULETTE

The “game” Russian Roulette is defined as “a deadly game of chance in which a person spins the cylinder of a revolver holding only one bullet, aims at his head, and pulls the trigger” (*Webster, 1991*). Uncertainty, or user inability to predict the outcome of pulling the trigger, is the essence of the game and is the primary concern here.

3.1 Machine behavior and User Interaction

User interaction in Russian Roulette consists of two manual inputs; first, a spin of a revolver’s six chambered cylinder holding only one bullet, and second, a trigger pull. Feedback in this user-machine interface occurs only after the user’s second input. Figure 3-1 is a State Transition Diagram description of this machine.

Figure 3-1. State Transition Diagram of Russian Roulette Pistol.

The six numbered circles represent the six chambers of the revolver's cylinder. The α arcs are transitions that correspond to cylinder spins. In this figure all spins begin at cylinder 1. The transitions on β arcs correspond to trigger pulls. A complete game description would include six spins originating at each of the six chambers, but for brevity and clarity only spins originating from chamber 1 are illustrated in Figure 3-1.

Figure 3-2. An Example of Two Russian Roulette Action Sequences.

Figure 3-2 presents an example of the sequences of action while playing Russian Roulette. The bullet is in chamber 6 (dark). The first spin begins at chamber 1, moves 7 chamber rotations, and lands on 2. A trigger pull (β) rotates the cylinder to chamber 3 but no discharge results because the chamber is empty. Now the revolver is handed to the next player. This player's spin (α') begins in chamber 3 and lands in chamber 1 followed by a trigger pull (β') to chamber 2. Again, the chamber is empty and there is no discharge.

The two β transition trigger pulls (β and β') described in Figure 3-2 resulted in no discharge and are of no consequence because the bullet in the Russian Roulette “game” occupies chamber 6, the “Danger” state. However, as a result of a subsequent chamber spin, the cylinder lands at chamber 5. A trigger pull (β) now will result in a transition to Danger: a discharge.

Figure 3-3. “Danger” State Identified.

Figure 3-3 represents the Russian Roulette system with all (β) transitions identified. Transitions (β) within the “Don’t Care” region are not of concern because a bullet discharge cannot result. The β transition from state 5 to state 6 will cause a bullet discharge and is dangerous to the player.

Figure 3-4 is the Russian Roulette system representation in which transitions between non-dangerous states appear as a single transition. This is a reduced description because all β transitions in the “Don’t Care” region result in the same end state: no

discharge. On the other hand, the transition from “Don’t Care” to “Danger” has a different outcome and therefore is displayed distinctly from other transitions.

Figure 3-4. “Danger” State Identified.

From the user’s point of view a trigger pull may result in a firing or no firing. Can the user anticipate the outcome of the β transition? The answer is no. The user has no means of reliably identifying and discriminating between states and subsequent transitions. Without knowledge of the current state, the user cannot predict the transition to the unsafe state. This is the whole point of the game, of course.

Figure 3-5. Russian Roulette Ambiguity Defined.

In Figure 3-5, the content of Figure 3-4 is rearranged into the standard STD State Transition Diagram notation. It shows, again that transition β can result in more than one outcome. In the Chapter One introduction, finite state machine definitions established that a transition out of a state resulting in more than one outcome is non-deterministic. Figure 2-2(B) described this graphically. Figure 3-5 is identical to the non-determinism described in Figure 2-2(B) (depicted here in miniature for comparison).

In the next section terminology is introduced that will be used throughout this thesis for specifying the user's task and the task's specification, or constraint.

3.1.1. Task Goal and Task Specification

To evaluate a human-machine system, a clear description of the machine's behavior (e.g., Figure 3-1) and the user's sequence of actions to accomplish a task is needed (e.g., Figure 3-2).

In analyzing the system two terms must be defined: *Task Goal* and *Task Specification*. A Task Goal is a description of the user's goal. In Russian Roulette, the Task Goal is either to discharge the gun or, for as long as possible, to avoid the gun discharge. In this specific case, the Task Goal involves two user actions: first the cylinder spin, and second, pulling the trigger.

A Task Specification is a constraint on how the task actions are completed. An accepted Task Specification is that the user must be able to say, with absolute certainty, that the task is completed. In this case, the Task Specification should be, "Can a human

operator unambiguously determine if the trigger pull after a spin will result in a bullet discharge?” Clearly, as shown above, this specification is not met in the Russian Roulette game.

3.1.2 Redesign

Let’s say that you are tasked with the job of modifying the Russian Roulette game to give the player an opportunity to “play” the game but, if desired, to stop before a bullet discharges. In other words, you must make the “game” safe. The term “safe” means notifying a participant of the cylinder status following a spin if a transition out of that cylinder will be deadly.

An initial solution is to assign a human or machine to count the number of cylinder rotations on each “play” and stop the “game” if the trigger pull after a spin would cause a firing. This solution may be difficult to implement, but is still a viable option.

A second solution is to mechanically or electronically identify the transition from “Don’t care” to “Danger” and notify the “player.” Transitions into empty chambers are inconsequential for this analysis because they remain in the “Don’t Care” region. To resolve the danger zone ambiguity, a sensor could be mounted on the chamber before the bullet (e.g., cylinder 5) alerting the “player” that a bullet is in place and the next trigger pull (β transition) would result in a discharge. Now the question can be asked whether the Task Specification is met. The answer is “yes.” The game is now safe (deterministic) because the transition from Safe to Unsafe can be reliably identified.

3.2 Summary

The Russian Roulette example summarizes the basic tenets of the proposed methodology. Russian Roulette was first described as a type of Finite State Machine formalism with all possible responses. Next, the user's Task Goal was identified: to experience a discharge or no discharge. Then, a Task Specification was added: unambiguously determine if the next trigger will result in a bullet discharge. Finally, the user's Task Goal and the Task Specification were evaluated within the system description to evaluate the performance of the system and the operator.

The proposed methodology is next applied to portions of two complex systems. Case Study Two, a VCR, is examined in the Statecharts formalism. Case Study Three, the "delete next altitude" (DNA) of a modern commercial airliner's automatic flight guidance control, is described in the State Transition Diagram formalism.

CHAPTER 4: CASE STUDY TWO: CONSUMER VCR.

Next, a consumer videocassette machine (Emerson model 4000) will be described using a combination of State Transition Diagram and Statecharts formalisms. A Finite State Machine representation of a TV and VCR interaction allows us to identify non-deterministic structures in the design. The Statecharts language description of the VCR programming menu allows a structured description of the VCR's hierarchical menus.

4.1 Description

The following is a description of the TV/VCR and of the “on screen” programming menu. A complete system description will not be presented here. Instead, the focus our system’s examination and an operator will be confined to specific VCR functions.

4.1.1 TV/VCR

The Emerson 4000 functions include an “on-screen” programming MENU; VCR tape playing, VCR tape recording, tape fast-forward, and tape rewind, pause, and stop. The current VCR and TV description is confined to the “on-screen” programming MENU, its functions, and user interactions.

The user turns the VCR “on” and then selects its output for display on an attached TV using a hand-held remote control. Figure 4-1 is the initial “on-screen” programming MENU visible on a TV after the user selects VCR output.

Figure 4-1. VCR “On-Screen” menu.

The “up” and “down” keys on the remote control move the flashing square through the four top level menu items: CLOCK SET, TIMER PROGRAM SET, AUTO REPEAT, and LANGUAGE.

An “Enter” key press with the flashing square on a menu item opens that item’s submenu.

The TV and VCR system’s examination begins with a Finite State Machine (FSM) description (Figure 4-2) of the progressive keystrokes necessary to display the “on-screen” programming MENU.

Figure 4-2. TV/VCR Model.

The concurrent TV/VCR system’s presentation in the Figure 4-2 depicts four possible TV and VCR state combinations with the following regimen: the TV’s two operating modes, normal BROADCAST reception or VCR INPUT, are the top line items of each state. (Both the TV and VCR are electrically powered “on” in this description). To provide a signal for display on the TV, the user must first press the “tv/vcr” button (and then “play” or, in this case study, “menu”). The second line item is the VCR machine

output, (TV/VCR OFF or TV/VCR ON). On line three user selection of VCR function is depicted: e.g., “menu,” “play,” “rewind,” “fast-forward,” “stop,” “record,” or “pause”. The focus of this analysis is on using the “menu” function.

In this machine, if TV/VCR is electrically powered “on” but the output in the OFF state, the VCR will not provide a signal to the TV. That is, when VCR functions such as “menu” or “play” are selected, nothing will be seen on the TV screen unless the TV/VCR output is enabled.

States 1-4 in Figure 4-2 describe the possible system states and transitions required to display the “on-screen” program menu. In state 1, the starting point of this system’s description, the TV is in a normal BROADCAST mode, and the VCR TV/VCR and MENU are both OFF. What the user sees on screen is the TV broadcast.

In Figure 4-2, a “menu” button selection transitions the system from state 1 to state 2, but causes no change in the TV display. State 2 is the situation in which the TV is in a normal BROADCAST mode, the TV/VCR output OFF and the VCR MENU is selected. The “on screen” menu, however, is not visible because the internal TV/VCR output is OFF although the VCR is electrically powered. A “tv/vcr” button selection transitions from state 1 to state 3, but again causes no change in the TV display. State 3 is the condition in which the TV is in a normal BROADCAST mode, the TV/VCR output to the TV ON, but the VCR MENU OFF. In this state, VCR output goes to the TV, overriding normal broadcast reception. However, the TV display remains unchanged because the MENU output is OFF

and therefore the only information available for display is the normal TV broadcast. In state 4, the “on-screen” menu is finally displayed because both MENU and TV/VCR outputs are selected “on.”

Figure 4-3 contains the same four configurations described previously (in Figure 4-2). However, State 4 is expanded. Now, a more in-depth description of user and menu interaction can be undertaken using the Statecharts formalism.

Figure 4-3. Navigation Commands to Display the On-Screen Menu.

When entering state 4, what the user sees is the “on-screen” menu (Figure 4-1). This is defined as a superstate named ON SCREEN MENU (and detail its internal configuration, later).

4.1.2 Submenus

The superstate MENU in Figure 4-3, state 4, includes two substates that are hierarchically related to it. Each of these substates, arbitrarily labeled A and B, include two submenus of their own. Submenu A includes the menus CLOCK SET and TIMER PROGRAM NUMBER, which share common navigation commands. The submenus AUTO REPEAT and LANGUAGE are contained in submenu B because they also share common commands.

In Figure 4-3, state 4, the effect of pressing the “menu” and “enter” buttons is seen. Transitions originate and terminate at specific hierarchical levels (superstate or substate). Next consider the transitions going into and coming out of the superstate MENU. For example, examining a menu or enter transition reveals that each produces a different result when executed within the superstate MENU than within its substates A and B. In substate A, the “menu” command transitions from that substate to the superstate MENU. The “menu” command in substate B exits to the state previously defined as state 3: the condition in which the TV is in a normal BROADCAST mode, the TV/VCR output to the TV is ON, but the VCR MENU is OFF. An “enter” selection also transitions to different states from submenus A and B. “Enter” in submenu A exits to state 3, while “enter” within submenu B transitions within that submenu.

4.1.3 On-Screen

The Statecharts depiction in Figure 4-4 is a more detailed description of the on-screen menu, including all states and transitions within the substates and submenus.

Figure 4-4. VCR Main Menu Navigation.

In the “on-screen” menu, “up” and “down” keys cause cursor movement between menu items. Pressing “enter” opens the selected “on-screen” menu’s submenu. So for example, if CLOCK SET is the active item and “enter” is selected, the CLOCK SET submenu opens with the DAY field as the default selection (indicated by the ball-arc-arrow symbol). Within submenu A, each data field must receive an “enter” to progress through the menu. Notice that in the last fields of both CLOCK SET and TIMER PROGRAM SET, pressing “enter” transitions from those fields to state 3, a transition away from a specific region of the superstate MENU and the submenu A. “Menu” in submenu A transitions to the on-screen menu with CLOCK SET as the default selection.

To select AUTOREPEAT the user presses the “enter” button. The same applies for reaching the LANGUAGE submenu. Both AUTOREPEAT and LANGUAGE are encapsulated within the B substate. Pressing “enter” cycles between submenu items. Pressing “menu” at any time exits both B substates to state 3. Note that in submenu A, “enter” is the exit condition, while in submenu B, “menu” is the exit condition.

“Cancel” is active only in TIMER PROGRAM SET, and when selected, immediately returns the user to the top menu default entry item, TIMER PROGRAM NUMBER. In all other submenus, a “cancel” command creates no apparent transition.

Notice that when exiting the A substates on “enter” and the B substates on “menu,” returning to the on-screen menu requires another “menu.” So when additional programming is desired using the TIMER PROGRAM SET menu, the key selection sequence to

return from a previous programming task is: “enter,” “menu,” “down,” “enter.” The equivalent key sequence when exiting the B submenu is: “menu,” “menu,” “down,” “enter.”

4.2 Evaluation

Now that all the VCR’s substates and submenus have been modeled using Statecharts, the system is ready to be evaluated. The TV/VCR will be examined first and then the “on-screen” menu. The examination process begins by defining Task Goals and Task Specifications, and then proceeds to analyze the system accordingly.

4.3 TV/VCR Problem One: Programming the Menu Display

4.3.1 Task Goals and Task Specifications

Next the VCR system’s description will be examined and two user Task Goals formulated. The Task Goals represent routine user tasks and allow an examination of operator and system interaction. The first Task Goal is to display the programming MENU on the TV screen. The Task Specification is that the user must be able to perform this task reliably from the information the VCR interface presents. This specification requires that the interface “unambiguously” present the programming details so that there is never a case in which the user tries to drive a system to state A, but instead lands in state B due to lack of information from the interface. The system’s analysis proceeds in light of the Task Goal and the constraining Task Specification. The

interaction between the user and the machine is evaluated to determine if the operator can accomplish the task, given the Task Specifications.

Figure 4-5. Menu Display Commands.

Figure 4-5 is the TV and VCR in a Finite State Machine system description (as previously presented in Figure 4-2). Figure 4-5, state 1, shows that the TV is in BROADCAST, TV/VCR is OFF and MENU is OFF. Selecting “menu” does nothing from the user’s perspective; it puts the user in state 2 and the TV is still in BROADCAST. An additional selection, “tv/vcr”, is required to reach the Task Goal (state 4). While in state 1 and selecting “tv/vcr” the user ends up in state 3 and still sees only TV broadcast. An additional selection, “menu,” is required to reach the Task Goal (state 4).

From the user’s point of view, states 1, 2, and 3 are identical – only TV broadcast is seen. They are also indistinguishable by looking at the interface, one cannot tell which

state one is in. To emphasize this fact, in Figure 4-5, states 1-3 are encapsulated and call it state E.

4.3.2 Analysis

Figure 4-6 is an abstraction of the Figure 4-5 components, with states A, B, C, and D representing states 1, 2, 3, and 4 respectively. Transitions “ α ” and “ β ” represent “menu” and “tv/vcr” transitions. The earlier menu command descriptions established that states A, B, and C are shaded and called state E because these three states are indistinguishable. State D, however, is easily identified by the presence of the “on-screen” programming MENU. Previous progressive analysis showed that an “ α ” input in state A causes a transition to state B. A “ β ” transition from state A moves us to state C. States B and C, however, are indistinguishable.

Figure 4-6. Abstraction of Menu Display Commands.

The earlier menu command descriptions established that states A, B, and C are indistinguishable from the user’s point of view, so the three are now consider as a single state, E. The previous user input descriptions also established that inputs “ α ” and “ β ”

occur within state E and that the interface does not convey these state changes. Figure 4-7(A) shows that within state E, transition α will either have no visible effect, or trigger a transition to state D; and the same is true for transition β . The user may not know which transition will occur on the same input. What basically exists here is, of course, a non-deterministic machine (Figure 4-7(B)), identical to the one discussed in Chapter 2 (2-2(B)).

Figure 4-7(A) Non-determinism

Figure 4-7(B) Non-determinism

As a consequence of this non-deterministic design, both experienced and inexperienced users have a difficult task determining why the “on-screen” MENU is not available with the “*menu*” command. The user has no means of determining the current system state and therefore the next state after a transition selection. Feedback in this scenario would provide the user with an indication of the specific VCR condition (e.g., MENU ON and TV/VCR OFF) after which the user could determine the next machine state based on the next input.

A possible scenario for this ambiguity is the user walking into a room and assuming that the VCR is turned “on” with “menu” selected, but “menu” is not displayed on the TV. The current state is not annunciated for a new user; unless the user knows the

VCR's recent history, a mistake is likely. Another "menu" input would place the user in the state with TV/VCR and MENU OFF, while a "tv/vcr" input would activate the MENU even if the user were expecting to next press "play" and view the video just inserted.

4.4 Machine Behavior and User Interaction

Figure 4-8. Menu Navigation

The superstate MENU (Figure 4-8) contains two substates labeled A and B. Submenu A contains two additional states, CLOCK SET and TIMER PROGRAM NUMBER. Entry into any of these substates is accomplished by first selecting that menu item from the main “on-screen” menu, and then pressing “enter”.

As discussed earlier, state 3 is reached from superstate MENU with different user inputs (depending on which substate the system is in). In substate A, all data input fields must be complete before an “enter” command transitions out of CLOCK SET and TIMER PROGRAM NUMBER SET. A “menu” command anywhere within the A substate instantly exits to the superstate MENU.

AUTO REPEAT and LANGUAGE respond alike to “enter” and “menu.” “Enter” causes transitions between submenu items in a continuous loop, while “menu” initiated an exit to state 3. From within B, the superstate MENU is not directly available as it from within A.

4.4.1 Task Goal and Task Specification

The Task Goal is now to navigate from the on-screen menu through all the submenus, i.e., to visit all submenus. Entry into the submenus from the initial “on-screen” menu is consistent for all four submenus. Upon opening the “on-screen” menu, CLOCK SET is the default entry. (“enter” input at this point will open the CLOCK SET submenu). “Up” or “down” arrow key selections in the “on screen” menu move the

cursor to TIMER PROGRAM SET, AUTO REPEAT, or LANGUAGE. (“enter” input at this point would open the respective submenu).

Within the CLOCK SET and TIMER PROGRAM SET submenus, “enter” moves sequentially from the menu top to the bottom. At the menu’s end of both CLOCK SET and TIMER PROGRAM SET, the user transitions out of the superstate MENU to State 3 (TV BROADCAST and MENU “off” state). The effect for the viewer is that the VCR is now “off.”

Unlike “enter” in the A submenu, “enter” in the B submenu causes closed loop transitions between submenu items. “Menu” in the B submenu transitions the system back to state 3 (TV BROADCAST and MENU “off” state). This transition is again to an ambiguous state.

4.4.2 Analysis

Figure 4-9. Abstraction: Navigation from Submenus to Main Menu.

The Figure 4-9 is a simplification of the details provided in Figure 4-8 by removing submenu details and transitions labels. The E state, enclosed by the hashed lines, is composed of three states not differentiated to the user (discussed in the TV/VCR

section). Transitions from the substates A and B on an α event are either to superstate D (back to main MENU) or to state E. The “ γ ” command (which corresponds to “enter”) also evokes different transitions: from A and an inter-menu loop in B. Distinct results are visible to a user because a “ γ ” input in the substate B menu moves the cursor on screen from item to item in a loop. (The non-determinism discussed in the previous section is present in the α transition from substate B.)

The Task Goal is met. A user can program and visit all the submenus in this system. However, this task can be quite confusing because of the inconsistency in navigating transitions.

The Task Specification now asks if the user can reliably navigate from one submenu to the other without encountering ambiguity in the command selection. The answer is no. In one instance reentry to the program menu is direct, such as from submenu A back to the “on-screen” menu. In another instance, reentry is indirect. From submenu B, an exit from “menu” causes a transition to a state that is indistinguishable to the user (state E). A user attempting to navigate back to the main “on-screen” menu must know that “menu” takes him or her back to the “on screen” menu. This rule will work most of the time. However, if someone else makes any input into this system (e.g., press “tv/vcr”) the user will become confused. The user will not be able to discriminate in which one of the E states (1, 2, or 3) the system currently resides.

Confusion that results from ambiguous machine states and unexpected transitions lead to user frustration. The VCR's infamous flashing 12:00 is often cited as the antithesis of machine complexity and ambiguity when a user simply accepts defeat. The seemingly inconsistent transitions between menus identified in this system contribute to such user frustration.

4.5 Population Stereotype

An additional item about "on-screen" menu navigation relates to the population stereotype violation. A population stereotype is an accepted and expected result of an action within a given population (Wickens, 1992 p. 332). It is commonly held that a task which reflects a long history of experience serves as a major source of error if an expectation is not met. Loveless (1959) further observes, "It is a well established principle of ergonomics that the direction of movement of a control should be 'compatible' with the resulting movement of a display. Certain direction-of-motion relationships between controls and displays are expected by the operator, and performance may be poorer on arrangements which do not conform to these expectations." For example, in the United States an "up" movement of a light switch is expected to turn a light ON, while down is expected to be OFF. Also, by convention a user expects an up arrow to produce an upward movement, and conversely, a down arrow to produce downward movement on a selected object.

In the case of the VCR system, at the menu's top, a "down" arrow selection moves the cursor to the menu's bottom (LANGUAGE) as expected, and an "up" arrow goes from the bottom (LANGUAGE) to the top (SET CLOCK). But in-between the top and bottom, "down" goes up and "up" down. The "up" and "down" arrows in Figure 4-10 illustrate the motion which is opposite expectations and mentally confusing.

Figure 4-10. Arrow Movements.

4.6 User Manual

The user's manual may be expected to guide a user and to develop an understanding of the system. In fact, this particular manual does not always assist the user in system conceptualization, and may even confound the already confusing problem. A scenario may be developed for a user to view the "on-screen" menu in one of three preinstalled

languages and then return to the “on-screen” menu for programming tasks in the select language. The manufacturer’s guide (Emerson, p. 36) suggests the following selections: “menu”, “up/down,” “enter,” and “menu.”

Figure 4-11. Emerson User’s Manual, p. 36.

The system description discussed earlier (Figure 4-4) shows that after a “menu” command within the Language submenu, the system transitions to the ambiguous state of the MENU “off” and TV BROADCAST “on” state. In fact, after the “enter” command the user would have to press “menu” yet again, i.e., twice, for Main Menu activation. A “tv/vcr” button press in the idle state would place both the VCR and the MENU in idle modes

after which both a “tv/vcr” and a “menu” selection would resume programming or “tv/vcr” and a “play” would allow viewing of an available VCR tape.

4.7 Discussion

The two Task Specifications show that the VCR user must know the current VCR mode in order to reliably predict the next mode. The system is ambiguous when these specifications guide the system analysis. More accurately, under the assumption that an operator has no history with the last action (transition) taken, human-machine system is non-deterministic. The user cannot resolve which of two different transitions will occur after the user selects “tv/vcr” and “menu”: either the VCR programming menu is displayed, or no action apparently occurs. A trial-and-error process of eliminating “tv/vcr” and “menu” combination selections is one method of approaching this ambiguity. Another approach is to simply turn the machine “off” and begin selections again from a known state, i.e., OFF.

Earlier in the discussion of mode error, the descriptions used established that non-determinism, or the inability to anticipate the next system configuration, is one of the factors producing mode ambiguity. This results in user confusion and subsequent mode error. A machine system interface is a user’s source of information about a current machine state. This VCR interface does not aid the user in determining an active mode prior to reaching the “on-screen” menu, which is the first interface presentation of system status. Between power “on” and the “menu” display, the user has no idea of

system condition. This holds true even if the user has an absolutely accurate user-model of the system's behavior. The display is insufficient for accomplishing the Task Specification.

4.8 Summary

With the VCR example, the machine was described first. The methodology employed a combination of Finite State Machine and Statechart languages to describe machine behavior, human tasks, and the resulting interaction. Two instances of potential machine-induced errors are identified. Specifically, the interface does not provide all the information necessary to operate the device correctly and efficiently.

Statecharts language features allowed us to represent a structured description of user interaction with this system. The Task Goal defined an operator attempting to perform a task at a specific point in the system. Concurrent analysis of the Task Goal and the system, with a Task Specification constraint, reveals whether an operator can successfully complete a chosen task. In the two instances selected, clearly one could not have been completed and the second completed only with some difficulty. In each Task Goal analysis, abstractions of a particular function identified a problem. Additionally, a population stereotype violation was identified, a common human factors problem, during detailed examination of the system reaction to user input.

CHAPTER 5. CASE STUDY THREE: COMMERCIAL AIRLINER AUTOFLIGHT FUNCTION

The final application of the proposed methodology examines the *Delete Next Altitude (DNA)* function of a Flight Management Computer (FMC). The DNA function is part of the automatic flight guidance system installed in a modern commercial transport aircraft.

First a brief description of the components making up the FMC unit will be given, followed by a more detailed description of the DNA function and pilot interaction with it. Then a Finite State Machine model of the system will be built using a realistic flight scenario. Next, Task Goals and Task Specifications will be constructed and analyzed. Before describing the DNA function, a brief overview of FMC system components is presented to help develop a basic understanding of this system. There are four major components of the Flight Management System: the Flight Management Computer/Control Display Units, an Autopilot/Flight Director System, an Autothrottle, and the Inertial Reference Units. The Inertial Reference Units (IRS) will not be discussed because the pilot normally has no direct control of their functions.

When connected, these components comprise a completely automated system called the Flight Management System (FMS) that functions to control the aircraft's path over the ground (lateral navigation) as well as its altitude (vertical navigation). Lateral navigation (LNAV) guides the aircraft over specific geographical points (waypoints) that

correspond to the route of flight from the departure point to the destination. The Vertical Navigation (VNAV) component of the FMS controls the altitude in all phases of flight. The VNAV modes of interest in this investigation are VNAV SPEED, abbreviated VNAV SPD, VNAV PATH, and ALTITUDE HOLD (ALT HLD). Later, the DNA function is explored as part of the FMS's Vertical Navigation. But first, a brief familiarization discussion of the three major FMS components identified above is in order.

5.1 Description: The FMC Components and Pilot Interface

The Mode Control Panel (MCP) in Figure 5-1(A) is a control device through which the pilot provides input to the Flight Management Computer (FMC) for setting automatic modes. The two areas of interest to us are the Altitude Set window/knob and the "Delete Next Altitude" button. The Mode Control Panel location for the "DNA" button is unique because DNA is an FMC function that has its controls on the Mode Control Panel.

The Altitude Set knob on the Mode Control Panel (Figure 5-1(A)) directly affects both autopilot and FMC guidance. The Mode Control Panel altitude setting overrides and constrains the FMC during a climb, at cruise, and in descent. For example, if the FMC is programmed to climb to 10,000' but the Mode Control Panel is set at 5,000, in a climb the aircraft will level at 5,000.' This Mode Control Panel/FMC altitude control relationship will be significant to the DNA investigation.

**Figure 5-1. (A) Mode Control Panel. (B) Electronic Attitude Director Indicator
(C) Flight Management Computer/Control Display Unit.**

The Electronic Attitude Director Indicator (EADI) is the primary feedback instrument for the FMS. Figure 5-1(B) depicts the EADI's components. Along the top of the EADI, the horizontal window displays performance information about the state of three FMS components: Autothrottle, Vertical Navigation, and Lateral Navigation. This display comprises the "Flight Mode Annunciator" (FMA) which indicates the current

modes of the aircraft. The Autothrottle mode (Thrust) indicates the engine performance requests. During climb, the Autothrottle mode is N1, which is the maximum normal thrust allowable for a climb. In the level cruise condition, the Autothrottle mode is FMC SPD, indicating that the FMC economy speed requirement is governing the engine thrust requirements. During descent, the Autothrottle mode is Idle, meaning the throttles are at idle and minimum fuel is consumed by the engines.

The second window, Lateral, displays Lateral Navigation performance and is beyond the scope of this paper.

The current vertical mode is displayed in the right-most EADI window. Several vertical modes are possible, including Vertical Navigation Speed (VNAV SPD), Vertical Navigation Speed Path (VNAV PATH), and Altitude Hold (ALT HLD). When Vertical Navigation (VNAV) is engaged after takeoff, the airplane's path and speed are controlled by the FMC. In a climb, the vertical Flight Mode Annunciator (FMA) indication is VNAV SPD because the Flight Management Computer (FMC) has scheduled a maximum engine climb thrust setting (N1) and the elevator pitch control will maintain the desired climb speed. When the aircraft transitions to level flight, the Flight Mode Annunciator (FMA) displays VNAV PATH because now the engine thrust controls aircraft speed while the aircraft maintains a level path. When Vertical Navigation (VNAV) is disengaged and holding a constant altitude, Altitude Hold (ALT HLD) is displayed on the vertical Flight Mode Annunciator (FMA) mode window.

The altitude tape on the left side of the Electronic Attitude Director Indicator (EADI) provides the pilot with the current aircraft altitude. By viewing the EADI, the pilot has knowledge of the vertical mode, the lateral mode, the autothrottle mode, and the aircraft altitude.

The Control Display Unit (CDU) keypad enables pilot inputs to the Flight Management Computer (FMC), the heart of the FMS system (Figure 5-1(C)). The Flight Management Computer (FMC) databases contain aircraft and engine performance characteristics for climb, cruise, and descent. Additional databases retain navigational aid locations, and airport information. The Flight Management Computer (FMC) uses the databases for computations guiding lateral navigation (LNAV) for the aircraft route, and altitude (VNAV) guidance altitude over points on the flight route. The focus of this work is the vertical navigation (VNAV) component of the FMS system

Earlier the illustrations depicted that the Flight Mode Annunciator (FMA) portion of the Electronic Attitude Indicator (EADI) displays the current mode of VNAV, LNAV, and the Autothrottle. The Control Display Unit (CDU) can also be used to monitor Flight Management System (FMS) functions. In addition to the keypad used for data entry, the Control Display Unit (CDU) has “pages” which contain information specific to a flight phase or aircraft performance requirement. The types of performance information available cover the take-off (TO), climb (CLB), cruise (CRZ), and descent (DES) regimes. Information about the aircraft route of flight, altitude constraints, and fuel

consumption are also available on various CDU pages. For the current purposes, the route (RTE), climb (CLB), cruise (CRZ), and descent (DES) “pages” contain the vertical navigation (VNAV) and lateral navigation (LNAV) performance information that allows a pilot to monitor the DNA function.

Geographical points defining a route of flight may have altitudes associated with them. The correlation between altitudes and route waypoints is present on the CDU RTE pages. When an altitude constraint at a specific route waypoint requires a constrained altitude crossing, the numbers associated with the waypoint are in a large font. (If a constraint is not present, the numbers associated with waypoint are in a small font, indicating an advisory crossing altitude). This simple distinction, large font or small font, is the CLB, CRZ, and DES alerting cue to the pilot that the aircraft must be at a specific altitude at a specific point.

5.2 Delete Next Altitude Function

The DNA function of the Vertical Navigation (VNAV) system is new to the Flight Management System. DNA has added the following four capabilities to the FMS (Smith Industries TDM 1888, 1991). (See Appendix 1 for Functions Table)

1. The DNA function can delete the next altitude constraint along a route of flight. The Mode Control Panel altitude must first be set above (for a climb) or below (for descent) the next altitude constraint and the “DNA” button pressed. An altitude constraint at the final route waypoint cannot be deleted using DNA.

2. A second DNA function enables increasing the cruise altitude while the aircraft is in the climb or cruise phase. Setting the Mode Control Panel altitude above the current cruise altitude and pressing DNA initiates the climb. To use this function in a climb, there must be no altitude constraints remaining on climb waypoints. The cruise altitude cannot be lowered using DNA.

3. The third DNA capability will resume a climb/descent after stopping a climb/descent to comply with a Mode Control Panel altitude constraint. To resume a climb/descent (while in climb, cruise, or descent) from an altitude hold constrained by the Mode Control Panel altitude, the Mode Control Panel altitude is set above/below, respectively, and the DNA button pressed.

4. The fourth and final DNA function initiates a step climb or a transition to the descent phase while in cruise. To initiate a step climb, the Mode Control Panel altitude is raised up to an allowable cruise altitude and the DNA button pressed. The FMC cruise altitude is reset automatically to the Mode Control Panel altitude. To initiate a transition to the descent phase, the Mode Control Panel altitude is set below the cruise altitude and the DNA button pressed.

5.3 Evaluation

In previous chapters the game of Russian Roulette and the VCR system were described in a Finite State Machine language first, then Task Goals and Task Specifications constructed. In this Chapter a complete flight scenario will be modeled

first in the Finite State Machine language (FSM). After the scenario description, Task Goals and Task Specifications will be examined.

The advantage of using a scenario is the ability to observe both the machine and the operator during task performance. If the system does not allow the operator to complete a task, or does not report that a task is complete when it in fact is, this examination method should enable us to view either discrepancy.

5.3.1 Scenario of FMS and DNA in Flight

The OCCAM ONE departure profile is a Standard Instrument Departure (SID) designed as a mandatory route for aircraft leaving congested airports. SIDs frequently contain altitude constraints at route waypoints to ensure vertical separation of aircraft. The RAZOR ONE arrival profile is a Standard Terminal Arrival Routing (STAR), and like the SID, is a required routing when assigned as part of an arrival procedure. A STAR may also contain mandatory altitude constraints at route waypoints to allow vertical traffic separation. The OCCAM ONE and RAZOR ONE profiles represent actual profiles that are in use in the U.S and do contain altitude constraints. For this systems purposes, the departure climb profile, the cruise conditions, and the arrival descent profile require the pilot to track FMC modes and intervene, by deleting the next altitude constraint when asked to do so by FAA Air Traffic Control (ATC). The scenario methodology allows description of all FMC modes during the profiles, and all possible pilot interactions with the DNA function.

Appendix 2 contains the Standard Departure Routing and the Standard Arrival Routing with constraints chosen to illustrate the Delete Next Altitude function. The departure and arrival depicted are similar to actual departures and arrivals in current use.

When a SID and/or STAR such as the OCCAM ONE and the RAZOR ONE are included as part of a flight plan, they are selected from the FMC database and activated during the preflight planning. SIDs and STARs are stored in the FMC databases as part of the airport information. As mentioned earlier, SID or STAR geographical waypoints have associated altitudes as crossing constraints that are loaded into the FMC along with the profile. After takeoff, with the FMS system engaged, the aircraft will automatically fly to the defined points crossing at the required altitudes with Mode Control Panel Altitude Set coordination.

Figure 5-2. SID and STAR Profile Overview.

Previewing the profile of this scenario including the SID, STAR, and cruise sections (Figure 5-2), three constraints mandating crossing route waypoints at exact altitudes

during the climb phase are evident: 4,000' at both CLSTR and MVRK, and JESTR at 8,000'. (The hashed lines above and below the altitude indicate a mandatory crossing at the stated altitude. A hashed line below a number would indicate the minimum altitude, while one above would indicate a maximum altitude). Top of Climb (TOC) at 31,000 begins the cruise phase of flight and includes a step climb to 35,000 before the Top of Descent (TOD) waypoint. The descent profile requires crossing ANJEE, SKUNK, and BOLDR at the altitude constraints of 16,000', 12,000', and 10,000', respectively.

Description and analysis of the system begins with a description of the notations used to model the scenario. Each circle represents a FMS configuration, and each line item within the circle indicates the status of a sub-component. In Figure 5-3(A), the top line item is the pitch mode; i.e., VNAV SPD, VNAV PATH, or ALT HLD. The second line item is the Autothrottle mode, i.e., N1, FMC SPD, or IDLE. The third line item is the upcoming waypoint, i.e., the waypoint the aircraft is flying toward, and its associated altitude constraint given in thousands of feet. For example, "Cluster" (abbreviated CLSTR) is the name of the first waypoint in this example with a 4,000' restriction (abbreviated 4). Therefore, the entire waypoint description is CLSTR 4. The fourth line item depicts the altitude (in thousands of feet) set in the Mode Control Panel (MCP4). Finally, the active phase of the FMC is given, i.e., climb (CLB), cruise (CRZ), or descent (DES). In Figure 5-3, climb (CLB) is the active state in all three depictions, as the aircraft is in the initial climb to its first altitude constraint.

Figure 5-3(A) Climb State Figure 5-3(B) Visible State Figure 5-3(C) Invisible State

Figure 5-3(B) describes the system configuration during a transition from climb (or descent) to level flight. In this configuration, the pitch mode annunciation changes to VNAV PATH. The autothrottle mode is FMC SPD, indicating that maximum engine power (N1) is no longer necessary to maintain the speed set in the FMC. The remainder of this configuration's modes and parameters remain the same as in Figure 5-3(A).

The configuration in Figure 5-3(C) uses a dotted circle to indicate that the VNAV sub-mode, ALT CAP, is actually not annunciated to the pilot. The pilot knows when the maneuver to capture the (level) altitude begins because the annunciation changes from VNAV SPD to VNAV PATH. However, the pilot does not know when the transition from VNAV PATH (ALT CAP) to VNAV PATH (ALT HOLD) takes place. It turns out, as will be seen later, that this piece of missing information is indeed important.

Figure 5-4 is a State Transition Diagram showing aircraft configurations during an automatic flight to the CLSTR waypoint. The initial configuration (indicated via the small arrow) is labeled as 1 below the state-circle. In this state, VNAV SPD is automatically controlling the aircraft's pitch. The engines are governed at their maximum N1 power to

maintain climb speed. The FMC is set for a 4,000' altitude constraint at the CLSTR, and the Mode Control Panel is set for 4,000'. (The setting of the Mode Control Panel altitude to match the FMC altitude is done by the pilot according to Standard Operating Procedures.) The FMC is engaged in a climb (CLB) mode.

Figure 5-4. Initial Climb to First Constraint at CLSTR.

Transition lines in the State Transition Diagrams are either solid or broken. Solid lines indicate manual events (e.g., pressing a button). Broken lines indicate automatic events (e.g., the transition from state 1 to state 2). The automatic transition (broken line) from VNAV SPD to VNAV PATH (ALT CAP) is indicated to the pilot on the Flight Mode Annunciator (FMA). This transition is triggered by the following condition becoming TRUE:

$$\Delta \text{Alt (plane, CLSTR)} = K_1$$

The condition says that when the difference (delta) between the aircraft altitude and the altitude selected for CLSTR is at some predetermined value (K_1), the condition becomes TRUE and the transition to state 2 fires. The transition from state 2 to state 3 is triggered by the following condition:

$$\Delta \text{Alt (plane, CLSTR)} = 0.$$

This condition says that when the difference (ΔAlt) between the aircraft altitude and the altitude selected for CLSTR equals zero, the transition to VNAV PATH takes place automatically and the aircraft is now at level flight.

Figure 5-5 expands on the initial climb to CLSTR by adding possible pilot interactions with the Delete Next Altitude (DNA) function.

Figure 5-5. DNA Event Deleting Next Altitude Constraint

The label “DNA/FMC (CURRENT = MVRK)” on the transition lines from states 2 and 3 to 5, and from 1 to 4, means that the pilot has pressed the “DNA” button, and as a consequence the altitude constraint at the upcoming waypoint (CLSTR) is deleted and the next waypoint (MVRK) altitude becomes the current altitude constraint. (There would be no altitude change in this particular example because CLSTR and MVRK have the same altitude constraint.) This is the first capability added to the FMC by the DNA function.

The statement,

$$\Delta \text{ Dist. (plane, MVRİK)} = 0$$

immediately following state 3 indicates that the distance between the aircraft position and the waypoint is zero, and therefore the constraints of the next waypoint now become the active constraints. Thus, when the aircraft (plane) is beyond the CLSTR waypoint ($\Delta \text{ Dist.} = 0$), MVRİK becomes the current waypoint for the FMC.

A primary requirement for a DNA event that will result in an altitude change is that the Mode Control Panel (MCP) altitude must be different from the current Flight Management Computer (FMC) altitude. The FMC stored "OCCAM ONE" departure profile limits the aircraft to 4,000' at the first two departure waypoints, CLSTR and MVRİK. JESTR, the third waypoint on the departure, has an 8,000' constraint. In order for the climb to continue, the higher 8,000' altitude must be set in the Mode Control Panel (MCP) altitude set window. A DNA selection in Figure 5-5 state 4, would produce no transition because a higher MCP altitude is not selected. The condition, no DNA effect, is indicated by the loop with the DNA label.

Figure 5-6 shows that the Mode Control Panel altitude set has been raised to 8,000' (state 2). Let us examine this in a circumstance where ATC clears the flight to 8,000' in state 1. Increasing the Mode Control Panel altitude (MCP 8) to 8,000' (state 2) changes the altitude configuration, but the aircraft continues on a path to CLSTR with a FMC target altitude of 4,000' (FMC CLSTR 4). The aircraft will level at 4,000'. What has happened is that a transition to an altitude configuration with the Mode Control

(FMC MVRİK 4). In state 6 and when at MVRİK ($\Delta \text{Dist. (plane, JESTR)} = 0$) an automatic climb is begun to 8,000' (state 7). The aircraft proceeds through states 8 toward JESTR (state 9). There is no intervention by the crew other than changing the Mode Control Panel altitude set during the initial climb.

Figure 5-7. CLSTR to JESTR with Mode Control Panel Increases and DNA Commands.

If, during the initial climb (state 1), 8,000' is selected on the Mode Control Panel altitude set (state 2), it is now possible to climb directly to 8,000,' bypassing the 4,000' constraints. The first "DNA" deletes 4,000' at CLSTR and the aircraft is now in state 10; "DNA" again deletes 4,000' at MVRİK and a transition to state 7 occurs, enabling the climb and finally leveling at 8,000' and proceeding to JESTR, state 9.

In the case of the OCCAM ONE departure, if the Mode Control Panel altitude set window is at 31,000' (Figure 5-8), three successive "DNA" selections during the initial climb would allow an unconstrained climb to 31,000.' Selection of "DNA" in state 1,

followed by state 2 “DNA”, and finally a state 3 “DNA” would enable the aircraft to climb unconstrained to 31,000’. In addition, 31,000’ could be selected early in the profile and automatic transitions performed from 4,000’, to 8,000’, to 31,000’ without active flight crew participation.

Figure 5-8. Unconstrained Climb to 31,000.

The final set of configurations will examine the cruise DNA profile.

Figure 5-9. Cruise Profile.

State 1 in Figure 5-9 represents the level cruise state following the profile climb. To increase cruise altitude, the Mode Control Panel (MCP) altitude set must be higher (e.g., MCP 35) than the current setting (state 2). A “DNA” at this point would automatically initiate a transition to the new altitude, 35,000’ in this example. With the DNA event, the new altitude is automatically entered into the FMC saving the pilot the need to enter this value. This capability is the second Flight Management Computer functional addition cited on the list of functions (page 64).

If a lower cruise altitude is needed, the DNA function cannot be used for that purpose.

The full description of the descent profile will not be detailed here. The entire flight profile describing all the climb, cruise, and descent flight phases is provided in Appendix 3.

5.4 DNA General Models

A pattern of system behavior emerges while viewing the climb profile model (OCCAM ONE) as a whole. The repetitious pattern is as follows: each Mode Control Panel (MCP) altitude setting increase exposes the possibility of a DNA, another Flight Management Computer Speed (FMC SPD) Climb (CLB), and another Vertical Navigation Altitude Capture (VNAV (ALT CAP)) while travelling from one waypoint to the another. This pattern allows us to collapse the Climb, Cruise, and Descent flight profiles into three generic models. Figure 5-10 is a General Climb Model containing archetypal

characteristics of all possible events in the Climb profile. Waypoints (and associated altitudes) have been replaced by 'active' and 'next,' terms that correspond to the current waypoint and the following waypoint, respectively.

5.4.1 General Climb Model

The General Climb Model in Figure 5-10 will now be examined. The initial climb (e.g., to CLSTR (state1)) without a Mode Control Panel (MCP) altitude set change is followed by a VNAV PATH (ALT CAP) (state 2), and VNAV PATH (ALT HLD) (state 3). No altitude change can occur in state 3 until a higher Mode Control Panel altitude is set. (Automatic transitions are dotted lines and manual transitions are solid lines, following the previous convention.)

The condition for transitioning from CLSTR to MVRIK prior to the CLSTR altitude capture is explained by the DNA event out of state 1 to state 7. An automatic transition to level flight occurs if the DNA event happens in a climb.

If the aircraft is in level flight prior to CLTSR, state 3, a DNA event (DNA/FMC (Active = Next)) will transition (solid line loop, state 3 to state 3) from level flight to CLSTR, to level flight inbound to MVRIK. If the aircraft reaches CLSTR without a pilot input (i.e., "DNA"), an automatic transition ($\Delta \text{Dis (plane, active)} = 0$ / FMC (Active = Next)) will result, taking the aircraft from level flight toward CLSTR, to level flight toward MVRIK (dotted line loop, state 3 to state 3).

Figure 5-10. General Climb Model.

A Mode Control Panel (MCP) altitude set increase to 8,000' during the climb to the active waypoint transitions us from state 1 to state 4. When the aircraft is level (state 3), a transition to state 6 takes place only when the Mode Control Panel (MCP) altitude is set for the next altitude. At state 6, an automatic transition (dotted line out of state 6) occurs back to state 1 upon reaching the downstream waypoint. While at both states 5 and 6 (indicated by both states enclosed in a single box), if DNA (DNA/FMC (active = next)) is selected before passing the constraining waypoint and the next higher altitude is

set, another climb sequence will occur taking us back to state 1. During a climb when a higher Mode Control Panel (MCP) value is set (state 4) and “DNA” selected, the next altitude becomes active and the aircraft transitions back to state 1. This climb phase pattern is broken when the scenario departs the climb phase (state 2) for the cruise phase.

5.4.2 General Cruise Model

The transitions from the General Climb Model’s state 2 ($\Delta \text{ALT (plane, cruise)} = 0$ / $\text{FMC} = \text{CRZ}$) to the “top of climb” cruise altitudes are similar to intermediate climb transitions discussed previously. The Cruise Model configurations have changed little from the Climb Model. The FMC and MCP altitudes now include Cruise, as well as Active and Next. There are two additional cruise DNA functions in the General Cruise Model (Figure 5-11) that need explanation. The first is that “DNA” allows cruise altitude increases (state 1 to state 2, and back to state 1), but not cruise altitude decreases (state 1 to state 3). Manual pilot input to the FMC and MCP may command a lower cruise altitude.

Further, “DNA” commences a descent before a FMC computed Top of Descent (TOD) point (state 3), DNA/FMC (Cruise = Des)) to intercept a profile descent. In both instances, the MCP altitude is set to the desired altitude before the “DNA” event (at which time the new altitude becomes the FMC current altitude).

Note, also, the transition from state 5 to state 6 that occurs when the FMC altitude has transitioned to a lower altitude while the Mode Control Panel (MCP) altitude remains at cruise. Because of the lack of guidance, the FMS will transition out of VNAV PATH to altitude hold and the aircraft will not begin a descent. Pilot inputs (lower MCP altitude and VNAV selection) are needed to initiate further descent from state 6 to states 7 and 8. A nominal profile descent begins with an automatic transition from state 4 to state 8. An early transition using the DNA function transitions from state 3 to state 9.

5.4.3 General Descent Model

Figure 5-12. General Descent Model.

The General Descent Model has an emergent pattern that is similar to the Climb Model, but its inverse (Figure 5-12). A distinctive feature here is that at the end of the descent, DNA cannot remove the last altitude constraint and therefore a minimum descent protection is provided.

5.5 FMC Examination with Two Task Goals and Task Specifications

The VNAV and DNA General Climb Model will now be examined with two Task Goals that represent routine user tasks. This will allow examination of a pilot's interaction with the system. Task Specifications associated with each Task Goal will pose questions about acceptable system performance from the human-machine perspective. The Task Specification "unambiguously" means that there should never be a case in which the user tries to drive the system to state A, but due to lack of information instead lands in state B. The interaction between the user and the machine is analyzed to determine if the operator can accomplish the task, given the specifications.

5.5.1 Task Goal One: Mode Control Panel Altitude Set Changed While in VNAV(ALT CAP)

The first Task Goal asks that the Mode Control Panel (MCP) altitude set knob to be turned while the aircraft is in the VNAV (ALT CAP) mode. The Task Specification asks: "Does the pilot know the outcome?" The point at which the FMC transitions from a climb or descent to level flight is annunciated by VNAV PATH. The fact has been established however, that a sub-mode VNAV (ALT CAP) exists before the aircraft reaches level flight.

Figure 5-13 is a simplified General Climb Model with the two VNAV (ALT CAP) states in a box.

Figure 5-13. VNAV (ALT CAP)

The observation was made earlier that the transition from VNAV (ALT CAP) to VNAV (ALT HOLD) is not displayed to the pilot. This VNAV ambiguity can be examined with the current methodology to analytically verify if the behavior is acceptable or not.

In the VNAV (ALT CAP) state, if the altitude window selection on the Mode Control Panel (MCP) is changed, one of three out comes may result (Figure 5-14).

1. If a Mode Control Panel altitude set change is greater than 200' to 300' and the new target altitude is ahead of the aircraft (higher in a climb, lower in a descent), the Mode Control Panel will stop the climb or descent and capture the new altitude. The autopilot will revert to MCP Speed and MCP Altitude Hold at the new altitude ignoring the FMC altitude.

2. If a new altitude setting is behind the aircraft (lower in a climb, higher in a descent), the FMC reverts to the Control Wheel Steering (CWS) pitch mode. In the CWS mode, FMC inputs cease and current flight control inputs hold aircraft pitch until a different input is made at the pilots' control wheels. Control Wheel Steering provides no protection against departing the FMC profile and a normal flight regime.

3. The last FMS behavior is the expected and desired action. When the conditions for the transitions are met, the FMS captures and holds the Mode Control Panel altitude set in the window.

A fourth possible outcome of an Altitude Set change ahead during CLIMB VNAV (ALT CAP) (not depicted in Figure 5-14) is the aircraft continuing the climb to the new altitude while remaining in VNAV. The point is that the pilot may not know which of the four behaviors will result from a MCP Altitude Set input during VNAV (ALT CAP).

The following three FMS behaviors (Figure 5-14) can occur with pilot input to the Mode Control Panel (MCP) altitude set in VNAV (ALT CAP).

Figure 5-14. Mode Control Panel Altitude Set Changes in VNAV(ALT CAP) Climb.

5.5.1.1 Analysis

The non-determinism resulting from the Mode Control Panel (MCP) altitude change in the VNAV (ALT CAP) mode is somewhat similar to the behavior of the VCR mode ambiguity when the user could not discriminate between three states during an input that attempted to display the “on screen” menu. An input in the ambiguous VCR system could either display the desired information or disconnect the VCR from the TV. Previous discussions established that when a user’s input can create more than one machine behavior, the user will not know which machine behavior to expect.

In the case of the FMS, two possible system behaviors exist while the aircraft is in VNAV PATH. These states are VNAV PATH (ALT CAP) and VNAV PATH (ALT HLD). But the VNAV PATH (ALT CAP) state is under-specified because the Flight Mode Annunciator (FMA) interface does not clearly specify the conditions differentiating the two states. The pilot cannot discern between the two.

Figure 5-15. Ambiguity with Multiple Outcomes on the Same Input.

The Task Specification for this Task Goal is that there should never be a case in which the user wants VNAV (ALT HLD), but due to lack of information instead lands in another unexpected state. This Task Specification is unsatisfactory because multiple ambiguous outcomes are possible on the same VNAV configuration input (Figure 5-15).

Figure 5-16(A) is an abstract representation of the non-determinism identified in the VNAV PATH configuration, in which more than one outcome is possible from the same input (Mode Control Panel altitude set change on α). Figure 5-16(B) is the same type of non-determinism defined in Figure 2-2(A).

Figure 5-16(A). VNAV Non-Determinism

Figure 5-16(B). Classic Non-Determinism.

5.5.2 Task Goal Two: Mode Control Panel Altitude Set Not Advanced to Next Constraint.

Consider the following sequence of events: the aircraft is level and flying toward the active waypoint, and the pilot forgets to reset the Mode Control Panel (MCP) altitude or is not issued a higher altitude in preparation for the next climb constraint. In

this case, when the aircraft is beyond the active waypoint by an unspecified distance (Δ Dist = 0), the aircraft will not be able to climb to the next altitude.

Figure 5-17. Unable to Achieve Next Waypoint Altitude Constraint.

Figure 5-17 shows the sequence of states that leads to a lower Mode Control Panel (MCP) altitude setting than necessary to reach the next waypoint at the constraining altitude.

The scenario description begins in state 1 with the altitude selection in the Mode Control Panel on the active constraint. In this example, the Mode Control Panel altitude

is not set higher than the current altitude. The system automatically progresses beyond state 3 and transitions to state 4 with the following trigger:

$$\Delta \text{ Dist (Active, Old)} = 0$$

When the aircraft is beyond state 3, the FMC does not initially acknowledge that a Mode Control Panel altitude setting change is needed because of a higher constraint at the next waypoint. Previous description established that if a higher altitude is required and selected prior to the transition to state 4 (e.g., state 5 and state 6), the aircraft will automatically, or following a “DNA,” climb to that altitude.

The FMC database contains aircraft performance parameters that enable the computer to constantly calculate the aircraft’s climb, cruise, and descent capabilities based on the current inputs (e.g., temperature, altitude, and weight). When the FMC calculates that the aircraft can no longer arrive at the next waypoint and meet the altitude constraint, it will present a CDU message stating, “UNABLE NEXT ALTITUDE.” This is the transition from state 4 to state 7 in Figure 5-17 (ALT HLD, FMC SPD, FMC ACTIVE, MCP OLD).

After state 4, the transition out of VNAV to state 7 is based on FMC performance K_3 parameters that are not available to the pilot. This transition trigger is:

$$\Delta \text{ Dist (plane, Active)} = K_3$$

This means that if the airplane distance from the next constraint is K_3 and given the FMC calculation from the current constraint to the next, then “UNABLE NEXT ALTITUDE” is issued as the aircraft enters state 7, (i.e., ALT HLD, FMC SPD).

5.5.2.1 Task Goal and Task Specification

The final Task Goal for this example is about a recovery from a common operator error, or more specifically, a *slip* and inadvertent transition off the nominal path. The pilot (or controller) erred and now wants to recover and return to the profile without violating an ATC restriction. The Task Goal, therefore, is to return to VNAV and the profile, following this error of omission. The Task Specification asks the following:

Can the pilot identify what mode the aircraft is in? Can the pilot, unambiguously, determine what the next mode of the system will be? Specifically, can the pilot tell when it will no longer be possible to recover and still comply with all ATC restrictions?

Figure 5-18 illustrates the possible sequences of action.

The aircraft continues in level flight (VNAV PATH (ALT HLD)) until an automatic transition is generated at the latest point from which the FMS could commence a climb, and meet the requirements of the next constraint (state 4), based on performance parameters not visible to the pilot. Note that pressing “DNA” at this point will not solve the problem because the Mode Control Panel altitude set has not been changed from the current altitude. At the transition point, a CDU text message notifies the pilot that the next waypoint constraint is not achievable by displaying “ UNABLE NEXT

ALTITUDE.” The VNAV button on the Mode Control Panel remains illuminated throughout this sequence, even after the FMA indicates a transition from VNAV to ALTITUDE HOLD.

Figure 5-18. Unable Next Altitude.

In order to transition from state 5 to state 1 (Figure 5-18), a higher altitude must be selected on the Mode Control Panel (MCP). This action extinguishes the VNAV light, transitions to the VNAV (ALT HLD) configuration, and arms the system for a climb. Selecting VNAV will allow the FMS to command a climb. However, the next altitude constraint may be unachievable.

5.5.2.2 Analysis

Two problems are revealed with this Task Goal. The first problem is that when the transition from VNAV PATH (ALT CAP) to VNAV PATH (ALT HLD) occurs, the Mode Control Panel interface is in contradiction to the Flight Mode Annunciator (FMA). Figure 5-19 depicts the type of non-determinism evident here. The fact has been established that VNAV will not function at this point because ALTITUDE HOLD is the active and FMA display mode even while the VNAV light is also on.

The second problem is that when the transition from VNAV (ALT HLD) to ALTITUDE HOLD (ALT HLD) occurs, the distance and performance parameters ($\Delta \text{Dist (plane, next)} = K_2$) creating the transition are not available to the pilot. As a consequence, when the transition does occur and the pilot is notified, it is too late to climb and meet the next constraint. This is similar to the Russian Roulette example earlier and to the VCR example in which not all of the states are visible. In this case, as far as the user is concerned, the system will continue in its current mode until told otherwise. But too late the user discovers that this is not the case. Figure 5-19 displays the type of non-determinism present with this Task Specification.

Figure 5-19. Mode Ambiguity

5.6 Summary

In this chapter the capabilities of the proposed methodology were explored further by examining the Flight Management Systems Delete Next Altitude (DNA) function installed on a new aircraft.

First, a realistic flight scenario was constructed which exercises the VNAV mode and the Delete Next Altitude function. A State Transition Diagram model of the FMS VNAV and all possible DNA states was constructed to represent flight profiles. The aircraft configurations were superimposed on the flight segments, and then clearly defined so that a test pilot operating the system or a design/evaluation team could agree on the system configuration. Detailed Finite State Machine descriptions for the FMS functions were examined and developed to accurately and realistically show interactions between the pilot's tasks and the automated flight guidance system.

Repeated and consistent patterns in the full Finite State Machine representation allowed us to construct generic models for the climb, cruise, and descent profiles. The generic models allowed us to examine the operator and system more clearly, because the large models' size and complexity masked some otherwise obvious system behaviors.

The Delete Next Altitude function is new feature added to the FMS. An examination was proposed which would allow a methodological evaluation of this function. The Task Goals were selected which would exercise the DNA system in all flight regimes. So while asking "what-if" questions about operator actions and observing the resulting behaviors, it was discovered that potentially hazardous anomalies existed.

To test the Task Goals, multiple simulator sessions were conducted, during which these anomalies were examined while simultaneously attempting to understand the operator's role and the system response.

The Task Specifications assigned to each specific Task Goals were the selected method for testing the human and machine system to ensure that the Task Goal was met. For example, a Task Specification might ask, "If the Task Goal is completed, could the operator then tell the task was complete?" Task Specifications can be asked in many different ways and answered with dichotomous terms such as "safe" or "unsafe." In many instances the user/machine interface failure revolved around non-deterministic machine behavior.

The airliner FMS is a concrete example of a dynamic, complex system in which the information provided by the interface is critical for human-machine interaction. The observed systems performances have shown that the FMS does not always represent its current state to the pilot, and therefore does not provide information about the next state. Specifically, both the VNAV PATH (ALT CAP) state and the VNAV PATH (ALT HLD) state following an 'UNABLE NEXT ALTITUDE' alert are not displayed, or are inaccurately represented. If the pilot does not possess knowledge about the current state and appropriate transitions out of that state, that pilot's action may create either an error of omission or an error of commission. An error of omission would occur, for example, if a DNA event were attempted but not completed because the operator did not verify the

action taken produced the desired results. An error of commission would result if multiple deletions occurred while attempting a DNA event, but there were no visible FMC results (simulator example cited earlier).

CHAPTER SIX: DISCUSSION AND CONCLUSIONS

6.1 Objectives and Approach

The objectives of this thesis were as follows: First, to describe a formal methodology for evaluating human-machine interaction. The proposed approach is the synthesis of a machine description and a user's task description. The second stated objective was to apply the proposed methodology to a variety of simple and complex human-machine systems. To meet this objective, three existing systems were examined and user interaction with specific system functions evaluated.

Finite State Machine languages were selected to describe the machine's behavior. To describe the user's goal, the user's task objective in the system was specified. No attempt was made to describe the user's perceptual or cognitive processes while achieving a task goal.

However, it is important in assessing the Task Goal to ask the question, "Can the user achieve this goal reliably?" This question encompasses what was defined as Task Specification. If the Task Goal is unequivocally completed, there is no ambiguity. However, if the task completion is in doubt based on the Task Specification, i.e., if it is

ambiguous, then the user will experience confusion and frustration. An error may take place.

An important element of the proposed methodology is the concept of building real-world scenarios to evaluate user interaction in the context of necessary sequences of actions to complete tasks. The use of scenarios allows systematic analysis. This capability provides a roadmap with which to examine both the system and human components in detail.

6.2 Method

In this thesis, three systems of increasing complexity were evaluated. The “game” of Russian Roulette provided a platform for explaining the methodology: In this first instance the methodology combined two elements: (1) a Finite-State-Machine description of a pistol, and (2) the user’s Task Goals. The user-system interaction analysis found unequivocally that the user could not tell whether the system was transitioning to a safe (no-bullet) state or an unsafe (discharge-bullet) state, which, of course, is the entire objective of the “game.” In this description the essence of non-determinism was captured and then how it related to human-machine interaction described.

The second model described a TV and VCR system in which several Task Goals were identified involving a user displaying certain VCR functions on the TV screen. This system was more hierarchical and complex than the previous Russian Roulette example,

because programming the TV/VCR requires navigation between layered menus. This hierarchy and complexity required that the model be built in a richer language than the State Transition Diagram. With the choice of the Statecharts language, the VCR's hierarchical menu relationships were elaborated and the complex navigation commands of the menu easily depicted. After the complete model was constructed, selection of Task Goals completed, and Task Specifications delineated, ambiguity and resulting difficulty for the user were evident. The user was unable to unequivocally determine the VCR state from the information presented on the displays.

The third and final system model was a sophisticated aircraft navigation system and its new function: Delete Next Altitude (DNA).

Analysis of this system began with construction of a full-mission flight scenario from departure through the cruise flight, and subsequently a descent on a planned profile. The scenario was designed to exercise the Delete Next Altitude function and related system components. In order to accomplish our goal, the scenario contained exacting altitude constraints along an actual Federal Aviation Administration route of flight. The complexity of this scenario allowed us to examine many of the subtleties of this system component and the user interaction.

After constructing an extensive State Transition Diagram model around the flight scenario, a pattern emerged which allowed simplification of the scenarios into more general models, ones that were reduced and simpler. Then several Task Goals were

evaluated with their Task Specifications and the system further inspected to identify potential problems. Ambiguity and non-determinism were evident. The user was unable to unequivocally determine the next state of the system given the information presented on the cockpit displays.

What this methodology has demonstrated is that the synthesis of a machine description with a task description notation enables a system and human operator to be viewed simultaneously. Instances in which the system ambiguously completes a user's request can be identified as design deficiencies. Error becomes apparent when the system enters an unexpected state while the users expects it to be in another.

One advantage of the proposed methodology is that it allows for easier communication between experts from differing fields. It is an easily understood language that allows members of the design team (i.e., human factors specialists, designers, and software engineers) to examine the machine behavior and user interaction during the design phase. In addition, the language can be used to track and document design problems identified after the system is fielded.

Many of the ambiguities in current-day human-automation systems boil down to the notion of non-determinism—a very basic notion in computer-science and control theory.

Figure 6-1. Non-Determinism

In Russian Roulette, two basic system states were identified; chamber empty or chamber with a bullet. The user cannot discriminate between the two states. Pulling the trigger, call it event α , resulted in either a transition to a bullet discharge (D in Figure 6-1), or no discharge (back to E). From the user's perspective, the system is non-deterministic. Examination of the Russian Roulette "game" with the proposed methodology clearly identified this ambiguity, which is what the "game" is all about.

In the TV and VCR analysis the user was required to make two selections in order to display the "on screen" menu, but was not told the current state after either input. One ambiguity becomes apparent when the user selects "menu" with no apparent change in VCR or VCR output. The non-determinism seen in Figure 6-1 is again present. In fact, an internal change has occurred. Likewise, a "tv-vcr" selection outputs the VCR's menu to the TV, but unless another selection is also made, no VCR output will be seen. The problem for the user is the desired state, "on-screen" programming menu, may not be reached because the combination of states necessary to achieve the final state are not unambiguously shown to the user. Both "tv-vcr" and "menu" selection must be made in order to display the menu, but the path to that state is not displayed to the user.

Through use of the proposed methodology, the system description and task requirements clearly revealed the dilemma for the user.

The aircraft Delete Next Altitude (DNA) Function is a Flight Management Computer (FMC) function that gives pilots the option of reducing the keystrokes necessary to remove an altitude constraint when compared to older versions of the Flight Management Computer.

The Flight Management Computer (FMC) scenario model describes a profile climb, a cruise, and a descent. When analyzing this system it is immediately clear that two vertical navigation (VNAV) sub-modes are indistinguishable. VNAV PATH(ALT HOLD) and VNAV PATH (ALT CAP) are displayed on the pilot's instruments as one mode (VNAV PATH). The pilot cannot distinguish between these two submodes. Deleting the Next Altitude while in VNAV PATH(ALT HOLD) results in deletion of an altitude (D) constraint, while doing the same in VNAV PATH (ALT CAP) results in no action whatsoever (E).

Another observation was made while examining this system. If a pilot makes a Flight Management Computer input using the Mode Control Panel (MCP) Altitude Set selector knob, the aircraft autopilot may transition into a mode in which there is no altitude guidance. The proposed methodology enabled the examination of this anomaly. Again, the ambiguity of the type in Figure 6.1 was present, but with more than one ambiguous mode. This finding is potentially significant to safety of flight.

The proposed methodology allows examination of simple systems (e.g., Russian Roulette) as well as complex, dynamic systems (e.g., an aircraft FMC). The significance of this is that a single methodological application is possible across a variety of system complexities. This application clearly shows that non-determinism is a consistent thread that generates user confusion.

TAFEI (Task Analysis for Error Identification), an early attempt at error identification methodology, describes tasks and machine state transitions in terms of legal and illegal user inputs. In the TAFEI methodology, strict adherence to task flows is required in order to reach the desired machine state, and only certain actions are allowable. If an allowable action results in the transition to a desired state, it is legal. If the same action cannot be taken in a given state because the machine will not respond, that action is considered illegal. The limitation of this process, unlike our proposed methodology, is that it considers only certain tasks, and user actions outside of these tasks are not examined.

There is much miscommunication in the way such systems are currently evaluated: Engineers employ a descriptive methodology unreadable by the person responsible for describing a machine's workings to the non-technical user. Likewise, the textual descriptions developed for user manuals are of little value to an engineer interested in hardware or software design. A significant advantage of the proposed methodology is

that it is cross-disciplinary and is easily understood by members of a design team from a variety of disciplines.

In the process of describing complex systems, scenario developments are possible and in some instances desirable, so that all possible user interactions can be examined in the context of a specific scenario. This is the approach selected for the Delete Next Altitude examination. However, scenario development is not always necessary, as the Russian Roulette and VCR programming examples have shown. If the system is not complex, or has only a few modes, an evaluator does not need to construct a scenario to examine user interactions.

6.3 Limitations

The methodology described in this thesis does not include the user's cognition or perception. Instead, the operator is assumed to be an automaton performing specific tasks that are designed to examine a system and an operator's actions. The perfect operator assumption is a constraint caused by our current level of methodology development. But there is an assumption of what is known and knowable by this operator.

A potential limitation of the proposed methodology, one not yet sufficiently explored, is a design team's inability to specify all of a system's possible states. A product requirement document may completely describe the desired system functions. Product specifications allow description of the machine using our proposed

methodology. However, until a product prototype is operational and tested, it may be difficult to anticipate a user's attempt to create unexpected or undesirable machine states.

6.4 Benefits

The proposed methodology provides a bridge between machine models and human factors' task analysis tools whose goal is to evaluate an interface. Engineering and Human Factors paradigms are different: the first asks, "how does it work," and the second asks, "how do humans operate it?"

The proposed methodology is intended to evaluate human-automation systems. Special attention has been focused on machine modes, states, and transitions. Modes are a fact of life given today's technology, and resolving ambiguity associated with modes is critical, but elusive. Yet a systematic evaluation of a machine's modes, and an examination of the machine's functionality in its modes using the Task Goal and Task Specification methodology, will eventually reveal machine errors if they are present. As has been shown in this work, when error, i.e., machine ambiguity, is present, its occurrence can be isolated to specific machine functions. From this discovery, resolution of the error involves clarification of the machine's functions and redesign of the user's interface to present the machine's current state.

A final note about *slips* and *mistakes*. The proposed methodology clarifies the approach to error assignment. If the human correctly processes incomplete or unclear

information from a machine and acts accordingly, a resultant error will be the outcome. The machine must support feedback so the user knows about the error. Such an error should not be assigned to the human, but to the design.

6.5 Future Research

To date, this methodology has been applied only to existing systems and systems in their final stage of design. Further research should document a product design process using the proposed methodology.

Methodology validation and refinement could begin with a design group consisting of engineers, human factors practitioners, and domain experts. A longitudinal process would allow a continuing group of experts to view a design concept with all possible user interactions with this system.

Future work may include a facility that assumes a cognitive paradigm. An assumption could be made that the operator has learned a set of rules that govern his/her actions while interacting with the system. The combined cognitive model, including the user's internalized rules and the machine model's hardware rules, could then be examined during user interaction. The comparison of a user's mental model, i.e., conceptual working model, with the actual interaction model would test the user's knowledge and beliefs about the target system. From this comparison, areas of disagreement between the two could be examined in an experimental situation to see if the areas of disagreement are where most errors occur. Does the user need additional information to accomplish a

task? Does the user's mental model include the appropriate modes and transitions between modes? These are the types of questions that could be asked using models of human-machine interaction.

Finally, several software packages exist that are useful for specifying machine behavior. Task analysis software packages also exist to document the flow of human activity while performing tasks. A synergistic software application to simultaneously build a system, and describe a user's interaction, would be a useful tool for automating the proposed methodology

This ultimate goal of this work is the safety and well-being of humans working in a technological environment. The overall concern is with supporting a worker's mental satisfaction, reducing frustration, and enhancing productivity. As this work has proven, in at least three examples ranging from a mundane consumer product to a very complex avionics system, this methodology supports all of these ends.

References

- Baber, C. and Stanton, N.A. (1994). Task analysis for error identification: a methodology for designing error-tolerant consumer products. Ergonomics, 37 (11), 1923-1941.
- Billings, C. E. (1996). Human-centered aviation automation: Principles and guidelines. (NASA Technical Memorandum 110381). Moffett Field, CA: NASA Ames Research Center.
- Boeing Commercial Airplane Group (1998). Statistical Summary of Commercial Jet Airplane Accidents, Boeing Commercial Airplane Group, Seattle, WA.
- Chi, H.U. (1985). Formal Specification of user interfaces: a comparison and evaluation of four axiomatic approaches. IEEE Transactions on Software Engineering, 11(8), 671-685.
- Degani, A., Heymann, M., Meyer, G., Shafto, M. (2000). Some Formal Aspects of Human-Automation Interaction. (NASA Technical Memorandum TBA), Moffett Field, CA: NASA Ames Research Center.
- Degani, A. (1996). Modeling human-machine system: On modes, error, and patterns of interaction. Unpublished Doctoral Dissertation, Georgia Institute of Technology, Atlanta.

Degani, A., Shafto, M., & Kirlik, A. (1996b). Mode Usage in Automated Cockpits: Some Initial Observations. Proceedings of International Federation of Automatic Control: Man-Machine Systems (IFAC-MMS), Boston, MA.

Degani, A. and Andre, A. (1996). Do You Know What Mode You're In? An Analysis of Mode Error in Everyday Things. Proceedings of the 2nd Conference on automation Technology and Human Performance. Daytona Beach, FL., University of Central Florida.

Dorner, D. (1997). The Logic of Failure. Reading, MA: Addison-Wesley.

Emerson Corporation. 4 Head VideoCassette Recorder with On-Screen Programming and Remote Control. Owner's Manual, VCR 4000.

Hopcroft, J.E. and Ulman, I. (1979). Introduction to Automata Theory, languages, and computation. Reading, MA: Addison-Wesley.

Harel, D. (1987). Statecharts: A visual formalism for complex systems. Science of Computer Programming, 8, 231-274.

Kieras, D. (1994). A Guide to GOMS Task Analysis. University of Michigan

Kieras, D. (1993). Using the Keystroke-Level Method of Estimated Execution Times. University of Michigan

Kirwan, B. and Ainsworth, L.K. (1992). A Guide to Task Analysis. Washington, DC: Taylor and Francis.

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.

Norman, D. (1988). The Psychology of Everyday Things. New York: Harper and Row.

Parks, K.S. (1997) Human Error. In Handbook of Human Factors and Ergonomics. (Ed. G. Salvendy). New York: John Wiley.

Peterson, D.L. (1977). Petri Nets. Computing Surveys, 9, 223-252.

Ryder, J.M. and Redding, R.E., (1993). Integrating Cognitive Task Analysis into Instructional Systems Development. Education Technology Research & Development, 41 (2), 75-96.

Reason, J. (1990). Human Error. New York Cambridge University Press.

Sarter, N.B. and Woods, D.D. (1995). How in the World Did We Ever Get into That Mode? Error and Awareness in Supervisory Control. Human Factors, 37 (1), 5-19.

Salvendy, G. (1997). Handbook of Human Factors and Ergonomics (2nd ed.). New York: John Wiley & Sons, INC.

Smith Industries (1991). TDM 1888. Operational Characteristics, Update 5 & 6 Series. Flight Management Computer System. Smith Industries, Aerospace & Defense Systems, INC., Grand Rapids, MI.

Stammers, R.B. (1995). Factors limiting the development of task analysis. Ergonomics, 38 (2), 588-594.

Turing, A.M. (1936). On Computable Numbers, With An Application to the Entscheidungsproblem. Proceedings of the London Mathematical Society, 42, 230-365.

Webster's (1994). Encyclopedic Unabridged Dictionary of the English Language. New York, NY: Random House.

Wickens, C.D. (1992). Engineering Psychology and Human Performance (2nd ed.). New York: Harper Collins.

Appendix

- 1. Altitude Intervene (Delete Next Altitude) Function Chart.**
- 2. Standard Departure and Standard Arrival Routing.**
- 3. Delete Next Altitude State Transition Diagram.**

TABLE 4-6
ALTITUDE INTERVENTION - VNAV ALT OPTION ENABLED

DESIRED FUNCTION	FMC PHASE	ANNUNCIATION		OTHER INDICATIONS	ACTION REQUIRED
		A/T	DFCS		
CLB now (Delete FMC Altitude Constraints)	CLB	FMC SPD	VNAV PTH	ACQ/HLD of FMC Altitude Constraints	Set MCP ALT > Aircraft Push <Alt>
CLB now (Resume)	CLB	FMC SPD	VNAV ALT	Not ACQ/HLD of FMC Altitude Constraints	Set MCP ALT > Aircraft Push <Alt>
CLB via FLT PLN	CLB	FMC SPD	VNAV PTH	ACQ/HLD of FMC Altitude Constraints	Set MCP ALT > Aircraft
Delete FMC Altitude Constraints	CLB	N ₁	VNAV SPD	NA	Set MCP ALT > Constraints Push <Alt>
Increase CRZ ALT	CLB	N ₁	VNAV SPD	No FMC Climb Altitude Constraints	Set MCP ALT > CRZ ALT Push <Alt>
CLB now (Increase FMC CRZ ALT)	CRZ	FMC SPD	VNAV PTH	ACQ/HLD of FMC CRZ ALT	Set MCP ALT > CRZ ALT Push <Alt>
CLB now (Resume)	CRZ CLB	FMC SPD	VNAV ALT	ACQ/HLD of MCP ALT	Set MCP ALT > Aircraft Push <Alt>
Increase CRZ ALT	CRZ CLB	N ₁	VNAV SPD	NA	Set MCP ALT > CRZ ALT Push <Alt>
DES now (Change to ACT DES)	CRZ	FMC SPD	VNAV PTH	ACQ/HLD of FMC CRZ ALT	Set MCP ALT < CRZ ALT Push <Alt>
DES now (Resume CRZ DES)	CRZ DES	FMC SPD	VNAV ALT	ACQ/HLD of MCP ALT	Set MCP ALT < Aircraft Push <Alt>
DES now (Change to ACT DES)	CRZ DES	FMC SPD	VNAV PTH	-1000 ft/min vertical speed	Set MCP ALT < CRZ ALT Push <Alt>
DES via FLT PLN	CRZ	FMC SPD	VNAV PTH	ACQ/HLD of FMC CRZ ALT	Set MCP ALT < CRZ ALT
DES now (Delete FMC Altitude Constraints)	DES	RETARD or IDLE or FMC SPD	VNAV PTH	ACQ/HLD of FMC Altitude Constraints	Set MCP ALT < Aircraft Push <Alt>
DES now (Resume)	DES	FMC SPD	VNAV ALT	Not ACQ/HLD of FMC Altitude Constraints	Set MCP ALT < Aircraft Push <Alt>
DES via FLT PLN	DES	RETARD or IDLE or FMC SPD	VNAV PTH	ACQ/HLD of FMC Altitude Constraints	Set MCP ALT < Aircraft
DES via FLT PLN (Above Path)	DES	FMC SPD	VNAV ALT	Above DES Path	Set MCP ALT < Aircraft Push <Alt>
DES via FLT PLN (Below Path) ⁽¹⁾	DES	FMC SPD	VNAV ALT	Below DES Path	Set MCP ALT < Aircraft
Delete FMC Altitude Constraints	DES	RETARD or IDLE or FM C SPD	VNAV SPD or VNAV PTH	Not ACQ/HLD of FMC Altitude Constraints or MCP ALT	Set MCP ALT < Constraints Push <Alt>

Appendix 1 . Altitude Intervene (Delete Next Altitude) Functions Chart.

OCCAM ONE DEPARTURE (LNSAY1.LNSAY) (PILOT NAV) (RADAR REQUIRED)

Aircraft filled at or below 13000' shall be cleared via the Seal Beach or Santa Catalina transition. Aircraft filled at or above 14000' shall be cleared via the Falco transition.

This SID requires minimum climb gradients of:
 Rev 9, 970' per NM to 1400'.
 Rev 27, 260' per NM to 400'.
 Rev 9 departure requires 300' ceiling and 1 mile visibility.

One speed-Kts	75	100	140	200	250	300
900' per NM	325	433	660	867	1063	1300
970' per NM	713	850	1425	1800	2375	2825

TAKE-OFF
 Rev 9: Climb via runway heading until leaving 4000', then turn LEFT to a 275° heading to intercept and proceed via MZB R-314. Thence

Direct distance from San Diego Intl-Lindbergh (Rev 9) to:
 Marine Int 23 NM
 Direct distance from San Diego Intl-Lindbergh (Rev 27) to:
 Clair Int 11 NM

MONTEREY ONE ARRIVAL (MRY1-MRY) (TURBOJET AIRCRAFT ONLY) (RADAR & ONE REQUIRED) (RWYS 26L/R)

Direct distance from Marine Int to:
 San Francisco Intl Apr 14 NMA

Expect radar vector to final approach course
 Expect radar vector to final approach course
 Expect clearance to cross at 10000'.
 Expect clearance to cross at 12000'.
 Expect clearance to cross at 10000'.

FOR NASA RESEARCH PURPOSES ONLY -
 NOT TO BE USED FOR NAVIGATION

Appendix 2. Standard Departure Routing (STAR) and Standard Arrival Routing
 Charts of Delete Next Altitude Scenario.

JSTR
8,000

CRZ - Top of Climb (TOC)
31,000

CRZ - Review

CRZ - Revised Cruise Altitude

Appendix 3. Delete Next Altitude State Transition diagram.