ESRC Centre for Research in
Development, Instruction and Training

DEPARTMENT OF PSYCHOLOGY, UNIVERSITY OF NOTTINGHAM
UNIVERSITY PARK, NOTTINGHAM, NG7 2RD, U.K.

Modelling Learning as it Happens in a Diagrammatic Reasoning Task

Frank E. Ritter and Peter A. Bibby

Technical Report No. 45
March 1997

Email: Frank.Ritter@psychology.nottingham.ac.uk

Phone +44 (0) 115 951-5292 Fax +44 (0) 115 951-5324

Modelling Learning as it Happens in a Diagrammatic Reasoning Task

Frank E. Ritter and Peter A. Bibby

ESRC Centre for Research in Development, Instruction & Training
Department of Psychology
University of Nottingham
Nottingham, England NG7 2RD

Frank.Ritter@nottingham.ac.uk

Technical Report No. 45

Abstract

We have developed a process model of problem solving with a simple control panel device. The model accounts well for many aggregate measures, including those from a study reported here (N=10): problem solving strategy, average fault-finding time, and the relative difficulty of faults. To further test the model, we compared the model's sequential predictions—the order and relative speed that it examined interface objects and answered with a subject solving five tasks, making it one of the first models to have its sequential predictions compared with human data as they both learn. We found that the predicted actions matched and mismatch in systematic ways. The correspondences showed that: (a) subjects were reflecting or checking their work; suggesting that learning mechanisms can, in some instances, model the speed of learning while not completely modelling the mechanism; (b) mouse movements in the interface provide support for the model's predictions of what the subject attended to; providing further evidence that protocols can be augmented with mouse traces; and (c) while the comparison of sequential predictions may be becoming more tractable, the effects of learning raises new difficulties, such as assigning credit to model structures that change with performance.

Acknowledgements

We would like to thank Ellen Bass, Gordon, Baxter, Fernand Gobet, Clayton Lewis and Mark Steedman for comments and discussions. Sam Marshall provided extensive comments and programmed the model; Shara Lochun assisted with the analysis.

Support for this work has been provided by a grant from the Joint Council Initiative in HCI and Cognitive Science, number SPG 9018736, and by the ESRC Centre for Research in Development, Instruction and Training.

Table of Contents

1. Understanding How Learning Occurs 1
    - 1.1 Previous models of learning tested with sequential data 1
    - 1.2 The next step in testing models that learn 2
    - 1.3 The diagrammatic reasoning task 3
    - 1.4 The Diag model 3
    - 1.5 New mechanisms and approaches in the model 5
2. Comparison with Aggregate Data 7
    - 2.1 The experimental task 7
    - 2.2 The aggregate data 7
    - 2.3 Qualitative comparison 8
    - 2.4 Quantitative comparison 8
    - 2.5 The need for a finer level of comparison 13
3. Comparison with Sequential Data 13
    - 3.1 The sequential data 13
    - 3.2 Our protocol analysis theory 14
    - 3.3 The alignment 14
    - 3.4 Operator support 16
    - 3.5 The time course of the comparison 18
    - 3.6 Review of alignment 18
4. Summary and Conclusions 21
    - 4.1 Reasoning with diagrams 22
    - 4.2 Learning through reflection 22
    - 4.3 Further development of trace-based protocol analysis 23
    - 4.4 Practical, usability problems of trace-based protocol analysis 27
    - 4.5 Implications for future models and modellers 28
5. References 30

1. Understanding How Learning Occurs

There is a speedup effect in the time taken to solve problems that is nearly universal across tasks and subjects (Rosenbloom & Newell, 1987). A major question has been how do people learn to perform such tasks more quickly? Cognitive models have used several mechanisms to account for learning, such as knowledge compilation and strengthening in the ACT family of models (Anderson, 1993), connection strengthening in PDP models (Rumelhart, McClelland, & group, 1986), and rule creation from impasses using analogical reasoning (VanLehn & Jones, 1993) and chunking mechanisms (Feigenbaum & Simon, 1984; Larkin, 1981; Rosenbloom & Newell, 1987).

Each of these learning mechanisms has successfully accounted for improvement and has proven useful in a explaining patterns of learning in a variety of different domains (e.g., Tower of Hanoi: Anzai & Simon, 1979; Ruiz & Newell, 1989; physics problem solving, Larkin, 1981; Lisp programming, Singley & Anderson, 1989; computer games, Bauer & John, 1995). However, the learning produced by these mechanisms has been compared with two particular types of human data. Most often, response times aggregated across trials, subjects, or both, have been used (e.g. Ram, Narayanan, & Cox, 1995). A few models have also been compared with behaviour before and after sometimes extensive amounts of practice performing the task. That is, the models using these mechanisms started by performing the task like a novice subject, and after extensive application of the learning mechanisms performed like expert subjects (Larkin, 1981).

1.1 Previous models of learning tested with sequential data

There is a problem that arises from the cross-sectional nature of the comparisons done so far. The learning mechanisms utilised by these models have not been compared with individual behaviour while learning was occurring. The actual path for an individual from novice to expert, to a certain extent, remains predicted but untested. The types of behaviour that occur during learning and that may lead to expert behaviour have rarely been examined by direct comparison with a process model while a subject learns.

There are a few pieces of work where learning has been examined in the most detail. Perhaps the model that was first compared to actual learning is that of Anzai and Simon (1979). The changes in strategy across four episodes of solving the Tower of Hanoi was modelled by their adaptive production system. While the model's behaviour was not compared to the subject's behaviour on an action-by-action level, Anzai and Simon did show that the model produced the same qualitative strategy shifts as the subject.

VanLehn (1989; 1991) examined learning in several domains including the Tower of Hanoi. He modelled learning by proposing what and when rules were learned but these rules were not created by an automatic mechanism. Nintendo-Soar (Bauer & John, 1995) modelled how novices could learn to become experts, but the model was only compared with expert data. Able (Larkin, 1981) was a model of the transition between novice and expert behaviour in physics problem solving in terms of the application order of principles. It used an impasse driven learning mechanism to automatically learn rules while solving problems. The acquisition of these rules represented the transition from novice to expert. However, only the initial and final behaviour were compared with novice and expert behaviour.

There are also several Act models that also come close to examining learning while it happened. While the ACT (Anderson, 1993) architecture implementations now include learning capabilities, typically models built within these frameworks that have been tested with sequential data (e.g. the Anderson tutors) model learning by adding hand-written rules based on the subject's performance. Blessing and Anderson (1995) provide an ACT model that learns, but its predictions were ested with aggregate data.

Cascade (VanLehn & Jones, 1993) has modelled how good and poor students learn while reading solved physics problems. In its most complete comparison, Cascade was used to model nine individual subject's behaviour while they studied 28 worked physics problems, learning as they examined the problems and sometimes explained the steps to themselves. Cascade generally predicted the goals that subjects would explain and how what was learned would transfer to other problems. This work shows that being able to model subjects while they learn is possible and useful. This work did not examine the time course of learning and ignored the order of actions, which are both examined here, but ignoring these aspects allowed them to examine far more data.

Perhaps the first model to have its behaviour compared with a subject while they both learned was Altmann's (1995) model of browsing by a programmer. While the subject did not particularly appear to learn, the model predicted that the subject was learning episodic and other knowledge while examining program code.

1.2 The next step in testing models that learn

A next step in studying the time course of learning in detail appears to be to compare the performance of a model including timing information as it learns with a subject's behaviour while the subject learns. In addition to testing the model in question, this may help us see where and how learning occurs.

We have developed a process model that learns, called Diag, that performs several problem-solving tasks with a simple control panel device based on a switch and light interface. The model can locate faults based on the status of the switches and whether or not lights are on. Diag and its architecture predicts that automatic compilation and improvement while performing a task will occur in a measurable way in this domain. Because it learns while solving problems, it accounts fairly well for most aggregate measures of behaviour in this area, even while we know that subjects are learning.

We examined the model's sequential predictions with respect to actual subject behaviour. This comparison is a type of protocol analysis (Ericsson & Simon, 1993; Ritter & Larkin, 1994), and serves two purposes here:

(a) Diag's behaviour, the actions it takes and the information it uses from the environment, are core features of what makes it a process model. The order and speed that the model examines interface objects and interacts with them are predictions of subjects' sequential behaviour and attention while solving the same tasks. The model's sequential behaviour is essentially a series of predictions about what steps the subject will do and the information they will use to perform the steps. Given the difficulty inherent in creating the model, in addition to the model's general behaviour, these sequential predictions should also be used to understand the subject's behaviour. And

(b) The results of the comparison process will help indicate where to improve the model. The model, based on previous models in this area, appears to match the data fairly well. However, Diag, in order to perform the task, must make additional predictions about behaviour that can be tested as well. The model may learn faster or slower within a trial than the subject, it may perform in too efficient a manner, leaving out steps the subject does, and so on.

Comparison of a model's sequential behaviour is an old technique (e.g. Feldman, 1962), and the tools that we used suggest that it is becoming more tractable. For example, there are now tools for automatically aligning the model's predictions with the subject's behaviour (Ritter & Larkin, 1994; Sanderson, Scott, Johnston, Mainzer, Watanabe, & James, 1994).

This approach may be becoming easier for theoretical reasons as well. Mouse movements can be used as another stream of protocol data to support the model's predictions of attention and mental structures (Kennedy & Baccino, 1995; Peck & John, 1992; Ritter & Larkin, 1994). They are less ambiguous and this assists in the alignment process

In the remainder of this paper, we will first explain the task that both subjects and the model completed. The model will be explained, including how it accounts for the aggregate measures for 10 subjects doing exactly the same task. Next, the details of how the sequential behaviour comparison was performed are presented. Finally, we will discuss the results and what we discovered from comparing the sequential predictions of a model that learns. Most importantly, we will see that the sequential comparison allows us to point out that reflection can be an important aspect of learning in this task, even when the task is short and the reflection not completely verbalised.

1.3 The diagrammatic reasoning task

The task that Diag and subjects solve consists of trouble-shooting a control panel device using a memorised schematic. The device is similar to Kieras and Bovair's (1984) laser-bank device. Figure 1 displays the circuit schematic, and Figure 2 displays the interface display that the subjects saw, showing the switches in the schematic along with a light indicating that each component is working. On the interface display, and later in the text and figures of this paper, straightforward abbreviations are used to refer to objects in the schematic. For example, PS stands for power supply, EB1 is energy booster 1, and so on.

Before troubleshooting the device, subjects were given: (a) a general introduction to the problem; (b) basic information on the interface; and (c) a schematic of the underlying circuit. Subjects were then told that one component in the circuit was faulty, and were asked to indicate which one by clicking on it with the mouse. Their reaction times and choices were recorded and analysed for latency, correct number of choices, and so on. Previous work in this area has resulted in summaries of behaviour in a series of related tasks such as operation of non-faulted circuits (Bibby & Payne, 1993), and process models that perform multiple tasks but do not learn (Bibby & Payne, 1996)

1.4 The Diag model

The explanation of the Diag model can be broken down into two parts. First, we note the model's global structure and the routine mechanisms used. Second, we explain those mechanisms that are new contributions or that are confirmations that mechanisms work in a new domain.

Diag is built using the Soar cognitive architecture (Laird, Newell, & Rosenbloom, 1987; Newell, 1990). The model we report here is a further development of an existing model (Bibby & Reichgelt, 1993). The earlier version of the model was good at explaining subjects' performance on the first few trials but learned far too quickly. While subjects continued to improve, the model reached asymptotic behaviour rapidly and did not continue to learn. The model was changed by adding ATTEND and COMPREHEND operators to the top space, consistent with Newell's (1990) recommendations for modelling interaction.

The present version of Diag consists of 186 initial production rules. Figure 3 shows how these are organised into 7 problem spaces and 21 operators in a hierarchical organisation. Lower level operations are used to understand what is observed from the interface and to suggest what external action to perform in the top problem space.

Knowledge of the schematic is represented within the model as linked lists organised as 'routes' through the circuit as if the schematic was memorised. Diag recalls and follows these linked lists to find the next component to examine. Visual interface information is represented as declarative structures for lights and switches. This corresponds to the diagram condition reported in Bibby and Payne (1993) and implements a common strategy (#1 in Bibby & Payne, 1996).

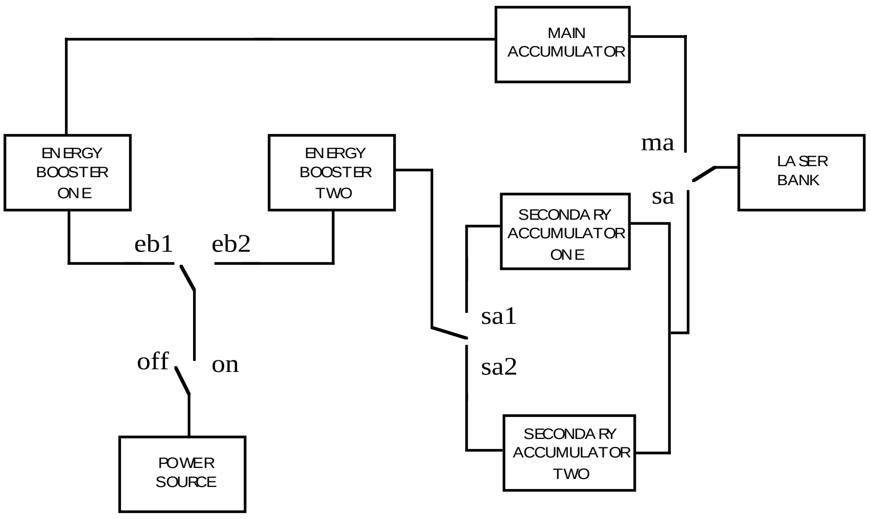


Figure 1. Schematic of the laser-bank device.

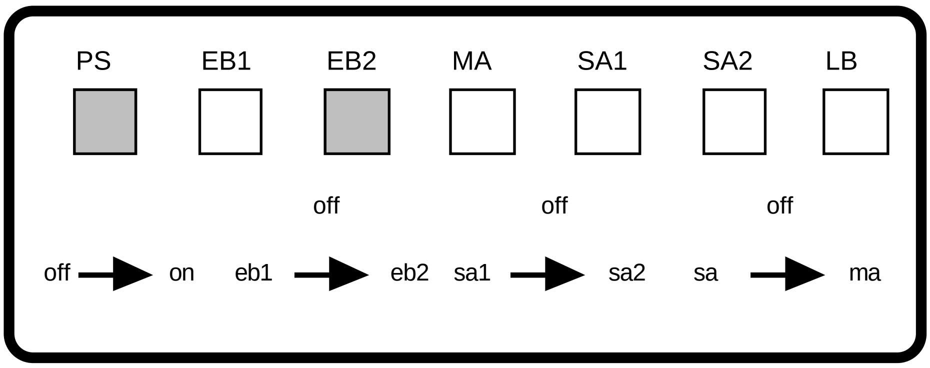


Figure 2. The laser-bank interface that the subject uses to find the faulty component. Boxes (the lights) below the abbreviations and above the switches (arrows) are greyed out to indicate components are working.

When a problem is presented to Diag, the complete status of the interface and what can be seen is represented in the top state as a 'fake' real world. This information is accessed by the combination of an ATTEND and a COMPREHEND operator that select the relevant object to examine and retrieves information from this structure. The organisation of components on the interface diagram and the use of the ATTEND and COMPREHEND operators moving sequentially across the visual representation causes the components to be checked in sequence.

Diag typically learns additional rules while performing a series of troubleshooting tasks. What is learned depends on the series of episodes. The model learns how to perform actions more directly without internal search, and which objects to attend to on the display without internal deliberation. Both of these types of learning occur while performing the task. If fully learned problems are not repeated, the model can learn a total of 217 new rules (chunks) over 20 problems.

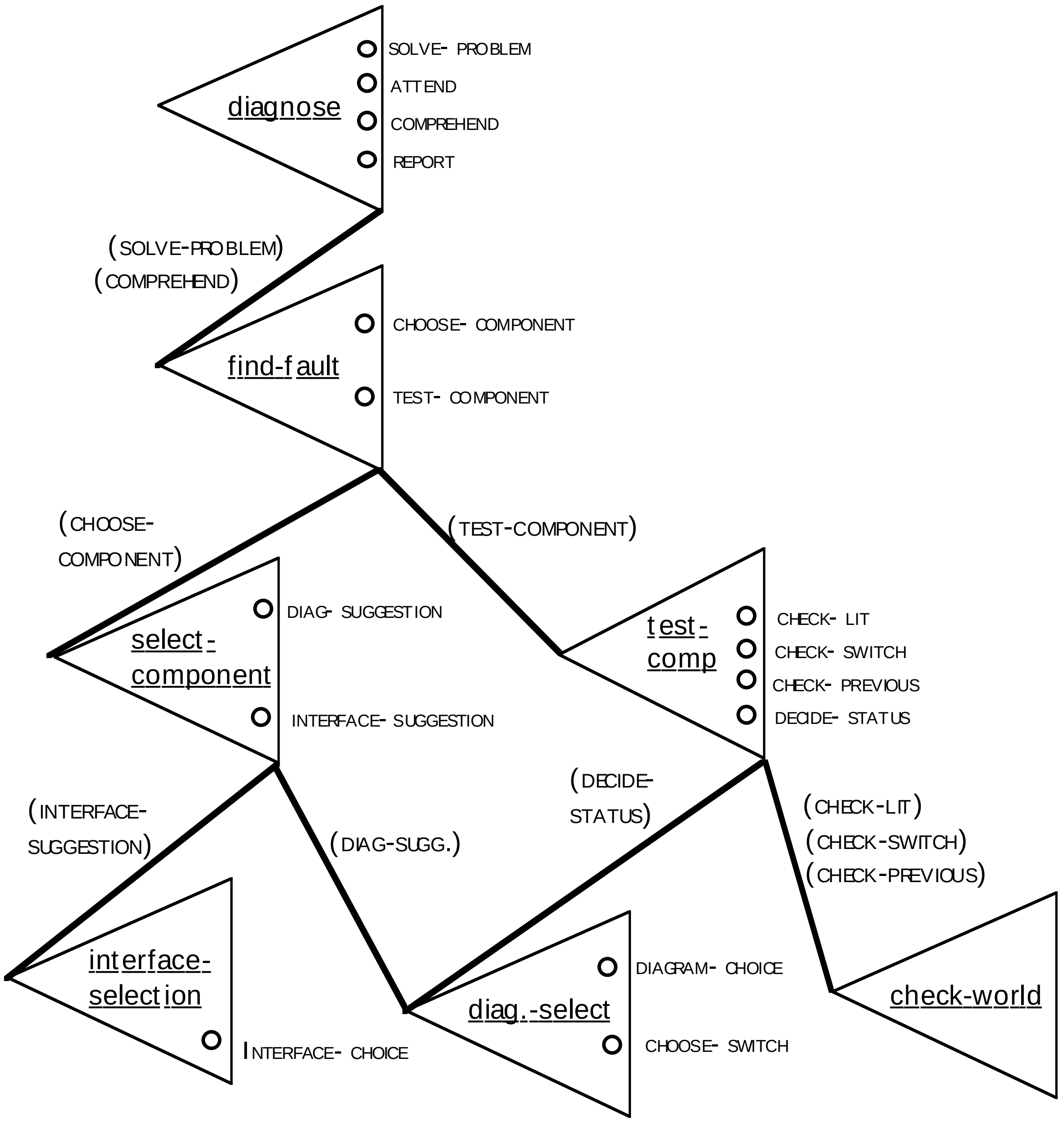


Figure 3. Structural organisation of the Diag model showing its hierarchical structure. Triangles represent problem spaces (with underlined names); circles represent operators (in small caps). Links to sub-problem spaces to implement operators are represented by a link labelled with the operator names in parenthesis.

1.5 New mechanisms and approaches in the model

Diag incorporates new modelling mechanisms and approaches. Two of these are generally interesting and are likely to be applicable to other models of learning: Diag interweaves problem solving and learning and Diag interacts with an external world using an internal representation of the world. One other mechanism may be architecture specific or may be applicable to other architectures: Diag interacts with the outside world solely through the top state in its hierarchy.

General mechanism: Learning while performing the task. As a result of the Soar architecture, but a characteristic of nearly all problem solving behaviour, Diag learns while performing the task.

Diag learns how to resolve impasses in performance when they occur. The knowledge that is learnt can be applied in later episodes. What is learned, when it is learned and how it is learned arises from an interaction between the architecture, the knowledge, the organisation of the knowledge, and the experience the model has of problem solving. In Diag, these interact to create three types of learning. (a) Operator implementation, where specific knowledge about how to apply an operator is learned through search in another problem space. This is done for several operators (e.g. CHOOSE-COMPONENT). (b) Operator creation, augmentation, and proposal, where an operator is created in one problem space for use in another. This is done solely for ATTEND.
(c) State augmentation rules, which augment the state with derivable knowledge. For example, the rule can be learned that if EB1 has been checked and the current light you are looking at (the main accumulator) is not lit, the component MA is broken.

During the course of solving the fault-finding tasks newly learned rules transferred across episodes. However, there was no transfer of newly learned rules within an episode because the model does not backtrack in its problem solving. These learned rules typically implement operators such as DIAG-SUGGESTION (suggesting which component to check based on the diagram) or CHECK-SWITCH-DIAGRAM (deciding how switches should be positioned if a device is not working). The most powerful rules, those that speeded up performance the most, proposed the ATTEND operator earlier on the basis of previous problem solving within an episode. We have called these rules episodic chunks since they contain information about what problem-solving has already taken place. These learned rules provide the model with a long-term memory for its own reasoning.

General mechanism: ATTEND and COMPREHEND operators. Many previous cognitive models have had the whole world directly accessible for problem solving. For example, models that solve the Tower of Hanoi might have the contents of the three posts completely included in working memory. In simple tasks, such as the Tower of Hanoi, this probably results in little difference in behaviour because perception in such cases is relatively infrequent, effortless, and error free.

In more complicated tasks, particularly tasks that use visual processing, having access to the entire external state may hide essential features of the task and the interaction between the problem solver and the environment. Rather than view the entire world at once, we used explicit operators to represent acquiring the perceptual components of the task. This created a different structure for problem solving. The model was continually deciding where to look, looking there, checking the part it saw, and then deciding where to look next (if necessary). In later work, we have started to represent the world outside of the model even more explicitly, and using even more specific behaviour to examine and interact with the world (Bass, Baxter, & Ritter, 1995; Jones & Ritter, In press).

Perhaps specific mechanism: Serialisation through the top level. Diag uses a distinct approach for interacting with the outside world by using the top-level state. This mechanism is so far only used by the Soar architecture. This approach may be a general answer for how to serialise learned behaviour. If this is true in some global sense, other architectures should use it; after some time, if this approach is used only in Soar, it suggests that there is something unusual about the Soar architecture, and that the Soar architecture is not general enough, or that the other architecture designers have not attempted to solve this problem.

In an architecture that supports learning, when a hierarchical or sub-goaling control approach is used and learning and output can occur at any level, the model can end up learning a series of external actions to apply to a given initial state. On seeing a similar state, this knowledge can later apply, proposing to do the several operations in parallel (everything at once) that had been initially performed serially (in order). One way to solve this problem is to serialise all input and output through the top level of the hierarchy. The Soar architecture requires input from the outside world to appear on the top state, and for output to the outside world to be initiated from changes to the top state. This is an explicit architectural commitment that appears to be necessary in order to learn how to interact with the outside world (Laird & Rosenbloom, 1995, p. 35).

To avoid doing everything at once, an approach taken in Soar is to force all output through the top state. If a subgoal requires an external action to be performed, this must be added to the top state, typically by removing the goal stack and selecting an operator to add the request to the top state. The goal stack may then be re-built (the rebuilding of the problem solving context might not occur if the input from the outside world gives rise to a different situation requiring a different response). Diag uses a hierarchy of problem spaces and goals with this method of interacting. It is similar to the method of goal reconstruction proposed in Teton (VanLehn & Ball, 1991).

This approach to interaction makes a testable prediction, that there will be a loss of context when initiating an interaction with the environment. We have not directly tested this, but we have since found evidence that this effect may exist (Irwin & Carlson-Radvansky, 1996). This may not be a real result, it may be merely an artefact of the way this model is built in this architecture. Yet, previous attempts to learn to interact have had problems where they learned too much too quickly, and ended up attempting to do several things at once. It may turn out that not this but a similar mechanism is necessary for serialising interaction with the outside world.

2. Comparison with Aggregate Data

Ideally, models or theories will openly state the lists of regularities that they explain up front. This can help establish a minimum and continually raising set of requirements for models to achieve in a given domain, and eventually, across domains (Newell, 1990). This has also been called 'criterion-based psychological modelling' (Ritter, 1992).

Our initial belief in this model came from the fact that it could account for several regularities in aggregate measures in similar tasks using other taught representations besides a schematic. Table 1 notes the regularities that Diag accounts for that have been reported in previous work. These regularities are robust, and have been found in other studies (Bibby & Payne, 1993, 1996; Bibby & Reichgelt, 1993), as well as being consistent with the aggregate and protocol studies reported here. These regularities can be separated into qualitative effects and quantitative effects. We examine a selection of them below.

2.1 The experimental task

As noted in Table 1, the first test of a process model is that it can do the task of interest. The current version of Diag can do two tasks; fault finding: it can find the fault in the device given a configuration of lights and switches, and the switch task: it can find the switch that needs to be flipped to make the device work given a display and the assumption of a fault-free system. Normally, noting that a process model can do the task is redundant, but in this case, we wish to emphasise that the model can do more than the fault-finding task, although only the fault-finding task is examined here.

2.2 The aggregate data

In order to directly compare this version of Diag against performance on the task it models, data was collected on ten subjects solving fault-finding tasks (and only solving fault-finding tasks, unlike previous studies). After subjects memorised the schematic, their response times were measured and recorded across 20 problems (trials). Subjects saw different, randomised series of trials, so we also ran Diag on the same series.

We compared the model to the subjects' performance in three ways. The first analyses examines the qualitative nature of the models behaviour and compares it to the strategy that subjects reported using. The second set of analyses provide aggregated indices of performance over all 10 subjects. The third set of analyses look at individual subject's performance in comparison to the model’s predictions. The times taken when errors were made by subjects solving the fault-finding task were removed from the analyses since the model did not make any errors.

Table 1. Selected regularities in solving the troubleshooting task.

Perform the task. As a minimum, the model must predict that subjects can perform the basic task. This is also an important aspect to being a process model, that is, specifying the initial process a subject uses to perform the task.

Problem-solving strategy. The approximate problem solving strategy that most subjects used is exemplified in a subject's retrospective protocol that we quote here:

> "...check each component in turn to see if its indicator light is lit. If it is, move on to the next component. If it is not lit, check to see whether it should be lit. If it is not receiving power, move on to the next component. If it is receiving power, then it is broken" (Bibby & Payne, 1996).

Faults with components remote from the power supply take longer to solve. The serial strategy that most subjects adopt leads to a difference in reaction time for diagnosing the possible faults both quantitatively and qualitatively. Figure 4 (below) shows the average time to solve the seven fault-finding tasks for 10 subjects averaged over 20 trials.

Subjects become faster with practice. After practice with the same fault or with different faults subjects get faster. However, the rate of improvement is not smooth for individual subjects, apparently as different amounts of learning transfer between different fault-finding trials. Surprisingly, the rate of improvement is not well fit by a power law for individual problem-solving times. For example, the second time to solve a particular fault is slower than is expected and often slower than the first trial.

2.3 Qualitative comparison

Three qualitative effects stand out of the data that correspond to the previous findings but are based on this task and this circuit representation. These can be compared with the model's performance on a general level.

Problem solving strategy. Diag performs the fault-finding tasks using the general strategy that subjects use. Sequential checking of components from left to right is emergent from the interface representation and the model's knowledge about how to check components and their connections.

Remote faults take longer. Faults with components remote from the power supply are further down the routes and take subjects longer to solve. The model also takes more time for more remote faults. Figure 4 shows that as faults occur on components further to the right of the display and further along the schematic paths, they generally take longer for subjects and the model to find them.

Practice leads to faster responses. With practice subjects can find faults faster. This improvement shows substantial transfer across tasks. Knowledge in tracing through the circuit improves performance finding faults elsewhere in the circuit.

2.4 Quantitative comparison

An analysis of the aggregate data was conducted. First, the time taken to solve the different fault-finding tasks were compared to the number of model cycles (decision cycles) it took to solve that task. Second, the time per trial was compared to the number of model cycles per trial. Third, the average time per subject was compared to the average number of model cycles for each run of the model solving the same series of problems.

Figure 4 shows the comparison between the time for each fault and the model cycles for each fault aggregated over all the subjects and trials. For both the time taken to solve the problem and the model cycles, there is a general increase in time taken to solve the problems from left to write across the device interface, though this is not as apparent as in previous research (Bibby & Reichgelt, 1993). The previous research looked at the first time a task was attempted, ignoring learning. Learning has a strong effect. For example, if PS is the first fault that a subject attempts it will take a long time. At the same time, due to inter-task transfer, if LB is first encountered later in the sequence it will take substantially less time to solve.

Overall, the pattern of similarity between the model's cycles and the subjects' problem solving time is striking. The correlation between the fault-finding problem solving time and the model's decision cycles is r = 0.99 (p < 0.05).

Figure 5 shows the problem solving time and the model cycles aggregated over subjects for the 20 trials. Again there is a strong relationship between the time the model and the subject took to solve the problems (r = 0.99, p < 0.05). There are three points of interest that can be seen from this comparison. First, these data do not show a perfect power law of practice. Nerb, Krems and Ritter (1993) have argued that the power law is often artifactual and the result of averaging over aggregate data. Individual subjects rarely fit a power function as well as might be expected. What is clear from this set of data is that the power law does not necessarily emerge even when aggregating over subjects. As this model indicates, decrements in time with practice depend on the task on each trial and a smooth curve will only occur with averaged data. Second, a post-experiment analysis of the problem series indicated that there was a problem with the random assignment of faults, particularly that the first two problems were not randomly distributed, appearing on average early in the circuit. To obtain a smooth curve, the average difficulty of problems must remain the same across the series. Finally, based on the relative scales in Figures 4 and 5 and the theoretical predictions of the model cycle taking 50 to 500 ms (Newell, 1990), it appears that the model underestimates the solution time. This is a common problem in models of this kind. Models are often too intelligent, either not performing all the task actions or performing them more effectively.

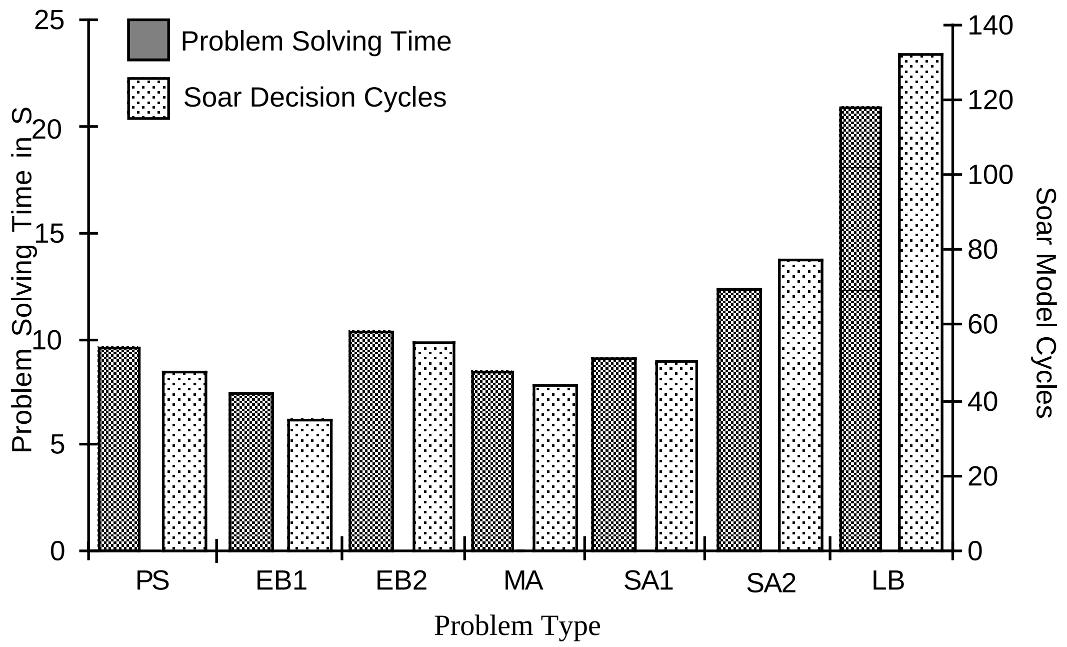


Figure 4. Comparison of problem solving time for subject and model for all correct trials aggregated by fault type.

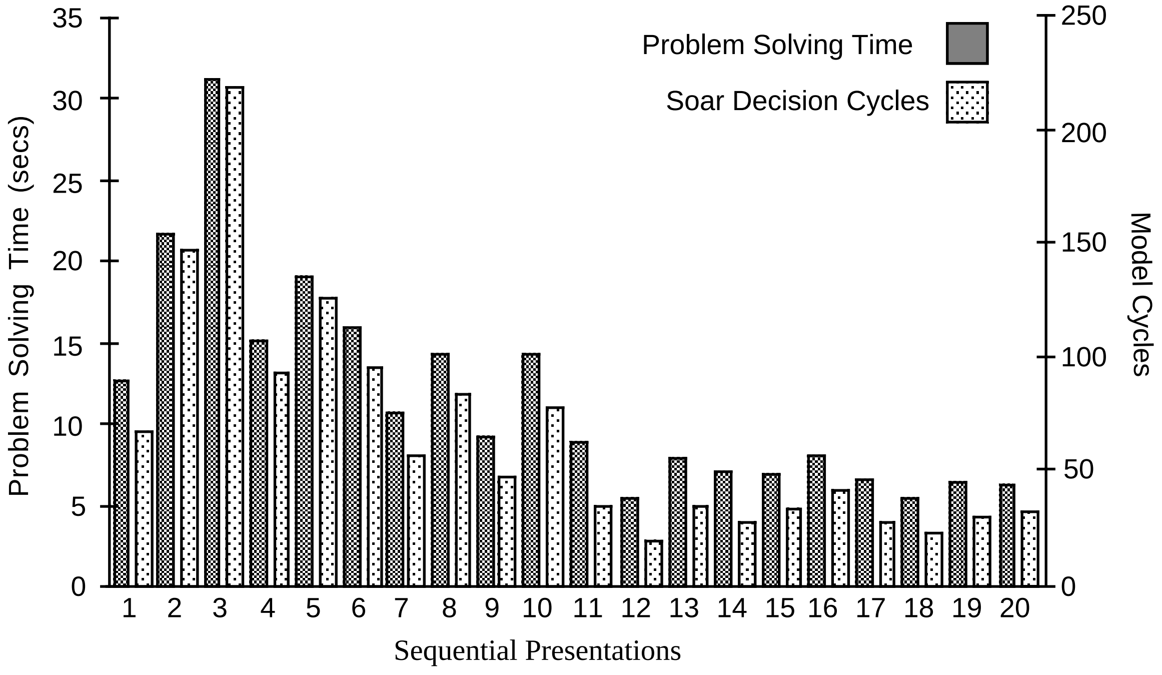


Figure 5. Comparison of average problem solving time for subjects and model aggregated by trial for all correct trials. The model saw 10 different fault orders corresponding to the series seen by the 10 subjects.

Figure 6 shows the total problem solving time per subject and the total model cycles for Diag solving the same series of problems. The correlation between subjects' overall performance and the model's cycles is r = 0.32 (p > 0.1). This result is surprisingly different from aggregating over task or trial. Simply using the aggregate time per subject and the average number of model cycles per run of the model we obtain an apparently contradictory picture. To make sense of this pattern of results each set of model cycles per run of the model were regressed onto the problem solving times for each subject individually.

Table 2 shows that the average proportion of variability accounted for overall is 82%. However, the regression was not significant for two of the subjects (S1 and S5). When these subjects are removed from the analysis the average proportion of variability increased to 92%. A second check on this result is to examine the B coefficient that represents the number of seconds per model cycle. Newell (1990) suggests that the it will be between 30 and 300 ms. According to this analysis S1 has a rate of 14 ms and S5 has a rate of 7 ms. Both these values are implausibly fast. All the other subjects' rates lie with the range that Newell specified, and that have been found in previous work. When S1 and S5 are removed the average rate is 154 ms per cycle. Where Soar problem solving models have been compared with subjects, the model cycle rate is generally long (341 ms, Nerb, Krems, & Ritter, 1993; 145 ms, Peck & John, 1992 analysed in Ritter, 1992), indicating that models are too intelligent, performing the task more efficiently than subjects (taking less cycles) or not performing as much of the task as the subjects (e.g. not modelling moving the mouse). Models of simple tasks often come quite close to matching data with a 100 ms cycle without adjustments such as reading rate and disambiguating regions of text (Lewis, 1993, p. 201), simple reaction times (Newell, 1990), and covert visual attention (Wiesmeyer, 1991).

For those subjects who were well matched by the model’s predictions, the model's times indicate that not all differences in the subjects' total times were due to individual differences. The model's times indicate that some of the differences were due to the different series of faults that each subject saw. Figure 7 shows one subject's problem solving times plotted against Diag's model cycles. Similar patterns occur for those other seven subjects where the model cycles were a good predictor.

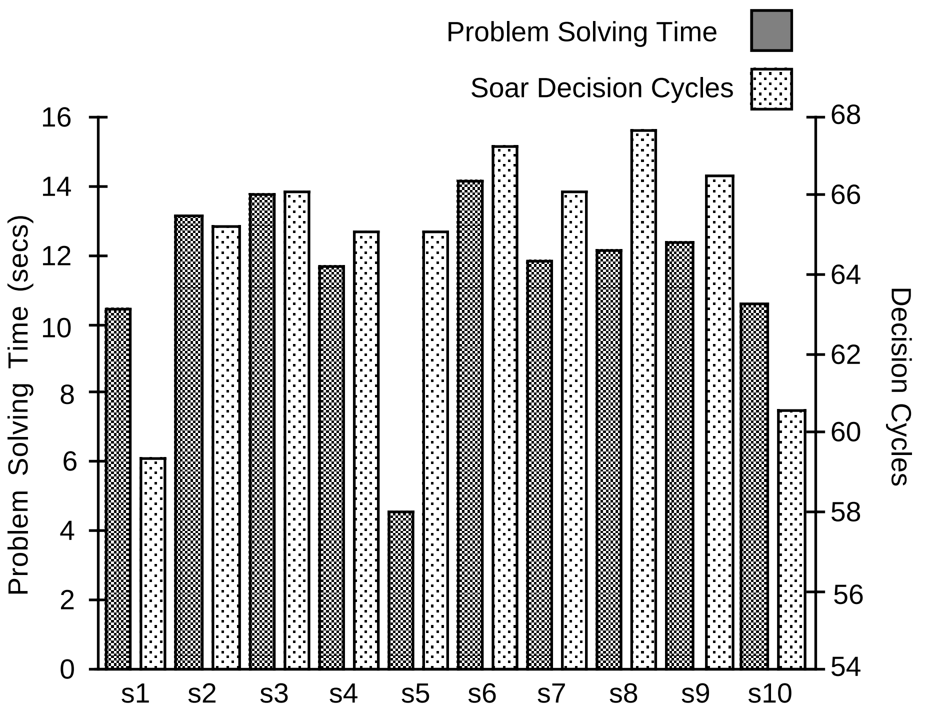


Figure 6. Comparison of problem solving time for subject and model aggregated by subject for all correct trials. In the comparison, the model saw the same fault order as seen by the subject.

Table 2. Regression coefficients for regressing model time on subject time for correctly solved problems in each series. B is in units of second per model cycle. N is the number of correct answers for each subject out of the 20 possible. * indicates that the r value is not significant at the .05 level, all others are significant at the .05 level.

Subject	R^2	B	N
S1	0.028 *	0.014	19
S2	0.960	0.170	16
S3	0.939	0.176	19
S4	0.880	0.147	20
S5	0.066 *	0.007	20
S6	0.955	0.175	18
S7	0.923	0.150	19
S8	0.839	0.117	19
S9	0.984	0.149	17
S10	0.871	0.151	18
Mean	0.824	0.138	18.4

We examined how the fit varied across time for the eight subjects for whom Diag's model cycles provides a good fit to their problem solving times. When the fit was compared across repeated exposure to a particular type of fault another regularity appeared. For the first example of a task the model was generally very good at predicting the time taken to solve the problem. For the second example of the same fault the model often underestimated the time it took subjects to solve the problem. For the third example of the task the model often overestimated the time it took subjects to solve the problem. Out of 47 occasions across subjects and tasks where it was possible to find the same fault that had been successfully identified three times, this pattern occurred 32 times. Figure 8 gives four examples of this pattern.

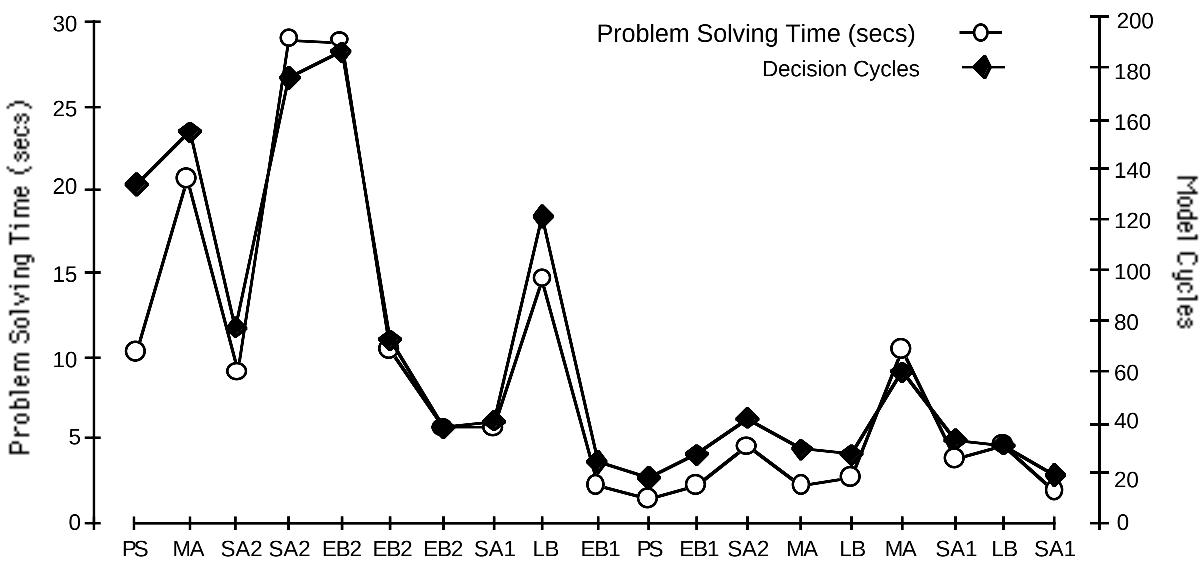


Figure 7. Comparison of the reaction times of subject 9 with Diag's solution times for the same series of faults.

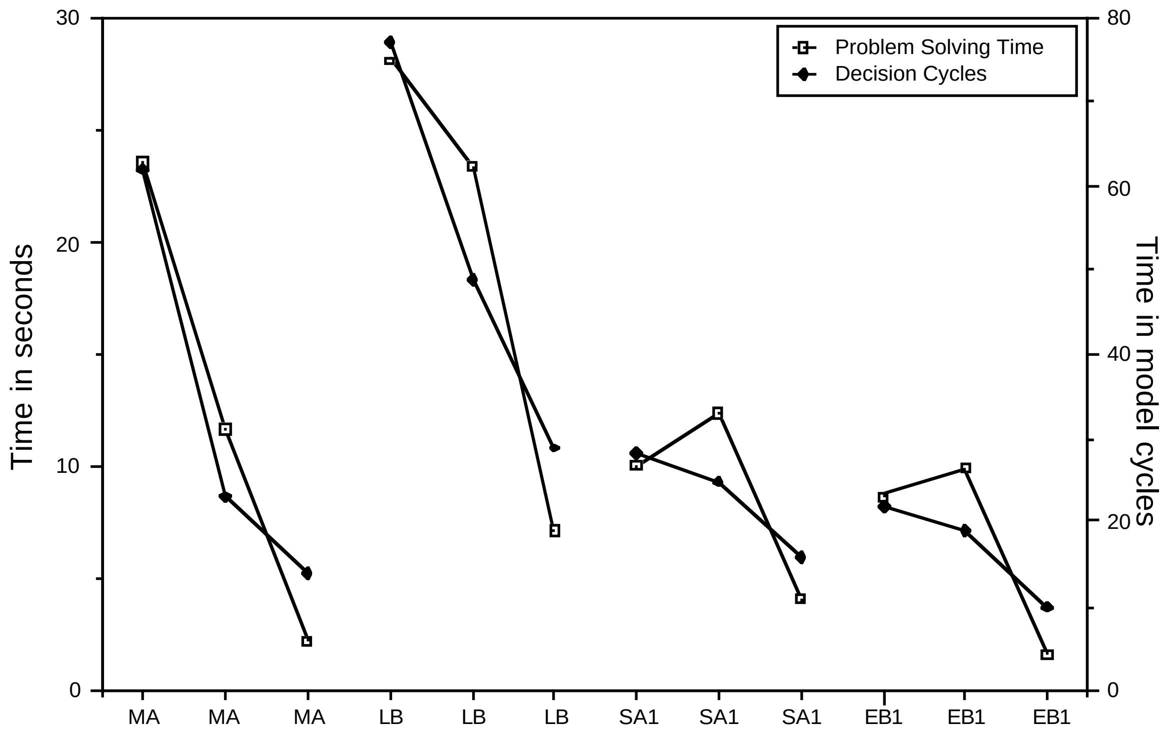


Figure 8. Comparison of a single subject's reaction times with Diag's solution times arranged by exposure to type of fault.

Interestingly, it did not appear to matter whether the fault had been seen early or late in the learning trials or whether the tasks were close together or far apart in occurrence. In Figure 8 the initial MA fault was the third trial for this subject. The second MA fault was trial 9 and the final MA fault was trial 11. The first SA1 fault was encountered on trial 8, the second SA1 fault was on trial 13 and the third SA1 fault was on trial 20.

In order to estimate the size of the differences between the model and the data on the second and third same fault trials, the times taken on those trials was transformed into model cycles by dividing by the estimated cycles/s. A difference was calculated between the model's actual cycles and the predicted cycles. The difference in cycles between the model and the subject on the second trial was approximately constant (mean = 7.75 s; sd = 1.136). At the same time the difference in cycles between the model and the subject on the third example of the fault was also constant, although this effect was not quite as reliable (mean = -2.91 s; sd = 1.33).

2.5 The need for a finer level of comparison

Model development can be seen as a two step process, of first establishing that a model is worth taking seriously because, for example, it can account for a large amount of the variance in the data, and then second, finding ways to improve the model (Grant, 1962). Initially, in order to understand how well this version of Diag fits the data, we compared its performance with aggregate measures. It generally fits this data rather well, for it performs the task of interest and accounts for a large amount of the variance in reaction times a series of faults. Because the model accounts fairly well for the data these comparisons do not provide strong suggestions for where the model can be improved.

One way to improve Diag is simply to cover more types of data and a broader task. There remain many regularities that this model does not explain. For example, a worthy and important regularity (although one not addressed by this model) is to explain how the initial performance was learned through observation, instruction or other methods (e.g., Huffman & Laird, 1995; Kitajima & Polson, 1996).

Another way to improve Diag is to explore in more detail the area of its behaviour that it already models well, that of fault finding. One way of doing this is to compare Diag's sequential actions with the sequential behaviour of a subject. Diag's actions, because they are generated within a cognitive architecture, can be treated as predictions that subjects will take the same steps, in the same order, and at comparable times.

Using the sequential predictions of Diag has other advantages as well. It will allow us to explore why the second trial for a particular fault takes longer, and it will allow us to look inside an otherwise atomic reaction time, perhaps seeing how and where learning occurs. This type of sequential comparison is not often done, so while it is becoming easier, it is not yet routine. This comparison may also provide additional methodological suggestions.

3. Comparison with Sequential Data

In order to further understand Diag's behaviour and find out where to improve it, we compared its sequential predictions (its actions) with a subject's verbal and non-verbal protocol while solving five problems. This comparison highlighted several places to improve the model, allowed us to see a new way that the subject learned, and also allowed us to further understand and extend the methodology of trace-based protocol analysis (Ritter & Larkin, 1994).

3.1 The sequential data

We recorded a subject on video tape while he solved a series of five fault-finding problems. He was a university student without extensive electronics experience recruited for a small payment. As in previous studies (Bibby & Payne, 1993), the subject had been instructed in the task. In this case he had studied only the schematic representation of the circuit to the point where he had memorised it. After a warm-up talk-aloud task he performed the task five times and his talk-aloud protocol was recorded and later transcribed. His mouse movements were coded into regions around the major components in the interface and his mouse click on the faulty part indicated the end of a episode. He performed the task correctly four times, generating 50 verbal utterances and 41 codable mouse movements. His performance times are comparable with existing subjects. While this data may not seem extensive, it represents about 1% of the total verbal protocol data compared with sequential model predictions through 1994 (Ritter & Larkin, 1994).

3.2 Our protocol analysis theory

The basic protocol theory that we used to guide the alignment of the model's predictions and the subject's actions was taken from Ericsson and Simon (1993), with the extensions suggested by Peck and John (1992), and Ritter and Larkin (1994). The basic tenets of this approach are listed in Table 3.

3.3 The alignment

The data from the five episodes were semi-automatically aligned with the trace of the model solving the same series of faults. Table 4 shows the results of these alignments. The second section of the table indicates and for each episode how well verbal utterances were matched. These verbal columns indicates how many verbal utterances were aligned with the model's trace. About one-third of the utterances were not codable (like "uhhm" and "So...") and were excluded from later analysis.

The worst correspondence occurred during Episode 4 where the subject got the wrong answer. This should be somewhat expected, the model solved the problem correctly, and much of what the subject said and did would thus not match, although the initial behaviour corresponded reasonably well. This episode was dropped from further analyses.

The last column in the second section notes the verbalisation rate. It has been conjectured that experts may verbalise less (although Ericsson & Simon, 1993, p. 250-251, report studies where expertise increases verbalisation; vanSomeren, Barnard, & Sandberg, 1994, p. 34 report that expertise decreases verbalisation). While the subject is certainly not an expert, this drastic improvement, along with the verbalisation rate decrease in Episode 5, suggests that verbalisation may indeed be an inverted U-shaped curve, with novices also less able to verbalise.

The third section of the table indicates how well the mouse movements could be aligned. There were many mouse movements that were not codable (e.g. random jinks and preliminary movements), and there were also a relatively large amount of mouse movements over items in the interface that could not be aligned with objects that the model was considering. These may indicate

Table 3. Basic tenets used in creating the alignment between Diag's actions and the subject's utterances and mouse movements.

- Task actions of the subject, that is, reporting the fault by clicking on it, were aligned with the model's task actions, generation of a request to report the fault.
- Mouse cursor positions that did not stay in one place for more than 100 ms were noted as "not alignable".
- Mouse positions held longer than 100 ms were aligned with the model's operator that adds or uses such state information.
- Utterances were aligned to operators that add state information, act on state information or both;
- Utterances and the model operations giving rise to such structures were aligned in order of entry to the model's working memory where possible

areas where the model could be improved (e.g. it should more directly examine switches), or they could indicate actions in the motor movements that are not yet modelled, as several of these movements represent initial movements towards a target.

The last section of the table deals with time and the relative speeds of the model and subject. The first column in this section is the time it took the subject to solve the task. This time varied between tasks because faults earlier in the circuit (starting from the power supply) took less time, and faults later in the series took less time due to practice. Later columns show how many model cycles (decision cycles) the model took to solve the problem, and the relative ratio between model and subject time.

Having a good measure of the processing rate is important. It will allow us in the future to make *a priori* predictions of reaction times and the time course of problem solving behaviour based on the model's performance on the task, without empirical measurement. Fixing the cycle rate in architectures will be a complicated problem because there are so many uncertainties, including the subjects, the tasks, the strategies to solve the tasks, and the architecture itself. It will take numerous tasks and repeated attempts to fix a proper value for each parameter, just as the measurement of atomic weight had to evolve.

Theoretically, the predicted order of magnitude of this rate should be between 3 and 30 cycles per second (Newell, 1990). The preliminary attempt to measure that rate here fits that range. The rate probably should not wander in this range, but be a constant. The results here suggest that this rate will be closer to 3 cycles per second rather than 30.

There are several reasons to believe that this rate is not yet fully settled. The model does not interact with the environment, which would increase what the model did per task and thus decrease the time per cycle. The model does not verbalise. Subjects, while they talk aloud, typically take longer to solve a problem. Either comparing the model to a subject that did not verbalise or having the model verbalise would also decrease the time per cycle. And, as we shall see shortly, the model does not include a significant amount of reflection near the end of the problem solving. People appear to do this and including it would increase what the model did and thus also decrease the time per cycle.

The subject's performance improves rather quickly in this simple task over the five trials. It is hard to compute because his tasks vary, but based on the model's more repeatable performance, if Episode 6 were a power supply fault, it would be solved in 53% of the first episode’s time; if Episode 6 was a main accumulator fault, it would be solved in 7% of the initial time (there is much more processing to be lost, and there are no switches requiring interaction).

The last set of columns labelled learning notes that the model learns 101 new rules in the course of problem solving. The number of rules learnt and applied per episode varies between episodes indicating different amount of transfer, both in the rules that can transfer and thus the rules to be learnt. The rate of learning is fairly constant across episodes when adjusted by model cycles at one new rule learned every 6.4 cycles (range: 5.8 to 7.4) or 0.81 s (range: .43 to 1.7). This is approximately as predicted (Newell, 1990, p. 318). The rate of transfer is a rule every 1.59 cycles (range: 0.5 to 3.3) or 0.59 s (range: 0.065 to 1.46). Note that the transfer rate varies, indicating that there are differences between episodes. This variance suggests that the noise that is typically seen with practice with respect to the power law of practice is not always noise, but sometimes represents genuine differences from the power law due to differential transfer.

Ritter and Larkin (1994) reported that verbal utterances appeared to be uttered approximately 1 s after the model predicted that the mental structures entered working memory. Without this adjustment the matched behaviours correlated on average r = 0.788 across episodes, with the subject's times adjusted this way they correlated r = 0.780. This study does not support this lag, but it lacks the extensive required task actions to fully measure it.

Table 4. How the model accounted for segments in each of the five episodes. NA indicates utterances (like "Uhmm") or mouse movements (like the first of two moves towards a target) that were not appropriate to code against the model. The last set of columns indicates the number of cycles the model took to find the fault; the time the subject took to find the fault; and the corresponding ratio, which includes in its calculation an additional 2.5 s to model the output of the mouse click, but does not include any slowdown of the subject due to verbalisation.

	Utterances			**Mouse**		**Processing times**			**Learning**	
Episode & fault	**Matched/ possible**	**NA**	**Words per min.**	**Matched / possible**	**NA**	**Model cycles**	**Subject Time (s)**	**Model Cycle per S**	**New rules**	**Trans-fer**
1: Power supply	3/3 (100%)	(3)	56.9	2/5 (40%)	(5)	58	21	3.14	9	0
2: Main accum-ulator	10/12 (83%)	(3)	98.7	6/17 (35%)	(25)	186	46	4.28	25	3
3: Laser bank	12/14 (86%)	(1)	131.0	2/3 (67%)	(8)	301	27	12.29	47	19
4: Main accum-ulator *	6/17 (35%)	(13)	41.0	2/11 (18%)	(18)	60	13	5.71	10	19
5: Energy booster 1	3/4 (75%)	(1)	87.7	5/5 (100%)	(7)	58	17	4.00	10	2
Totals or mean	34/50 (68%)	(21)	83.1	17/41 (41%)	(63)	663	124	5.95† 5.88‡	101	43
Excluding Episode 4	28/33 (85%)	(8)	93.6	15/30 (50%)	(45)	603	111	5.97† 5.93‡	91	24

* Subject's answer was incorrect.
†Weighted by time.
‡Weighted by episode.

3.4 Operator support

Figures 9 and 10 are operator support graphs (John & Vera, 1992) created from the comparisons summarised in Table 4. Figure 9 shows the operations and their order as performed by the model in Episode 1. The small blocks indicate the model operators connected in order of application; these are annotated by symbols shown in the legend indicating the type of data matched to the operators. Implicit mouse movements are those movements that must be compared to internal actions of the model; overt mouse movements are those that can be matched to overt task actions. When a data segment was not matched, a symbol indicating the data and its type appears at the bottom of the figure, between the nearest matched segments. Figure 10 shows all the segment matches across the four correct episodes. Most operators that would be verbalisable or matched by mouse actions have multiple support. Together, they suggest that most of the operators in the model are either supported by data or are required to perform the task.

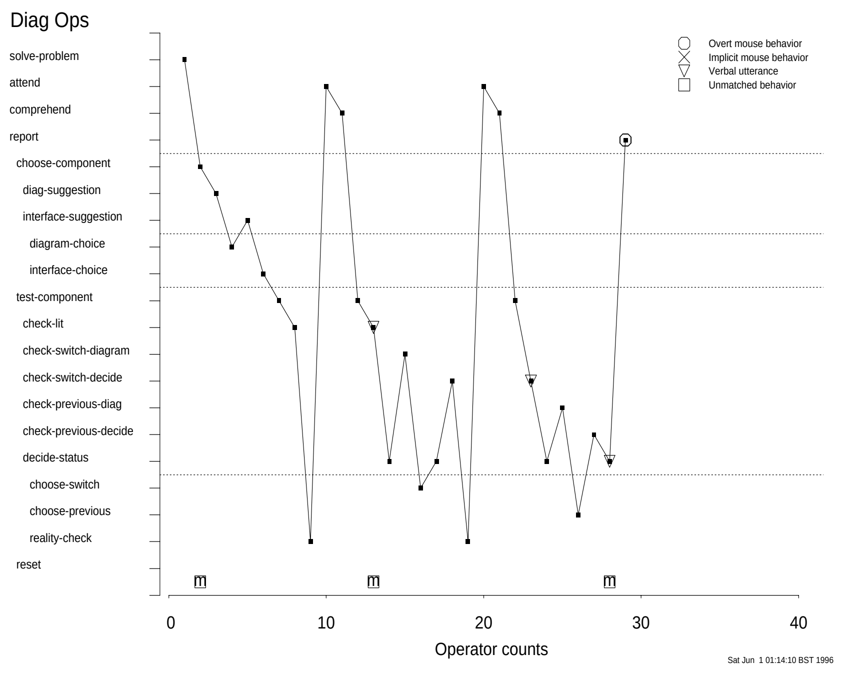


Figure 9. The support each operator in the model received in Episode 1.

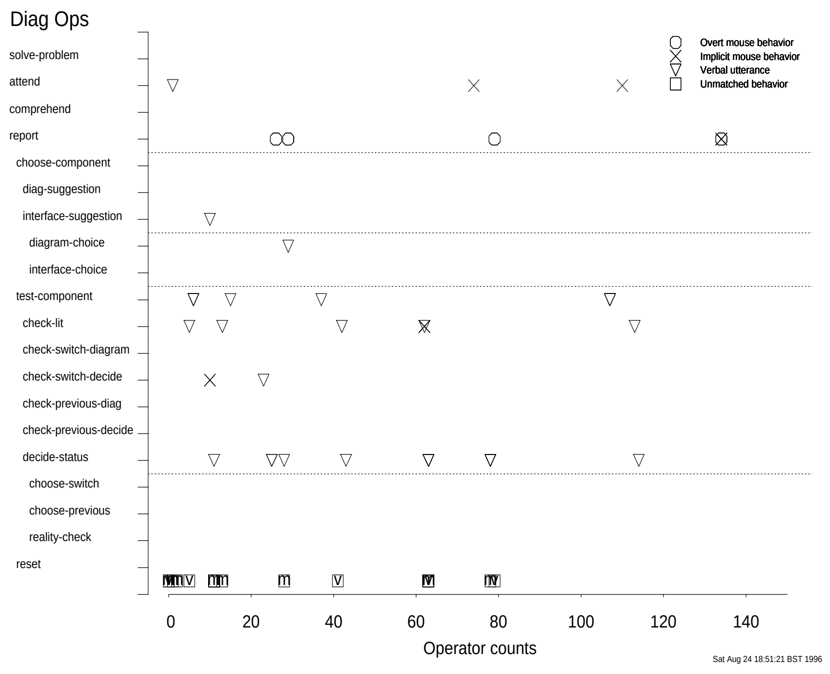


Figure 10. Support for each operator in the model across all correct episodes.

3.5 The time course of the comparison

Figure 11 shows the time course of how the model's sequential predictions matched the subject's overt and implicit behaviour while correctly solving the tasks. In these figures the y-axis represents the model's time and the x-axis the subject's time. Each correspondence between their behaviours is indicated by a symbol indicating what type of subject data was matched, and its location on the axis indicate the relative times. Unmatched subject behaviour is indicated by boxes at model time 0, with their type (mouse or verbal) indicated by the letter in the box.

If the model and the subject were solving the task at the same rate, a straight line with unitary slope would be created by the plotted correspondences. Deviations from this line indicate that the model and the subject are performing parts of the task at different relative rates, with some typical problems having diagnostic patterns (Ritter & Larkin, 1994).

In Episode 1, shown in Figure 11.a, the rates at which the subject and the model perform the task do not match so well. The relatively horizontal lines indicates that interface objects in the subject's early verbal utterances could be matched to objects the model used quite late in the process, and that the subject spent a considerable portion of their time with the solution in mind. There is also evidence that the model and the subject performed two actions in different orders; this is indicated by the negatively sloped segment around 6 s on the subject's axis.

Figure 11.b (Episode 2), shows a slightly better correspondence. Initially, the subject checks the power supply switch and the model does not. This is reasonable behaviour for the subject; the model only checks the switch if the light is off, although in this case, the switch is after the device. Then the subject and model perform the task in a similar way for about 10 s. The subject then appears to dwell on the main accumulator for the remainder of the episode, going back at the end to recap the end steps of their reasoning processing.

Figure 11.c (Episode 3), shows the subject and model performing the task in roughly comparable times until the subject jumps from the main accumulator to the next object, the laser, faster than the model makes the transition. He then dwells on examining the laser indicator's state (it is off), and later dwells on that it is broken before reporting it as the fault.

Figure 11.d (Episode 5), shows the subject and model now performing the task in roughly the same way, at a relatively constant rate. On this episode there was only one unmatched action.

3.6 Review of alignment

How well the model's actions could be aligned with the model varied in clear ways across these episodes. While the mismatches are not so bad that we should not take this model seriously, they are now large enough and systematic enough to make suggestions for improving the model in ways that the aggregate data could not do.

Early on, the subject performed actions that could not be aligned with the model and dwelt on items of interest. The early episodes included more backtracking by the subject and periods where the model was not doing anything (e.g. the plateau in Episode 2 starting at 12 s) while the subject clearly was. If we examine the utterances during these periods, we can find indirect evidence for reflection because problem solving is not going on. Although the order of the mouse movements more directly suggests this, the order of the utterances would not indicate this; the comparison with the model highlights this effect. We also saw that the subject examined the switches more often, which the model checked only when it had to, assuming default settings.

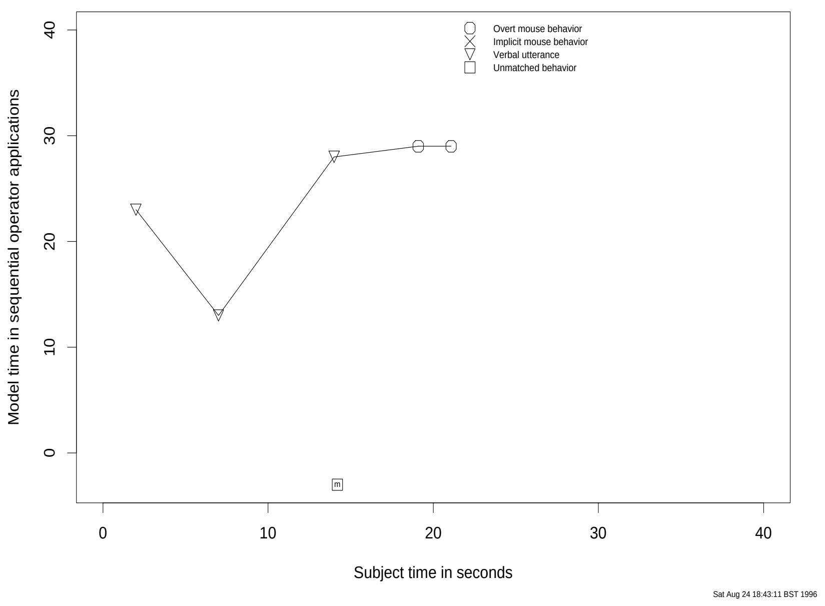


Figure 11.a. The relative processing rate plot for Episode 1.

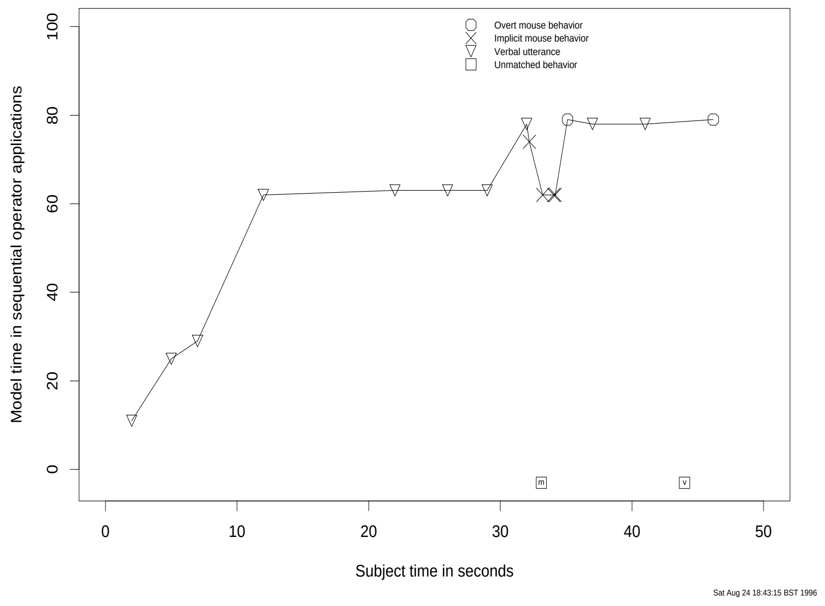


Figure 11.b. The relative processing rate plot for Episode 2.

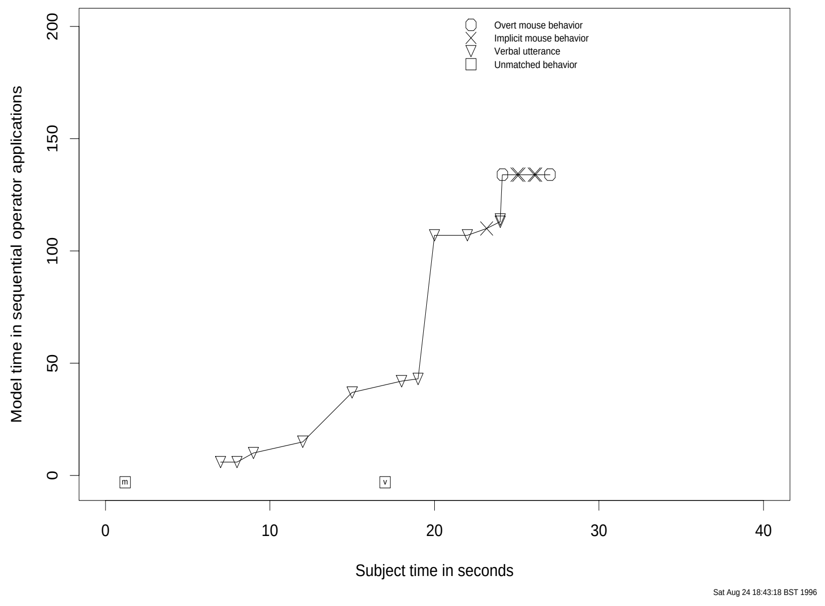


Figure 11.c. The relative processing rate plot for Episode 3.

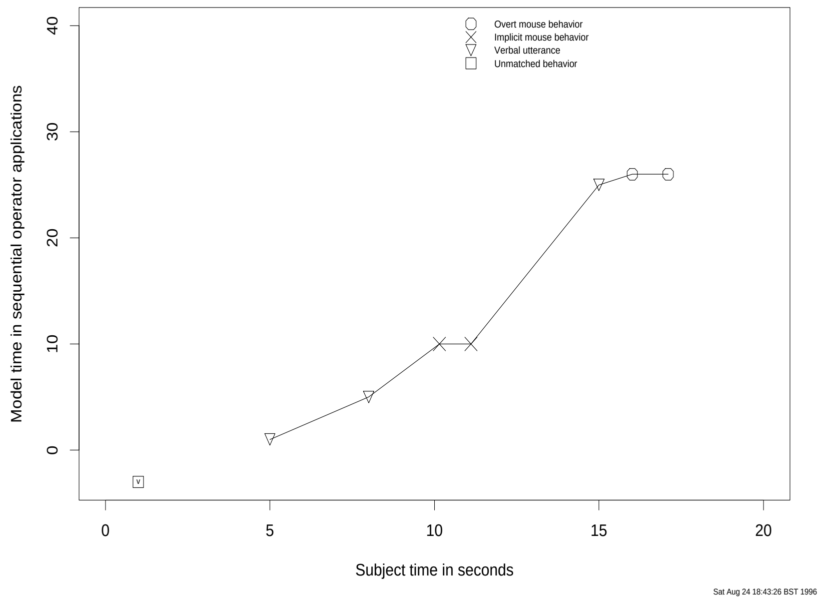


Figure 11.d. The relative processing rate plot for Episode 5.

We could only see that reflection, or a similar process, was occurring because of direct, detailed comparison between predicted and actual behaviour. The quality of the reaction time predictions of the model suggested that the model and the subject performed the task in roughly the same way, and because it accounted so well for the reaction times while learning was occurring.

We do not have complete access to what the subject was doing with the time they spent at the end of the early episodes, or exactly what was learned. We suspect that the time spent at the end of the task in some kind of apparent review served several roles in this task. It may have been fruitless time, not directly leading to improved performance except by its removal. It may have been reflection that led to increased confidence in the current strategy, which would not directly show up, but in that the increased confidence lead to less reflection. It may have led to a strategy that more closely matches the model, but this may be contradicted by how the subject and model perform fairly equivalently in the initial aspects of the task. The next step for this model is to include one or more of these mechanisms.

A tentative explanation can now be put forward for the effect in Figure 8, where the subject takes a relatively longer time than the model on the second repetition of a problem, but not on the first or third. While the protocol subject did not see a problem three times, it appears that the model is learning in a more linear way than subjects, who appear to spend more time on early trials reflecting.

It is not clear in complex models like Diag how many degrees of freedom there are. We believe that there are more degrees of freedom in Diag than in a ten term polynomial function. However, there are reasons to believe that the number of degrees of freedom are smaller than one might expect. The architecture is fixed for this model, the fundamental assumptions in it are taken as given, and offer no degrees of freedom for fitting this data. The initial knowledge of 173 production rules fits the initial behaviour. Later behaviour must be matched by the initial and learned rules that arise in problem solving, for no action is taken to adjust the learned rules other than what the architecture automatically creates. Overall, we believe that this use of a fixed architecture including learning and performance of the task substantially decreases the degrees of freedom in the model, however, we are not yet able to quantify this decrease.

The data accounted for by the model is also larger than it appears. In addition to the standard role that reaction times play, this model attempts to predict aspects of their sequential nature, that is, the order of the actions. The protocols that are matched are particularly dense because they include verbal utterances. These utterances come from a vocabulary of unknown size, so we do not know of a way to quantify the amount of additional data this provides. Overall, we do not feel that the ratio of degrees of freedom to the amount of data is unreasonable.

4. Summary and Conclusions

Diag is one of the first models that has had its sequential predictions compared with human data as they both learn using their own learning processes. That this learning effect is based on the architecture, starts to get the investment in an architecture to pay off. This admittedly has not yet happened as often as architectural proponents would like and analysts need (Ritter, 1995).

Comparing the sequential predictions of a model that learns with sequential subject data was a worthwhile exercise. The model fits the data fairly well, accounting for several types of aggregate data well and even predicts individual trial reaction time data for the majority of subjects. When the time course of behaviour within an individual episode was examined with respect to the model's sequential predictions, however, the model's behaviour did not compare so favourably with the subject's. Indeed, it suggested that the simple use of the learning mechanism provide in the model's architecture does not account for how the subject appears to have been learning through a process similar to reflection, either directly leading to new strategies or indirectly by decreasing the amount of reflection.

This detailed level of analysis suggests several important problems for understanding learning by using modelling, points out a new direction for learning research, and encourages the development the model tracing approach. The model sheds some light on how people reason with diagrams and learn through reflection. The comparison of the model's performance with sequential data pointed out some theoretical and practical problems with this methodology. And, most importantly, it has told us where to improve our model. We explore these statements in turn.

4.1 Reasoning with diagrams

We can derive some implications from the model because it fairly well reflects behaviour on reasoning with diagrams. The model supports the idea that subjects still use the diagram information in a cyclical, iterative fashion even after some practice at the task. It suggests that subjects, like the model, learn *only* the information from the diagram that is relevant to the context of each stage of the problem-solving. That is, while they use the diagram, they only learn to apply the part of the diagram that they use.

The comparisons here showed that subjects still used the diagram after solving the task. This is supported by the single subject's behaviour directly, and by how well the model's time predictions fit the aggregate data. We believe that if subjects were learning more or less about the diagram than the model was, the match would be poorer.

A deeper understanding of this task elaborates the view of diagrammatic reasoning (Larkin & Simon, 1987). The learning mechanism in the model and the subject data suggests that after practice, performance of this type of task is recognition-driven rather than model-driven behaviour. But during learning, the interface is used as an external resource to support problem-solving and recognition.

4.2 Learning through reflection

The speed up in performing this task could have come from several sources, including improved planning, faster typing and other sources. For example, in a pseudo-algebra task, Blessing and Anderson (1995) found that improved performance came not only from step-skipping but also from decreased initial planning time and decreased typing times.

The existing model and how it uses the chunking learning mechanism speeds up the model's performance using several strategies already. Some learned rules are useful for implementing operators directly, and some rules implement the effects of operators directly as augmentations to the state. Other learned rules represent the use of proposing operators in the top state, which are shifts of external attention. With practice with the task, rules based on information requiring less problem solving get created. The detailed comparison between the subject's and the model's behaviour suggests that despite the existing methods the model is not learning in all the ways the subject did, and should also reflect on its performance.

By noting the time course of the work through comparison with a model performing the task, we saw that subjects spent time *at the end of the task* performing a sub-task that disappeared with practice. The type of utterances are similar to the type of utterances towards the ends of the episodes of Anzai and Simon's (1979) subject who also reflected. The similarity of the utterances and behaviour suggests that in this task it was a more covert type of reflection that lead to a faster response. Without the explicit comparison including time, it would have been more difficult to see this behaviour as including reflection.

<u>The need for reflection.</u> This work suggests that an important cause of speedup for the subject was reflection. If this is true, part of the speedup could be attributed to an improved diagnosis strategy, perhaps including step-skipping (Blessing & Anderson, 1995), and part of the speedup may be due simply to less time spent reflecting.

If reflection is necessary to learn in this domain, then practice under highly paced conditions, where subjects did not have time to reflect, should decrease their performance. Nielsen and Kirsner (1994) have created an example model of this effect. In a highly paced problem solving task, the subject may have a stack of several subgoals. In order to learn anything, the model must find the solution in the bottom-most subgoal and learn a rule before the situation changes destroying the stack. If the external situation is too highly paced, the model cannot solve the initial problem, and thus cannot reason about more complex problems.

Including reflection-based learning appears to be necessary for cognitive architectures to progress. Most learning mechanisms provide simple speedup. The rule strengthening mechanism in ActR and rule composition (Lewis, 1987) do not support strategy changes very well. The most typical uses of the chunking mechanism in Soar also will not produce strategy changes (although, see Bauer & John, 1995, and Bass et al., 1995).

"Listening to the architecture" failed here. Another interesting aspect of this work is that the need for and role of reflection in learning cannot come initially from 'listening to the architecture' (e.g. Newell, 1990, p. 331). The need for reflection does not arise, at least not directly, out of this architecture, although some architectures may choose to support it directly (Ram, et al., 1995). Reflection appears to arise from the subject's knowledge—knowledge that suggests that performance is not optimal; knowledge that performance could be improved; and knowledge about how to improve performance. In Anzai and Simon's (1979) model that learns through a type of reflection, they note the interplay between general learning mechanisms and the importance of specific task knowledge. This is particularly true for this task.

On the other hand, the Soar architecture can provide support for reflecting and offers some suggestions about how to reflect in a useful way. A preliminary model of this type of reflection uses an operator to consider all the knowledge available after an action to decide whether or not to reflect, an abstract copy of the problem space to consider what else could have been done, and creates new rules to transfer what is learned in the reflective space directly into modified behaviour (Bass, et al., 1995). It is consistent with IML (Ram, et al., 1995),, except it attempts to use existing mechanisms within the architecture.

4.3 Further development of trace-based protocol analysis

The analysis of the sequential behaviour was fairly successful. It helped understand where the model performed well and poorly, and it was able to suggest that the subject was reflecting in some way. It also raised some serious questions about how to apply this model tracing technique where learning occurs. We do not assume that these are fatal flaws for this methodology, but wish to highlight the difficulties in some detail to assist future work.

Despite the breadth and depth of these difficulties, none of them preclude using this approach to develop models. At the end of the day, two questions remain about models: are they worth taking seriously (which this one is), and can you tell where to improve them (which we could despite these difficulties). These difficulties restrict seeing every place of improvement; they restrict what we can see and how quickly we can see it, but they do not totally obscure the model so that it cannot be improved.

Avoiding the danger of averaging across even a single trial. This test of using sequential data demonstrated an additional way that models can be wrong. Diag successfully fits reaction times for a series of tasks while learning occurred but it did not match the time course of behaviour generating those reaction times. The main result that the sequential analysis of the sequential data revealed was that reaction times can be well modelled while their internal generation is incorrect. This is doubly important in this case because the analysis was able to illuminate a new learning approach that subjects and earlier work using aggregate measures did not discover (Bibby & Payne, 1993; Bibby & Reichgelt, 1993; Kieras & Bovair, 1984).

The dangers of aggregating over individual data (Delaney, Reder, Staszewski, & Ritter, in press; Siegler, 1987; Walker & Catrambone, 1993) can be expanded to include the danger of aggregating over a *single trial from a single individual*. When several steps make up a reaction time, then it is possible to have the right basic units but get them in the wrong order or exclude some in a regular way so as to still get a reasonable correlation with individual reaction times. When there are ambiguities with a model or differences between models, one of the ways forward will be to use within-trial timing to resolve ambiguities.

<u>Formalising partial and multiple matches.</u> Diag operationalises a theory of performance in a fine grained way. In doing so, it attempts to extract much more from the data beside the time it took to create the response. However, decreasing the grain of the comparison and including learning strains the previous theory of comparison. This raises several questions about how to formalise this comparison in terms of partial matches.

With this fine grained model, multiple aspects of a match become visible. With a smaller comparison grain size, the several aspects to a match become visible:

(a) that the action occurred;

(b) that the action occurred in the correct order;

(c) that the data structure was mentioned;

(d) that the utterance included each of the possibly many attributes and values that the model's data structure had.

If we maintain the view that comparisons should indicate where the model can be improved, future comparisons may stop reporting whole behaviours as being matched, but instead reporting a set of sub-comparisons. The subcomparisons are not as useful for keeping score as they are for finding where the model can be improved.

This fine grained a model can lead to resegmentation of utterances. Consider, perhaps, what could be segmented as a complete thought:

```
That goes to the main accumulator which is on and lit so
that's OK
```

When this is compared with the model's trace, in order to compare completely comparable items, it must be resegmented into several smaller items:

```
That goes to the main accumulator

which is on

and lit

so that's OK
```

So we are able to present a more complete and theoretical definition of what comprises a complete thought—items that appear in the model's trace. How to segment still remains a problem for sections that are not modelled. Any atomic utterance that is noted as 'verbal uncoded' may be ambiguous in its granularity of decomposition unless there are model components that can be used to decompose it. If the model suggests that the utterance: "That goes to the main accumulator which is on and lit so that's OK" decomposes into four steps, all matched, then "That goes to the laser bank and it's on and lit so that's OK" should also decompose into four steps, even if all those segments currently are uncodable.

In order for trace-based protocol analysis to bring out the fact that, say, model A, which has the right elements in the wrong order, is not as good as a model B, which has the right elements in the right order, we need a more complete coding scheme that takes account of these factors. For example, alignments with the subject on the previous version of Diag (v.16) did not differ much on pure 'hits', but qualitatively the newer version of Diag had much stronger ties in the verbal and mouse matches in the order and level of detail matched.

Utterances may match multiple objects in the model, for the model objects might not be atomic or might appear as part of several problem solving steps. For example, the utterance "The main accumulator is on" may match two operators and their corresponding state information: selecting the main accumulator (operator and/or result) and checking its state (operator and/or result).

Many utterances also *imply* substeps that are necessary elements in the task (and remain as sub-steps even in a fully-chunked or automatic model). For example, if the subject's first utterance is "PS is lit", then the subject must have gone through a decision of what to check first or else decided to just use the fovea's contents to start. Strictly speaking, the mismatched data that suggested that subjects were reflecting, could have been matched to grossly out of order model actions. When and how to align actions matched out of order remains to be formalised.

Clearly, models can still be compared with data, and doing so is useful for improving the model. At this point, as our models become more complicated, how to do this in a clear, consistent and efficacious manner requires additional thought.

Supporting comparisons of hierarchical model structures. Hierarchical models pose a new problem when aggregating support for a model structure. In the past, if a single model structure gave rise to behaviour and a subject exhibited this behaviour, the structure could be simply credited as supported by data. In Diag, because of its hierarchical nature, there are often several structures active at any one time, so there is always a question of where to assign credit. If the state information is shared between the two levels, which level is considered as the corresponding structure? This problem exists even if you initially create the minimal model necessary to do the task, and expand it only to account for subject data (e.g., as in Peck & John, 1992 and in most other models).

There are several ways out of this dilemma of how to assign credit for the correspondences, none of them completely satisfactory. One could simply provide support to only the top level in the goal hierarchy. Lower levels would be allowed, but because they do not directly give rise to external behaviour, they would not receive support (and would rely on the necessity to do the task to avoid being unfalsifiable). One could provide support to the lowest level, which is what we did here. This tends to leave the top levels unsupported until learning occurs. An additional approach is to provide support for all operators or their states in the hierarchy when the subject and model correspond. This seems a bit generous.

If we continue with this learning mechanism and this type of model, we will need to consider passing on support to higher levels. A structure supported at a lower level may be incorporated in a higher level structure. This could give rise to structures for which no direct evidence has been found as a whole, but which has support for each of its components.

While even Newell and Simon (1972) matched behaviour to production rules acting like operators, strictly speaking, this is not correct and has lead us into this dilemma. Under strict terms, only like can be compared with like. The subjects' utterances, or, more fairly, the references of those utterances, are references to internal states. These should only be compared with state representations in the model. Similarly, output is not a process (at least in the Soar architecture yet), but a state annotation picked up by the motor system for actual action.

Operator/state equivalence breaks down. Newell and Simon (1972, p. 157) noted that subjects' sequential behaviour could be compared with the behaviour of a process model in terms of either the states or the operators (where simple productions have been sometimes treated as operators). This has been a fruitful simplification often used (e.g. John & Vera, 1992). It reduces the size of the model's trace for comparison, and simplifies the matching process.

This work indicates that there will be difficulties keeping this equivalence as cognitive models become more sophisticated. There are several problems in assigning credit and performing the comparison. Operators in this model were more complicated than previous models. They take arguments that can give rise to slightly different instantiations with slightly different behaviours. Does the operator have support if the operator is correct and the arguments are wrong to correspond to the subject's behaviour? When composition and chunking of operators or productions are possible, how is credit accumulated across trials and across learning episodes? It appears that the safest course is to back up to comparing talk-aloud utterances about state information with traces of the state of the model.

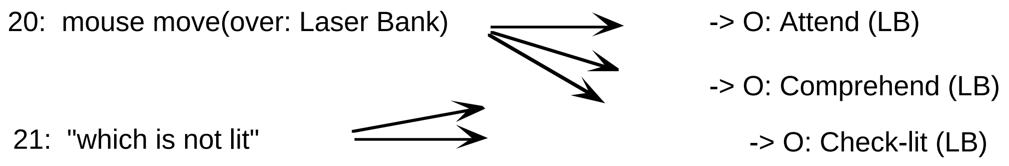
Increased model complexity also causes some problems when comparing the subject's behaviour with the model's state information. As process models have developed, their data structures have become more sophisticated. The possibility of an atomic matching of an utterance is becoming less possible. Representations are developing sub-structures, and the comparison is sometimes no longer atomic. There appears to be a trade off between the ability to compare the model and its power to perform complicated tasks.

So far, there has been support in several studies for having mouse moves over an external display show support for the structures in a model that use the information under the mouse. As we move towards modelling mouse and eye movements themselves (Anderson, Matessa, & Douglass, 1995; Bass, et al., 1995; Baxter & Ritter, In press), how can we use the mouse-move-inferred comparisons when perception is modelled? Modelling the comprehension in detail will actually make the comparison process more difficult because the mouse movement should now be compared with perception in the model, rather than simply support for cognition. Figure 12 shows how this has required splitting the use of the mouse and verbal data into a comparison with perception and comprehension operators. This correspondence and that external information has been acquired must be remembered by the analyst and used again to support the cognitive action that uses the data later.

These problems make detailed comparison even less appealing because they drastically increases the workload of the analyst and obscure the interpretation of operators, the process generators in the model. Many possible solutions currently appear to be either wrong or intractable. Our hope is that the architecture will clear up, offering a way back towards the basic equivalence of operators and states.

Knowing where to partition operators. As we created a model of a 'Switch-Changing' Task in addition to the fault-finding task, it emerged that while there are guidelines about *how* to partition behaviour into operators and states (Newell, Yost, Laird, Rosenbloom, & Altmann, 1991), there

Figure 12. Example match of verbal and mouse movements to multiple model actions in Episode 3, segments 20-21.



are no guidelines on *where* to partition behaviour, particularly internal behaviour, into operators. Numerous architectures support operators, but there are no guidelines on when and where to split behaviour into operators. On one level, this may be the result of the increasing complexity of models as well as (perhaps) the undecidability of models (e.g., as argued for by Anderson, 1990), but some informal or sketching guidelines should be possible about how to partition knowledge to solve tasks. People can be trained to create models that account well for transfer (Kieras & Bovair, 1986; Nerb, et al., 1993; Singley & Anderson, 1989), but it is acknowledged they have to be trained at certain labs (Kieras, 1985, p. 72).

Problems of alignment due to learning. Comparing the predictions of models that learn with data raises some problems that Newell and Simon never had. Process models typically include hierarchical organisation, and in Soar this must be the case for the learning mechanism to work. That is, there must be a problem space that has reached an impasse, and a lower problem space that solves the problem thus creating a new rule.

This approach raised several questions about how to do the comparison.

(a) If a verbal protocol element originally matches to a structure in a subgoal, how to aggregate match statistics when in a later trial multiple structures are replaced by a single structure?

(b) Can we assume that the granularity of the verbal protocol reflects the granularity of the subject's task decomposition? (We doubt it.)

(c) In an ideal situation, the protocol also becomes 'compiled' in the same places as the problem solving becomes chunked. The utterances should reflect increasingly high-level state information at the same rate at which learned rules replace low-level goals. This appears to be supported by results in knowledge acquisition in expert systems and in protocol analysis (Ericsson & Simon, 1993, p. 126).

Learning increases the need for data as well. Initial behaviour may become masked by later learning, so that some structures will only be used in the early stages of the model's behaviour or in novel situations. This increases the amount of data needed to test the model. An equally serious problem is that the structures may be mutable, and support for an early version of the structure should not be carried through to a later version of the same structure.

Overall, however, learning should improve the decidability of models. With learning, the internal structures used early in the task must lead to or be consistent with the later learned behaviour. This increases the impact any episode can have on a model, for it suggests that a model should be consistent with behaviour over practice.

4.4 Practical, usability problems of trace-based protocol analysis

Trace-based protocol analysis (TBPA) is still performed much by hand. The speed and acceptability of TBPA will depend not just on how to perform it, but how to perform it in a timely manner. Therefore, we report several practical problems we found that influenced the analyses. We note them here for use in developing the methodology.

SPA requires familiarity with tools. We used the alignment tool in SPA (Ritter & Larkin, 1994) which is based on the GNU Emacs editor. While we had previously found that existing Emacs users could pick up and use SPA rather directly (Ritter, 1992), users not familiar with Emacs found the combination of SPA and learning Emacs quite daunting. We have created a manual for the underlying spreadsheet that will ameliorate this problem to a certain extent, and in the future we will have to budget time to teach Emacs as well. The alignment tool may become part of the GNU-Emacs general release, so it has the potential to be installed at most sites automatically. The automatic alignment algorithm has been incorporated into a more menu-driven tool, but this too requires practice (Sanderson, et al., 1994).

Automatic alignment is most useful for unambiguous actions. The automatic alignment process, which worked well and to great advantage in previous work (Ritter & Larkin, 1994), was less useful here. The automatic alignment algorithm works much better with unambiguous data and where there is a high proportion of unambiguous data (e.g. mouse clicks) to constrain the match of more ambiguous data (e.g. verbal utterances). The ratio of verbal utterances to mouse actions was higher in this data set, so the automatic alignment was less useful here. We should explore how to provide better verbal matches, perhaps automatically generating possible correspondences from the model's trace, rather than simply allowing the analyst to enter possible correspondences to look for (T. Simon, personal communication, July 1994, generated a set of regular expressions for aligning verbal utterances with a model trace for solving the Tower of Hanoi). This approach, while certainly not theoretically complete, would in many situations be useful none-the-less.

Interpretation requires the model designer at hand. We found that we often had to have the model's designer available to interpret the trace to data correspondences. A solution appears to be to have the designer document all of the model's operators, structures, and the learning mechanisms clearly so that others can do the alignment. This is generally a good thing.

A clean state change trace. Operator or rule applications and the states they create have often been seen as equivalent descriptions of a model. This has been used successfully in the past (e.g., Peck & John, 1992; Ritter & Larkin, 1994; VanLehn, 1991) and supported in the general theory (Newell & Simon, 1972, p. 157). This appears to be a useful and often tenable simplification. The work here suggests that this is not tenable when learning occurs or where operators are complex. State traces are the backup and more fundamental approach (Ericsson & Simon, 1993). They are more detailed and verbose than operator traces. They must be made simple and easy to use when using a complex model that learns.

4.5 Implications for future models and modellers

Given that we have used an iterative technique (Ritter & Larkin, 1994), based on Grant's (1962) approach for developing models by noting where they can be improved rather than proving them, a natural question is where to improve the model. A secondary question is to where the technique can be improved. We can note several places that this work has suggested improvements and the direction we are heading.

The current model is incomplete in numerous ways. The model accounts for less than 10% of the variability in trial reactions times for some subjects (two out of ten). Examining additional subjects will provide additional strategies for performing and learning in this domain. We do not yet, for example, model the interaction in detail; improvements in motor skills with practice are omitted, and input from vision is not fully modelled. There is no account of how the task is acquired in the first place.

In addition to including new capabilities and comparison with aggregate data, models that learn must continue to be compared with single subject sequential data. Our protocol study demonstrated the dangers of averaging over even a single reaction time. This level of comparison allows us another way to examine the validity of the learning mechanism in addition to learning rate, percent correct, and time to perform the task.

Applying the SPA software and techniques developed elsewhere (Ritter & Larkin, 1994) to a new data set and new model made several suggestions for improvements. A problem that we have been able to solve is making the automatic alignment spreadsheet easier to use. New users have had some problems learning how to use the Dismal spreadsheet, perhaps just because it is a new piece of software. We have created a manual for it, and it has received some use with favourable comments. The same algorithm has been incorporated now into MacShapa as well, which offers an alternative, more menu-driven interface.

Building the model and gathering the data still remains expensive, and reducing the effort required remains important. The use of MacSHAPA (Sanderson, et al., 1994) or other tools for directly transcribing and coding from video offer to reduce the effort for gathering and analysing this data.

The aspects of the SPA environment developed to support Soar modelling in trace-based protocol analysis became defunct with the Soar6 release in 1993. The environment for developing Soar models continues with the SDE (Hucka, 1994), a structured, integrated editor similar but more extensive than its predecessor, and the TSI (Ritter & Baxter, 1996) a graphic interface for Soar. While these systems no doubt help create models, they will not solve the main problem because it is the lack of mechanisms and understanding of how learning and knowledge representations influence current and later behaviour.

There are only starting to emerge 'good programming' guidelines for process models. These guides have existed in simpler languages for quite a while, and note how to perform common actions in a way that is easy for an analyst to understand, that is efficient to perform, and that does not interfere with other behaviours. This is an area that urgently needs attention. The creation of a tutorial (Ritter & Young, 1994) is a very small step in the right direction. Extensive work will be required from each architecture's community in this area.

Inspired by this result, we have started to implement a model of reflection in Soar in another domain (briefly reported in Bass, et al., 1995). We have started to sketch out a general statement of the types of knowledge required and operationalised them in a limited way. Most knowledge in Soar and other process models is used in the same goal structure as it as learned. Hierarchical rules like those generated in learning how to press buttons in the Seibel task are a classic example (Rosenbloom & Newell, 1987). New rules are learned while performing the task, and then apply when next solving the same type of task. A reflection-based mechanism proposes to create a new type of learning mechanism in Soar, one that gives rise to rules that are created after performing the task, and are not used in the same goal stack as when they are learned. This will be an exciting general learning mechanism, providing another implementation of reflective learning (Chi, Bassok, Lewis, Reimann, & Glaser, 1989; Ram, et al., 1995).

5. References

Altmann, E. M., Larkin, J. H., & John, B. E. (1995). Display navigation by an expert programmer: A preliminary model of memory. In I. R. Katz, R. Mack, & L. Marks (Eds.), *Proceedings of the CHI '95 Conference on Human Factors in Computer Systems,* (pp. 3-10). New York, NY: ACM SIGCHI.

Anderson, J. R. (1990). *The adaptive character of thought*. Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Anderson, J. R. (1993). *Rules of the mind*. Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Anderson, J. R., Matessa, M., & Douglass, S. (1995). The ACT-R theory and visual attention. In *Proceedings of the Seventeenth Annual Conference of the Cognitive Science Society,* (pp. 61-65). Hillsdale, NJ: Lawrence Erlbaum Associates.

Anzai, Y., & Simon, H. A. (1979). The theory of learning by doing. *Psychological Review*, *86*, 124-140.

Bass, E. J., Baxter, G. D., & Ritter, F. E. (1995). Using cognitive models to control simulations of complex systems. *AISB Quarterly*, *93,* 18-25.

Bauer, M. I., & John, B. E. (1995). Modeling time-constrained learning in a highly interactive task. In I. R. Katz, R. Mack, & L. Marks (Eds.), *Proceedings of the CHI '95 Conference on Human Factors in Computer Systems,* (pp. 19-26). New York, NY: ACM SIGCHI.

Baxter, G. D., & Ritter, F. E. (In press). Model-computer interaction: Implementing the action-perception loop for cognitive models. In *The 1st International Conference on Engineering Psychology and Cognitive Ergonomics,* (pp. 8 pages). October 1996, Stratford-upon-Avon:

Bibby, P. A., & Payne, S. J. (1993). Internalisation and the use specificity of device knowledge. *Human-Computer Interaction, 8*, 25-56.

Bibby, P. A., & Payne, S. J. (1996). Instruction and practice in learning to use a device. *Cognitive Science*, *20*(4), 539-578.

Bibby, P. A., & Reichgelt, H. (1993). Modelling multiple uses of the same representation in Soar. In A. Sloman & et-al. (Eds.), *Prospects for Artificial Intelligence,* (pp. 271-280). Amsterdam: IOS Press.

Blessing, S. B., & Anderson, J. R. (1995). How people learn to skip steps. *Submitted*.

Chi, M. T. H., Bassok, M., Lewis, M. W., Reimann, P., & Glaser, R. (1989). Self-explanations: How students study and use examples in learning to solve problems. *Cognitive Science*, *13*(2), 145-182.

Delaney, P. F., Reder, L. M., Staszewski, J. J., & Ritter, F. E. (in press). The strategy specific nature of improvement: The power law applies by strategy within task. *Psychological Science*.

Ericsson, K. A., & Simon, H. A. (1993). *Protocol analysis: Verbal reports as data*. Cambridge, MA: The MIT Press.

Feigenbaum, E. A., & Simon, H. A. (1984). EPAM-like models of recognition and learning. *Cognitive Science*, *8*, 305-336.

Feldman, J. (1962). Computer simulation of cognitive processes. In H. Borko (Ed.), *Computer applications in the behavioral sciences* (pp. 336-359). Englewood Cliffs, NJ: Prentice-Hall.

Grant, D. A. (1962). Testing the null hypothesis and the strategy and tactics of investigating theoretical models. *Psychological Review*, *69*, 54-61.

Hucka, M. (1994). *The Soar Development Environment*. Ann Arbor, MI: Artificial Intelligence Laboratory, U. of Michigan. Also available through http://www.cs.cmu.edu/afs/cs/project/soar/www/soar-archive-software.html.

Huffman, S. B., & Laird, J. E. (1995). Flexibly instructable agents. *J. of AI Research*, *3*, 271-324.

Irwin, D. E., & Carlson-Radvansky, L. A. (1996). Cognitive suppression during saccadic eye movements. *Psychological Science*, *7*(2), 83-87.

John, B. E., & Vera, A. H. (1992). A GOMS analysis of a graphic, interactive task. In *CHI'92 Proceedings of the Conference on Human Factors and Computing Systems (SIGCHI),* (pp. 251-258). New York, NY: ACM Press.

Jones, G., & Ritter, F. E. (In press). Modelling transitions in childrens' development by starting with adults. In *European Conference on Cognitive Science,* . Manchester, UK.

Kennedy, A., & Baccino, T. (1995). The effects of screen refresh rate on editing operations using a computer mouse pointing device. *The Quarterly Journal of Experimental Psychology*, *48A*(1), 55-71.

Kieras, D. (1985). The role of cognitive simulation models in the development of advanced training and testing systems. In N. Frederiksen, R. Glaser, A. Lesgold, & M. G. Shafto (Eds.), *Diagnostic monitoring of skill and knowledge acquisition* (pp. 365-394). Hillsdale, NJ: LEA.

Kieras, D., & Bovair, S. (1984). The role of a mental model in learning how to operator a device. *Cognitive Science*, *8*, 255-273.

Kieras, D., & Bovair, S. (1986). The acquisition of procedures from text: A production system model. *Journal* *of Memory and Language*, *25*, 507-524.

Kitajima, M., & Polson, P. G. (1996). A comprehension-based model of exploration. In M. J. Tauber, V. Bellotti, R. Jeffries, J. D. MacKinlay, & J. Nielsen (Eds.), *Proceedings of the CHI '95 Conference on Human Factors in Computer Systems,* (pp. 324-331). New York, NY: ACM SIGCHI.

Laird, J. E., Newell, A., & Rosenbloom, P. S. (1987). Soar: An architecture for general intelligence. *Artificial Intelligence*, *33*(1), 1-64.

Laird, J. E., & Rosenbloom, P. S. (1995). Evaluation of the Soar cognitive architecture. In D. M. Steier & T. M. Mitchell (Eds.), *Mind matters* (pp. 1-50). Hillsdale, NJ: LEA.

Larkin, J. H. (1981). Enriching formal knowledge: A model for learning to solve textbook physics problems. In J. R. Anderson (Ed.), *Cognitive skills and their acquisition* (pp. 311-334). Hillsdale, NJ: LEA.

Larkin, J. H., & Simon, H. A. (1987). Why a diagram is (sometimes) worth ten thousand words. *Cognitive Science*, *11*(1), 65-99.

Lewis, C. H. (1987). Composition of productions. In D. Klahr, P. Langley, & R. Neches (Eds.), *Production system models of learning and development* (pp. 329-358). Cambridge, MA: MIT Press.

Lewis, R. L. (1993). An architecturally-based theory of human sentence comprehension. Ph.D. thesis, Carnegie-Mellon University.

Nerb, J., Krems, J., & Ritter, F. E. (1993). Rule learning and the power law: A computational model and empirical results. In *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*, (pp. 765-770). Hillsdale, NJ: Lawrence Erlbaum Associates.

Newell, A. (1990). *Unified theories of cognition*. Cambridge, MA: Harvard University Press.

Newell, A., & Simon, H. A. (1972). *Human problem solving*. Englewood Cliffs, NJ: Prentice-Hall, Inc.

Newell, A., Yost, G. R., Laird, J. E., Rosenbloom, P. S., & Altmann, E. (1991). Formulating the problem space computational model. In R. F. Rashid (Ed.), *Carnegie Mellon Computer Science: A 25-Year commemorative* (pp. 255-293). Reading, MA: ACM-Press (Addison-Wesley).

Nielsen, T. E., & Kirsner, K. (1994). A challenge for Soar: Modeling proactive expertise in a complex dynamic environment. In *Singapore International Conference on Intelligent Systems (SPICIS-94),* (pp. B79-B84).

Peck, V. A., & John, B. E. (1992). Browser-Soar: A computational model of a highly interactive task. In *Proceedings of the CHI '92 Conference on Human Factors in Computer Systems,* (pp. 165-172). New York, NY: ACM.

Ram, A., Narayanan, S., & Cox, M. T. (1995). Learning to troubleshoot: Multistrategy learning of diagnostic knowledge for a real-world problem-solving task. *Cognitive Science*, *19*(3), 289-340.

Ritter, F. E. (1992). TBPA: A methodology and software environment for testing process models' sequential predictions with protocols. PhD thesis, Carnegie-Mellon University.

Ritter, F. E. (1995). Review of "Soar: An architecture in perspective". *Philosophical Psychology*, *8*(3), 301-305.

Ritter, F. E., & Baxter, G. D. (1996). Able, III: Learning in a more visibly principled way. In U. Schmid, J. Krems, & F. Wysotzki (Eds.), *Proceedings of the First European Workshop on Cognitive Modeling,* (pp. 22-30). Berlin: Forschungsberichte des Fachbereichs Informatik, Technische Universität Berlin.

Ritter, F. E., & Larkin, J. H. (1994). Using process models to summarize sequences of human actions. *Human-Computer Interaction*, *9*(3&4), 345-383.

Ritter, F. E., & Young, R. M. (1994). Practical introduction to the Soar cognitive architecture: Tutorial report. *AISB Quarterly*, *88*, 62.

Rosenbloom, P. S., & Newell, A. (1987). Learning by chunking, a production system model of practice. In D. Klahr, P. Langley, & R. Neches (Eds.), *Production system models of learning and development* (pp. 221-286). Cambridge, MA: MIT Press.

Ruiz, D., & Newell, A. (1989). Tower-noticing triggers strategy-change in the Tower of Hanoi: A Soar model. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society,* (pp. 522-529).

Rumelhart, D. E., McClelland, J. L., & group, t. P. r. (1986). *Parallel distributed processing: Explorations in the microstructure of cognition. Volume 1: Foundations*. Cambridge, MA: The MIT Press.

Sanderson, P. M., Scott, J., Johnston, T., Mainzer, J., Watanabe, L., & James, J. (1994). MacSHAPA and the enterprise of exploratory sequential data analysis (ESDA). *International Journal of Human-Computer Studies*, *41*, 633-681.

Siegler, R. S. (1987). The perils of averaging data over strategies: An example from children's addition. *Journal of Experimental Psychology*, *115*, 250-264.

Singley, M. K., & Anderson, J. R. (1989). *The transfer of cognitive skill*. Cambridge, MA: Harvard University Press.

VanLehn, K. (1989). Learning events in the acquisition of three skills. In *Proceedings of the 11th Annual*

Conference of the Cognitive Science Society, (pp. 434-441). Hillsdale, NJ: Lawrence Erlbaum Associates.

VanLehn, K. (1991). Rule acquisition events in the discovery of problem-solving strategies. *Cognitive Science*, *15*(1), 1-47.

VanLehn, K., & Ball, W. (1991). Goal reconstruction: How Teton blends situated action and planned action. In K. VanLehn (Ed.), *Architectures for intelligence* Hillsdale, NJ: Lawrence Erlbaum Associates.

VanLehn, K., & Jones, R. M. (1993). Learning by explaining examples to oneself: A computational model. In S. Chipman & A. L. Meyrowitz (Eds.), *Foundations of knowledge acquisition: Cognitive models of complex learning* (pp. 25-82). Boston, MA: Kluwer.

vanSomeren, M. W., Barnard, Y. F., & Sandberg, J. A. C. (1994). *The Think Aloud Method: A practical guide to modelling cognitive processes*. London/San Diego: Academic Press.

Walker, N., & Catrambone, R. (1993). Aggregation bias and the use of regression in evaluating models of human performance. *Human Factors*, *35*(3), 397-411.

Wiesmeyer, M. D. (1991). An operator-based attentional model of rapid visual counting. In *Proceedings of the Thirteenth Annual Conference Cognitive Science Society*, (pp. 552-557). Hillsdale, NJ: Lawrence Erlbaum Associates.