Symbiosis of Human and Artifact
Y. Anzai, K. Ogawa and H. Mori (Editors)
© 1995 Elsevier Science B.V. All rights reserved

From Novice to Expert Decision Behaviour: a Qualitative Modelling Approach with Petri Nets.

Matthias Rauterberg

Work and Organizational Psychology Unit, Swiss Federal Institute of Technology (ETH)
Nelkenstrasse 11, CH-8092 Zurich, Switzerland

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 behaviour described in a finite discrete state space. The sequences of decisions and actions produced by users contain much information about the mental model of this user, the individual problem solution strategies for a given task and the underlying decision structure. We distinguish between (1) the *logical structure*, (2) the *sequential goal structure*, and (3) the *temporal structure*. The analysing tool AMME can handle the recorded decision and action sequences and come up automatically with an extracted net description of the task dependent decision model (the logical structure). This basis model was filled up with additional elements to reconstruct one empirical action sequence of an expert user. Four different models are presented and their predictive power discussed.

1. INTRODUCTION

Learning is a permanent process that changes our long-term knowledge base in an irreversible way. The structure of our long-term memory changes to more complexity and higher abstraction. Learning increases constantly the complexity of the mental model. What mental models are and how they work, is quite unclear. Carroll and Reitman-Olson [2] summarise 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 behaviour. … (2.) Investigate whether people have and use mental models of various kinds. … (3.) Determine the behaviours that would demonstrate the model's form and the operations used on it. … (4.) Explore alternative views of Sequence/Method representations and the behaviour 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 behaviour. … (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 modelling approach that contributes to points (1), (3), (4), (5), (7), (10) and (11). We are primarily interested in a bottom-up, behaviour driven and not in a top-down, theory driven approach.

One of the most elaborated modelling approach is SOAR [5]. Newell [7] describes SOAR as follows: "Soar is … a *symbolic computational system*. … Soar is organised around *problem spaces*, that is, tasks are formulated as search in a space of states by means of operators that produce new states, where operators may be applied repeatedly, to find a desired state that signifies the accomplishment of the task. … Soar is organised entirely as a *production system*, that is, its long-term memory for both program and data consists of parallel-acting condition-action rules. … Soar incorporates a *goal hierarchy*. … Soar learns continuously from its experience by *chunking*, which constructs new productions (chunks) to capture the new knowledge that

Soar developed (in working memory) to resolve its difficulties" ([7] pp. 30-32). Soar is based on impasse-driven learning. "While Soar is performing a task by using the behaviour model in working memory, it is also learning. It is building chunks every time it impasses from one problem space to another, … These chunks constitute the acquisition of knowledge for doing the task" ([7] pp. 62-62). The knowledge generated by chunking and stored in the long-term memory represents only successful trials. Knowledge of unsuccessful attempts is not in memory. Learning in Soar means that long-term memory contains evidence only of the sequence of effective actions. But, what could it mean if the majority of our long-term memory consists only of *unsuccessful trials?* Soar seems to be a typical representative of a top-down, theory driven approach for *error-free skilled behaviour*. Why do we believe that a bottom-up is better than a top-down approach? The answer refers to the following assumption.

Most of the known modelling approaches is 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" ([12] 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 behaviour and the complexity of the assumed mental model. But this assumption seems to be wrong.

Based on the empirical result in [11], that the complexity of the observable behaviour of novices is larger than the complexity of experts, we must conclude that the behavioural 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 behaviour. Learning how to solve a specific task with a given system means that the behavioural 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 [8]. Petri nets can be interpreted in our context by using a suitable pair of concepts for the sets S (signified by a circle '( )') 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 [4]). In this sense, our approach can be subsumed under 'logic modelling', too. 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 operation. This sequence is called a 'process' (see Figure 1). An *elementary process* is the shortest meaningful part of a sequence: (s') -> [t'] -> (s").

If the observable behaviour can be recorded in a complete ...-> (state) -> [transition] -> (state) ->... process description (see Figure 1), then the analysis and construction of the net structure of this process are simple: you have only 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. But, if the observable behaviour can only be recorded in an incomplete (e.g., ...->

(state) -> [transition] -> [transition] ->... or ...-> (state) -> (state) -> [transition] ->...) process description, then the analysis and construction of the net structure of this process are difficulty. 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 (cf. [11]).

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 information of 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 of our example sequence (Figure 1) is the Petri net given in Figure 2.

Measurable features of the behavioural 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 easily done based on a protocol of the user's behaviour 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 [6]. With C_{cycle} we have a useful quantitative metric to measure behavioural complexity. We are discussing the advantages and disadvantages of four different quantitative metrics in the context of an empirical investigation elsewhere (see [10]). 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.

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

The measure C_{cycle} of the model-1 in Figure 2 is [18 - 13 + 1 = 6]; the complexity of the net shown in Figure 2 is six. But, what could this number mean? McCabe [6] 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 behaviour 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 behaviour to estimate MC. We call the complexity of the observable behaviour the 'behavioural complexity'. This 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 (see [9]). For the reconstruction we chose one part of a log file of an expert user (see Figure 1). The whole process of the shown example is based on 12 transitions and 12+1=13 dialog states. The expert user started from the main menu and made with the ASCII-key 'd' the module 'data' active and with the ASCII-key 'a' the routine 'browse'. Pressing the function-key 'F3' he tried to reach a dialog state where he could change the actual data set, but this operation was only possible in the main menu. The system changed to the dialog state 's3' (wrong input state) and responded with the output message "Press space to continue." This message was incorrect implemented, so that the user tried unsuccessfully to leave the 'wrong input state'. Only when he 'found' the function-key 'F9', he could escape from 's3'. The wrong output message was the reason to press three times the space key '_'.

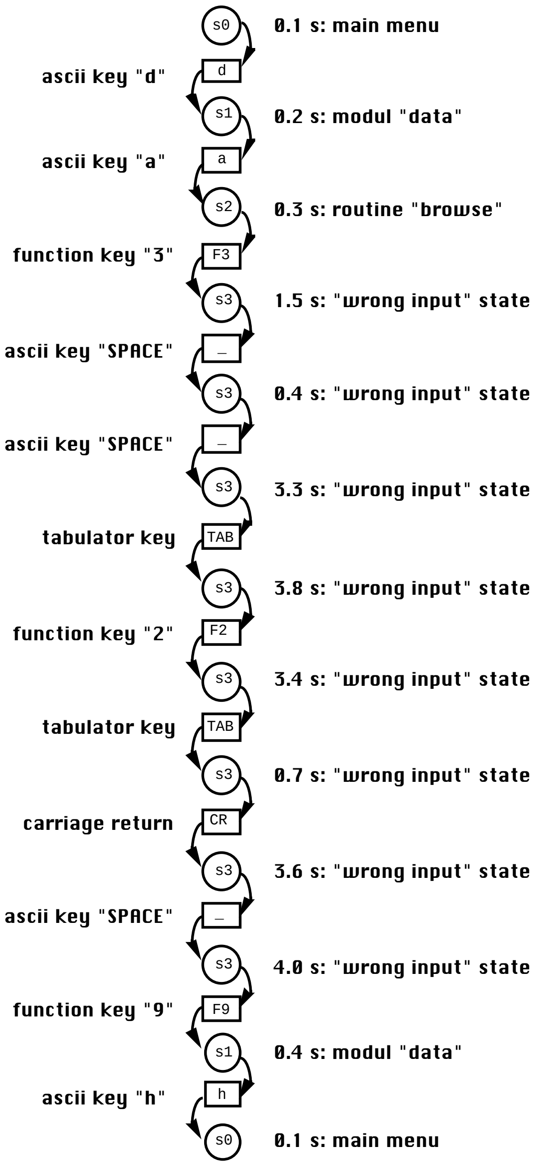

Figure 1. The original behavioural sequence of an expert with a relational database system (cf. [9]).

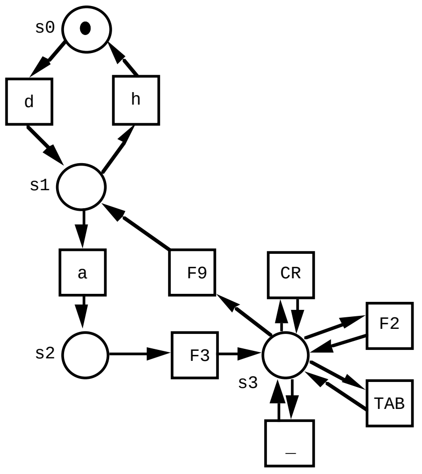

Figure 2. *Model-1*: the pure 'logical structure' of our example sequence in Figure 1.

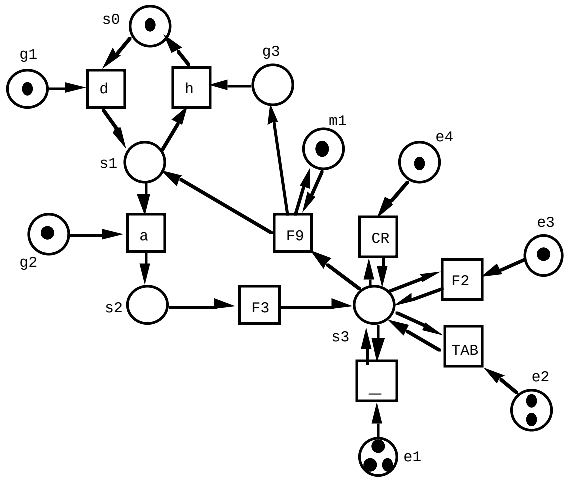

Figure 3. *Model-2*: the model-1 was supplemented with S-elements as *goals* (g1), …, (g3), as *memory places* (m1), and as *extinction places* (e1), (e2), …, (e4).

Mental models consist of three different types of knowledge: (1) the 'pure' logical structure of the task, (2.) the sequential structure of all goals, and (3.) the temporal structure of all operations. The pure logical structure is automatically extracted with our tool AMME (cf. [11]). This net is called model-1. Model-1 does not contain any knowledge about goals and time.

For a first attempt to simulate the sequence in Figure 1 we build model-2 with goals and memory elements. In model-2 (see Figure 3) the S-element (m1) is free for 'recall' of the solution. The three S-elements (g1), ..., (g3) are included in model-2 to simulate the sequential goal structure of the valid transitions outside state 's3'. The four marked S-elements (e1), ..., (e4) are included in the model to simulate the 'extinction'-rate as one aspect of the memory. The number of marks is positively correlated with the extinction-rate of the corresponding transition. If the transition is so often made active as marks are on the S-element then this transition can not be fired anymore. This consequence is a model of learning caused by unsuccessful trials. Model-3 has a complete goal structure for all transitions (see Figure 4). We combined the goal-element of transition '_' with the S-element for the extinction-rate. The activation of transition 'F3' fills this S-element with an additional mark, so that we have then the same extinction-rate as in model-2. Model-4 has exactly the same structure as model-3 (cf. Figure 5). The only difference between model-3 and model-4 is the time delayed transition 'F2'. This delay increases the probability that the transitions '_' and 'TAB' exceeds completely their extinction-rate before transition 'F2' is fired.

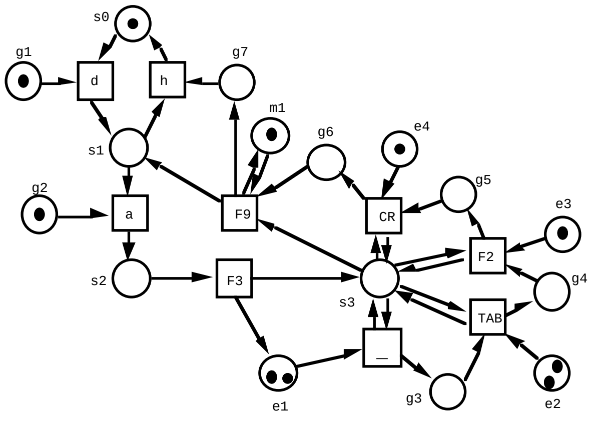

Figure 4. *Model-3*: this Petri-net is equivalent to model-2 with four additional goals to simulate the sequential structure.

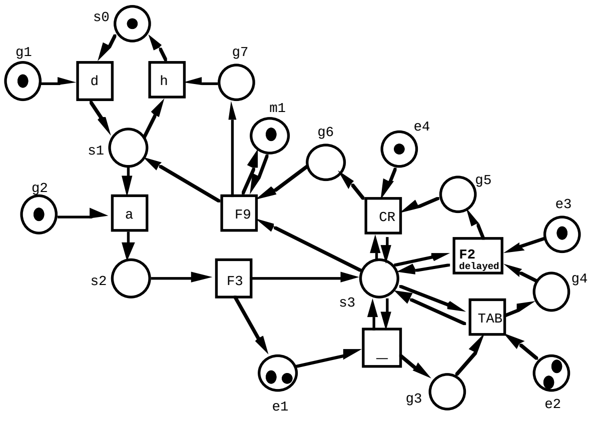

Figure 5. *Model-4*: this Petri-net is equivalent to model-3 with an additional time delay of transition 'F2'.

4. VALIDATION OF THE FOUR MENTAL MODELS

To validate the four different mental models, a simulation study was carried out. With a Petri-net simulator each of our four models was implemented, marked and executed. We generated different task solving sequences with each model. To estimate the difference between the original sequence (cf. Figure 1) and each simulated sequence, we used the following procedure:

1. We numbered consecutively all operations ('transitions', respectively) in Figure 1 ['d' = '1', 'a' = '2', 'F3' = '3', …, 'h' = '12']. The number R is the rank-position of each transition t in the original 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: ['d', 'h']. The rank-positions R of these both transitions are: ['1', '12'] (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 of the difference between the simulated sequence and the original sequence. N is the number of all transitions in a sequence. The maximum of R_{org} is equal to N_{org} (N_{org} in Figure 1 is 12). SR is only valid for simulated sequences that fulfil the following condition: $N_{sim} \leq N_{org}$. For example, SR of the shortest sequence ['1', '12'] is 10%.
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%. Interesting is to note that the structure of model-3 does not distinguish from the structure of model-4. Only the delayed transition 'F2' increases SR by 9%. Additionally to the positive correlation between MC and SR we can see that the variance (measured by the standard deviation) decreases continually. This result indicates that the predictive power increases from model-1 to model-4.

Table 1. The model complexity (MC) and similarity ratios (SR) of model-1, -2, -3, and -4.

	Model-1	Model-2	Model-3	Model-4
MC: absolute value	6	8	13	13
SR: mean	43 %	57 %	86 %	95 %
SR: standard deviation	± 33 %	± 23 %	± 12 %	± 1 %
SR: minimum … maximum	10% … 83%	32% … 93%	68% … 94%	94% … 96%
number of simulated sequences	11	12	8	8

5. DISCUSSION AND CONCLUSION

Three different results are important: (1.) Our assumption that 'learning how to solve a specific task with a given system means that behavioural complexity decreases and cognitive complexity increases' seems to be correct. (2.) We can conclude form the validation results that we must discriminate between the *logical decision structure* of a task, the *sequential goal structure*, and the *temporal structure*. The logical structure of a task can be extracted automatically with our analysing tool AMME, and the complexity of this logical structure can be measured with the McCabe-measure. The temporal structure can be measured with the planning time per operation and transition, respectively. Learning the temporal structure means to accelerate the task solving process. (3.) The results of the goal structures of model-3 and -4 show us that we must take into consideration the knowledge of unsuccessful attempts. Our hypothesis is that the majority of our long-term knowledge consist of inhibitions. To model this aspect means to change the type of the arc between (e1) –> [_] from an *activator* to an *inhibitor* in model-4. In our modelling approach we can not neglect knowledge of unsuccessful trials. One psychological dimension of the goal and time structure seems to be self-confident in doing the right things at the right time avoiding all unsuccessful ways tried before.

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] 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).

[4] 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.

[5] J. Laird, A. Newell and P. Rosenbloom, SOAR: An architecture for general intelligence. Artificial Intelligence 33 (1987) 1-64.

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

[7] A. Newell, Unified theories of cognition and the role of SOAR. In: J. Michon and A. Akyürek, Eds., SOAR: A Cognitive Architecture in Perspective (Kluwer, 1992, pp. 25-79).

[8] 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).

[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 (1992) 227-236.

[10] 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).

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

[12] 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, pp. 253-288).