Memory for a short sequence of assignment statements (part 1 of 2)

Experiment performed at the 2004 ACCU Conference
First published in CVu vol. 16 no. 6

Derek M. Jones
derek@knosof.co.uk

1 Introduction

The process of comprehending source code often involves reading some statements on a line by line basis. Some of the information read only needs to be remembered for a short period of time, while other information needs to be remembered over a longer period.

This article reports on an experiment carried out during the 2004 ACCU conference that investigated the consequences of a limited capacity short term memory on subjects performance in some of the tasks needed to comprehend short sequences of code. The source code used contained two commonly occurring constructs, assignment statements and if statements. Subjects ability to recall the numeric values assigned to particular identifiers and to correctly deduce which arm of an if statement is executed were used as measures of their performance.

It is hoped that this study will provide information on the impact that different kinds of identifier character sequences have on the cognitive resources needed during program comprehension. Also, advantage was taken of the if statements used in the experiment to try and duplicate the pattern of subject performance seen in some studies of human reasoning. The results seen in some of these studies suggest that the ordering of operands in a pair of relational expressions has an impact on peoples performance in evaluating it.

Few developers appreciate how short the *short* in short term memory actually is. It only has the capacity to hold information on a few statements at most. It is hoped that the results of this study will bring home 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 assignment problem, while part two discusses the if statement results.

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, see Figure 1) 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.

The phonological loop, Figure 1, which can hold approximately 2 seconds worth of sound is the primary short term memory system of interest in this study. The sound-bite that can be held by the phonological loop correspondes to 7±2 digits[7] spoken in English (the variation is highly correlated with differences in the rate at which people speak; faster speakers can remember more) and 9.9 digits spoken in Chinese (i.e., the spoken form of the number words is shorter).

While some of the characteristics of human memory (e.g., forgetting) are often criticized, they can provide useful functionality. People who can readily remember and later accurately recall information report that their conscious thoughts are repeatedly interrupted by *unforgotten* information.[12] It would make sense for human memory to be optimized for the information recall demands that frequently occur in everyday life and various studies[1] appear to confirm this evolutionary priority. For instance, forgetting is not necessarily the result of a poorly designed memory system. Studies[1] have found an exponential decay in the likelihood that information will be needed after a given period of time has elapsed since it was first encountered and that the rate at which information is *lost* from memory also has an exponential form. 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

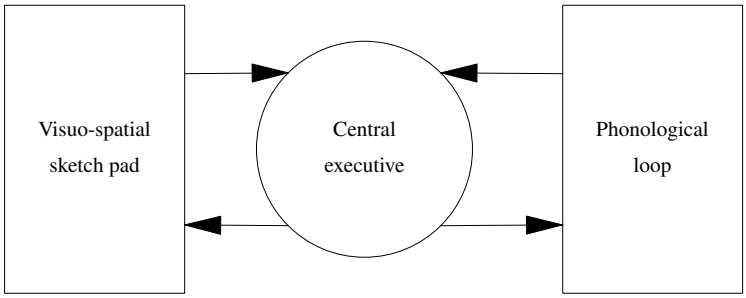

Figure 1: Model of working memory. The phonological loop can hold approximately 2 seconds worth of sound, while the visio-spatial sketch pad holds a visual image that degrades within about 1.5 seconds. From Baddeley.[2]

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 (e.g., the world record times of long distance runners)

It could be claimed that the underlying problem is one of using of a computing platform (i.e., the human brain/mind) for a purpose for which it was not designed.

3 Human reasoning

A commonly used model of the human mind is that of a very powerful computer with the reasoning faculties based on mathematical logical. George Boole (after whom the term *boolean* is named) titled his book[3] "*An investigation of the laws of thought on which are founded the mathematical theories of logic and probabilities*". However, the results of many studies[13] are not consistent with this model of human reasoning.

Studies of various kinds of reasoning, involving logical statements, have discovered patterns in subjects performance that are believed to be characteristic of how people solve reasoning problems. If the performance of subjects deducing the behavior of source code if statements also exhibit patterns (e.g., differing numbers of errors made for different representations of the same logical condition), it may be possible to use this information to reduce the number of errors made by developers in comprehending source code. This topic is covered in detail in part 3 of this article, which discusses subject if statements evaluation performance.

4 Experimental setup

The experiment was run by your author during one 45 minute sessions of the 2004 ACCU conference held in Oxford, UK. Approximately 300 people attended the conference, 40 (13%) of whom took part in the experiment. Subjects were given a brief introduction to the experiment, during which they filled out background information about themselves, and they then spent 20 minutes working through the problems. All subjects volunteered their time and were anonymous.

One subject wrote at the start of the response sheets that they were dyslexic. Given that no other medical information was known about other subjects there was no reason to treat this subject any differently than these other subjects. Your author knows one other person working in software engineering who is dyslexic, who was not at the ACCU conference, but does not know of any research on the performance impact dyslexia has on software development activities.

4.1 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 conclusions answers to a large number of related problems would be needed. Therefore it had to be possible to create a number of variations on the same underlying problem. Problems

also had to be created that were not too easy or too difficult (if all subjects answered all question correctly, or all incorrectly, no useful information would be obtained).
By using various rules of thumb (e.g., short term memory can contain two seconds worth of sound), simplifying assumptions (discussed below), and practicing on himself, your author settled on a problem that involved recalling information about three assignment statements and selecting the appropriate arm of an if statement. There are several reasons for using two kinds of statements in the code read by subjects:

- both kinds of statements frequently occur in source and using them together allowed the questions asked of subjects to reflect the kind of questions they have to answer when comprehending source code. 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),
- experience from another (unpublished) experiment found that when performing a single task some subjects became very focused on improving their performance by looking for, and using, patterns in the questions. It was hoped that by forcing subjects to switch between two tasks this unintended focusing behavior would not be a significant factor.

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 32 problems (one per sheet of paper).

```
---------------------- first side of sheet starts here ----------------------
 prevented =  58;
 liberation =  83;
 conception =  94;
---------------------- second side of sheet starts here ---------------------
if ((e > a) && (u < a))
   if (u > e)

      ..............
   else
      ..............
              remember  would refer back   not seen
 suspend =      ____          __              __
 prevented =    ____          __              __
 liberation =   ____          __              __
 conception =   ____          __              __
```

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 software developers.
The instructions given to subjected 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 are then asked 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 value of three different variables and recalling these values later. The variables and their corresponding values appear on one side of the sheet of paper and your response needs to be given on the other side of the same sheet of paper.

1. *Read the variables and the values assigned to them as you might when carefully reading lines of code in a function definition.*
2. *Turn the sheet of paper over. Please do **NOT** look at the assignment statements you have just read again, i.e., once a page has been turned it stays turned.*
3. *Assuming that the condition specified in the first if-statement is true, which arm of the nested if-statement will be executed? Treat the paper as if it were a screen, i.e., it cannot be written on. Mark the arm you think will be executed with a cross or a tick.*
4. *You are now asked to recall the value of the variables read on the previous page. There is an additional variable listed that did not appear in the original list.*
   - *if you remember the value of a variable write the value down next to the corresponding variable,*
   - *if you feel that, in a real life code comprehension situation, you would reread the original assignment, tick the "would refer back" column of the corresponding variable,*
   - *if you don't recall having seen the variable in the list appearing on the previous page, tick the "not seen" column of the corresponding variable.*

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

4.2 The set of possible questions

It was hoped that at least 32 people (on the day 40) would volunteer to take part in the experiment and it was estimated that each subject would be able to answer 32 problems (on the day 22.7) in 20-30 minutes (on the day 20 minutes). Based on these estimates the experiment would produce 1024 (on the day 884) answered problems.

Given the 8 different ways of ordering the operands and operators appearing in the chosen from of the if statement conditional expression and the 4 different questions that can be asked, it is possible to create 32 different if statement problems.

It was decided to use four sets of identifiers in the assignment problems, with each set containing four different identifiers. The possible values assigned to these identifiers were drawn from a set of four possible two digit integer literals (the rationale is discussed below). Given 16 possible identifiers and 4 possible numeric values (8 had been intended, but a bug in the generation script meant that only 4 were ever used), it is possible to generate 80,640 different sets of 3 assignments (the same identifier or value only being allowed to occur once in any set of assignments). However, if all identifiers within a given set are considered to be equivalent and all two digit values are considered equivalent, then there are only 4 different sets of assignments (a set containing single digit constants had also been planned, which would have created 8 different sets of assignments).

Combining 32 different if statement problems with 4 different sets of assignment problems creates a total of 128 different problems (256 had been intended). Given 1024 answers then there would be 8 answers for each different problem (assuming subjects answered all problems).

The problems and associated page layout were automatically generated using a C program and various awk scripts to generate troff, which in turn generated postscript. The identifier and constant used in each assignment statement was randomly chosen from the appropriate set and the order of the assignment statements (for each problem) was also randomized. The (corrected) source code is available on the experiments web page.[9]

5 Selecting identifiers and integer constants

Studies have found that people's performance in processing character sequences can vary between different kinds of sequences. For instance, frequently used character sequences (i.e., words) are recognized faster

and are more readily recalled than rare ones, also many performance characteristics are slower and more error prone for non-words compared to words, recognizing known subsequences (e.g., ibmciairs) within a longer character sequence allows it to be divided up into a smaller number of larger chunks[15] (i.e., such recognition reduces information content and requires less storage resources).
Some of the factors affecting peoples performance in recalling recently read information include:

- the encoding used for the information. For instance, a sequence having the same form as a word in a language known by a person can be encoded in a sequence of sounds that is shorter than the sequence of sounds representing the individual characters,
- the extent to which people are able to maintain the information in short term memory. This will depend on the short term memory resources consumed by the encoded information and other calls on short term memory resources between when the information is originally encoded and when it needs to be recalled,
- the extent to which the information is already stored in longer term memory subsystems. For instance, this information may exist because a character sequence has been encountered before, or its sound pattern matches (or rhythms with) that of a known word. It is also possible that a persons brain happens to store a given character sequence into a longer term memory subsystem, when it is encountered.

The identifier attributes varied in this study were the amount of short term memory storage required to hold their spoken form (the number of syllables was used as an approximate indicator of storage requirements; the effects of phonological complexity[14] were ignored), and they were either a word (i.e., they were established in long term memory) or a sequence of unrelated characters. The identifiers thus belonged to one of four possible sets of character sequences.

5.1 Identifier character sequences

A variety of different kinds of character sequences are used to represent identifiers in source code. Some are recognizable words or phrases, some abbreviated forms of words or phrases, while others have no obvious association with any known language (e.g., they may be acronyms that are unknown to the reader). It is to be expected that subjects memories of an identifier will be sound based, rather than vision based. For instance, a character sequence representing a known word is likely to be remembered as the spoken form of that word, while a sequence of unrelated characters might be remembered as the spoken form of each individual character.
Subjects are likely to have read many distinct character sequences every day for most of their lives. Many of these character sequences will have been stored in every subjects long term memory and be readily available for recall. Creating a character sequence that only evokes a response from a subject's short term memory is likely to be impossible. Whatever character sequence is chosen, it is likely that there will be some form of association with the contents of a subject's long term memory. The best that can be achieved is to use a set of character sequences, for identifiers, that all result in the contents of long term memory having the same impact on performance for all subjects.
Experience shows that developers sometimes read source code so quickly that visually similar, but different, identifiers are treated as being the same identifier. To reduce the possibility of this occurring during the experiment an attempt was made to use visually distinct character sequences (this involved arranging for ascending e.g., t, and descending e.g., p, characters to occur at different relative locations in a sequence).
All words used in the study had a frequency of occurrence of between 1 per 18 million words and 1 per 24 million words (word frequency counts were based on the British National Corpus[11]). The Collins Advanced Learners English Dictionary was used for syllable counts.
The four sets of identifiers used in assignment statements were:

1. a single character whose spoken form contained a single syllable. The least frequently used letters in written English are *wybvkxjqz*. For reasons lost in the mists of time, the letters *wxyz* rather than the overall less frequent (and not sequential) *xjqz* were used,

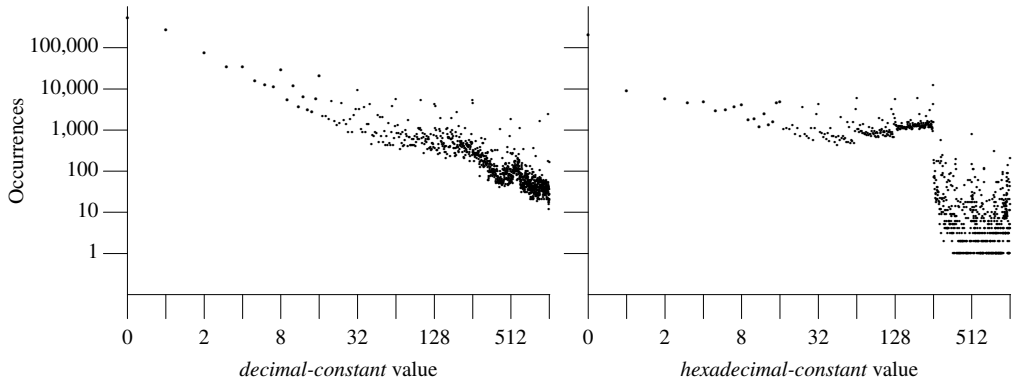

Figure 2: Occurrences, in the visible form of various applications written in C. of integer constants with different values.[10]

2. an English word whose spoken form contained one syllable (i.e., van, guy, tip, mud),
3. three characters whose spoken form is likely to contain three syllables (i.e., vcq, qmt, bfj, rpl). That is the characters did not represent an English word. Google was used to reduce the possibility that the character sequence did not denote an acronym that was likely to be contained in subjects long term memory (e.g., IBM). Google returned a page count[1] of between 10,000 and 34,000 matches for the character sequences used (most other such sequences each returned over 100,000 matched pages).
4. an English word whose spoken form contained three syllables (i.e., conception, suspend, prevented, liberation).

In the rest of the article the term *short identifier* denotes an identifier whose spoken form is short (i.e., it contains a single syllable) and the term *long identifier* denotes an identifier whose spoken form is long (i.e., it contains three syllables). In practice the only reliable method of finding out the duration of the spoken form of word is to average the time taken by various people to say the repetitively word.

The character sequences first selected did not appear to share any common sounds that might result in increased interference between them when held together in short term memory.[2]

5.2 if statement identifiers

It was intended that the only cause of interference between the identifiers used in the two forms of statements should be contention for short term memory resources. For this reason the identifiers chosen for the two kinds of statements were distinct, both in terms of visible appearance and sounding different.

The most frequently used letters in written English are *etaoinsrhldcu*. For reasons lost in the mists of time, the single letters *aeu* rather than overall more frequent (and not all vowel) *eta* were used,

5.3 Selecting integer constants

Measurements of the frequency of integer constants in various contexts have found that some values occur more frequently than others. Measurements of source code have found various patterns between numeric values and the frequency with which they appear in the visible source code, Figure 2.

The following integer constants were chosen (the digit 7 was not used in any value because its spoken form has two syllables):

- single digit numbers. The values 5, 6, 8, and 9 were chosen because they all have approximately the same frequency of occurrence in source code and other contexts, and have a spoken form containing a

[1] At the start of April 2004.

[2] Your author would not claim to any special knowledge on how common sounds (*phonological similarity* is the technically correct term and proposals have been made for measuring it[14]) might be measured or which sounds might interfere with each other.

single syllable. However, due to a bug in the script generation program no single digit numbers were used in this study,

- two digit numbers. These have the advantage over three digit numbers in that they are all likely to be encoded using a single spoken form (many three digit numbers have many possible spoken forms e.g., 869 might be spoken as eight-six-nine or eight hundred and sixty nine).

6 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 some subjects will have detected what they believe to be a pattern in the problems and will then attempt to use this information to improve their performance. Possible patterns appearing across problems include:

- a bug in the problem generation script meant that the identifier that did not appear in the list of assignment statements always appeared first in the list of to be recalled information. At least one subject noticed this pattern (he raised it during discussions after he completed the experiment),
- the number of identifiers used was a very small subset of those that could have been used. This meant that the first character of each character sequence was unique to that identifier (i.e., there was only one identifier starting with any given letter of the alphabet). At least one subject noticed this (in discussions after completing the study he said that he had saved time by only encoding the first few letters of the longer identifiers),
- the ordering of the identifiers in the assignment statements and in the to be filled in list of recalled information was the same. It is not known if any subjects noticed this pattern and used it to improve their performance.

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., assignment 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 were only required to tick a box to indicate that they *would refer back* to locate the information. The cognitive effort needed to tick a box is likely to be less than would be needed to actually refer back. Studies have found[4] 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.

A previous experiment (unpublished), involving a source comprehension task that only contained conditionals, found that some subjects solution strategies changed during the course of answering questions. Initially these subjects obtained their answers by applying the traditional algebraic strategies usually associated with solving logic problems. However, developers familiarity with problem solving is not confined to source code comprehension and is often applied to the problem of minimizing the effort they need to expend on the task. In the case of having the solve a sequence of conditional problems some subjects switched to a pattern matching strategy. That is, they looked for (and claimed to have found) patterns in the questions that enabled them to quickly provide what they believed to be the correct answer (i.e., the answer to a question was based on matching it to a pattern having a known answer). It is possible that the intervening assignment problem did not provide sufficient cognitive demand (i.e., distraction) that in some cases subjects gave answers to the if statement problem based on patterns they believed to exist in the sequence of problems they saw.

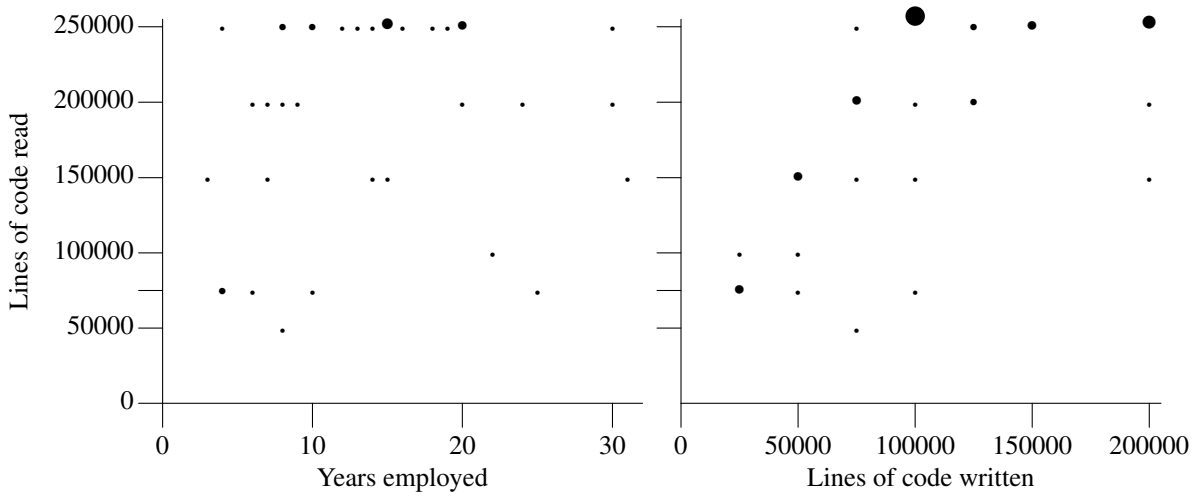

Figure 3: The plot on the left depicts number of line of code read against number of years of professionally experience. The plot on the right depicts the number of lines of code read against the number of lines of code written, for each subject. The size of the circle indicates the number of subjects specifying the given values. In those cases where subjects listed a range of values (i.e., 50,000-75,000) the median of that range was used.

7 Results

7.1 Subject experience

Traditionally, developer experience is measured in number of years of employment performing some software related activity. However, the quantity of source code (measured in lines) read and written by a developer (developer interaction with source code overwhelmingly occurs in its written, rather than spoken, form) is likely to be a more accurate measure of source code experience than time spent in employment. Interaction with source code is rarely a social activity (a social situation occurs during code reviews) and the time spent on these activities may be small enough to ignore. The problem with this measure is that it is very difficult to obtain reliable estimates of the amount of source read and written by developers. This issue was also addressed in a study performed at a previous ACCU conference.[8] While it was hoped that some of the problems encountered in that study were solved in the current study, the results, Figure 3, suggest that the upper range of possible answers is still insufficient to cover the amount of code that subjects believe they have read.

Plotting the number of lines read against number of lines written gives a ratio of approximately 2.5 lines read per line written. Your authors experience suggests this ratio ought to be greater than 25. One possible reason for this difference is that the questions asked (e.g., "*How many lines of code would you estimate you have {**read**|**written**}, in total, over your career*") are open to various interpretations. For instance, does reading previously read code count towards the total number of lines read (previously read lines that a developer has forgotten about might be thought to result in more learning than lines reread after a time delay of a few minutes), and how should changes that modify part of an existing line be counted?

It has to be accepted that reliable estimates of lines read/written are not likely to be available until developer behavior is closely monitored (e.g., eye movements and key presses) over an extended period of time.

A plot of problems answered against experience, Figure 4, does not show any correlation between the two quantities. The number of subjects in each quadrant is approximately the same.

7.2 Assignments

The following discussion breaks the results down by individual subject and by kind of identifier used in the assignment statements. The raw results for each subject are available on the studies web page.[9] While there is enough raw data to perform detailed statistical analysis, none is performed. There are enough threats to

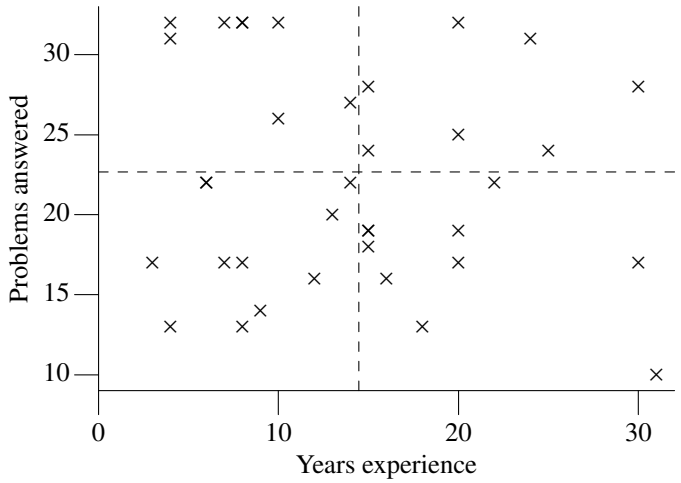

Figure 4: Plot of the number of problems answered against the number of years of professional experience of the subject. Dashed lines represent the mean number of problems answered (22.3) and the mean number of years of experience (14.5). The problems answered/years experience pairs (22, 6), (32, 8), and (19, 15) occurred for two subjects each.

validity to render the conclusions from any such detailed analysis spurious. However, it is hoped that some general conclusions can be drawn from the results obtained.

A total of 844 sets of assignment statements were remembered/recalled giving a total of 2,547 answers to individual assignments. The answer given to 43 assignment statement questions was an **x** in the *remember* column (only for some of the initial problems answered by a few subjects). This response was treated as indicating that the subject believed they knew the answer. However, since no value was specified it was not possible to verify the accuracy of the response. Therefore answers having this form were ignored (they were not counted in any category).

The number of incorrect *not seen* answers decreased from 13% (averaged over answers from all subjects) for the first eight problems to 7% for the ninth and subsequent problems. It is inevitable that some subjects will have noticed that the correct answers was always the first identifier in the response list. However, not all subjects noticed this pattern (i.e., they continued to give incorrect answers; in some cases a greater percentage of incorrect answers).

7.2.1 Individual subject performance

There is a great deal of variation in subject performance. Correct recall performance varied between 0 and 96%, instances where subjects would refer back varied between 0 and 94%, while incorrect answers varied between 1 and 40% of all answers given by any subject. This extreme variation suggests that the experimental design aim of creating problems whose solution stretched the limits of subject's short term memory capacity was achieved. Had the problems required more or less short term memory capacity then it is likely that the various in subjects performance would have been narrower (i.e., subjects would have be likely to have provided a fewer or a greater number of incorrect or *would refer back* answers).

A *would refer back* response does not imply that a problem has exceeded a subjects short term memory capacity. It could imply that the subject is a very cautious individual, or that they were distracted by other thoughts while answering a particular problem.

If subject performance was consistent for all problems answered, it would be expected that averaged results for the first few problems answered would be the same as for the last few problems. Figure 5 plots *would refer back* and incorrect answer performance for the first eight and for the ninth and all subsequent problems answered. The lack of clustering of the bullets with the crosses means there was little correlation between the two sets of results. There are approximately twice as many dots below the crosses as there are above,

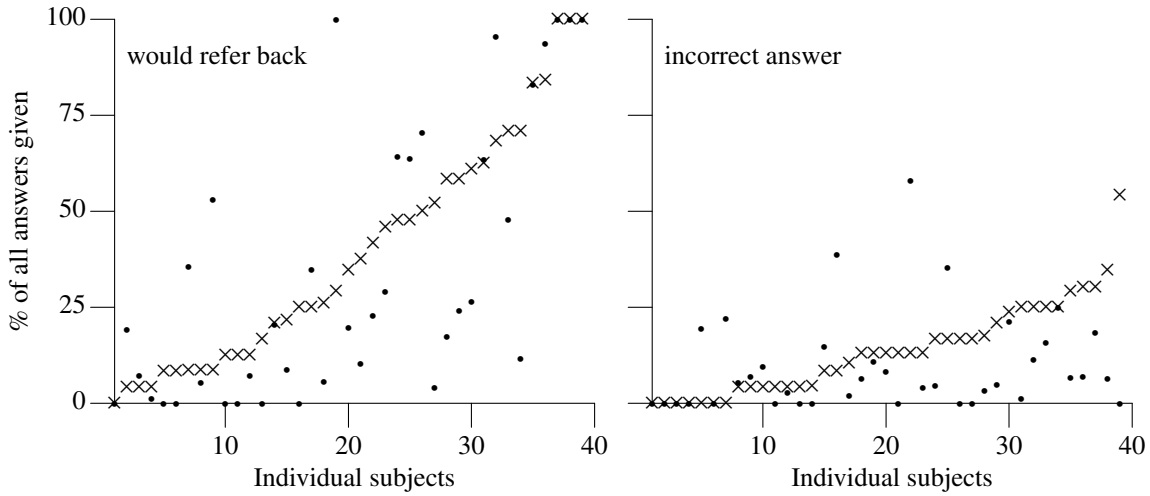

Figure 5: Left graph is the percentage of *would refer back* answers for the first 8 problems (crosses) and for the ninth and subsequent problems (bullet) answered (subjects are ordered by increasing *would refer back* response rates). Right graph is the percentage of incorrect answers with subjects being ordered by increasing percentage of incorrect.

which suggests that an individuals performance improved as more problems were answered.

One possible reason for an increase in performance is because answering problems enabled subjects to learn something that was beneficial in answering subsequent questions (e.g., the first identifier in the list of assignment questions was always the one that did not appear in the previous assignment statements). One possible reason for a decrease in performance is that subjects became fatigued through having to answer so many questions that constantly stretched the capacity limits of their short term memory.

It might be thought that a subject answering a greater number of questions would be more likely to give incorrect or *would refer back* answers. Figure 6 shows that this is not the case. Fitting a least squares line through the data shows that both the percentage of incorrect and *would refer back* answers decreased as more questions were answered. As pointed out earlier it is possible that some subjects were able to detect and use patterns in the presentation of the problems to improve their performance. This improvement in performance could take the form of an increase in the number of problems answered as well as an increase in the number of correct answers.

7.2.2 Different kinds of identifiers

The analysis of individual subject results suggests that their performance improved as they answered more problems were answered. The analysis of the results for different kinds of identifiers takes this behavior into consideration by dividing the results in two; those from the first eight problems answered and those from the ninth and all subsequent problem answers.

There are a number of surprises in the results, Figure 7, (at least for your author):

1. for the first eight problems the pattern of answers for the identifiers composed of three unrelated letters does not follow that of the identifiers composed of three syllable words. One explanation for the three unrelated letter behavior is that these letter sequences are likely to be completely unknown to subjects (they were selected on this basis). When asked to recall information about previously seen assignment statements subjects were initially unable to make use of any longer term memory associations as a recall aid, and so opted for the *would refer back* option. As more problems were answered, and subjects encountered more instances of the three unrelated letter sequences used, it is possible that some information about these letter sequences became stored in longer term memory

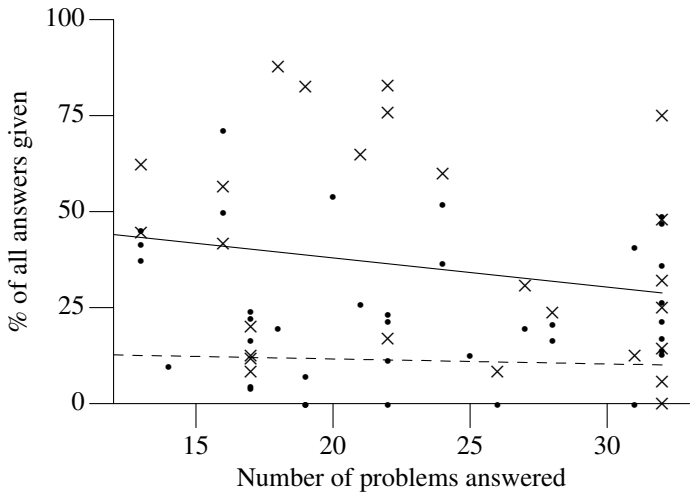

Figure 6: The percentage of *would refer back* answers (crosses, least squares line unbroken)) and incorrect answers (bullet, least squares line dashed) plotted against the number of problems answered by each subject.

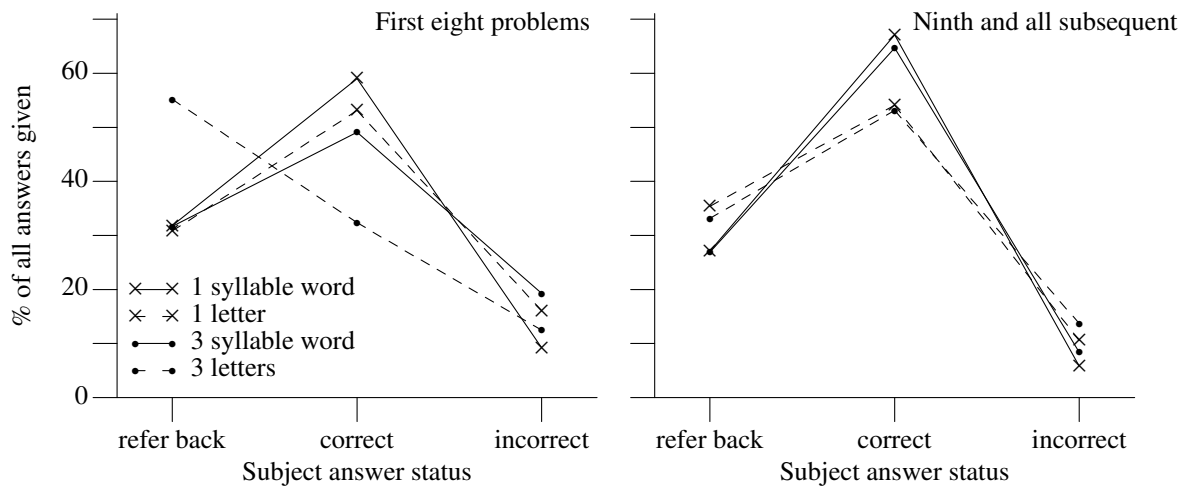

Figure 7: The percentage of *would refer back*, correct and incorrect answers for each kind of identifier, averaged over all subjects. The left graph is based on answers to the first eight problems, while the right graph uses the answers from the ninth and subsequent problems answered.

subsystems and subjects were able to make use of this *new* existing knowledge,[3]

2. for the first eight problems subject performance is best for short identifiers. For the ninth and subsequent problems the results showed what might be called a *word superiority effect* (i.e., a greater number of correct answers). This suggests that after some practice the contents of a persons longer term memory (i.e., their experience in using words) has a greater impact on performance than limits on their short term memory capacity. The extent to which solving the if statement problem may have resulted in a degrading of the contents of short term memory (i.e., assignment statement information) is discussed in part two of this article.

7.3 Kinds of recall errors

If the repetitive process of remembering assignment information caused the numeric values seen to be stored in longer term memory, then it would be expected that the set of values recalled in error would converge to the set of values seen during the experiment. The results, Table 1, show a small increase (between the first eight, and the ninth and subsequent answers) in the number of incorrect answers given that appear somewhere in the list of assignment statements that a subject saw for a given problem (fifth row). There is a larger increase in the number of incorrect answers given that come from the set of all values seen during the experiment (last row).

Table 1: Number of various kinds of recall errors made by subjects when answering the assignment problem. The percentage is calculated using the total at the top of the corresponding column. The phrase *in list* refers to the constant values appearing in the list of assignment statements read immediately prior to the if statement. The phrase *in set* refers to the set of all possible constant values appearing in assignment statements. The first digit is the most significant digit.

	first eight	ninth and subsequent	total
total recall errors	126	158	284
both digits incorrect	64 (51%)	104 (66%)	168
only first digit incorrect	34 (27%)	27 (17%)	61
only last digit incorrect	28 (22%)	27 (17%)	55
answer given in list	56 (44%)	76 (48%)	132
first digit in list	28 (22%)	23 (15%)	51
last digit in list	20 (16%)	18 (11%)	38
answer given in set	61 (48%)	91 (58%)	152

8 Discussion

Based on both years of employment and the claimed number of lines of code read/written the subjects taking part in the experiment have a significant amount of software development experience.

The number of years of software development experience is likely to have a high correlation with a subjects age. While cognitive performance has been found to decrease with age,[5, 6] age does not appear to have been a factor affecting the number of questions answered in this experiment (however, most subjects are likely to be younger than the age at which studies find a significant age decrease in performance; 50s and over).

The aim of creating a problem that would require approximately 30 seconds to answer was not met. The average time taken to answer problems was 67 seconds, over twice that intended in the experimental design. It is possible that a subject's short term memory resources were completely consumed by solving the if statement problem.

Given the experience of the subjects participating in this experiment any learning affects that occurred are likely to be caused by patterns in the presentation of the problems (e.g., particular identifiers always appearing in a given order). Known patterns include:

[3] Subjects were not asked to provide a guess for those cases where they *would refer back*, so it is not possible to measure the accuracy of any information they might have believed they had on a given assignment statement.

- using a relatively small, compared to the number of problems seen by a subject, set of identifiers. The results show that when answering the initial problems recall performance was significantly better for short identifiers. The change in performance characteristics, as subjects answered more problems, could have been caused by subjects learning the limited number of different identifiers used in the experiment, or it could have been caused by something else being learned. Repeating the experiment using a greater number of different identifiers will help answer this question,
- using a relatively small, compared to the number of problems seen by a subject, set of constant values. The issues here are the same as those for using a small set of identifiers,
- listing the identifiers in the same order in the recall list as they appeared in the assignment list. Subjects could have used this information to answer problems without remembering any identifier information. While identifiers sometimes need to be recalled in the same order in which they are read in the source, this is not always the case. Repeating the experiment using different relative orderings will remove this possible threat to validity,
- having the first identifier in the recall list as the identifier that did not appear in the assignment list. While the problem appears to be difficult enough without this identifier, its presence provides a mechanism for estimating the amount of guessing made by subjects in their answers.

More results are discussed in the second part of this article.

9 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.

10 Acknowledgments

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

References

At the end of each entry, the pages on which that entry is cited are listed in parentheses.

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

[2] A. D. Baddeley. *Essentials of Human Memory*. Psychology Press, 1999.

[3] G. Boole. *An Investigation of the Laws of Thought*. Dover Publications, 1973.

[4] 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.

[5] A. S. Gilinsky and B. B. Judd. Working memory and bias in reasoning across the life span. *Psychology and Ageing*, 9(3):356–371, 1994.

[6] D. Z. Hambrick and R. W. Engle. Effect of domain knowledge, working memory capacity, and age on cognitive performance: An investigation of the knowledge-is-power hypothesis. *Cognitive Psychology*, 44(4):339–387, 2002.

[7] D. M. Jones. The 7 ± 2 urban legend. MISRA C 2002 conference `www.knosof.co.uk/cbook/misart.pdf`, Oct. 2002.

[8] D. M. Jones. I_mean_something_to_somebody. *C Vu*, 15(6):17–19, Dec. 2003.

[9] D. M. Jones. Experimental data and scripts for short sequence of assignment statements study. `www.knosof.co.uk/cbook/accu04.html`, 2004.

[10] D. M. Jones. An interpretation of the C standard. Knowledge Software, Ltd, 2004.

[11] G. Leech, P. Rayson, and A. Wilson. *Word Frequencies in Written and Spoken English*. Pearson Education, 2001.

[12] A. R. Luria. *The mind of a mnemonist*. Harvard University Press, 1986.

[13] K. Manktelow. *Reasoning and thinking*. Psychology Press, 1999.

[14] S. T. Mueller, T. L. Seymour, D. E. Kieras, and D. E. Meyer. Theoretical implications of articulatory duration, phonological similarity, and phonological complexity in verbal working memory. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 29(6):1353–1380, 2003.

[15] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 623–656, 1948.