Modeling Behavior in Complex and Dynamic Situations - The Example of Flying an Automated Aircraft

Wolfgang Schoppek, Robert W. Holt, Melanie S. Diez, & Deborah A. Boehm-Davis
University of Bayreuth, Germany and George Mason University, Fairfax, VA

In basic research, cognitive modeling has proven a valuable methodology for explicating theoretical assumptions, testing their dynamic interactions, and exploring the scope of theories. Cognitive architectures such as ACT-R or Soar provide a common basis for different models and enhance communication and exchange of solutions. In applied contexts too, 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, as real tasks typically require more knowledge and are more complex than laboratory tasks, content independent cognitive architectures do not sufficiently constrain the modeling of these tasks. We argue that - despite this problem - models of behavior in real world tasks should also be developed within established architectures. In doing so, it is important to specify what parts of the model are derived directly from the architecture and for what parts new solutions had to be developed. Thus, models benefit from the broad empirical confirmation of the architecture, which in turn benefits from the identification of domains where it needs to be extended. As an example for this approach, we present an ACT-R (Anderson & Lebiere, 1998) model that simulates the interaction between airline pilots and the flight management system.

Characteristics of the task

The task we modeled is flying down a simulated Boeing 747-400 from the end of the cruise phase to the initial approach fix using the automation of the aircraft. This shall be accomplished under a variety of conditions, such as different ATC clearances or descent profiles. There are two basic modes of automation that can be used for that task: A fully automated mode - called VNAV - where the autopilot receives most of the reference values from a preprogrammed flight plan; in a semiautomatic mode the reference values must be provided by the pilot.

If there are no last minute changes in the flight plan, VNAV is the preferred mode, because it optimizes the flight profile. However, if ATC requires quick changes of the flight plan, the pilot can respond more flexibly using the semiautomatic mode. In both basic modes, the behavior of automation and aircraft must be monitored and reference values must be provided in a timely manner.

When we compare the characteristics of the flying task with those of more typical ACT-R tasks, such as memorizing lists of words (Anderson et al., 1998) or discriminating previously learned statements from distracters (Anderson & Reder, 1999), we find differences in many dimensions. The time scale for the flying task ranges from minutes to hours - much longer than the seconds to minutes of typical ACT-R tasks. Unlike these tasks, the flying task takes place in a dynamic environment that changes rapidly and autonomously. An environment like that often requires revision of plans due to changes in the situation, interrupting ongoing activities due to unexpected events, and deferring actions, because opportunities have to be awaited. There are also differences in the goal structure. Typical ACT-R tasks can be accomplished with a well defined goal hierarchy consisting of one main goal and subordinate goals related in a means-ends fashion. Flying, in contrast, involves heterogeneous goals that may compete for limited resources. For example, the goal of watching the plane pass a critical waypoint competes with the goal of encoding a new ATC clearance. Finally, much more previous knowledge must be brought to the flying task than to a typical ACT-R task.

All these differences require the modeler to find new solutions that are not obvious in the architecture and cannot be derived from existing models that successfully predict behavior in less complex tasks.

ACT-Fly model

The theoretical background of the model is a combination of GOMS (Card, Moran & Newell, 1983) and ACT-R. We developed ACT-R representations of GOMS elements and a way of translating GOMS analyses to ACT-R code. With earlier versions of the model, we found that relying 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 situational 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 situational awareness, we introduced an additional level to the control structure that operates in a non-sequential, knowledge based manner.

The activities demanded from the pilots range from situation specific decision making (e.g. deciding which mode to use for a specific leg) to the execution of standard procedures (e.g. entering an altitude restriction into the FMC). To account for the variety of actions, ACT-Fly's control structure is based on a goal stack limited to three levels with a clear division of responsibilities among the levels.

Level 1 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

accomplished these goals. Also, level 1 serves as manager for level 2. Finally, level 1 contains some basic problem solving productions. The goal chunk (chunk is the ACT-R term for declarative memory element) 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 ACT clearances.

Level 2 can be characterized as the method level. It is the level of operating described by frameworks like GOMS. Similar to GOMS, our methods consist of operators, submethods, and selection rules. 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, but rather, superordinate goals are released to memory and retrieved later on. This design has several advantages. First, the concept of a goal stack has been criticized for providing unrealistically perfect memory for goals (Altmann & Trafton, 1999). 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 and nested method. With a more traditional goal stack, the system would be "blocked" for the time such a method is executed. Thus, our solution makes the model more flexible and ready to handle interruptions.

The steps of most 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 simulate 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 the associative learning mechanism of ACT-R. There is an Excel spreadsheet tool available that allows easy translation of NGOMSL analyses to ACT-R code. Methods of ACT-Fly 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 situational awareness.

Level 3 represents the interface between central cognition and peripheral systems. Since ACT-Fly does not model perceptual or motor processes, input-output operations are simulated 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 popped, the I-O-chunks remain as episodic traces in memory.

The design process of ACT-Fly revealed a number of problems that appeared to be common, or even typical for complex tasks, but for which there are no standard solutions in extant ACT-R models:

Deferring actions: In dynamic systems, effects of actions often unfold slowly. In these cases, checking the success of an action must be deferred, while in the meantime other things are done. The problem for modeling is how the deferred intention is remembered on time. To simulate intention memory we use a mechanism that inhibits the representations of deferred actions for a certain time.

Expectations: One undesired type of event that can lead to errors is the "automation surprise" (Sarter & Woods, 1995). It occurs when the behavior of the automation does not match the pilots' expectation. 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 implicitly through production rules that respond to "unexpected" outcomes.

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 another method when one method fails after some repeated applications. We simulated time estimation by using the time function provided by ACT-R.

To summarize: 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 our solutions to the identified problems are only crude approximations, 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.

Acknowledgments

This research has been supported by grants NAG 2-1289 from the NASA and 99-G-010 from the FAA.

References

Altmann, E. & Trafton, J.G. (1999). Memory of goals: An architectural perspective. *Proceedings of the twenty first annual meeting of the Cognitive Science Society*. (pp. 19-24). Hillsdale, NJ: Erlbaum.

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.

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

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.