EXPER.DOC: Report about the Macintosh 'languages' experiment

by: Geert de Haan

supervised by: Thomas Green

MRC Applied Psychology Unit, Cambridge, UK.

1986

Contents:

(1) short intro (reminder)
(2) how we got there
(3) method and procedure
(4) results
(5) possible improvements
(6) how to run a session
(7) working of the program
(8) appendixes

(1) Introduction

The experiment was devised to compare the effects of distorting the semantical reference of a command-name and distorting the syntactical 'consistentness' of a command-sequence. It was the subjects task to make the screen of a Macintosh look like a series of example-screens, using a Mac-like interface to manipulate twenty pictures.

Possible actions included: insertion of a shape; deletion of a picture; copying the shape of one picture into another; exchanging two pictures; painting a picture; and patterning a picture. Each command started with the selection of a name from a menu, followed by (in two of the six commands the selection of an extra word from the second menu) clicking the mouse on one of the pictures or an icon.

In four of the commands the same extra word had to be selected, followed by clicking the mouse on a picture or the trash can. As additional commands, there were 'cancel' to cancel the effects of incorrect or unplanned actions, so that the program would again recognize new input, and there was a help facility, which presented a screen with text that briefly indicated how a command had to be specified (appendix 0).

Three different languages were used:

- a language where the name of the command represented the action (ie. insert for insertion);
- a language where the command-name indicated the number of times a certain word had to be chosen from a menu within the sequence;
- a language which lacked both of these features.

Results indicated no difference between the conditions, and if there was, a difference in the unexpected direction (ie. the hardest language seemed easiest to use). These results included: the number of screens made, and the number of errors in various categories.

(2) How we got there

In brief, it started with two questions:

- first a question following studies into markers and artificial languages. This meant something like: are we able to replicate the results of the marker-studies when the language used is a command language.
- the second question was into an empirical test of the TAG theory.

To these, two other main questions were added:

- first, TRG proposed to take a look at the specificity of command-elements, and the role that played within the confusability between two or more different commands.
- secondly, GdH came up with a suggestion to look at the relative influence of on the one hand structural consistency (as earlies versions of TAG do strongly emphasis), and on the other hand the clarity of the command-names.

The latter question was combined with some of the marker question developed within the experiment, while the former would if time permitted be reserved for a later experiment. This resulted in the following idea: build some language to be able to manipulate pictures (ie. command-name--point at something--point at something else); add some element that distorts the structure (ie. insert the selection of 'select' or 'monst' on the '--' places in some commands); and choose nonsense words as command-names.

This is the basis language, and at the same time the worst: three structures, and six names, which neither indicated the action, nor which structure was to be used where. The other two conditions consisted of this language, with an added feature to make the task easier: in one case command-names which actually named the action they invoked; in the other case command-names which indicated the number of times that (in this case) the word 'monst' occurred in the sequence.

Other conditions that received some thought were: six good names and one structure; three bad names and one structure; and six good names and one structure, but following the name one of three non-words had to be specified.

The latter was a preliminary language version for the specificity-hypothesis, while the former two, in combination with basis language and the language with the six good names could be used as a test of TAG. This was however skipped, because this could as well be done using three conditions.

(3) Method and procedure

The method used in the experiment was a between subject design, in which the subjects were assigned to the conditions in order of arrival, but with the constraint that if a subject wasn't able to reach a criterion, then the next subject would be assigned to the same condition. The criterion consisted of having produced more than one screen.

During the last three days an additional method was adopted because of GdHs impression that conditions didn't seem to differ very much. This consisted of trying to match the ages of the subjects with between the conditions; or in fact: trying to make triples of subjects belonging to different conditions, each covering roughly the same age plus or minus three years.

The subjects varied in date of birth between 1938 and 1967 with the modus about 1960. Each served for one hour.

The three conditions differed in that the command-name of the sequence served either as a marker for the action to be performed (ie. 'insert' to insert), or as a marker for the number of times a certain word had to be chosen within the sequence (ie. 'two-monst' for a sequence which required twice the selection of monst; 'one-monst'....). A final condition didn't have markers for either the structure or the action (appendix 0).

The procedure was changed after the first two subjects. At first the procedure would be:

- arrival and a short presentation;
- reading and preforming the instructions to get used to the Macintosh way of operation (appendix 1);
- reading and performing the instructions to manipulate the pictures (appendix 2);
- reading the task instructions (appendix 3), and doing the task; and
- finally filling in the forms that ask questions about the languages (appendix 4),
- followed by the 'thanks a lot' and a short explanation about the purpose of the experiment.
Thereafter, more time was spent into showing how things worked, and giving examples of the command-sequences, accompanied by verbally repeating the most important things of the instructions (ie." now I've done something wrong, and have to select cancel, for else...").

(4) Results

A total of 42 subjects took part in the experiment. From those 7 were not able to make one screen within the time given: one subject in condition one (ie. the syntax markers); one in condition two (ie. the function markers); and five in the last condition (ie. no markers). A further six subjects were not able to make more than one screen: two subjects in condition one, one in condition two, and three in condition three.

There seemed to be a remarkable difference in attitudes between more and less successful subjects: while the former subjects left an impression of being cheerful, going to attend the task as if it was an enjoyable game, the latter ones seemed to be frightened by the mere presence of a computer and often started with remarks like "oh I will never be able do the job, for I don't understand computers".

The data were first subjected to an analysis program, resulting in time and error measures. Corrected for the analysis programs' bug, the frequencies of errors and requests for 'help' are shown in table 1, in which each subject is represented as one cell.

Note that all subsequent analysis was done by hand, so minor errors could occur. The numbers between brackets represent the number of example screens made by that subject.

Table 1. Error and help request means

<table>
<thead>
<tr>
<th>Condition 1</th>
<th>Condition 2</th>
<th>Condition 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error</td>
<td>Help</td>
<td>Error</td>
</tr>
<tr>
<td>21 (3)</td>
<td>45</td>
<td>13 (2)</td>
</tr>
<tr>
<td>19 (5)</td>
<td>8 B</td>
<td>40 (3)</td>
</tr>
<tr>
<td>36 (4)</td>
<td>22 BL</td>
<td>17 (2)</td>
</tr>
<tr>
<td>22 (4)</td>
<td>45 BL</td>
<td>12 (5)</td>
</tr>
<tr>
<td>56 (3)</td>
<td>30 L</td>
<td>39 (6)</td>
</tr>
<tr>
<td>17 (3)</td>
<td>17</td>
<td>5 (2)</td>
</tr>
<tr>
<td>30 (4)</td>
<td>36 B</td>
<td>16 (3)</td>
</tr>
<tr>
<td>35 (3)</td>
<td>31 L</td>
<td>16 (5)</td>
</tr>
<tr>
<td>18 (2)</td>
<td>41</td>
<td>36 (3)</td>
</tr>
<tr>
<td>21 (3)</td>
<td>29</td>
<td>19 (2)</td>
</tr>
<tr>
<td>Totals:</td>
<td></td>
<td>275 (34)</td>
</tr>
<tr>
<td>Means:</td>
<td></td>
<td>8.08</td>
</tr>
</tbody>
</table>

Data about on criterion scores

<table>
<thead>
<tr>
<th>Condition 1</th>
<th>Condition 2</th>
<th>Condition 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 (1)</td>
<td>38 L</td>
<td>16 (1)</td>
</tr>
<tr>
<td>11 (1)</td>
<td>34 L</td>
<td>16 (1)</td>
</tr>
<tr>
<td>7 (1)</td>
<td>31 L</td>
<td>25 (1)</td>
</tr>
<tr>
<td>Totals:</td>
<td></td>
<td>293 (36)</td>
</tr>
<tr>
<td>Means:</td>
<td></td>
<td>8.14</td>
</tr>
</tbody>
</table>

As table 1 shows, the mean number of errors deviates from the expected direction and would even indicate that the by all means hardest condition was actually the most easiest. This is about the same for the number of help requests, though the low number of help requests in condition two could be explained as indicating that the use of English command names provides the user with a basis to work from, whereas the other two conditions do not.
Because of these results and the large between subject variability, a further analysis was done on the data from those subjects who produced equal or more than four screens. These subjects are marked in table one by a 'B' for best. The totals and means are shown in table 2.

Table 2. Totals and means for the 'best'.

<table>
<thead>
<tr>
<th>Condition 1</th>
<th>Condition 2</th>
<th>Condition 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error</td>
<td>Help</td>
<td>Error</td>
</tr>
<tr>
<td>Totals:</td>
<td></td>
<td>Totals:</td>
</tr>
<tr>
<td>107 (17)</td>
<td>67 (16)</td>
<td>48 (15)</td>
</tr>
<tr>
<td>Means:</td>
<td></td>
<td>Means:</td>
</tr>
<tr>
<td>6.30</td>
<td>6.53</td>
<td>4.19</td>
</tr>
</tbody>
</table>

As can be seen, this does not change the results from table one drastically, therefore as a last resort the analysis was repeated on the data falling within one standard deviation from the mean per condition (ie. giving the midmean) In effect this means that statistically half of the data are thrown away, these scores are marked 'L' from left out in table 1. Results are shown in table 3.

Table 3. Means on data within one Standard Deviation

<table>
<thead>
<tr>
<th>Condition 1</th>
<th>Condition 2</th>
<th>Condition 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.20</td>
<td>5.67</td>
<td>3.83</td>
</tr>
</tbody>
</table>

A possible way of finding an explanation for the results would be to take a closer look at error patterns. This was done, using the following error categories:

- the addition of a 'select Monst' or 'select Select' within the command sequence;
- the subtraction of the same;
- the addition and subtraction of the same (ie. put it on the wrong place);
- the addition of two of the same;
- the subtraction of two;
- incomplete command specifications;
- wrong command names; and finally
- the presence of wrong elements (ie. copy a pattern).

This way of looking at the data has two weaknesses: there is a subjective element (ie. did the subject mean to copy or to exchange), and errors could be scored in more than one way (ie. both a wrong command name and the addition of 'select Monst'). This however was avoided because it would incorporate much more ambiguity. For condition one (ie. the syntax markers), the number of 'select Monst' is checked on the name of the command, while sometimes the rest of the sequence could be in accordance to the intended action. For the other two conditions, the number of 'select Monst' is checked on the intended action, as inferred from the sequence, while sometimes the number could be in accordance with the chosen command name. Results are presented in table 4.

Table 4. Error scores.

<table>
<thead>
<tr>
<th>Condition 1</th>
<th>Condition 2</th>
<th>Condition 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition of a 'Monst'</td>
<td>26</td>
<td>35</td>
</tr>
<tr>
<td>Subtraction of a 'Monst'</td>
<td>75</td>
<td>47</td>
</tr>
<tr>
<td>Addition plus subtraction</td>
<td>10</td>
<td>14</td>
</tr>
<tr>
<td>Addition of two 'Monst'</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>Subtraction of two 'Monst'</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Incomplete commands</td>
<td>78</td>
<td>74</td>
</tr>
<tr>
<td>Wrong command name</td>
<td>28</td>
<td>1</td>
</tr>
<tr>
<td>Wrong command element</td>
<td>44</td>
<td>36</td>
</tr>
<tr>
<td>Totals:</td>
<td>273</td>
<td>213</td>
</tr>
</tbody>
</table>
The first thing that table 4. shown is that again condition three seems most easy. Secondly, table 4 shows that far the most frequent 'error' consists of incomplete commands, probably due to the subjects' change of mind. It does seem however that this analysis does produce any further interesting views on the data. As neither of the four ways of analysis yielded any usable results, it would be most safe to say that there are none. Especially as none of the theories of command language learnability is able to explain these findings.

Nevertheless, a possible explanation for the relative easy-ness of condition three might be (among other things) a difference in the way in which the language imposes a learning strategy. In this case, the difficulty of condition three might force the user to search for the 'trick', and if found the task would be much easier, while if not the task could not be fulfilled. In this way, it would explain the high number of subjects in that condition who were not able to do the task and or reach criterion.

(5) Possible improvements

One could think about a number of ways to improve the experiment, even though at first all seemed to be perfect. A first improvement is pointed to be the error data: many of the subjects in condition one didn't seem to have understood that the command names were markers for the number of 'select'. This could easily be solved by just telling them.

A second improvement could be to allow for more Time: none of the subjects seemed even distantly able to make the ten screens as was at first proposed. This means that not only was there far less data collected than would be possible, but also only data from the first learning stages. Which in general show far more variability than data from later stages, and might obscure condition variability.

Other ways to reduce or cope with large differences between subjects are: post hoc grouping of subjects on measures as perceived difficulty and the like. This however requires more subjects with all related problems. One might think of choosing more equal subject, and for instance test only undergraduates etc. Changes can be made to the contents of the three conditions, to allow for more between condition variance.

These changes might involve constraining the number of times that the same command may be used, so as to force for a larger variability in command choice than was found (some subjects never even used the commands copy and exchange, as the same result could be attained by using insert and delete). One might also think about using larger, and maybe even real life languages. In the latter case, it would of course become impossible to test the effects of markers (unless some macro facility exists), but there still could be a test on the TAG theory. A final way to improve the experiment is the use of additional data, like verbal protocols, to enable stratification of subjects' data, as well as forcing the subjects to explicitly think about how to do what.

(6) How to run a session

The program 'newdraw' runs from the Pascal interpreter disc. It is useful to have a second disc for the data, as space on the Pascal disk is limited. Open the program in the usual way and select 'run' from a menu. Now, the program will ask for a help file and an output file. After doing this, the program will build up the screen and prompts for a click of the mouse to start. By then, only the 'cancel' command will work, and it is only after about twenty actions (click the mouse or select from a menu), and a click of the mouse outside a box (asking for more practise or continue) that the screen is rebuild and all the commands work.

After each correct command, the program will check if the target display has been reached; if so the codes and times of the actions will be outputted to file after which the screen is rebuild.... After five (or six ?) screens or about fifty minutes, the program will stop. Each trial or screen can be ended by typing an 'X', and the session or program can be ended by typing a paragraph sign (top left of the keyboard).
Whenever an error has occurred, except for clicking the mouse at a place where there is nothing to select or a menu opening not resulting in a choice, then the parser will reject all further input until the next 'cancel'. The choice of 'help' results in the display of the helpfile text, minus the first line, and until the next click of the mouse.

Finally, when a command, except help and cancel has been chosen either successful or unsuccessful then a 'nauty boy' message is displayed, and further invocations of that command will only be possible after the choice of another command.

Data can be analysed by the program 'anal', which is to be found on the second Pascal disc of TML compiled Pascal. This program counts how many times a certain action (ie. chose command 'xyz') was done, how long it took to since the previous action, means and variances. This is outputted into a datafile, while another user selectable file will contain the errors made.

Beware for the fact that this program contains a bug: when the last command of a trial has less than four elements, then that command is wrongly considered as an error!

(7) Working of the program

Every action by the user is recorded, and after a trial outputted to the output file in the following way: index since begin of trial, a code of the action, a number of what was acted upon, and time in sixty Hertz after begin of trial. This recording is completely independent of the parsing process. The actions are coded in the following way:

1: a nonsense action;
2: choice from the options menu;
3: choice from the command menu;
4: click on an icon;
5: click on a picture;
6: click on the trash;
7: type a key;
8: click the mouse after a 'click the mouse...' prompt.

Where the action acted upon is coded like the following:

after '1': 0: no menu choice, 2: click in the menubar, and 3: click between the icons;
after '2' and '3': the number of the item, from top to down;
after '4' or '5': the number of the icon or picture, from left to right, and top to down;
after '7': the hexadecimal keynumber.

The first line, outputted after every trial consists of five zero's. After each action, a pre-processor puts a code in an array, for parsing later on, and depending on the system or language, it adds or removes elements. The pre-processor also checks the number of times, a certain command has been chosen. The system is known, because the first line in the help files contains it's number (1..5).

When the number of actions could be appropriate for a command, the parser is called, which checks the codes that are by now the same for each language. If parsing was successful, or 'cancel' was selected, then the array pointer is reset to one, and after no-success the pointer is incremented until it is pointing at the last element in the array, which is outside of the reach of the parser (ie. in case of error).
After successful parsing, the data in an array containing the data describing the pictures is changed, and compared to the target data in an other array.
After a match or one of the special keys, a trial ends.

The target and picture data consist of: the presence of a picture, and if so the bended-ness of it's corners, and the number of the icon that is similar in colour or pattern (minus three). N.B. It might be most easy to use this program itself to build different target files, than the file 'pictarg', currently used.
Appendix 0: the conditions

(1) INSERT: select 'No-Monst'--click on a shape--click on an empty place.

DELETE: select 'Two-Monst'--select 'Monst'--click on a place--select 'Monst'--click on the trash-can.

COPY: select 'One-Monst'--click on a picture--select 'Monst'--click on a place.

EXCHANGE: select 'Two-Monst'--select 'Monst'--click on a place--select 'Monst'--click on another place.

PAINT: select 'No-Monst'--click on a colour--click on a picture.

PATTERN: select 'One-Monst'--click on a pattern--select 'Monst'--click on a white picture.

(2) As (1) with the words: INSERT, DELETE, COPY, EXCHANGE, PAINT and PATTERN instead of NO-MONST, TWO-MONST....ONE MONST.

(3) As (1) with the words: BLARK, RAGOLE, RAGOLE, DAPSE, DAPSE and BLARK instead of NO-MONST, TWO-MONST....ONE-MONST.

Note: The actions invoked by the commands are: insert one of three shapes into an empty place; delete a place; copy the form of one picture into another picture or place; exchange the contents of two place; paint a picture; and give a white picture a pattern.