Opportunities and Challenges of Modeling User Behavior in Complex Real World Tasks

WOLFGANG SCHOPPEK & DEBORAH A. BOEHM-DAVIS

University of Bayreuth, Germany
George Mason University, Fairfax, VA

Keywords: cognitive modeling, human-machine interaction, automation, task analysis, human error

1. Introduction

In basic research, cognitive modeling has proven a valuable methodology for developing theoretical assumptions, testing their dynamic interactions, and exploring the scope of theories. Cognitive architectures such as ACT-R (Anderson & Lebiere, 1998) or Soar (Rosenbloom, Laird, & Newell, 1992) provide a common basis for different models and enhance communication and exchange of solutions. In applied contexts, modeling of real tasks and operators could further the understanding of human-machine systems; validated models could provide an objective guide to design and training decisions. However, they only provide a first step towards rapid development of valid real world models, because the architectures do not constrain many decisions the modeler has to make. This paper will illustrate this issue through a case study of developing and validating a model of complex real-world behavior – that of flying an aircraft.

1.1 Requirements for modeling complex real world tasks

Models of real world tasks have requirements that are not always applicable to models used solely for research purposes. Specifically, to be useful, models first must be developed in reasonable amounts of time. Although what is regarded as reasonable will depend on the specific task and the questions the model should answer, the typical development time of one man year is too long for most applied cases. Second, models must be valid. Depending on the application or area of interest, there are different kinds of validity for models: structural, predictive, etc.. We claim that first of all, models must make correct predictions about some aspects of performance, ideally without the need for empirical testing. Considering how rare it is for models to make accurate predictions even for basic research, this requirement may be even more ambitious than the first.

In this paper, we argue that, in general, cognitive architectures can help shorten development time as they come with a built-in structure for representing tasks that can help form the foundation for the model-building activity. We further argue that cognitive architectures can also help address the second requirement – that of ensuring validity. Most cognitive architectures are based on assumptions that have been developed and tested empirically. For example, within the ACT-R cognitive architecture, early theories of skill acquisition were compared against empirical data (Anderson, 1982). More recently, models of skill acquisition have been applied to and validated in a broader range of applications (Taatgen, 1999; Lee & Anderson, 2001; Taatgen & Lee, 2003). As a consequence, standard techniques for modeling learning of procedures from written instructions are now available.

The problem with architectures, however, is that they are content independent and do not sufficiently constrain the modeling of specific tasks. Table 1 illustrates this for the example of decision making. Assuming that decisions can be described by some form of mapping from actions to conditions, and that conditions, actions, and the decision rule itself are mentally represented, there are a number of possibilities for these representations, which may occur in many different combinations. All the possibilities shown in Table 1 can be implemented in an architecture like ACT-R, but the architecture does not suggest what combinations a researcher should select for modeling a given task.

To leverage cognitive architectures for modeling real world tasks, standard solutions that are architecturally grounded are needed to represent entities such as "decision making", "acquisition of cognitive skills", or "intention management". Theoretical standard solutions would be formulated in the language of a specific architecture and empirically tested (not ad hoc) in a range of applications, thus having the status of theories.

Table 1: Framework for a symbolic theory of decision making that would support modeling of complex real world tasks. The sinuous lines indicate that various combinations between representations of conditions, decision rules, and actions are possible. Solutions used in ACT-Fly are printed in italics.

Condition	**Decision rule**	**Action**
Class of situations - classification by similarity - classification by rules	Similarity matching	*Method* *- represented procedurally* *- represented declaratively*
Specific parameters of a situation	*Fixed symbolic rule*	*Single step*
Combination of the former two ("situation plus exception")	Deduction of rule from general knowledge	

Such theories can best be developed through tight cooperation between applied and basic researchers. Modeling in basic research often aims at developing architecturally grounded theories of certain processes, but it needs to be informed by the uses to which it will be put in order to be useful to applied researchers. Basic models often ignore how the processes they represent are influenced by real-world context. Thus, from the view of the basic researcher, models of complex real world tasks are beneficial because the applied model is a good test bed for the value of the theory. Also, if the modeler of a real-world task carefully specifies what parts of the model are derived from the architecture and for what parts new solutions had to be developed, the basic researcher is informed about ways in which the architecture needs to be extended.

Thus, we are arguing here that the efficiency of modeling behavior in real world tasks can be improved when models are developed within established architectures even though not all solutions to specific modeling problems can be derived from the architecture. For our project, we chose ACT-R as our architecture, because it is a widely accepted psychological theory with a broad empirical basis and a modeling language. On its symbolic level, ACT-R distinguishes between declarative memory, made up of a network of typed "chunks", and procedural memory, consisting of production rules. ACT-R also assumes a subsymbolic level ascribing continuous parameters such as activation or utility to each symbol. In memory retrievals, for example, both levels determine what is retrieved. On the symbolic level, the memory element must match the symbolic specification of the retrieval command; given one or more chunks are matching the specification, the subsymbolic level selects the most active of these chunks and determines the retrieval latency.

We used the ACT-R architecture to develop a model, called "ACT-Fly", that simulates the interaction between airline pilots and the flight management system when completing a descent. The goals for the modeling project were to (1) gain an idea of how long it might take to produce a model of this complex real-world task using a cognitive architecture and (2) determine how well this model could predict human performance. In developing the model, we started with GOMS task analyses and then implemented them in ACT-R 4.0. Some problems posed by the task could neither be solved with GOMS nor with standard ACT-R; thus we added minor extensions and modifications to ACT-R, which are described below.

1.2 Characteristics of our modeled task

Until recently, typical tasks modeled with ACT-R were laboratory experiments such as memorizing lists of words (Anderson, Bothell, Lebiere, & Matessa, 1998) or discriminating previously learned statements from distracters (Anderson & Reder, 1999); tasks characterized by clear, shallow goal hierarchies, a repetition of very short trials with the same underlying structure, a static environment, and low demands for prior knowledge. Flying, in contrast, involves heterogeneous goals that may compete for limited resources. For example, the goal of monitoring the plane passing a critical waypoint (location in space) competes with the goal of encoding a new clearance from air traffic control (ATC). Also, much prior knowledge must be brought to the flying task.

The specific task we chose to model is flying a simulated Boeing 747-400 from the end of the cruise phase to the initial approach fix using the aircraft's automated flight management system. A number of different specific scenarios were flown to include a variety of conditions, such as different ATC clearances or descent profiles. The task was a vertical navigation task, chosen because this is one of most error-prone aspects of automated flying (see, for example, Sarter & Woods, 1992).

Two basic modes of automation are available for accomplishing the task: a fully automated mode - called VNAV - where the autopilot receives most of the reference values from a preprogrammed flight plan; and semiautomatic modes (referred to as FLCH and V/S), where the reference values must be provided by the pilot. VNAV is generally the preferred mode of operation because it optimizes the flight profile and fuel consumption. However, if ATC requires quick changes to the flight plan, the pilot can respond more flexibly using semiautomatic modes. In all of these modes, the behavior of automation and aircraft must be monitored and set points must be provided on time.

Even though the range of tasks modeled within the ACT-R framework has increased considerably in the past several years, this task presented a number of challenges. For example, although the challenge of modeling dynamic tasks has been addressed in the past few years, resulting in models of air traffic control (Lee & Anderson, 2001) and driving tasks (Salvucci, 2001), the dynamics involved in the present task are different (relatively slow). Users typically have to wait for minutes until they can judge the success of a certain intervention. Because new and rather unpredictable demands can arise during the waiting time, multiple interleaved streams of behavior are the result. Thus, some of the features of the present task required us to find new solutions that are not obvious in the architecture and cannot be derived from existing models that successfully predict behavior in other tasks.

The flying task is also difficult to validate as it is a typical supervisory control task, where the user must mainly observe what the automation is doing, resulting in very sparse records of observable behavior. This makes the comparison between simulated and real behavior difficult.

2. ACT-Fly model

In developing our model, we drew theoretical background from a combination of GOMS (Card, Moran & Newell, 1983) and ACT-R. We conducted task analyses based on flight manuals, interviews with subject matter experts, and interactions with the flight simulator. These analyses produced a set of methods by which pilots use the flight management system to fly a descent which we represented using GOMS elements in an NGOMSL (Kieras, 1997) framework. We then had to develop ACT-R representations of those GOMS elements and NGOMSL methods and a way of translating those analyses to ACT-R code. The translation from the NGOMSL code to ACT-R code was accomplished through an Excel spreadsheet tool. This idea is similar to Salvucci & Lee's (2003) "ACT-Simple", which allows the translation of a simple modeling syntax into ACT-R. Compared to ACT-Simple with its focus on keystroke level processing, our solution is targeted at modeling decision making through a representation of higher level procedures.

With early versions of the model, we found that when we relied entirely on methods, the model was too rigid to respond to unexpected events. Specifically, we found that the sequential structure of methods often did not match the less predictable order of events in the environment. Another problem with the method-only controlled version was the lack of situation awareness. The scope of a method is typically limited to local aspects of a task, and so there was no inherent need to create a "big picture". To achieve more flexibility and better situation awareness, we introduced an additional level to the control structure that operates in a non-sequential, rule based manner. In the terms of Table 1, situations at this level are classified by rules, decisions are also made by a fixed symbolic rule, and the actions taken are either single steps or the execution of a method.

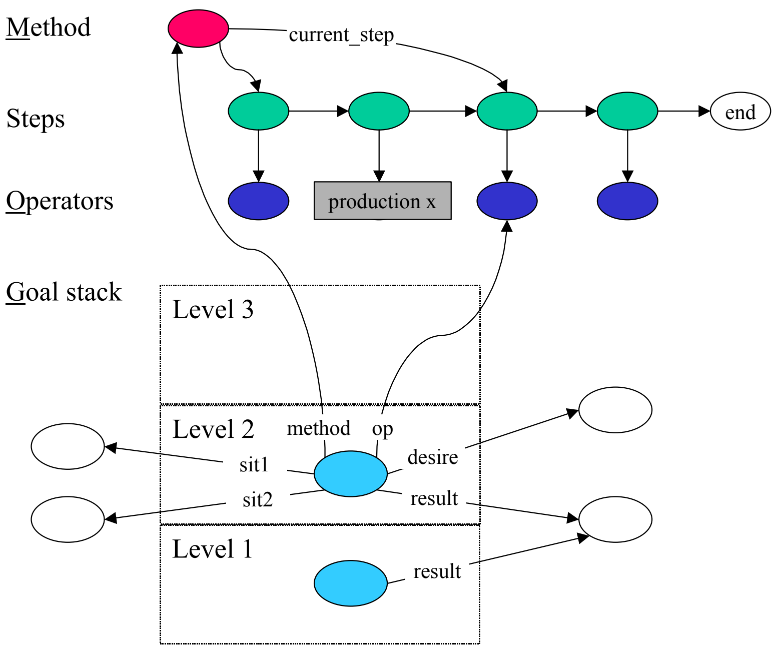

Figure 1: Declarative structures of ACT-Fly. The structure at the top represents a method. The structure at the bottom shows the goal stack. Level 1 and Level 2 are holding one goal chunk each. The goal chunk on Level 2 has its slots filled with chunks representing the situation (e.g. the current method) and current setpoints ("desire"). Results are passed across the levels.

The model communicates via a TCP/IP socket connection with a PS1 747-400 desktop flight simulator manufactured by Aerowinx. The model writes its commands and requests to a socket and reads answers from it. Likewise, the connection software reads commands and requests from the socket, forwards the commands to the simulator, gets the requested values from the simulator, and writes them to the socket.

Technically, the model is divided into three parts. A domain independent part contains generic operators and functions that handle the execution of hierarchical methods and the management of intentions. A domain specific part contains domain knowledge such as methods and rules. The third part contains the interface to the external environment. The benefits of this separation are that the general part can be used for modeling other tasks (done e.g. by Holt, Hansberger, Chong, & Boehm-Davis, 2002, and Schoppek, 2002), and that, to a certain extent, the general part can be improved without changing the domain specific parts.

The activities demanded from the pilots range from situation-specific decision making (e.g. deciding which mode to use for a specific leg[1]) to the execution of standard

[1] leg: the flight path between two waypoints

procedures (e.g. entering an altitude restriction into the flight management computer). To account for this variety of actions, ACT-Fly's control structure was based on a goal stack limited to three levels with a clear division of responsibilities among the levels. Each of the levels is represented by a goal chunk of a specific type (chunk is the ACT-R term for a declarative memory element).

Level 1 is the bottom level. It can be characterized as the decision making level. At this level, rule-based decisions are made as to what goals are pursued and what methods are selected to accomplish these goals. Level 1 also serves as the goal manager for Level 2. Finally, Level 1 contains some basic problem solving productions. The goal chunk representing this level stores molar information about the situation, such as the phase of flight, the position of the aircraft in the flight plan, or the status of ATC clearances.

Level 2 can be characterized as the method level. It is the level of operation described by frameworks like GOMS. Similar to GOMS, our methods consist of operators, subgoals, and decision steps. Level 2 can execute hierarchical methods of virtually any depth on one level. This is possible because subgoals are not stacked on top of each other. Rather, superordinate goals are released to memory and retrieved later on. Storage and retrieval of goals are handled on Level 1. This design has several advantages. First, the concept of a goal stack has been criticized for providing unrealistically perfect memory for goals (Altmann & Trafton, 2002). In ACT-Fly, goals do not simply appear on top of the goal stack once the previous goal has been popped, but must be retrieved from memory - a process that can fail and can predict certain types of errors. Second, as control is returned to Level 1 after the execution of each submethod, the course of action can be corrected during the execution of a long, nested method. With a more traditional goal stack, the system would be “blocked” for the time such a method is executed. Although Altmann & Trafton (2002) also argue that suspended goals are subject to decay, their model does not explicitly allow a mechanism for interrupting the execution of a current task to perform an alternate task. Thus, our solution makes the model more flexible and ready to handle interruptions.

Most steps within methods are represented as declarative chunks linked through associations. Thus, the retrieval of the next step is cued by the current method and the previous step, but is not constrained symbolically. That enables the model to produce errors of omission and of commission in the execution of methods. Another advantage of the associative linking of steps is that methods are learned “by doing”, using ACT-R’s associative learning mechanism. This representation of procedures is not standard in ACT-R modeling, but there is some evidence supporting this assumption. For example, Byrne and Bovair (1997) explained their findings on the “post-completion error” with a spread of activation from one step to the next; successful models of associative sequence learning also exist (Altmann, 2000; Lebiere & Wallach, 2001). Using the execution of one step to cue execution of the next step through associations between the two steps corresponds best to an intermediate state of proficiency as it is expressed by Rasmussen’s (1986) rule-based behavior. However, that is not the only mechanism available for representing procedures in our model; it was also possible to integrate completely proceduralized sections into a method to be executed as a whole. The ACT-Fly methods serve different functions. There are methods that perform input operations to the automation, methods that do mental calculations with flight parameters to support decisions, and methods that return classifications of the current situation to maintain situation awareness.

We assume three basic types of operators, which are part of every step. (The separation of steps and operators allows the occurrence of the same operator in multiple steps.) Internal operators perform memory operations, comparisons, or mental calculations; they are represented as single production rule. External operators perform actions such as pressing buttons or dialing values. Perceptual operators provide representations of the environment such as current values read from displays or ATC clearances.

Level 3 represents the interface between central cognition and peripheral systems. Since ACT-Fly does not simulate perceptual or motor processes in detail, input-output operations are modeled on an abstract level. When the model requests information from the environment, a specialized chunk is pushed on Level 3, completed with the requested information (through the TCP/IP-socket connection with the flight simulator), and the results are transferred to the goal chunk of Level 2. Similar steps are performed for motor commands. After being cleared from the goal stack, the I-O-chunks remain as episodic traces in memory.

Our solution of a goal stack with three levels has similarities with the "memory for goals" approach by Altmann & Trafton (2002) (which in the meantime has become a standard modeling technique in ACT-R). In particular, goal processing at Level 2 complies with Altmann & Trafton's approach. However, assuming separate levels for problem solving, procedural processing, and perceptual-motor processing made modeling of our complex task much easier. We believe it reasonable to assume that goals corresponding to different levels do not interfere with each other when a pending goal is retrieved from memory. Moreover, the goals of Level 3 never stay there for longer short periods of time, making it plausible that the pending goal on Level 2 is virtually always retrieved correctly.

3. ACT-Fly's predictions and performance

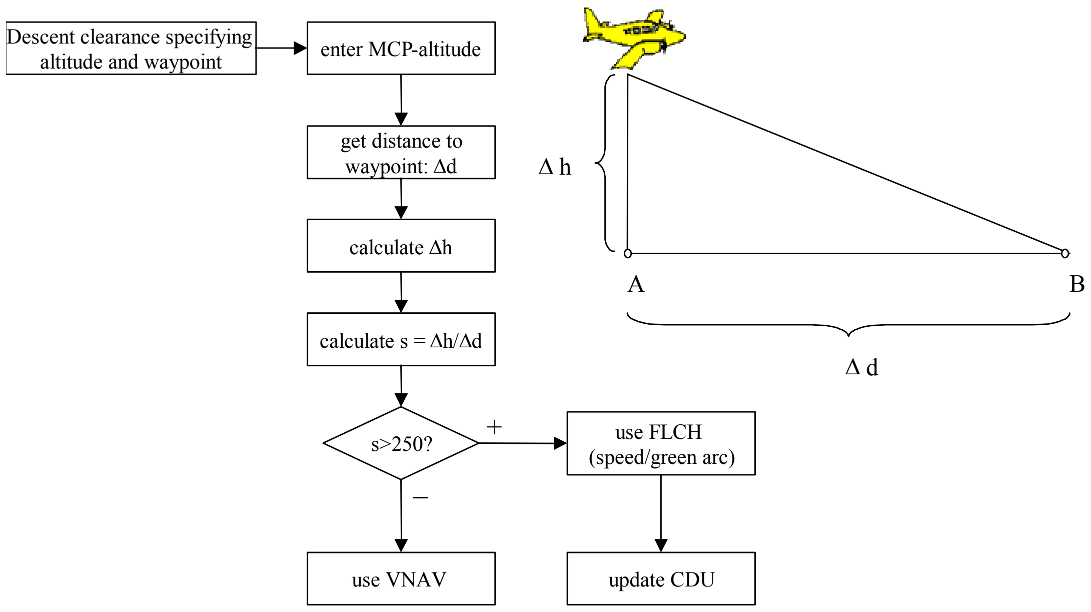
Descent clearance specifying altitude and waypoint
enter MCP-altitude
get distance to waypoint: Δd
calculate Δh
calculate s = Δh/Δd
s>250?
+
use FLCH (speed/green arc)
−
use VNAV
update CDU
Δ h
A
B
Δ d

Figure 2: Flow chart of the central method of ACT-Fly that decides what mode to use for a given descent leg. The boxes represent goals for which ACT-Fly has appropriate methods.

Central to the behavior of the ACT-Fly model is a method that decides what mode of the flight management system to use for a given leg. A flow chart of this method is depicted in Figure 2. This was an area where it became clear that existing models in current architectures could not provide guidance about how to represent this task. For our model, we interviewed several subject matter experts and discovered several alternative methods that could have been used. From these reports, we abstracted the method shown in Figure 1, which is more explicit and simpler than any individual model described by an expert. ATC-clearances and situations in which a waypoint gets close can trigger execution of this method. Our goal was to determine how well the use of this single abstracted model could account for typical pilot behavior.

Performance of the ACT-Fly model was assessed by flying informal as well as standardized simulated scenarios. In these scenarios, the model ran on one computer, the flight simulator on a second computer. Most of the communication between the programs was handled through an TCP/IP socket connection. Only a few values from the flight simulator had to be entered manually on request of the model; ATC clearances were entered manually, too. Most scenarios started with the simulated aircraft being close to the "top of descent point" calculated by the flight management system. ACT-Fly recognizes the situation on its own initiative. Time synchronization was accomplished by letting ACT-Fly, which usually runs faster than the flight simulator, wait for the simulator when its simulated time was more than five seconds ahead of real time.

Figure 3 shows the results of simulations of two descent scenarios. For each scenario, we ran six simulations. As can be seen from the mean deviations around each data point, there was little variability in model performance. Generally, the model flew the desired paths precisely. In the scenario without ACT-clearances, the model selected the VNAV mode for most of the legs, except for the leg between BOLDR and MENLO, where FLCH was used. In the second scenario, where a clearance was issued after the top-of-descent point (T/D) had been passed (making VNAV an inappropriate mode choice), the model selected the semiautomatic FLCH mode right after the clearance.

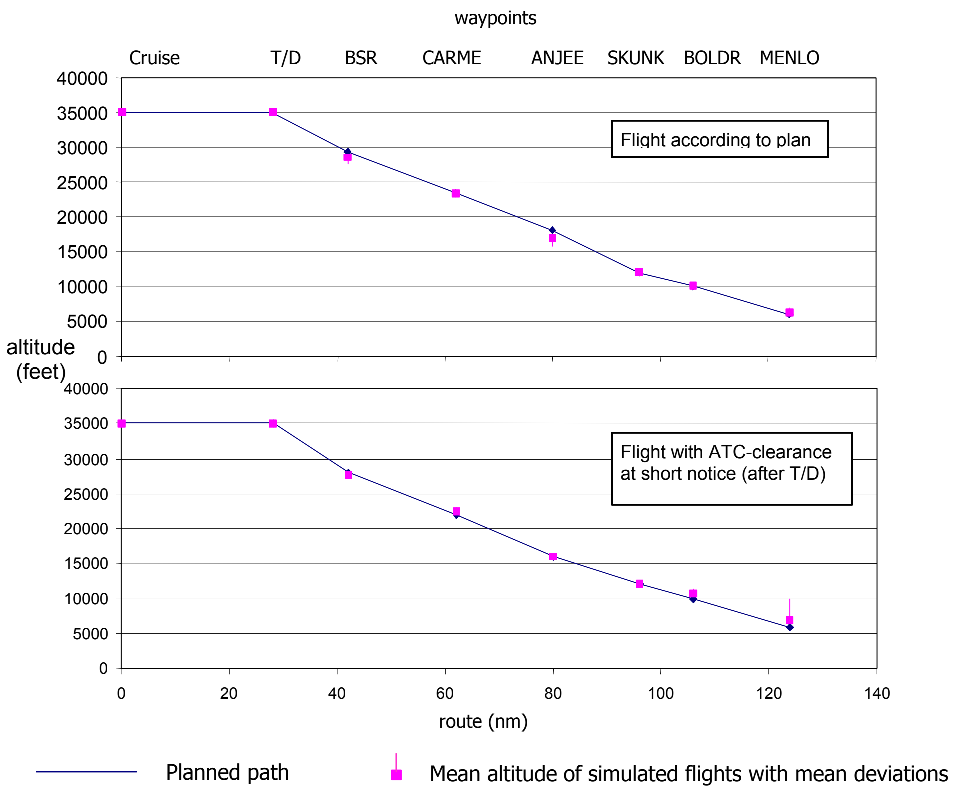

Figure 3: Flight paths of simulated flights in two scenarios (six simulations in each scenario). The upper panel shows data from a scenario without ATC clearances, the lower panel data from a scenario with the clearance "Cross ANJEE at 16000".

In all twelve simulations, each consisting of approximately 16 simulated minutes and 1300 production firing cycles, four errors of omission occurred. One of these caused the large deviation at waypoint MENLO visible in the lower panel of Figure 2, because the model could not recover from the error and failed to update the altitudes in the mode control panel. Although the single complete failure to recover out of four errors of omission is too low for a stable estimation, we think that this proportion is much worse than what is found in humans (Reason, 1990). Observations from our informal testing support this view.

The model committed a number of other errors; these impaired performance more gradually. For example, there were three cases where the model failed to resume a deferred action (intention). In general, error recovery was best when the method in which the error occurred was directly triggered by the identification of a relatively stable situation (initiated by Level 1 of the control structure). Error recovery was worse when the affected method was triggered by another method (initiated by Level 2), and when it was triggered by a transient situation.

To compare performance of real pilots with the model, we conducted a study with five commercial pilots, certified to fly the Boeing 777 aircraft (which has a flight management system very similar to the Boeing 747-400). Four of the pilots served as first officers on the 777; one served as a captain. The participants flew two scenarios with the aforementioned PS1 flight simulator. Flight performance, prompted recall of flight parameters, and eye-tracking data were recorded. (For results of the eye-tracking aspect of this work, see Diez, Boehm-Davis, Holt, Pinney, Hansberger, & Schoppek, 2001). As the results in Table 2 indicate, the pilots' performance was unexpectedly poor. In the first scenario, all pilots violated the altitude restriction of 11000 feet at waypoint PANZE. The altitude restriction was much better met in the second scenario. However, four of five pilots initiated an early descent, which was unnecessary in Scenario 2 (and increases fuel consumption). The frequent mode changes commanded by the pilots are also remarkable.

	Scenario 1: Sea Isle approach			Scenario 2: Big Sur approach		
	early descent?	mode (sequence)	crosses PANZE at	early descent?	mode (sequence)	crosses ANJEE at
Good solution	**yes**	**V/S or FLCH**	**11,000**	**no**	**VNAV**	**16,000**
Subject 1	no	V/S	19,500	yes	V/S, VNAV V/S, VNAV	16,000
Subject 2	yes	VNAV	16,200	yes	VNAV	16,000
Subject 3	yes	FLCH, VNAV, FLCH, VNAV	13,000	yes	VNAV, FLCH, V/S, VNAV, FLCH, HOLD	16,200
Subject 4	no	VNAV	15,700	no	VNAV	16,000
Subject 5	yes	V/S, FLCH, VNAV, FLCH	11,000	yes	VNAV	16,000

Although this poor performance is troubling, we believe that it does not closely reflect the pilots' line performance and should be attributed, at least in part, to the setting of our study: The pilots reported that the simulation functioned reasonably well in simulating the responses of a real aircraft; however, there were two important differences between the simulated situation and a real cockpit. Our participants had to accomplish the flying task alone, whereas in reality, there is a distinct division of labor between two crew members. Second, the interface required keyboard and mouse entries rather than the use of knobs and dials. The combination of these factors likely influenced pilot performance.

4. Discussion

The process of designing the ACT-Fly model revealed a number of phenomena that appear to be common, or even typical for complex tasks, but for which there are no standard solutions in extant ACT-R models. We developed ad hoc solutions with no claim that they represent theories. For all these phenomena, theoretical solutions that account for a broad range of documented effects would advance both the efficiency of modeling complex tasks, and the scope of the ACT-R framework.

Deferring actions: In dynamic systems, effects of actions can unfold slowly. In these cases, checking the success of an action must be deferred, while other things are being done. For example, in VNAV mode, after entering a new reference value for altitude, the pilot must wait until the next waypoint has been passed before deciding how to continue. The problem for modeling is how the deferred intention is remembered on time. To simulate intention memory, we used a mechanism that inhibits the chunks representing the deferred actions for a certain time. After each completion of a method, the model tried to retrieve deferred goals from memory, which fails as long as these are inhibited. After the inhibition has terminated, there is a chance to retrieve the chunks, but only for a limited time, after which they may be forgotten.

Expectations: One undesired type of event that can lead to errors is "automation surprise" (Sarter & Woods, 1995). It occurs when the behavior of the automation does not match the pilots' expectation (e.g. when the aircraft suddenly levels off when the pilots expect it to continue the descent). We included two mechanisms to model expectations. One involves the retrieval of a chunk that represents a situation-action-situation sequence. The other models expectations through production rules that respond to outcomes not associated with specific behavior. The first mechanism requires a controlled action demanding central resources; the second mechanism does not really form expectations to be compared with the actual outcomes. Rather, it responds to unfamiliar situations. We think that both solutions do not sufficiently reflect the character of the process of forming expectations as an autonomous background mechanism, but since input and output can be connected exclusively through the central production selection process in ACT-R, such mechanisms cannot be implemented in principle. This strong claim of the ACT-R architecture might be reconsidered in future developments.

Estimation of time: We identified several processes that rely on estimation of time: the resumption of intentions, the periodic repetition of monitoring behaviors, and the decision to try a different method when one method fails after some repeated applications. We simulated time estimation simply by using the numerical output of the time function provided by ACT-R. Phenomena like the dependence of subjective time on workload could not be produced with this solution (Leuchter & Urbas, 2003). If one assumes internal timers whose signals are processed depending on, e.g., workload, there is again the question if this is done by the central production selection mechanism or by some other background processes.

Judging the results of our modeling effort, we can state that some of the objectives have been met, some have not. Following is a list of strengths of the model.

1. Errors of omission and errors of commission could be produced. The underlying dynamic is such that errors of omission are more likely under high working memory load (Schoppek, 2000).

2. We showed that a large fraction of (even flexible) behavior in the use of the flight management system can be explained by (predefined) procedures in the absence of declarative memory about the system.

3. The model can be used to test the effectiveness and flexibility of alternative procedures. In our model, the ACT-Fly method we developed for deciding what mode to use for a given descent profile proved effective in a range of scenarios. It would have been easy to develop and test alternative methods for making this decision by inserting a new method in its place.

4. The separation of general and domain specific parts allows for improvements in the general part with few changes to the specific part of the model. Further,

the procedural aspects of new tasks can be modeled quickly without needing to work at the general part.

5. We discovered mechanisms common for supervisory control tasks that are in principle difficult to implement in ACT-R (see above).

On the other hand, the model has a number of shortcomings:

1. Once the model has produced an error of omission (or commission), it is – unlike humans – not good at recovering from the error. In some simulations, the whole subsequent behavior of the model was disturbed by one error.

2. We did not succeed in modeling continuous mode awareness and automation surprise. The first is due to the strictly frame-oriented structure of the goal chunk in ACT-R (which corresponds largely to working memory in other frameworks). Once a piece of information is stored in one of the slots of the goal chunk, it can reliably be retrieved as long as the goal is active, or unless the slot is changed deliberately. The alternative of storing the information about the current mode in declarative memory rather than in the goal chunk does not produce the desired effects either, because in ACT-R, declarative memory can only be accessed in a controlled manner returning "conscious" feedback about the result of the retrieval attempt. Thus, we did not find a satisfactory solution for modeling inadvertent forgetting of mode information.

3. The model generally has poor situation awareness, supposedly because it has too little general knowledge about flying. We believe that real pilots classify (and identify) situations much more by similarity to known situations than by rules, as ACT-Fly does. It is possible that the procedure we developed for the descent decision is tailored too much to the support of decision-making and too little to the support of situation awareness. To test procedures more realistically, their implications for decisions and situation awareness should be judged.

4. We were able to build this model over a period of 10 months. This is a fairly long interval. However, it is shorter than what likely would have been required had we not been working with an existing cognitive architecture.

Most aspects of the task for which ACT-R did not provide enough constraints followed from the task's dynamic and the requirement to interleave subtasks during long time intervals. Although we were able to identify potential solutions to the identified problems, they represent only crude approximations to long-term theories that could be reused. Nonetheless, they can be regarded as hints how the scope of ACT-R could be extended to reasoning and action in more complex and dynamic environments.

5. Acknowledgments

This research has been supported by grants NAG 2-1289 from the NASA and 99-G-010 from the FAA. We thank Melanie Diez for her help in developing the scenarios for the simulator and for collecting empirical data from pilots.

6. References

Altmann, E. M. (2000). Memory in chains: Modeling primacy and recency effects in memory for order. *Proceedings of the 22nd annual conference of the Cognitive Science Society* (pp. 31-36). Hillsdale, NJ: Erlbaum.

Altmann, E. M. & Trafton, J. G. (2002). Memory for goals: An activation-based model. *Cognitive Science, 26,* 39-83.

Anderson J.R. (1982). Acquisition of cognitive skill. *Psychological Review, 89,* 369 - 406.

Anderson, J.R., & Lebiere, C. (1998). *Atomic components of thought.* Mahwah, NJ: Erlbaum.

Anderson, J.R., Bothell, D., Lebiere, C. & Matessa, M. (1998). An integrated theory of list memory. *Journal of Memory and Language, 38,* 341 -380.

Anderson, J.R. & Reder, L.M. (1999). The fan effect: New Results and new theories. *Journal of Experimental Psychology: General, 128,* 186 -197.

Byrne M.D., & Bovair S. (1997). A working memory model of a common procedural error. *Cognitive Science, 21,* 31 - 61.

Card, S.K., Moran, T.P. & Newell, A. (1983). *The Psychology of Human - Computer Interaction.* Hillsdale, NJ: Erlbaum.

Diez, M., Boehm-Davis, D.A., Holt, R.W., Pinney, M.E., Hansberger, J.T., Schoppek, W. (2001). *Tracking pilot interactions with flight management systems through eye movements*. Paper presented at the 11th International Symposium on Aviation Psychology, Columbus, Ohio. [http://hfac.gmu.edu/~mdiez/MelPubs/Tracking%20pilot%20interactions.PDF]

Holt, R. W., Hansberger, J., Chong, R., & Boehm-Davis, D. A. (2002). Modeling aviation crew interaction using a cognitive architecture. In *Proceedings of the 24th Annual Meeting of the Cognitive Science Society*, Fairfax, VA.

Kieras, D. (1997). A guide to GOMS model usability evaluation using NGOMSL. In M. Helander, T. K. Landauer, & P. Prabhu (Eds.), *Handbook of Human-Computer Interaction*, (Second ed., pp. 733-766). New York: Elsevier.

Lebiere C., & Wallach D. (2001). Sequence learning in the ACT-R cognitive architecture: Empirical analysis of a hybrid model. In C. L. Giles R. Sun (Eds.), *Sequence Learning: Paradigms, Algorithms, and Applications Lecture Notes in Computer Science* (pp. 188 - 212). Heidelberg: Springer.

Lee F.J., & Anderson J.R. (2001). Does learning of a complex task have to be complex? A study in learning decomposition. *Cognitive Psychology, 42,* 267 - 316.

Leuchter, S. & Urbas, L. (2003). Modeling dynamics and timing for operating human-machine systems. In F. Detje, D. Dörner, & H. Schaub (Eds.), *The Logic of Cognitive Systems. Proceedings of the Fifth International Conference on Cognitive Modeling* (pp. 279-280). Bamberg: Universitäts-Verlag.

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

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

Rosenbloom, P. S., Laird, J. E., & Newell, A. (Eds.). (1992). *The SOAR Papers: Research on Integrated Intelligence*. Cambridge, MA: MIT Press.

Salvucci, D. D. (2001). Predicting the effects of in-car interface use on driver performance: An integrated model approach. *International Journal of Human-Computer Studies*, *55*, 85 - 107.

Sarter N.B., & Woods D.D. (1992). Pilot interaction with cockpit automation I: Operational experiences with the flight management system. *International Journal of Aviation Psychology, 2,* 1 - 28.

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

Schoppek, W. (2000). Learning and performance of sequential action under different workload conditions: An ACT-R model. In N. Taatgen; J. Aasman (eds.), *Proceedings of the Third International Conference on Cognitive Modeling* (pp. 295 – 296). Veenendaal: Universal Press.

Schoppek W. (2002). Examples, rules, and strategies in the control of dynamic systems. *Cognitive Science Quarterly, 2,* 63 - 92.

Taatgen N.A. (1999). A model of learning task-specific knowledge for a new task. *Proceedings of the 21th annual conference of the cognitive science society* (pp. 730 - 735). Mahwah, NJ: Erlbaum.

Taatgen N.A., & Lee F.J. (2003). Production compilation: A simple mechanism to model complex skill acquisition. *Human Factors, 45,* 61 - 76.