Goal setting mechanism in Petri net models of human decision making

M. Rauterberg, M. Fjeld, and S. Schluep

Researchgroup Human Machine Interaction (MMI)
Institute for Hygiene and Applied Physiology (IHA)
Swiss Federal Institute of Technology (ETH)
Clausiusstrasse 25, CH-8092 ZURICH, Switzerland
+41-1-632 70 82, rauterberg@iha.bepr.ethz.ch

ABSTRACT

To support the human factors engineer in designing a good interactive system, a method has been developed to analyze the empirical data of the interactive decision behavior described in a finite discrete state space. The sequences of decisions and actions produced by users contain much information about their mental models, the individual problem solution strategy for a given task and the underlying decision structure. We distinguish between (1) the *logical structure* (the 'device model'), (2) the *sequential goal structure*, and (3) the *temporal structure*. The analyzing tool AMME can handle the recorded decision and action sequences and automatically extracts a net description of the task dependent decision model (the logical structure). This basic model is extended by further elements to reconstruct an empirical expert user sequence. This article presents two modeling strategies: (1) *event-driven*, versus (2) *parallel* goal setting processes. Both strategies add sequential structure to the logical structure. Three different models are presented and their predictive power is discussed.

Index terms: Behavior driven modeling, cognitive modeling, Petri net, goal setting strategy.

1. INTRODUCTION

What mental models are and how they work, is quite unclear. Carroll and Reitman-Olson [2] summarize their research recommendations as follows: "(1) Detail what a mental model would consist of and how a person would use it to predict a system's behavior. … (2) Investigate whether people have and use mental models of various kinds. … (3) Determine the behaviors that would demonstrate the model's form and the operations used on it. … (4) Explore alternative views of Sequence/Method representations and the behavior predicted from them. … (5) Explore the types of mental representations that may exist that are not mechanistic. … (6) Determine how people intermix different representations in producing behavior. … (7) Explore how knowledge about systems is acquired. … (8) Determine how individual differences have an impact on learning of and performance on systems. … (9) Explore the design of training sequences for systems. … (10) Provide system designers with tools to help them develop interfaces that invoke good representations in users. … (11) Expand the task domain to more complex software" ([2] pp. 59-61). In this paper we present a modeling approach that contributes to points (1), (3), (4), (5), (7), (10) and (11). We are primarily interested in a bottom-up, behavior driven and not in a top-down, theory driven approach.

Most of the known modeling approaches are based on the assumption that the "mental model maps completely to the relevant part of the conceptual model, e.g. the user virtual machine. Unexpected effects and errors point to inconsistency between the mental model and the conceptual model" ([16] p. 258). This one-to-one mapping between the mental model and the conceptual model of the interactive system implies a *positive* correlation between the complexity of the observable behavior and the complexity of the assumed mental model. But this assumption seems to be wrong.

Based on the empirical result in [12], that the complexity of the observable behavior of novices is larger than the complexity of experts, we must conclude that the behavioral complexity is *negatively* correlated with the complexity of the mental model. If the cognitive structure is too simple, then the concrete task solving process must be filled up with a lot of heuristics or trial and error behavior. Learning how to solve a specific task with a given system means that the behavioral complexity decreases and the cognitive complexity increases. Now, one of the central question is: What kind of knowledge is stored in the cognitive structure? Before we are able to give a preliminary answer to this question, we have to introduce our complexity measure.

2. THE MEASUREMENT OF COMPLEXITY

The symbolic representation of the machine system consists of the following elements: 1. objects (things to operate on), 2. operations (symbols and their syntax), and 3. states (the 'system states'). The mental model of the user can be structured in representing: objects, operations, states, system structure, decision and task structure. A net can be described as a mathematical structure consisting of two non-empty disjoint sets of nodes (S-elements and T-elements), and a binary flow relation (F). The flow relation links only different node types and leaves no node isolated [9]. Petri nets can be interpreted in our context by using a suitable pair of concepts for the sets S (signified by a circle or an oval '( )', where an oval means original behavior) and T (signified by a square '[ ]') and a suitable interpretation for the flow

relation F (signified by an arrow '->'). Bauman and Turano [1] showed, that Petri nets are equivalent to formalism based on production rules (like CCT of Kieras and Polson [7]). In this sense, our approach can also be subsumed under 'logic modeling'. The main operations (relations) between two Petri nets are *abstraction*, *embedding* and *folding* [4].

The *folding operation* in the Petri-net theory is the basic idea of the approach presented in this paper. Folding a process means to map S-elements onto S-elements and T-elements onto T-elements while keeping the F-structure. The result is the structure of the performance net. Each state corresponds to a system context, and each transition corresponds to a system action. This sequence is called a 'process' (see Fig.1). An *elementary process* is the shortest meaningful part of a sequence: (s') -> [t'] -> (s'').

If the observable behavior can be recorded in a complete ...-> (state) -> [transition] -> (state) ->... process description (see Fig. 1), then the analysis and construction of the net structure of this process are simple: you only have to count the number of all different states and transitions used, or to mark on a list the frequencies of each state and transition used in the process.

However, if the observable behavior only can be recorded as an incomplete (e.g.,...-> (state) -> [transition] -> [transition] ->... or ...-> (state) -> (state) -> [transition] ->...) process description, then the analysis and construction of the net get difficult. You have to find out the correct state (transitions, respectively) between both transitions (states, respectively). Unfortunately, this is the most frequent case in practice. For these cases we need automatic tool support. In the last years we developed a tool, that gives us the possibility to analyze any processes with an incomplete process description, that are generated by finite state transition nets [12].

The aim of the 'folding' operation is to reduce the elements of an observed empirical decision process to the minimum number of states and transitions, with the reduced number of elements being the 'logical decision structure'. Folding a decision process extracts the embedded net structure and neglects the amount of repetitions, the sequential order, and the temporal structure. A simple pattern matching algorithm looks for all 'elementary processes' in the sequence. A composition algorithm (the folding operation) is now able to build up the Petri net combining all elementary processes. The result of a folding operation applied to the behavioral sequence (Fig. 1) is the Petri net given in Fig. 2.

Measurable features of the behavioral process are: number of states and transitions totally used, number of different states and different transitions used, dwell time per state and transition, etc. These measurements can be done, based on a protocol of the user's behavior automatically recorded by an interactive software program (the dialog system) in a 'log file'.

To measure complexity we use the C_{cycle} metrics of McCabe [8]. With C_{cycle} we have a useful quantitative metric to measure behavioral complexity. We are discussing the advantages and disadvantages of three different quantitative metrics in the context of an empirical investigation elsewhere [11]. The complexity measured with C_{cycle} is defined by the difference of the total number of connections (F: arrows) and total number of net elements (T-transitions plus S-states). The parameter P is a constant to correct the result of Formula 1 in the case of a sequence (F – (T + S) = -1); the value of P in our context is 1.

$$\mathbf{C_{cycle} = F - (T + S) + P} \qquad \mathbf{(1)}$$

The C_{cycle} value of the model-1 in Fig. 2 is [32-25+1=8]; the complexity of the net shown in Fig. 2 is eight. But, what could this number mean? McCabe [8] interprets C_{cycle} as the *number of linear independent paths* through the net. Other interpretations of C_{cycle} are *number of holes* in a net or *number of alternative decisions* carried out by the users.

Observing the behavior of people solving a specific problem or task, is our basis for estimating 'model complexity (MC)'. The cognitive structures of users are not directly observable, so we need a method and a theory to use the observable behavior to estimate MC. We call the complexity of the observable behavior the 'behavioral complexity'. This behavioral complexity can be estimated by analyzing the recorded concrete task solving process. The necessary task solving knowledge for a given task is constant. This knowledge embedded in the cognitive structure of the mental model can be reconstructed.

3. RECONSTRUCTION OF THE MENTAL MODEL

We carried out an empirical investigation to compare different types of interfaces [10]. For the reconstruction of a user's mental model we chose one log file of an expert user (see Fig. 1). The whole process of the shown example is based on 26 task solving transitions (user actions or automatic system actions) and overall 26+1=27 states (dialog or internal system states).

The user's task was to find out how many data records there are in a given data base (DB) consisting of file A, file B and file C of a given database (DB). By startup, the system automatically goes from the (Main menu) to the (Start menu) where it opens the DB ['G_2'] and then automatically goes back to the (Main menu) with ['M_3']. Now, the user selects function key ['F_3'] and goes to the file selection menu, where he selects file A by pressing key ['1'], bringing him back to the (Main menu). Pressing key ['i'] brings him to the (Info) module. With key ['d'] all DB information is automatically displayed sequentially on three different screens (3* ['M_22']) and the system automatically goes back to the (Info) module with ['M_11'].

But the *task relevant* information is only given on the first screen that is--without interrupting the scrolling output--overwritten with additional information. To find out how to stop the scrolling process the user tries key ['d'] again, and the same reoccurs. After the third time pressing ['d'], the user correctly follows up with a blank ['BL'] to interrupt scrolling

successfully. Hence the task relevant DB information stays on the screen. Pressing function key ['F_10'], the rest of the output is displayed automatically (2* ['M_22']).

Back in the info module, pressing key ['h'] brings the user to the (Main menu). Pressing key ['h'] anew, leads the user back to the start menu. Automatically, the DB is reopened ['G_2'] and the user selects function key ['F_10'] to finish off successfully.

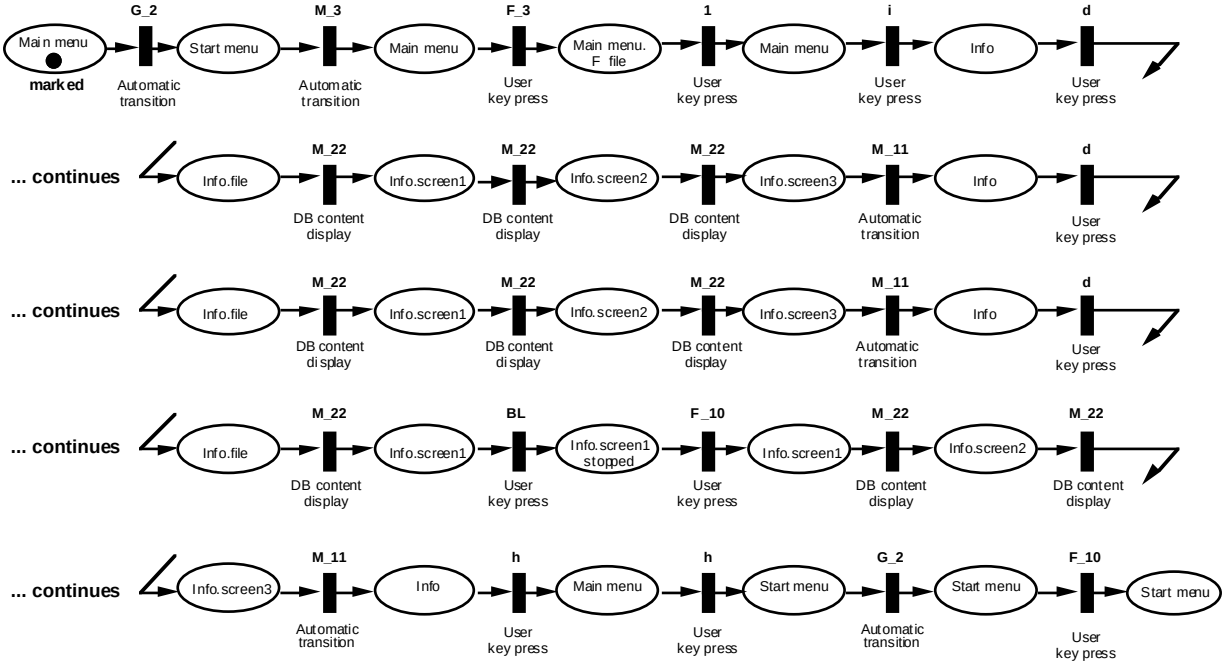


Fig. 1. The original behavioral sequence of an expert with a relational database system [10].

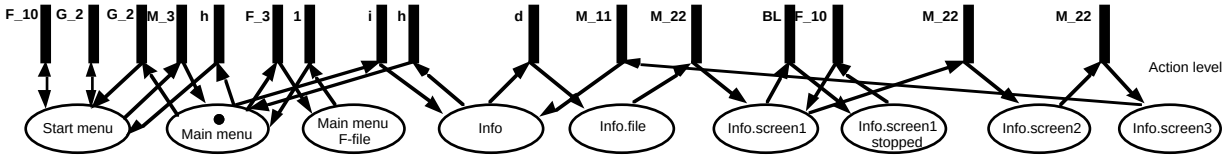


Fig. 2. *Model-1*: the pure 'logical structure' (the device model) of the logfile example sequence in Fig. 1.

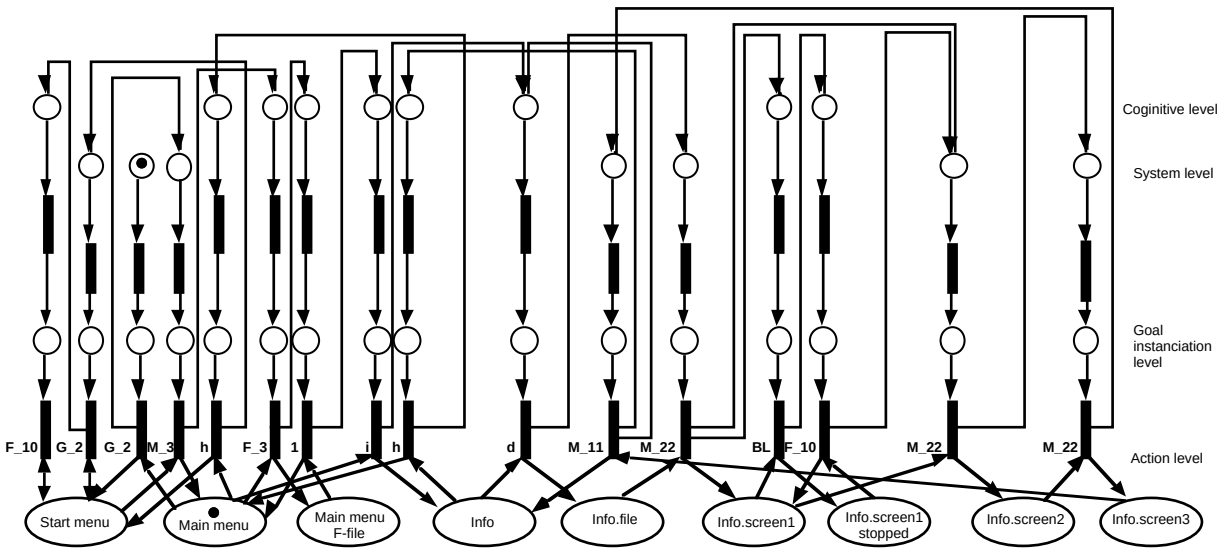


Fig. 3. *Model-2*: this Petri-net is equivalent to model-1 with structure to model event-driven goal setting.

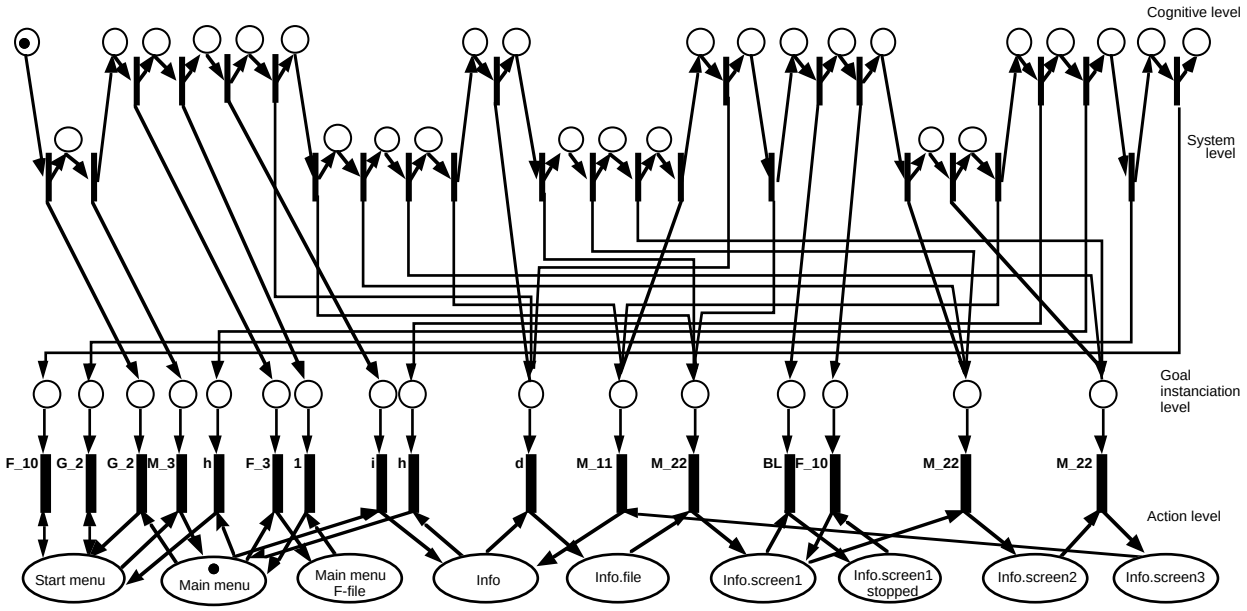


Fig. 4. *Model-3*: this Petri-net is equivalent to model-1 with additional structure to model parallel driven goal setting.

Mental models consist of three different types of knowledge: (1) the pure logical structure of the task (the device model), (2) the sequential structure of all goals, and (3) the temporal structure of all actions.

Model-1: The pure logical structure is automatically extracted with our tool AMME [12] [13]. This net is called model-1 (see Fig. 2). Model-1 does not contain any knowledge about goals and time.

The idea for *event-driven goal setting* processes was motivated by the Action Regulation Theory [5]. Action regulation theory offers a coherent body of principles for human-centered task and work design. For Hacker ([6], p. 61) the work task is "the central category of psychological consideration of activity..., as decisive specifications for the regulation and organisation of the activities occur with the 'objective logic' of its contents". This quotation makes clear that for all, who follow the activity or action regulation theory, the *task* becomes very important in the analysis of behavior. Great importance is attached to the concept of the *complete task*. Each complete task and action cycle starts with a goal setting part (see Fig. 5).

Characteristics of complete tasks, which must be taken into consideration when analysing and/or designing human activities, are, according to the concept of action regulation theory presented here (see Fig. 5):

(1) Task dependent setting of subgoals which are embedded in the superimposed task goal;

(2) Independent action preparation in the sense of taking on planning functions; and, selection of the means including the necessary actions for goal attainment;

(3) Mental or physical performance functions with feedback on performance pertaining to possible corrections of actions;

(4) Control with feedback on results and the possibility of checking the results of one's own actions against the set (sub-)goals.

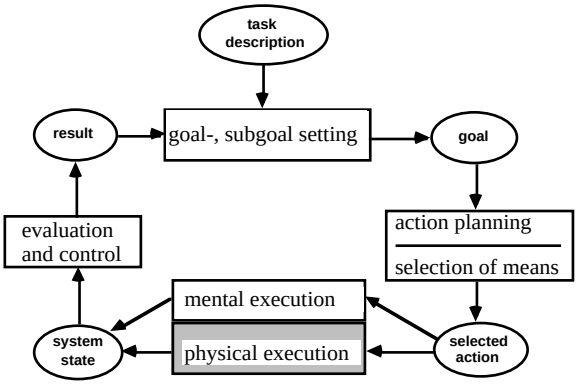


Fig. 5. The complete action-cycle: the central concept of the action regulation theory (see also [15]).

The content of the rule base (S-, T-, and F-rules: the complete system description file as part of AMME, see [13]) guiding the generation process of the 'cognitive part' of the mental model will be implemented in two ways:

a) To generate models with *event-driven goal setting* processes (action-driven goal setting, resp.), and
b) to generate models with *parallel goal setting processes*. Parallel goal setting means that the mental goal setting process is running parallel to the action level on which the dialog actions are executed.

The mental process of the user takes place at the cognitive level. The user has no control over the actions on the system level, because all actions on the system level are automatically carried out by the system itself. The selected actions (see Fig. 5) at the goal instanciation level are set from the cognitive level as well as from the system level. There is a direct synchronization between the action level and the two upper ones, cognitive and system level.

Model-2: Our *event-driven goal setting* example is given in Fig. 3. Each action (event, resp.) on the action level directly activates a mental process on the cognitive level. The mental process instanciate on the goal instanciation level the corresponding 'goal' (selected action, resp.) for the next action.

Model-3: The idea of a *parallel goal setting* process (see Fig. 4) was motivated by von Cranach's discussion of plans, 'anticipating representations', and intention ([3] p. 40). For each goal at the goal instanciation level to be set, an anticipated counterpart must be set beforehand on the cognitive level. In this process there is no synchronization between the cognitive (or system) and the action level.

In the last part of this article, model-1 (Fig. 2), model-2 (Fig. 3) and model-3 (Fig. 4) are going to be empirically validated.

4. VALIDATION OF THE THREE MENTAL MODELS

To validate the three different mental models, a simulation study was carried out. With the Petri-net simulator PACE each of our three models was implemented, marked and executed. We generated different task solving sequences with each model. When running each model, we used the following stopping conditions:

• For model-1, we ran the model until we entered the (Start menu) a fourth time.
• For model-2 and model-3 we ran the models until the Petri nets were dead.

To estimate the similarity between the original sequence (see Fig. 1) and each simulated sequence, we used the following procedure:

1. We numbered consecutively all actions ('transitions', respectively) in Fig. 1 ['G_2' = '1', 'M_3' = '2', 'F_3' = '3', …, 'F_10' = '26']. The number R is the rank-position of each transition [t] in the original behavioral sequence.
2. We attached these numbers to all generated transitions [t] of each simulated sequence. For example, the shortest sequence we found, was generated with model-1: ['G_2', 'F_10', 'G_2', 'F_10]. The rank-positions R of these four transitions are: ['1', '19','25', '26'] (compared with the original sequence).
3. We calculated a 'similarity ratio' (SR) as follows:

$$SR = \left[1 - \left\{ \sum_{t=1}^{N_{sim}} \left| R_{org,t} - R_{sim,t} \right| + \sum_{N_{sim+1}}^{N_{org}} \max\left(R_{org}\right) \right\} \Big/ N_{org}^2 \right] * 100\%$$

SR is a sufficient measure for the similarity between the simulated sequence and the original sequence. N is the number of all fired transitions in a sequence. The maximum of R_{org} is equal to N_{org} (N_{org} in Fig. 1 is 26). SR is only valid for simulated sequences that fulfill the following condition: $N_{sim} \leq N_{org}$. For example, SR of the shortest sequence ['1', '19','25', '26'] is 6%.

4. We averaged the similarity ratios of all simulated sequences per model (see Table 1). The results in Table 1 show that with increasing complexity of the mental model (MC), the similarity ratio (SR) tends to 100%. There seems to be a positive correlation between MC and SR and we can also see that the variance (measured by the standard deviation) decreases continually with increasing MC. This result indicates that the predictive power increases from model-1 to model-3.

Table 1. The model complexity (MC) and similarity ratios (SR) of model-1, -2, and -3.

	model-no.		
	1	2	3
MC: absolute value	8	23	35
SR: mean %	36	66	89
SR: standard deviation	± 30	± 17	± 10
SR: min.…max. %	6..87	52..89	69..100
# simulated sequences	12	12	12

5. DISCUSSION

In the original sequence (Fig. 1), there is a cyclic behavior with a learning effect. The user tried twice to see the task relevant information, but without success. The third time going through the cycle, he found the right actions, and succeeded. This is a typical learning effect based on trial and error. Analyzing learning effects with Petri nets was achieved with other strategies (see [14]). The observed learning effect seems to be difficult to simulate with event-driven goal setting, because repeated behavior cannot be integrated in the modeling. For the parallel goal setting approach though, repeated behavior is reflected in the model, with improved results from one to the next repetition. However, since there is no synchronization between the cognitive and the action level for the parallel goal setting approach, the similarity is still less than 100%.

Both the methods have good and poor aspects: Event-driven goal setting assures good synchronization. Although the knowledge about how to solve part of the requirements is included in the model, it tells nothing about a learning effect. The parallel goal setting approach supports a learning effect, which can be seen by looking to the model-3. An integration of the advantages from each of the two strategies will be a possible target for our further research.

We have to choose a stop criteria for model-1. The reason is that the simulations with this model never stopped before reaching N_{org} number of transitions. We looked for characteristics in the original behavioral sequence to find a distinctive pattern of state(s) and/or transition(s). We settled on the number of times a simulation passes by the state (Start menu). When a simulation of model-1 comes the fourth time to the state (Start menu), this means that the simulation stops. This led to the results shown in Table 1.

6. CONCLUSION

We can conclude from the validation results that the SR value is considerably higher for the models with added goal setting (model-2 and model-3) than for the pure logical structure (model-1). This justifies the idea of adding a goal setting structure to the logical one. So it seems to be possible to develop a completely automatic modeling tool based on a bottom up approach. Moreover, we see that with increasing model complexity (MC), the mean value of SR increased and the standard deviation got smaller. Comparing the two different modeling approaches, parallel goal setting performed better than event-driven goal setting. We were able to model learning effects with the parallel goal setting strategy.

REFERENCES

[1] R. Bauman and T.A. Turano, Production based language simulation of Petri nets. Simulation 47 (1986) 191-198.

[2] J. Carroll and J. Reitman-Olson, Mental models in human-computer interaction. In: M. Helander, Ed., Handbook of Human-Computer Interaction (North-Holland, 1991, pp. 45-65).

[3] M. von Cranach and R. Harré (eds.), The analysis of action. Cambridge University Press, 1982.

[4] H.J. Genrich, K. Lautenbach and P. S. Thiagarajan, Elements of general net theory. In: W. Bauer, Ed., Lecture Notes in Computer Science 84 'Net Theory and Applications' (Springer, 1980, pp. 21-163).

[5] W. Hacker, Action regulation theory and occupational psychology. Review of German empirical research since 1987. The German Journal of Psychology 18(2) (1994) 91-120.

[6] W. Hacker, Arbeitspsychologie. Huber, 1986.

[7] D.E. Kieras and P.G. Polson, An approach to the formal analysis of user complexity, International Journal of Man-Machine Studies 22 (1985) 365-394.

[8] T. McCabe, A complexity measure, IEEE Transactions on Software Engineering, SE-2 (1976) 308-320.

[9] C.A. Petri, Introduction to general net theory, pp. 1-19. In: W. Bauer, Ed., Lecture Notes in Computer Science 'Net Theory and Applications' (Springer, 1980).

[10] M. Rauterberg, An empirical comparison of menu-selection (CUI) and desktop (GUI) computer programs carried out by beginners and experts. Behaviour and Information Technology 11 (1992) 227-236.

[11] M. Rauterberg, A method of a quantitative measurement of cognitive complexity. In: G.C. van der Veer, M.J. Tauber, S. Bagnara and A. Antalovits, Eds., Human-Computer Interaction: Tasks and Organisation (CUD, Roma 1992, pp. 295-307).

[12] M. Rauterberg, AMME: an automatic mental model evaluation to analyze user behaviour traced in a finite, discrete state space. Ergonomics 36 (1993) 1369-1380).

[13] M. Rauterberg, A Petri net based analyzing and modelling tool kit for logfiles in human-computer interaction. In H. Yoshikawa & E. Hollnagel (Eds.), Proceedings 'Cognitive Systems Engineering in Process Control--CSEPC'96 (pp. 268-275). Kyoto University (1996).

[14] M. Rauterberg and R. Aeppli, Learning in man-machine systems: the measurement of behavioural and cognitive complexity. In Proceedings of IEEE International Conference on Systems, Man and Cybernetics--SMC'95 (Vol. 5, 1995, IEEE Catalog Number 95CH3576-7, pp. 4685-4690). Piscataway: IEEE.

[15] M. Rauterberg and E. Ulich, Information processing for learning systems: an action theoretical approach. In Proceedings of IEEE International Conference on Systems, Man and Cybernetics--SMC'96 (Vol. 3, 1996, IEEE Catalog Number: 96CH35929, pp. 2276-2281). Piscataway: IEEE.

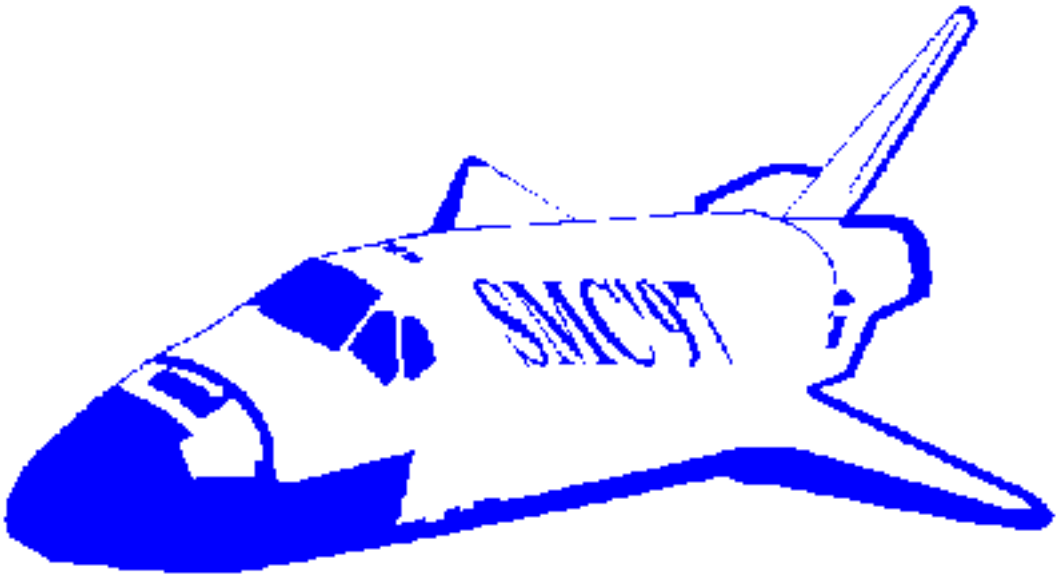
[16] Van der Veer, S. Guest, P. Haselager, P. Innocent, E. McDaid, L. Oesterreicher, M. Tauber, U. Vos and Y. Waern, Desiging for the mental model: an interdisziplinary approach to the definition of a user interface for electronic mail systems. In: D. Ackermann and M. Tauber, Eds., Mental Models and Human-Computer Interaction 1 (North-Holland, 1990, pp. 253-288).

SMC '97 Conference Proceedings

1997 IEEE International Conference on Systems, Man and Cybernetics

Conference Theme

Computational Cybernetics and Simulation



Volume 3 of 5

97CH36088-5

October 12-15, 1997

Orlando, USA

IEEE Catalog Number: 97CH36088-5

ISBN (Softbound Edition): 0-7803-4053-1