task analysis
human-computer interaction
logfile recording
Petri-net

IPO–Center for Research on User-System Interaction Eindhoven University of Technology (EUT), Eindhoven, The Netherlands

IHA–Institute for Hygiene and Applied Physiology Swiss Federal Institute of Technology (ETH), Zürich, Switzerland

Task Analysis in Human-Computer Interaction – supporting action regulation theory by simulation

From Matthias Rauterberg and Morten Fjeld

Summary

Task solving processes produced by computer users (observed process) contain much information about their mental task 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 task structure, device model, resp.). This net was extended by goal setting structures (modeling) and can be executed (simulated process). The aim of simulation is functional equivalence between observed and simulated processes. Three modeling strategies, event-driven, parallel goal setting, and regulation-driven goal setting are presented and discussed. Regulation driven goal setting leads to full functional equivalence.

Zusammenfassung

Aufgabenanalyse in der Mensch-Computer Interaktion – Bestätigung der Handlungsregulationstheorie mittels Simulation

Aufgabenanalyse / Mensch-Computer Interaktion / Logfileaufzeichnung / Petri-Netz

Der unmittelbar beobachtbare und automatisch protokollierbare Aufgabenbearbeitungsprozess in der Mensch-Computer Interaktion enthält sehr viel Informationen über das mentale Aufgabenmodell des jeweiligen Benutzers: über seine individuelle Problemlösungsstrategie, sowie die zugrundeliegenden kognitiven Entscheidungsstrukturen für die jeweilige Aufgabe. Das von uns entwickelte Werkzeug AMME analysiert diese Art von protokollierten Aufgabenbearbeitungsprozessen und extrahiert automatisch die aufgabenabhängige Entscheidungsstruktur (auch "logische Aufgabenstruktur" oder "Gerätemodell" genannt) in Form eines Petrinetzes. Dieses Petrinetz wurde nun um Zielsetzungsstrukturen ergänzt, sodass es im Rahmen einer Simulation sinnvoll ausführbar wird. Diese Simulation erzeugt selbst wiederum einen Aufgabenbearbeitungsprozess, der mit dem ursprünglich empirisch beobachteten Prozess auf funktionale Äquivalenz verglichen werden kann. Um eine möglichst gute Übereinstimmung zu erzielen, wurden drei verschiedene Modellierungsstrategien eingesetzt: (1) Ereignis getriggert, (2) Zielbildung als mental paralleler Prozess, und (3) im Sinne der Handlungsregulationstheorie als "vollständige" Aufgabe. Diese dritte Modellierungsstrategie ergibt eine 100%ige funktionale Äquivalenz zwischen beobachtetem und simuliertem Aufgabenbearbeitungsprozess.

1 Introduction

Tasks as goal oriented actions are one of the most important aspects of Human-Computer Interaction (HCI). We can hardly think of any significant activity not being goal related. Understanding how people bring about their deliberate actions can hardly fail to have general, useful value as well as intrinsic, scientific interest. The development of tools to support the study of goal-directed tasks is in the central focus of work psychology.

From former projects we can conclude that it is possible to build tasks models showing high functional equivalence with the original task solving behaviour (Rauterberg 1993). In this paper different analysis and modeling approaches will be formulated (event-driven, parallel and regulation-driven goal setting strategies) and Petri net models will be elaborated, executed and discussed. The functional equivalence is assured by means of a similarity norm measuring the difference between model and real task solving behaviour. The similarity between model output and real task solving behaviour varied with the modeling approach, but seems to be subject independent.

Our approach is founded on the tradition of activity theory. Hence, the notion of complete regulation respectively tasks, is of crucial importance. With the regulation-driven goal setting strategy, we managed to analyse, model and simulate complete tasks. We achieved a convincing similarity between real and modelled task solving behaviour.

The main goal of the presented work is to offer work psychologists a powerful, accessible and user friendly modeling tool. The tool shall offer support in the study of goal-directed task solving behaviour by generating executable, Petri net based, task models.

Contrary to classical approaches, we suggest automatic logging of behavioural data. This approach is well suited for the context of HCI. Each logfile contains the complete task solving process in form of an action sequence. This clear, objective way of data collection gives good reason to investigate task solving processes in the HCI context. It is a frequent problem that sequential data sets are large and complex. Therefore, our approach goes for computer-based generation of task models, based on summaries of sequential data. Our concept has proven its strength in building and using such models for existing HCI situations.

Several methods are developed and used to study task solving processes in HCI:
- questionnaires and interviews (Scott, Osgood & Peterson 1979),
- scoring rationale of observable behaviour (McDaniel & Lawrence 1990),
- protocol analysis based on actions and utterances (Ericsson & Simon 1984),
- formal models (Kieras & Polson 1985) and executable models (Ritter & Larkin 1994).

The normal design cycle to construct an executable model of human behaviour is a task dependent top down approach (de Haan, van der Veer & van Vliet 1991). Given a top down constructed model, it has to be validated with empirical data (Schröder et al. 1990, Ritter & Larkin 1994).

1.1 On task models and modeling in Human-Computer Interaction

Geoffrion (1989) describes modeling processes on four different levels:
- modeling traditions,
- modeling paradigms,
- model classes, and
- specific models.

In the context of this project our modeling tradition is cognitive ergonomics for the design of interactive software systems. "Cognitive ergonomics is oriented towards optimising human-machine systems, according to three types of criteria: characteristics of human cognitive processes, software science knowledge, and

knowledge in diverse work domain technologies" (Green & Hoc 1991, p. 301).

All classes of task models describing user's goal-directed task solving behaviour are our *modeling paradigm.* Activity theory is our theoretical background to look on human task solving behaviour as a goal-directed action sequence.

All interactive processes generated by users over different tasks with a concrete interactive system forms our *model class.* In our context a *specific model* is a concrete user solving a specific task with a real computer system (Peck & John 1992).

There are different formalisms for constructing models of cognitive processes: different kinds of grammars BNF (Reisner 1981), EBNF (Reisner 1984), GOMS (Card, Moran & Newell 1983), Cognitive Complexity Theory (Kieras & Polson 1985), and Soar (Newell 1990) etc. Most of these methods have only a textual representation of the model's content.

Using any of these methods the investigator must always design the pure (more or less 'error free') user and/or task model in a top down approach based on a task analysis (Booth 1991, Ritter & Larkin 1994). Then he can try to prove his process model step by step with 'error free' data (Churchill 1992). This is often difficult, time consuming, and expensive (Kellog & Breen 1990). To overcome these obstacles Ritter and Larkin (1994) describe an approach for building process models that is partially supported by software tools.

Ritter and Larkin (1994) present a methodology for an iterative cyclic modeling process and show how it can become tractable, in particular, by means of good computer tools. It is a methodology for developing and using process models to summarise sequential data, particularly for HCI tasks. This methodology is called Trace Based Protocol Analysis (TBPA). Ritter and Larkin (1994) describe a prototype environment to present Soar/Model-Testing (SMT), giving integrated support to TBPA.

The similarity of a TBPA/SMT model with the empirical data can be controlled by the Dynamical Structure and the Process Rate Graph tools. They both give an immediate graphical impression of what model parts are supported, and not, by protocolled data and indicate where there is potential for model improvement. On the other hand there is little inherent guide for improvement when inspecting the model itself by the Graphic Model Display. Finally, TBPA/SMT offers no direct metric for similarity but only graphical considerations. This most elaborated modeling approach in the context of AI is only partially supported by software tools.

Oberquelle (1984) shows, that several notions of 'model' are in use:

(1) a model of an axiom system, (2) a known system with structure and behaviour analogous to the system under consideration, (3) a prototypical system in the sense of 'model farm', or (4) an abstract description of the relevant aspects of a system.

Class (1) models are normally given in terms of mathematical formulas. A class (2) model can be called an 'exact analogy' in the sense of a special case of a metaphor (e.g., 'neural' networks). Class (3) models are developed in the context of tutorial systems (e.g., Schröder et al. 1990). Class (4) models are useful to classify and explain phenomena of the modelled system. We present in this paper a method to generate class (4) models (we call them "tasks or mental models").

Tasks models consist of different structures to represent knowledge (Dutke 1994). Following the concept of Rasmussen (1986) we have to take into consideration (1) a knowledge based, (2) a rule based, and (3) a skill based level. Rasmussen (1986, p. 140) was not so much interested in the *true* description of some person's mental task models in some specific situation, but rather in a useful description of possible mental task models that can be effective for persons in various tasks, and therefore useful for system design and evaluation.

In the context of research about goal directed task solving behaviour, a quite similar concept was developed, called *activity theory* or *action regulation theory* (von Cranach & Harré 1982, Frese & Sabini 1985, Hacker 1986, Volpert 1994).

1.2 Activity Theory–Action Regulation Theory

Leontyev's three-level schema (Leontyev 1978) describes the sphere of analysis and directs the attention to the transformations going on between three levels (Leontyev 1978, von Cranach 1982, Hacker 1986, Engeström 1991):

(1) motive –> activity, (2) goal –> action, and (3) instrumental conditions –> operations.

(Note the similarity to the model in (Rasmussen 1986): knowledge based, rule based, and skill based level.)

These three levels are organised in a hierarchical structure where the top level of activities includes several actions that are performed by appropriate operations. In a 'pure' objective way only the operational level can be observed and analysed. The goal setting and motivational level must be derived or investigated by indirect methods (e.g., questionnaire, interview, thinking aloud) based on the introspective observations of the investigated subjects.

Action regulation theory (Hacker 1994) offers a coherent body of principles for human-centred task and work design. For Hacker (1986) 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". Therefore, in the context of action regulation theory, the *task* has great importance for the behavioural analysis. The concept of *complete task* calls for particular attention.

Hellpach (1922, p. 27) described the task concept as follows: "A task consists of one's own planning and design–if not design of the task, design of its solution, including choice between various possibilities, decision for one of these, taking responsibility for the decision, the ways of performance, while completing the task the constant, infinitesimal evaluation of success related to the mentally represented goal, and finally, the belief that it was well done". Hellpach's description already included the main elements of the concept of the complete task as defined by Tomaszewski (1981, p. 23): "A complete task contains three basic elements: diagnosis of the starting situation, outline (project) of a new situation and a program of realizing activities. During the course of the task performance, these three elements are supplemented by a continuous diagnosis of the changing situation". These are the basic constituents of Hacker's concept of *complete activity* (Hacker 1986).

Characteristics of complete tasks are, according to the concept of action regulation theory presented here (Ulich et al. 1991):

(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.

Solving a task or problem–in the context of action regulation theory–means that the user has to follow four steps: (1) goal setting, (2) planning and selection of means, (3) perform the selected action, (4) check the outcome against the intended goal.

Incomplete activities–or partialized actions–"lack, to a large extent, possibilities for independent goal-setting and decision-making, for developing individual work styles or sufficiently precise feedback" (Hacker 1987, p. 35). Complete activities (or tasks) offer the possibility of setting goals and sub-goals, as well as of offering decision-making possibilities during the various phases of task completion, and therefore provide latitude of activity or action. Complete activities are therefore becoming fundamental for realising the concept of action regulation. Goals are organised in a hierarchical or heterarchical tree structure (Hacker 1986).

In activity theory, task solving behaviour can be treated at three hierarchical levels. The conscious part takes part at the intermediate action-level whereas the operation-level describes the automatic sequences of behaviour. Both levels are embedded in a context, the uppermost activity-level. Certain behaviour can merge between the action- and the operation-level, called "focus shift" (Nardi 1996). The focus shift can either be seen as learning, where behaviour that used to require conscious attention has been integrated by the subject, and becomes an automatic operation. Such operations are performed fluently without paying conscious attention. Or there's the opposite shift, behaviour merging from the operation to the action level, for instance when a previously know behaviour has to be reactivated. This effort can be required after significant absence from practice but also due to new, modified or erroneously designed artefacts.

"Action Theory seems to be an integrative long-term approach that is still developing especially with the development of hierarchically subordinate sub approaches. Action Theory is still more a heuristic broad-range framework than a final theory. ... The integrative power of Action Theory will bridge some interrelated gaps: the gap between cognition (and knowledge) and action, the gap between cognition and motivation (see goal orientation), and even the gap between basic and more applied approaches ..." (Hacker 1994, p. 113). Our main research interest is bridging both gaps.

1.3 The task domain in HCI

A user of a computer system must learn the 'language', i.e., a set of symbols, their syntax, and operations connected to them, to evoke [inter-]action sequences (the interactive 'processes') related to task and sub task functions. So, the user representation of a system structure is a model of a virtual machine. A 'virtual machine' is defined as a representation of the functionality of a system (functional units and their behaviour). The most important point for the user is the relation between task and machine, and not so much the internal structure of the machine's system. Consequently, the task for the human factors engineer is to model a suitable interface as a representation of the virtual machine which can serve as a possible mental representation for the user.

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.

In the context of human-computer interaction we have to operationalize the theoretical terms: (1) *activity* is a task like 'writing a letter', (2) *action* is an instruction to the computer to do something (e.g., create a new document), and (3) *operations* are a sequence of steps to perform an action (e.g., input of the command 'p', 'r', 'i', 'n', 't', 'CR'). A task solution is given in form of a sequential order of operations. All related operations can be grouped to actions. The complexity of a task depends on the number of different actions/operations, the length of the goal-directed sequence, and the size of the action/operation space (number of possible actions/operations in a given system state).

In the context of human-computer interaction a sequence of operations can easily be described as linear state-transition diagram: s1 –> t1 –> s2 –> t2 –> etc. To model the user's knowledge with finite state-transition nets, Sanderson, Verhage and Fuld (1989) showed that a state space approach works well for the domain of process control. The most powerful environment to simulate state-transition diagrams is a Petri net simulator.

1.4 Modeling with Petri nets

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) (Peterson 1981). The flow relation links only different node types and leaves no node isolated (Petri 1980). 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 '->').

1-save Petri nets consist of the three net elements (S, T, F) and can be interpreted as *causal nets*, modeling causal relations only. If we want to simulate dynamic processes in time, we need a new element to signify activities. This additional element is called a *token*.

Condition-event nets run with unmarked tokens. An event occur if certain preconditions are fulfilled and after the occurrence, certain post conditions hold. A token in the circle of an S-element means that the corresponding condition holds. If we need a distinction between different types of token, we mark them in a characteristic way.

Place-transition nets run with marked tokens.

These three net types are all designated *Petri nets*. Törn (1985) gives us an overview of four important advantages of Petri nets:

- Petri nets are theoretically well founded.
- Petri nets are well suited for describing asynchronous concurrent processes.
- A Petri net simulator tool is simple and easy to learn.
- Both top down and independent modeling (including validation) is possible.

The means-activity interpretation allows one to describe the static structure of a system with several active and passive functional components: means (S) = real or informational entity, and activity [T] = (repeatable) interaction with or action of a system. The flow relation F means: [a] -> (m), the activity *[a]* (e.g., a user action) produces means *(m)* (e.g., a system state); (m) -> [a], activity *[a]* uses means *(m)* (Oberquelle, Kupka & Maass 1983).

The classification of Petri nets starts from three basic net types (Bernadillo & De Cindio 1992):

- Petri Net systems of type-1 are characterised by 'boolean tokens', i.e. places are marked by at most 1 unstructured token.
- Petri Net systems of type-2 are characterised by 'integer tokens', i.e. places are marked by several unstructured tokens – they represent counters.
- Petri Net systems of type-3 are characterised by high-level tokens, i.e. places are marked by structured tokens where information is attached to them.

Bauman and Turano (1986) showed, that Petri nets of type-1 are equivalent to analysis and modeling approaches based on production rules (like CCT of Kieras & Polson 1985). In this sense, the presented approach can be subsumed under 'logic modeling', too.

2 The user driven task analysis approach

The current state of our own work can be described as a new, partially automatic, bottom up approach to construct a formal description of user task and problem solving behaviour. Our methodology generates task and mental models in a clear, objective way. The formalism we use is the net theory. Our method is strictly based on objective recorded, goal-directed action sequences. Figure 1 gives a good overview of our modeling strategy: (1) Collect behavioural data (generated by an unknown cognitive task structure), (2) generate a device model of the behavioural sequence, (3) add goal setting structure, (4) add sequential and temporal information from the behavioural sequence, (5) execute the generated model, and (6) validate functional equivalence between the original behavioural sequence and the model sequence.

The developed method is an integrated modeling environment supported by the tool kit AMME. The automatic and therefore highly objective way of collecting behavioural data can be done in the context of human-computer interaction with the method 'logfile recording'. To record a user's goal-directed action sequence the investigator needs a specially prepared interactive software product (or additional recording software; e.g., the LOG software of Regenass 1995).

2.1 The basic idea

The main operations (relations) between two Petri nets are *abstraction, embedding* and *folding* (Genrich et al. 1980). The *folding operation* is the basic idea of the approach implemented in AMME. 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 a device model (Figure 1, step 2). Each state corresponds to a system context, and each transition corresponds to a user's action or operation.

The aim of the 'folding' operation is to reduce the elements of an empirically observed task solving process to the minimum number of states and transitions, with the reduced number of elements being the logical 'task structure' or 'device model' (as a part of the whole dialog and system structure of the interactive software). Folding a task solving process extracts the embedded net structure and neglects the amount of repetition, the sequential action order and the temporal structure of the process.

2.2 Complete versus incomplete logfiles

Recording the observable behaviour in form of a *complete* ...-> (state) -> [transition] -> (state) - >... process description makes the analysis and construction of the net structure very 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 recorded process.

But, if the observable behaviour can only be recorded in an *incomplete* (e.g., ...-> (state) -> [transition] -> [transition] ->... or ...-> (state) -> (state) -> [transition] ->...)

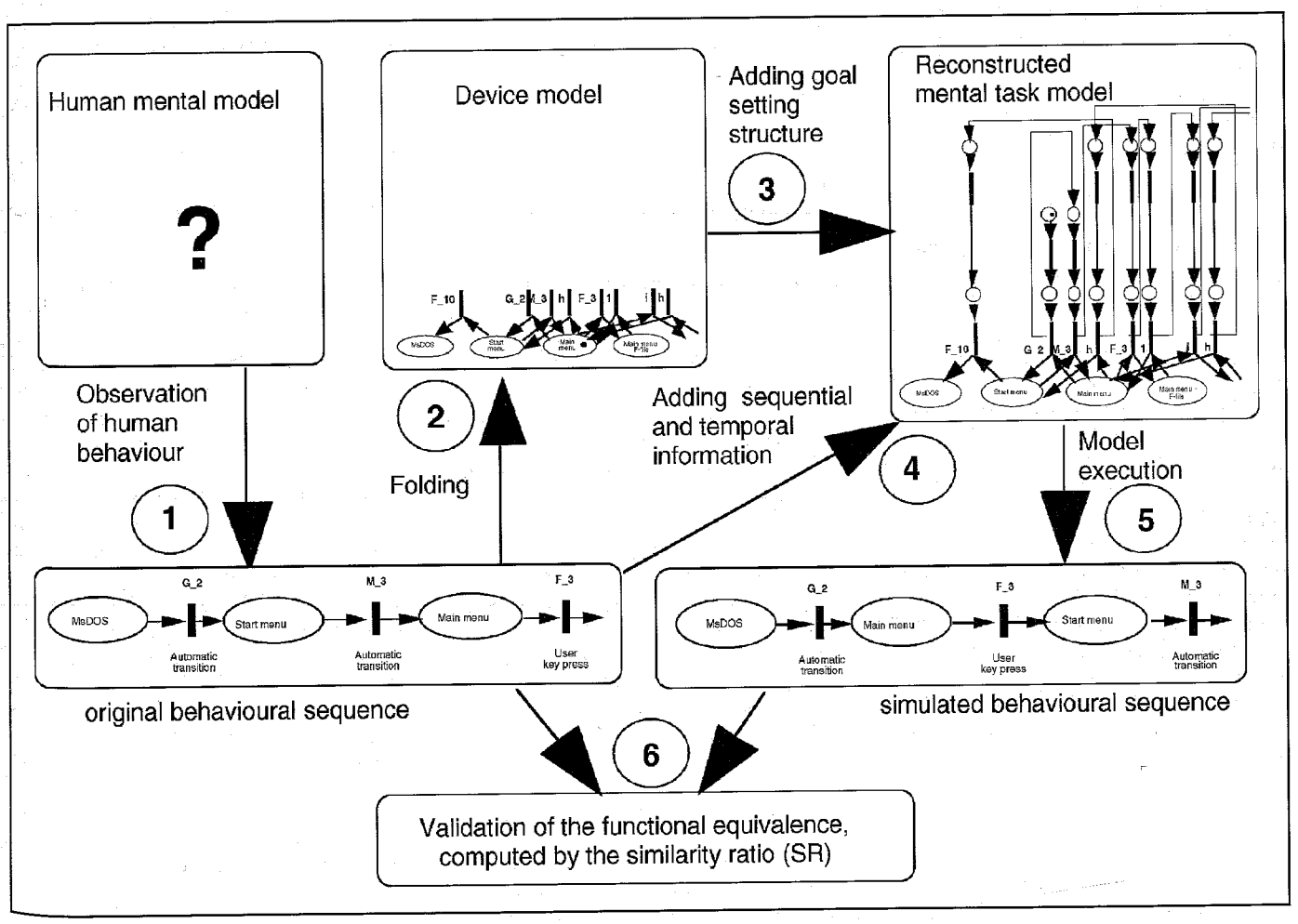

Figure 1: The six steps in our task analysis and modeling concept.

process description, then the analysis and construction of the net structure is difficult. You have to find out the correct state (transitions, resp.) between both transitions (states, resp.). Unfortunately, this is the most frequent case in practice, because most of all existing interactive software products do not have a complete internal representation of all possible system states or transitions.

For all interactive software products with no internal state representation we need automatic tool support. For these cases we developed the tool kit AMME, that gives us the possibility to analyse any processes with an incomplete process description, that are generated by finite state transition nets. The price we have to pay is the definition of a complete state transition list of all relevant transitions beforehand (= the complete internal state-transition representation of the software product).

2.3 The tool AMME

The current stage of development of our tool AMME can be described as follows; the whole system AMME consists of seven different sub-programs (see Rauterberg 1993, 1996b, Rauterberg & Fjeld 1997):

(1) An interactive *dialog system* with a logging feature, generating the task solving process description. This description should be automatically transformed to a logfile with an appropriate syntactical structure. However, a logfile can also be hand written by the investigator (e.g. based on protocols of observations).

(2) The *net generation program* AMME, analysing the interactive process sequence, extracting the minimal net structure (device model) and calculating different quantitative measures of the generated net. AMME needs three input files: (i) a complete system description on an appropriate level of granularity, (ii) the interactive process description, and (iii) a support file for the graphic output ("defaultp.ps" is part of the tool kit). AMME produces five different output files: (i) a *protocol file* (*.pro") with different quantitative measures of the process and of the extracted net, (ii) a *Petri net description file* ("*.net") in a readable form for the Petri net simulator PACE, (iii) a plain text file ("*.ptf") with the *connectivity matrix* for KNOT, (iv) a plain text file ("*.mkv") with the *probability matrix* for the Marcov chain analysing software SEQUENZ, and (v) a PostScript file ("*.ps") to print the *net graphic* for pattern matching 'by hand'.

(3) The *Petri net simulator* PACE is implemented in Smalltalk 80 and consists of a graphical editor and an interactive simulator with graphical animation. PACE can deal with hierarchical nets, refinement of T- and S-elements, timed Petri nets as well as stochastic Petri nets. Smalltalk 80 standard classes are available for token attributes.

(4) The *net analysing program* KNOT, computing the similarity between pairs of nets. With the multidimensional scaling (MDS) module of KNOT (Kruskal nonmetric MDS algorithm) we can compute a MDS solution for any set of nets.

(5) The Marcov analysing software SEQUENZ, offering a method to compare user triggered sequences. These sequences are transformed into first-order Marcov-chains. Similarity between such lattices can be directly obtained by summation of the differences between latticecells. Also the resulting distances provide an input to the MDS models.

(6) Any Postscript interpreter (e.g., Ghostscript) that can read and print the output file *.ps.

(7) Any text processing software can handle the pure ASCII files, *.pro.

The current version of AMME is restricted to process descriptions that can be traced in a *finite, discrete state space* with an upper limit of different states. A further restriction is the constrained syntax of the logfiles that serve as input for AMME. To transform a given logfile to the appropriate form, several tools can be applied; like Coco/R, YACC, or any other tool that can convert text strings into other formats. The actual version of AMME is freeware and available for IBM or compatible PCs (with MsWindows =3.0) via Internet.

2.4 Measuring task complexity

Measurable features of the task solving process are: task solving time (#TST), total number of states (#AS) and of transitions (#TT) used (see Rauterberg 1992a, 1995b, 1996a). These measurements can be easily done based on the analysis of the logfile itself. But, logfiles *must* be analysed with a tool like AMME to get the following two metrics: (1) number of different states (#DS) and (2) number of different transitions (#DT). These both numbers are the basis to calculate net complexity. We investigated in (Rauterberg 1992a) the advantages and disadvantages of four different quantitative metrics. With the Ccycle metric we found a useful quantitative metric to measure complexity.

The complexity measured with Ccycle is defined by the difference of the total number of connections (#T: transition) and the total number of states (#S: state). The parameter P is a constant to correct the result of Formula 1 in the case of a sequence (#T – #S = -1); the value of P in our context is one.

$$Ccycle = \#T - \#S + P$$
with #S = #T and P=1
(Formula 1)

We interpret Ccycle as the *number of linear independent paths* through the net. Other interpretations of Ccycle are *number of holes* in a net or *number of alternative decisions* carried out by users. If #S is bigger than #T then we have to calculate Ccycle in a slightly different form. #F is the number of arrows in the net.

$$C'cycle = \#F - (\#T + \#S) + P$$
with #S > #T and P=1
(Formula 2)

In the investigation of Rauterberg (1992a) novices and experts were classified and selected by their amount of expertise with electronic data processing. This expertise was measured with an 115-item questionnaire and with structured interviews. The novice group (N=6) was instructed for 1.5 hours in handling the database system. The expert group (N=6) had 1,740 hours of experience in operating the same database system. Their total computer experience of about 7,500 hours was the result of their daily work using different types of computers and software systems. The duration of the actual task solving session was about 30 minutes. Each keystroke with a time stamp was recorded in a logfile. Each user needed about 50 minutes for the whole task solving process (4 tasks, individual sessions). The behavioural complexity (#BC) of each completed task solving process is measured as follows:

#BC = Ccycle*(complete task solution)*
(Formula 3)

A significant difference in #BC between novices and experts was found (N = 24, #BCnov = 17 ± 6, #BCexp = 12 ± 5; df=1, F = 10.3, $p \leq .003$). This important result indicates, that the complexity of the observable behaviour correlates *negatively* with the complexity of the cognitive structure (the 'mental model'). We explain this result as follows: 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 strategies. Learning how to solve a specific task with a given system mean, that #BC decreases and the complexity of the mental model increases (Rauterberg 1993, Rauterberg & Aeppli 1995).

2.5 Measuring 'routinization'

A routine task can be identified by a large process description (= long logfile; e.g.

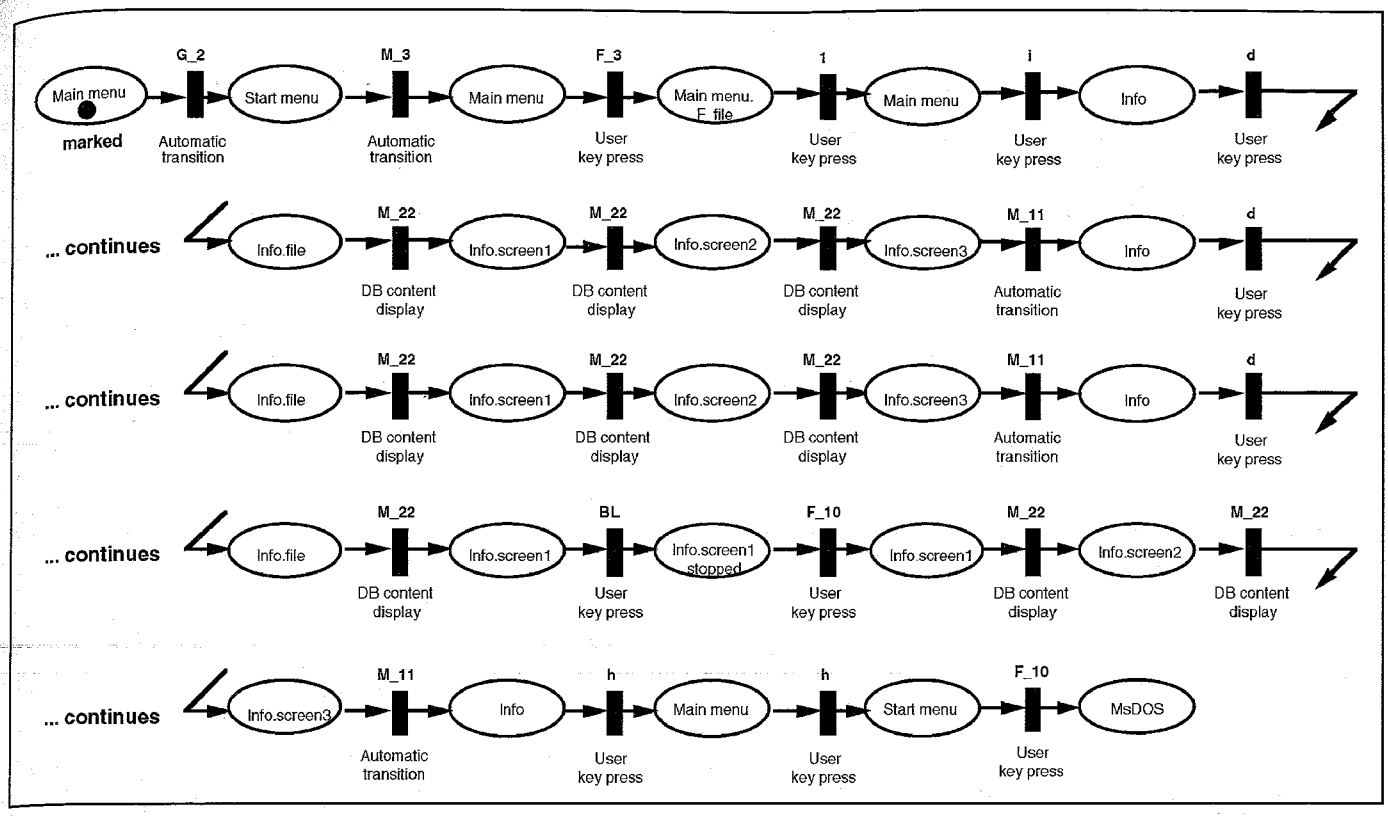

Figure 2: The original task solving sequence of an expert with a relational database system.

#TT ⩾ 10) and a small size of the embedded net structure (e.g. #DT ≤ 10) (see Rauterberg 1993): The user is always running in loops and using the some system operations to solve the task; we designate this kind of tasks 'routine tasks'. The ratio of the total number of transitions in the process description (length of the logfile = #TT) to the number of different transitions in the folded net (#DT) is a good measure of the "degree of routinization" (#R).

#R = #TT / #DT
(Formula 4)

This measure combines the information of the total amount of repetition of each transition (#TT) with the information of all necessary transitions (#DT). The information of the sequential order of all transitions is neglected.

2.6 Measuring 'personality styles'

To measure the dimension of 'action versus state orientation' all users filled out a personality questionnaire (see Rauterberg 1992a, 1994). The independent variables were (1) the level of expertise (novices versus experts) and (2) the four different tasks. All user actions were protocolled with time stamps in logfiles. With AMME we could extract all task dependent dialog states from 48 logfiles. The dependent variables are: (1) total task solving time (#TST) and (2) number of different states per Petri net (#DS). The ratio of #TST divided by #DS is the mean duration time per state. This average of the duration or dwell time per state can be interpreted as *user's thinking time* (#MTT) to plan the next action (see Formula 5).

#MTT = #TST / #DS
(Formula 5)

We aggregated all #MTT's over the four tasks and correlated this global value of #MTT per user with all scale scores of the 'action versus state orientation' questionnaire. We found a *negative* correlation between #MTT and scale-1: 'success leads to action orientation in thinking' (see Kuhl 1981). This significant correlation ($R = -.75$; $p \leq .005$; $N=12$) means, that users with high scores in 'action orientation in thinking' caused by success have a short dwell time per state, and vice versa. Experts are–measured with the questionnaire–more 'action oriented' than novices ($N=12$, $df=1$, $F=11.40$, $p \leq .007$). We can conclude that state oriented persons need–on average–more time per dialog state, than action oriented persons.

3 Reconstruction of the mental task model

3.1 Observation of human behaviour

Based on the data of the empirical investigation of Rauterberg (1992b), we used log files of five different expert users solving the same task to reconstruct human mental models (see step 1 in Fig. 1). The user's task was to find out how many data records there are in a given data base consisting of file A, file B and file C.

An example of the task solving process of an expert user is presented here (Fig. 2). First, the system goes from (Main menu) to (Start menu) with system transition ['G_2'], where it opens the database and automatically returns 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 ['I'] brings him to the (Info) module. Pressing ['d'], all database information is displayed sequentially on three different screens (3* ['M_22']) and the system automatically goes to (Info) with ['M_11']. But part of the *task relevant* information is visible only on the first screen, which is overwritten by the two following ones. Trying once more to see all information, the user tries ['d'] again, but the same reoccurs. After a third time pressing ['d'], the user correctly follows up with a blank ['BL'] to interrupt the scrolling. Hence, the first part of the task relevant information stays on the screen. Pressing ['F_10'], the rest of the information is displayed (2* ['M_22']) and the system automatically goes to (Info) by ['M_11']. Pressing ['h'] twice brings the user to (Main menu) and then to (Start

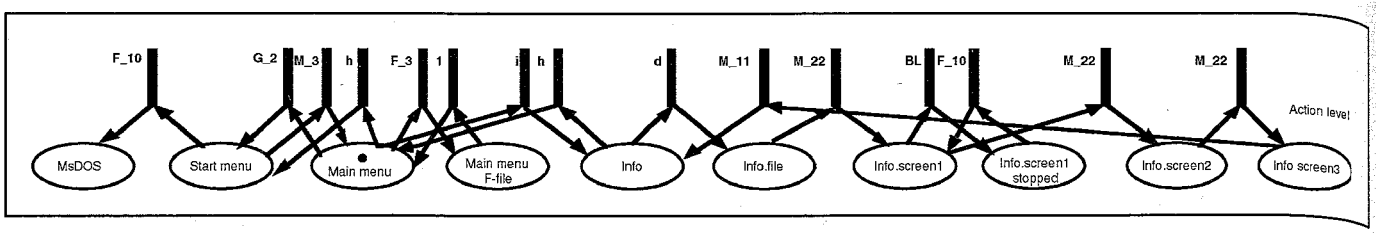

Figure 3: The device model of the task solving sequence in Fig. 2.

menu). Pressing ['F_10'] brings the user to (MsDOS).

The logged sequence contains three different types of knowledge: (1) the pure logical structure of the task (the device model; see Fig. 3), (2) the sequential structure of all goals, and (3) the temporal structure of all actions.

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; see characteristics of complete tasks, chapter 1.2. Activity Theory), (2) the goal instanciation level (corresponding to B), (3) the action level (corresponding to C), and (4) the feedback level (corresponding to D).

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.

3.2 Folding

Modeling approach-1: The pure logical structure is automatically extracted with our tool AMME (see step 2 in Fig. 1). This net is called device model (see Fig. 3). The device model does not contain any knowledge about goals and time. It 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). This device model represents the action level (see Fig. 4).

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 the device model. 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.

3.3 Adding goal setting task structure, sequential and temporal information

In a first modeling approach, *event-driven goal setting*, we added a minimal structure to the device model, to increase the *functional equivalence* between the simulated behavioural sequence and the original behavioural sequence (see step 3 and 4 in Fig. 1).

Modeling approach-2: Each action (event, resp.) on the action level *directly* activates a mental process on the cognitive level. The mental process sets on the goal instanciation level the corresponding 'goal' for the next action. There is a direct synchronisation between the action level and the upper levels.

A big problem of this modeling 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.

Modeling approach-3: The parallel goal setting process was introduced by von Cranach's discussion of plans, 'anticipating representations', and intention (Cranach & Harré 1982). For each goal at the goal instanciation level to be set, an anticipated counterpart must be set beforehand on the cognitive level. Parallel goal setting means that the mental goal setting process

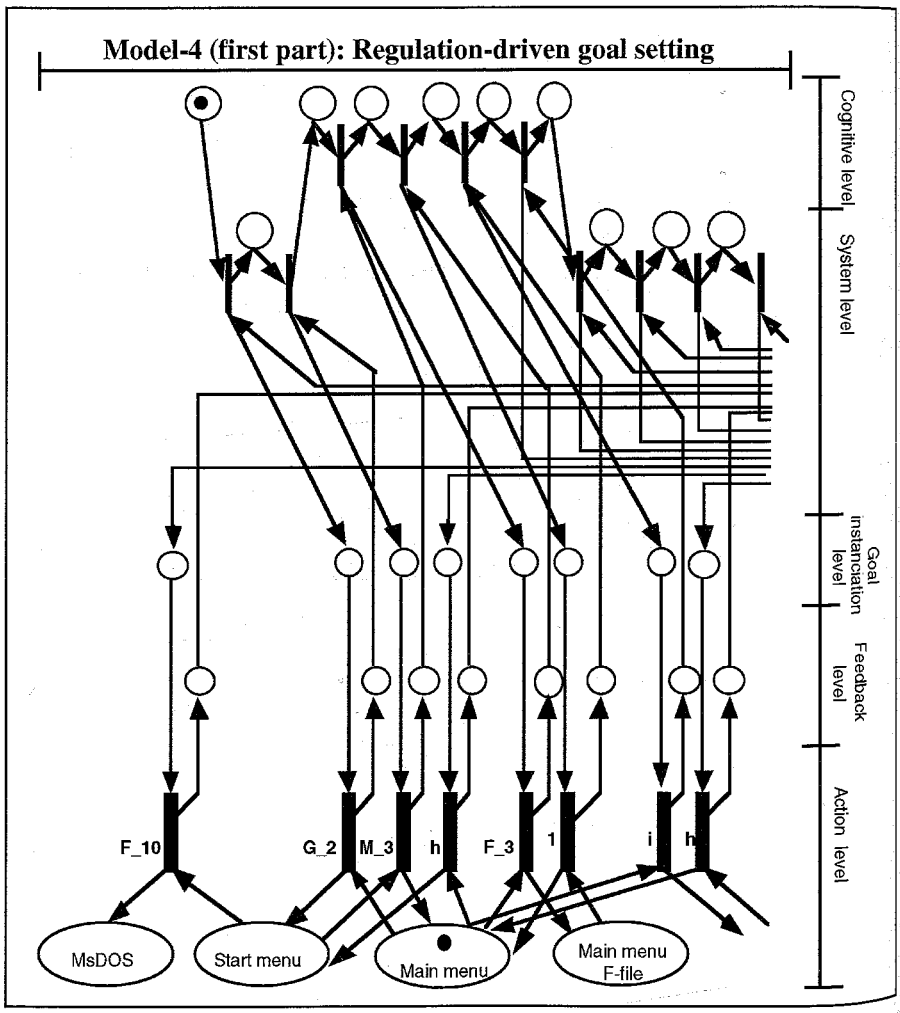

Figure 4: Regulation-driven goal setting: this Petri-net is equivalent to parallel goal setting with added structure for feedback from action level to higher levels.

is running parallel to the action level, on which the dialog actions are executed. 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 modeling approach.

Modeling approach-4: Regulation-driven goal setting aims to reach a complete synchronisation between the action level and the cognitive level. We added feedback loops to the modeling approach-3 (see Fig. 4). This upward directed structure guarantees that the cognitive goal setting level can not be faster than the observable action process on the action level.

The modeling approach-4 generates the most complex model compared with the other ones. Figure 5 shows that it implements a complete activity cycle. 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 with parallel goal setting: modeling approach-3). The next step is to find out which model approach can reach the maximum of 100% in functional equivalence with the original behavioural sequence generated by a human expert.

A pseudocode modeling algorithm for the implementation of regulation-driven goal setting can be described as follows:

```
behavioural_sequence :=
GENERATE_SEQUENCE_MODEL (logfile);
device_model := GENRATE_DEVICE_MODEL
(logfile);
FOR EVERY transition IN device_model DO
  INSERT input_place;
  INSERT output_place;
  CONNECT input_place TO transition;
  CONNECT transition TO output_place;
END;
FOR transition := FIRST_TRANSITION
(behavioural_sequence) TO
  LAST_TRANSITION
  (behavioural_sequence) DO
  next_transition :=
NEXT_TRANSITION_IN_BEHAVIOURAL_SEQUENCE
  (transition);
  device_transition :=
EQUIVALENT_TRANSITION_IN_DEVICE_MODEL
  (transition);
  CONNECT actual_transition TO
  INPUT_PLACE_OF
  (device_transition);
  CONNECT OUTPUT_PLACE_OF
  (device_transition) TO
  next_transition;
END;
first_state := FIRST_STATE
(behavioural_sequence);
INSERT_TOKEN IN first_state;
INSERT_TOKEN IN EQUIVA-
LENT_STATE_IN_DEVICE_MODEL
(first_state);
```

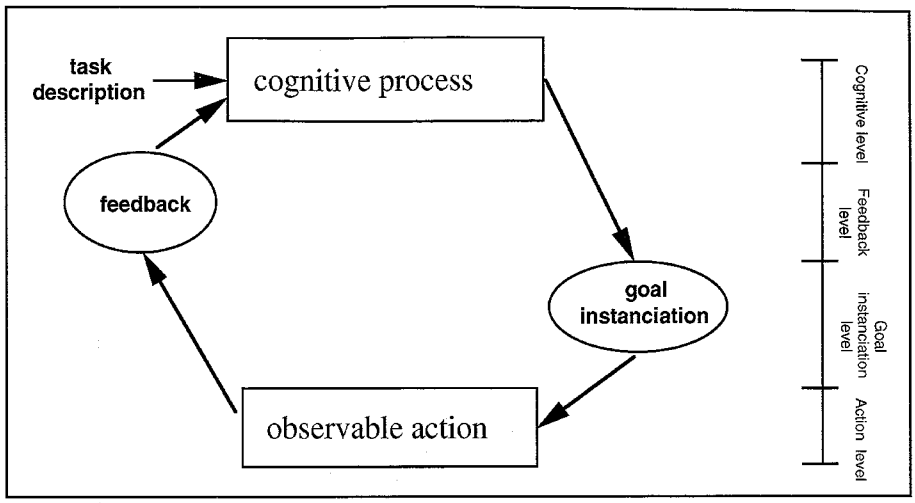

Figure 5: A complete activity cycle as the theoretical framework for regulation-driven modeling.

3.4 Task model execution

To validate the different goal setting strategies, a simulation study of the modeling approache-1, -2, -3 and -4 was carried out (see step 5 in Fig. 1). We modelled the task solving processes of five expert users, all solving the same task, as device model (modeling approach-1), with event-driven goal setting structures (modeling approach-2), with parallel goal setting structures (modeling approach-3) and with regulation-driven goal setting structures (modeling approach-4). With the Petri-net simulator PACE all the models (5(#subjects) * 4(model-1, -2, -3)=20) were implemented and simulated. We generated six different simulated behavioural sequences with each model, giving a total of 120 sequences. Each simulation stopped either because the net was dead or Nsim=Norg.

3.5 Validation of the functional equivalence

To estimate the similarity between the original sequence (see step 6 in Fig. 1) and each simulated sequence, we used the following procedure that guarantees a *functional equivalence* strictly based only on (re-)produced actions:

1. Number all transitions ['G_2' = '1', 'M_3' = '2', 'F_3' = '3', ..., 'F_10' = '25'] in Fig. 2 consecutively. The number R is the rank-position of each transition [t] in the original behavioural sequence.

2. 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 modeling approach-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 $n \leq m$, 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.

3. Calculate a 'similarity ratio' (SR; see Formula 6). 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 Rorg is equal to Norg (Norg in Fig. 2) is 25). SR is only valid for simulated sequences that fulfil the following condition: Nsim ≤ Norg. SR of the above example ['F_3', '1', 'G_2', 'F_10'] is 13%.

4. Average 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 task model (Ccycle), the similarity ratio (SR) tends to 100%. We can also see that the

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

Formula 6

Table 1: The model complexity (Ccycle) and similarity ratio (SR) of the modeling approaches-1, -2, -3 and -4 [std:=standard deviation].

	modeling approach no. 1	modeling approach no. 2	modeling approach no. 3	modeling approach no. 4
Ccycle: (mean ± std):	13 ± 5	43 ± 17	57 ± 25	101 ± 43
Ccycle: (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	5*6 = 30	5*6 = 30	5*6 = 30	5*6 = 30

standard deviation of SR decreases continually with increasing Ccycle.

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

4 Discussion and Conclusion

From the outcomes of our statistical analysis we can conclude two results: (1) event-driven goal setting is better than the pure device model, and parallel goal setting is better than event driven goal setting, and (2) the most elaborated task modeling approach, regulation-driven goal setting, can guarantee a SR of 100%. This result seems to be an interesting confirmation for the theoretical power of the action regulation theory (Hacker 1994).

In the original sequence (Fig. 2), 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 (Rauterberg & Aeppli 1995). It seems to be difficult to simulate learning effects with event-driven goal setting, because repeated behaviour cannot be integrated in that modeling approach. For the parallel goal setting and regulation driven 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 level and cognitive level. Although the knowledge about how to solve the task is included in the model, it tells nothing about learning effects. The parallel goal setting approach supports modeling of learning effects. An integration of the advantages from each of the strategies can be reached with the regulation-driven goal setting approach (see modeling approach-4).

We can conclude from the validation results that the SR value is considerably higher for the models with added goal setting (modeling approach-2, -3, and -4) than for the pure logical structure (device model). So it seems to be possible to develop a completely automatic task modeling tool based on a bottom up approach. Moreover, we see that with increasing task model complexity (Ccycle), the mean value of SR increased and the standard deviation got smaller. Again, parallel goal setting performed better than event-driven goal setting, and regulation-driven goal setting performed better than parallel goal setting. With the parallel and regulation-driven goal setting strategies, we were able to include learning effects from the real task solving process. Regulation-driven goal setting seems to be the most promising task analysis and modeling approach and should be the basis of further developments.

5 References

Ackermann, D. (1987) Handlungsspielraum, Mentale Repräsentation und Handlungsregulation am Beispiel der Mensch-Computer-Interaktion. PhD Thesis (Univ. of Bern).

AMME (1996) *Download address:* URL: http: // www.ifap. bepr.ethz.ch/~rauter/amme.html

Bauman, R. & Turano, T.A. (1986) Production based language simulation of Petri nets, Simulation, 47: 191-198.

Bernardinello, L. & De Cindio, F. (1992) A survey of Basic Net Models and Modular Net Classes. In: W. Bauer, Ed., Lecture Notes in Computer Science 609 (Springer).

Booth, P.A. (1991) Errors and theory in human-computer interaction. Acta Psychologica, 78:69-96.

Card, S.K., Moran, T.P. & Newell, A. (1983) The psychology of human computer interaction. (Erlbaum).

Churchill, E. (1992) The formation of mental models: are 'device instructions' the source?. In: G.C. van der Veer, M.J. Tauber, S. Bagnara & A. Antalovits, Eds., Human-Computer Interaction: Tasks and Organisation (CUD, pp. 3-16).

Cranach, von M. & Harré, R. (1982, eds.) The analysis of action (Cambridge University Press).

de Haan, G., van der Veer, G.C. & van Vliet, J.C. (1991) Formal modeling techniques in human-computer interaction. Acta Psychologica, 78:27-67.

Dutke, S. (1994) Mentale Modelle: Konstrukte des Wissens und Verstehens (Arbeit und Technik Band 4) (Verlag für Angewandte Psychologie).

Engeström, Y. (1991) Activity Theory and Individual and Social Transformation. Activity Theory 7/ 8:6-17.

Ericsson, K.A. & Simon, H.A. (1984) Protocol analysis (MIT Press).

Frensch, P. & Funke, J. (1995, eds.) Complex Problem Solving: The European Perspective (Erlbaum).

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

Geoffrion, A.M. (1989) Integrated modeling systems. Computer Science in Economics and Management, 2:3-15.

Green, T.R.G. & Hoc, J.M. (1991) What is cognitive ergonomics?. Le Travail humain, 54:291-304.

Hacker, W. (1986) Arbeitspsychologie (Huber).

Hacker, W. (1987) Software-Gestaltung als Arbeitsgestaltung. In K.-P. Fähnrich (ed.) Software-Ergonomie. State of the Art 5 (Oldenbourg, pp. 29-42).

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

Hoffmann, J. & Rauterberg, M. (1994) From novice to expert decision behaviour: an automatic modeling approach with Petri nets. In: K. Pawlik (Hrsg.) Abstracts des 39. Kongreß der Deutschen Gesellschaft für Psychologie in Hamburg 1994. (Band I, S. 294-295), (Hogrefe).

Hoppe, H.U. (1988) Task-oriented parsing – a diagnostic method to be used by adaptive systems, In: Proc. of CHI'88, ACM: Washington, pp. 241-247.

Interlink, Inc. (1991) The KNOT software (P.O. Box 4086 UPB, Las Cruces, NM 88003, USA).

Johnson, S. C. (1975) YACC – yet another compiler-compiler, Technical Report No. 32 (Bell Laboratories).

Jones, C. (1993) An integrated modeling environment based on attributed graphs and graph-grammers. Decision Support Systems, 10, 255-275.

Kellog, W.A. & Breen, T.J. (1990) Using Pathfinder to evaluate user and system models. In: R. W. Schvaneveldt, Ed., Pathfinder Associative Networks (Ablex, pp. 179-195).

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

Kuhl, J. (1981) Motivational and functional helplessness: the moderating effect of state vs. action orientation, Journal of Personality and Social Psychology 40:155-170.

Leontyev, A.N. (1978) Activity, consciousness, and personality (Prentice Hall).

McDaniel, E. and Lawrence, C. (1990) Levels of cognitive complexity: an approach to the measurement of thinking (Springer).

Mössenböck, H. (1990) Coco/R – a generator for fast compiler front-ends, Technical Report No. 127 (Computer Science Department of the ETH Zürich).

Nardi, B. (1996) Context and Consciousness, Activity Theory and Human-Computer Interaction (MIT Press).

Newell, A. (1990) Unified Theories of Cognition. (Harvard University Press).

Oberquelle, H. (1984) On models and modeling in human-computer co-operation. In: G.C. van der Veer, M. J. Tauber, T.R.G. Green & P. Gorny, Eds., Lecture Notes in Computer Science 178 'Readings on Cognitive Ergonomics' (Springer, pp. 26-45).

Oberquelle, H., Kupka, I. & Maass, S. (1983) A view of human-machine communication and cooperation. International Journal of Man-Machine Studies, 19:309-333.

PACE (1997) *Distributor address:* IBE Ingenieurbüro, Software und Simulation Engineering, Postfach 1142, D-85623 Glonn Germany.

Peck, V.A. & John, B.E. (1992) The Browser model: A computational model of a highly interactive task. Proceedings of the CHI'92 Conference on Human Factor in Computing Systems (ACM, pp. 165-172).

Peterson, J.L. (1981) Petri net theory and the modeling of systems (Englewood Cliffs).

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

Polson, P.G. & Kieras, D.E. (1985) A Quantitative Model of the Learning and Performance of Text Editing Knowledge, In: Borman, L. & Curtis, B. (Eds.), Human Factors in Computing Systems: CHI '85, pp. 207-212

Rasmussen, J. (1986) Information Processing and Human-Machine Interaction (North-Holland).

Rauterberg, M. & Aeppli, R. (1995) Learning in man-machine systems: the measurement of behavioural and cognitive complexity. In: Proceedings of IEEE International Conference on Systems, Man and Cybernetics 'Intelligent Systems for the 21st Century'. (Vol. 5, IEEE Catalog No. 95CH3576-7, pp. 4685-4690).

Rauterberg, M. & Aeppli, R. (1996) How to measure the learning process in man-machine systems. In: A. Özok & G. Salvendy (eds.) Advances in Applied Ergonomics. (USA Publishing, pp. 312-315).

Rauterberg, M. & Fjeld, M. (1997): An Analysing and Modelling Tool Kit for the Human-Computer Interaction. In M.J. Smith, G. Salvendy & R.J. Koubek (Eds.), Design of Computing Systems: Social and Ergonomic Considerations (HCI'97) (Advances in Human Factors/Ergonomics, Vol. 21B), pp. 589-592. Amsterdam: Elsevier.

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

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

Rauterberg, M. (1993) AMME: an Automatic Mental Model Evaluation to analyze user behaviour traced in a finite, discrete state space. Ergonomics 36(11):1369-1380.

Rauterberg, M. (1994) Action vs. state orientation: an empirical validation in man-computer interaction. In: K. Pawlik (Hrsg.) Abstracts des 39. Kongreß der Deutschen Gesellschaft für Psychologie in Hamburg 1994. (Band II, S. 551), (Hogrefe).

Rauterberg, M. (1995a) Human information processing in man-machine interaction. In: A. Grieco, G. Molteni, E. Occhipinti & B. Piccoli (Eds.) Work with Display Units 94. (North-Holland, pp. 221-226).

Rauterberg, M. (1995b) From novice to expert decision behaviour: a qualitative modeling approach with Petri nets. In: Y. Anzai, K. Ogawa & H. Mori (Eds.) Symbiosis of Human and Artifact: Human and Social Aspects of Human-Computer Interaction. (Advances in Human Factors/Ergonomics, Vol. 20B), (Elsevier, pp. 449-454).

Rauterberg, M. (1995c) About a framework for information and information processing of learning systems. In: E. Falkenberg, W. Hesse & A. Olive (eds.) Information System Concepts – Towards a consolidation of views. (Chapman&Hall, pp. 54-69).

Rauterberg, M. (1996a) How to measure cognitive complexity in human-computer interaction. In: R. Trappl (ed.) Cybernetics and Systems '96 (Vol. 2). (Austrian Society for Cybernetic Studies, pp. 815-820).

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

Regenass, A. (1995) LOG version 5.1 (Universität Bern, Institut für Psychologie).

Reisig, W. (1992) A Primer in Petri Net Design (Springer).

Reisner, P. (1981) Formal grammar and human factors design of an interactive graphics system. IEEE Transactions on Software Engineering, SE-7:229-240.

Reisner, P. (1984) Formal grammar as a tool for analyzing ease of use. In:. J.C. Thomas & M.L. Schneider, Eds., Human Factors in Computing Systems (Ablex, pp. 53-78).

Ritter, F. & Larkin, J. (1994) Developing process models as summaries of HCI action sequences. Human-Computer Interaction 9:345-383.

Sanderson, P.M., Verhage, A.G. & Fuld, R.B. (1989) State-space and verbal protocol methods for studying the human operator in process control. Ergonomics, 32:1343-1372.

Schiele, F. & Hoppe, H.U. (1990) Inferring task structures from interaction protocols, In: D. Diaper & al. (Eds.), Human-Computer Interaction – INTERACT '90, Elsevier Science Publishers B.V. (North-Holland), pp. 567-572.

Schmid, U. & Meseke, B. (1991) Deskription und Analyse komplexer Verhaltenssequenzen: Benutzerstrategien beim Arbeiten mit CAD-Systemen, Zeitschrift für experimentelle und angewandte Psychologie, 38:307-320.

Schröder, O., Frank, K.-D., Kohnert, K., Möbus, C. & Rauterberg, M. (1990) Instruction-based knowledge for a functional, visual programming language. Computers in Human Behavior 6(1):31-49.

Schvaneveldt, R.W. (1990, Ed.) Pathfinder associative net works – studies in knowledge organization (Ablex Publ).

Scott, W.A., Osgood, D.W. & Peterson, C. (1979) Cognitive structure: theory and measurement of individual differences (Wiley).

Tomaszewski, T. (1981). Struktur, Funktion und Steuerungsmechanismen menschlicher Tätigkeit. In T. Tomaszewski (ed.) Zur Psychologie der Tätigkeit (Deutscher Verlag der Wissenschaften, pp. 11-33).

Törn, A.A. (1985) Simulation nets, a simulation modeling and validation tool, Simulation, 45:71-75.

Trompedeller, M. (1995) A Petri Net Classification and related Tools (URL http:// www.dsi.unimi.it/ Users/ Tesi/ trompede/ petri/ home.html)

Ulich, E. (1994, 3rd edition) Arbeitspsychologie (Poeschel).

Ulich, E., Rauterberg, M., Moll, T., Greutmann, T. & Strohm, O. (1991) Task orientation and user-oriented dialog design. International Journal of Human-Computer Interaction 3(2):117-144.

Volpert, W. (1985) Epilogue-system models and process models. In: M. Frese & J. Sabini (eds.) Goal directed behaviour: the concept of action in psychology (Lawrence Erlbaum, pp. 357-365).

Volpert, W. (1994) Wider die Maschinenmodelle des Handelns-Aufsätze zur Handlungsegulationstheorie (Pabst).

Adresses of the autors:

Matthias Rauterberg
IPO–Center for Research on User-System Interaction, Eindhoven University of Technology (EUT), Den Dolech 2, 5600 MB Eindhoven,
The Netherlands

Morten Fjeld
IHA–Institute for Hygiene and Applied Physiology, Swiss Federal Institute of Technology (ETH), Clausiusstrasse 25, 8092 Zürich, Switzerland