Impact of semantic association on information recall performance

(part 1 of 2)

Experiment performed at the 2012 ACCU Conference
First published in CVu vol. 2x no. x

Derek M. Jones
derek@knosof.co.uk

1 Introduction

Comprehending source code involves reading it to obtain information which may only need to be remembered for a short period of time or may need to be remembered over a longer period.

One of the first major discoveries in experimental psychology was of a feature of human memory that has become commonly known as *short term memory*. People are able to temporarily retain a small amount of information in memory whose accuracy quickly degrades unless an effort is made to 'refresh' it, and the information is easily overwritten by new information.

This article reports on an experiment carried out during the 2012 ACCU conference that investigated the impact of a limited capacity short term memory on subjects' performance in commonly occurring tasks that occur when comprehending short sequences of code. The tasks involved subjects ability to recall some of the variable names appearing in the conditional expression of two if-statements and to correctly deduce which arm of an if-statement is executed.

The experiment is derived from and takes account of results from previous ACCU conference experiments. In particular the 2004 experiment[4] which also asked subjects to select which arm of an if-statement they thought would be executed and the 2007 experiment[5] which found that developers appeared to make use of identifier names to make operator precedence decisions. It is hoped that this study will provide information on the effect that identifier names have on developer performance during program comprehension.

Few developers appreciate how short the *short* in short term memory actually is; its capacity has been found to correspond to approximately two seconds worth of sound, with some people have less capacity and some more. This is sufficient to hold information on a few statements at most. The analysis of the results from this study will hopefully highlight to developers the consequences of short term memory limitations on their code comprehension performance.

This article is split into two parts, the first (this one) provides general background on the study and discusses the results of the if-statement memory/recall problem, while part two discusses selecting executed arm of if-statement problem.

1.1 The hypothesis

Measurements of source code[5] have found that some English words usually appear in variables that are the operands of bitwise or logical operators (e.g., flag) while some English words usually appear in variables that are the operands of arithmetic operators (e.g., count). The 2007 ACCU experiment found that developers appeared to make use of the occurrence of these words when asked to make binary operator precedence decisions.

The hypothesis tested by this experiment are that:

1. developers are less likely to confuse identifiers appearing as operands in an expression if the operand names are consistent with common usage for the corresponding binary operator,
2. developers are less likely to be able to correctly recall the identifiers appearing as operands in an expression (e.g., not remember their name or use a different name) when one or more identifiers are not consistent with common usage for the corresponding binary operator.

It is difficult to evaluate whether nonconsistent operand names will be less likely to be recalled because they fail to match a known pattern or be more likely to be recalled because they stand out as being different than expected (which may result in a reduction in resources used to remember consistent names leading to a greater chance of confusion among consistent names).

2 Characteristics of human memory

Models of human memory often divide it into two basic systems, short term memory (the term *working memory* is technically more correct, while short term memory can be applied to any one of several memory subsystems and long term memory). This two subsystem model is something of an idealization in that there is not a sharp boundary between short and long term memory; there is a gradual transition between them.

The *short term memory* subsystems are a gateway through which all input memory data input must pass. They have a very limited capacity and because new information is constantly streaming through them, a particular piece of information rarely remains in it for very long. Information in short term memory is either quickly lost or stored in another, longer term, memory subsystem.

Declarative memory is a long-term memory that has a huge capacity (its bounds are not yet known) and holds information on facts and events. Two components of declarative memory of interest to the discussion here are episodic and semantic memory. Episodic memory[2] is a past-oriented memory system concerned with *remembering*, while semantic memory is a present-oriented memory system concerned with *knowing*.

While some of the characteristics of human memory (e.g., forgetting) are often criticized, they do provide useful functionality. People who can readily remember and later accurately recall information report that their conscious thoughts are repeatedly interrupted by *unforgotten* information.[8] It would make sense for human memory to be optimized for the information recall demands that most commonly occur in everyday life and various studies[1] appear to confirm this evolutionary priority; there seems to be an exponential decay in the likelihood that information will be needed again after it is first encountered and this is the rate at which information is *lost* from memory. Thus the likely need for information and the rate at which it is forgotten decrease in the same way.

Studies have found that practicing the remembering and recalling of information does not lead to a general improved performance on remember/recall tasks (i.e., in the sense that exercise can lead to muscle growth, leading to increased strength). However, extended practice does provide people with the opportunity to try out different information memorization strategies and there are cases where people have discovered strategies (e.g., using mnemonics having associations with a persons existing network of memories) that lead to significant performance improvements for particular kinds of information.

3 Experimental setup

The experiment was run by your author during a 40 minute lunch time session at the 2012 ACCU conference (www.accu.org) held in Oxford, UK. Approximately 370 people attended the conference, 22 (5.9%) of whom took part in the experiment. Subjects were given a brief introduction to the experiment, during which they filled in background information about themselves, and then spent 20 minutes answering problems. All subjects volunteered their time and were anonymous.

3.1 The problem to be solved

Each problem seen by subjects was intended to involve memory processes that operate over a time frame of approximately 30 seconds. It was expected that the characteristics of short term memory would have a significant impact on subjects performance within this time frame.

To obtain statistically reliable results a large number of answers to related problems are needed. Therefore it is necessary to create lots of variations to the same underlying problem, also problems should not be too easy or too difficult (if all subjects answered all question correctly, or all incorrectly, no useful information would be obtained). The intent is that the study thus has some claim to being *ecologically valid* (i.e., the behavior in the experimental situation is characteristic of a real life environment).

The following is an example of one of the problems seen by subjects. One side of a sheet of paper contained three assignment statements while the second side of the same sheet contained the if statements and a table to hold the recalled information. A series of X's were written on the second side to ensure that subjects could not see through to identifiers and values appearing on the other side of the sheet. Each subject received a stapled set of sheets containing the instructions and 38 problems (one per sheet of paper).

```
---------------------- first side of sheet starts here ----------------------
if ( (cars + lorries) == amount )
   x = 1;
if ( (settings | register) == final )
   x = 0;
---------------------- second side of sheet starts here ---------------------
```

```
if ((e > a) && (u < a))
   if (u > e)

      ..............
   else
      ..............
if ( (__________________  ___  __________________) == amount )
   x = 1;
if ( (__________________  ___  __________________) == final )
   x = 0;
```

In practice software developers do not make a remember/not remember decision, there is always the opportunity to refer back to previously read information. The selection remember/*would refer back* more accurately reflects the decision made by developers.

The instructions given to subjects followed that commonly used in memory related experiments. Subjects see the material to be remembered, then perform an unrelated task (chosen to last long enough for the contents of short term memory to have degraded), and then have to recall the previously seen information. The sequence "remember->unrelated task->recall" has an obvious parallel in source code comprehension; i.e., "sequence of assignments->conditional test->use of identifiers previously assigned to". The following written instructions were given to subjects:

Instructions

This is not a race and there are no prizes for providing answers to all questions. Please work at a rate you might go at while reading source code.

The task consists of remembering the names of four different variables, plus two binary operators, and later recalling and writing them down. The variable names and operators are contained in two if-statements appearing on one side of a sheet of paper and your response needs to be given on the other side of the same sheet of paper.

1. *Read the conditional expression contained in each if-statement as you would when carefully reading code in a function definition at work.*
2. *Turn the sheet of paper over. Please do **NOT** look at the if-statements you have just seen again, i.e., once a page has been turned it stays turned.*
3. *For the nested if-statement appearing at the top of the new page, please place a cross in those arms of the inner if-statement that you think could be executed. This could be only one arm or could be both arms of the if-statement.*
4. *You are now asked to recall some of the variables and operators seen on the previous page.*
   - *if you can remember the name of any variable write it in the corresponding underlined portion of the conditional expression.*
   - *if for one or more variable names or operators, you feel that in a real life code reading situation you would refer back to the previously seen if-statement, write "refer back" for the corresponding variable name or operator,*

*If you do complete all the questions do **NOT** go back and correct any of your previous answers.*

3.2 Generating the problem

Each problem contained two parts, the to be remembered information question and the which arm will be executed question. The generation of the to be remembered information is covered here and generation of the other question is covered in the second part of the article.

The to be remembered information was contained within an if-statement having the following form:

```
1  if ( (cars + lorries) == amount )
2     x = 1;
```

with the expression to the left of the equality operation (either a `==` or `!=`) always containing a single binary operator and its two operands, and the expression to the right always containing a single identifier.
The binary operator was either arithmetic (one of `-` or `*`) or bitwise (one of `|` or `&`).
Each operand is an identifier consisting of a word consistent with or not-consistent with the operator, where consistency is defined as appearing in an identifier that commonly appeared as an operand of one kind of operator and rarely appearing as the operand of the other kind of operator.
In the following discussion C denotes an operand whose name is consistent with the associated operator and N an operand whose name is not consistent; a is an arithmetic operator and b is a bitwise operator. So CaC is an expression containing an arithmetic operator and two consistently named operands and $\frac{CaC}{NbC}$ is a pair of expressions.
The to be remembered information consisted of a pair of expressions, each containing two identifiers and an operator. It was thought likely that subjects would sometimes give answers where the order of operands within an expression would be swapped and that swapping might also occur for operands between expressions within a pair. The generation of the expression pairs was balanced so that any operand swapping could be analysed.
If recall performance is better when operand names are consistent and worse when operand names are not consistent (compared to neutral operand names), then between expression operand swapping is less likely to occur in the pair $\frac{CaC}{CbC}$ than in the pair $\frac{CaC}{CaC}$; the pair $\frac{CaN}{CbN}$ would be expected to be the most likely to produce answers containing a swap between expressions.
The following are the operand combination patterns generated for this experiment:

- The four possible operand/operator combinations where all of the operand names are consistent with their corresponding operator, i.e., $\frac{CaC}{CbC}$, $\frac{CaC}{CaC}$, $\frac{CbC}{CbC}$ and $\frac{CbC}{CaC}$,
- the 16 possible operand/operator combinations where one of the operand names is not consistent with their corresponding operator, i.e., $\frac{NaC}{CbC}$, $\frac{CaN}{CbC}$, $\frac{CaC}{NbC}$, $\frac{CaC}{CbN}$ and so on for the three other possible operator combinations,
- 24 of the possible operand/operator combinations where two of the operand names are not consistent with their corresponding operator. The 24 combinations contained three sets, eight where the two nonconsistent operand names were horizontally aligned (i.e., both appeared in the same expression, e.g., $\frac{NaN}{CbC}$), eight where the two nonconsistent names were vertically aligned (i.e., one in each expression and both in the same operand position, e.g., $\frac{NaC}{NbC}$) and eight where the two nonconsistent names were diagonally aligned (i.e., one in each expression and their appeared in different operand positions, e.g., $\frac{NaC}{CbN}$).

Table .1: The identifier names commonly occurring in the operands of the corresponding binary operator; anonymous names commonly occurring in operands to all binary operators.

Bitwise	Arithmetic	Anonymous
flag	offset	value
state	rate	field
bits	size	temp
options	length	i
status	count	data
mask	maximum	current
active	minimum	id
done	width	buf
started	index	
shift	height	
success		
reset		
0x17	25	

It was decided to concentrate the analysis on the above 48 combinations and not to investigate operand/operator combinations where three of the operand names are not consistent with their corresponding operator.

Measurements of source code from a previous experiment[5] were used to obtain three lists of English words, one containing words that primarily occurred in variables appearing as the operands of bitwise or logical operators, one containing words primarily occurred in variables appearing as the operands of arithmetic operands and the third containing words or letter sequences commonly occurring with all kinds of operators. The literals `0x17` and `25` were included so that the generated expressions were more representative of real code (which includes lots of literals). See Table .1.

The problems and associated page layout were automatically generated using a shell script and various awk programs to generate troff, which in turn generated postscript and this was printed to paper.

The operand combination pattern was randomly selected (the ratio 30% all operands consistent, 40% one operand not consistent, 30% two operands not consistent was used) and identifiers/constant from the required set were randomly chosen, once used the identifier/constant was not used again until all other identifiers in that set had been used. The order of the two statements (for each problem) was also randomized. The source code is available on the experiments web page.[7]

4 Threats to validity

Experience shows that software developers are continually on the look out for ways to reduce the effort needed to solve the problems they are faced with. Because each of the problems seen by subjects in this study has the same structure it is possible that subjects will detect what they believe to be a pattern in the problems and then attempt to use this perceived information to improve their performance.

The context in which an expression occurs in real life code reading situations may provide additional semantic assistance in recalling identifier names and this assistance is not present in this experimental setting. One subject wrote about the lack of application domain context on the Comments page.

Several subjects commented that they started out with no memorization strategies and then adopted one after answering several questions. It is possible that this change of strategy may have had an effect on the distribution of the kinds of answers they gave.

While the kind of problems used commonly occur during program comprehension, the mode of working (i.e., paper and pencil) does not. Source code is invariably read within an editor and viewing is controlled via a keyboard or mouse. Referring back to previously seen information (e.g., expressions in statements) requires pressing keys (or using a mouse). Having located the sought information more hand movements (i.e., key pressing or mouse movements) are needed to return to the original context. In this study subjects only needed to leave the answer blank, or write "?" or "refer back" to indicate that they *would refer back* to locate the information. The cognitive effort needed for these actions is likely to be less than would be needed to actually refer back. Studies have found[3] that subjects make cost/benefit decisions when deciding whether to use the existing contents of memory (which may be unreliable) or to invest effort in relocating information in the physical world. It is possible that in some cases subjects ticked the *would refer back* option when in a real life situation they would have used the contents of their memory rather than expending the effort to actually refer back.

Subjects were asked to specify *would refer back* as the answer if they felt that in a real life code reading situation they would refer back to the previously seen if-statement. Some subjects wrote "?" for the operator or operand name, while others wrote nothing. All three cases, "refer back", "?" and blank were treated as being equivalent.

Two of the words used (i.e., `bits` and `options`) were the plural form of the word. In a few cases answers failed to include the final letter s and sometimes the word `flag` with a final letter s was given as an answer. An end of word s was added/removed from an answer if this resulted in it becoming a correct answer (around 10 instances). In some cases it was not clear from the written answer what the final letters of some words were and they were assumed to be those needed to complete the corresponding word appearing in the question.

One subject wrote in the comments that they were not a native English speaker another that they were

dyslexic.

One subject wrote in the comments that they answered the recall problem before answering the nested if statement problem. This subject had the 4th highest percentage of correct answers (see subject 19 in Figure .1; subject 1 in the raw data).

5 Results

It was hoped that at least 30 people (on the day 22) would volunteer to take part in the experiment and it was estimated that each subject would be able to answer 25 problems (on the day 19.6, sd 8.1) in 20-30 minutes (on the day 20 minutes). Based on these estimates the experiment would produce 750 (on the day 430) pairs of if statements were remembered/recalled.

The raw results for each subject and the scripts used to process them are available on the experiment's web page.[7]

5.1 Expected direction of behaviors

- When all operand names are consistent with its operator subjects will be less likely to make between expression mistakes for $\frac{CaC}{CbC}$ compared to $\frac{CaC}{CaC}$ because any swapped operands would not be consistent with their operators; in the second form any swapping of operands is consistent with the operators,
- fewer correct answers when an if statement pair contains one or more operand names that are not consistent with their operator,

There is no expectation of behavior bias for answers involving operator recall.

5.2 Actual results

Most of the results presented here take the form of 2-dimensional contingency tables (see Table .2). Given the hypothesis that the rows of the table are drawn from the same distribution (i.e., the same underlying process) a Pearson chi-squared test can be used to calculate the p-value for this hypothesis. A very low p-value (e.g., 0.01 or 1 in 100) is taken to imply that the rows are drawn from different distributions (i.e., there is a difference in the processes involved).

A low p-value tells us that randomly generated rows are unlikely to be this different unless they are generated by different processes.

How big an effect is the difference between rows? Cramer's V is used as a measure of effect size, its value varies between 0 (no effect) to 1 (complete effect).

An awk script was used to convert the answers to a comma separated list which was then processed using the R statistical package. The R function `assocstats` from the package `vcd` was used to obtain the values for the Pearson chi-squared test and Cramer's V.

Incorrect answers for the operand name are broken down into those that are wrong and those that are the name of a different operand appearing in the question (i.e., one of three possible other names). Writing an expression containing an operand name that is wrong might result in a compile time error while writing code an expression with operand names swapped is much less likely to generate a diagnostic from the compiler.

Table .2: Number of operand answers that are wrong, in various operand positions or subject *would refer back*.

	Wrong	1	2	3	4	Would refer back
1st operand	9	372	7	7	0	36
2nd operand	16	7	350	1	8	49
3rd operand	18	7	1	320	15	70
4th operand	5	1	7	10	331	77

Table .2 summarizes the operand answers. The rows are the operand numbers (first operand of first expression is number 1, second operand is 2, first operand of second expression is number 3 and its second operand is 4).

The first column is the total number of wrong answers for each operand position The next four columns give the total number of answers appearing in different operand positions, for instance if the first operand in the first expression is `flag` and the answer gives this name as the second operand of the first expression then column 2/row 1 is incremented by one. The last column is the total number of *would refer back* answers for that operand.

5.2.1 All operand names consistent

Table .3 contains the total number of answers in various categories for questions where all operands have names that are consistent with their corresponding operator; if an operand name appearing in the question appeared in the answer in an incorrect position it was counted as a *swapped* operand (column 4 in the table).

Table .3: Total, for all subjects (percentage for each row in brackets), of wrong, correct, *would refer back* and swapped operand answers for the four combinations of operator ordering where all operand names are consistent with their corresponding operator.

Operator pairs	Wrong (%)	Correct (%)	Would refer back (%)	Swapped (%)
$\frac{a}{a}$	4 (3)	99 (82)	13 (10)	4 (3)
$\frac{b}{b}$	5 (6)	103 (83)	10 (8)	6 (4)
$\frac{b}{a}$	2 (1)	113 (76)	29 (19)	4 (2)
$\frac{a}{b}$	3 (2)	115 (77)	25 (16)	5 (3)

To test hypothesis 1 the $\frac{CaC}{CaC}$ and $\frac{CbC}{CbC}$ totals are added together and also the totals from $\frac{CaC}{CbC}$ and $\frac{CBC}{CaC}$ are added, giving two rows. A Pearson chi-squared test on these two rows produces p-value=0.016, i.e., the rows are likely to be drawn from different distributions (1 in 62 chance of this result occurring if they are drawn from the same distribution).

Hypothesis 1 predicts that the row totals are drawn from different distributions, but predicts a direction of difference that is the opposite of that seen in the answers; in practice there were more *would refer back* answers when the operators are different (the hypothesis predicts less).

The value of Cramer's V is 0.138, a very small effect (while *would refer back* answers almost doubled the overall contribution of that particular kind of answer is much smaller than Correct answers).

5.2.2 Operand names not consistent

Table .4 contains the total number of answers of various kinds for questions where all operand names are consistent, one operand has a name that is not consistent with its corresponding operator and two operands have names that are not consistent. The row percentages are remarkable similar and a Pearson chi-squared test gives p-values that reflect this fact; the consistency of operand/operator naming does not have any impact on subject recall performance (i.e., no support for hypothesis 2).

Table .4: Total, for all subjects, of wrong, correct, *would refer back* and swapped answers for the 8 cases where all operand names are consistent, the 16 cases where one operand has a name that is not consistent with its corresponding operator and 24 of the cases where two operands have names that are not consistent.

Operand kind	Wrong (%)	Correct (%)	Would refer back (%)	Swapped (%)
All consistent	14 (2)	430 (79)	77 (14)	19 (3)
One not consistent	16 (2)	552 (79)	87 (12)	37 (5)
Two not consistent	20 (4)	387 (79)	70 (14)	11 (2)

5.3 Impact of operand position

What impact does the relative position of the operand have on recall performance (i.e., first or second in the expression, or appearing in the first or second expression)?

Table .5: Total, for all subjects, of wrong, correct, *would refer back* and swapped answers all questions, first row the first operand of an expression and second row the second operand of an expression.

Operand kind	Wrong (%)	Correct (%)	Would refer back (%)	Swapped (%)
1st operand	27 (3)	692 (80)	106 (12)	37 (4)
2nd operand	21 (2)	681 (79)	126 (14)	34 (3)

Table .5 breaks down the total answer count by operand position, either to the left of the operator (first row) or the right (second row). A Pearson chi-squared test gives p-value=0.44, which strongly suggests that the row values are drawn from the same distribution (i.e., any difference in totals is likely to be the result of random variation).

Table .6: Total, for all subjects, of wrong, correct, *would refer back* and swapped answers all questions, first row the first expression and second row the second expression.

Operand kind	Wrong (%)	Correct (%)	Would refer back (%)	Swapped (%)
1st expression	25 (2)	722 (83)	85 (9)	30 (3)
2nd expression	23 (2)	651 (75)	147 (17)	41 (4)

Table .6 breaks down the total answer count by expression position, either to the first expression (first row) or the second expression (second row). A Pearson chi-squared test gives p-value=6.4e-05, which very strongly suggests that the row values are drawn from different distribution.
The value of Cramer's V is 0.113, a small effect (for the same reasons given above).
Does operator recall performance also vary between expressions?

Table .7: Total, for all subjects, of wrong, correct, *would refer back* and swapped operator answers for the first and second expression.

	Wrong (%)	Correct (%)	Would refer back (%)	Swapped (%)
1st operator	9 (2)	378 (87)	32 (7)	12 (2)
2nd operator	16 (3)	347 (80)	55 (12)	13 (3)

Table .7 summarizes the operator answers. A Pearson chi-squared test gives p-value=0.024 (1 in 42 chance of being drawn from the same distribution). There is a greater chance of a *would refer back* answer being given for an operator in the second expression, compared to the first.

5.4 Individual subject performance

Table .8 lists the mean and standard deviation of subject operand recall performance.

Table .8: Mean and standard deviation of the various kinds of subject answers (normalised as a percentage), i.e., mean and sd of values in Figure .1, plus the mean number of questions answered by subjects.

	Wrong	Correct	Would refer back	Swapped	Questions answered
mean	3.3%	75.8%	16.3%	4.5%	19.6
sd	3.1	19.4	15.3	4.0	8.1

Figure .1 is a plot of the total number of operand answers for each subject as a percentage of all answers they gave, the subjects are ordered in increasing percentage of correct answers. There appears to be a negative correlation between percentage of correct answers and *would refer back* answers, the Pearson correlation coefficient is -0.96 (95% confidence interval: -0.98 -0.91, p-value=8.6e-13). The only other strong correlation is between percentage of correct answers and swapped operand names, -0.76 (95% confidence interval: -0.89 -0.50, p-value = 4.1e-05).

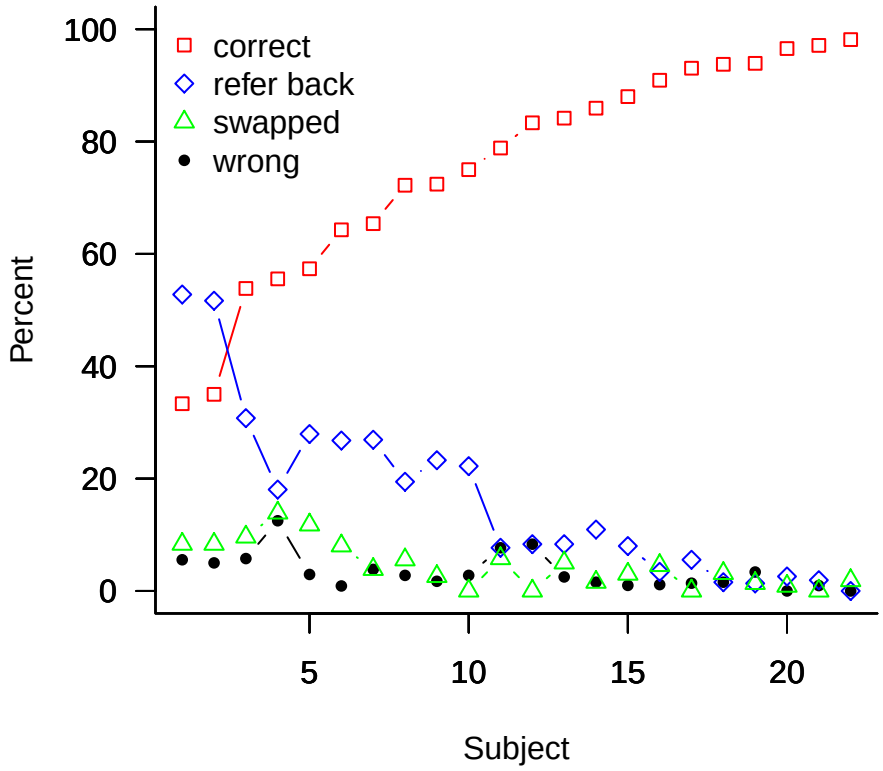

Figure .1: The percentage of correct (square box), *would refer back* (diamond), swapped (triangle) and incorrect operand answers (bullet) for each subject. Subjects are ordered by percentage of correct answers given.

As in previous memory experiments[6] there were a few subjects who give a very high percentage of *would refer back* answers.

After correct answers *would refer back* is almost always the next most common answer, followed by operand names being swapped with another operand and then wrong answers.

The wide variation in individual subject performance suggests that the experimental design aim of creating a problem whose solution stretched the limits of subjects' short term memory capacity was achieved. Had the problems required more or less short term memory capacity then it is likely that the variations in subjects performance would have been smaller (i.e., subjects would have been likely to have given fewer or a greater number of wrong or *would refer back* answers).

6 Discussion

The number of questions answered by subjects (19.6) is similar to the number of questions answered in 2004 (22.7) when the same nested conditional if statement was used between the remember/recall problem.

The largest effect found is that subjects' correct recall performance is lower on the second expression, compared to the first.

Experiments have shown that when subjects are asked to remember a list of words there is a *primacy effect*

(i.e., they have better recall performance for items at the start of a list) and a *recency effect* (i.e., they have better recall performance for items at the end of a list).[?]

Is there a serial order in the if statement problem? With only two operands and two expressions every operand could be said to be either at the start or end of a list. Also the order in which subjects read and possible reread the information to be remembered is not known.

The order in which subjects wrote their answers is likely to be first expression followed by second expression. Perhaps the time taken writing the answer to the first expression caused information on the second expression to be lost from memory. Depending on the real life activity being performed this delay may or may not occur (i.e., a delay occurs when writing code but little delay need occur when a developer is processing code in their head)

In some contexts swapping an operand name is a mistake that could result in a fault being introduced into the code (e.g., if the operator involved was subtract or divide) and in other contexts swapping the operands of an operator is unlikely to result in a fault (e.g., the addition operator).

7 Conclusion

Subject operand recall performance, for a binary operator and the names of its two operands appearing in the control expressions of two consecutive if statements, was found to depend on:

- Whether the subexpression containing the operands appears in the first or second if statement (more *would refer back* answers are given for the second; not proposed as a hypotheses),
- whether the two if statement subexpressions, when they both have all operand names consistent with their respective operator, contains the same or different operators (more *would refer back* answers are given when the operators are different; the opposite behavior predicted by hypothesis 1).

No statistically significant difference in operand recall performance was found between all operand names being consistently named, one operand not consistently named or two operands not consistently named (no evidence for hypothesis 2).

8 Further reading

For a readable introduction to human memory see “Essentials of Human Memory” by Alan D. Baddeley. A more advanced introduction is given in “Learning and Memory” by John R. Anderson. An excellent introduction to many of the cognitive issues that software developers encounter is given in “Thinking, Problem Solving, Cognition” by Richard E. Mayer.

9 Acknowledgments

The author wishes to thank everybody who volunteered their time to take part in the experiment, the ACCU for making a conference slot available in which to run it and Roger Orr for encouraging conference attendees to take part.

References

1. J. R. Anderson and R. Milson. Human memory: An adaptive perspective. *Psychological Review*, 96(4):703–719, 1989.

2. A. Baddeley, M. Conway, and J. Aggleton. *Episodic Memory: New Directions in Research*. Oxford University Press, 2002.

3. W.-T. Fu and W. D. Gray. Memory versus perceptual-motor tradeoffs in a blocks world task. In *Proceedings of the Twenty-second Annual Conference of the Cognitive Science Society*, pages 154–159, Hillsdale, NJ, 2000. Erlbaum.

4. R. N. A. Henson. *Short-term Memory for Serial Order*. PhD thesis, University of Cambridge, Nov. 1996.

5. D. M. Jones. Experimental data and scripts for short sequence of assignment statements study. http://www.knosof.co.uk/cbook/accu04.html, 2004.

6. D. M. Jones. Operand names influence operator precedence decisions. *C Vu*, 20(1):5–11, Feb. 2008.

7. D. M. Jones. Effects of risk attitude on recall of assignment statements. *C Vu*, 23(6):19–22, Jan. 2012.

8. D. M. Jones. Experimental data and scripts for impact of semantic association on information recall performance. http://www.knosof.co.uk/dev-experiment/accu12.html, 2012.

9. A. R. Luria. *The mind of a mnemonist*. Harvard University Press, 1986.