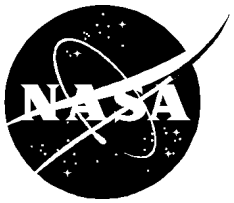
NASA/TM-2000-209600



Some Formal Aspects of Human-Automation Interaction

Asaf Degani, Michael Heymann, George Meyer, Michael Shafto
Ames Research Center, Moffett Field, California

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

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

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

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

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

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

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

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

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

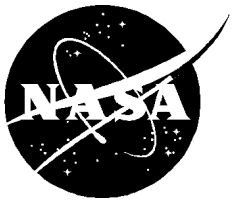
- Write to:

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

Acknowledgements

This work was conducted as part of NASA's base research and technology effort, human-automation theory sub-element. The authors were supported by Grant NCC 2-798 from the NASA Ames Research Center to the San Jose State University. Portions of the research appeared in a paper, entitled "Pilot-Autopilot Interaction: A Formal Perspective", presented at the 10th International Aviation Psychology Symposium, held at Columbus, Ohio in 1999. We gratefully acknowledge the support of one aircraft manufacturer and its staff for providing extensive simulator-time and engineering resources necessary to build and validate the models described in this paper. Special thanks are also due to Captain David Austin for flying the flight simulator and providing helpful insights. Kevin Jordan provided support to this project. The authors thank Rowena Morrison, Lance Sherry, Alex Kirlik, and Barry Crane for helpful comments on an earlier draft.

NASA/TM-2000-209600



Some Formal Aspects of Human-Automation Interaction

Asaf Degani
San Jose State University Foundation
San Jose, California

Michael Heymann
Technion, Israel Institute of Technology
Haifa, Israel

George Meyer
Ames Research Center
Moffett Field, California

Michael Shafto
Ames Research Center
Moffett Field, California

National Aeronautics and Space Administration
Ames Research Center
Moffett Field, California 94035

Summary

This paper discusses a formal, mathematically-based, approach to the analysis of operator interaction with machines, in general, and with complex and automated control systems, in particular. It addresses the problem of correctness of displays by asking whether the display provides the necessary information about the machine to

enable the operator to perform a specified task successfully and unambiguously. A formal methodology for verification of interface correctness is outlined. Additionally, a formal procedure for display synthesis, whose objective is to provide a succinct and correct interface for the specified task, is briefly discussed. Special attention is placed on the analysis of pilots' interaction with automated flight control systems onboard a modern commercial aircraft.

Introduction

With the accelerated introduction of advanced automation into a variety of complex, human-operated systems, unexpected problems with overall system performance have been observed (refs. 1-3). Many of these problems have been attributed to deficiencies in communication and coordination between the human and the machine. They are especially acute in cases where the human and the machine share authority over the system's operation (ref. 4). Notable examples of such systems are modern commercial aircraft with advanced flight management systems (ref. 5).

While the partition of authority between the human and machine is commonly at the disposal of the human operator, the machine itself, under some levels of authority, can automatically trigger transitions between configurations. Moreover, in safety-critical systems such as autopilots, automatic envelope protection devices that assume authority and cannot be overridden by the operator, are sometimes incorporated.

One important and well-documented class of advanced automation systems are Automatic Flight Control Systems (AFCS) of modern jetliners. In recent years, faulty pilot interaction with the AFCS has become a major concern in the civil transport industry. This problem has variously been termed as lack of mode awareness, mode confusion, or automation surprises (refs. 6, 7). Two main factors have frequently been cited in accident and incident reports and in the scientific literature, as being responsible for such breakdowns. (1) The user has an inadequate "mental model" of the machine's behavior (refs. 8, 9). (2) The interface between the user and the machine provides inadequate information about the status of the machine (refs. 10, 11). Both factors may limit the user's ability to reliably predict or anticipate the next configuration (e.g., mode) of the machine, and hence may lead to false expectations, confusion, and error (ref. 12).

The interaction of the human operator with the machine is performed by means of an interface through which the user sends and receives signals and stimuli to and from the machine. In many systems, the interface consists of two separate (and disjoint) components: the input-interface, or control panel, through which the user sends inputs (signals and/or commands) to the machine, and the output interface, or display, through which the user receives information from the machine. The user, based on knowledge of and familiarity with the machine (typically obtained from training and operating manuals), possesses a model of the machine's behavior (which we shall refer to as the "user-model").

Traditionally, most of the human factors research on display design has focused on perceptual and cognitive compatibility between the human and the interface format (e.g., ref. 12). Much less research was conducted on the relationship between the interface and the machine being controlled. Notable exceptions are the current work in cognitive and ecological psychology (see for example ref. 13), and the past work in the area of manual control (ref. 15).

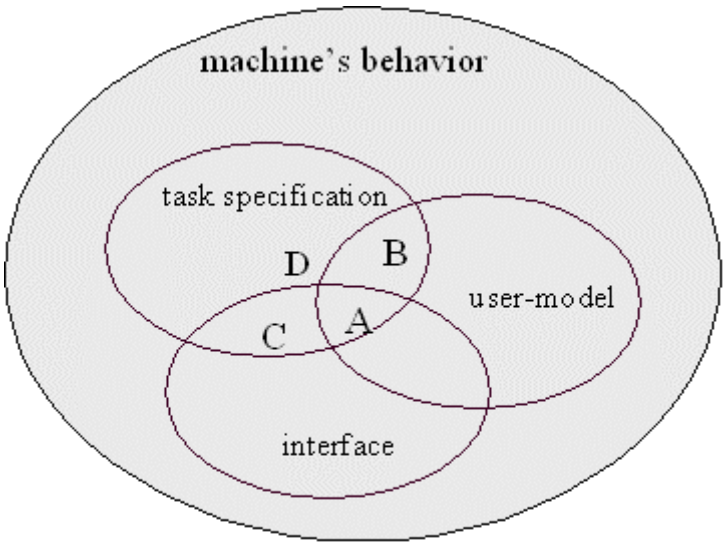


Figure 1. Components of human-machine interaction.

Faulty interaction of the user with the machine, which can lead to catastrophic results in high-risk systems such as commercial aircraft, is variously attributed to either machine failures, or human errors, with the latter sometimes blamed on interface design inadequacies. However, the distinction between human error and interface design inadequacies has remained blurred (see e.g. ref. 15). Furthermore, the possibility of failures because of ambiguous responses of the machine to pilot interaction, has not received much attention to date. The reason for this is the complexity of such advanced automation systems and the absence of rigorous methods for their systematic analysis.

The objective of the present paper is to propose a formal and systematic approach for analyzing human-machine systems, as well as a formal methodology for developing correct and succinct interfaces. The paper is organized as follows: First, we discuss the role of the task specification as related to reliable human-machine interaction. In particular, we examine the interrelations among the machine's behavior, the user-interface, the user's model of the machine, and the task. We then describe vertical flight modes in the autopilot of a modern commercial aircraft, and analyze possible pilot interactions with this system. We follow with a brief description of how a formal analysis can be performed to verify the correctness (or incorrectness) of the user-model and interface. Finally, we outline a methodology for display synthesis whose object is to provide a succinct interface that is correct for the specified task.

Formal Aspects of Human-Machine Interfaces

In the present section we shall examine, in some detail, the required interrelation between the user-interface, the user-model of the machine, and the task specification for reliable and unambiguous human-machine interaction. It will be useful to make several basic assumptions:

- The underlying machine's behavior is given, and can be modeled formally within the framework of a well-defined modeling formalism.
- The machine's actual behavior is deterministic; that is, at each (internal) state of the machine, its response to every action by the user or to external signals is unique and unambiguous.
- The set of operational requirements (tasks) is formally specified.
- The input interface (i.e., control panel) is rich enough to allow the user to execute any one of the specified tasks.
- The user has knowledge of the machine's responses (obtained e.g. from user-manuals and training). This "user-model" of the machine's behavior can be represented formally.

As mentioned earlier, in the interaction between the user and the machine, four elements play a central role. (1) The machine's behavior, (2) the operational requirements, or task specification, formally described in the context of the machine's behavior, (3) the user-model of the machine's behavior, and (4) the interface through which user obtains information about the machine's state and responses. These four elements must be suitably matched in order to insure correct and reliable user-machine interaction.

Figure 1 schematically describes the interrelation between these elements. The large circle represents the set of all machine behaviors. Each of the three inner circles represents the region where one of the elements is 'adequate.' In the present paper we shall always assume that the machine's behavior is given and the task is adequately specified. We are interested in considering the interrelation among the "task specification," "user-model," and "interface". Thus, region **A** represents the situation where all three elements are adequate, and correct interaction is possible. Region **B** represents the situation where the task specification and the user-model are adequate, but the interface is inadequate. Region **C** represents the situation where the task specification and the interface are adequate, but the user-model is not. Finally, region **D** represents the case where both the interface and the user-model are inadequate for the task.

To illustrate the possible discrepancies between the machine's behavior, the user-model, and the interface in relation to reliable task execution, consider the simple machine described in figure 2.

The system starts at state 1, and upon execution (by the user) of event "α ", moves along to either state 2 or state 3, according to whether the condition "C1" is true or false. The dashed edges represent transitions that take place automatically and instantaneously. Thus, if state 2 is reached, the system moves to state 5 immediately, while if it reaches state 3, it moves to either state 5 or to state 4 depending on whether condition "C2" is true or false.

Suppose that the task specification is just to drive the system to state 5 (call it S1). In this case, regardless of whether conditions "C1" and/or "C2" are true or false, this will be the guaranteed outcome of executing event "α". The user does not need to know

the exact path that the system will take from state 1 to state 5. An adequate user-model for the system of figure 2 and specification S1 is shown in figure 3(a).

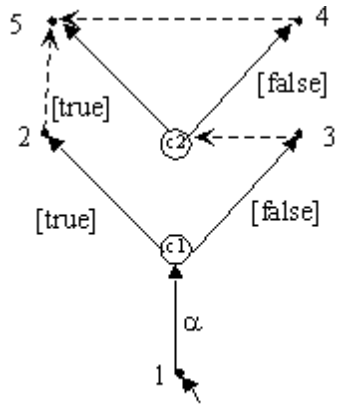


Figure 2. Simple machine example.

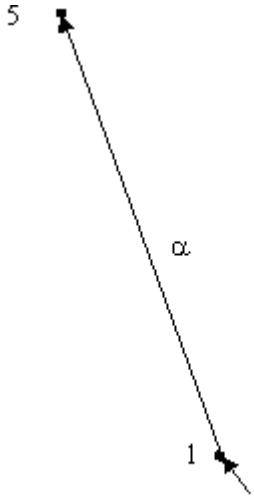


Figure 3(a). User model for specification S1.

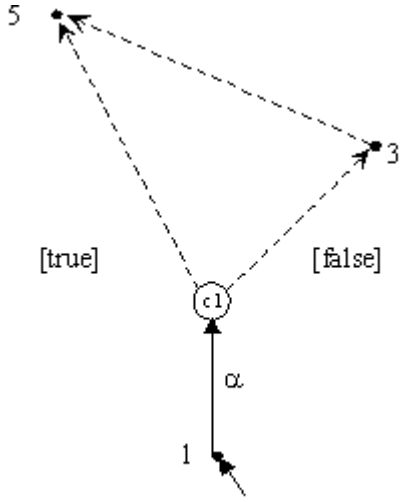


Figure 3(b). User model for specification S2.

On the other hand, if the task specification is to drive the system from state 1 to state 5 via state 3 (call it S2), it is necessary that condition "C1" be FALSE. However, since we do not care whether state 4 is visited or not, it does not matter whether condition "C2" is true or false. An adequate user-model of the system in this case must exhibit the possible paths implied by condition "C1," but the user-model can still be a reduced description of the system as given in figure 3(b). To correctly interact with the system and execute the latter task specification (S2), the user needs to know whether condition C1 is true or false before executing the event "α". (This information must be given to the operator through the interface).

The case in which the user has a correct model (for the specification), but is not provided with an adequate interface (e.g., indicating whether C1 is true or false), is represented by region **B** of figure 1. Clearly, in this case, correct task execution cannot be guaranteed.

Another possibility is that the user has an inadequate user-model. In particular, suppose the user is unaware of the existence of condition C1 (Figure 3(a)), and assumes that the transition to state 5 always traverses state 3. A correct display that indicates the status of condition C1 would not be of any help in this instance, since the user would not associate its value to the machine's behavior. This is the case represented by region **C** of Figure 1. Finally, the worst case (region **D** in figure 1) is when neither the user-model nor the interface are adequate for the specified task.

A Case Study: Automatic Flight Control System

In this section we shall describe and analyze one element of a flight control system onboard a modern jetliner. Specifically, we describe the transition behavior of the autopilot among several vertical flight modes, and examine possible pilot's interaction with these modes. The data for this study was obtained through a series of extensive flight simulations using this autopilot. We begin by describing the control panel through which the pilot interacts with the autopilot, and the display through which the pilot obtains information about the system's behavior.

Interface Description

Figures 4(a) and 4(b) are schematic illustrations of the relevant elements of the "Guidance Control Panel" (GCP) and "Electronic Attitude Display Indicator" (EADI), respectively.

Three primary vertical flight modes of the aircraft are of interest to us: the "Hold Altitude" mode, in which a specified altitude value is maintained constant, and the two distinct climb/descend modes: "Change Level" and "Vertical Speed". The GCP illustrated in figure 4(a) includes three buttons for engagement (by the pilot) of these hold-altitude and climb/descend modes. In the top portion of the GCP are two windows, one indicating the selected altitude setting, and the other indicating the selected vertical-speed setting. The pilot can change the altitude setting by rotating the altitude knob, and change the vertical speed by sliding the vertical-speed wheel up or down.

Figure 4(b) is a schematic illustration of the Electronic Attitude Display Indicator (EADI) of a "glass cockpit" aircraft. It includes the "Flight Mode Annunciator" which indicates (in the top portion of the EADI) the current modes of the aircraft. In figure 4(b) the current vertical mode, displayed in the right-most window, is "Vertical Speed" (the "Thrust" and "Lateral" modes of the aircraft are beyond the scope of this paper). The altitude tape, which provides the pilot with an indication of the current altitude of the aircraft, is displayed in the left side of the EADI. By viewing both the GCP and the EADI, the pilot has knowledge of the GCP altitude setting, the vertical speed setting, the active vertical mode, and the current aircraft altitude at any time.

[figure 4 lost]

Description of Machine Transition Diagram

Figure 5 is a state transition diagram that describes the transitions among several vertical-flight modes of the autopilot under consideration. For operational reasons (to be explained below) the "Hold Altitude" mode is represented twice: Once in its neutral manifestation and once in the "V/S armed" manifestation. Similarly, the

"Vertical Speed" mode appears once as parameterized by a target altitude (V/S to GCP setting) and once as unparameterized (V/S unconstrained climb/descent, without a target altitude). The "Change Level" mode is always parameterized by its target altitude. Also shown is the transition mode "Capture GCP setting." The transitions among the various modes are depicted as the labeled arrows to be explained in detail below.

Each of the modes, represented by a rounded rectangle, can be thought of as a distinct aircraft activity which is fully defined in the autopilot and has an associated set of parameters. For example, the "V/S to GCP setting" represents a climb or descent activity, and is parameterized by the value of the vertical speed setting (positive for a climb and negative for a descent) and the target-altitude setting. Similarly, the "V/S free climb/descent" represents an unconstrained climb or descend activity (and is parameterized only by the value of the vertical speed setting). The "Change Level" mode is fully specified by the target-altitude setting. The two "Hold altitude" activities are parameterized by the altitude setting. Finally, the "Capture GCP setting" is parameterized by the target-altitude.

In this system some of the modes have associated dynamics, and some of the transitions among them are triggered by the internal dynamics of the system. Thus, in the "V/S to GCP setting" activity, the vertical speed setting (which is not explicitly exhibited in figure 5) determines the rate-of-change in altitude. When the altitude reaches a value that satisfies the condition $[\mathrm{Alt} \in \underline{\mathrm{Set}}(\varepsilon)]$, an automatic transition to the mode "Capture GCP setting" takes place. This condition $[\mathrm{Alt} \in \underline{\mathrm{Set}}(\varepsilon)]$ is triggered by the equation

'altitude_error' + ['altitude_error_rate' * 'abs_val ('altitude_error_rate')] / (2 * Nz * g) = 0

becoming true, (where Nz is the normal acceleration during the capture phase). The transition from "Change-Level to GCP setting" to "Capture GCP setting" is triggered dynamically by a similar condition. In the "Capture GCP setting" mode, the level-off maneuver to the target altitude (as set in the GCP) is then executed. When the condition $[\mathit{Alt} = \underline{\mathit{Set}}]$ is met, an automatic transition to "Hold Altitude" is triggered by the autopilot.

Several mode transitions are engaged by the pilot manually. These include the transition from "Change-Level to GCP setting" to "V/S to GCP setting" and its reverse transition, as well as the transition from "Hold Altitude (V/S armed)" to "Change-Level to GCP setting."

The remaining mode transitions are triggered indirectly by the pilot through the change of a parameter. (Parameter changes sometimes trigger subtle mode transitions that cause significant changes in aircraft behavior and are a potential source for confusion and hazard.) Analysis of such transitions and their relation to display design are the focus of the ensuing discussion.

The pilot can change the setting of the GCP altitude at any time. There is a qualitative difference in the autopilot's behavior depending on whether the altitude is changed to a value ahead or behind a specific critical altitude (e.g., the current altitude of the aircraft).

[figure 5 lost]

Here by ahead we mean in a temporal sense, that is, "higher than" the current altitude when climbing, and "lower than" the current altitude when descending. (Similarly, behind means "lower than" when climbing, and "higher than" when descending). These GCP setting changes are shown in figure 5 by the transition labels "=>" for ahead and "<=" for behind. The notation $/\underline{Set} = GCP$ means that the GCP target altitude is substituted by the new setting. Similarly, $/\underline{Set} = Newset$ denotes the resetting of the vertical speed value.

Thus, when the autopilot is in the activity "V/S to GCP setting," changing the GCP altitude dial to a value behind the current aircraft altitude triggers a transition to "V/S free climb/descent." In this activity, the aircraft continues the current climb/descent without any effective altitude constraint. When in "V/S free climb/descent" activity, changing the GCP altitude dial to a value ahead the current aircraft altitude triggers a transition back to "V/S to GCP setting."

Of particular interest are the consequences of changing the GCP altitude setting while in the "Capture GCP setting" activity. Here the critical altitude is not the current aircraft altitude, but rather the altitude when the autopilot transitions into the "Capture GCP setting" mode (called henceforth the capture-start altitude). Thus, changing the GCP altitude setting to a value behind the capture-start, triggers a transition to "V/S free climb/descent." On the other hand, changing the setting to a value ahead of capture-start, triggers a transition to "V/S to GCP setting," and the new altitude becomes the target altitude. (If, upon entering the "V/S to GCP setting" mode, the condition $[\mathrm{Alt} \in \underline{\mathrm{Set}}(\varepsilon)]$ is satisfied by the new GCP setting, an instant transition back to "Capture GCP setting" takes place). The main point is, that by changing the GCP altitude to a value ahead of capture-start, the capture condition is retained, while changing to a value behind capture-start results in an unconstrained climb or descent.

Analysis of Pilot-Autopilot Interaction

Next we examine the pilot's ability to operate the aircraft reliably with respect to the vertical-flight modes just described. Specifically, we wish to examine whether the pilot can anticipate the next mode or autopilot activity, that will result from his or her interaction with the system. To this end, we focus our attention on transitions that are triggered by the pilot upon changing the GCP altitude.

We begin by examining the pilot's interaction with the vertical-speed mode (V/S climb/descend), which we have partitioned into two sub-activities "V/S to GCP setting" and "V/S free climb/descend" because of the distinct responses of the autopilot when in these two activities. We shaded the two activities in figure 5 to emphasize the fact that they belong to the same principal mode.

First let us examine the transition between the two "Vertical Speed" sub-activities. Note in figure 5, that by changing the GCP altitude to behind or ahead of the current altitude and vice versa, transitions are triggered between the "V/S to GCP setting" and the "V/S free climb/descend" sub-activities. The principal activity is "Vertical Speed," (the shaded area in Figure 5) which is well annunciated on the display. However,

there is no annunciation of the transition between the two sub-activities. The pilot can deduce which sub-activity the aircraft is in (that is, whether capture will or will not eventually occur), only by comparing the current aircraft altitude with the GCP altitude setting. (This is implicit knowledge, which many pilots are unaware of and thus have been surprised when the aircraft failed to capture the altitude set in the GCP. In pilots' jargon such an occurrence is referred to as "kill the capture").

Let us now examine the autopilot's response to changes in the GCP altitude setting while the aircraft is in the "Capture GCP setting" mode. Recall that if the GCP altitude is reset to a value ahead of capture-start, the aircraft will capture the newly set altitude by transitioning to the "V/S to GCP setting" activity. However, if the GCP altitude is reset to a value behind the capture-start altitude, the newly set altitude will be ignored, and the autopilot will enter the unconstrained "V/S free climb/descent" activity.

It is noteworthy that the capture-start altitude is always behind the target altitude. Therefore, when the GCP altitude is reset behind the current aircraft altitude, eventual capture is uncertain. In particular, capture will or will not take place, depending on whether the GCP setting is ahead of or behind the capture-start altitude, as can be seen in figure 6.

However, the pilot has no explicit indication of the capture-start altitude, and this information is not retained by any display onboard the aircraft. Therefore, the pilot will be unable to predict reliably whether the autopilot will transition to the "V/S to GCP setting" activity (with eventual capture of the newly set altitude), or into "V/S free climb/descent" activity (and "killing the capture").

User-model

It is interesting to examine the autopilot behavior as described above in relation with the behavior described in the user-model , in this case, the pilot training manuals published by the manufacturer. A full description of the relevant training material is presented graphically in figure 7. The dashed self-loop transitions in the "Vertical Speed" and "Change Level" modes are not explicitly described in the manual, but can be deduced by a careful reader. It is also noteworthy that in the training material there is no differentiation between the two "Vertical Speed" sub-activities, but it is stated that when the aircraft's vertical speed is set "away from" the GCP altitude setting, altitude capture will not take place.

As before, let us focus on the aircraft's response to changes of the GCP altitude setting in the various modes. From the description provided in the training manual, the following responses can be deduced:

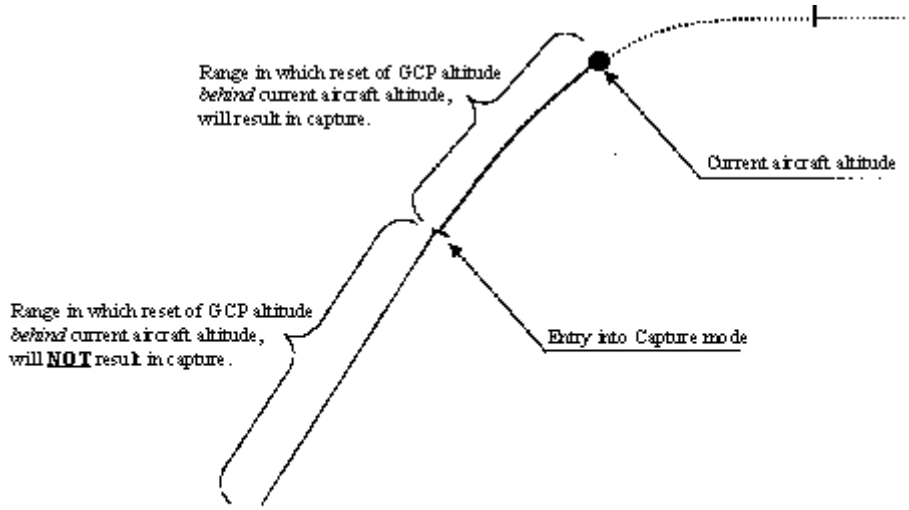


Figure 6. Capture profile.

[figure 7 lost]

> If the GCP altitude is reset behind the current aircraft altitude while in the "Vertical Speed" mode, the aircraft will continue its flight away from the selected altitude. This is consistent with the actual autopilot behavior (as described in the previous sub-section).

Next, we turn to the responses when the autopilot is in the Capture mode. According to figure 7, the system will respond to a change in the GCP altitude setting by reverting back to the "Vertical Speed" (V/S) mode. If the GCP is reset behind the current altitude so that the aircraft flies away from the GCP altitude setting, the current vertical speed will be retained and the aircraft will continue flight at this vertical speed without eventual altitude capture. That is, every change of the GCP setting to a value behind the current altitude, kills the capture. As discussed earlier and shown in figure 6, this description is not the way the autopilot actually works!

Finally, it may be noteworthy that there is another potentially surprising behavior that may be encountered by the pilot. This one occurs when the GCP altitude is reset while the aircraft is in the "Change Level " mode. Here the pilot can expect the aircraft to adjust the climb rate to reach the designated altitude set in the GCP (at a constant airspeed and saturating throttle). The GCP altitude can be reset ahead or behind the current altitude with assured capture. However, if the GCP altitude is changed after (the automatic) transition from "Change Level to GCP setting" to "Capture GCP setting", the autopilot reverts to the "Vertical Speed" mode so that now the eventual capture will depend on the altitude to which the GCP is changed (as described earlier). Thus, even though there is no formal discrepancy between the user-model and the response of the autopilot, the subtle and automatic transition from the

"Change Level to GCP setting" to "Capture GCP setting," causes a qualitative change in the autopilot behavior that the pilot may not anticipate.

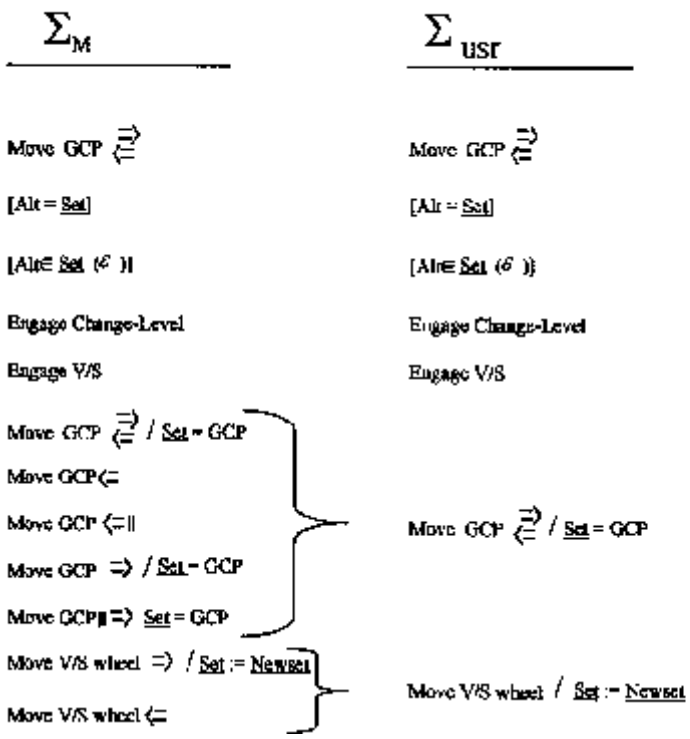
Incident Report

An important source for obtaining information about pilot interaction with automated flight control systems, is NASA's Aviation Safety Reporting System (ASRS). (Following an incident, pilots can submit a detailed report to the ASRS, and are provided with certain assurances of anonymity and waiver of legal action against them (by the US Federal Aviation Administration, see Advisory Circular 00-46D)). A search of the ASRS database revealed several incident reports involving the specific autopilot described above, which referenced changing the GCP altitude during the "Capture to GCP setting" activity (Aviation Safety Reporting System, 1998). Following is one (slightly edited) excerpt. We shall first present the pilot's report and then our interpretation of the incident:

> *On climb to 27,000 feet and leaving 26,500 feet. Memphis Center gave us a clearance to descend to 24,000 feet. The aircraft had gone to "Capture" mode when the first officer selected 24,000 feet on the GCP altitude setting. This disarmed the altitude capture and the aircraft continued to climb at approximately 300 feet-per-minute. There was no altitude warning and this "altitude bust" went unnoticed by myself and the first officer, due to the slight rate-of-climb. At 28,500, Memphis Center asked our altitude and I replied 28,500 and started an immediate descent to 24,000 feet (ref. 17).*

In this incident, the first officer set a new altitude value (24,000) while the aircraft was climbing and in the "Capture" mode to 27,000 feet. As discussed earlier, changing the GCP altitude to a value behind the capture-start altitude will result in a transition to "V/S free climb/descent" and to an unconstrained climb (which in this case was arrested manually by the crew at 28,500 feet). Due to the slow rate of climb (300 feet-per-minute) computed by the autopilot at this altitude, the continuation of the climb beyond 27,000 feet was not easily noted by the crew.

Table 1. Correspondence between machine events and user-model events.



The incident clearly indicates the subtle pilot interaction (changing the GCP altitude value) that triggered the transition from "Capture GCP setting" to "V/S free climb/descent"-two qualitatively different activities. Further, it points out the subtlety of the display cues available to the pilots about the transition between these two activities (compounded here, of course, by the slow rate-of-climb at which the aircraft was leaving 27,000 feet).

Formal analysis of user-model correctness

The consistency of the user-model and the machine model can also be analyzed formally as described briefly below. A detailed exposition of a formal methodology for analysis of user-model and display correctness is given in Heymann & Degani (forthcoming).

Let Σ_M denote the set of events, or *transition labels* , of the actual machine model. Since the user-model is an abstraction (simplification) of the machine model, the events that appear in the user-model, are a reduced subset of the machine's event set as explained next.

The event set Σ_M consists of three disjoint subsets; 1. Σ_M^o - the set of *observed-events* that includes all machine events that are presented in the display and appear also in the user-model, 2. Σ_M^m - the set of *masked* events (that consists of groups of two or more events each). Each group has a single representative in the user-model (and in

the display). 3. Σ_M^u - the set of *unobserved-events* that are neither displayed nor appear in the user-model.

In view of the above, the event set Σ_{USR} of the machine's user-model consists of the union of the event sets Σ_M^o, the event set $\Pi(\Sigma_M^m)$ (where $\Pi(\Sigma_M^m)$ denotes the set of events obtained after masking the events in Σ_M^m), and the "empty event" ε that represents the set of unobserved events.

We turn now to the autopilot example of the previous section. The first five events in Table 1 are observed events as can be seen from the correspondence between the two columns of the Table. The remaining events are masked events that consist of two groups, that comprise, respectively, five and two events. Each of these groups is mapped to a single event in the user-model as seen in the Table. (There are no unobserved events.)

In actual operation, the machine is driven by events from Σ_M. The user tracks the progress of the machine with the aid of the display, where he observes events in Σ_{USR}, and the user-model at his disposal. Thus, the user-model and the machine evolve concurrently. But they are only partially synchronized because not all machine events are observed, and some are masked, and because the user-model is only an abstraction of the actual machine's behavior.

We shall say that the user-model is adequate for a given task, if under all operating conditions of the machine, the user is able to execute the task successfully, while reasoning about the progress of the machine through the user-model at hand. Thus, the "concurrent operation" (as discussed above) of the machine and the user-model, must never lead to an ambiguity, in so far as execution of the task as specified is concerned.

In the present autopilot example, the task specification may consist, for example, of the requirement that at each instant and under all operating conditions, the user must be able to deduce (unambiguously) whether or not the GCP altitude will eventually be captured.

To formally decide whether the display and associated user-model are adequate for the specification, we proceed by forming the *composite model* obtained by computing the *masked synchronous product* of the machine model and user-model. This product is computed as follows. Suppose that the machine is at configuration q at which a transition labeled α is defined, leading to a configuration q' (we denote this by $q \xrightarrow{\alpha} q'$). Assume that when the machine is at configuration q, the user-model is at a corresponding configuration p. The event α can be either *observed, masked, or unobserved*.

If α is an observed event, it is required, for adequacy of the user-model, that a corresponding transition be also defined at p, leading to, say, p'. That is, there must exist a transition $p \xrightarrow{\alpha} p'$. (Otherwise we would conclude that

the user-model is an incorrect abstraction of the machine behavior, regardless of the task specification.) In the composite model there will appear a transition labeled α, from the configuration (pair) (q,p) to (q',p'). That is, there will be in the composite model a transition

$(q,p) \xrightarrow{\alpha} (q',p')$.

If α is a masked event, there must be a corresponding transition $p \xrightarrow{\Pi(\alpha)} p'$ in the user-model, where $\Pi(\alpha)$ is the (masked) image of α in Σ_{USR}. The transition in the composite model will appear as $(q,p) \xrightarrow{\Pi(\alpha)} (q',p')$. The fact that the event labels in the composite model are taken as those from the user-model is because the composite model is formed "from the point of view of the user".

Finally, if α is unobserved, the transition in the composite model will appear as $(q,p) \xrightarrow{\varepsilon} (q',p)$ since there is no corresponding transition in the user-model and the transition is "viewed" by the user as the empty or *silent* transition.

For the user-model to be adequate for the task specification, two conditions must be met. First, as stated earlier, the user-model must not block transitions in the composite model. That is, if a transition is defined in the machine model, there must be a corresponding transition in the user-model. In case of an *observed* transition there must be a corresponding transition with the same label in the user-model, and in case of a *masked* transition there must be a corresponding transition labeled by the masking image. Thus, in particular, every configuration that is reachable in the machine model is also reachable in the composite model. Finally, the composite model must not exhibit nondeterminism (i.e., ambiguity) with respect to the task specification.

If the non-blocking condition is not satisfied, we must conclude that the user-model is an incorrect abstraction of the machine behavior. If the composite model exhibits nondeterminism with respect to the task specification, we must conclude that the abstraction is "too coarse" for the specification and hence inadequate. In this case more detail must be included in the user-model in order to satisfy the specification.

In the case of the autopilot, the composite machine is obtained by forming the *masked synchronous product* of the machine model of figure 5 and the user-model of figure 7 with the aid of the event correspondence given in Table 1. The result is the composite model of figure 8.

It is readily seen in figure 8 that the user-model is inadequate in that there are two configurations in the composite model from which nondeterministic transitions take place. Specifically, from the *Capture GCP* configuration, there are two transitions labeled *move GCP* that nondeterministically lead to either *V/S to GCP setting* or to *V/S Free Climb/Descent,* without the user being able to distinguish between them. Nondeterministic transitions emanate also from the *Hold Altitude (V/S Armed)* configuration sharing the label *Move V/S wheel*.

Display Synthesis and Interface Resolution: A Synopsis

We have seen in a previous section an example of an advanced automation system in which an inadequate display for the specified operational task can lead to ambiguous and faulty interaction of the operator with the machine. ASRS incident reports corroborate these findings.

As stated earlier, a central objective of the research reported in the present paper, is to develop a methodology for systematic display synthesis. Below we shall give a brief heuristic outline of the proposed methodology, and describe its underlying concepts in an informal way using a simple illustrative example. A detailed technical exposition of the methodology can be found elsewhere (Heymann & Degani, forthcoming).

Let us consider the simple discrete system described in figure 9:

This system consists of 4 states, and is initialized at state 1 (as described by the entering arrow). There are 3 buttons labeled α, β, and γ, that the user can press at discretion, and these trigger transitions among the states as described in figure 9 by the labeled arrows. If for example, the system is at state 1 and β is pressed, it moves to state 2. If it is at state 2 and α is pressed, it moves to state 3 and if β is pressed it moves to state 4. If it is at state 4 and α is pressed, nothing happens, etc.

[figures 8, 9, 10 lost]

The user is permitted at any time to press any button that she/he wishes, provided the user does not drive the system to state 4 (which represents a danger zone). The task specification in this very simple example consists therefore of the requirement never to enter state 4. Our objective is to "synthesize" a display and user-model that will enable the user to navigate the system safely; that is, will indicate at each instant which of the buttons, if any, the user is permitted (or not permitted) to press so as not to violate the specification. Formally, this implies a partition of the state set into two disjoint subsets as shown in figure 10, one being the subset of legal states, that consists of the states 1, 2 and 3, and the other being the set of illegal states, or state 4.

Clustering the set of legal states into one super-state and the set of illegal states into another super-state yields the system depicted in figure 11, in which all transitions among legal (or respectively, illegal) states are now viewed as self-loops in the corresponding super-states.

We immediately note that the event (button) labeled γ never leads to the illegal state set, and this button may always be pressed without risk of violating the specification. However, this is not the case for the transitions labeled α and β, which may or may not be legal, depending on internal detail inside the legal super-state. Thus, to decide whether the buttons α or β may be pressed, it is not sufficient to know that the current state is legal. More detailed information is needed.

The question that arises immediately is how much more detail is actually required. Obviously, we could provide the user with a full description of the system (in our example, figure 9), and at each instant let the user deduce from it which transitions are legal. But this may be (and usually is) an unnecessary level of detail. We are aiming at designing (or deriving) a display and user-model that, while sufficient and reliable, is

as succinct as possible. The interface resolution problem consists of deciding systematically how much information is minimally sufficient.

[figures 11, 12 lost]

In other words, we need to refine the description of figure 11 to the point where non-determinism with respect to entry to the illegal set is removed. Specifically, the resolution must determine unequivocally when it is legal to press the buttons α or β, or not.

Figure 12 depicts a "resolved" system where we distinguish between state 1 and the cluster of states 2 and 3. This is thus an adequate user-model and interface for the system that satisfies the specification.

The task specification considered in the above example is of course quite trivial, and our methodology is designed to handle much more complex specifications. The most immediate generalization of the above example, which also explains the methodology required for the autopilot example of the previous section, proceeds along the following line of thought. We are given a state-machine description of the system, in which the set of states is partitioned into a disjoint set of state clusters that represent disjoint activities as described, for example, in figure 13. Thus, each state cluster represents an activity, and the display must be designed so as to inform the user unambiguously, at each instant, what the consequences (i.e. the ensuing activities) are of every event the user can trigger. To this end, the partition into the specified activities may not be a fine enough partition (just as in the autopilot example), and further refinement (sub-partitioning) may be necessary. The procedure to achieve the coarsest refinement that supports the necessary display is called interface resolution and is described in detail in Heymann & Degani, (forthcoming) (ref. 18).

[figure 13 lost]

Conclusions

In this paper we have investigated the precise connection between the machine's behavior, the task specification, the required user interface, and the user-model for ensuring that correct and unambiguous interaction between the user and the machine be possible. In particular, we have focused attention on the problem of verifying that a given display and user-model are adequate for the specified task. We also addressed the problem of how a correct display and user-model for a given task can be synthesized.

We have shown how this problem is crucial in pilot interaction with cockpit automation by exploring the vertical autopilot modes onboard one modern commercial airliner. We have demonstrated how in this autopilot, serious ambiguities exist with respect to the pilot's interaction with the machine. These ambiguities are directly attributable to inadequacy of the autopilot's display and prevent the pilot from being able to determine whether the aircraft will level-off at the specified altitude. Such ambiguities, which directly lead to, so-called, automation surprises, are documented in many pilot incident reports.

References

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

2. Wiener, E. L. (1993). Crew coordination and training in the advanced-technology cockpit. In E. L. Wiener, B. G. Kanki, & R. L. Helmreich (Eds.), Cockpit resource management. San Diego: Academic Press.

3. Wiener, E. L., Chute, R. D., & Moses, J. H. (1999). Transition to glass: pilot training for high-technology transport aircraft (NASA Contractor Report 208784). Moffett Field, CA: NASA Ames Research Center.

4. Mellor, P. (1994). CAD: Computer aided disasters. High Integrity Systems, 1(2), 101-156.

5. Abbott, K., Slotte, S. M., & Stimson, D. K. (1996). The interface between flightcrews and modern flight deck systems. Washington, DC: Federal Aviation Administration.

6. Woods, D., Sarter, N., & Billings, C. (1997). Automation surprises. In G. Salvendy (Ed.), Handbook of human factors and ergonomics (pp. 1926-1943). New York: John Wiley.

7. Billings, C. E. (1996). Aviation automation: The search for a human centered approach. Hillsdale, NJ: Erlbaum.

8. Javaux, D., & De Keyser, V. (1998). The Cognitive Complexity of Pilot-Mode Interaction: A Possible Explanation of Sarter & Woods' Classical Result. In G. Boy, C. Graeber & J. Robert (Ed.), Proceeding of the International Conference on Human-Computer Interaction in Aeronautics Conference (pp. 49-54). Montreal, Quebec: Ecole Polytechnique de Montreal.

9. Sarter, N. B., & Woods, D. D. (1995). How in the world did I ever get into that mode? Mode error and awareness in supervisory control. Human Factors, 37(1), 5-19.

10. Indian Court of Inquiry. (1992). Report on accident to Indian Airlines Airbus A-320 aircraft VT-EPN at Bangalore on 14th February 1990. Indian Government.

11. Norman, D. A. (1990). The 'problem' with automation: inappropriate feedback and interaction, not 'over-automation.' Phil. Trans. Research Society London, B 327, 585-593.

12. Degani, A., Shafto, M., & Kirlik, A. (1999). Modes in Human-Machine Systems: Constructs, representation, and classification. International Journal of Aviation Psychology, 9(2), 125-138.

13. Wickens, C. D. (1992). Engineering psychology and human performance. New York: Harper-Collins.

14. Vicente, K. J. (1999). Cognitive work analysis. Mahwah, NJ: Lawrence Earlbaum.

15. Rouse, W.B. (1977). Applications of control theory in human factors-II [special issue]. Human Factors, 19(5).

16. Aviation Week and Space Technology

. (1995). Automated cockpits: who's in charge? Part 1 and 2. 142(5 and 6).

17. Aviation Safety Reporting System (1998). FMC altitude capture function reports. Search Request No. 5183. Mountain View, CA: Battelle Memorial Institute.

18. Heymann M., & Degani A. (forthcoming). On formal abstraction and verification of human-machine interfaces: the discrete event case.