2003-01-2998

Detecting and Simulating Pilot Errors for Safety Enhancement

Todd J. Callantine
San Jose State University/NASA Ames Research Center

ABSTRACT

This paper presents research on computational models of pilot activities for detecting and simulating pilot errors, and discusses how these techniques may be used to enhance aviation safety. The discussion addresses improved feedback about pilot errors to support training, and the use of actual and simulated error data to understand how errors may impact new air traffic management concepts and flight deck automation. The research is supported by the System-Wide Accident Prevention project of the NASA Aviation Safety Program.

INTRODUCTION

Understanding how human error impacts safety in complex systems, and using this understanding to improve training programs and design new systems, are important goals of human factors research. Methods aimed at characterizing errors, preventing them, and mitigating their effects—including cognitive engineering, human reliability analysis (HRA), human error identification (HEI), and usability assessment techniques—have received considerable attention. Cognitive engineering seeks to provide the tools necessary to "engineer systems that leverage the unique capabilities for awareness that humans bring to these systems against the complex problems that arise" [12, p. 174]. HRA attempts to determine the probability that an operator will fail to properly respond to an event [14]. HEI techniques focus on characterizing observed errors; "Simply put, error analysis is an essential component of safety management" [24, p. 320]. All of these methods generally address safety-critical 'boundaries' and human cognitive strengths and limitations that bear on situation awareness.

Good designs enhance situation awareness. They explicate 'boundaries' between safe and unsafe performance to the interacting human and machine agents that need to be aware of them. Robust systems enable agents to easily identify when a safety-critical boundary is in danger of being crossed, and take the necessary action(s) to prevent it. For human operators, errors play a crucial role in developing situation awareness [12, 22, 23].

DESIGNING FOR SAFETY—Examining accidents and incidents helps address error-related problems through, for example, training targeted at preventing observed errors. But designers face considerable challenges developing, implementing, and evaluating the safety of new designs. Future aviation and Air Traffic Management (ATM) systems that implement new operational concepts, for example, require new operator procedures and automation to improve efficiency without sacrificing safety and robustness (e.g, [19]). However, it is practically impossible to apply cognitive engineering methods such as cognitive task analysis to complex systems like ATM at 'design time' [12]. In a similar vein, Johnson criticizes HRA as lacking the proper scope and predictive power to inform systems development. He asserts among other things that such approaches fail to take group interactions and contextual issues into account [15]. Usability-oriented analyses, such as the Cognitive Walkthrough [18] and the Technique for Human Error Assessment (THEA) [11, 17] would also benefit from a clear 'starting point,' with important safety-critical interactions already identified.

SIMULATION-BASED APPROACHES — To avoid problems with 'paper and pencil' analyses of complex systems, research in domains like aviation and ATM has relied primarily on simulation, particularly real time human-in-the-loop (HITL) simulation, to collect data for analysis with existing analysis methods (e.g., [19, 20]). However, as the number of participants grows, the required development effort, as well as the costs and logistical issues associated with conducting such simulations, also increase (e.g., [9]). Furthermore, analyzing data from a large-scale simulation is time-consuming, and it may be difficult to achieve the proper focus for the analysis because of the large number of (intended and unintended) experimental factors. Data that reveal 'interesting' interactions may receive undue attention to the exclusion of other salient interactions that might also arise in practice. Methods to reduce the analysis burden, maintain consistency, and recognize the relative importance of interactions are therefore needed.

Fast-time simulations ATM researchers have recently turned to fast-time simulations with embedded agents that represent human operators and automation (e.g., [5, 13, 21, 25]). While real-time HITL simulations enable researchers to simulate only a few conditions per day, fast-time simulations provide an opportunity to examine, Monte Carlo fashion, data from numerous trials across multiple conditions. Fast-time simulations are limited by the fidelity of the simulated agents, but can be helpful for exploring safety-critical boundaries and analyzing the potential benefits of new concepts. They can also provide a basis for selecting experimental conditions to test in follow-on HITL studies. However, to support design, methods are needed for systematically simulating errors and examining their effects on safety.

DETECTING AND SIMULATING ERRORS—Against this backdrop, this paper describes how a computational model of a ubiquitous and reasonably well-understood class of agents in the ATM system—pilots flying glass cockpit aircraft—can serve several useful purposes, including:

- Detecting errors that HITL simulation pilots make to increase the effectiveness of pilot debriefings and improve analyses, to support ATM system design.
- Systematically simulating pilot errors in fast-time agent-based simulations of new ATM operational concepts, to support safety analyses and HITL simulation experimental design.
- Detecting errors that glass cockpit simulator pilots make, and capturing the error context, to improve training.

The pilot activity model may reflect current-day aircraft automation and procedures—either for training enhancement, or the design of near-term future ATM systems that accommodate current-day aircraft. Alternatively, the model may need to be expanded to represent why and how operators should perform activities using new automation and procedures. In either case, the same model can support all of the above purposes in a complementary manner.

This paper presents research that address pilot error simulation and pilot error detection—using the same model—within the Crew Activity Tracking System (CATS) framework. CATS was initially developed to provide the knowledge required by pilot aiding and training systems [8] and has since been used as an analysis tool (e.g., [1, 7]) and as the basis for intelligent agents [5]. The paper aims to show how the CATS model of pilot activities can be leveraged to support safety enhancement in the broader context of training and future ATM system development research.

The remainder of the paper is organized as follows. It first presents a generic CATS activity model. It then describes how CATS can detect pilot errors with such a model of pilot activities and discusses an example of CATS detecting an error that might otherwise go unreported. The paper then describes how simulated pilot agents use the model, and presents a methodology for systematically simulating precisely the sorts of errors CATS detects. The paper then discusses implications of this research for simulator-based training programs and future ATM system design efforts.

CATS ACTIVITY MODEL

A CATS model is a model of operator activities tailored to systems that have modes of operation (including, of course, modern 'glass cockpit' aircraft). A CATS model represents the 'correct paths through the system' at multiple levels of abstraction, down to the level of operator actions that correspond to various button presses, dial settings, and data entries that an operator can perform using the system's interface. A CATS model can be constructed for a single operator, or it can represent the activities of multiple operators, such as a two-person flight crew. This section provides an overview of a CATS model in its generic form, and describes the properties of a CATS model important for constructing one.

A CATS model is normative, because it represents the 'preferred' methods for accomplishing required functions. A CATS model also has descriptive qualities, however, because it represents all the possible ways that an operator can correctly accomplish required functions (cf. [26]). A given airline may train pilots to use a specific autopilot mode in a particular situation, such as the Vertical Navigation (VNAV) mode for descents from cruise altitude to 10,000 feet—this is a so-called 'preferred method.' However, a pilot may opt to descend using a different mode, such as the Flight Level Change (FL CH) mode. (Some future ATM concepts specify 'one right way' of complying with certain clearances; a CATS model can also represent 'pure normative' operations.)

Figure 1 shows the generic form of a CATS model. High-level activities (referred to as 'functions') are supported by generic methods for performing them, each of which is preferred in different situations. (If a particular control is not part of a preferred method in some situation, then the question of why the interface includes it necessarily arises.) Methods may share 'common activities,' as shown in Figure 1. For example, whether a flight crew chooses VNAV or FL CH mode to descend, they must first ensure that a new target altitude is set. For new interface designs, the hierarchy of activities in the CATS model can end above the action level, to provide top-down interface design guidance; the CATS model is completed when the operator actions allowed by the prototype interface are specified.

REPRESENTING OPERATIONAL CONTEXT — Each activity in a CATS model contains conditions that represent when the activity is 'nominally preferred,'

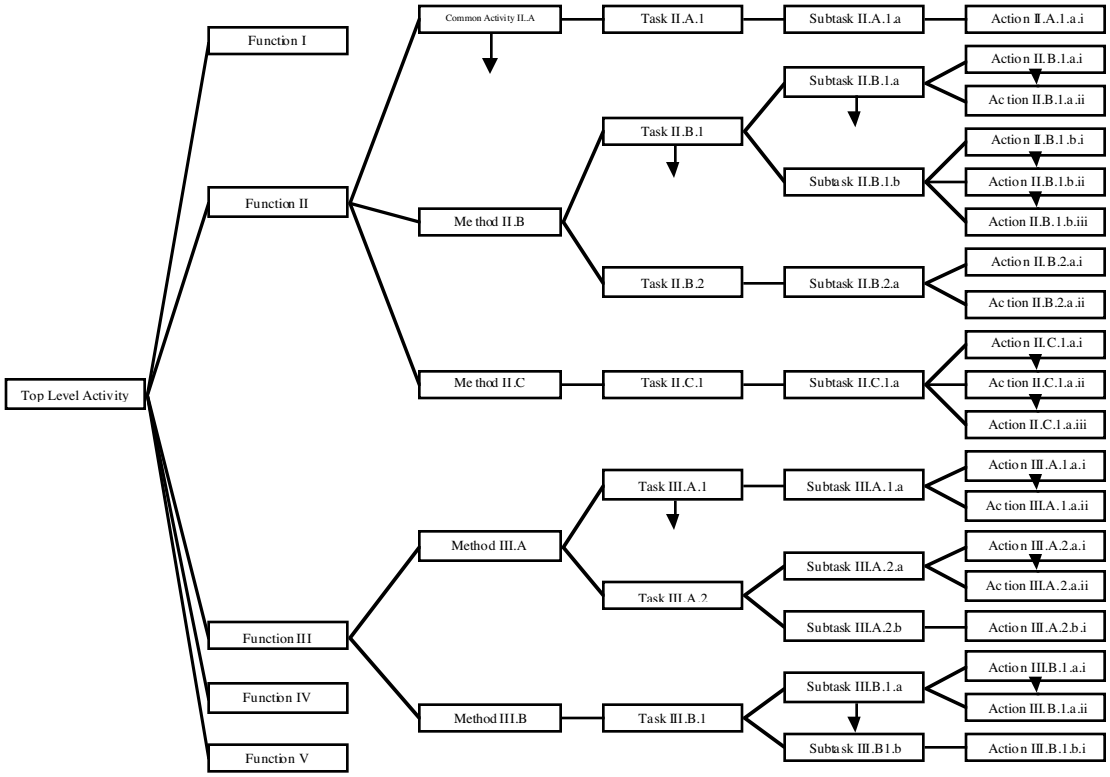

Figure 1. Generic CATS model structure.

represented as 'AND-OR trees' (rules) comprised of Boolean-valued 'context specifiers' named to express contextual information like operators do (Table 1 below provides some examples). At any given time, a particular set of context specifiers summarizes the current operational context. Context is defined as salient states, constraints, and key relationships between them. This definition imparts a 'constraint satisfaction' flavor to operator behavior. A variety of models are suitable for generating the current set of context specifiers. Previous implementations have typically used rules, although Bayesian Belief Networks, fuzzy rules, or other models may be used for resolving context.

REPRESENTING PROCEDURES, CONCURRENCY, AND COGNITIVE ACTIVITIES — Given a current context, evaluating the conditions in the CATS model yields the current set of activities that the operators should preferably perform. The CATS model says nothing about which of two or more activities in the current set should be performed first, simply that the current context satisfies conditions for performing all of them. Historical and predicted future values of states and constraints are in some cases important, but as long as the current context includes this information, the CATS model itself is memoryless—no history information about past activities is necessary.

The arrows in Figure 1 express that the CATS model can represent procedures for controlling the system, in addition to unordered activities. For example, 'Common Activity II.A' precedes either 'Method II.B' or 'Method II.C.' True to the memoryless property, procedural dependencies are represented via the AND-OR trees of conditions in the model. Performing any procedural step should produce a change that is reflected in the current state of the controlled system and, in turn, in the current operational context. When a pilot, for example, performs a procedural step, the context changes such that the next step of the procedure 'enters the current set' of preferred activities. For example, performing 'Common Activity II.A' should produce a change in context that specifies that one of the different methods should follow. Procedures that do not interact form separate 'branches' of the CATS model tree structure.

The CATS methodology views supervisory control behavior as constraint satisfaction; the CATS model represents a series of transformations from state and constraint information into 'what to do.' While it is designed to allow abstraction, to formulate useful models in a reasonable period of time, a CATS model can include cognitive, perceptual, or verbal activities to support cognitive engineering analyses. The generic model in Figure 1, for example, decomposes some activities into a single child activity. This expresses the possibility that some cognitive activities required to perform a given activity may have been omitted from the model for parsimony. (Intervening levels above actions may also be omitted, i.e., actions can link directly to tasks in Figure 1.) If the CATS model includes cognitive or perceptual activities, expected 'durations' can be used to delay prediction of the next procedural step.

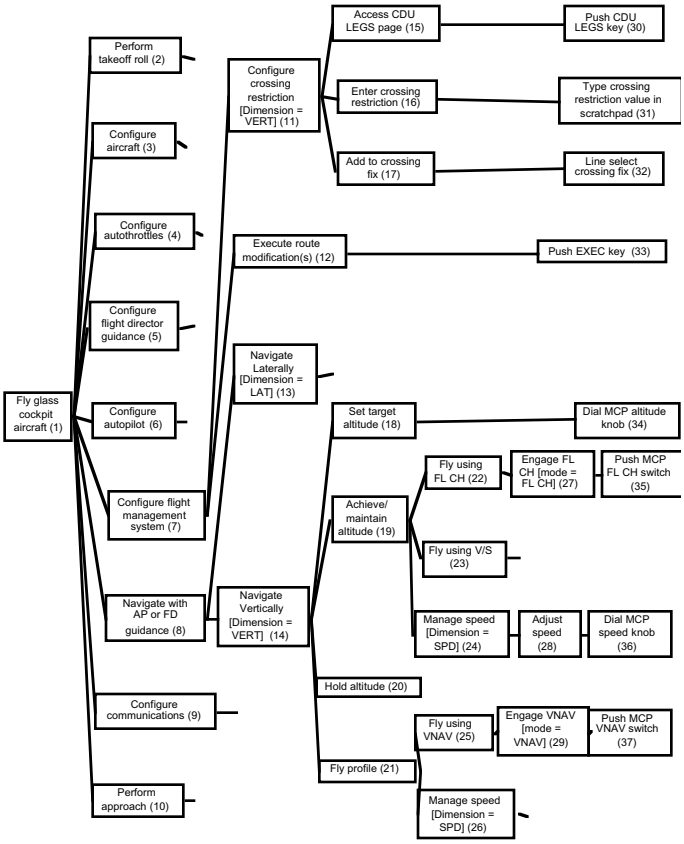

Figure 2. Fragment of a CATS model for B757 operations.

MODEL OF BOEING 757 (B757) PILOT ACTIVITIES — Figure 2 shows a fragment of a CATS model of B757 pilot activities developed according to the above principles (another example appears in [1]). The model decomposes the highest level activity, 'fly glass cockpit aircraft,' into sub-activities as necessary down to the level of pilot actions. Figure 2 illustrates eight actions. All actions derivable from aircraft data are included in the complete model. Each activity in the model is represented with conditions (rules) that express the context under which the activity is nominally preferred. The parenthesized numbers in Figure 2 refer to Table 1, which lists the AND-OR trees that comprise the conditions.

Figure 2 illustrates how the alternative methods and common activities shown in Figure 1 are manifest in the B757 CATS model. The activities 'achieve/maintain altitude' and 'fly profile' represent alternative methods that support the 'navigate vertically' activity. The activity 'set target altitude' is an activity common to both methods. The multiple methods structure appears recursively at a lower level in the model, because 'fly using FL CH' and 'fly using V/S (Vertical Speed mode)' both support the 'achieve/maintain altitude' activity. 'Manage speed' is an activity common to both of these methods. A different 'manage speed' appears in support of 'fly profile,' because speed management may be performed in a different manner when flying an FMS

Table 1. AND-OR trees of conditions under which the CATS model in Figure 2 represents activities as 'nominally preferred.' CATS predicts an activity when its conditions—plus all the conditions of its parent activities—are satisfied by the current operational context.

(1) start-of-run
(2) (not above-runway-elevation)
(3) (and (not above-clean-speed) (not flight-surfaces-within-limits) (not gear-within-limits))
(4) (not autothrottle-armed)
(5) (not flight-director-on)
(6) [(and (not autopilot-cmd-mode-engaged) above-1000-feet-AGL)]
(7) (or (not programmed-route-within-limits) route-uplink-received)
(8) (and above-1000-feet-AGL (or autopilot-cmd-mode-engaged flight-director-on))
(9) (not comm-frequency-within-limits)
(10) (or approaching-glideslope-intercept-point approach-localizer-intercept-point)
(11) (not crossing-restriction-within-limits)
(12) route-modifications-within-limits
(13) (or autopilot-cmd-mode-engaged flight-director-on)
(14) (or autopilot-cmd-mode-engaged flight-director-on)
(15) (not cdu-page-LEGS)
(16) (and cdu-page-LEGS (not crossing-restriction-built))
(17) (and cdu-page-LEGS crossing-restriction-built)
(18) (not mcp-altitude-within-limits)
(19) (or (and (not current-altitude-within-limits) (not profile-within-limits-for-now)) expedite-needed)
(20) (and current-altitude-within-limits (not profile-within-limits-for-now))
(21) profile-within-limits-for-now
(22) (or (not altitude-close-to-target) expedite-needed)
(23) altitude-close-to-target
(24) (or fl-ch-engaged vs-engaged)
(25) profile-within-limits-for-now
(26) vnav-engaged
(27) (not fl-ch-engaged)
(28) (not target-speed-within-limits)
(29) (and (not vnav-engaged) (not capturing-required-altitude))
(30) (not cdu-page-LEGS)
(31) (not crossing-restriction-built)
(32) crossing-restriction-built
(33) route-modifications-within-limits
(34) (not mcp-altitude-within-limits)
(35) mcp-altitude-within-limits
(36) (not target-speed-within-limits)
(37) mcp-altitude-within-limits

profile. The "DIMENSION' indications are a way of indicating to CATS which activities in the model are the highest level activities that are supported by multiple methods. Earlier versions fixed the level at which methods (i.e., autoflight mode selections) were represented, so such designators were not necessary (see [8]). For comparison to research that considers human errors involved with Flight Management System (FMS) Control and Display Unit (CDU) manipulations (e.g., [11]), the model fragment in Figure 2 also illustrates one of numerous FMS configuration tasks. (This is for illustration only, because the data used in the present research do not include CDU action data.).

ACTIVITY TRACKING FOR ERROR DETECTION

This section describes CATS, and how CATS uses the B757 model to detect pilot errors. CATS implements a methodology for activity tracking in a computer-based system that has been validated to work in real and fast time [3, 8]. Figure 3 depicts the CATS architecture and

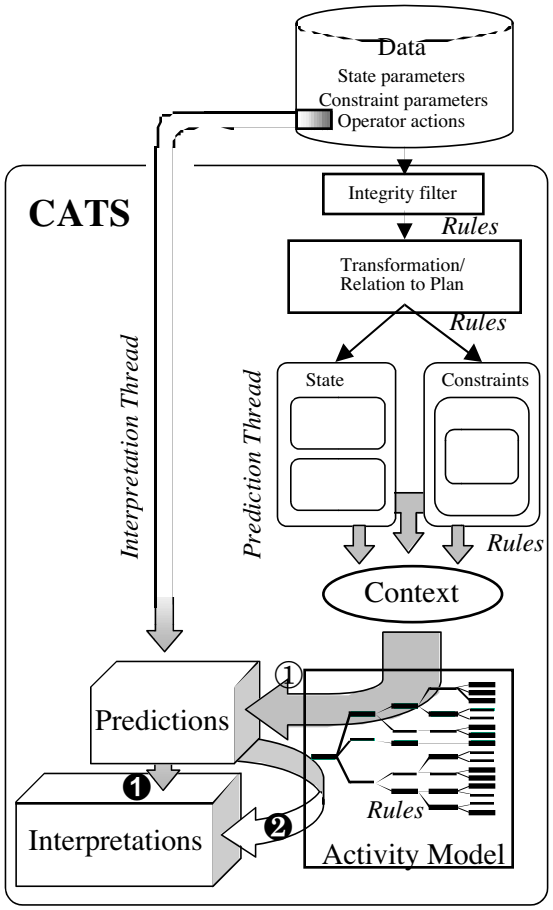

Figure 3. Information flow in the Crew Activity Tracking System.

processing method generically. To derive the set of 'context specifiers' that represent the current operational context, CATS requires that the controlled system is properly instrumented to provide salient state, constraint, and operator action data. This requirement presents problems when using flight data from real aircraft (as discussed in [1]), but not for research flight simulators that receive clearances via data link communications [19, 20].

ACTIVITY TRACKING PROCESS — Two threads comprise the activity tracking methodology as implemented in CATS: a 'prediction thread' responsible for generating the context information necessary to predict nominally preferred activities, and an 'interpretation thread' that interprets operator actions. Using its context representation, CATS generates predictions from its activity model (①). The prediction thread searches the CATS model top-down, paring the search as it goes by only considering activities whose 'parent' activities are predicted. (In addition to speeding the search procedure, this scheme allows for parsimony in specifying the conditions in the model.) The 'interpretation thread' in CATS compares detected operator actions to predicted activities (❶), and assesses actions that it cannot directly interpret using the predictions by periodically referencing the activity model until enough new data has arrived to disambiguate possible interpretations (❷). Thus, activity tracking is not merely the detection of operational 'deviations' (e.g., 'altitude below glidepath'), but instead additionally considers the operator's role in causing any such deviations.

Error Types CATS identifies two types of errors: errors of omission, and errors of commission. It further identifies 'right action/wrong value' errors (e.g., dialing the MCP altitude—the right action—to the incorrect target value). CATS identifies errors by using information in the CATS model that enables it to assess actions (or inactions) in light of the current operational context and the context that results from operator actions (or inactions). The CATS error detection scheme entails 'closing the loop' between a representation of correct task performance and the controlled system, and evaluating feedback from the controlled system to ensure it 'jibes' with correct operator activities. Given that the system is operating normally and providing 'good data,' this is a powerful concept. Note that CATS does not base its determinations on a 'formulaic' representation of how errors would appear in a trace of operator activities, nor attempt to further classify errors, e.g., 'reversals' (cf. [14]). This is because the CATS model does not represent the 'steps' of procedures as 'step A follows step B,' but instead represents procedures implicitly by explicitly specifying the conditions under which operators should preferably perform each action, as discussed above

CATS detects errors of omission by starting a timer when it determines that an activity should be performed. If CATS does not detect an action that supports that activity (either the preferred action, or one that is part of an alternative valid method) before the timer expires, CATS signals a possible error of omission. In a detailed implementation, the length of time should be context-dependent. A complete representation of the constraints on operation provides the necessary information [2].

CATS interprets a detected action as correct whenever it either matches a prediction, or supports an alternative valid method for performing a required high-level activity. After CATS receives updated contextual information, if it determines that an action does not meet these criteria, it signals an error of commission. Some actions, such as 'browsing actions' that do not directly bear on satisfying operational constraints, should be identified as such, rather than as possible errors (see the discussion in [16]). This is a subject of continuing research.

ERROR DETECTION RESULTS — This subsection briefly describes an example of CATS detecting an actual error from flight data collected from the NASA Langley B757 ARIES aircraft. During a flight, the crew received a

clearance to 'climb and maintain 16,000 feet.' (An observer recorded the constraint imposed via the clearance in the absence of data link communications.) Because the target altitude on the aircraft's Mode Control Panel (MCP) showed 10,000 feet, the context specifier 'mcp-alt-within-limits' does not appear in the current context representation. Using its model, CATS therefore predicts that the crew should 'dial MCP altitude knob' to set the new target altitude. Instead, the crew pushes the VNAV switch. CATS could not interpret this action right away, and after repeated attempts still could not reconcile the action with the current context representation, which reflected that the aircraft was still in level flight in Altitude Hold (ALT HOLD) mode at the previous altitude. CATS therefore flagged the VNAV switch action as an error. As detailed in the step-by-step description of the error presented in [3], within about forty seconds, the crew corrected the situation by setting the proper target altitude and initiating a VNAV climb. Thus, the 'error' did not result in any incident. This was the only minor slip CATS detected in data from several flights. In previous simulator research, CATS also detected some minor errors [8]. While CATS detects errors successfully, the rarity of even minor errors illustrates the difficulty of error-related research.

TRAINING AND ANALYSIS APPLICATIONS — Connecting CATS to piloted flight simulators affords error detection in both simulator-based training applications and HITL ATM simulations. For training applications, CATS can detect errors, including minor slips like that described above, that instructors might not notice. Because CATS records detailed information about the context in which the error occurred, instructors can analyze the error and, if warranted, place other trainees in the identical situation. Much of this process may be automated using a CATS version that includes training support. Similarly, in HITL simulation applications, CATS enables observers to focus on specific interactions, and detects 'other' errors observers may not notice. It also operates online and supports data playback to support detailed debriefings [1].

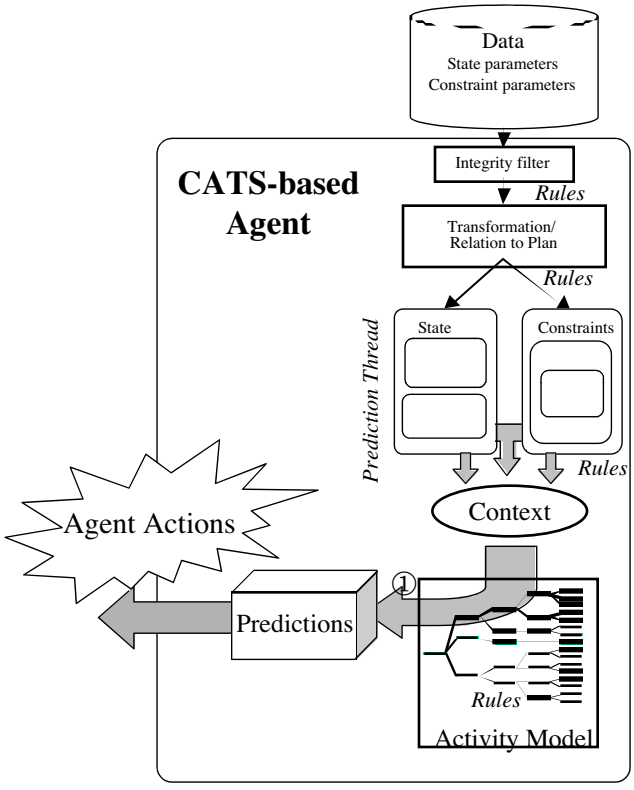

Figure 4. CATS-based agent that performs 'preferred correct' activities based on a CATS model.

SIMULATING ERRORS

This section describes how simulated pilot agents use the same model that is used for error detection in a complementary way. It first describes some previous research CATS-based agents, then describes how CATS-based pilot agents can 'fly' simulated aircraft and make realistic errors in a systematic way to support safety analyses.

CATS-BASED AGENTS — The nominal CATS agent scheme is shown in Figure 4: the CATS-based agent simply executes the actions that a CATS activity tracking implementation predicts for the same operational context. Research on air traffic controller agents illustrates how CATS-based agents can use knowledge external to the CATS model and context representations [5]. They use the CATS model simply to structure the high level 'flow' of activities. When the context dictates that some activity should be performed, the air traffic controller agents reference a 'skill library' and 'control rules' to select applicable strategies and determine, for example, the precise values to include in clearances. By representing beliefs about the current operational context separately from the true context, and manipulating the way the agents update their beliefs probabilistically, such agents can be made to err in reasonably realistic ways [4]. The manipulations lead the agents to miss important cues, perform procedural operations out of order, and forget key information. As a result, the agents fail to perform activities correctly.

This research seeks CATS-based pilot agents that make realistic errors. The pilot agents are different from the air traffic controller agents because, given that pilots know the what the constraints on the aircraft's flight path are, the CATS model of B757 pilot activities encapsulates all the knowledge (i.e., rules) about how to fly the aircraft using the autopilot. Simply manipulating the context probabilistically can have undesirable effects, because context is represented at multiple levels of abstraction. If a high-level context specifier is removed from the active set, the top-down prediction search may stop at a high level in the model. The scheme can simulate erroneous inaction, but in other cases, agents may perform activities that have nothing to do with the current operational context (e.g., a agent that needs to set a new MCP target

altitude might instead tune the radio or lower the landing gear).

Context-oriented error generation in pilot agents The pilot agents therefore produce errors using the process depicted in Figure 5 [6]. First, the error-generation process identifies any actions the operator should perform through the normal CATS prediction process using the set of context specifiers that accurately represent the current operational context. It then uses a selected 'level' in the model to identify all the activities in the currently predicted branches of the model within a specified distance from an action. A slight variation, referred to as using the 'expanded set of activities,' selects activities not only from the currently predicted branches in the tree, but from all branches below a predicted activity at a certain level. As described below, applying the error generation process with the expanded set of activities is sometimes useful.

After it identifies a set of relevant activities, the process identifies every context specifier that appears in the CATS model conditions for performing each activity in the relevant set. The process next compares this set to the set of true context specifiers, and identifies those that are 'not present, but could be if the context is erroneously formulated' and those that are 'present, but might not be if the context is erroneously formulated.' The process then follows a strict procedure for adding or removing, as necessary, all combinations of context specifiers from the true set. For each chosen

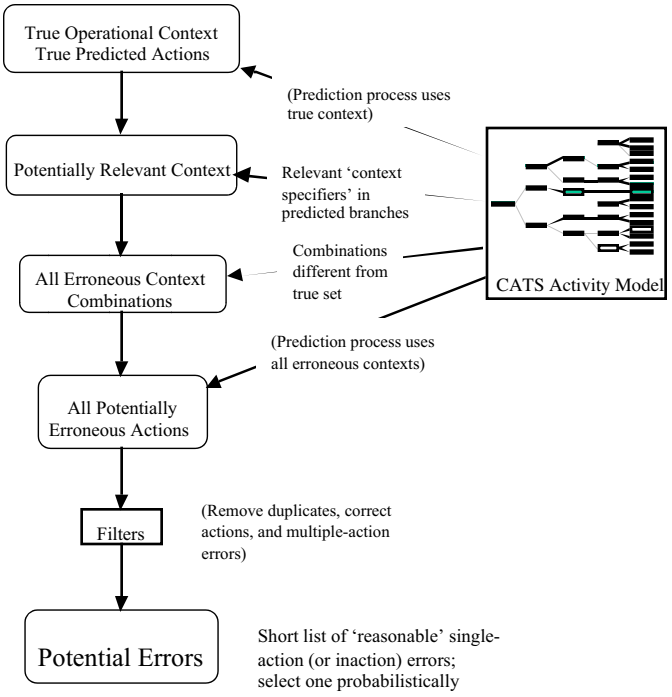

Figure 5. Error-generation process that uses the CATS model and prediction process to produce errors 'related' to correct actions in a given context, and maintain links to the error context.

combination, the process adds the context specifiers that do not appear and removes the ones that do appear in the true set. The process then predicts activities based on each of these revised sets. Once it has produced a set of potentially erroneous predictions, the process invokes several filters. From each set of predicted activities, the process identifies actions, removes actions that happen to match correct actions, and removes duplicate sets of predicted actions.

Realism and Correspondence to CATS Error Detection Errors generated for a variety of situations are given in [8]. They show that the error-generation process makes pilot agents err as if they miss important cues, perform procedural operations out of order, and forget key information. The error-generation process produces errors that are reasonable for a given context; the errors generated are not 'too far' from the correct actions in the true context. The process yields either 'inaction' or incorrect actions that appear in the same branch(es) of the CATS model as the correct actions. However, simply selecting these actions as possible errors does not provide information about the context that led to the error. Instead, the process automatically records the erroneous context representation associated with errors. For pernicious errors, capturing the context enables designers to better understand the errors and prevent them.

A CATS-based agent can produce the errors CATS detects if both are using the same model. For the example above, in which the crew slipped and attempted to engage a autopilot mode to climb without first setting the new target altitude, the CATS prediction is for the crew to perform the 'dial MCP altitude knob' action to set the new target altitude. Using only the predicted branch of the CATS model, the error-generation process yields three 'single-element' errors: '[no action],' 'push FL CH switch,' and 'push LNAV switch.' As noted above, the crew actually performed the 'push VNAV switch' action. The reduced set of errors generated from only the predicted branch of the CATS model does not list this option; however, this action is included in the set of errors generated from the expanded set of activities. Thus, this example supports the notion that using the expanded set of activities as input to the error generation process yields the required set of 'reasonable' errors that might occur in a particular operational context.

Validation Issues A rigorous validation strategy entails connecting both a CATS-based agent and a CATS activity tracking implementation to the same controlled system. Then, when the CATS-based agent applies the error generation process to produce an error, CATS would detect the error online. Further research is needed to construct such a test bed. In addition, research is required on how to make a given erroneous context persist long enough to produce 'lasting errors' that CATS would detect. In essence, the error-

generation process would need to 'freeze' the erroneous context so that the agent would not correctly perform the action that was erroneously omitted on the next processing cycle. Also, if the context shifts immediately to the 'true' context, an error of commission would be followed immediately by a corrective action. This would limit the utility of a safety analysis utilizing such agents, because the agent would correct its errors before their effects could be manifested in the system.

Safety Analysis Applications Given that a new operational concept implemented in a fast-time simulation with embedded agents reveals no safety problems under nominal conditions, how can designers assess the robustness of the system in the presence of errors? One way might be to invoke the error generation process probabilistically, then probabilistically select a single error from the possible set to execute on the controlled system. While this scheme could produce interesting results, it fails to address many of the same problems of HITL simulations. It may yield an enormous body of data over numerous Monte Carlo trials, but it provides no clear indication about how relevant error-related interactions are in terms of overall system operations. Given that a simulated concept operates safely under nominal conditions, a more principled method to examine robustness in the face of errors should first identify errors with safety consequences. Such a scheme might work as follows:

1. For an individual agent, apply the error-generation process only when the context changes—when CATS predicts a 'new set' of activities, as in [6].
2. Record the entire system state (perhaps by dumping it to a file).
3. Allow the simulation to 'play out' for each error in turn, starting from the most recent recorded state, log relevant metrics, and observe the effects of the error.
4. Proceed deterministically to the next error (or the next set of errors for another simulated agent).

Is this practical? Consider a fast-time simulation of a new terminal-area ATM concept, in which controller agents issue clearances to traffic transitioning to final approach. Further consider that we are interested in the effects of pilot errors that lead to aircraft non-compliance on overall system safety; agents must be able to detect and cope with errors. Now assume that under the concept a controller agent nominally issues at most $N_c = 3$ clearances to a given aircraft, and that the error-generation process produces at most $N_e = 5$ realistic errors in response to any given clearance. Further assume that a scenario with $N_a = 20$ aircraft takes at most $T_{sim} = 3$ minutes to execute in fast time. Considering each error for each aircraft independently, an upper bound on the time required to generate all possible single-error effects is:

$$N_c \cdot N_e \cdot N_a \cdot T_{sim} = 15 \text{ hours}$$

In fact, the time per run would decrease as recorded states progress toward the end of the scenario. Moreover, it might be sufficient to play out each error condition for only a short time, subject to terminating conditions imposed according to the controller agent's simulated workload or observed aircraft separation violations or airspace violations. The time is further reduced if, for the particular scenario, the controller only needs to clear some aircraft. And, if multiple instances of the simulation run on multiple computers, with each simulation generating errors for a few designated aircraft, more economies result. Considerable additional research is needed to construct such a test bed, because developing the required controller agent, handling multiple simultaneous errors, as well as other issues all need to be addressed. Nonetheless, such an approach may prove effective for identifying issues that warrant further investigation in detailed HITL simulations.

DISCUSSION

This paper has described how a computational model of pilot activities, used within the CATS framework, can detect and simulate pilot errors. It generally addresses three problems with current approaches, viz., errors that go unnoticed, poor error predictions, and the effort involved with gathering data and characterizing either observed or potential errors and their impacts. This section discusses several ways in which such capabilities can support safety enhancement efforts.

ERROR DETECTION — First, previous research has shown that online analysis of pilot performance reduces overhead associated with simulation data analysis by automatically detecting errors, logging the context in which occur, and enabling interactions to be depicted graphically and replayed to support detailed subject debriefings [1, 7]. Many of the advantages realized in HITL settings also apply to simulator-based training programs. Second, CATS-based error detection supports other error assessment and usability techniques; with detected errors in hand the analyst can focus on examining interface issues. For example, detected errors together with contextual information provide a clearer 'starting point' for 'Cognitive Walkthrough'-style usability analyses [18]. Perhaps a particular interface made it difficult for operators to figure out what to do, whether what they did produced the desired effect, or whether some action was in fact an error that they should have noticed and corrected. The approach also may be thought of as 'jump-starting' scenario-based analysis techniques, such as THEA [11, 17]. More generally, the context-oriented nature of the CATS approach supports analyses of situation assessment problems that lead to error conditions [10]. With required inputs to such techniques in hand, designers need less time to derive the information necessary to begin developing solutions to any

observed problems. Finally, carrying the error-detection application a bit further provides support for cognitive engineering methods. For example, the CATS model can be modified to better represent the conditions under which operators should perform activities. The model can also use more detailed representations of context that better map to operator representations discovered through debriefings. The updated model thereby reflects an evolving understanding of the required tasks and situation awareness (cf. [12]).

ERROR SIMULATION — The CATS agent error-generation capability could be usefully applied in fast time simulations of new concepts. First, pilot agents that err in fast time would enable designers to investigate how the overall ATM system responds to pilot errors. Second, its emphasis on the link between realistic errors and the context in which they may occur means that it too could provide many of the advantages of the complementary error-detection applications above. Third, analyzing realistic errors in a fast-time simulation setting could help tailor HITL simulation studies to examine operating regimes where errors are most likely to impact safety. In general, designers could use such an approach to address situation awareness-related problems and their effects on system safety and robustness early in the design process.

CONCLUSION

This paper presented research on computational models of pilot activities for detecting and simulating errors. It described research in two areas: activity tracking, in which a model of nominally correct pilot activities is used to detect errors, and error simulation, in which similar models, together with methods for manipulating agent beliefs, form the basis for agents that err in realistic ways.

REFERENCES

1. Callantine, T. (2001). Analysis of flight operational quality assurance data using model-based activity tracking. SAE Technical Paper 2001-01-2640. Warrendale, PA: SAE International.
2. Callantine, T. (2002). A representation of air traffic control clearance constraints for intelligent agents. In A. El Kamel, K. Mellouli, and P. Bourne (Eds.), Proceedings of the 2002 IEEE International Conference on Systems, Man, and Cybernetics, #WA1C2, CD-ROM.
3. Callantine, T. (2002). Activity tracking for pilot error detection from flight data. Proceedings of the 21st European Annual Conference on Human Decision Making and Control, Glasgow, 16-26.
4. Callantine, T. (2002). CATS-based agents that err. NASA Contractor Report 2002-211858. Moffett Field, CA: NASA Ames Research Center.
5. Callantine, T. (2002). CATS-based air traffic controller agents. NASA Contractor Report 2002-211856. Moffett Field, CA: NASA Ames Research Center.
6. Callantine, T. (2003). Error Generation in CATS-based Agents. NASA Contractor Report 2003-212263. Moffett Field, CA: NASA Ames Research Center.
7. Callantine, T., and Crane, B. (2000). Visualizing pilot-automation interaction. Abbot, K., Speyer, J., and Boy, G. (Eds). HCI-Aero 2000 International Conference on Human-Computer Interaction in Aeronautics, Toulouse: EURISCO, 87-92.
8. Callantine, T., Mitchell, C., and Palmer, E. (1999). GT-CATS: Tracking operator activities in complex systems. NASA Technical Memorandum 208788. Moffett Field, CA: NASA Ames Research Center.
9. Callantine, T., Prevôt, T., Battiste, V., and Johnson, W. (2003). Agent-based support for distributed air/ground traffic management simulation research. Proceedings of the AIAA Modeling and Simulation Technologies Conference, Austin, TX, August.
10. Endsley, M. (1999). Situation awareness and human error: Designing to support human performance. Proceedings of the High Consequence Systems Surety Conference, Albuquerque, NM.
11. Fields, R., Harrison, M., and Wright, P. (1997). THEA: Human error analysis for requirements definition. Technical Report 2941997, York, UK: University of York Computer Science Department.
12. Flach, J., and Rasmussen, J. (2000). Cognitive engineering: Designing for situation awareness. In N. Sarter and R. Amalberti (Eds.), Cognitive Engineering in the Aviation Domain. Mahwah, NJ: Lawrence Erlbaum Associates, 153-179.
13. Gore, B., and Corker, K. (2000). A systems engineering approach to behavioral predictions of an advanced air traffic management concept. Proceedings of the 20th Digital Avionics Systems Conference, Philadelphia, PA, 4.B.3-1-4.B.3-8 (CD-ROM).
14. Hollnagel, E. (2000). Looking for errors of omission and commission or “The Hunting of the Snark” revisited. Reliability Engineering and System Safety, 68, 135-145.
15. Johnson, C. (1999). Why human error modeling has failed to help systems development. Interacting with Computers, 11, 517-524.
16. Mitchell, C. (2000). Horizons in Pilot Training: Desktop Tutoring Systems, In N. Sarter and R. Amalberti (Eds.), Cognitive Engineering in the Aviation Domain. Mahwah, NJ, Lawrence Erlbaum Associates, 211-251.
17. Pocock, S., Harrison, M., Wright, P., and Johnson, P. (2001). THEA: A technique for human error assessment early in design. In M. Hirose (Ed.), Human-Computer Interaction: INTERACT'01, Amsterdam: IOS Press, 247-254.
18. Polson, P., and Smith, N. (1999). The Cockpit Cognitive Walkthrough. In R. Jensen, B. Cox, J. Callister, and R. Lavis (Eds.), Proceedings of the 10th International Symposium on Aviation

Psychology, Columbus, OH: The Ohio State University Department of Aviation, 427-432.
19. Prevôt, T., Lee, P., Callantine, T., Smith, N., and Palmer, E. (2003). Trajectory-oriented time-based arrival operations: Results and recommendations. Proceedings of the 5th USA/Europe Air Traffic Management Research and Development Seminar, Air-Ground Cooperation Track, Budapest, Hungary, June.
20. Prevôt, T., Palmer, E., Smith, N., and Callantine, T. (2002). A multi-fidelity simulation environment for human-in-the-loop studies of distributed air ground traffic management. AIAA 2002-4679. Reston, VA: American Institute of Aeronautics and Astronautics.
21. Pritchett, A., Lee, S., Abkin, M, Gilgur, A., Bea, R., Corker, K., Verma, S., Jadhav, A., Reynolds, T., Vigeant-Langlois, L., and Gosling, G. (2002). Examining air transportation safety issues through agent-based simulation incorporating human performance models. Proceedings of the 21st Digital Avionics Systems Conference, Irvine, CA, 7.A.5-1-7.A.5.13 (CD-ROM).
22. Rauterberg, M., and Aeppli, R. (1996). Human errors as an invaluable source for experienced decision making. In A. Mital, H. Krueger, S. Kumar, M. Menozzi, and J. Fernandez (Eds.), Advances in Occupational Ergonomics and Safety, Vol. 1, Cincinnati, OH: International Society for Occupational Ergonomics and Safety, 131-134.
23. Reason, J. (1992). Human Error. Cambridge: Cambridge University Press.
24. Shorrock,S., and Kirwan, B. (2002). Development and application of a human error identification tool for air traffic control. Applied Ergonomics, 33, 319-336.
25. Stroeve, S., Blom, B., van der Park, M. (2003). Multi-agent situation awareness error evolution in accident risk modeling. Proceedings of the 5th USA/Europe Air Traffic Management Research and Development Seminar, Air-Ground Cooperation Track, Budapest, Hungary, June.
26. Vicente, K. (1999). Cognitive work analysis. Mahwah, NJ: Lawrence Erlbaum Associates.

CONTACT

Todd J. Callantine, Ph.D., is investigating human performance modeling and model-based design methods for future aviation systems. He can be contacted at tcallantine@mail.arc.nasa.gov.