How to model behavioural and cognitive complexity in human-computer interaction with Petri nets

How to model behavioural and cognitive complexity in human-computer interaction with Petri nets

M. Rauterberg, S. Schluep and M. Fjeld

Human-Machine Interaction Group (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

Decisions and actions produced by computer users (observed process) contain much information about their mental models, individual problem solving strategy and underlying decision structure for a given task. Our tool AMME analyses observed processes and automatically extracts a Petri net description of the task dependent decision structure (logical structure). This net is extended by goal setting structures (modelling) and executed (simulated process). The aim is functional equivalence between observed and simulated processes. Three modelling strategies, event-driven, parallel goal setting, and feedback controled goal setting are presented.

1 Introduction

What mental models are and how they work, is quite unclear. Most of the known modelling 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 [1]. Unexpected effects and errors point to inconsistency between the mental model and the conceptual model" [13]. To overcome these obstacles we follow a bottom-up, consequently data driven modelling strategy .

2 The Measurement of Complexity

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 [8]. 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 behaviour) and T (signified by a square '[ ]') and a suitable interpretation for the flow relation F (signified by an arrow '->'). The main operations (relations) between two Petri nets are *abstraction*, *embedding* and *folding* [3].

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'. If the observable behaviour 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.

If the observable behaviour 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. We developed the tool AMME [11], that gives us the possibility to analyse any processes with an incomplete process description.

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. The result of a folding operation applied to the behavioural sequence (see Fig. 1) is the Petri net given in Fig. 2.

To measure the net complexity we use the C_{cycle} metrics of McCabe [7] (see Formula 1). With C_{cycle} we have a useful quantitative metric to measure behavioural complexity. 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 [30 -(15+10) +1 = 6]; the complexity of the net shown in Fig. 2 is six. But, what could this number mean? McCabe [7] 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.*

IEEE International Workshop on
Robot and Human Communication
0-7803-4076-0/97/ $10.00 © 1997 IEEE

Observing the behaviour of people solving a specific problem or task, is our basis for estimating 'model complexity (C_{cycle})'. The cognitive structures of users are not directly observable, so we need a method and a theory to use the observable behaviour to estimate C_{cycle}. The behavioural complexity can be estimated by analysing 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 [9]. For the reconstruction of user's mental models we used log files of five different expert users solving the same task. 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).

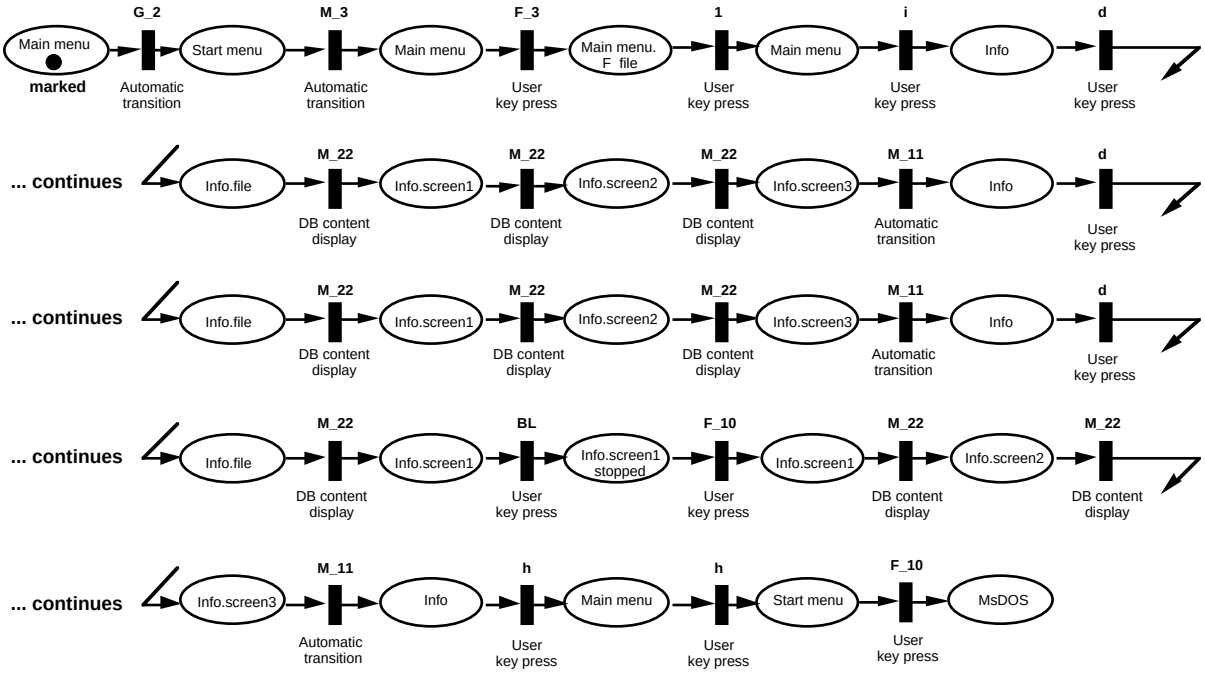

Fig. 1. The original behavioural sequence of an expert with a relational database system [9].

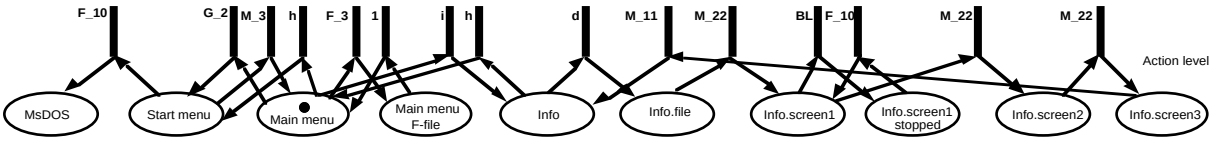

Fig. 2. *Model-1*: the pure 'logical structure' (the device model) of the logfile example sequence in Fig. 1.

As an example the task solving process of one expert user is presented here (see Fig. 1). The whole process is based on 25 task solving transitions (user actions or automatic system actions) and overall 25+1=26 states (dialogue or internal system states). 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, overwritten by the scrolling output. Trying to stop the scrolling process, the user tries key ['d'] again, but the same reoccurs. After the third time pressing ['d'], the user correctly follows up with a blank ['BL'] to interrupt the scrolling. Hence the task relevant information stays on the screen. Pressing ['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. The user selects function key ['F_10'] to finish off in (MsDOS).

The logged sequence contains three different types of knowledge: (1) the pure logical structure of the task (the

device model; see Fig. 2), (2) the sequential structure of all goals, and (3) the temporal structure of all actions.

Action regulation theory [4] offers a coherent body of principles for task analysis. For Hacker [5] 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 behaviour. Great importance is attached to the concept of the *complete task*. Each complete activity cycle starts with a goal setting part. 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 introduced here. Four different parts of each complete activity cycle must be differentiated (see also Fig. 9):

(A) Task dependent setting of (sub-)goals which are embedded in the superimposed task goal;

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

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

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

Following the theoretical implications from the action regulation theory we can differentiate between (1) the cognitive level with the mental goal setting processes (corresponding to A), (2) the goal instanciation level (corresponding to B), (3) the action level (corresponding to C), and (4) the feedback level (corresponding to D).

Model-1: The pure logical structure is automatically extracted with our tool AMME [10] [11]. This net is called model-1 (see Fig. 2). Model-1 does not contain any knowledge about goals and time. The model-1 is a subset of the content of the complete system description file (S-, T-, and F-rules: the rule base to describe the whole interactive system behaviour as part of AMME [11]). Fig. 3 shows that in model-1 only the action level is represented.

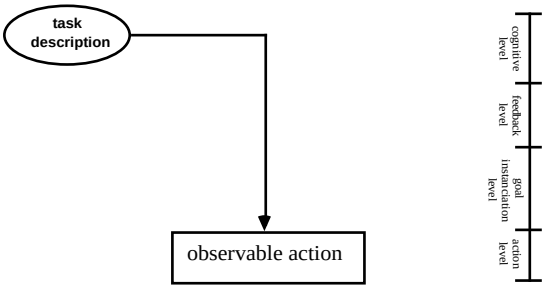

Fig. 3. In model-1 implemented part of the activity cycle.

But, we know that a mental process takes place at a cognitive level. So we have to find a way to model these mental processes by adding structure to model-1. Petri nets allow us to do this in a unified form: All the other levels above can be described and modelled with the same Petri net elements as the action level.

All observed actions (e.g. ['G_2'] and ['M_3']) that are automatically carried out by the system itself, are primarily not part of the cognitive level. Therefore, a system level must be a part of our model. Although the user cannot control the system level actions, he can take them into consideration. So, on the goal instanciation level a selected action is either set from the cognitive level or from the system level.

In a first modelling approach, *event-driven goal setting* , we added a minimal structure to model-1, to increase the *functional equivalence* between the simulated, new model and the original behavioural sequence.

Model-2: The *event-driven goal setting* model is given in Fig. 4. Each action (event, resp.) on the action level *directly* activates a mental process on the cognitive level. The mental process instanciates on the goal instanciation level the corresponding 'goal' for the next action. There is a direct synchronisation between the action level and the two upper levels.

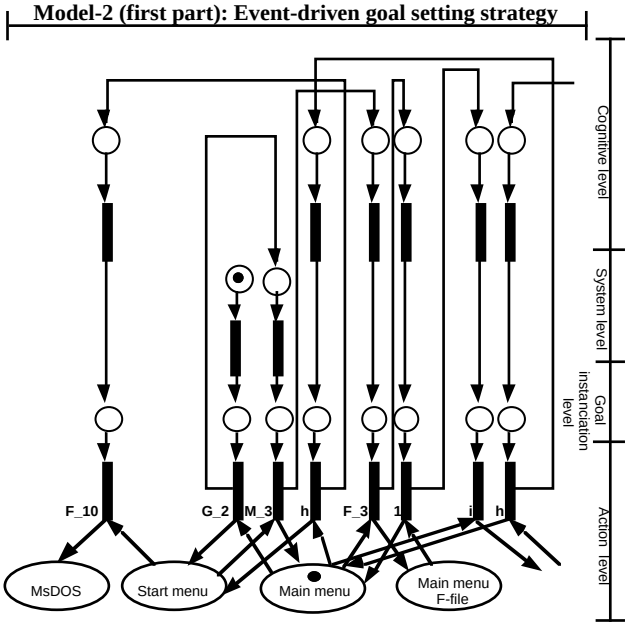

Fig. 4. *Model-2*: this Petri-net is equivalent to model-1 with added structure to model event-driven goal setting.

One of the most difficulties with this modelling approach is the impossibility to differentiate between at least two different subsequent goals from the same dialog state; this situation is always given if the user is going through a loop on the action level, coming back to the same dialog state, but going on in a different way than the last time.

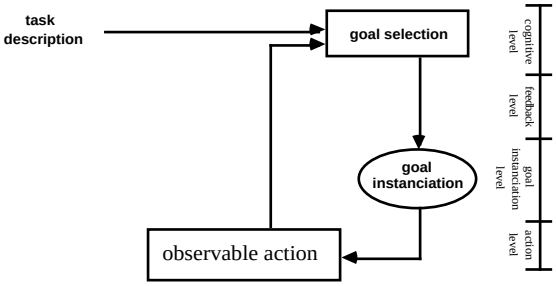

Fig. 5. In model-2 implemented part of the activity cycle.

Fig. 5 shows the implemented modelling approach of model-2: a closed loop between the action level and the goal selection and instanciation level.

Model-3: The *parallel goal setting* process (see Fig. 6) was introduced by von Cranach's discussion of plans, 'anticipating representations', and intention [2]. For each goal at the goal instanciation level to be set, an anticipated counterpart must be set beforehand on the cognitive level.

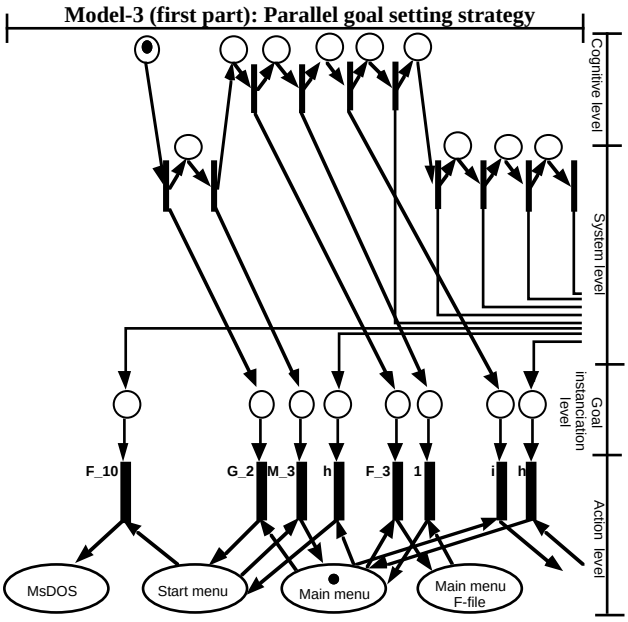

Fig. 6. *Model-3*: this Petri-net is equivalent to model-1 with added structure for parallel driven goal setting.

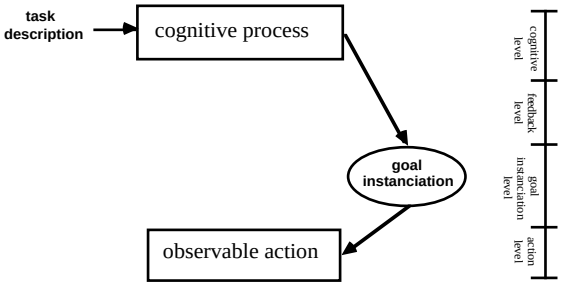

Fig. 7. In model-3 implemented part of the activity cycle.

Parallel goal setting means that the mental goal setting process is running parallel to the action level on which the dialogue actions are executed (see Fig. 7). There is no synchronisation from the action level upwards to the cognitive level, only from the cognitive level via the goal instanciation level downwards to the action level. This downward structure guarantees a partial synchronisation between the cognitive planning processes and the actions carried out. The cognitive planning process is always faster than the observable behaviour on the action level. This lack of synchronicity can lead to omission errors on the action level: slips and oversights occur and reduce the functional equivalence of this model.

Model-4: The reach a complete synchronization between the action level and the cognitive (system, resp.) level we added feedback loops to model-3 (see Fig. 8). These upward directed structure guarantees that the cognitive goal setting level can not be faster than the observable action process on the action level.

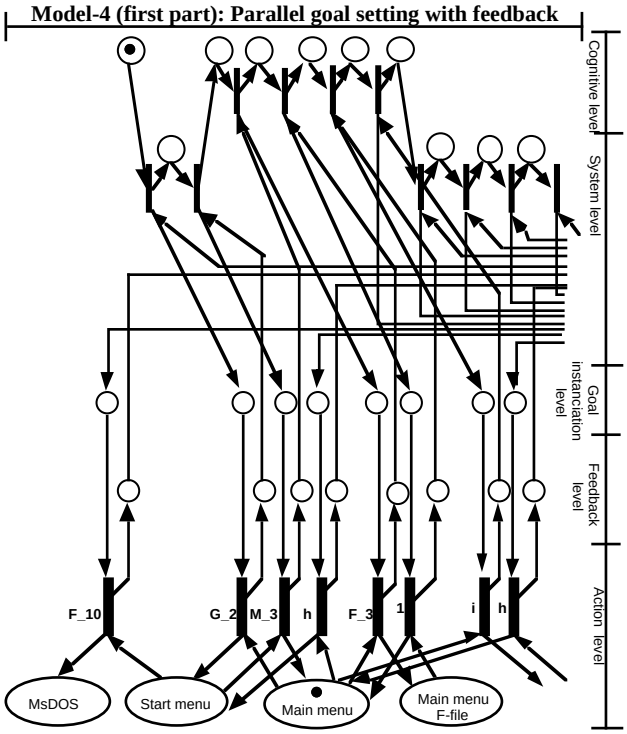

Fig. 8. *Model-4*: this Petri-net is equivalent to model-3 with added structure for feedback from action level to higher levels.

Model-4 is the most complex model compared with the other ones. Fig. 9 shows that a complete activity cycle is implemented in model-4. The feedback level is responsible for the fact that the cognitive process can not be faster than the observable action process (as we have seen in model-3). The next step is to find out which model can reach the maximum of 100% in functional equivalence with the original action sequence generated by a humen expert.

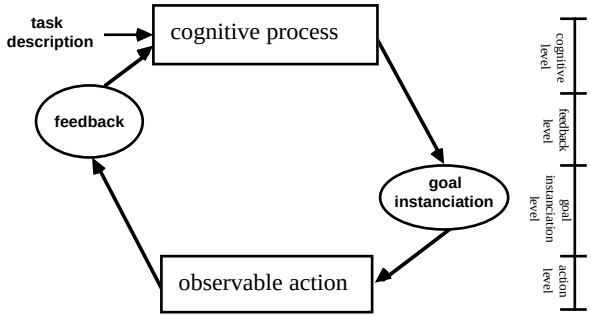

Fig. 9. A complete activity cycle implemented in model-4.

4 Validation of Cognitive Goal Setting Strategies

To validate the different goal setting strategies, a simulation study of model-1, -2, -3, and -4 was carried out. We modelled the task solving processes of five expert users, all solving the same task, as pure logical structure (model-1), with event-driven goal setting structures (model-2), with parallel goal setting structures (model-3), and with feedback controled goal setting (model-4). With the Petri-net simulator PACE all the models (5(#users)* 4(model-1,2,3,4)=20) were implemented and simulated.

We generated six different task solving sequences with each model, giving a total of 120 simulated sequences. Each simulation stopped either because the net was dead or $N_{sim}=N_{org}$.

To estimate the similarity between the original sequence (e.g. Fig. 1) and each simulated sequence, we used the following procedure that guarantees a metric for *functional equivalence* strictly based only on (re-)produced actions:

1. We numbered consecutively all actions ('transitions', respectively) in Fig. 1 ['G_2' = '1', 'M_3' = '2', 'F_3' = '3', …, 'F_10' = '25']. The number R is the rank-position of each transition [t] in the original behavioural sequence.
2. We assign these numbers to all generated transitions [t] of each simulated sequence. For example, one of the shortest simulated sequences we found, was generated with model-1: ['F_3', '1', 'G_2', 'F_10']. The rank positions R of these four transitions are: ['3', '4', '1', '19']. In general, if a transition appears m times in a original sequence and n times in the simulated sequence, the rules are as following: For the case n<=m, the simulated transitions get the ranks of the corresponding original transitions. For the case n>m, the first elements are handled as in the first case, whereas the rank of the elements m+1..n is m. For example, the sequence ['F_3', '1', 'h', 'M_3', 'h', 'M_3', 'h', 'F_10'] was generated with model-1 and became the rank positions ['3', '4', '23', '2', '24', '2', '24', '19'].
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 action sequence and the original action 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 25). SR is only valid for simulated sequences that fulfil the following condition: $N_{sim} \leq N_{org}$. For example, SR of the sequence ['F_3', '1', 'G_2', 'F_10'] with the rank positions ['3', '4', '1', '19'] is 13%.

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 (C_{cycle}), the similarity ratio (SR) goes up to 100%. There seems to be a positive correlation between C_{cycle} and SR and we can also see that the variance (measured by the standard deviation) decreases continually with increasing C_{cycle}. This result indicates that the predictive power increases from model-1 to model-4.

Table 1. The model complexity (C_{cycle}) and similarity ratio (SR) of model-1, -2, -3 and -4 [std:=standard deviation].

	model-no.			
	1	**2**	**3**	**4**
C_{cycle}: mean ± std	13 ± 5	43 ± 17	57 ± 25	101 ± 43
C_{cycle}: min…max.	6…18	22…68	30…97	55…170
SR: mean % ± std	41 ± 28	66 ± 21	88 ± 11	100 ± 0
SR: min…max. %	3…79	36…98	67…100	100…100
# simulated sequences	30	30	30	30

Three analysis of variances were done with StatView 4.02 (1993). Due to lack of variance we excluded the results of model-4 from the statistical analysis. The independent variables were model type (model-1,-2, and -3) and test person (1..5). The dependent variable was similarity ratio (SR). The only main effect that was significant was that of model type (ANOVA, df=2, F=38.743, p<0.001). The other main effect, test person (ANOVA, df=4, F=0.564, p≤0.690), and the interaction between person and model type (ANOVA, df=8, F=1.068, p≤0.395) were both not significant. So the similarity ratio increases from model-1 to model-3, and this seems to be independent of the test person. We have not tested whether our results depend upon task type. Furthermore, we got a clear correlation between model complexity (C_{cycle}) and similarity ratio (SR) (R=0.660, 95% interval [0.524…0.763], p<0.001).

From this outcome we can conclude two results: (1) model-2 is better than model-1, and model-3 is better than model-2, and (2) the most elaborated model-4 can guarantee a SR of 100%. This result seems to be an interesting confirmation for the theoretical power of the action regulation theory [4].

5 Discussion

Extracting the logical structure (model 1) out of the sequential user process with the folding operation in the Petri net theory, we get a state transition network, which represents the complexity of the task solving process. Repetitive action sequences are modelled over the same path in the Petri net model, thus reducing the measured complexity. If more than one alternative transition (action) from a given state can be activated, the next transition is selected randomly by the Petri net simulator. Therefore the simulated task solving sequences of model-1 are quite different. With model-1 we lost most the relevant information about the concrete task solving process.

With event based goal setting (model-2) we get back some task solving information. Differences between simulation and original action sequence occur, when an action activates more than one alternative goal state, because the original sequence shows different alternatives following that action. This happens mainly with repeated action sequences. Event based goal setting considers only the current action. For the correct selection of alternative decisions we would need more information about the overall task solving strategie and the user knowledge.

The parallel goal setting process (model-3) is not influenced by system interaction. Still simulations do not reach a similarity ratio of 100%. Looking at the simulation process, we see that the cognitive process can set goals without an imediate system interaction. If too many goals are set in advance, the system interaction process might skip over a partial goal sequence and continue with goals set later in the overall sequence. This can be interpreted as a cognitive planning process which is too fast for actual behaviour. If planning is done to far on without action following the planning process, the action process might loose trak of the early planning steps.

In the original sequence (Fig. 1), there is a cyclic behaviour with a learning effect. The user tried twice to see the task relevant information, but without success. The third time he succeeded. This is a typical learning effect based on trial and error. Analysing learning effects with Petri nets was achieved with other strategies [12]. It seems to be difficult to simulate learning effects with event-driven goal setting, because repeated behaviour cannot be integrated in that modelling concept. For the parallel goal setting approach though, repeated behaviour is reflected in the model, with improved results from one to the next repetition.

Event-driven goal setting assures good synchronisation between action and goal setting. Although the knowledge about how to solve the task is included in the model, it tells nothing about a learning effect. The parallel goal setting approach supports modelling of a learning effect. An integration of the advantages from each of the three strategies can be reached with the feedback controled goal setting approach (see model-4).

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, model-3, and model-4) than for the pure logical structure (model-1). So it seems to be possible to develop a completely automatic modelling tool based on a bottom up approach. Moreover, we see that with increasing model complexity (C_{cycle}), the mean value of SR increased and the standard deviation got smaller. Parallel goal setting performed better than event-driven goal setting, and feedback controled goal setting performed better than parallel goal setting. With the parallel and feedback controled goal setting strategies, we were able to include learning effects from the real task solving process.

References

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

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

[3] 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, 21-163).

[4] 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.

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

[6] 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.

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

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

[9] 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 (1992a) 227-236.

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

[11] 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 (268-275). Kyoto University (1996).

[12] 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, 4685-4690). Piscataway: IEEE.

[13] G. 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, 253-288).

Proceedings

6th IEEE International Workshop on

Robot and Human Communication

29 Sept. – 1 Oct. 1997 Sendai

Hikaru Inooka (General Chairperson)

'97 RO-MAN

IEEE Catalog Number: 97TH8299
Library of Congress Number: 97-72653
ISBN: 0-7803-4076-0 (Softbound Edition)
0-7803-4076-9 (Microfiche Edition)