SIMPLIFYING THE DEVELOPMENT OF COGNITIVE MODELS USING PATTERN-BASED MODELING

Marcus Heinath * **Leon Urbas** **

** Berlin University of Technology, Center of Human-Machine Systems*
Franklinstraße 28-29, FR 2-7/2, D-10587 Berlin
*** Dresden University of Technology, Institute of Automation*
D-01062 Dresden

Abstract: Usability evaluation can be significantly improved using cognitive modeling. Even though it can help to understand and predict human behavior, the method is not widely applied in industrial environments, due mainly to the time-consuming model development process. We developed a high-level framework for cognitive modeling, HTAmap, which aims to reduce the modeling effort. The process of building cognitive models is transformed into a pattern-oriented task as well as a modification task based on cognitive activity patterns, which constitute generic behavior blocks. In consequence, cognitive modeling is opened for a wider user group. *Copyright © 2007 IFAC*

Keywords: cognitive systems, human factors, behavior modeling, production systems.

1. INTRODUCTION

Cognitive modeling extends classical usability methods and expands the repertoire by a prospective method that provides insight into detailed cognitive aspects of human-machine interaction. In contrast to empirical user testing, which requires a fairly implemented prototype or mockup of the system, the use of cognitive models allows system evaluation in early design phases based on design specifications only. However, cognitive modeling is hardly investigated in industrial usability research of human-machine systems (Urbas et al., 2005). Today the most important constraint on practical application is the poor cost/benefit ratio caused by a lack of support tools for modeling and by high requirements on sophisticated knowledge in cognitive psychology and artificial intelligence programming (see Heffernan et al., 2003; Byrne, 2005; Tollinger et al., 2005 for an overview). The *Hierarchical Task Analysis mapper* (HTAmap) methodology has been developed to minimize the effort for developing cognitive models especially in the cognitive architecture ACT-R (Anderson et al., 2004). HTAmap focuses on two key features: (1) minimizing the gap between human factors engineering and cognitive psychology research methods by using a plain high-level description notation. This notation provides a direct connection between task analysis and cognitive modeling. In addition, (2) cognitive model components based on *cognitive activity patterns* (CAP) can be reused. The placement of HTAmap as an additional layer between task analysis and implementation of the cognitive model is illustrated in Figure 1.

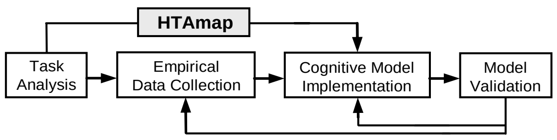

Fig. 1. Placement of HTAmap within the general cognitive modeling process.

This contribution aims to describe in detail the research background, the HTAmap research approach and the concept of implementation. The article closes with a description of a practical application including a first validation of HTAmap.

2. BACKGROUND & RELATED WORK

Cognitive architectures incorporate psychological theories and empirically based representations about aspects of human cognition that are relatively constant over time and task-independent. From that perspective, a cognitive model represents the procedural and declarative knowledge that is assumed to be required for performing a certain task. Building a cognitive model, the modeler must describe cognitive mechanisms in a highly-detailed as well as in a human-like way. Following Salvucci and Lee (2003) two levels of cognitive architectures can be differentiated. High-level architectures (e.g., GOMS; see John and Kieras, 1994 for an overview) describe human behavior on a basic level and define interactions as a sequence of actions. Low-level architectures (e.g. ACT-R, SOAR or EPIC, see Byrne, 2003 for an overview) describe human behavior at an atomic level (steps of roughly 50ms). They allow a more detailed insight into cognitive processes.

One of the most limiting steps during development of a cognitive model within a low-level architecture lies in the transfer of the results of the preliminary and required task analysis. The problem is caused by the different levels of task decomposition and knowledge formalization between both types of task description ("transformation gap"). Developing a cognitive model in ACT-R implies programming in a cognitive assembly language. Behavior is expressed in terms of production-rules (procedural knowledge) which manipulate declarative knowledge in terms of chunks. Hence, the development of high-level languages to model human cognition based on low-level cognitive architecture is a current issue in the cognitive research community (see Ritter et al., 2006 for an overview). The main goals are to simplify the model-building process and improve concepts for sharing and reusing model components.

Examples like ACT-Simple (Salvucci and Lee, 2003), G2A (St. Amant and Ritter, 2004) and ACT-Stitch (Matessa, 2004) represent such high-level frameworks that compile GOMS models to ACT-R models. The GOMS-based ACT-R models rely on a static, sequential structure (i.e. top-down control). This results on the one side in a quick and easy building process, but on the other side in an implicit neglect of guidance by external events (i.e. bottom-up processing) that is essential within dynamic systems. No options for conditional loops or free sequences of actions are possible. The compiled ACT-R models are mainly constrained to interact with event-based applications. Lebiere et al. (2005) pursue a different approach, and propose a hybrid framework consisting of a task network simulation tool (IMPRINT) and the cognitive architecture ACT-R. IMPRINT is used to model the high-level task structure; ACT-R handles selected cognitive and decision-making parts of the task. A fully-integrated IMPRINT-ACT-R model that represents a whole task network at the level of cognitive actions is still work in progress.

3. HIERARCHICAL TASK ANALYSIS MAPPER METHODOLOGY

The HTAmap methodology strives for goals similar to the GOMS- and IMPRINT-ACT-R approaches. To open cognitive modeling for a wider user group and make the developing task easier and more accessible, HTAmap provides:

- a structured formalization method to minimize the "transformation-gap" between the semi-formal, high-level task modeling and the formal, low-level cognitive model implementation
- a separated representation of task- and device-related knowledge within an integrated ACT-R model
- a strategy for model reuse as well as model adaptation with regard to the dynamic task environment and user skills

More precisely, HTAmap delivers cognitive modeling functionality based on predefined and modifiable *cognitive activity patterns* (CAP) and options for integrating separately defined device models of complex dynamic task environments (ACT-R graphical interface (AGI) model) and interfaces for embedding perception and action-models (AGI strategies). Hence, a large part of the cognitive model building process is transformed to a more simple pattern-oriented modification task (see Figure 2 for an overview).

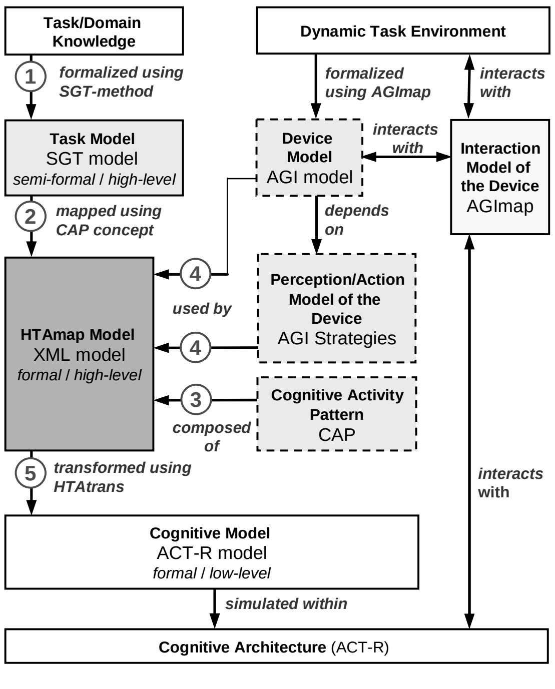

Fig. 2. The HTAmap methodology with the HTAmap model as integrating part between task-, device- and perception/action models (rectangles with dashed-line shown reusable/predefined elements).

3.1 HTAmap Modeling Process

The HTAmap approach integrates different levels of knowledge representation and cognitive model implementation within a single framework.

Step 1: Task Analyzing Process. To formalize task knowledge, a thorough task analysis has to be carried out. The *sub-goal template method* (SGT; Ormerod and Shepherd, 2004) is used within HTAmap (see Figure 2, step 1). This method was selected because it uses the simplicity and utility of the *hierarchical task analysis method* (HTA; Shepherd, 2001) and extends HTA by providing a nomenclature for stereotypical operator tasks, by specifying information requirements for each task and by terminating the lowest level of task decomposition. Within the SGT, four steps are essential (see Figure 3, upper part): (1) the initial task/subtask decomposition to the point where information-handling operations (IHO) are recognized and (2) re-described, followed by (3) the strategic decomposition and (4) the re-description in terms of sub-goal templates.

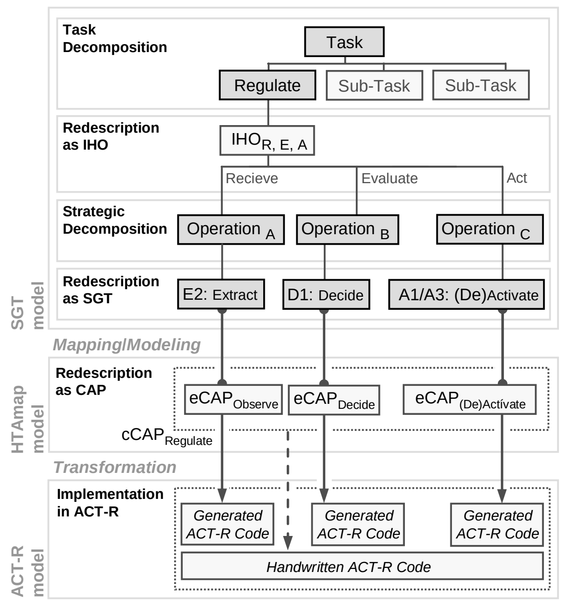

Fig. 3. Overview of the HTAmap mapping, modelling and transformation process.

Within the SGT method IHOs are divided into three classes: receiving (IHO$_R$), evaluating (IHO$_E$) and acting on information (IHO$_A$). The identified IHOs are re-described as predefined operator tasks regarding one of the five specified sub-goal templates of the SGT method: (A)ct, (E)xchange, (N)avigate, (D)ecide and (M)onitor. In addition for each task description level the SGT method defines plans that describe in which order IHOs are sequenced relative to each other.

Step 2: Mapping and Modeling Process. The redescription of the task in terms of SGTs is the starting point for building the HTAmap model (see Figure 2, step 2). CAPs predefined by an ACT-R expert are being used to solve the transformation-gap from high-level to low-level description. A CAP represents a generalized solution for execution of an operator task using cognitive resources to tackle a recurrent problem in a specific context. In general, a CAP describes the specific applications of an identified SGT at a less abstract cognitive level of ACT-R. In detail, a CAP comprises the ACT-R declarative and procedural structures and provides interfaces for parameterization of regarding various environments and user skills. For modeling perception, cognition and action processes, different CAPs are available in HTAmap (see Table 1).

Table 1 A selection of SGTs and associated CAPs

SGT$_{code}$	SGT$_{name}$	CAP$_{name}$	CAP$_{type}$
E2	extract (information)	scan	cCAP
M2	monitor (changes)	monitor	cCAP
E2	extract (information)	observe	eCAP
A1	activate (element)	activate	eCAP
D4	judge	evaluate	cCAP

Elementary cognitive activity patterns (eCAP) are the basic building blocks for tasks. They represent the lowest level addressed in the task analysis using the SGT method and are needed to describe tasks which cannot (or should not) be broken down into smaller subtasks. *Compound cognitive activity patterns* (cCAP) are used to represent the hierarchical structure of tasks. Complex task structures are represented via nesting of cCAPs: a cCAP can contain several eCAPs and/or other cCAPs. For example (see Figure 3, middle part) the cCAP$_{Regulate}$ contains the eCAP$_{Observe}$, the eCAP$_{Decide}$ and the eCAP$_{(De)Activate}$. The kind of execution of CAPs is specified by their *order-types* which strongly refer to the plan concept of the SGT method: fixed sequence (S1), contingent sequence (S2), free sequence (S3) and simple choice (S4).

The HTAmap model itself is built by instances of the selected CAPs and the specifications of order types (see Figure 2, step 3). As a result, the HTAmap-model represents the meta-description of an ACT-R model by the ends of a high-level SGT-task model. In general, the structure of an HTAmap model is a task-tree, but options for setting loops, returns and condition-based branching also convert the model structure to a kind of task-network. Figure 4 shows the layout of an HTAmap model.

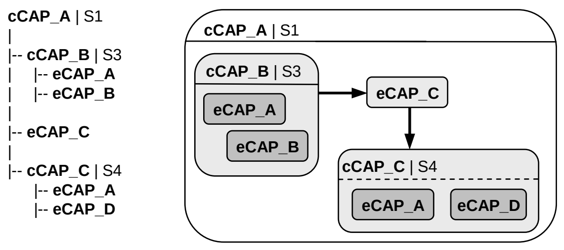

Fig. 4. An HTAmap model: on the left side as tree-view and on the right side as graphical-view.

In Figure 4 the cCAP_A is composed of cCAP_B, eCAP_C and cCAP_C ordered in a fixed sequence (S1: all CAPs are executed). cCAP_B contains eCAP_A and eCAP_B in a free order (S3: all CAPs are executed). The cCAP_C is a simple choice (S4: only a selected CAP is executed) with the options eCAP_A and eCAP_D. To represent the task structure from SGT analysis and to glue the CAPs on the same task level together and keep them inside the task hierarchy, task contexts (TC) are specified for each CAP. One task context per task level is needed, starting with TC = 0. For instance the cCAP_A has a TC = 0, because in this case it represents the highest hierarchical task level. The cCAP_B, eCAP_C and cCAP are on the same task level (one below cCAP_A), so they have a TC = 1. In the same manner eCAP_A, eCAP_B and eCAP_D would have a TC = 2. The task context indicates where the activity model is in the task execution.

Step 3: Adjustment Process. To adjust the generic HTAmap model regarding different task environments and individual types of user, the AGImap approach is used (Urbas and Leuchter, 2005). AGImap generates a description of the used tasks interface in terms of the ACT-R graphical interface (AGI), connects the AGI model with the dynamic task environment and also provides AGI strategies that handle the perception and action of ACT-R models with the dynamic task environment (see Figure 2, step 4). The high-level analyzed task is now specified in terms of the low-level cognitive architecture ACT-R and adjusted relating to a specific task environment and type of user (perception and action skills).

Step 4: Transformation Process. The last step within the HTAmap methodology is the transformation of the HTAmap model into ACT-R code by using the HTAtrans transformation engine. The latter uses the information provided by the HTAmap model to create an internal object model suitable for transformation. The CAP skeletons are converted to (Java) objects by utilizing Java Emitter Templates (JET) and the Eclipse Modeling Framework (EMF). With access to the CAPs and the task structure described by HTAmap, the internal object is used to create executable ACT-R code. (see Figure 2, step 5 and Figure 3, lower part).

3.2 HTAmap Implementation Framework

The HTAmap model represents a computational notation of high-level SGT task, written down in an XML based notation. Figure 5 shows an overview of the components used. The HTAmap model consists of three main parts. The first one is the *repository* definition section that specifies which repositories are used in the HTAmap model. The repositories themselves and their content are predefined by an ACT-R expert who is also familiar with the HTAmap methodology. The repository content represent generalized reusable components. The definition of this section allows the HTAmap access to a predefined "application library" of reusable model components (i.e., GUI types, eCAPs, cCAPs and AGI strategies). The second section includes the description of the task *environment* in terms of the AGI model elements containing the required information about the task domain, particularly with regard to elements of the interface (e.g. button or text-elements). The third section implies the analyzed task model represented by *activities*. The latter is a compound activity, containing several elementary and compound activities in a recursive structure (including all the meta information on how to connect pattern to represent the previously created task analysis). Activities are the instantiation of CAPs, whereby an eCAP corresponds to an elementary activity and a cCAP corresponds to a compound activity.

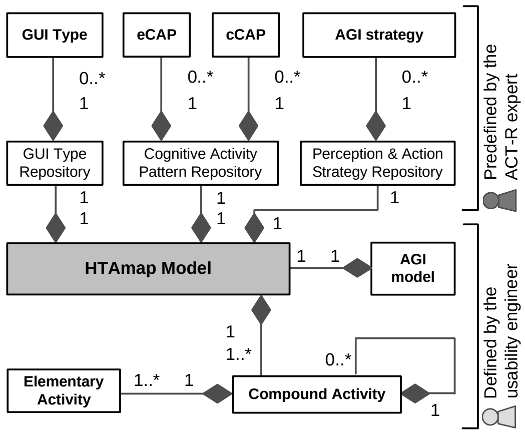
GUI Type
eCAP
cCAP
AGI strategy
0..*
1
GUI Type Repository
Cognitive Activity Pattern Repository
Perception & Action Strategy Repository
HTAmap Model
AGI model
Elementary Activity
Compound Activity
1..*
Predefined by the ACT-R expert
Defined by the usability engineer

Fig. 5. UML-diagram: HTAmap model with components and cardinalities.

To summarize, the HTAmap model repository section contains links to the reusable components (i.e. strategies and patterns) predefined by an ACT-R expert. The two other sections represent the task environment with respect to the AGI model and the modeled activities defined by the usability engineer, familiar with ACT-R but not an expert.

4. APPLICATION AND VALIDATION

To validate the approach, we modeled a dynamic process control task using the HTAmap methodology and transformed this to ACT-R code (see Figure 6). The resulting simulated behavior data was analyzed and compared to human behavior data and to behavior data generated by the simulation of two different handwritten ACT-R models.

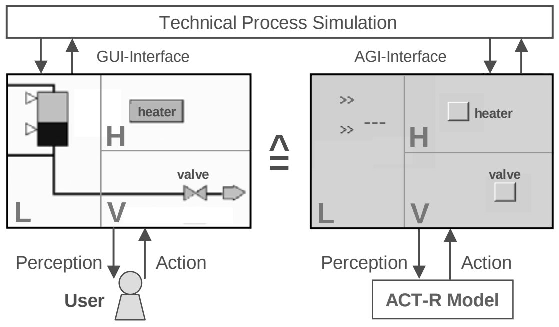

Fig. 6. Dynamic task environment with the graphical user interface (GUI) for the human (right) and the ACT-R graphical interface (AGI) for the cognitive models (left) and specified areas of interest (AOI): (H)eater, (V)alve and (L)evel.

4.1 Task and HTAmap Model Description

The control task is to stabilize the level of liquid in a container which is moderated by an inflow, an outflow (adjusted by a valve) and an evaporation (adjusted by a heater). For the model based investigation, the task structure and information requirements regarding the process simulation were analyzed. Based on this information four different ACT-R 6.0 models were developed representing the internal knowledge of an experienced user who is familiar with the technical process (as found in a survey of experts). The ACT-R models differ in their internal control structure. The first one is a handwritten state-based ACT-R model with maximal internal control (M_{maxC}). That means the sequence of executable production rules is fixed by explicit control states, regardless of the contextual situation of the task environment. The second model is a handwritten minimal control ACT-R model (M_{minC}). Minimal control (see Taatgen, 2005 for a detailed description) stands for the approach to use as few control states as possible. The model is bottom-up controlled rather than top-down; which means the task-relevant information from the task environment is in full control of the model. The third model is a generated HTAmap model (M_{HTA}) based on identified SGTs and their related CAPs. A graphical view of this model is depicted in Figure 7.

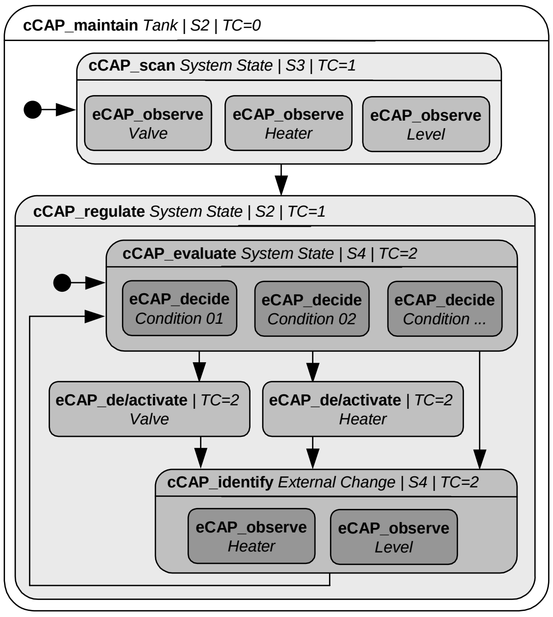
cCAP_maintain Tank | S2 | TC=0
cCAP_scan System State | S3 | TC=1
eCAP_observe Valve
eCAP_observe Heater
eCAP_observe Level
cCAP_regulate System State | S2 | TC=1
cCAP_evaluate System State | S4 | TC=2
eCAP_decide Condition 01
eCAP_decide Condition 02
eCAP_decide Condition ...
eCAP_de/activate | TC=2 Valve
eCAP_de/activate | TC=2 Heater
cCAP_identify External Change | S4 | TC=2
eCAP_observe Heater
eCAP_observe Level

Fig. 7. HTAmap model of the process control task: The CAPs are symbolized by rounded rectangles and their relation by arrows (black circle: starting point; S1...S4: SGT order type; TC: task context).

The HTAmap model combines top-down and bottom-up control processes. As a consequence of the used concept of task contexts it is mainly top-down controlled. But within several cCAPs the model is controlled on the basis information of task environment (bottom-up control). The fourth model (M_{HTAOpt}) is an optimized version of the third one using the production compilation approach (see Taatgen and Lee 2003 for a detailed description): two productions rules that fire in sequence are combined into one new rule that performs the same function as the two originally distinct production rules. Within the HTAmap methodology the production compilation mechanism is not used to represent learning aspects of the model but rather to speed up the model caused by artifacts of HTAmap. For example the sequencing of two CAPs (e.g. $cCAP_{scan}$ and $cCAP_{regulate}$, see Figure 7) requires a “set-relation” production rule (visualized by the arrow) that connect the last production rule of $cCAP_{scan}$ (finish-rule) with the first production rule of the subsequent $cCAP_{regulate}$ (start-rule). Production compilation is now used to combine the set-relation and finish rule to a new one.

4.2 Results and Discussion

Four ACT-R models (M_{HTA}, M_{HTAOpt}, M_{maxC}, M_{minC}.), with 30 runs per model and a running time of 45 sec. for the task, were applied in a within-comparison and compared with subjects (human) behavior (n=30). Parameters were (1) stabilization factor (ratio of level within-outside lower/upper limit over time), (2) valve regulation (number of interventions) and (3) main local scanpaths.

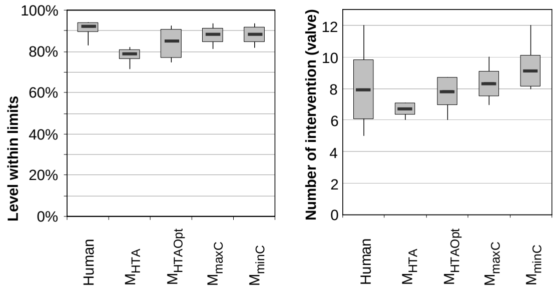

Fig. 8. Mean, SD, min-value and max-value for *stabilization factor* (left) and *valve regulation* (right).

The stabilization factor turns out to be lower for M_{HTA} compared to the other two (handwritten) ACT-R models and the empirical data (see Figure 8). This is explained by the lower “speed” of M_{HTA} than the other models as a result of “set-relation” production rules (requiring additional time for execution). However, production compilation reduces this difference between the modified model (M_{HTAOpt}) and M_{maxC} as well as M_{minC} to a mean of only 4%. In relation to the basic model (M_{HTA}) an improvement of almost 8% is enabled by the optimized model (M_{HTAOpt}). All ACT-R models resemble the human behavior in terms of valve regulation (within SD). M_{HTAOpt} also fits perfectly with the empirical data and no further modification is needed (see Figure 8).

Human local scanpaths (a consecutive sequence of fixations) regarding the specified areas of interest ((H)eater, (V)alve and (L)evel; see Figure 6) were analyzed and compared with local scanpaths predicted by ACT-R models using a simulation trace analyzer tool (SimTrA; Dzaack and Urbas, in press). The complete sequence of AOI fixations (fixations at the same AOI are treated as a single gaze fixation) are categorized as triples (e.g.; scanpath: LVLVLVHL; separated triple: LVL | VLV) and ordered by frequency. The six main scanpaths (frequency >5%) that cover 82% of empirical data were used for comparison (see Figure 9). The mean deviation between added up frequencies of M_{HTAOpt} and human is 3% (M_{minC}: 2%, M_{HTA}: 11%, M_{maxC}: 16%). Based on its substructure M_{HTAOpt} (i.e., state based control including bottom-up processing) is an appropriate candidate to simulate human visual behaviour.

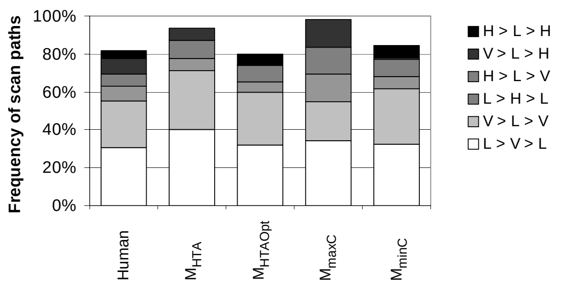

Fig.9. Mean frequencies for *important scan paths*.

5. CONCLUSION AND OUTLOOK

The presented approach provided ample evidence that building cognitive models using HTAmap extends the modeling process to a wider user group. Main reason is a simplification of reusing model fragments and consequently an increase of model application in the context of usability evaluation of Human-Machine Systems. The HTAmap model can be seen as a first approximation to model human behavior and provides a good starting point for further refinements of ACT-R models. In the currently, only a subset of CAPs is specified. The transfer of more “associated” production rules into CAPs require further validation steps. An editor for building HTAmap-models is work in progress. Mapping and validation of more complex process control tasks into ACT-R models using the HTAmap is part of future investigations.

ACKNOWLEDGEMENTS

This PhD project is funded by Deutsche Forschungs-gemeinschaft (research training group GRK 1013 prometei). Preliminary work was funded by VolkswagenStiftung. We thank Jeronimo Dzaack and Andre Wiesner for help in model analysis and building the software.

REFERENCES

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C. and Qin, Y. (2004). An integrated theory of the mind. *Psychological Review,* **111**, 1036-1060.

Byrne, M. D. (2003). Cognitive Architectures. In: *Handbook of Human-Computer Interaction* (J. Jacko and A. Sears, Eds.), pp. 97-117, Hillsdale, NJ: LEA.

Byrne, M. D. (2005). Cognitive architectures in HCI: Present work and future directions. In *Proc. HCI '05*. Mahwah, NJ: Erlbaum.

Dzaack, J. and Urbas, L. (in press). Cognitive Model Data Analysis for the Evaluation of Human Computer Interaction. In *Proc. HCI '07*.

Heffernan, N. T., Koedinger, K. R. and Aleven, V. A. (2003). Tools Towards Reducing the Costs of Designing, Building, and Testing Cognitive Models. In *Proc. BRIMS '03*.

John B. and Kieras D. (1994). The GOMS Family of Analysis Techniques: Tools for Design and Evaluation. *Technical Report*. Carnegie Mellon University, Pittsburgh, PA.

Lebiere, C., Archer, R., Warwick, W. and Schunk, D. (2005). Integrating Modeling and Simulation into a General-Purpose Tool. In *Proc. HCI 2005*. Las Vegas, NV.

Matessa, M. (2004). An ACT-R modeling framework for interleaving templates of human behavior. In *Proc. CogSci '04*, pp. 903-908.

Ormerod, T. C. and Shepherd, A. (2004). Using Task Analysis for Information Requirements Specification: The Sub-Goal Template (SGT) Method. In: *The handbook of task analysis for human-computer interaction* (D. Diaper and N., A. Stanton., Eds.), pp. 347-365. Mahwah, NJ: LEA.

Ritter, F. R., Haynes, S. R., Cohen, M., Howes, A., John, B., Best, B., et al. (2006). High-level Behavior Representation Languages Revisited. *Proc. ICCM '06*, pp. 404-407.

Salvucci, D. D. and Lee, F. J. (2003). Simple Cognitive Modeling in a Complex Cognitive Architecture. In *Proc. CHI 2003*, pp. 265-272.

Shepherd, A. (2001). *Hierarchical task analysis*. New York: Taylor & friends.

St. Amant, R. and Ritter, F. E. (2004). Specifying ACT-R models of user interaction with a GOMS language. *Cognitive Systems Research,* **6**, 71-88.

Taatgen, N.A. and Lee, F.J. (2003.) Production Compilation: A simple mechanism to model Complex Skill Acquisition, *Human Factors,* **45 (1)**, 61-76.

Taatgen, N. A. (2005). Modeling parallelization and flexibility improvements in skill acquisition: from dual tasks to complex dynamic skills. *Cognitive Science,* **29 (3)**, 421-455.

Tollinger, I., Lewis, R. L., McCurdy, M., Tollinger, P., Vera, A., Howes, A., et al. (2005). Supporting efficient development of cognitive models at multiple skill levels: exploring recent advances in constraint-based modeling. In *Proc. CHI '05*, pp. 411-420.

Urbas, L. and Leuchter, S. (2005). Model Based Analysis and Design of Human-Machine Dialogues through Displays. *KI – Künstliche Intelligenz*, **4**, 45-51.

Urbas, L., Dubrowsky, A., Dzaack, J. and Heinath, M. (2005). Entwicklungsmodelle in der Praxis: Ergebnisse einer Interviewstudie. *Bericht am ZMMS*, TU Berlin.