has less operator applications.

Figures 7-43 and 7-44 show the model fit displays for the modified version of Browser-Soar next to the original versions. These two displays show that the revised model has a denser level of support, the lines connecting the corresponding model and subject actions are closer together, and the RMSD and mean average deviation are lower. The rate of decision cycles to seconds ratio is also closer to the predicted mean, and visually the fit appears to be better. The modified version has slightly worse r^2, more so when the model time unit is decision cycles (.69 versus .59) than for operator applications (.78 versus .75). The correspondence rates in decision cycles and operator applications per second for the modified model also go down, as less is done.

It is hard to tell if these differences are important. It would perhaps become easier to tell after further revisions of the *Evaluate-current-window* operator, and with a more proper regression line (Kadane et al., 1981; Larkin et al., 1986). These results do point out that it is hard to distinguish learning on the single problem space level at this time grain. In order to clearly distinguish these two problem space representations we would have to look at more episodes, more subjects, or further constraints from data. Given the lack of real difference, parsimony would argue for using the simpler, modified version of Browser-Soar.

This analysis also calls into question the strict interpretation used. The subject must decide to move the mouse. The operators that were removed originally represented this choice. With a different interpretation function, these operators would have been supported and would not have been removable. As noted in the list of corrections available when the model's predictions mismatch the data (Table 2-6), the interpretation function can also change. This case raises the question of how to interpret data given Soar's hierarchical operators and state representation. This may remain a problem for some time.

7.5 Testing and extending the sequentiality assumptions of protocol generation theory

As noted in their initial description, the relative processing rate displays allow the sequentiality assumption of Ericsson and Simon's (1984) theory of verbal protocol production to be tested. That is, if verbalizations are produced in the order that the corresponding data structures appear in working memory. There is another aspect to this assumption, that inputs to operators will be reported before their outputs, but is a more specific form that will not be directly tested unless we run into problems. A model of what appears in working memory is currently necessary to test this assumption. There are no other ways to tell when information enters working memory, and thus that it is reported in order. Having a model of the contents of working memory also allows use to judge if the verbalizations are retrospective or prospective.

Browser-Soar provides predictions of the contents of working memory while using a specific on-line help system. By examining the relationship of these predictions with the subject's verbal utterances in the ten Browser-Soar episodes, the sequentiality assumption can be tested.

The predictions of the external task actions (mouse movements and button presses) can also be compared with the contents of working memory, but because getting the order of the external actions the same for both model and subject is essential for performing the task, in a well developed model like Browser-Soar there is not likely to be many mismatches. What will be interesting though, is using the external actions to compute how later (or early) the verbal utterances are.

Finding that this holds will not be an iron-clad proof that this assumption holds. If it is an assumption, then it cannot be proven, only shown that we meet it. If it is treated more as part of the theory of verbal protocol production, then there may be similar models of browsing behavior where the information is reported in a different order, and that the current set of verbal protocols would not match sequentially.

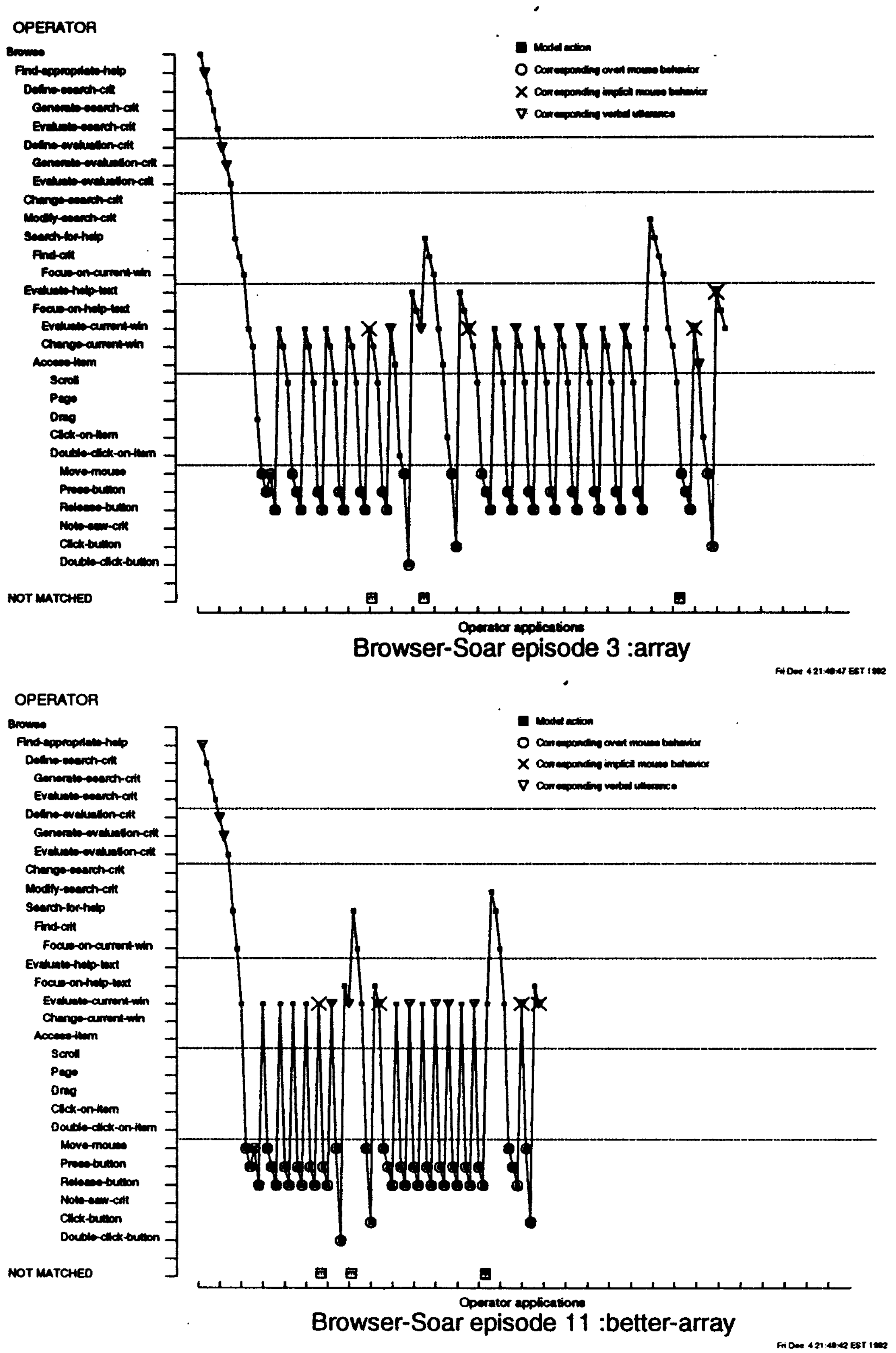
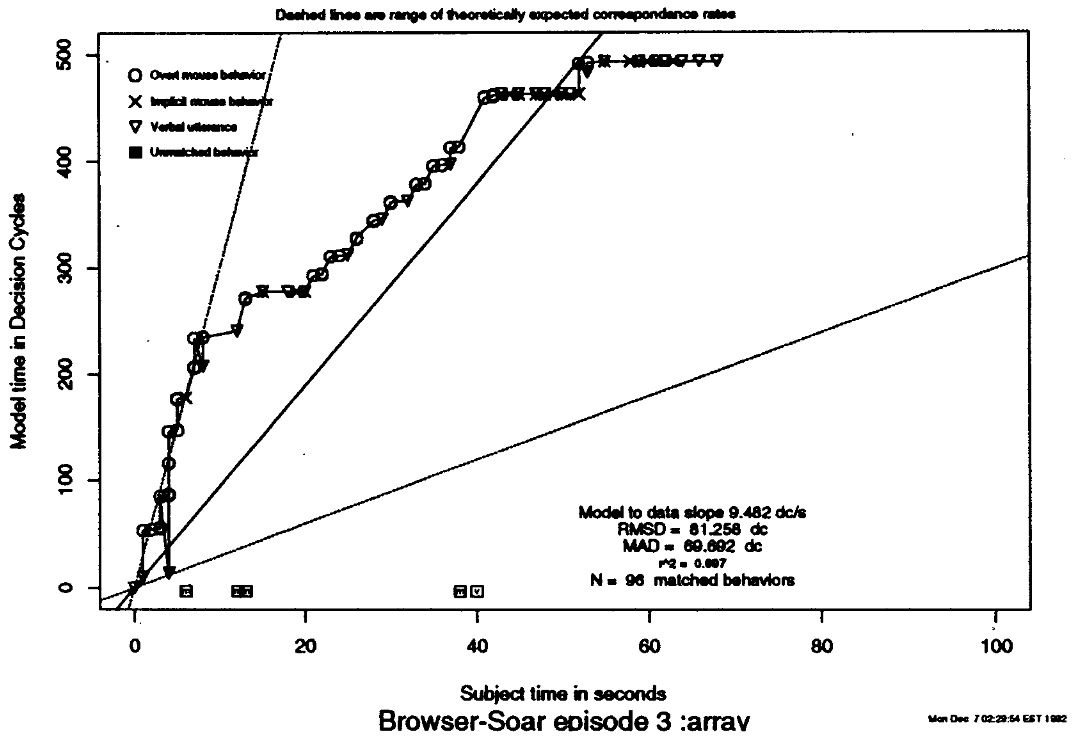


Figure 7-42: Operator support displays for the Array episode. The original Browser-Soar predictions are on the top, and the modified version on the bottom.



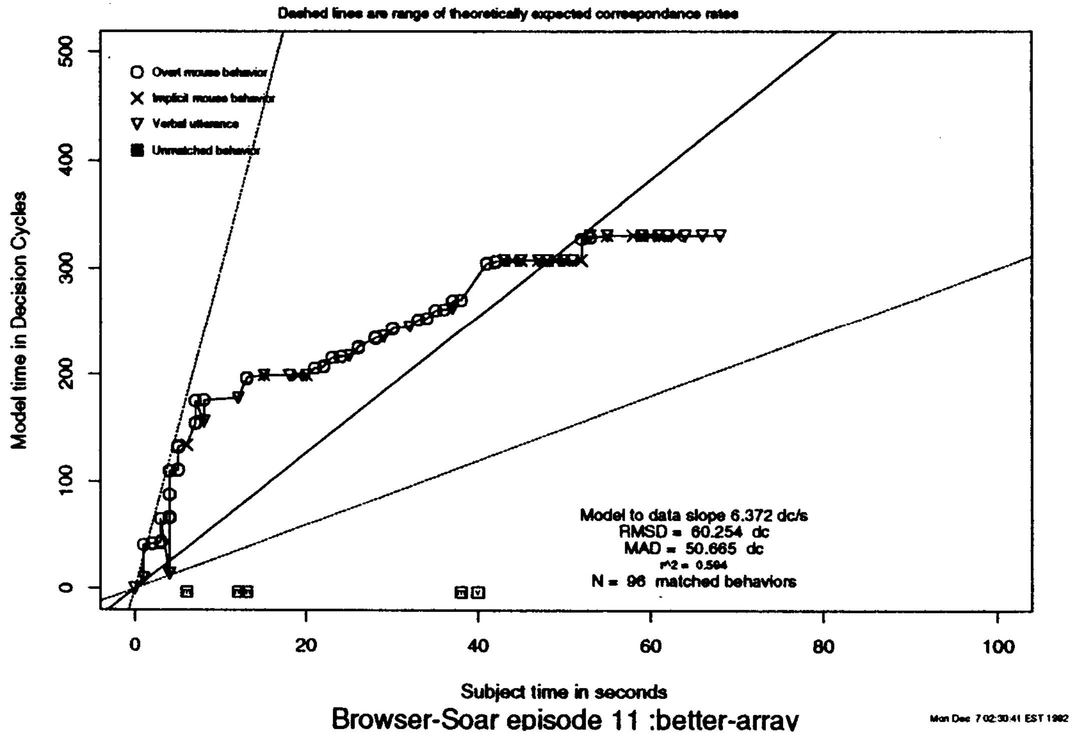
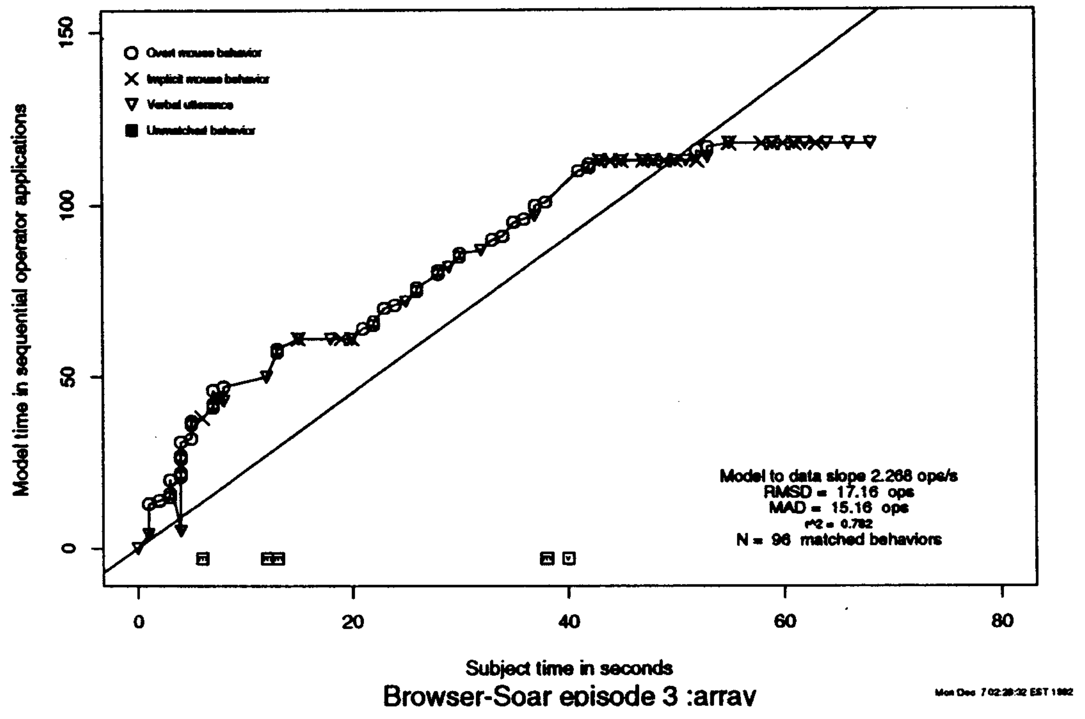


Figure 7-43: DC time based plots for the Array episode. The original Browser-Soar predictions are on the top, and the modified version on the bottom.



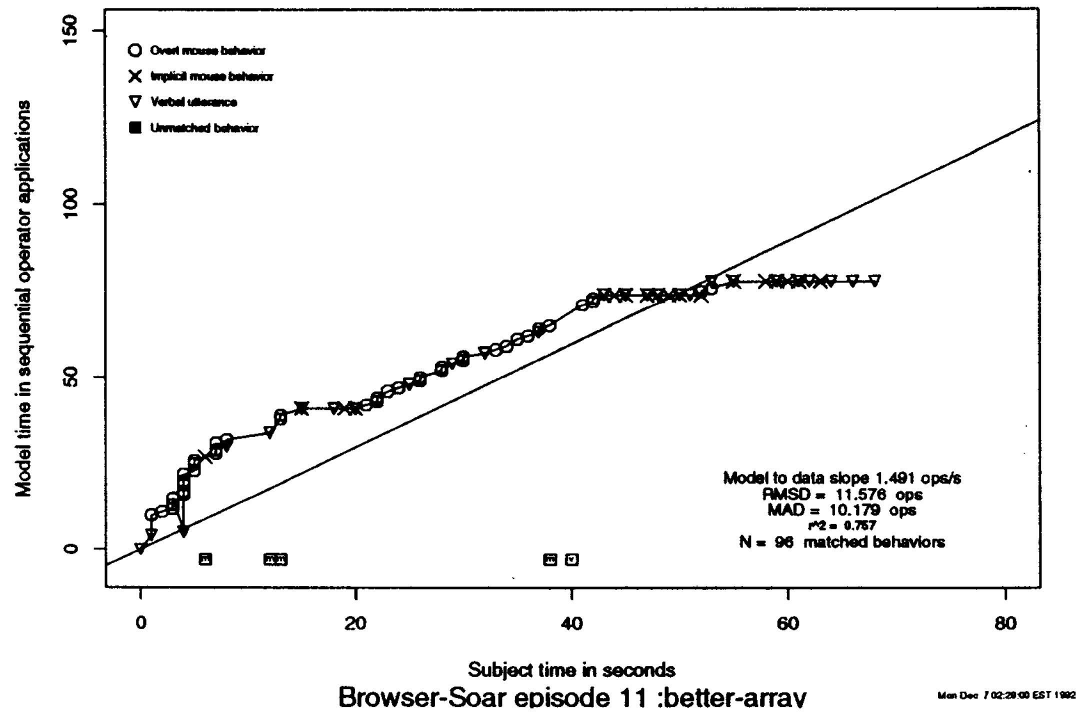


Figure 7-44: Relative processing rates displays based on operator applications for the Array episode. The original Browser-Soar predictions are on the top, and the modified version on the bottom.

7.5.1 Are verbalizations generated sequentially?

Of the 220 verbal utterances in the ten episodes, 195 can be aligned with the model's predictions. The remaining 25 are mostly too short to compare. The remaining segments make up 210 pairs of immediately sequential utterances that can be tested against the sequentiality assumption. This test can be performed by eye with the displays, and the initial analyses did this because it was so easy and direct. The final counts were taken from the data structure used to create the displays.

All 210 pairs follow the sequentiality assumption; for all the pairs, the later segment in each pair either matches the same model trace action as the first segment matches or a later model trace action. So this appears to be another constraint that Browser-Soar meets.

7.5.2 Are mouse actions generated sequentially?

In a similar way the mouse movements and mouse button actions can be tested for sequentially. Because these actions were used as fixed points to automatically align the subject's protocol and the model's trace, in order to match out of sequence they would had to have been moved by hand out of sequence, or items that could not be automatically aligned would have had to be aligned by hand.

Of the 404 mouse actions in the ten episodes, 373 can be aligned with the model's predictions.[9] These 373 actions make up 363 pairs of sequentially contiguous actions. Again, a preliminary examination of the displays showed that none matched the model out of order, and an analysis of the data base confirmed that.

7.5.3 Does the sequentiality assumption hold across verbalizations and mouse actions?

All the subject's actions can be tested for sequentiality. As explained in Chapter 5, this can be done by examining the connected correspondences in the relative processing rate displays. Starting from the first correspondence and moving along the line of correspondences, a connecting segment with a negative slope indicates that the second correspondence matched earlier in the model than the first correspondence, violating the sequentiality assumption. Simply examining the displays shows that several verbal utterances lag the mouse movements noticeably. Of the 624 total segments, 568 are aligned with the model's actions in the ten episodes.[10] These 568 actions make up 558 pairs of sequentially contiguous actions, and 21 pairs do not meet the sequentially assumption, that is, in these pairs, the second subject action is a verbal utterance that matches an earlier prediction than the first action that is a mouse action.

The lag of verbal utterances was computed by comparing the decision cycle number of the model prediction corresponding to the verbal utterance with the decision cycle of the previously matched mouse action. Figure 7-45 shows the distribution of these times. Across all verbal utterances in all episodes the average lag was 9 decision cycles, or roughly 1 second. This is an acceptable number, indicating that while some verbal utterances appear to have been produced quite late compared to the mouse movements, overall the subject was not providing retrospective reports.

Most of the verbal statements (144 out of 195) match the model's predictions sequentially, not matching earlier portions of the model than their proceeding segment. Based on their starting points these utterances can be considered as truly concurrent protocol — it is generated as the subject doing the task and it matched the predictions of the contents of working memory. The ends of the utterances have not been included in these analyses, although Peck and John included this length in their data set.

[9]An astute reader may note that there are five more mouse movements matched by subject actions in this analysis than in the original analysis reported by Peck and John. One of these discrepancies has been found so far, and it was a typo.

[10]An astute reader may again note that there are five more predictions matched by subject actions in this analysis than in the original by Peck and John. Even with a semi-automatic tool, analysts will make mistakes.

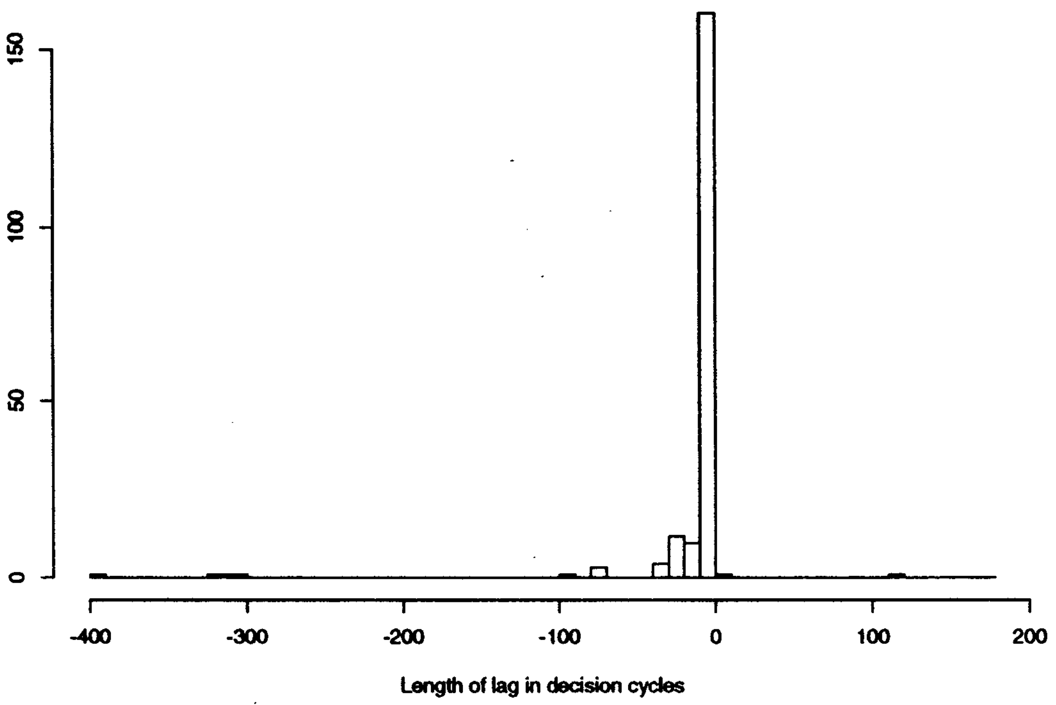


Figure 7-45: Histogram of the lags (in decision cycles) of the verbal utterances.

While these segments are not long generally, it is possible that their tail end ceases to be concurrent.

There are two prospective utterances, one in the axis episode, which upon inspection was an typo in alignment. The segment was properly concurrent, but misaligned by four decision cycles in the spreadsheet. The other utterance occurred in the Vars episode and is more interesting. It has a positive offset of 111 decision cycles (nominally 11 seconds). It is hard to see on the relative processing rate display because it is surrounded by several mouse movements, which is the cause of it being interpreted as early. When the segment is examined, it turns out that the verbal utterance is not so much prospective, but that the model's menu reading ability falls behind the subjects at that point, and the model has to perform an extra 100 cycles of work before it can match the verbal utterance.

The remaining 49 utterances all lag their previous segment, matching an earlier prediction. When an utterance lags, it lags on average 38 decision cycles, or roughly 4 seconds. Again this remains a modest amount. This amount of time is consistent with the amount of time items can exist in working memory. A very small number, three, lag over 300 decision cycles.

Characterizing the long lags Many short lags of the verbal utterances appear to be partly (but not completely) an artifact of the Browser-Soar model. The model does not read individual words but whole screens at a time, which leads to many of the short lags that occur late in an episode when the subject is reading a help text. Including predictions of reading individual words would remove this cause.

The three longest lags, however, are worrisome. They lag over three hundred decision cycles, and represent a mismatch on the order of 20 to 40 seconds. The problem space of the operator they match has long been removed from the goal stack, and several other problem spaces on that level have been used as well. When these segments are examined they are found to be statements of the search or evaluation criteria that occur after the search has started and numerous items have been examined. While an operator put them on the state, at the point they are uttered, they clearly represent state information that has been guiding the search for some time. Other operators could be refreshing them, but if that is what lead to these utterances, then the operator used to interpret them is still the wrong one.

Finding this lag in the literature. The actual lag of verbal protocols has not been computed in this way to my knowledge. It requires an architecture that makes predictions about the time to perform a task, external actions to provide fixed points of reference, and the predictions must be aligned to this detail. We can see a lag in other data sets, however. The verbal data used to develop HI-Soar (John & Vera, 1992; John, et al., 1990) can be fixed relative to the performance of external actions. The verbal protocols lagged behind the external actions so much that they were ignored when testing the model.

7.6 Conclusions about Browser-Soar and the TBPA methodology

Having performed these analyses, we can summarize the results into several suggested changes to Browser-Soar, which is the point of testing a process model. In general, Browser-Soar performed very well. The operators in the model that performed best were the ones that are essential to browsing on-line help systems: manipulating the mouse, choosing windows, and evaluating text items. On a higher level, testing Browser-Soar also generated some lessons for the methodology and for the environment that should be incorporated into the environment.

This methodology was stretched in a particular direction through testing Browser-Soar. Browser-Soar and the data used to test it have some very particular characteristics: (a) very close matches, (b) very routine behavior and typical problem solving by the subject, (c) a highly interactive task, (d) mostly a mental task (the perception and motor actions were routine). This example application did not deal with every type of data. It is easy to name several data features that have not been touched: (a) very bad matches between data and model, (b) perceptually based reasoning, (c) how to create a model in the first place, or drastically revise it, (d) tasks that cannot be modeled as search through or in problem spaces, and (e) extremely long or short protocols. Adding any of these features to the data and task is likely to add further lessons and stretch the methodology in a new way.

7.6.1 Some conclusions about Browser-Soar

The analyses performed suggest several ways to improve Browser-Soar. Most, if not all, are known to the authors of Browser-Soar, but the importance and location of the changes should be clearer after these analyses. These changes are presented in Table 7-32.

Browser-Soar's ability to predict large amounts of the data should also be clearer as well. Chapter 2 put forward the idea that analytic testing would not only point out where to improve a model, but it also would make it more believable by presenting it more clearly. Several diagrams and tables were created in performing these analyses that should make the model more believable. There are more visual descriptions of the model (Figure 7-33), its performance (Figure 7-35), a rough measure of the amount of knowledge in each problem space (Figure 7-36), and a picture of the calling order of its operators (Figure 7-38). Aggregated measures of which operators and problem spaces are used and how often have been presented (Table 7-29). The analytic displays show when operators are supported, and by which type of data (Figure 7-38 and the Appendix to this chapter), and the relative processing rates of the model and subject over time (Figures 7-39 and 7-40, and the appendix to this chapter).

Table 7-32: Suggested changes to Browser-Soar based on analyses performed.

- Operators without evidence, *Scroll*, *Page*, *Drag*, and *Click-on-item*, must be considered for removal from the model, or be supported with non-protocol data such as aggregate timing results.
- Fitt's law should be included in the model of moving the mouse.
- A more complete *Read* operator for reading text that takes longer.
- A less complete *Read* operator for reading menus faster, more like scanning.
- Overall, the model's performance is slightly lean, but this must be reevaluated after some other problems, most importantly the reading operator, have been improved.
- Include learning, and decrease the goal stack depth.
- Include state information in the trace and match to it.

7.6.2 Some conclusions about the methodology

Performing these analyses pointed out that it is nearly always good to have context, and sometimes it is required. Just providing information on a single item is often not enough to understand the item. The item's context is also needed. In several places, particularly in examining the model fit displays, users can now click on a data point and get a segment and a selectable amount of its context displayed.

Different grain sized operators and different commitments to operators lead to problems in the analysis, and should be avoided if possible. Soar in particular, as a general architecture for intelligence, provides the ability to model every action. As a unified theory of cognition it highlights the desire to provide a complete model, covering all the data. By definition, some portions of each model will be weaker than others.

Soar models are much finer in their grain size than Newell and Simon's (1972) systems; more actions occur that cannot be tested, such as goals and many problem spaces. Other items might be found, but are not found in every episode. It may be desirable to omit these items automatically and appropriately when performing an analysis.

While Newell and Simon (1972, p. 179) propose that states and operators are equivalent, the reanalysis of Browser-Soar shows that they are only equivalent for information purposes. When the timing of the correspondences is included, they are not equivalent. States, and the information they contain, last much longer. It may be possible to continue to match verbal utterances primarily to operators, but when this breaks down, one must match to the state. Using the state properly is not a trivial task, and will require designing and extending the trace. It will require further mechanations in the interpretation algorithms to find the appropriate items to support in the model when this does occur.

No problem spaces or goals are used to interpret the subject's behavior. Together their creation and selection make up a substantial portion of the model's behavior. What it would mean to match their prediction is not clear, problem spaces may be supported by their operators and states, goals by the indication of a lack of knowledge in some way. If they will not be directly supported, the cognitive modeler may desire to removal them from the trace if not the model.

Finally, we see that testing the model points out that the model is not complete without rules describing how to interpret the data with respect to the predictions. For example, the *Page*, *Scroll*, *Drag*, and *Click-on-item*, were considered for removal because they were not supported. A more generous interpretation of the mouse actions might have included the decision to click (e.g., to *Page*) as being supported as well.

Appendixes to Chapter 7

1 Alignment of the Write episode of Browser-Soar

```
Wed Nov 25 14:35:42 1992 - Dismal (0.83) report for user ritter
For file /afs/cs.cmu.edu/user/ritter/spa/browser/write/write5.spa

To print use  "enscript -r -c -0 -fCourier7 -L66  /afs/cs.cmu.edu/user/ritter/spa/browser/write/write5.dp"
------------------------------------------------------------

    A        B               C               D    E F  G   H  I                            J                                    K
 +--+---------------+----------------+-----------+-+-+-----+---+---+-----------------------------------------------+------------------------------------------+
 0 vapeck   25-Sep-91 revised 13-Jan-92 -FER            3-Jun-92                                                    To do:
 1 From original transcription by dmr, 16-Mar-90, and verbal transcription by s.esch,  Jun-91
 2 Transcription of the 15-Jun-89 of browser tape 2
 3
 4 Browsing for information about writing the values of variables to the screen.
 5 Previous Goal:     define the 'loop' construct that will label the x-axis
 6 Current Goal:      to figure out how to write the value of 'ExpCondNames' in
 7                    order to label the x-axis with experimental condition names
 8 Windows:           program window out front, right side and bottom of help win.,
 9                    only the left edge of the commands win., right side of execution window
10 Program Window     line 1  "***********"   line 2  "unit   DrawGraph"   line 3   "gorigin   110?, 340?"
11 Execution Window:  error box with message at the top
12 Help Text Window:  line 1  "coarse 'Typing-Paper' Coordinates"
13 Keyword Menu       line 1  "coarse" (selected)"
14 Hierarchial Menu:  line 1   "at   Positioning a Display"   (no lines selected)
15 Commands Window:   (not used)
16 Cursor:            positioned at the end of the 'put' command line within the 'loop' construct
17 Mouse:             located (+x) -3/4 in. from the end of the 'put' command line
18    (mouse is currently a line)
19                                    See also /afs/cs/project/soar/member/vapeck/browser-soar/current/episodes/write/01-22.write.complete.log
20 69 total behaviors                                 69
21 11 distinct operators evidenced in behavior                      Loaded from:
22                                                                  /afs/cs/project/soar/member/vapeck/browser-soar/current/LOAD-write.lisp
23 25 total verbals                                                 Loading /afs/cs.cmu.edu/user/ritter/spa/browser/op-trace-additions.lisp.
24
25
26  last time verbal/mouse matches information used by an operator.
27 21 total mouse movements                MTYPE is type of match:
28 9  unnecessary movements that give evidence            87   Total
29 6  necessary movements
30
31 24 Total mouse button actions                           4    B4 is actions before model coverage.
32                                                         2    SHORT is too short to code.
33 TIME is timestamp of action in s.                      14    CONT is segment continued from previous line.
34 DURATION is length in ms of behavior.
35 VERBAL is verbal protocol.                             22    V is verbal coded.
36 Mouse Action is the user's mouse movements.             6    MR is mouse required (these are just movements).
37 ST is Segment Type                                     11    MI is mouse inferred.
38 # is segment number                                    24    MBA is mouse button actions (necessary by default).
39 MTYPE is type of match
40 MDC is matched DC.                                      3    MUC is mouse uncoded.
41 DC is decision cycle in Soar model.                     1    VUC is verbal Un Coded.
42 SOAR TRACE is the literal Soar Trace
43
44    0.94              percent subject data matched
45    0.09              percent model matched
46    0.10              seconds/decision cycle
47
48                      total words              113
49
50 Time starts at 12400
51  T  Mouse actions   Window actions    Verbal     ST #  Mtype MDC  DC                    Soar trace                   Comments
52 0                                     I believe   v  1  short
53                                                                  0 G: g1                                             From:  raw-trace3.txt, 17-Mar-92
54                                                                  1 P: p4 (top-space                                  Operator names cleaned up by hand,
55                                                                  2 S: s5                                             6-july-92 FER
56                                                                  3 O: browse ()
57                                                                  4 ==>G: g19 (operator no-change
58                                                                  5   P: p26 (browsing
59                                                                  6   S: s39 ((unknown) (unknown)
60                                                                  7   O: find-appropriate-help
61                                                                  8   ==>G: g43 (operator no-change
62                                                                  9     P: p50 (find-appropriate-help
63                                                                 10     S: s59 ((unknown) (unknown)
64                                                                 11     O: define-search-criterion
65                                                                 12     ==>G: g65 (operator no-change
66                                                                 13       P: p72 (define-search-criterion
67                                                                 14       S: s79 ((unknown)
68 6                                     write       v  2     v  15  15       O: generate-search-criterion ((write))
69 9                                     write       v  3     v  15
70 13                                    write       v  4     v  15
71    M(+x) (R of prog win)                                  B4                                                     B-Soar doesn't model bringingup the
72                 mouse line to pointer                                                                                     help win.
73                                                                 16       O: evaluate-search-criterion            *** generated search criterion 'write' **
74                                                                 17     O: define-evaluation-criterion
75                                                                 18     ==>G: g103 (operator no-change
76                                                                 19       P: p110 (define-evaluation-criterion
77                                                                 20       S: s117 ((unknown)
78 14                                    Can I write v  5     v  21  21       O: generate-evaluation-criterion ((value-of-something))
79 15 M(+y) (top of screen)                              B4                                                         unmodeled (before the task) mouse movement
80 15 M(-x-y) (portion of help win below prog win)       B4
81 16 C            help win comes forward                B4
```

```
S1  T  Mouse actions   Window actions     Verbal     ST #  Mtype MDC  DC                  Soar trace                          Comments
   +--+----------------+-------------------+-----------+-+---+-----+---+---+------------------------------------------------+------------------------------------------+
82  16 M(+x+y) (R of 'coarse' at top of keyword menu)mm  6    mus
83     --(+x-y) (just L of keyword down arrow)                cont
84                                                                     22      O: evaluate-evaluation-criterion               *** generated evaluation criterion
85                                                                     23    O: search-for-help                                   'value-of-something' **
86                                                                     24    =>G: g137 (operator no-change
87                                                                     25      P: p144 (search-for-help
88                                                                     26      S: s155 ((to-be-found write) (value-of-something)
89                                                                     27      O: find-criterion (keyword)
90                                                                     28      =>G: g161 (operator no-change
91                                                                     29        P: p160 (find-criterion
92                                                                     30        S: s170 ((to-be-found write) (value-of-something)
93                                                                     31        O: focus-on-current-window
94                                                                     32        O: evaluate-current-window
95                                                                     33        =>G: g239 (operator no-change
96                                                                     34          P: p244 (evaluate-items-in-window
97                                                                     35          S: s256 ((to-be-found write) (value-of-something)
-98                                                                    36          O: read-input (coarse)
99                                                                     37          O: attempt-match
100                                                                    38          O: read-input (comment)
101                                                                    39          O: attempt-match
102                                                                    40          O: read-input (comp_x)
103                                                                    41          O: attempt-match
104                                                                    42          O: read-input (compute)
105                                                                    43          O: attempt-match
106                                                                    44          O: read-input (constant)
107                                                                    45          O: attempt-match
108                                                                    46          O: read-input (cos_x)
109                                                                    47          O: attempt-match
110                                                                    48          O: read-input (cursor)
111                                                                    49          O: attempt-match
112                                                                    50          O: read-input (detain)
113                                                                    51          O: attempt-match
114                                                                    52        O: change-current-window
115                                                                    53        =>G: g423 (operator no-change
116                                                                    54          P: p430 (mac-methods-for-change-current-window
117                                                                    55          S: s438 ((to-be-found write)
118                                                                    56          O: scroll (keyword)
119                                                                    57          =>G: g451 (operator no-change
120                                                                    58            P: p450 (mac-method-of-scroll
121                                                                    59            S: s467 ((to-be-found write)
122 17 M(+x) to (keyword dn arrow)                mm  7    mr   60  60            O: move-mouse (keyword down)
123·17 D                keyword menu scrolls       mb  8    mba  61  61            O: press-button
124 18                 keyword menu scrolls
125 19                 keyword menu scrolls
126 20                 keyword menu scrolls
127 21                                      write   v  9    v   32
128 22 U               scrolling stops            mb 10    mba  62  62            O: release-button
129 22 M(-x+y) (R of item2, keywd menu:wrong (write imm 11  mi   63  63        O: evaluate-current-window
130 23                                      wrong?  v 12    v   63
131                                                                    64        =>G: g507 (operator no-change
132                                                                    65          P: p514 (evaluate-items-in-window
133                                                                    66          S: s524 ((to-be-found write) (value-of-something)
·134                                                                   67          O: read-input (wrong)
135                                                                    68          O: attempt-match
136                                                                    69          O: read-input (wrongv)
137                                                                    70          O: attempt-match
138                                                                    71          O: read-input (xin)
139                                                                    72          O: attempt-match                        We are left matching operators
140                                                                    73          O: read-input (xout)                    for we have not states.
141                                                                    74          O: attempt-match
142                                                                    75          O: read-input (zaltered)
143                                                                    76          O: attempt-match
144                                                                    77          O: read-input (zanscnt)
145                                                                    78          O: attempt-match
146                                                                    79          O: read-input (zarrown)
147                                                                    80          O: attempt-match
148                                                                    81          O: read-input (temptemptemp)
149                                                                    82          O: attempt-match
150                                                                    83        O: change-current-window
151                                                                    84        =>G: g696 (operator no-change
152                                                                    85          P: p703 (mac-methods-for-change-current-window
153                                                                    86          S: s711 ((to-be-found write)
154                                                                    87          O: scroll (keyword)
155                                                                    88          =>G: g724 (operator no-change
156                                                                    89            P: p731 (mac-method-of-scroll
157                                                                    90            S: s740 ((to-be-found write)
158 23 M(+x-y) (keyword up arrow)                 mm 13    mr   91  91            O: move-mouse (keyword up)
159 23 D               keyword menu scrolls       mb 14    mba  92  92            O: press-button
160 23 U               scrolling stops            mb 15    mba  93  93            O: release-button
161 24                                      no      v 16    v   63
162 24 M(-x-y) (2nd keyword from bottom, xin)     mm 17    mi   94  94        O: evaluate-current-window
163 24                                      Ha  ha haaa v 18 vuc
164 25                                      write.  v 19    v   94
165                                                                    95        =>G: g777 (operator no-change
166                                                                    96          P: p784 (evaluate-items-in-window
167                                                                    97          S: s794 ((to-be-found write) (value-of-something)
168                                                                    98          O: read-input (user-vars)
169                                                                    99          O: attempt-match
170                                                                   100          O: read-input (vbar)
171                                                                   101          O: attempt-match
172                                                                   102          O: read-input (vector)
173                                                                   103          O: attempt-match
174                                                                   104          O: read-input (write)
175                                                                   105          O: attempt-match
176                                                                   106      O: access-item (keyword)
177                                                                   107      =>G: g870 (operator no-change
178                                                                   108        P: p885 (mac-methods-for-access-item
179                                                                   109        S: s893
180                                                                   110        O: click-on-item (1000)                   1000 is an unnamed item
181                                                                   111        =>G: g899 (operator no-change
182                                                                   112          P: p906 (mac-method-of-click-on-item
183                                                                   113          S: s913
184 25 M(+y) (3 items up to 'write')              mm 20    mr  114 114          O: move-mouse (keyword unspecified)
```

```
 SI  T  Mouse actions   Window actions     Verbal      OT #  Mtype MDC  DC                          Soar trace                                  Comments
   +--+-----------------+----------------------+-----------+-+--+-----+---+---+------------------------------------------------------------+---------------------------------------------+
185 25 C                mouse pointer to watch        mb 21   mba 115 115             O: click-button
186 26                  'write' help text appears     io
187 27                  'write' becomes bold & moves  io
188                                                                          116       O: evaluate-help-text
189                                                                          117       =>G: g927 (operator no-change
190                                                                          118         P: p934 (evaluate-help-text
191                                                                          119         S: s943 ((assessed write) (value-of-something)
192                                                                          120         O: focus-on-help-text
193 28                                   convenient way to v 22       v 121 121         O: evaluate-current-window
194 29                                         write out short      cont
195 30                                         of text that loo     cont
196 31                                         in your program      cont
197 32                                         the text command     cont
198 32 M(-x-y) (3/4 dn help text scrollbar below elevmm 23         mi 121
199 33                                         umm...        v 24 short
200 33 M(-x-y) (bottom R quad of help text win)        mm 25         mi 121
201 36                                         show common v 26       v 121
202 38                                         are used    v 27       v 121
203 39                                         to display  v 28       v 121
204 41                                         so thats what I      cont
205                                                                          122         =>G: g966 (operator no-change
206                                                                          123           P: p973 (evaluate-prose-in-window
207                                                                          124           S: s984 ((assessed write) (value-of-something)
208                                                                          125           O: read-input
209                                                                          126           O: comprehend
210                                                                          127           O: compare-to-criteria
211 42 M(+x+y) (mid of keywd scrollbar, over elev)     mm 29        eme
212 43                                         is show com v 30       v 128 128       O: change-search-criterion ((assessed write))       *** changed search criterion 'write' **
213                                                                                                                                       *** changed search criterion 'show' **
214                                                                          129       O: search-for-help
215                                                                          130       =>G: g1025 (operator no-change
216                                                                          131         P: p1032 (search-for-help
217                                                                          132         S: s1043 ((to-be-found show) (value-of-something)
218                                                                          133         O: find-criterion (keyword)
219                                                                          134         =>G: g1049 (operator no-change
220                                                                          135           P: p1056 (find-criterion
221                                                                          136           S: s1066 ((to-be-found show) (value-of-something)
222                                                                          137           O: focus-on-current-window
223 43 M(-x-y) (-1/2 in R of keyword 'zanscnt')        mm 31         mi 138 138           O: evaluate-current-window                      goes by but doesn't stop on zanscnt,
224 43 -- (+x+y) (keyword scroll bar, above elevator)              cont
225 43 -- (+y) (above keyword up arrow)                             cont
226                                                                          139           =>G: g1092 (operator no-change                      micro-codable as:
227                                                                          140             P: p1099 (evaluate-items-in-window                162 O: read-input (zanscnt)
228                                                                          141             S: s1109 ((to-be-found show) (value-of-something)
229                                                                          142             O: read-input (write)
230                                                                          143             O: attempt-match
231                                                                          144             O: read-input (wrong)
232                                                                          145             O: attempt-match
233                                                                          146             O: read-input (wrongv)
234                                                                          147             O: attempt-match
235                                                                          148             O: read-input (xin)
236                                                                          149             O: attempt-match
237                                                                          150             O: read-input (xout)
238                                                                          151             O: attempt-match
239                                                                          152             O: read-input (zaltered)
240                                                                          153             O: attempt-match
241                                                                          154             O: read-input (zanscnt)
242                                                                          155             O: attempt-match
243                                                                          156             O: read-input (zarrown)
244                                                                          157             O: attempt-match
245                                                                          158           O: change-current-window
246                                                                          159           =>G: g1276 (operator no-change
247                                                                          160             P: p1283 (mac-methods-for-change-current-window
248                                                                          161             S: s1291 ((to-be-found show)
249                                                                          162             O: scroll (keyword)
250                                                                          163             =>G: g1304 (operator no-change
251                                                                          164               P: p1311 (mac-method-of-scroll
252                                                                          165               S: s1320 ((to-be-found show)
253 44 M(-y) (keyword up arrow)                        mm 32         mr 166 166               O: move-mouse) (keyword up)
254 44                                         so let's ju v 33       v 128
255 44 D                                                  mb 34        mba 167 167               O: press-button
256                      keyword menu scrolls & stops
257 44 U                                                  mb 35        mba 168 168               O: release-button
258                                                                          169           O: evaluate-current-window
259                                                                          170           =>G: g1350 (operator no-change
260                                                                          171             P: p1365 (evaluate-items-in-window
261                                                                          172             S: s1375 ((to-be-found show) (value-of-something)
262                                                                          173             O: read-input (use)
263                                                                          174             O: attempt-match
264                                                                          175             O: read-input (user-vars)
265                                                                          176             O: attempt-match
266                                                                          177             O: read-input (vbar)
267                                                                          178             O: attempt-match
268                                                                          179             O: read-input (vector)
269                                                                          180             O: attempt-match
270                                                                          181             O: read-input (write)
271                                                                          182             O: attempt-match
272                                                                          183             O: read-input (wrong)
273                                                                          184             O: attempt-match
274                                                                          185             O: read-input (wrongv)
275                                                                          186             O: attempt-match
276                                                                          187             O: read-input (xin)
277                                                                          188             O: attempt-match
278                                                                          189           O: change-current-window
279                                                                          190           =>G: g1547 (operator no-change
280                                                                          191             P: p1554 (mac-methods-for-change-current-window
281                                                                          192             S: s1562 ((to-be-found show)
282                                                                          193             O: scroll (keyword)
283                                                                          194             =>G: g1575 (operator no-change
284                                                                          195               P: p1582 (mac-method-of-scroll
285                                                                          196               S: s1591 ((to-be-found show)
286 44 D                                                  mb 36        mba 197 197               O: press-button
287                      keyword menu scrolls & stops   io
```

```
S1  T  Mouse actions    Window actions          Verbal       ST #  Mtype MDC DC            Soar trace                         Comments
   +--+---------------+-----------------------+------------+-+---+-----+---+---+------------------------------------------+------------------------------------------+
288 44 U                                                     mb 37  mba 198 198      O: release-button
289 45                                     sure I know how         cont                                                        Continued from what matched dc 120
290                                                                         199  O: evaluate-current-window
291                                                                         200  =>G: g1622 (operator no-change
292                                                                         201    P: p1629 (evaluate-items-in-window
293                                                                         202    S: s1639 ((to-be-found show) (value-of-something)
294                                                                         203    O: read-input (ten)
295                                                                         204    O: attempt-match
296                                                                         205    O: read-input (text)
297                                                                         206    O: attempt-match
298                                                                         207    O: read-input (touch)
299                                                                         208    O: attempt-match
300                                                                         209    O: read-input (unit)
301                                                                         210    O: attempt-match
302                                                                         211    O: read-input (use)
303                                                                         212    O: attempt-match
304                                                                         213    O: read-input (user-vars)
305                                                                         214    O: attempt-match
306                                                                         215    O: read-input (vbar)
307                                                                         216    O: attempt-match
308                                                                         217    O: read-input (vector)
309                                                                         218    O: attempt-match
310                                                                         219  O: change-current-window
311                                                                         220  =>G: g1011 (operator no-change
312                                                                         221    P: p1018 (mac-methods-for-change-current-window
313                                                                         222    S: s1026 ((to-be-found show)
314                                                                         223    O: scroll (keyword)
315                                                                         224    =>G: g1039 (operator no-change
316                                                                         225      P: p1046 (mac-method-of-scroll
317                                                                         226      S: s1055 ((to-be-found show)
318 45 D                                                     mb 38  mba 227 227      O: press-button                       these scrolls, all within 1 s in the human,
319                     keyword menu scrolls & stops                                                                           don't correspond to this novice like model.
320 45 U                                                     mb 39  mba 228 228      O: release-button                     -- some of this will chunk up in the human.
321                                                                         229  O: evaluate-current-window
322                                                                         230  =>G: g1086 (operator no-change
323                                                                         231    P: p1093 (evaluate-items-in-window
324                                                                         232    S: s1903 ((to-be-found show) (value-of-something)
325                                                                         233    O: read-input (string)
326                                                                         234    O: attempt-match
327                                                                         235    O: read-input (syntaxlevel)
328                                                                         236    O: attempt-match
329                                                                         237    O: read-input (ten)
330                                                                         238    O: attempt-match
331                                                                         239    O: read-input (text)
332                                                                         240    O: attempt-match
333                                                                         241    O: read-input (touch)
334                                                                         242    O: attempt-match
335                                                                         243    O: read-input (unit)
336                                                                         244    O: attempt-match
337                                                                         245    O: read-input (use)
338                                                                         246    O: attempt-match
339                                                                         247    O: read-input (user-vars)
340                                                                         248    O: attempt-match
341                                                                         249  O: change-current-window
342                                                                         250  =>G: g2075 (operator no-change
343                                                                         251    P: p2082 (mac-methods-for-change-current-window
344                                                                         252    S: s2090 ((to-be-found show)
345                                                                         253    O: scroll (keyword)
346                                                                         254    =>G: g2103 (operator no-change
347                                                                         255      P: p2110 (mac-method-of-scroll
348                                                                         256      S: s2119 ((to-be-found show)
349 46 D                                                     mb 40  mba 257 257      O: press-button
350                     keyword menu scrolls & stops         io
351 46 U                                                     mb 41  mba 258 258      O: release-button
352 47 M(-x-y) ('showh' 2nd f/ top of list)                  mm 42   mi 259 259  O: evaluate-current-window
353 47 M(+x+y) (-1/4in R of 'showb', the 1st item)           mm 43   mi 259
354 48 M(+x) (just R & below keyword up arrow)               mm 44   mi 259                                                    partial move to get ready to scroll
355 48 -- (+x) (up arrow)                                    mm    cont                                                        again. Fits law!
356 49                                         show B show v 45     v 259
357                                                                         260  =>G: g2150 (operator no-change
358                                                                         261    P: p2157 (evaluate-items-in-window
359                                                                         262    S: s2167 ((to-be-found show) (value-of-something)
360                                                                         263    O: read-input (showb)                   This is a patched in trace
361                                                                         264    O: attempt-match                        from dc 263 to 277
362                                                                         265    O: read-input (showh)
363                                                                         266    O: attempt-match
364                                                                         267    O: read-input (showo)
365                                                                         268    O: attempt-match
366                                                                         269    O: read-input (showt)
367                                                                         270    O: attempt-match
368                                                                         271    O: read-input (sign)
369                                                                         272    O: attempt-match
370                                                                         273    O: read-input (sin)
371                                                                         274    O: attempt-match
372                                                                         275    O: read-input (sinh)
373                                                                         276    O: attempt-match
374                                                                         277    O: read-input (size)
375                                                                         278    O: attempt-match
376                                                                         279  O: change-current-window
377                                                                         280  =>G: g2339 (operator no-change
378                                                                         281    P: p2346 (mac-methods-for-change-current-window
379                                                                         282    S: s2354 ((to-be-found show)
380                                                                         283    O: scroll (keyword)
381                                                                         284    =>G: g2367 (operator no-change
382                                                                         285      P: p2374 (mac-method-of-scroll
383                                                                         286      S: s2383 ((to-be-found show)
384 51 D                                                     mb 46  mba 287 287      O: press-button
385 51 U                                                     mb 47  mba 288 288      O: release-button
386                                                                         289  O: evaluate-current-window
387                                                                         290  =>G: g2415 (operator no-change
388                                                                         291    P: p2422 (evaluate-items-in-window
389                                                                         292    S: s2432 ((to-be-found show) (value-of-something)
390                                                                         293    O: read-input (scalex)
```

```
S1  T  Mouse actions   Window actions     Verbal     ST #  Mtype MDC DC                       Soar trace                      Comments
   +--+---------------+------------------+----------+-+---+-----+---+-+--------------------------------------------------------+--------------------------------------------+
391                                                                    294          O: attempt-match
392                                                                    295          O: read-input (sealey)
393                                                                    296          O: attempt-match
394                                                                    297          O: read-input (set)
395                                                                    298          O: attempt-match
396                                                                    299          O: read-input (setfile)
397                                                                    300          O: attempt-match
398                                                                    301          O: read-input (show)
399                                                                    302          O: attempt-match
400                                                                    303      O: access-item (keyword)
401                                                                    304      =>G: g2527 (operator no-change
402                                                                    305        P: p2534 (mac-methods-for-access-item
403                                                                    306        S: s2542
404                                                                    307        O: click-on-item (i2537)
405                                                                    308        =>G: g2548 (operator no-change
406                                                                    309          P: p2555 (mac-method-of-click-on-item
407                                                                    310          S: s2562
408 52 M(-x-y) ('showb', 3rd from bot, keywrd menu)  mm 48      mr 311 311          O: move-mouse (keyword unspecified)
409    -- (+y) ('show')                                       cont
410 52 C                                             mb 49     mba 312 312          O: click-button
411                                                                    313    O: evaluate-help-text
412                                                                    314    =>G: g2576 (operator no-change
413                                                                    315      P: p2583 (evaluate-help-text
414                                                                    316      S: s2592 ((accessed show) (value-of-something)
415                                                                    317      O: focus-on-help-text
416 53                                    I don't kno v 50       v 318 318      O: evaluate-current-window
417 54                                    show binary, pro    cont
418 55                                    show expression.    cont
419 56                                    is an infinite f    cont
420 56 M(-y) (middle R side of help text)            mm 51      mi 318
421 57 M(-y) (a little lower)                        mm 52      mi 318
422 58                                    uhm...      v 53 short
423 58 M(-y) (a little lower)                        mm 54      mi 318
424 60                                    but I wonde v 55       v 318
425 62 M(+x-y) (just L of dn arrow for help txt win) mm 56     mme
426 63                                    for         v 57       v 318
427                                                                    319      =>G: g2615 (operator no-change
428                                                                    320        P: p2622 (evaluate-prose-in-window
429                                                                    321        S: s2633 ((accessed show) (value-of-something)
430                                                                    322        O: read-input
431                                                                    323        O: comprehend
432                                                                    324        O: compare-to-criteria
433                                                                    325      O: change-current-window
434                                                                    326      =>G: g2658 (operator no-change
435                                                                    327        P: p2665 (mac-methods-for-change-current-window
436                                                                    328        S: s2673 ((accessed show)
437                                                                    329        O: scroll (help-text)
438                                                                    330        =>G: g2685 (operator no-change
439                                                                    331          P: p2692 (mac-method-of-scroll
440                                                                    332          S: s2701 ((accessed show)
441 63 M(+x) (down arrow)                            mm 58      mr 333 333          O: move-mouse (help-text down)
442 64 D               help text win. scrolls        mb 59     mba 334 334          O: press-button
443 65                                    markers     v 60       v 318
444 65 U                                             mb 61     mba 335 335          O: release-button
445                                                                    336      O: evaluate-current-window
446                                                                    337      =>G: g2744 (operator no-change
447                                                                    338        P: p2751 (evaluate-prose-in-window
448                                                                    339        S: s2762 ((accessed show) (value-of-something)
449                                                                    340        O: read-input
450                                                                    341        O: comprehend
451                                                                    342        O: compare-to-criteria
452                                                                    343      O: change-current-window
453                                                                    344      =>G: g2787 (operator no-change
454                                                                    345        P: p2794 (mac-methods-for-change-current-window
455                                                                    346        S: s2802 ((accessed show)
456                                                                    347        O: scroll (help-text)
457                                                                    348        =>G: g2814 (operator no-change
458                                                                    349          P: p2821 (mac-method-of-scroll
459                                                                    350          S: s2830 ((accessed show)
460 66 D               help text win. scrolls        mb 62     mba 351 351          O: press-button
461 67 U                                             mb 63     mba 352 352          O: release-button
462 68                                    okay        v 64       v 336                                                          Gin - This had been 332, a state?
463                                                                    353      O: evaluate-current-window                      30-jun-92 FER
464                                                                    354      =>G: g2864 (operator no-change
465                                                                    355        P: p2871 (evaluate-prose-in-window
466                                                                    356        S: s2882 ((accessed show) (value-of-something)
467                                                                    357        O: read-input
468                                                                    358        O: comprehend
469                                                                    359        O: compare-to-criteria
470                                                                    360      O: change-current-window
471                                                                    361      =>G: g2907 (operator no-change
472                                                                    362        P: p2914 (mac-methods-for-change-current-window
473                                                                    363        S: s2922 ((accessed show)
474                                                                    364        O: scroll (help-text)
475                                                                    365        =>G: g2934 (operator no-change
476                                                                    366          P: p2941 (mac-method-of-scroll
477                                                                    367          S: s2950 ((accessed show)
478 68 D               help text win. scrolls        mb 65     mba 368 368          O: press-button
479 69 U                                             mb 66     mba 369 369          O: release-button
480                                                                    370      O: evaluate-current-window
481                                                                    371      =>G: g2984 (operator no-change
482                                                                    372        P: p2991 (evaluate-prose-in-window
483                                                                    373        S: s3002 ((accessed show) (value-of-something)
484                                                                    374        O: read-input
485                                                                    375        O: comprehend
486                                                                    376        O: compare-to-criteria
487                                                                    377      O: change-current-window
488                                                                    378      =>G: g3027 (operator no-change
489                                                                    379        P: p3034 (mac-methods-for-change-current-window
490                                                                    380        S: s3042 ((accessed show)
491                                                                    381        O: scroll (help-text)
492                                                                    382        =>G: g3054 (operator no-change
493                                                                    383          P: p3061 (mac-method-of-scroll
 SI  T  Mouse actions   Window actions     Verbal      ST #  Mtype MDC  DC                  Soar trace                      Comments
   +---+----------------+-------------------+-----------+-+---+-----+---+---+-----------------------------------------------+------------------------------------------+
494                                                                     384          S: s3070 ((accessed show)
495 70 D               help text win. scrolls          mb 67   mba 385 385          O: press-button
496 72                                     well, I'll  v 68     v 370
497 72 U                                               mb 69   mba 386 386          O: release-button
498                                                                     387       O: evaluate-current-window
499                                                                     388       =>G: g3104 (operator no-change
500                                                                     389         P: p3111 (evaluate-press-in-window
501                                                                     390         S: s3122 ((accessed show) (value-of-something)
502                                                                     391         O: read-input
503                                                                     392         O: comprehend
504                                                                     393         O: compare-to-criteria
505                                                                     394   =>G: state no-change
506                                                                     395     =>G: g3152 (goal no-change
507                                                                     396       =>G: g3159 (goal no-change
508                                                                     397         =>G: g3166 (goal no-change
509                                                                     398           =>G: g3173 (goal no-change
510                                                                     399             =>G: g3180 (goal no-change
```

2 Displays of each analytical measure for each episode of Browser-Soar

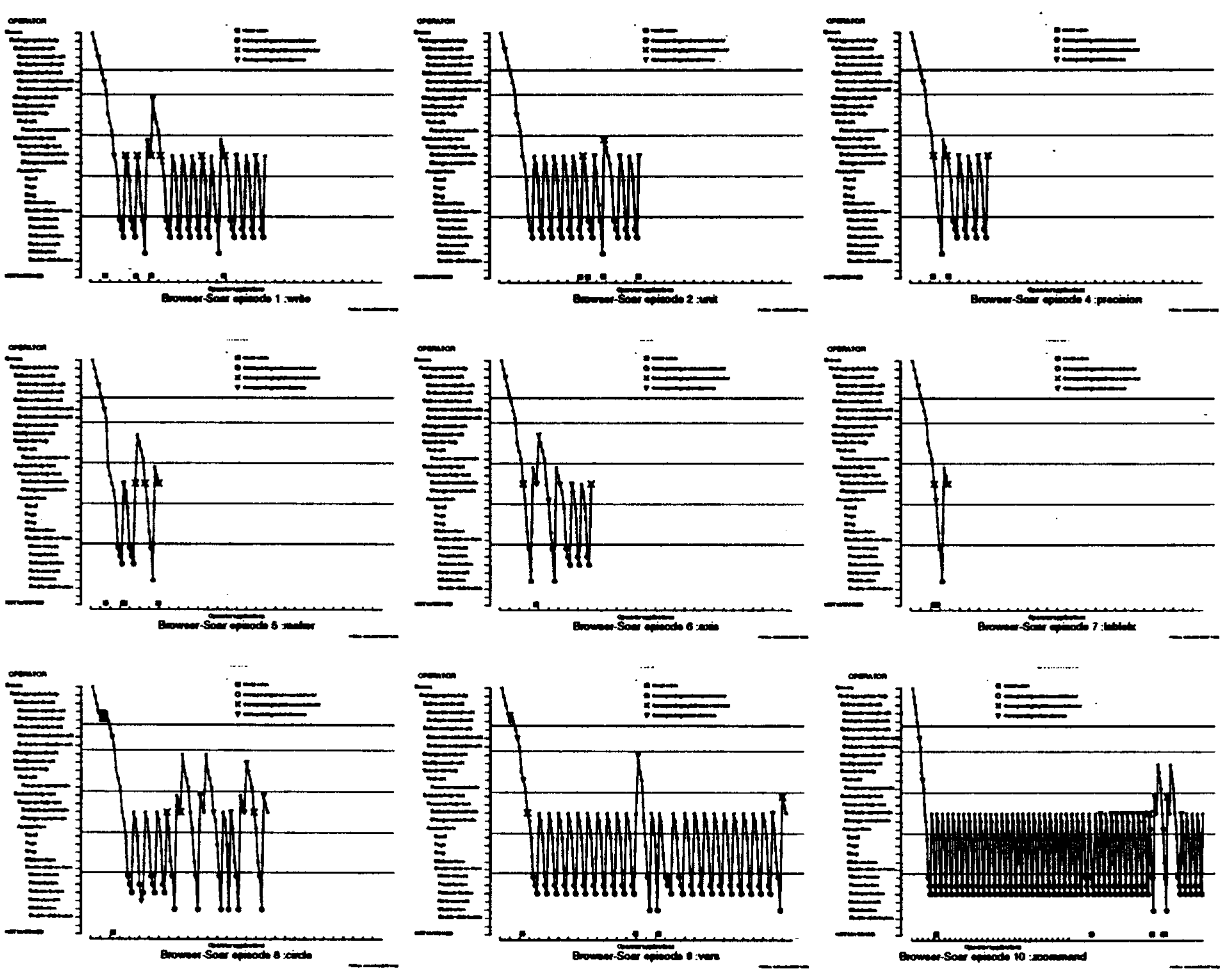
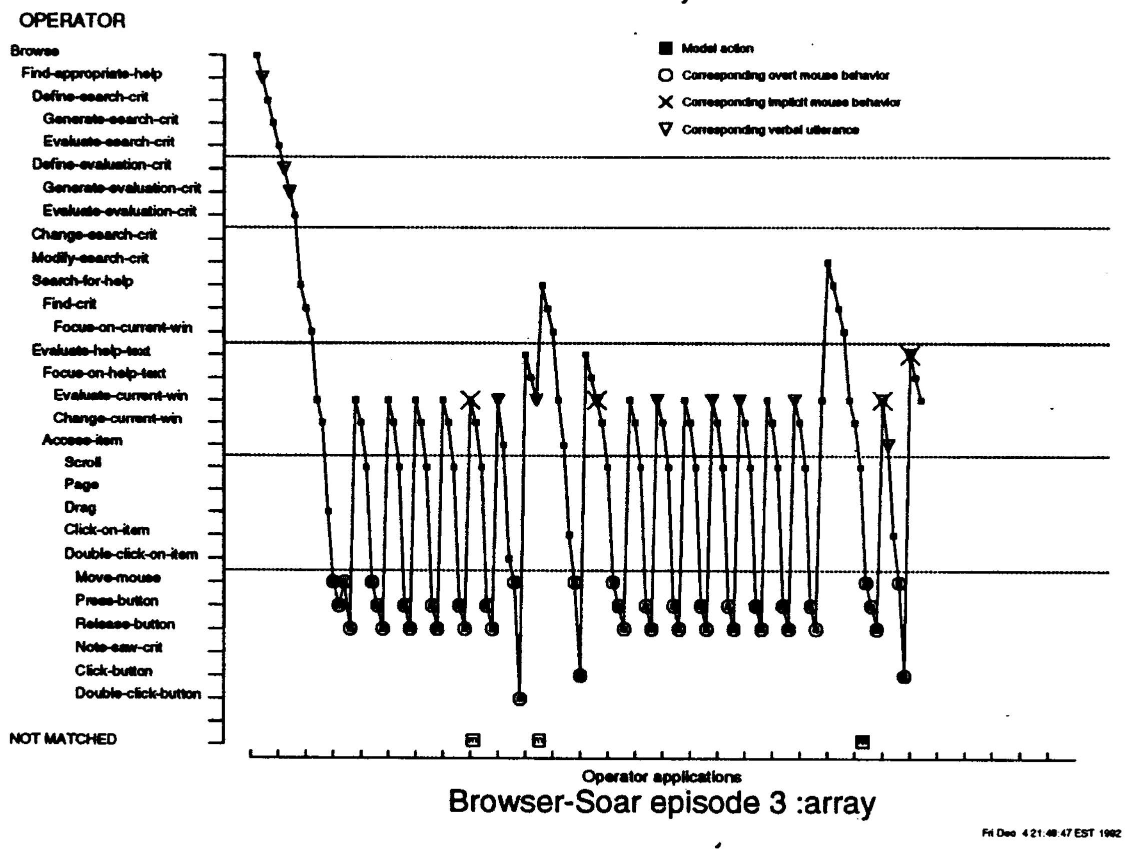


Figure 46: The operator support displays for each of the episodes.



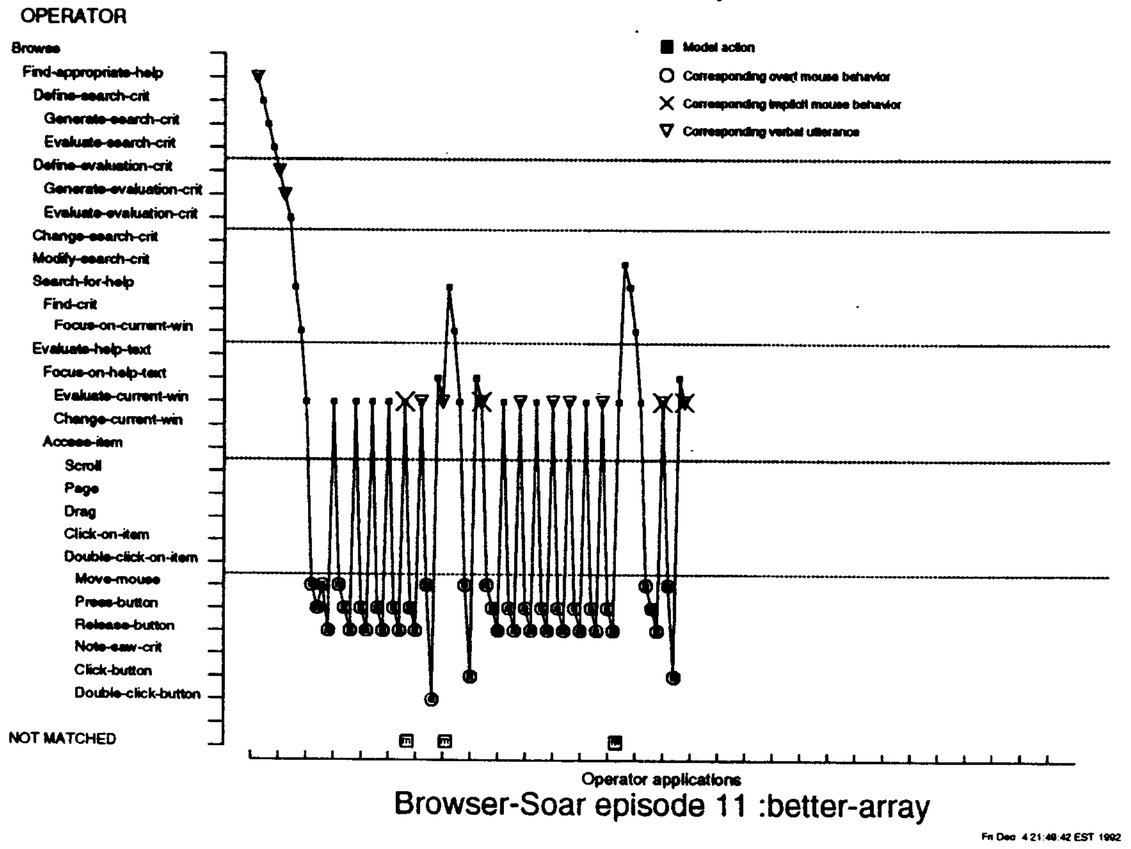


Figure 46: The operator support displays for each of the episodes (cont.).

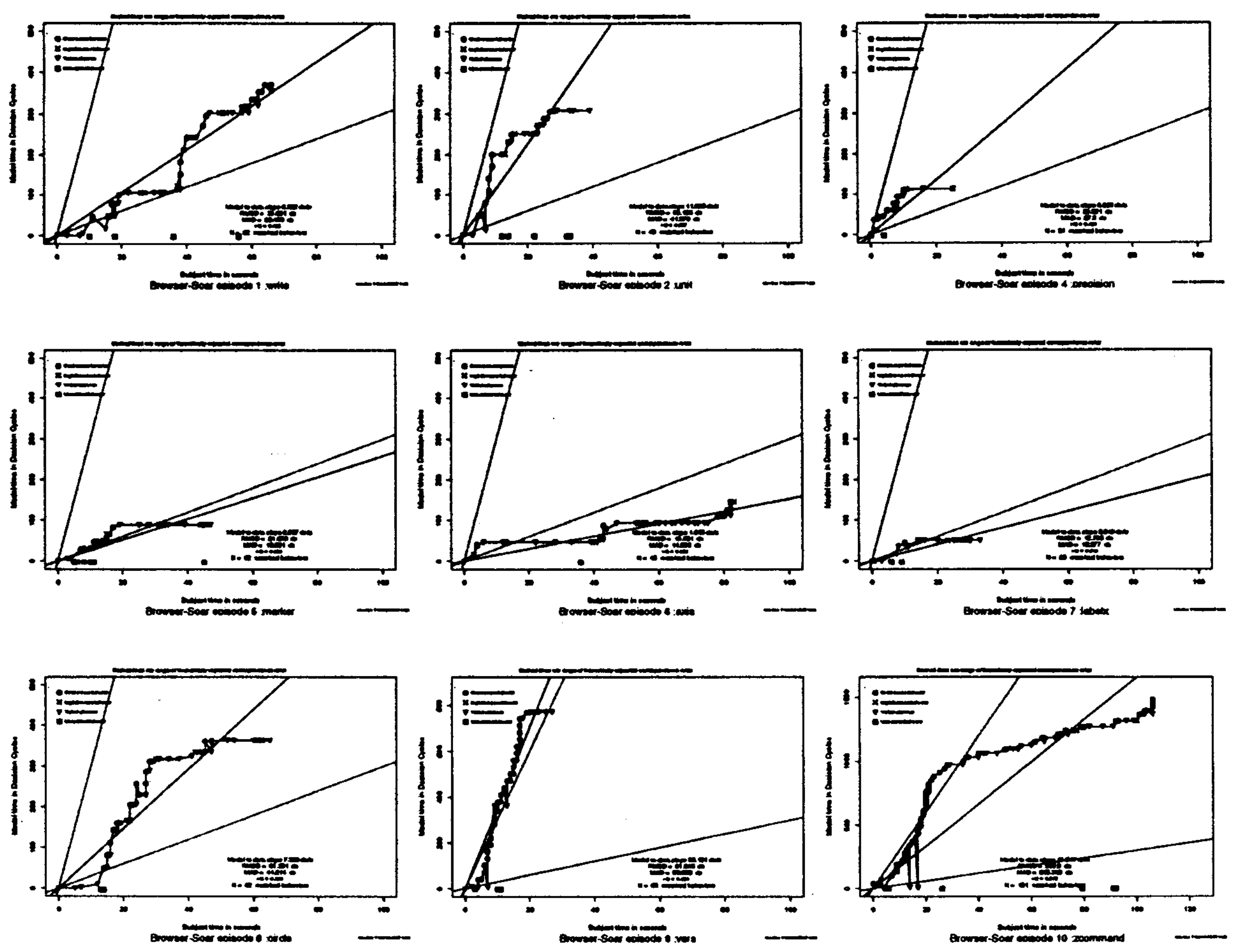
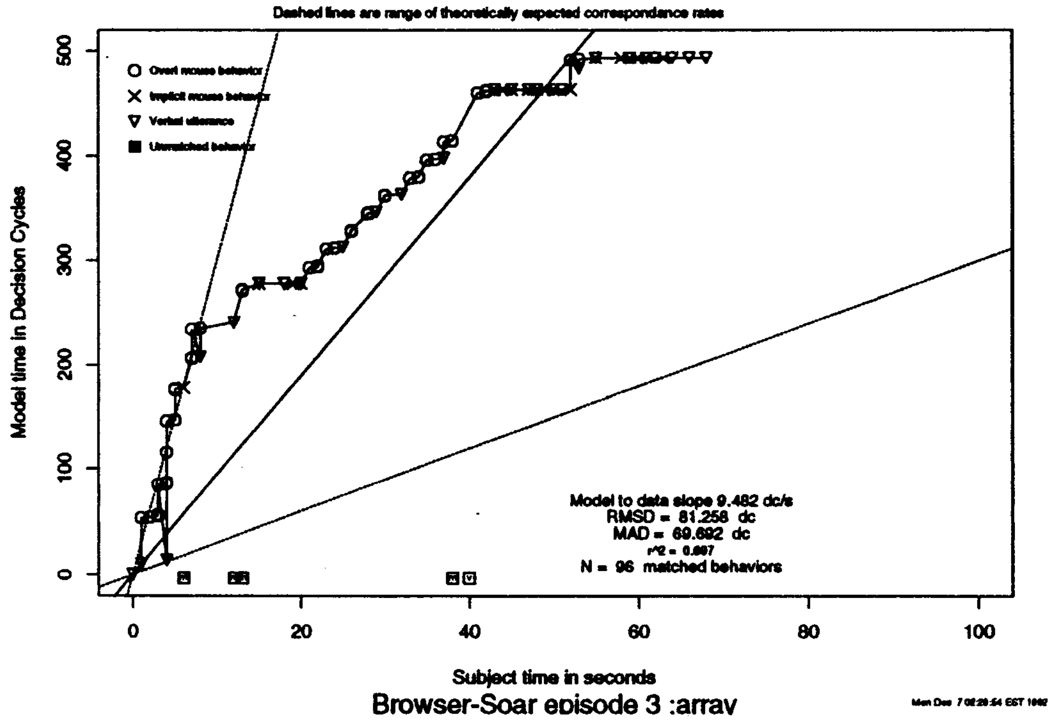


Figure 47: The relative processing rates displays based on decision cycles for each of the episodes.



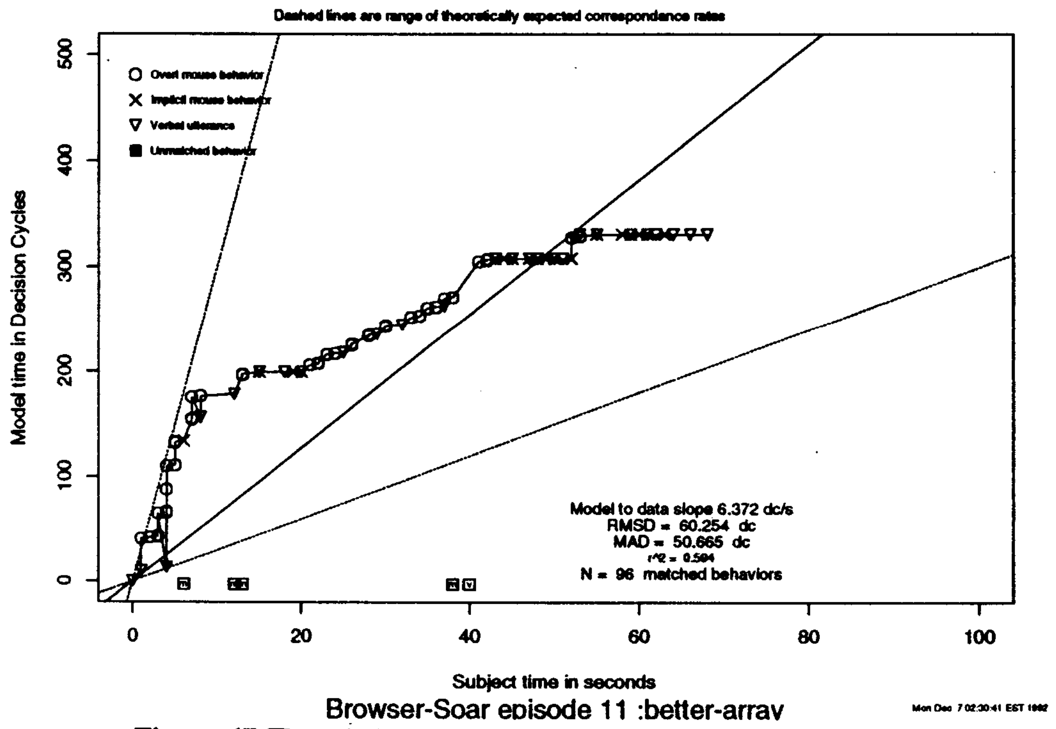


Figure 47: The relative processing rate displays based on decision cycles for of the episodes (cont.).

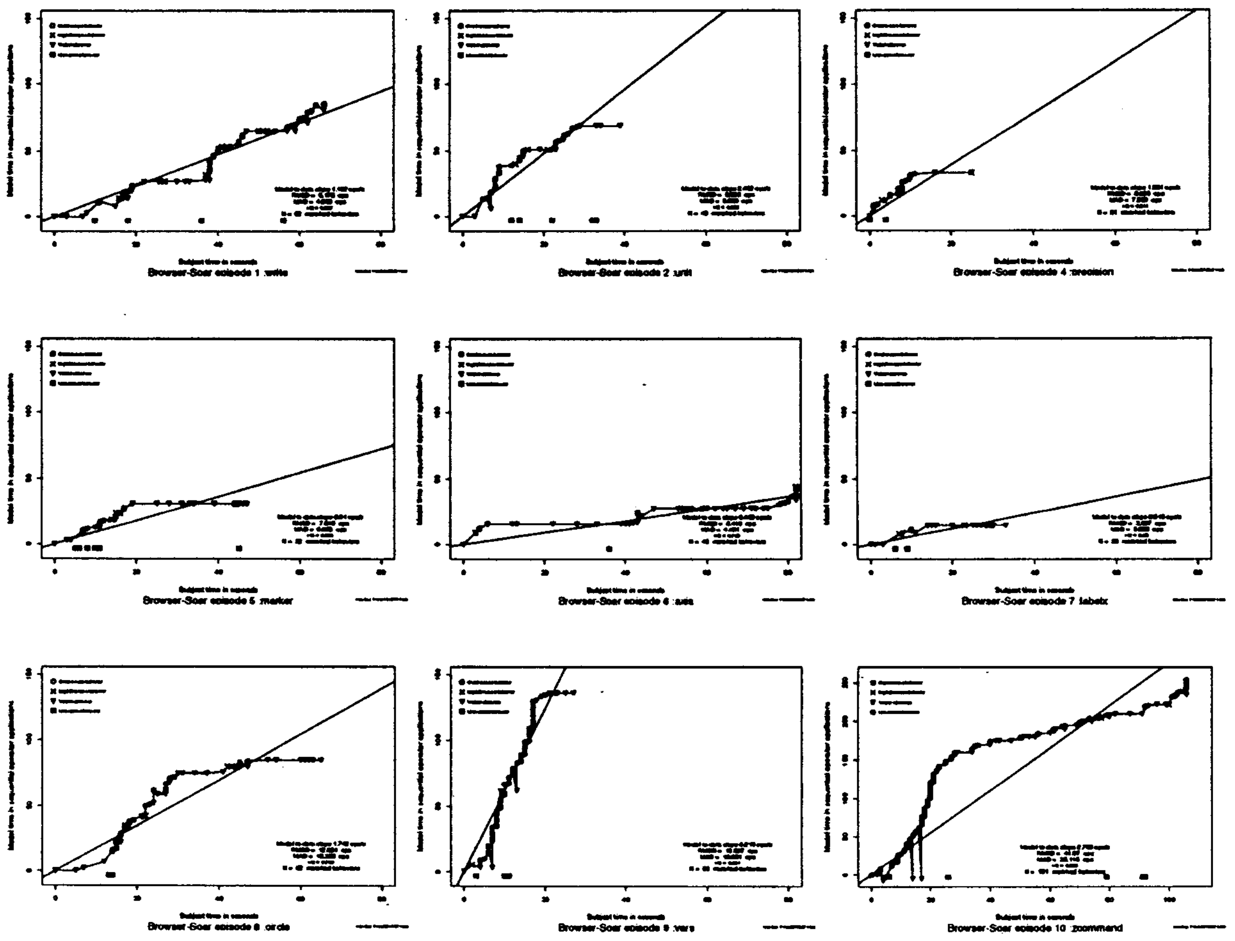
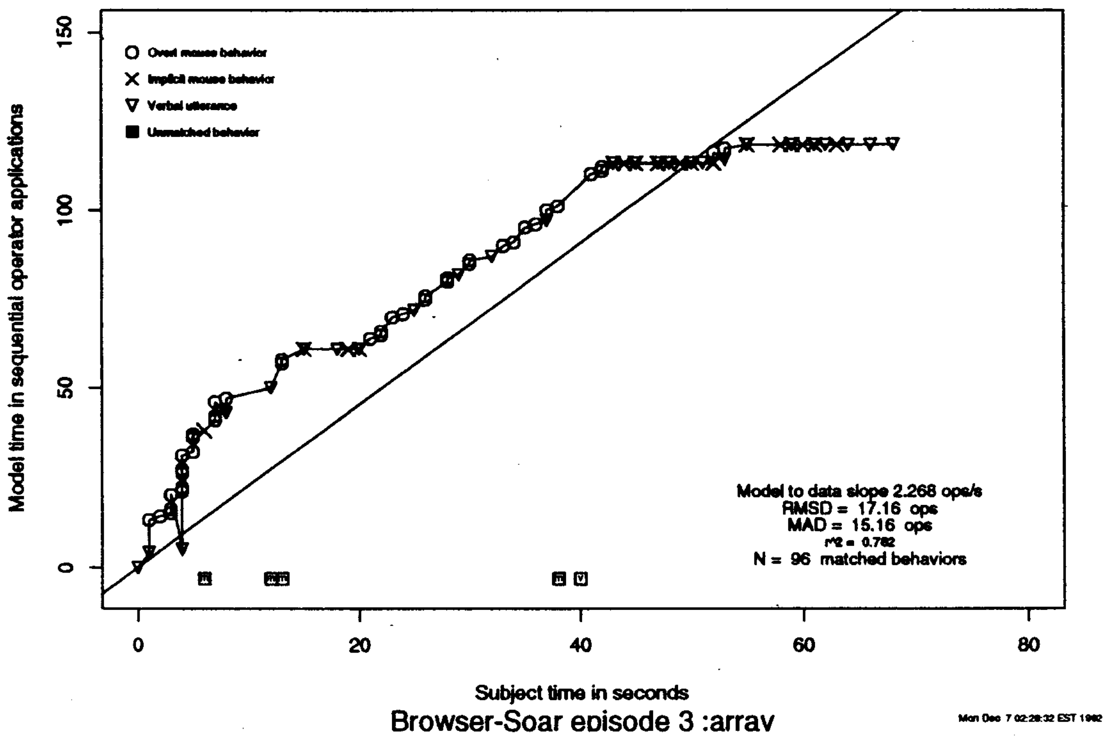


Figure 48: The relative processing rates displays based on operator applications for each of the episodes.



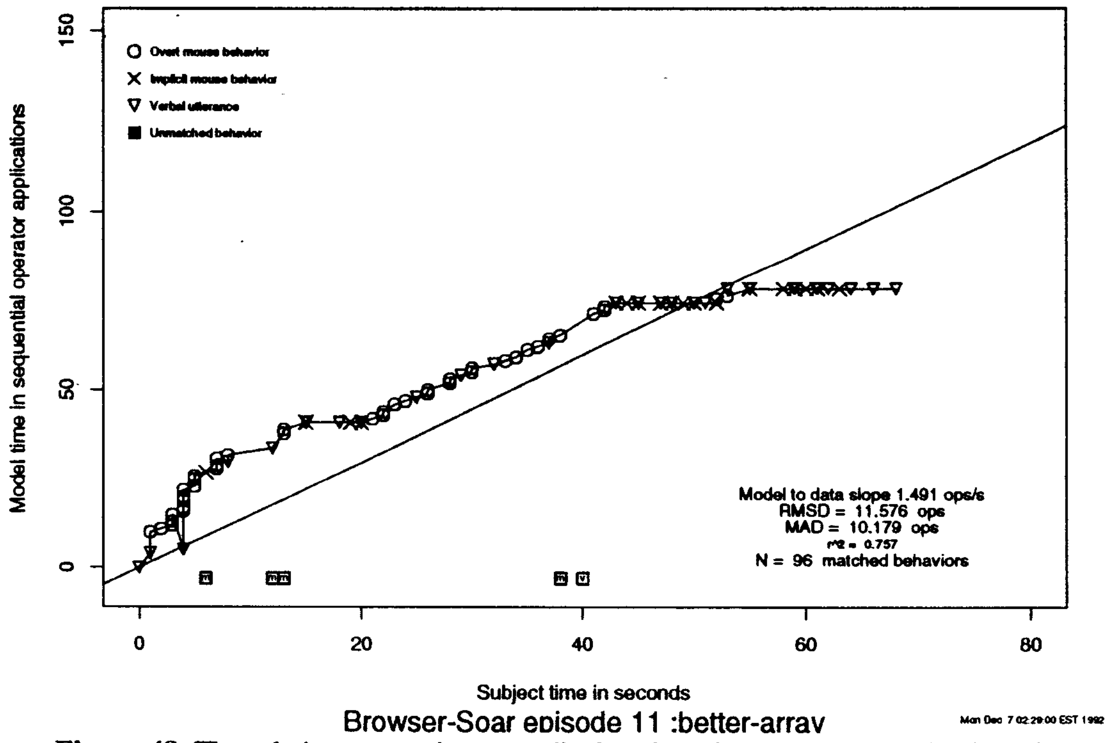


Figure 48: The relative processing rates displays based on operator applications for each of the episodes (cont.).

Chapter 8

Performance demonstration II: Use of Soar/MT components by others

While the environment is integrated, its components have been developed separately. As each component became available, it was spun off for use by others performing subsets of the tasks involved in model testing. The number of users of each tool, their comments, or both, provided feedback on how the various tools help perform (Tesler, 1983) specific tasks of model testing. Together they provides an estimate of the current and potential impact of the whole environment.

Spa-mode has had no use outside of this thesis. As noted earlier, the total environment, but for the displays, was used by V. Peck to perform two episodes of the Browser-Soar reanalysis. The underlying Dismal spreadsheet has had three to four additional users. It still has many problems, so a survey probably will not point out inadequacies not already known.

A survey was conducted of Soar users to find the strengths and weaknesses of the Developmental Soar Interface (DSI).

The other pieces of software either are not used by enough users (Spa-mode, Dismal), or they are so widely used that undertaking a survey is a more serious proposition (S-mode) than can be undertaken as part of this work. Portions of the DSI should no longer be considered pieces of developmental software, for out of the 60 Soar users responding to the survey, two-thirds now use some portion of it every time they use Soar.

8.1 Usage of the Developmental Soar Interface to develop Soar models

The three modules of the DSI (Soar-mode, Taql-mode, and the SX graphic display), have been through several releases. How to obtain them is explained in Appendix I. One or more of the modules are installed at each of the four principle Soar sites in the US, and at sites in Germany and the Netherlands, with over 40 researchers using one or more of the modules.

In the Fall of 1992, a survey (included as an appendix to this chapter) was sent to members of the Soar community identified through the Soar project's mailing lists, workshop attendance lists, and presenters at workshops, as most likely to use Soar in a routine way. In addition to the users directly targeted, an announcement of the survey was emailed to the general Soar mailing list, and an announcement was made at the Soar XI workshop in October, 1992.

Out of the 69 potential users identified, 63 returned a survey (a 92% response rate).. The three people who never actually used Soar were dropped from later analyses. If users that were personally known did not fill in an item, or misidentified a portion of the DSI, this was corrected. Of the people responding, 50 are current members of the Soar community, and 13 are former members.

Table 8-33 shows a listing of the usage patterns. The columns list the components used, with each row representing a single user. The rows are grouped by the sets of components used. The primary tool used is Soar-mode, with 37 of the 60 users reporting using it. The SX graphic display has only been used as a routine tool for debugging by its developer and two other users, but 14 people have used it to create pictures of Soar models and to give demonstrations of their models. Taql-mode has been used and put aside by several people as they became more familiar with the TAQL grammar.

In users' responses of why they did not use additional modules, the largest number of responses (14) was that they did not use TAQL, so they did not need Taql-mode. (This would not necessarily translate into 14 users if they used TAQL.) The next largest concern (12) noted problems with installation and not knowing how to use the tools. Speed (5) was also a concern, and this concern was not limited just to the graphic display, a few users thought that Soar-mode and Taql-mode were slow to load. Most potential users of the SX graphic display were put off by how much it slowed down the system, and while only half the users reported dissatisfaction with its speed, this does not mean that

Table 8-33: Survey responses categorized by usage pattern.
Each row represents a user. Totals do not include "tried" users.

	Frequency of usage				
Components used	Soar	Soar-mode	SX	Taql-mode	Totals
EVERYTHING	daily	daily	Weekly	daily	7
	daily	daily	special	weekly	
	daily	daily	special	daily	
	daily	daily	daily	special	
	weekly	weekly	weekly	monthly	
	weekly	weekly	special	special	
	weekly	weekly	special	daily	
SX & SOAR-MODE	daily	daily	weekly		8
	daily	daily	special	tried	
	daily	daily	special		
	daily	daily	special		
	daily	daily	special		
	daily	daily	special		
	weekly	weekly	special		
	weekly	weekly	weekly	tried	
TAQL-MODE	daily	tried		daily	1
SOAR- & TAQL-MODE	daily	daily		daily	6
	daily	daily	tried	daily	
	daily	daily		daily	
	daily	daily		daily	
	weekly	weekly		weekly	
	monthly	monthly		monthly	
SOAR-MODE	daily	weekly	tried		17
	daily	daily		tried	
	daily	daily			
	daily	daily			
	daily	daily			
	daily	daily			
	daily	daily			
	daily	daily			
	daily	daily			
	weekly	weekly		tried	
	weekly	weekly			
	weekly	weekly			
	weekly	weekly			
	weekly	weekly			
	weekly	weekly			
	monthly	monthly			
	monthly	monthly	tried	tried	
NOTHING	daily	tried		tried	21
	daily	tried			
	daily				
	daily				
	daily				
	daily				
	daily				
	daily				
	daily				
	daily				
	daily				
	daily				
	weekly				
	weekly				
	weekly				
	weekly				
	monthly	tried		tried	
	monthly				
	quarterly				
	na				
	na				
Totals	60	38	15	14	60

they were satisfied with it. There were no underlying problems reported with the metaphor, representations, and manipulation of the problem space level objects.

Other users had problems with the underlying systems that the tools were built on. Several users (4) reported that they did not have a machine that could run the X window system, and some users (2) did not know or want to learn Emacs. A few users, perhaps four or five, use a Macintosh exclusively, or nearly exclusively, and the current environment is unavailable to them.

While only two respondents had not heard of all the software, a few were misinformed. One user did not know that they were using Soar-mode (but loaded it in their startup files), and one did not know that they were using Taql-mode (but when reporting useful Soar-mode features included a feature only in Taql-mode).

Use in video productions. The SX graphic display has been used to make three videos of Soar and Soar models that have been shown outside of CMU. A 20 minute tape of NTD-Soar was shown at a NASA contractors' meeting and as part of a research talk at Queen Mary & Westerfield College, both in the Spring of 1992. A 2 minute video showing the basic interaction method with the DSI and how Soar uses the Garnet toolkit has been shown four times: at the CHI '91 Garnet Special interest group meeting, at the CHI '92 Doctoral Consortium, May 1992, and as part of research talks at the Applied Psychology Unit in Cambridge, England and at Queen Mary & Westerfield College in the Spring of 1992.

Work is underway to create an introductory video explaining Soar (Newell, P., et al., forthcoming). This video is a demonstration of what will be a general capability to take a graphic description of Soar models and create high quality graphic output suitable for commercial broadcast. The initial depictions of the Soar model were created with the SX graphic display and then sent to a commercial computer graphics company for visual enhancement. The project is expected to be completed in the Spring of 1993.

Impact of the DSI on the next release of Soar software: Soar6. In the next release of the Soar software, called Soar 6, several of the features of the DSI have been incorporated or have encouraged the Soar 6 developers to include similar features. These include a very customizable trace, hooks for interacting with Soar-mode, and a better command line interpreter. Soar 6 is still under development; given time, we hope to migrate additional features to Soar 6, such as the ability to display the match set continuously, and the ability to provide a display of which productions will fire on the next decision cycle.

8.2 Usage of S-mode to create functions in S

S-mode has been distributed through three sources that make its total usage hard to compute. It appears, however, to be one of the dominant ways of interacting with S. It was first placed in 1991 in the GNU-Emacs archives at The Ohio State University. This makes it available via anonymous FTP. S-mode has also been distributed via anonymous FTP from the authors' machines. The number of users who picked it up in these two ways cannot be known.

The second mode of distribution, through a statistics software mail server, allows an approximation of a lower bound. Statlib, run by Dr. Michael Meyer at CMU, is a system for distributing statistical software and datasets by electronic mail. The system keeps track of the mail requests for each piece of software and can provide a listing of who requested each piece. Since S-mode was first placed in the Statlib server, there has been 1,043 requests for it, including requests for updated versions (personal communication, M. Meyer, October, 1992).

The exact size of its distribution is confounded further by the nature of GNU-Emacs' copy protection and the nature of S-mode's installation. GNU-Emacs and S-mode are copylefted, which means that users are entitled to (and indeed legally obligated to) provide others with copies upon request, although a copying fee can be charged. How many sites have passed S-mode on would be impossible to compute. GNU-Emacs and its extensions are installed primarily on multi-user machines and distributed file systems. Once installed, many users can use the same piece of software although on different machines. For example, S-mode has been installed with GNU-Emacs on the Andrew system at CMU (and I don't even know who installed it). Any of the approximately 5,000 Andrew users at CMU can use S-mode.

Appendix to Chapter 8: Survey distributed to Soar users

```
                Survey on the Developmental Soar Interface
                             Frank Ritter
                              12-Oct-92

I'm writing up my thesis and would like to get a better headcount of
how many people use the DSI, and how they use it.  Your comments will
also be used to improve the current interface and serve as background
for future versions.

* How often do you use Soar?
  Daily     Weekly     Monthly     Quarterly     Other (describe)
----------------
* Which of the following have you heard of and which have you used?
                                        Heard of       Have used
  SX graphic display (triangle thingy)  Y  N           Y  N
  Soar-mode                             Y  N           Y  N
  Taql-mode                             Y  N           Y  N

* For items you've heard of, but never used, have you considered using any?
  Any specific reasons why you have not used them?




* Are there any features that you would like to see added to the Soar
  interface for programming, editing, or understanding Soar models ?




If you have not used any items, you can quit here.  Thank you.
```

SX graphic display (triangle thingy)
----

* How often you use it ?   (tick one and/or write in a modifying number)

Daily	Weekly	Monthly	Quarterly
Tried once or twice	Never		

Special purpose (e.g., demos, making figures; please explain)

* If you don't use the SX graphic display, why don't you use it?

* How do you use it? (you may tick more than one)

I've only tried it.	
I use it for special debugging.	I use it for routine development.
I use it for demos.	I use it to make presentation diagrams

* How long have you used it (e.g., 3/91 to present) ?

* What are the most valuable features ?

* What are the worst problems/bugs/factors stopping you from using the SX graphic display more often?

```
----------------
Soar-mode, extensions to gnu-emacs for editing productions.
----------------
* How often you use Soar-mode ? (tick one and/or write in a modifying number)

  Daily                      Weekly         Monthly            Quarterly
  Tried once or twice        Never
  Special purpose (e.g., demos, making figures; please explain)

* How do you use Soar-mode?  (you may tick more than one)

    I use it for special debugging.     I use it for routine development.
    I use it for demos.                 I've only tried it.

* How long have you used soar-mode (e.g., 3/91 to present) ?


* What are the most valuable features of Soar-mode?




* What are the worst problems/bugs/factors stopping you from using
  Soar-mode more often?




* If you don't use Soar-mode, why don't you use it?
```

Taql-mode (extensions to Emacs for editing TAQL constucts)

----

* How often do you use taql-mode ? (tick one and/or write in a modifying number)

Daily	Weekly	Monthly	Quarterly
Tried once or twice	Never		

Special purpose (e.g., demos, making figures; please explain)

* How do you use taql-mode? (you may tick more than one)

I've only tried it.	I use it for demos.
I use it for special debugging.	I use it for routine development.

* How long have you used taql-mode (e.g., 3/91 to present) ?

* What are the most valuable features of taql-mode?

* What are the worst problems/bugs/factors stopping you from using taql-mode more often?

* If you don't use taql-mode, why don't you use it?

Additional on-line & hardcopy copies available from Frank Ritter@cs.cmu.edu

Please return surveys by email or hardcopy to Frank Ritter@cs.cmu.edu

Chapter 9

Contributions and steps toward the vision of routine automatic model testing

Compared with Chapter 1, we are not in the same place in many ways, and we are considerably further along toward the capacity to perform routine process model testing. Progress has been made on defining a methodology for testing the sequential predictions of process models. A computer environment has been implemented to support this methodology, and this environment has been used to test an actual model with actual data. Portions of the environment are used by researchers around the world. The environment was used to test and extend the sequentiality assumption of Ericsson and Simon's (1984) theory of verbal protocol production. The path to an intelligent automatic modeling system based on agent tracking is clearer. Only model testing (open analysis) has been considered in this work, but the methodology and environment should largely be applicable to using models to classify sequential behavior (closed analysis) for such things as cognitive-based testing (Ohlsson, 1990).

The central problem: dealing with large amounts of information. Within the methodology of TBPA the essential problem in testing process models still appears to be one of manipulating and understanding the large amounts of information involved: the model, its predictions, and the data used to test it. Scientists do not decry the difficulty of model creation and manipulation as often as they have the amount of bookkeeping required for testing the sequential predictions. The size of the data sets prove a real problem; the amount of qualitative information used in this task is relatively large given the analyst's limited processing capabilities.

Each of the steps in TBPA requires manipulating large amounts of information. This is a central problem that runs through this work, and it is fought in every tool in the Soar/MT environment. Two approaches have been developed for dealing with it. The first is to automate as many tasks as possible, and to support the analyst for the remainder. The second is to design and use visual displays of information.

Secret weapon #1: Automate and support. Automating aspects of each step reduces the work load required of the analyst. Soar/MT assists the analyst by automatically aligning unambiguous parts of protocols, creating model-based summary displays of the comparison, and providing many aids for displaying and manipulating the model. Although the automatic processes fall short of the ideal speed, and still must be speeded up through better algorithm and data structure design, they have proved useful in their current state. The process is not so inherently large or computationally intensive that so-called super-computing will be required.

The data set presented with Browser-Soar (Peck & John, 1992) is not the largest data set ever used to test a model (although it is fairly large, see Table 2-2), but Soar/MT has substantially speeded up the analysis of this data set. We can now imagine analyzing enough protocol data to achieve Ericsson and Simon's (1984) vision of verbal reports as data.

Supporting the analyst in performing the tasks that are not yet automated has required careful design of the displays and manipulation tools for the large amount of information. The current maximum size of the predictions and data, not including the model, is about 330 Kb. The analyst cannot directly visualize and manipulate information sets the size of a small phone book (5,000 names at 60 bits per name, or 300 Kb total). Special displays have been created to show the important trends in the data, which is the next secret weapon.

Secret weapon #2: Scientific visualization of qualitative information. Appropriate visual displays can support faster processing rates and provide new insights (Larkin & Simon, 1981). Visual displays of qualitative information have become central to quantitative data analysis in many domains and they have lead to the major methodology of scientific visualization.

Visual displays should now be considered essential for performing each step of protocol analysis and process model testing. Visual displays help the analyst understand the model's structure and performance, relating them to each other in a single display, the SX graphic display. Tabular displays of the model's predictions, the data, and their correspondences show simple and directly where the model's predictions do and do not match the data. Other displays aggregate the correspondences in terms of the model components and in terms of relative processing rates. These displays summarize where the model performs well and where it performs poorly, providing clues about where and how to improve the model's fit to the data.

9.1 A methodology for testing the sequential predictions of process models

Trace Based Protocol Analysis (TBPA), a methodology for testing the sequential predictions of process models with protocol data has been defined through listing its inputs, processing steps, and their requirements. TBPA tests a model by running it to generate a trace of how the model performs the task. This trace provides a set of theoretical predictions of what will be found in a subject's verbal and non-verbal protocol, and it is used to interpret the data. TBPA is designed to be an integrated and iterative process, so a summary of where the predictions are unmatched in the protocol is then used to modify the model, and the model is run again. The necessary inputs to TBPA, its steps, and the processing requirements for each step to perform the testing routinely, were specified in enough detail to create a computer environment to support this methodology.

<u>Clarification of the testing process.</u> What it means to test the model became clearer from specifying each step in the process. What are tested in any given episode are the model's predictions. The comparison of the predictions with the data is not just one of alignment. The model's predictions are used to interpret the data. With unambiguous data, such as mouse clicks on menu items, the process appears to be one of simple alignment and it can be treated that way. When the data are verbal protocols, then the items in the trace may provide substantial guidance for interpreting the meaning and function of the information described verbally.

Some theories require every prediction to be matched, but the theory of verbal protocol used to interpret the utterances (Ericsson & Simon, 1984) states that not every possible prediction will be found. The model's predictions are predictions of what could be found in the subject's verbal protocol.

<u>The need for declarative versions of models.</u> It is necessary for model based analysis to refer to structures of the model and to note which parts of the model did and did not apply, or were and were not supported. It is necessary to have declarative representations of process models representing procedural knowledge. Running the model to create the structures upon demand is not enough. There is the simple problem that the structures will be created and then disappear as the context changes. There is also a more complicated problem of coverage, on any given run not all the possible structures will be created. Examining the initial implementation of the model is not adequate either, the model might learn from its environment, and computing all the model's structures is equivalent to running it.

At a minimum, it is necessary to create a description of the model computed by observing the model's performance over time, although combinations of the other methods, such as derivation from the static structure, are a useful adjunct. Although this method is the best way to build the model, even this model is not guaranteed to be complete.

The DSI creates a declarative representation of Soar models. While the Soar model runs, the DSI displays and remembers which and how often the problem spaces, states, and operators have been applied. By watching the model as it runs the DSI builds up as complete a view of the model as is possible. The resulting description can be used by other components in the environment. The interpretation environment can use it to initially code the data. The saved model can be used to summarize the correspondences created though interpreting and aligning the data with respect to the predictions.

9.2 Each step in the methodology was supported in a software environment

An environment to support an analyst performing TBPA has been created based on its requirements. The environment directly supports the main tasks of model tracing; interpreting and aligning the model's predictions with the data, both automatically and semi-automatically; aggregating the comparison data in a variety of displays designed to show how to improve the model; understanding and modifying the model based on how it does not fit.

The steps were specified and broken down to a level that they could be performed automatically, or semi-automatically. Building, loading, and running models was supported in a semi-automatic way. Many small tasks are supported through keystroke macros in the structured editors and smarter interfaces. Finding the emergent properties of Soar models (listing the problem spaces and their operators) is supported, as is counting how often they are instantiated. Unambiguous portions of the subject data are now matched automatically. The same algorithm can be used to interpret and align the data in an incomplete and heuristic fashion, requiring the analyst only to check and clean up the approximate interpretation. Finally, the analytic displays can be automatically created from the comparison data.

The environment also supports the requirements of integrating the steps, automating the tasks where possible, and supporting the analyst for the rest. The environment and the methodology it supports were tested by testing a process model, and in the process learning new things about the model and its fit to the data. The tasks in TBPA that the environment support overlap with other tasks often performed in cognitive model building and modification, data manipulation with a tabular display, and exploratory data analysis.

Sub-portions of the environment supported other users doing the sub-tasks for different reasons, the DSI for AI modeling, Dismal for spreadsheets, and S-mode for statistics and graphing. A survey of users of the DSI found that over half the Soar community uses some portion of the DSI whenever they use Soar. It would be safe to say that pieces of the environment supporting these tasks are in use by over 500 researchers around the world.

The analyses are fast enough to be considered routine. A minute long episode of subject data (approximately 20 verbal segments and 30 motor actions in the browsing task) can now be compared with the model's predictions in 2.5 hours given sufficient inputs, the process model and transcribed data. This is almost within automating range; when it took 60 hours to perform (estimate derived from Ohlsson, 1980), too many under specified processes were required, and automating this task was not conceivable.

Example testing of Browser-Soar using TBPA. The methodology was demonstrated on the Browser-Soar (Peck & John, 1992) model. A set of suggestions for improving Browser-Soar was generated, and one of them was implemented. This lead to a slightly better fit, but more importantly, to a much more parsimonious model. Browser-Soar and its data set did not push this methodology in all directions, but this was good. It allowed making headway on some problems by avoiding others.

9.2.1 Interpreting and aligning the model's predictions and the data

This thesis explored the automatic alignment of unambiguous data to model predictions. The Card algorithm for doing this was slightly improved, and its behavior characterized more clearly.

A spreadsheet approach to the comparison process was demonstrated, and it appears to visually support many of the necessary operations on the data that would otherwise require extensive computation by hand. For example, areas where the predictions match the data in a denser manner is clearly presented. The spreadsheet was also effective in supporting the analyst in easily adjusting the alignment manually when necessary.

9.2.2 Analyzing the results of the testing process

A lack of clarity about what measures are necessary or desirable for measuring predictions fit to the data may have contributed to the lack of progress. The review in Chapter 2 outlined the uses and abuses of several of these measures, and championed Grant's (1962) approach of analytic testing, of finding out where the model can be improved.

A display for showing the support of operators in the model was automated, and an additional family of displays were produced for presenting and analyzing the relative processing rate of the subject with respect to the model. These two sets of displays can be created automatically from the comparison data. They have shown the periodicity of human browsing behavior, the types of mismatches between model and data, and ways to improve the fit of the model. There are many ways for data to not match the model. Additional graphs will be necessary, so an environment is provided to assist in editing and designing these graphs.

9.2.3 Steps related to manipulating the model: Prediction generation and modification

While the model's components are used throughout the analyses, the process model itself is directly involved in two steps, that of generating the sequential predictions, and the final step of revising the model based on the testing process.

Generating the predictions. Generating the model's predictions in a way that they can be used for automatic alignment has required extending infrastructure from the model (in this case, a Soar model) out further toward the data. This has resulted in a better trace — one that is less ambiguous and more readable by humans. Based on the example analysis, we also found that a problem space model must provide state traces in addition to operator traces.

The improved trace lead to an unexpected benefit. We found that deriving aggregate measures in the trace was useful for comparing models and describing their behavior in general terms.

Manipulating and creating models. The Developmental Soar Interface demonstrates the feasibility and utility of several design principles. Across the environment it was possible to meet the design shown in Table 9-34.

Table 9-34: The ease of use and learnability design features met by each tool in the environment.

- Provide a path to expertise through:
  - Menus to drive the interface.
  - Keystroke accelerators available and automatically placed on menus for users to learn.
  - Help provided for each command on request.
  - Hardcopy manuals also available on-line through the menu.
- Treat structures on the theoretical level as first class objects.
- Provide a general tool with macro facilities.

These features make the task of inserting the model's knowledge into Soar easier. Keystroke level models can be presented as evidence for this, as well as the fact that approximately two-thirds of the Soar community now use some portion of the DSI in their daily work.

Node based graph display. Many structure display algorithms draw the complete structure, forcing the user to scroll a window pane across it. Presenting Soar's working memory contents is such a structure display task. The set of tasks users need to perform when examining the structures within working memory have been identified, and a display meeting these requirements has been designed and implemented. The task analysis lead to a different design than a big scrollable window — a node-based design that allows users to open up individually selected nodes in working memory, close their parents, and so on. The users seem pleased, and it provides a much faster display.

General results about Soar. The visual and structural representations in the Developmental Soar Interface highlighted several features of Soar models and the TAQL macro language. For TAQL, the templates within the structured editor provided a measure of the cumbersome size of the TAQL syntax.

For several specific models we were able to display how their behavior is not best characterized as just search in problem spaces. Behavior within many models now includes routine behavior, search through problem spaces, migration of knowledge between problem spaces, and composition of knowledge.

Within Soar models in general, displaying their behavior graphically pointed out how ephemeral problem spaces and their structures are. In many ways the application and interactions of objects on the problem space level should be considered as emergent behavior. The structure of the model is only available from repeated viewing; the model itself has no representation of itself, and cannot conjure up all the problem spaces and operators that are possible.

9.2.4 The synergy from integration

The environment receives much of its power from integration. The model, its behavior, the subject data, and the comparison of the model and the data all exist in the same environment. This supports several analyses that would be difficult without the integration and it allows them to be much more fluid. Integration allows: (a) direct, preliminary coding of the protocols based on the model's components; (b) appropriate mixed (text and symbolic graphics) presentation of data in the DSI; (c) appropriate mixed (text and symbolic graphics) presentation of data in the analyses; and (d) the portions of the trace that were well aligned and not well aligned could be directly compared with the model's structures.

9.3 Validated and extended the sequentiality assumption of protocol generation theory

Using the TBPA methodology and the Soar/MT environment, the Browser-Soar model and data of Peck & John (1992) were re-examined. Besides providing a test-bed for the methodology and environment, this effort yielded the following new scientific result.

The verbal protocol production theory of Ericsson and Simon (1984) assumes that working memory structures are reported in the order that they enter working memory. This assumption can be tested with a model that predicts when objects enter working memory. The Soar/MT display of the relative processing rates of the Browser-Soar model and the subject provided a direct visual test of this assumption. The underlying data structures were then directly queried to confirm and count the number of sequential and non-sequential pairs of events there were. In every episode of the Browser-Soar, the sequentiality assumption was found to hold for the verbal protocol. An examination of the non-verbal protocol segments found that they too were always performed in the same order as the model, both for overt task actions, and for actions that were not directly related to the task, such as moving the mouse pointer over words being read on the screen.

The two data streams appeared to be presented in a non-sequential order. Verbal utterances typically lagged 10 to 30 simulation cycles (approximately 1 to 3 s) behind the overt actions; and rarely (3/300) they lagged up to 400 simulation cycles (approximately 40 s).

The shorter lags were probably reports of working memory delayed by workload associated with the task, and minor inconsistencies in the model. Examination of the correspondences showed that the primary cause of the long lags was probably an artifact of the interpretation process. The verbal utterances in the analysis were matched to operators rather than to the state information created by the operators. This approximation simplified the analysis considerably, and it should remain available — it is a valuable technique. But it must be seen as only an approximation; one that will sometimes lead to inconsistencies in the comparison. Any operator that sets up long lasting state information can cause this problem.

As a result of these analyses it is proposed that the sequentiality assumption holds for both verbal utteranances and task actions. Including motor task actions as part of the protocol provides reference points for fixing the correspondences between the predictions and subject's actions, and allows the lag of the verbal utterances to be measured.

9.4 Progress toward the vision of routine applied theoretically guided protocol analysis

This work has made appreciable progress toward the vision of automatic modeling. All the parts of Soar/MT are part of a grand vision of what an integrated modeling and data analysis system would need to do, and could do. The major steps and inputs have been identified as the parts of TBPA, and a prototype environment has been created that an automatic modeling system would need. The next steps will be to create initial models, and to provide a more intelligent process for interpreting ambiguous data with respect to the model's predictions.

Because this environment is based on an architecture for general intelligence, it is conceptually possible to add knowledge to the architecture of how to perform parts or all of the analysis. To do this completely would require incorporating a complete model of the analyst. However, the architecture used in this environment, Soar, also learns. So perhaps an easier, but less direct way to automate this task might be through having a Soar-based agent learn to perform the analyses by watching a series of analyses. As it watched a series of routine analyses over similar episodes be performed, it could follow along, learning how to run the analyses, and then driving the analyses programs itself.

Not that we are there, but we can now see further down the path toward completely automatic modeling. If NL-Soar (a Soar system for interpreting natural language) were to be incorporated, then Soar/MT might take in instructions for different experiments, and use the models that NL-Soar creates from reading the instructions as initial models to predict the behavior of subjects for each experiment (Lewis, Newell & Polk, 1989; Newell, 1991). The alignment also could be automated. The non-verbal overt actions can be compared directly; the verbal utterances would have data structures, the predictions, laying around that are designed to be sufficient to parse them. NL-Soar (Lehman et al., 1991) is available as a potential parser designed to use these predictions.

This style of protocol analysis requires further computer science and AI work: performing the alignment of predictions to natural language, running the models more quickly, and gathering better statistics. But it remains a task within psychology: the real use is for comparing protocols against models' predictions.

Remaining problems. Many problems remained in this methodology and environment. I would like to note a few here to admit its deficiencies, to warn potential users of the current specificity of the tasks Soar/MT can address, and to suggest directions for future work.

How to aggregate support from the predictions to the model structures is not always as straightforward as it appeared in the sample analysis of Browser-Soar. There is a problem of specifying how the predictions are used to interpret the data. There is also a problem in specifying how to aggregate support for model components. Across episodes, the structures in the model that generated the predictions remain and summarize the behavior over time. The current model implemented its operators rather directly and in the same manner each time. This need not be the case. Consider an *Add* operator such as Siegler uses in his work modeling children's arithmetic knowledge (Siegler, 1988; Siegler & Shrager, 1984). Different operands result in different reaction times and error patterns. Assigning support to an operator in this case must be differentiated by the operator's arguments, and a representation for this must be developed. So there is an additional step to TBPA, not yet made explicit, of translating the support that individual predictions receive from the data back to the structures in the model that generated them.

The analyst is currently left with an abduction task of improving the fit with indications of where the model does not fit and with tools for understanding and modifying the model. There are some simple rules that would apply in specific circumstances, and these were noted in the chapter describing the graphical measures of model fit. The possibility of finding a more complete and algorithmic description, like Heise (1987, 1989; Corsaro & Heise, 1990; Heise & Lewis, 1991) provides for his models, should be explored.

Speed, always and everywhere — the analyst always desires a faster system that performs more complicated analyses automatically. Partial views of the data and model are included in this. The recent translation of Soar to the C language offers a speedup in the basic architecture. Taking advantage of this may require translating the DSI into C.

Directions for future work. The way to improve this methodology is the same way to improve a model, by testing and using it on additional models and data sets. Some preliminary discussions have taken place with other researchers about using Soar/MT to test their process models, usually models implemented in Soar.

The software environment could be automated further, and as noted in Chapter 3, the next direct step toward automatic agent modeling would be to represent the knowledge to perform a single step as a Soar model. This would provide further automation. One of the potential places for doing this would be to have NL-Soar parse the verbal utterances, another would be to further automate the generation of the analytical diagrams.

9.5 Concluding remarks

We build our theories, test them, then modify them, iterating through a loop. This loop was described briefly and perhaps for the first time with respect to process models and protocol analysis by Feldman (1962, p. 342). But not surprisingly, it is like all theory testing in science. Models are not primarily tested to be rejected (as the popularization of Popper's (1959) views goes), or tested simply with a significance test to determine their value, but models are tested in order to improve them (Grant, 1962; Newell, 1990, p. 14). By using protocols to test these models, we are not attempting to *code* a segment so that it is *coded*, but we are using the data to build a model (e.g., a simulation process model). That is, to test whether subjects perform the same actions in the same order as the model predicts.

Because they will allow us to see new things, new analyses and tools are also science (Hall, 1992; Laird & Rosenbloom, 1992; Newell, 1991; Ohlsson, 1990; Simon, 1991). New scientific problems are found this way (Toulmin, 1972). Indeed, much of what science consists of — what is passed on from generation to generation of scientists — is just technique (Ohlsson, 1990; Toulmin, 1972).

Because of the difficulties associated with creating process models and of manipulating protocol data, sometimes analysts have lost sight of this fundamental nature of protocol analysis. The technique of testing process models' predictions of sequential behavior has been nudged forward just a bit.

References

Afifi, A., & Clark, V. (1984). *Computer-aided Multivariate Analysis.* New York: Van Nostrand Reinhold Company.

Altmann, E. (February, 1992). Toward Human-scale task performance: Preliminaries. Talk presented at the Soar X workshop.

Anderson, J. R. (1983). *The architecture of cognition.* Cambridge, Massachusetts: Harvard University Press.

Anderson, J. R. (in press). *Rules of the Mind.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Anderson, J. R., & Bower, G. H. (1973). *Human associative memory.* Hillsdale, NJ: Lawrence Erlbaum Associates. Third revised printing 1979.

Anderson, J. R., Conrad, F. G., & Corbett, A. T. (1989). Skill acquisition and the LISP tutor. *Cognitive Science, 13*(4), 467-505.

Anderson, J. R., Farrell, R., & Sauers, R. (1981). Learning to program in Lisp. *Cognitive Science, 8,* 87-129.

Anderson, J. R., Greeno, J. G., Kline, P. J., & Neves, D. M. (1981). Acquisition of problem-solving skill. In Anderson, J. R. (Ed.), *Cognitive skills and their acquisition.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Anjewierden, A., Wielemaker, J., & Toussaint, C. (1990). Shelley — Computer aided knowledge engineering. In Wielinga, B., Boose, J. H., Gaines, B. R., Schreiber, G, & van Someren, M. (Eds.), *Current trends in Knowledge acquisition (Proceedings of the 4rth European workshop on knowledge acquisition, EKAW-90, Amsterdam.* Amsterdam: IOS Press.

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

Ardis, M. A. (1987). Template-Mode for GNU Emacs. Available from The Ohio State University elisp archives on archive.cis.ohio-state.edu as file pub/gnu/emacs/elisp-archive/modes/templatemode.tar.Z.

Atwood, M. E., & Poulson, P. G. (1976). A process model for water jug problems. *Cognitive Psychology, 8,* 191-216.

Bates, D., Kademan, E., & Ritter, F. E. (Fall 1990, revised Fall 1991). *S-mode for GNU Emacs.* Available from the Statlib software archive (S is a statistics package, Statlib is statlib@lib.stat.cmu.edu).

Becker, R.A., Chambers, J.M., & Wilks, A.R. (1988). *The New S Language.* Pacific Grove, CA: Wadsworth and Brooks/Cole.

Bentler, P. M. (1980). Multivariate analysis with latent variables: Causal modeling. *Annual Review of Psychology, 31,* 419-456.

Bhaskar, R. (1978). *Problem solving in semantically rich domains.* Doctoral dissertation, Carnegie-Mellon University.

Bhaskar, R., & Simon, H. A. (1977). Problem solving in semantically rich domains: An example from engineering thermodynamics. *Cognitive Science, 1,* 193-215.

Bree, D. S. (1968). *The understanding process as seen in geometry theorems.* Doctoral dissertation, Carnegie Mellon University.

Brooks, F. P. (1975). *The mythical man-month: Essays on software engineering.* Reading, MA: Addison-Wesley Pub. Co.

Brown, C. R. (1986). The verbal protocol analysis tool (VPA): Some formal methods for describing expert behavior. In *Proceedings 2nd Symposium on Human Interface, Oct. 29-30, Tokyo, Japan,* 561-567.

Brown, J. S., & Burton, R. B. (1980). Diagnostic models for procedural bugs in basic mathematical skills. *Cognitive Science, 2,* 155-192.

Brueker, J., & Wielinga, B. (1989). Models of expertise in knowledge acquisition. In Guida, G., & Tasso, C. (Eds.), *Topics in expert system design.* North Holland: Elsevier Science Publishers B.V.

Card, S. K., Moran, T. P., & Newell, A. (1980). The keystroke-level model for user performance time with interactive systems. *Communications of the ACM,* Vol. *23*(7).

Card, S., Moran, T., & Newell, A. (1983). *The psychology of human-computer interaction.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Carley, K. (1988). Formalizing the social expert's knowledge. *Sociological methods and research, 17*(2), 165-232.

Carpenter, P. A., Just, M. A., & Shell, P. (1990). Cognitive coordinate systems: Accounts of mental rotation and individual differences in spatial ability. *Psychological Review, 92,* 137-172.

Chambers, J.M., & Hastie, T.J., eds. (1992). *Statistical Models in S.* Pacific Grove, CA: Wadsworth and Brooks/Cole.

Cohen, M. S., Payne, D. G., & Pastore, R. E. (1991). Computerized task analysis. *SIGCHI Bulletin, 23*(4), 57-58.

Corsaro, W. A., & Heise, D. R. (1990). Event structure models from ethnographic data. In Clogg, C. (Ed.), *Sociological methodology: 1990.* Cambridge, MA: Basil Blackwell.

Dansereau, D. (1969). *An information processing model of mental multiplication.* Doctoral dissertation, Department of Psychology, Carnegie-Mellon University.

Diederich, J., Ruhmann, I, & May, M. (1987). KRITON: A knowledge-acquisition tool for expert systems. *International Journal of Man-Machine Studies, 26,* 29-40.

Dillard, J. F., Bhaskar, R., & Stephens, R. G. (1982). Using first-order cognitive analysis to understand problem solving behavior: An example from accounting. *Instructional Science, 11*(1), 71-92.

Doorenbos, R., Tambe, M., & Newell, A. (1992). Learning 10,000 chunks: What's it like out there? *Proceedings of the Tenth National Conference on Artificial Intelligence.* AAAI.

Dukes, N. F. (1968). N=1. *Psychological Bulletin, 64*(1), 74-79.

Embretson, S. E. (1992). Computerized adaptive testing: Its potential substantive contributions to psychological research and assessment. *Current directions in psychological science, 1*(4), 129-131.

Ericsson, K. A., & Simon, H. A. (1980). Protocol analysis: Verbal reports as data. *Psychological Review, 87,* 215-251.

Ericsson, K. A., & Simon, H. A. (1984). *Protocol analysis: Verbal reports as data.* Cambridge, MA: The MIT Press.

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

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

Feldman, J., Tonge, F. M., & Kanter, H. (1991). Empirical explorations of a hypothesis-testing model of binary choice behavior. Hoggatt, A. C., & Balderston, F. E. (Eds.), *Symposium on simulation models.* Cincinnati, OH, South-Western Publishing Company.

Fielding, N. G., & Lee, R. M. (Eds.). (1991). *Using computers in qualitative research.* London & Beverly Hills, CA: Sage.

Finlay, J., & Harrison, M. (1990). Pattern recognition and interaction models. Diaper, D., et al. (Eds.), *Human-computer interaction — INTERACT '90.* IFIP, Elsevier Science Publishers B. V.

Fisher, C. (1987). Advancing the study of programming with computer-aided protocol analysis. In Olson, G., Soloway, E., & Sheppard, S. (Eds.), *Empirical studies of programmers: Second workshop.* Norwood, NJ: Ablex.

Fisher, C. (1991). *Protocol Analyst's Workbench: Design and evaluation of computer-aided protocol analysis.* Doctoral dissertation, Department of Psychology, Carnegie-Mellon University.

Forgy, C. L. (1981). *OPS5 user's manual* (Tech. Rep.) CMU-CS-81-135. Department of Computer Science, Carnegie-Mellon University.

Free Software Foundation. (1988). *GNU Emacs.* Boston: Free Software Foundation. Directions for obtaining GNU-Emacs are available by FTPing file /pub/gnu/GNUinfo/FTP on prep.ai.mit.edu, using the anonymous FTP protocol.

Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of NP-Completeness.* New York, New York: W. H. Freeman and Company.

Garlick, S. & VanLehn, K. (1987). *CIRRUS: An automated protocol analysis tool* (Tech. Rep.) 6. Department of Psychology, Carnegie-Mellon University.

Gascon, J. (1976). *Computerized protocol analysis of the behavior of children on a weight seriation task.* Doctoral dissertation, Departement de Psychologie, Universitè de Montrèal.

Gottman, J. M., & Roy, A. K. (1990). *Sequential analysis: A guide for behavioral researchers.* Cambridge, UK: Cambridge University Press.

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

Greenblatt, R. D., Knight, T. F. Jr., Holloway, J., Moon, D. A., & Weinreb, D. L. (1984). The LISP machine. In Barstow, D. R., Shrobe, H. E., & Sandewall, E. (Eds.), *Interactive programming environments.* New York, NY: McGraw-Hill.

Greeno, J. G., and Simon, H. A. (1984). Problem solving and reasoning. In Atkinson, R. C., Herrnstein, G., Lindzey, G., and Luce, R. D. (Eds.), *Stevens' handbook of experimental psychology, 2nd edition, Volume II.* New York, NY: John Wiley & Sons. Also available as tech report UPITT/LRDC/ONR/APS-14.

Gregg, L. W., & Simon, H. S. (1967). An information-processing explanation of one-trial and incremental learning. *Journal of Verbal Learning and Verbal Behavior, 6,* 780-787.

Hall, S. (1992). How technique is changing science. *Science, 257,* 344-349.

Hansen, J. P. (1991). The use of eye mark recordings to support verbal retrospection in software testing. *Acta Psychologica, 76,* 31-49.

Hegarty, M. (1988). *Comprehension of diagrams accompanied by text.* Doctoral dissertation, Department of Psychology, Carnegie-Mellon University.

Heise, D. R. (August 1987). Computer assisted analysis of qualitative field data, Didactic seminar, Session 176, American Sociological Association, Chicago.

Heise, D. R. (1989). Modeling event structures. *Journal of Mathematical Sociology, 14*(2-3), 139-169.

Heise, D. R. (1991). Event structure analysis: A qualitative model of quantitative research. In Fielding, N. G., & Lee, R. M. (Eds.), *Using computers in qualitative research.* London: Sage.

Heise, D., & Lewis, E. (1991). *Introduction to ETHNO.* Dubuque, Iowa: Wm. C. Brown Publishers.

Hinton, G. E., & Sejnowski, T. J. (1986). Learning and relearning in Boltzmann Machines. In *Parallel distributed processing: Explorations in the microstructure of cognition. Volume 1: Foundations.* Cambridge, Massachusetts: The MIT Press.

Hirschberg, D. S. (1975). A linear space algorithm for computing maximal common subsequences. *Communications of the ACM, 18*(6), 341-343.

James, J. M., Sanderson, P. M., & Seidler, K. S. (1990). *SHAPA Version 2.0: Instruction manual and reference* (Tech. Rep.) EPRL-90-16/M. Engineering Psychology Research Laboratory.

Jefferys, W. H., & Berger, J. O. (1992). Ockham's Razor and Bayesian Analysis. *American Scientist, 80*(January-February), 64-72.

John, B. E. (1988). *Contributions to engineering models human-computer interactions, Volume 1.* Doctoral dissertation, Department of Psychology, Carnegie-Mellon University.

John, B. E. (1990). Applying cognitive theory to the evaluation and design of human-computer interfaces. Final report to US West sponsored research program.

John, B. E., & Vera, A. H. (May 1992). A GOMS analysis of a graphic, machine-paced, highly interactive task. *CHI'92 Proceedings of the Conference on Human Factors and Computing Systems.* NewYork: SIGCHI, ACM Press.

John, B. E., Vera, A. H., & Newell, A. (December 1990). *Toward Real-Time GOMS* (Tech. Rep.) CMU-CS-90-195. School of Computer Science, Carnegie-Mellon University.

John, B.E., Remington, R.W. & Steier, D.M. (May 1991). *An Analysis of Space Shuttle Countdown Activities: Preliminaries to a Computational Model of the NASA Test Director* (Tech. Rep.) CMU-CS-91-138. School of Computer Science, Carnegie-Mellon University.

Johnson, T. R., & Smith, J. W. (1991). A Framework for Opportunistic Abductive Strategies. *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society.* Hillsdale, New Jersey: Cognitive Science Society, Lawrence Erlbaum Associates.

Johnson, P. E., Dura'n, A. S., Hassebrock, F., Moller, J., Prietula, M., Feltovich, P. J., Swanson, D. B. (1981). Expertise and error in diagnostic reasoning. *Cognitive Science, 5,* 235-283.

Just, M. A., & Carpenter, P. A. (1980). A theory of reading: From eye fixations to comprehension. *Psychological Review, 87*(4), 329-354.

Just, M. A., & Carpenter, P. A. (1985). Cognitive coordinate systems: Accounts of mental rotation and individual differences in spatial ability. *Psychological Review, 92*(2), 137-172.

Just, M. A., & Carpenter, P. A. (1987). *The psychology of reading and language comprehension.* Newton, MA: Allyn & Bacon.

Just, M. A., & Carpenter, P. A. (1992). A capacity theory of comprehension: Individual differences in working memory. *Psychological Review, 99*, 122-149.

Just, M. A., & Thibadeau, R. A. (1984). Developing a computer model of reading times. In Kieras, D. E., & Just, M. A. (Eds.), *New methods in reading comprehension research.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Kadane, J. B., Larkin, J. H., & Mayer, R. H. (1981). A moving average model for sequenced reaction-time data. *Journal of Mathematical Psychology, 23*(2), 115-133.

Kaplan, C. (1987). Computer simulation: Separating fact from fiction. Published as Technical report #498 in the C. I. P. Series, Department of Psychology, Carnegie-Mellon University.

Karat, J. (1968). A model of problem solving with incomplete constraint knowledge. *Cognitive Psychology, 14*, 538-559.

Kennedy, S. (1989). Using video in the BNR usability lab. *SIGCHI Bulletin, 21*(2), 92-95.

Kiearas, D. (May 1992). Personal communication.

Klahr, D., & Dunbar, K. (1988). Dual space search during scientific reasoning. *Cognitive Science, 12*(1), 1-48.

Koedinger, K. R., & Anderson, J. R. (1990). Abstract planning and perceptual chunks: Elements of expertise in geometry. *Cognitive Science, 14*(4), 511-550.

Kolen, J. F., & Pollack, J. B. (1988). Scenes from exclusive-or: Back propagation is sensitive to initial conditions. *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society.* Cognitive Science, LEA.

Kowalski, B., & VanLehn, K. (1988). Cirrus: Inducing subject models from protocol data. *Proceedings of the Tenth Annual Conference of the Cognitive Science Society.* Cognitive Science, LEA.

Krishnan, R., Li, X., & Steier, D. M. (September 1992). Development of a knowledge-based mathematical model formulation system. *Communications of the ACM, 35*(9), 138-146.

Kulkarni, D., & Simon, H. A. (1988). The process of scientific discovery: The strategy of experimentation. *Cognitive Science, 12*, 139-176.

Laird, J. E., & Rosenbloom, P. S. (1992). In pursuit of mind: The research of Allen Newell. To appear in *AI Magazine.*

Laird, J.E., Congdon, C.B., Altmann, E. & Swedlow, K. (October 1990). *Soar User's Manual: Version 5.2* (Tech. Rep.) CSE-TR-72-90. Electrical Engineering and Computer Science Department, University of Michigan. Also available from The Soar Project, School of Computer Science, Carnegie-Mellon University, as technical report CMU-CS-90-179.

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

Langley, P., & Ohlsson, S. (1989). Automated cognitive modeling. *Proceedings of AAAI-84*. Los Altos, CA, Morgan Kaufman.

Langley, P., Bradshaw, G. L., & Simon, H. A. (1983). Rediscovering chemistry with the Bacon system. In Michalski, R. S., Carbonell, J. G., & Mitchell, T. M. (Eds.), *Machine learning, an artificial intelligence approach*. Palo Alto, CA: Tioga.

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

Larkin, J. H., & Simon, H. A. (1981). Learning through growth of skill in mental modeling. *In Proceedings of the Third annual conference of the Cognitive Science Society*. Cognitive Science Society, Lawrence Erlbaum Associates.

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

Larkin, J. H., Mayer, R. H., & Kadane, J. B. (1986). An information-processing model based on reaction times in solving linear equations. *Journal of Mathematical Psychology, 23*(2), 115-133.

Larkin, J. H., McDermott, J., Simon, D. P., & Simon, H. A. (1980). Models of competence in solving physics problems. *Cognitive Science, 4*, 317-345.

Lehman, J. F., Lewis, R. L., & Newell, A. (1991). Integrating knowledge sources in language comprehension. *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society*. .

Levy, B. (Fall 1991). Able Soar Jr: A model for learning to solve kinematic problems. Final project for PSY 85-711: Cognitive processes and problem solving.

Lewis, R. L., Huffman, S. B., John, B. E., Laird, J. E., Lehman, J. F., Newell, A., Rosenbloom, P. S., Simon, T., & Tessler, S. G. (July 1990). Soar as a Unified Theory of Cognition: Spring 1990. *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society*. Cambridge, MA.

Lewis, R. L., Newell, A., & Polk, T. A. (1989). Toward a Soar theory of taking instructions for immediate reasoning tasks. *Proceedings of the Annual Conference of the Cognitive Science Society*. Hillsdale, New Jersey: Cognitive Science Society, Lawrence Erlbaum Associates.

Lueke, E., Pagerey, P. D., & Brown, C. R. (1987). User requirements gathering through verbal protocol analysis. In Salvendy, G. (Ed.), *Cognitive Engineering in the Design of Human-Computer Interaction and Expert Systems*. Amsterdam: Elsevier Science Publishers.

Luger, G. F. (1981). Mathematical model building in the solution of mechanics problems: Human protocols and the MECHO trace. *Cognitive Science, 5*, 55-77.

Mackay, W. (1989). EVA: An experimental video annotator for symbolic analyses of video data. *SIGCHI Bulletin, 21*(2), 68-71.

MacWhinney, B. (1991). *The CHILDES project: Tools for analyzing talk*. Hillsdale, NJ: Lawrence Erlbaum Associates.

MacWhinney, B., & Snow, C. (1990). The Child Language Data Exchange System: An update. *Journal of Child Language, 17*, 457-472.

McClelland, J. L., & Rumelhart, D. E. (1988). *Explorations in parallel distributed processing: A handbook of models, programs, and exercises*. Cambridge, Massachusetts: The MIT Press.

McClelland, J. L., Rumelhart, D. E., & the PDP research group. (1986). *Parallel distributed processing: Explorations in the microstructure of cognition. Volume 2: Psychological and biological models.* Cambridge, Massachusetts: The MIT Press.

McConnell, C. (Spring 1992). *Ilisp: Fancy LISP interface for GNU Emacs that supports multiple dialects* (Version 4.12). Available from The Ohio State University elisp archives on archive.cis.ohio-state.edu as file pub/gnu/emacs/elisp-archive/packages/ilisp.tar.Z, and from katmandu.mt.cs.cmu.edu:/pub/ilisp/ilisp.tar.Z.

Miller, C. S., & Laird, J. E. (1991). A Constraint-Motivated Model of Lexical Acquisition. *Proceedings of the thirteenth annual conference of the Cognitive Science Society.* Cognitive Science, Lawrence Erlbaum Associates.

Milnes, B. G. (1988). The Soar Graphic Interface. Talk and demo presented at the Soar V Workshop.

Miwa, K., & Simon, H. A. (1992). Measuring individual differences by modifying production systems. Submitted for publication.

Motta, E., Eisenstadt M., Pitman, K., & West, M. (1988). Support for knowledge acquisition in the Knowledge Engineer's Assistant (KEATS). *Expert Systems, 5*, 6-28.

Myers, B. A., & Rossen, M. B. (May 1992). Survey on user interface programming. *CHI'92 Proceedings of the Conference on Human Factors and Computing Systems.* NewYork, ACM Press, Also available as Carnegie-Mellon School of Computer Science technical report CMU-CS-92-113.

Myers, B. A., Giuse, D. A., Dannenberg, R. B., Vander Zanden, V., Kosbie, D. S., Pervin, E., Mickish, A., & Marchal, P. (November 1990). Garnet: Comprehensive Support for Graphical, Highly-Interactive User Interfaces. *IEEE Computer, 23*(11), 71-85.

Myers, B. A., Guise, D., Dannenberg, R. B., Vander Zanden, B., Kosbie, D., Marchal, P., Pervin, E., Mickish, A., Kolojejchick, J. A. (1991). *The Garnet toolkit reference manuals, revised for Version 1.4* (Tech. Rep.) CMU-CS-90-117-R. School of Computer Science, Carnegie-Mellon University.

Neches, R. (1982). A process model for water jug problems. *Behavior research methods & instrumentation, 14*(2), 77-91.

Neches, R., Langley, P., & Klahr, D. (1987). Learning, development, and production systems. In Klahr, D., Langley, P., & Neches, R. (Eds.), *Production system models of learning and development.* Cambridge, MA: Massachusetts Institute of Technology.

Nelson, T. O. (1984). A comparison of current measures of the accuracy of feeling-of-knowing predictions. *Psychological Bulletin, 95*(1), 109-133.

Nerb, J. & Krems, J. (1992). Kompetenzerwerb beim Loesen von Planungsproblemen: experimentelle Befunde und ein SOAR-Model (Skill acquisition in solving scheduling problems: Experimental results and a Soar model.). FORWISS-Report FR-1992-001, Muenchen (Germany).

Neter, J., Wasserman, W., & Kutner, M. H. (1985). *Applied linear statistical models.* Homewood, IL: Irwin.

Neuwirth, C. M., Kaufer, D. S., Chandhok, R., & Morris, J. H. (1990). Issues in the design of computer support for co-authoring and commenting. *In Proceedings of the Third Conference on Computer Supported Cooperative Work (CSCW'90).* Computer supported cooperative work, Association for Computing Machinery.

Newell, A. (1968). On the analysis of human problem solving protocols. In Gardin, J. C., & Jaulin, B. (Eds.), *Calcul et formalisation dans les sciences de l'homme*. Paris: Centre National de la Recherche Scientifique. Excerpt published in Johnson-Laird, P. J., & Wason, P. C. (1977) (Eds.), "On the analysis of human problem solving protocols", *Thinking: Readings in cognitive science*, 46-61, Bath (UK): The Pitman Press.

Newell, A. (1972). A theoretical exploration of mechanisms for coding the stimulus. In Melton, A. W., & Martin, E. (Eds.), *Coding processes in human memory*. Washington, DC: V. H. Winstor.

Newell, A. (1973). You can't play 20 questions with nature and win. In Chase, W. G. (Ed.), *Visual information processing*. New York, NY: Academic Press.

Newell, A. (1982). The knowledge level. *Artificial Intelligence*, *18*, 87-127.

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

Newell, A. (1991). Desires and diversions. School of Computer Science Distinguished Lecture, Carnegie-Mellon University. December 4rth.

Newell, A. (1992). Unified theories of cognition and the role of Soar. In Michon, J. A., & Akyurek, A. (Eds.), *Soar: A cognitive architecture in perspective*. Dordrecht (the Netherlands): Kluwer Academic Publishers.

Newell, A. (1980a). Perception and production of fluent speech. In Cole, R. (Ed.), *Harpy, production systems, and human cognition*. Hillsdale, NJ: Lawrence Erlbaum Associates. Also available as CMU tech. report CMU-CS-78-140.

Newell, A. (1980b). Reasoning, problem solving and decision processes: The problem space as a fundamental category. In Nickerson, R. (Ed.), *Attention and performance VIII*. Hillsdale, NJ: Lawrence Erlbaum Associates. Also available as a Department of Computer Science, Carnegie-Mellon University tech report.

Newell, A., & Rosenbloom, P. S. (1981). Mechanisms of skill acquisition and the law of practice. In Anderson, J. R. (Ed.), *Cognitive skills and their acquisition*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Newell, A., & Simon, H. A. (1961). The simulation of human thought. In *Current trends in psychological theory*. Pittsburgh, PA: University of Pittsburgh Press. Also available as RAND tech. reports P-1734 and RM-2506.

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

Newell, A., & Steier, D. (1992). Intelligent Control of External Software Systems. *AI in Engineering*, Vol. *in press*. Also available as Technical Report EDRC 05-55-91, Engineering Design Research Center, Carnegie Mellon University, April, 1991.

Newell, A., Shaw, J. C., & Simon, H. A. (1958). Elements of a theory of human problem solving. *Psychological Review*, *65*, 151-166.

Newell, A., Shaw, J. C., & Simon, H. A. (1960). Report on a general problem-solving program for a computer. *Information processing: Proceedings of the international conference on information processing*. Paris, UNESCO, Also available as Rand tech. report P-1584; reprinted in *Computers and Automation*, July 1959..

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

Newell, P., Lehman, J., Altmann, E., Ritter, F., & McGinnis, T. (1992). The Soar video. forthcoming.

Norman, D. A. (1990). Approaches to the study of intelligence. *Artificial Intelligence*, Vol. *26*. Also to be published in Kirsh, D. (Ed.) (in preparation). *Foundations of artificial intelligence*. Cambridge, MA: MIT Press.

O'Reilly, R. C. (1991). *X3DNet: An X-Based Neural Network Simulation Environment*. Available from oreilly@cmu.edu, or via anonymous FTP from hydra.psy.cmu.edu as file pub/x3dnet/x3dnet.tar.Z.

Ohlsson, S. (1980). *Competence and strategy in reasoning with common spatial concepts: A study of problem solving in a semantically rich domain*. Doctoral dissertation, U. of Stockholm. Also published as #6 in the Working papers from the Cognitive seminar, Department of Psychology, U. of Stockholm.

Ohlsson, S. (1990). Trace analysis and spatial reasoning: An example of intensive cognitive diagnosis and its implications for testing. In Frederiksen, N., Glaser, R., Lesgold, A., & Shafto, M. G. (Eds.), *Diagnostic monitoring of skill and knowledge acquisition*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Ohlsson, S. (October 1992). Personal communication.

Olson, J. S., Olson, G. M., Storrosten, M., & Carter, M. (1992). How a group-editor changes the character of a design meeting as well as its outcome. Paper presented at the HCI Consortium meeting, February 1992.

Peck, V. A. (November 1992). Personal communication.

Peck, V. A., & John, B. E. (May 1992). Browser-Soar: A computational model of a highly interactive task. *CHI'92 Proceedings of the Conference on Human Factors and Computing Systems*. NewYork, ACM Press.

Pitman, K. M. (1985). *Cref: An editing facility for managing structured text* (Tech. Rep.) A.I. Memo No. 829. Massachusetts Institute of Technology, Artificial Intelligence Laboratory.

Platt, J. R. (1964). Strong Inference. *Science*, *146*(3642), 347-353.

Polk, T. A. (August 1992). *Verbal reasoning*. Doctoral dissertation, School of Computer Science, Carnegie-Mellon University.

Poltrock, S. E., & Nasr, M. G. (1989). *Protocol analysis: A tool for analyzing human-computer interactions* (Tech. Rep.) ACT-HI-186-89. Microelectronics and Computer Technology Corporation.

Popper, K. R. (1959). *The logic of scientific discovery*. New York, NY: Basic Books.

Priest, A. G., & Young, R. M. (1988). Methods for evaluating micro-theory systems. In Self, J. (Ed.), *Artificial Intelligence and Human Learning: Intelligent Computer-Aided Instruction*. London: Chapman and Hall.

Qin, Y., & Simon, H. A. (1990). Laboratory replication of scientific discovery processes. *Cognitive Science*, *14*(2), 281-312.

Quinlan, R. (1983). Learning efficient classification procedures and their application to chess end games. In Michalski, R. S., Carbonell, J. G., & Mitchell, T. M. (Eds.), *Machine learning:*

An artificial intelligence approach. Palo Alto, CA: Tioga.

Reiser, B. J., Anderson, J. R., & Farrell, R. G. (1985). Dynamic student modeling in an intelligent tutor for LISP programming. *Proceedings of the International Joint Conference on Artificial Intelligence - 85.* Los Angeles: International Joint Conference on Artificial Intelligence.

Reynolds, H. T. (1984). *Analysis of nominal data* (Tech. Rep.) 07-001. London & Beverly Hills, CA: Sage university paper series on quantitative application in the social sciences.

Ritter, F. E. (September, 1988). "Extending the Seibel-Soar Model". Presented at the Soar V Workshop held at CMU.

Ritter, F. E. (1989). Transparencies from Soar Meeting, May, 1989. FOKIBOFIT-Soar: A Soar model of the effect of problem-part frequency on feeling-of-knowing, Department of Psychology, Carnegie-Mellon University, Unpublished. Also presented as part of the Understand Seminar, PSY 85-811.

Ritter, F. E. (1991). *TAQL-mode Manual.* The Soar Project, School of Computer Science, Carnegie-Mellon University.

Ritter, F. E. (February, 1992). "Bruno Levy's Able-Soar, Jr. model". Presented at the Soar X Workshop held at The University of Michigan.

Ritter, F. E., & Fox, D. (1992). *Dismal: A spreadsheet for GNU-Emacs.* The Soar Project, School of Computer Science, Carnegie-Mellon University.

Ritter, F. E., & McGinnis, T. F. (1992). Manual for *SX: A graphical display and interface for Soar in X windows.* The Soar Project, School of Computer Science, Carnegie-Mellon University.

Ritter, F. E., Hucka, M., & McGinnis, T. F. (1992). *Soar-mode Manual* (Tech. Rep.) CMU-CS-92-205. School of Computer Science, Carnegie-Mellon University.

Rosenbloom, P. S. & Lee, S. (1989). Soar arithmetic and functional capability. Software provided with the Soar 5 distribution.

Rosenbloom, P. S., & Newell, A. (September 1982). *Learning by chunking, a production system model of practice* (Tech. Rep.) CMU-CS-82-135. Department of Computer Science, Carnegie-Mellon University.

Rosenbloom, P. S., Laird, J. E., & Newell, A. (1987). Meta-levels in Soar. In Maes, P., & Nardi, D. (Eds.), *Meta-Level Architectures and Reflection.* Amsterdam: North Holland Publishing Company.

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

Sakoe, H., & Chiba, S. (1978). Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on acoustics, speech, and signal processing, 26*(1), 43-49.

Samual, A. L. (July 1959). Some studies in machine learning using the game of checkers. *IBM J. of Research and Development, 3,* 210-229.

Sanderson, P. M. (1990). Verbal protocol analysis in three experimental domains using SHAPA. *Proceedings of the Human Factors Society 34th Annual Meeting.* Human Factors Society.

Sanderson, P., James, J., Watanabe, L., & Holden, J. (1990). Human operator behavior in complex worlds: Rendering sequential records analytically tractable. *Proceedings of the 9th Annual Conference on Human Decision Making and Manual Control.* Varese, Italy, Also available from the Engineering Psychology Research Laboratory, Department of Mechanical and Industrial Engineering, U. of Illinois, as technical report EPRL-90-12.

Sanderson, P. M., Verhage, A. G., & Fuld, R. B. (1989). State-space and verbal protocol methods for studying the human operator in process control. *Ergonomics, 32*(11), 1343-1372.

Schank, R. C. (1982). *Dynamic memory.* Cambridge, UK: Cambridge University Press.

Schroeder, D. (November 1992). Personal communication.

Shadbolt, N. R., & Wielinga, B. (1990). Knowledge-based knowledge acquisition: the next generation of support tools. In Wielinga, B., Boose, J. H., Gaines, B. R., Schreiber, G., & van Someren, M. (Eds.), *Current trends in Knowledge acquisition (Proceedings of the 4th European workshop on knowledge acquisition, EKAW-90, Amsterdam.* Amsterdam: IOS Press.

Sherwood, B. A., & Sherwood, J. N. (1984). The cT language and its uses: A modern programming tool. In Redish, E. F., & Risley, J. S. (Eds.), *The Conference on Computers in Physics Instruction Proceedings.* Redwood City, CA: Addison-Wesley. Also available as tech report UPITT/LRDC/ONR/APS-14.

Sherwood, B. A., & Sherwood, J. N. (1992). *The cT Language Manual.* Wentworth, NH: Falcon Software. The cT programming language is distributed by Falcon Software, Inc., P.O. Box 200, Wentworth, NH 03282; phone 603-764-5788, fax 603-764-9051. A site license is available for users at CMU.

Shrager, J., Hogg, T., & Huberman, B. A. (1988). A dynamical theory of the power-law of learning in problem-solving. Draft paper submitted to AAAI-88.

Siegler, R. S. (1988). Strategy choice procedures and the development of multiplication skill. *Journal of Experimental Psychology: General, 117*(3), 258-275.

Siegler, R. S., & Shrager, J. (1984). Strategy choices in addition and subtraction: How do children know what to do? In Sophian, C. (Ed.), *Origins of cognitive skills.* Hillsdale, NJ: Lawrence Erlbaum Associates.

ACM Special interest group on Artificial Intelligence. (April 1989). Special issue on: Knowledge Acquisition, *Sigart Newsletter (108).*

Special interest group on Artificial Intelligence. (1991). Special section on integrated cognitive architectures *Sigart Bulletin, 2(4),*

Special interest group on Computer-Human Interaction. (1989). Special issue on protocol analysis tools and methods, *SigChi Bulletin, 21(2),*

Simon, H. A. (1979). *Models of Thought.* New Haven, CT: Yale University Press.

Simon, H. A. (1989). *Models of Thought, Volume II.* New Haven, CT: Yale University Press.

Simon, H. S. (1990). Invariants of human behavior. *Annual Review of Psychology, 41,* 1-19.

Simon, H. A. (October 1991). Setting up research programs. Talk presented as part of the Graduate student professional seminar series: Interfacing the science and the profession, Department of Psychology, Carnegie-Mellon University.

Simon, H. A., & Newell, A. (1956). Models: Their uses and limitations. In White, L. D. (Ed.), *The state of the social sciences.* Chicago: University of Chicago Press.

statistics. S-mode is built on top of comint (the general command interpreter mode written by Olin Shivers), as an interface to S.

The latest version of S-mode is available from the Statlib email statistical software server by sending a blank message with subject "send index from S" to statlib@stat.cmu.edu, and following the directions from there. Comint is probably already available at your site, and already in your load path. If it is not, you can get it from archive.cis.ohio-state.edu (login name is anonymous, password is your real id) in directory /pub/gnu/emacs/elisp-archive/as-is/comint.el.Z. This version has been tested and works with (at least) comint-version 2.03. You probably have copies of comint.el on your system. Copies of comint are also available from ritter@cs.cmu.edu, and shivers@cs.cmu.edu.

S-mode is also available for anonymous FTP from attunga.stats.adelaide.edu.au in the directory pub/S-mode, and from the Emacs-lisp archive on archive.cis.ohio-state.edu.

The simple menu package

Updated versions (if any) of the simple-menu package used to provide the menus in S-mode, Soar-mode, and Taql-mode are available from the author or via FTP: from the elisp archive on archive.cis.ohio-state.edu as file pub/gnu/emacs/elisp-archive/interfaces/simple-menu<version>.el.Z. Iff you post me mail that you use it, I'll post you updates when they come out.