Extending ACT-R for modeling dynamics and timing for operating human-machine systems

SANDRO LEUCHTER & LEON URBAS

MoDyS Research Group, Center of Human-Machine Systems,
Technische Universität Berlin

Keywords: cognitive modelling, dynamics, human-machine-systems, cognitive architecture, ACT-R

1. Introduction

An important current trend for the ACT-R community is to apply cognitive modelling to real world problems like HCI (Byrne 2001) or human machine interaction (Gluck 2002). Important developments of the architecture that are necessary to do so are PM for integration to the task environment and new ways of goal management. An area that still needs attention is timing for operating human-machine systems. With this contribution we want to promote the application of cognitive architectures for engineering applications in dynamic human-machine systems (HMS). In this section the notion of HMS and their dynamics are defined and the objective of this work is presented in detail.

1.1 Human-Machine Systems

Many machines (i.e. technical systems) are used by humans. The term human-machine system denotes not only systems in which at least one human operates a technical system, but emphasizes the interaction between human and machine. Typically the technical system is fairly complex and shows a continuous dynamic behaviour that is influenced by interventions of its operator. While former research on HMS has focused on physical and ergonomic characteristics of the interaction for optimizing construction and force feedback properties, today the main topic is estimating consequences of automation. Since technical systems are getting more complex often cognition, memory span, and mental models are being of concern. Some typical examples of systems that are analysed and developed from an HMS viewpoint are in high risk environments like the aviation domain, energy and power management (nuclear power plant), and chemical process operation.

1.2 Objective

The objective for modeling behavior of HMS operators is to facilitate simulation based design support, training, and deployment in assistive technology.

1.2.1 Simulation based design support

In the development process of HMS' it is cost efficient to detect design flaws as early as possible. Thus tests are conducted with prototypes or even mockups instead of finished products. The use of simulations of the technical system is common practice in testing e.g. in automobile industry. Additionally simulations of cognitive capabilities of the potential user are effective for questions about human reliability (Amalberti 2002). They can be efficient in large scale multi-user scenarios (computer generated forces: Jones et al 1999) or when a single prototype test is very expensive (e.g. aircraft cockpit automation: Lüdtke 2002).

1.2.2 Training

Insights from simulations of different strategies in cognitive models can be measures for "difficulty of learning each strategy, efficiency of using each strategy once learned, generality of each strategy to the range of […] problems, retention of the strategies, and transfer" (Rittle-Johnson & Koedinger 2001). Although stated for simple arithmetic tasks these measures can be used to decide which strategy to train for operating HMS.

1.2.3 Deployment in assistive technology

Besides using simulations of cognitive processes as knowledge based support system (perfect cognitive models making no errors are *expert systems*) realistic (error making) cognitive models have the potential to be deployed in adaptive automation systems (Parasuraman et al 2000). Such systems adapt their behavior to external conditions normally detected in the technical system or its environment. Taking not only the technical or environmental state but also that of the operator into account leads to a more effective task allocation between automation and human operator. Realistic cognitive models running parallel to the HMS can be used to predict current or future operator states. A famous example for this kind of automation are intelligent tutoring systems (e.g. Leuchter & Urbas 2002).

The rationale of the use of the cognitive architecture ACT-R is to make the modeling process more efficient by introducing a priori constraints on memory and processing.

1.3 Dynamics

The need for timing arises from the dynamics of the operated system. Thus for this report the most important property of an HMS is its dynamics. Many approaches for conceptions on dynamics of HMS exist. Often they are directed by control theory or complex problem solving. Since they do not fully fit in modeling timing in this paper a scheme is proposed that is derived from experience gained modeling cognitive processes of air traffic controllers (Niessen et al. 1998) and process control.

This approach is presented object-oriented. Figure 1 shows the framework as a UML diagram (unified modeling language, common used in software engineering).

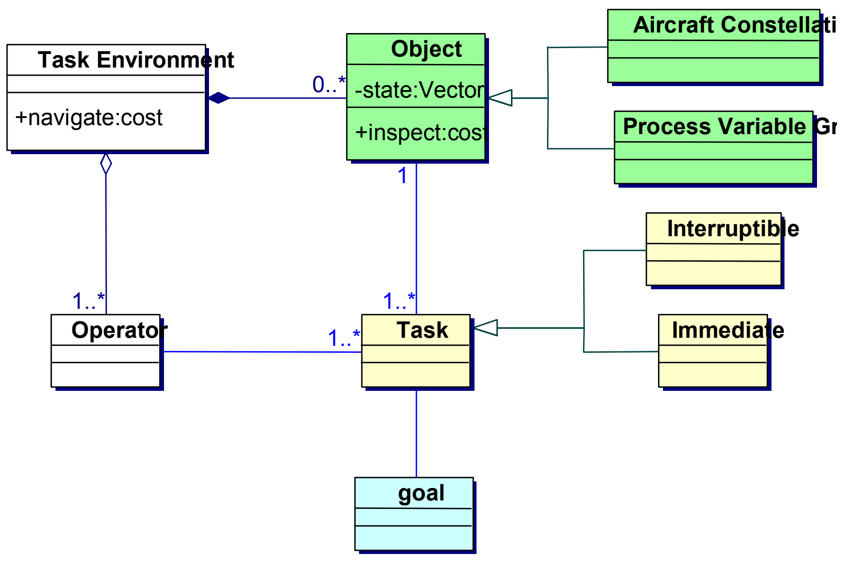

Figure 1: Framework for dynamics.

The diagram shows three columns. The left one is the defines the system boundary: The environment and the modeled subject(s). The middle column shows the concept itself. The right column shows concretizations. Above: in two domains (green): en-route air traffic control (ATC) and process control (PC e.g. chemical plant), below: concretizations of tasks.

The task environment consists of a number of objects. Example for objects are air-craft or their constellations (ATC) and process variables such as temperature or pres-sure (PC). One or more operators are responsible. An operator is assigned one or more tasks to fulfill. Every task is associated with exactly one object. A task is con-sequently associated with a goal. Tasks or more exact fulfillment of a task can either interruptible by other tasks or has to be immediately worked off.

Dynamics results first from new objects or tasks appearing in the situation. But for most tasks in HMS the state of the objects is also needed. It can be acquired by the HMS interface (often a screen). To avoid confusion not all information is normally displayed. But for the execution of a certain task an unusual information can be nee-ded. Then the object has to be inspected and that information gathered (e.g. trans-ponder code on ATC screen). Some interfaces are designed in several screens (e.g. control room displays for PC) so that the operator has possibly to navigate in the task environment to the needed information's display. The second reason for dynamics is the changing of objects state. Change can be initiated from actions of the operator and from environmental interference. Both result in a more or less delayed change behavior.

The treatment of this framework is that a task environment has a certain number of objects at a time and following a number of tasks/goals that are valid and have to be fulfilled at any one time (homeostatic and parallel goals, Aasman 1995). The charac-teristics is the amount of them (interruptible separated from immediate), the change rate and expenses for navigation/inspection. One application of this framework is a comparison between different domains/tasks to estimate the need to use working memory in favor for external memory (HMS interface screen) on the basis of inspec-tion/navigation cost and the number of continuous tasks.

This framework of describing dynamics is used in the following section to discriminate between certain task environments.

2. Task Environments

In this section the dynamics of several interactive task environments is discussed: car driving, en-route air traffic control, human-computer interaction, aircraft piloting, interactive computer games, and process control in chemical plants. The consequences for scheduling of subtasks are deduced.

Car driving: Although driving is not a top-down activity it is composed of several subtasks that are all active at the same time ("homeostatic", Aasmann 1995). Cnossen (2000, p. 40) and Salvucci (2001) enumerate subtasks of driving as: speed control, steering, visual search, and distance keeping. Car driving is a highly dynamic task because the state of the vehicle and its environment changes constantly possibly without prior warning.

Apart from modern automation like adaptive cruise control there is not much need for a memory for situation awareness because the situation can be easily directly perceived at environment (objects are primarily other cars) and instruments.

En-route air traffic control: Controllers monitor the movement of aircraft on radar screens. Although the traffic is planned in advance weather conditions or other unexpected events influence plan achievement. Thus controllers have to command pilots for other direction, speed or altitude to avoid dangerous approximation. The situation is dynamic: New objects (aircraft) enter the sector, the state of the objects (position, altitude, speed, direction) changes normally upon request of the controller. The change rate of tasks is rather low, because new aircraft enter a controlled area about every some minutes. Nevertheless it is necessary to monitor aircraft because pilots could not exactly perform commands or deviate from their routes. Controllers need a representation of the current situation in order to achieve anticipation, conflict resolution and monitoring (Niessen et al. 1998).

Human-computer interaction: HCI does not necessary include the notion of dynamics: Objects that are observed are GUI-elements that display values. Depending on the model that is displayed they are changing and have to be monitored. Although many dynamic human-machine-systems have computer mediated interfaces or can be used in PC based simulations there are no direct results for cognitive models of HCI.

Aircraft piloting: Through the introduction of *fly by wire* concepts and automation technology in the modern *glass cockpit* pilots mostly fulfill a supervisory control task. Objects are the measure instruments and displays. The subtasks are precisely in hierarchical standard operating procedures defined. A major part of all human factors errors in aviation results in lack of situation awareness in that the current mode of the automated system is misinterpreted (e.g. Lüdtke et al. 2002). An interpretation on the basis of the proposed dynamic modeling framework is that procedures have to be treated as objects which was the way how an existing ACT-R pilot model was realized (Schoppek et al. 2000).

Interactive computer games: Similar to car driving new objects in interactive computer games can occur and change their state without the player's interaction. In first person shooter games (Laird 2001) the most important feature is the position of objects (enemies or other characters). Since navigation is achieved through movement in virtual space inspection cost for objects are high. Because of this and to make deliberative behavior possible it is necessary to maintain a mental representation of the virtual world.

Process control in chemical plants: In contrast to computer games, air traffic control and car driving the super structure of the environment does in general not change, i.e. the pure number of objects does normally not change. Nevertheless, due to start up and shut down of single unit operations and the possibility to change the interconnection between the unit operations the relation between the objects and the resulting dynamic behavior of the process is subject to change. Furthermore the course of variables in time is most often not predictable by linear extrapolation. In consequence a mental representation of the current relations between the objects and a image of the history of the process is necessary for the supervisory control tasks.

Thus operators need not only a mental representation of the objects' values but also of their dynamical features as their state. Operators typically have several displays in the control room and one monitor can display several different views. They use structural presentations of the chemical plant where the values of process variables are shown and trend pictures of the development of a variable over time.

2.1 Requirements for Scheduling

There are different characteristics of dynamic task environments that demand different processing of the dynamics. Within the proposed framework dynamics is handled through composition of goals, the scheduling of their execution, memory for objects' state (situation awareness), and update of objects' state (which can be seen as one of the goals).

Since providing a memory with features such as adaptive decay and partial matching for a situation's objects' state can be easily achieved with modern production systems like ACT-R (Anderson & Lebiere 1998) we focus on goal scheduling.

The task characteristics of HMS environments is either more reactive or more deliberative. In reactive task environments objects and associated tasks are perceived, new goals are generated, and executed. In the deliberative case subtasks can be postponed and subtasks have different urgency and priority. A memory for postponed tasks and strategies for scheduling them is needed.

The resulting multiple tasking is not the same as that of Lee & Taatgen (2002). They report on multi-tasking in the time gaps between action in asynchronous cognitive subsystems. Also scheduling as a task itself (e.g. Nellen 2002) is not the scope of this treatment. The time resolution is much less fine than in the task described by Gray et al. (2000).

For **car-driving** scheduling of tasks/goals does not need sophisticated algorithms because they are equally important. If secondary tasks like cell-phone dialing are added breakpoints between the tasks have to be defined (Salvucci 2001).

In **en-route air traffic control** controllers have too limited resources to monitor every aircraft on the screen. Thus they have to schedule their updating sequence, anticipating, and conflict resolution. Niessen et al. (1998) suggest an algorithm that takes inferred importance and two timing measures into account: Every object stores a timestamp when it was last updated, and during anticipation a duration until an event (e.g. time to collision) is computed. Other examples for cognitive models with multitasking exist (AMBR: Gluck & Young 2001, and Lee & Taatgen 2002)

Most **HCI** problems can be formalized in hierarchical task analysis models like GOMS. Thus there is no need to schedule between subtasks. The situation changes when there are several primary tasks (e.g. Salvucci 2001) or when the displays show the state of a dynamic human-machine-system (e.g. air traffic control: Freed 1998)

In **aircraft piloting** there are procedures to be executed. Multi-tasking and scheduling are only needed on a very top level (sequencing of procedures: starting, following way points, initiating landing). One additional requirement is the handling of asynchronous air traffic control commands.

Interactive computer games: A multitude of parallel goals exist that have to be examined and according to the current situation postponed or pre-drawn. To do so the player has to take both his or her memory and the perceived situation into account.

Process control in chemical plants: Depending on the inferred situation changes of goals have to be made. A major concern is to be able to suspend and to resume tasks. An operator model has to store the points where operation on one task was left and another was resumed.

2.1.1 Timed Conditions

If cognitive models are applied in engineering it is usually the aim to predict frequencies of erroneous production selection or memory slips or execution or learning durations for the whole task (e.g. within the GOMS framework). But there are also errors according timing and thus the need not only to model adaptive sequences of production selection but also the use of time and duration in conditions of productions.

An example for such a modeling demand is in process control: Sometimes a task has to be abandoned when too much time has elapsed. Thus duration has to be recalled in productions' conditions. To achieve this the system's real time would have to be retrieved and stretched or compressed according to the current workload.

2.1.2 Objects Getting More Important Over Time

In (en-route) air traffic control scheduling depends on timing: It is important to update the state (mostly position and altitude) of the objects as often as possible. But due to the other subtasks aircraft can only be monitored from time to time. But the need to update an objects' features in the mental representation gets bigger the longer the object has not been modified. Thus decay of activation depending on accesses to chunks is a contrary concept.

Such chunks representing such special elements of the situation under supervisory control have to be used in a certain way: Special productions have to refresh their activation depending on the current need to update it. Niessen et al. achieve this behavior through direct manipulation of activation parameters from outside ACT-R in the production-cycle-hook.

3. Conclusions: Extensions to the Architecture

On the basis of a brief review of this tasks and their requirements for scheduling and multi-tasking some needs for ACT-R can be drafted to fit it to modeling operator cognition in complex dynamic human-machine systems. We are currently implementing these extensions for modeling process control of chemical plants.

3.1 Linking and Embedding to Task Environment

Although ACT-R/PM made it possible to connect a model with a task environment there are problems for engineering: Normally there exists a big simulation or an API not accessible from LISP or making it hard to create a GUI within the LISP process. While one can cope with this restriction (e.g. Ritter et al. 2002) it would be efficient to embed an ACT-R model into a high-level framework for the "normal" control of a system and only execute a specialized ACT-R sub model for questions like memory errors from there. This would help controlling a real world situation with a simple outer model (without ACT-R) and only pay attention to special situations for that a more precise ACT-R model would be built.

Inside the ACT-R model there were less need for multi-tasking and scheduling and additionally communication with the task environment could be achieved through appropriate instantiation of chunks in the model during its start-up.

3.2 Recall of Duration

A new function for recalling duration information has to be added. Setting a named reference point that is stored as chunks in the working memory and thus can be forgotten or confused allows for retrieving elapsed time since setting it.

Perception of time is depended to the workload and the "mode": If concentrating on duration measurement high workload leads to underestimating, if recall is retrospective high workload leads to overestimating elapsed duration. Recall has to be possible in both modes and must be stretched or compressed according to subgoaling.

3.3 Scheduling

Tasks are to be represented and executed as chunks from a ACT-R "middleware" such as ACT-GOMS (Schoppek et al. 2000). But in contrast to ACT-GOMS it must include a scheduler for subtasks. They are to be marked interruptible and immediate during modeling time. Priority and urgency like in PDL could further guide the scheduling process.

An additional requirement in chemical plant control is that some tasks may not be carried out in parallel with others and that there are other dependencies possible. But a scheduler should not take also such information into account but it had to be modeled explicitly because this is an important source of control errors.

4. Acknowledgments

This work has been supported by Volkswagen Foundation in the program "Junior Research Groups at German Universities".

This report is an extended version of a presentation given at the Fifth International Conference on Cognitive Modeling (ICCM) in Bamberg, Germany on April 10 - 12, 2003.

5. References

Aasmann, J. (1995). *Modelling Driver Behaviour In Soar.* Leidschendam: KPN Research.

Amalberti, R. (2002). Use and Misuse of Safety Models in Design. In F. Grandoni (Ed.), *Dependable Computing EDCC-4. 4th European Dependable Computing Conference, Toulouse, France, October 23-25, 2002. Proceedings* (p. 1). Berlin: Springer-Verlag (LNCS; 2485).

Anderson, J. R., & Lebiere, C. (1998). *Atomic Components of Thought*. Hillsdale, N.J.: Erlbaum.

Byrne, M. D. (2001). ACT-R /PM and menu selection: applying a cognitive architecture to HCI. *International Journal Human-Computer Studies*, *55* 1-44.

Cnossen, F. (2000). *Adaptive strategies and goal management in car driving.* Proefschrift Rijksuniversiteit Groningen. Retrieved Oct. 30, 2002 from http://www.ub.rug.nl/eldoc/dis/ppsw/f.cnossen/thesis.pdf.

Freed, M. A. (1998). *Simulating Human Performance in Complex, Dynamic Environments*. Evanston, Illinois: Phil. Diss., Northwestern University. Retrieved Oct. 30, 2002 from http://www.andrew.cmu.edu/~bj07/apex/docs/dissertation/dissertation.doc.

Gluck, K. A. (2002). *Proceedings of the Winter Workshop on ACT-R Models of Human System Interaction.* Retrieved Oct. 29, 2002 from http://www.dtic.mil/AFRL/afrl.html.

Gluck, K. A., & Young, M. J. (2001). The AMBR Model Comparison Project: Multitasking, the Icarus Federation, and Concept Learning. In J. D. Moore, & K. Stenning (Eds.), *Proceedings of the 23rd Annual Conference of the Cognitive Science Society, August 1-4, 2001, Edinburgh, Scotland*. Mahwah, NJ: Lawrence Erlbaum.

Gray, W. D., Schoelles, M. J., & Fu, W.-T. (2000). Modeling a continuous dynamic task. In N. Taatgen & J. Aasman (Eds.), *Proceedings of the Third International Conference on Cognitive Modeling* (pp. 158-168). Veenendal, NL: Universal Press.

Jones, R. M., Laird, J. E., Nielsen, P. E., Coulter, K. J., Kenny, P., & Koss, F. V. (1999). Automated Intelligent Pilots for Combat Flight Simulation. *AI Magazine*, *20*(1) 27-41.

Laird, J. E. (2001). It knows what you're going to do: Adding anticipation to a Quakebot. In *Proceedings of the fifth international conference on Autonomous agents 2001* (pp. 385-392). New York, NY: ACM Press.

Lee, F. J., & Taatgen, N. A. (2002). Multitasking as Skill Acquisition. In W. D. Gray, & C. Schunn (Eds.), *Proceedings of the 24th Annual Conference of the Cognitive Science Society* (pp. 572-577). Hillsdale, NJ: Erlbaum.

Leuchter, S., & Urbas, L. (2002). Simulation Based Situation Awareness Training for Control of Human-Machine-Systems. In Valery Petrushin, Piet Kommers, Kinshuk, & Ildar Galeev (Eds.), *IEEE International Conference on Advanced Learning Technologies. Media and the Culture of Learning. Kazan, Russia: Sep 9-12, 2002* (pp. 34-39). Palmerston North, New Zealand: IEEE Learning Technology Task Force.

Lüdtke, A., Möbus, C., & Thole, H.-J. (2002). Cognitive Modelling Approach to Diagnose Over-Simplification in Simulation-Based Training. In S. A. Cerri, G. Gouarderes & F. Paraguacu (Eds.), *ITS 2002* (pp. 469-506). Berlin: Springer-Verlag (LNCS; 2363).

Nellen, S. (2002). *How Humans solve Scheduling Problems: Analysis of Human Behavior in the Plan-A-Day task* (Diploma Thesis) Department of Psychology, Ruprecht Karls Universität, Heidelberg. Retrieved Oct 30, 2002 from http://www.ub.uni-heidelberg.de/archiv/2129.

Niessen, C., Leuchter, S., & Eyferth, K. (1998). A psychological model of air traffic control and its implementation. In F. E. Ritter & R. M. Young (Eds.), *Proceedings of the Second European Conference on Cognitive Modelling (ECCM-98)* (pp. 104-111). Nottingham: Nottingham University Press.

Parasuraman, R., Sheridan, T. B., & Wickens, C. D. (2000). A model for types and levels of human interaction with automation. *IEEE Transactions on Systems, Man and Cybernetics - Part A: Systems and Humans*, *30*(3) 286-297.

Ritter, F. E., van Roy, D., & St. Amant, R. (2002). A User Modeling Design Tool Based on a Cognitive Architecture for Comparing Interfaces. In C. Kolski, & J. Vanderdonckt (Eds.), *Computer-Aided Design of User Interfaces III, Proceedings of the 4th International Conference on Computer-Aided Design of User Interfaces CADUI'2002* (pp. 111-118). Dordrecht: Kluwer Academics Publisher.

Rittle-Johnson, B., & Koedinger, K. R. (2001). Using cognitive models to guide instructional design: The case of fraction division. In *Proceedings of the Twenty-Third Annual Conference of the Cognitive Science Society* (pp. 857-862). Mahwah, NJ: Erlbaum.

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*(1) 85-107.

Schoppek, W., Boehm-Davis, D.A., Diez, M., Hansberger, J., & Holt, R.W. (2000). Letting ACT-R fly - A model of the interaction between trained airline pilots and the flight management system. In *Proceedings of the Seventh Annual ACT-R Workshop 2000*, Carnegie Mellon University, Pittsburgh, PA.