LETTER *Special Section on Empirical Software Engineering*

Relationship between Code Reading Speed and Programmers' Age

Yukasa MURAKAMI[†], ***Nonmember***, **Masateru TSUNODA**[†a)], ***and*** **Masahide NAKAMURA**[††], ***Members***

SUMMARY According to the aging society, it is getting more important for software industry to secure human resources including senior developers. To enhance the performance of senior developers, we should clarify the strengths and weaknesses of senior developers, and based on that, we should reconsider software engineering education and development support tools. To a greater or lesser extent, many cognitive abilities would be affected by aging, and we focus on the human memory as one of such abilities. We performed preliminary analysis based on the assumption. In the preliminary experiment, we prepared programs in which the influence of human memory performance (i.e., the number of variables remembered in the short-term memory) on reading speed is different, and measured time for subjects to understand the programs. As a result, we observed that the code reading speed of senior subjects was slow, when they read programs in which the influence of human memory performance is larger.

key words: *developer's performance, code reading, mental simulation*

1. Introduction

In software development, the influence of developers' performance to project outcome such as productivity is not ignorable. Some studies have pointed out that the performance of each software developer is very different from each other. For example, debugging performance of developers was very different from each other, in experiments performed by Sackman et al. [8]. Likewise, code review performance was very different in experiments of Thelin et al. [12]. According to the aging society [10], it is getting more important for software industry to secure human resources including senior developers.

To maximize performance of senior developers, we should clarify the strengths and weaknesses of senior developers, and based on that, we should reconsider software engineering education and development support tools. This is because to reduce the difference of performance between developers, it is necessary to clarify factors affecting the difference. After that, we should consider software engineering education and development support tools which lessen the influence of the factors. For example, when performance of code review is different from each developer, and the reason is lower performance developers do not check consistency of specification and source codes. In this case, if "Check consistency of specification and source codes" is added to a checklist of review, the difference of the performance is expected to be decreased.

Therefore, as one of the factors of the developers' performance, we focus on developers' age. That is, we assume that when developers' age is high, some abilities relating to programming vary. Although there is no scientific evidence, some people mention the relationship between the performance of developers and their age. For example, an article [14] pointed out younger developers learn new coding methods and techniques more quickly than senior developers. However, there are very few studies which analyze whether developers' age affects their performance or not, as long as we know. Also, it is not clear which ability is affected by the age. Although other fields such as Web design cope with issues of the age [15], software engineering studies have not paid enough attention to the age.

Code reading is one of important activities related to software development. When we modify source codes or review them, we have to read them to understand them. To a greater or lesser extent, many cognitive abilities would be affected by aging, and we focus on human memory as one of such abilities. We performed analysis based on the assumption. That is, when developer's age is high and the reading speed of a program is greatly affected by human memory performance (i.e., the number of variables remembered in the short-term memory), understanding speed of the program is slow. If there is such a relationship, we should make a new development support tool which helps human memory for senior developers. Therefore, the goal of our study is to maximize the performance of senior developers through such tools.

2. Program Understandability Metrics

To measure the extent to which the reading speed of a program is affected by human memory performance (i.e., the number of variables remembered in the short-term memory) quantitatively, we use understandability metrics [4], [7]. The metrics evaluate understandability based on the human short-term memory model, instead of the complexity of the program. In other words, when the metrics regard that a program is difficult to understand, the reading speed of a program is greatly affected by human memory performance.

The metrics assume that the mental simulation [3] is applied to understand a program. It means a developer comprehends a program without computers and memos, but with

Manuscript received February 28, 2020.
Manuscript revised May 22, 2020.
Manuscript publicized September 17, 2020.

[†]The authors are with Kindai University, Higashiosaka-shi, 577–8502 Japan.
[††]The author is with Kobe University, Kobe-shi, 657–8501 Japan.
a) E-mail: tsunoda@info.kindai.ac.jp

DOI: 10.1587/transinf.2020MPL0002

Copyright © 2021 The Institute of Electronics, Information and Communication Engineers

his/her thought. It is often applied when a developer reads relatively small code fragments. When applying the mental simulation, values of variables should be remembered. However, it is not easy to remember many values. So, the studies [4], [7] assume that understandability of the program is low when the recalculation cost of the values of the variables that have disappeared from the short-term memory is high.

Some program complexity metrics such as the number of live variables [1] would be considered in terms of human memory performance. However, they do not assume the mental simulation. In the experiment, the subjects understood the programs based on mental simulation. Therefore, we found the metrics unsuitable for our study, compared with the understandability metrics [4], [7].

In the study [7], short-term memory is regarded as a FIFO queue, and they made understandability metrics. The basic idea of the metrics is that the size of FIFO queue is limited. When a developer refers a variable and its value is stored in the queue, the cost of the mental simulation is regarded as low. In contrast, when the value is not stored in the queue, the cost is regarded as high, since the developer should backtrack to the point where the value of the variable was changed. In the study, four metrics were defined as follows:

- ASSIGN: cost with regard to variable assignment.
- RCL: cost of recalling a value of a variable in short-term memory.
- BT_CONST: number of times about backtracking of a constant. The cost of acquiring a value of the constant.
- BT_VAR: the distance of backtracking of a variable. The cost of acquiring a value of the variable.

The study [4] pointed out that the metric BT_VAR does not consider recalculation cost, considering the number of times about updating variables. Also, the study mentioned that the metric is too sensitive to change order of lines of codes. To solve the problem, the study proposed two metrics. The number of times of updating on each variable is regarded as elements of a vector, and the authors defined a metric, SUM_UPD which is based on the sum of the elements. Also, they defined a metric, SUM_VAR, which is based on the variance of the elements. When SUM_UPD is larger or SUM_VAR is smaller, the understandability of the program is lower.

The metrics BT_CONST, BT_VAR, SUM_UPD, and SUM_VAR signify the recalculation cost when the values of the variables or the constants do not exist in the short-term memory. Therefore, the following applies when the values of the metrics are *low* (and the values of SUM_VAR are high):

- When the values disappear from the short-term memory (i.e., human memory performance is low and the number of variables remembered in the short-term memory is small): *low* recalculation cost is needed to obtain the values.
- When the values remain in the short-term memory (i.e., human memory performance is high, and the number of variables remembered in the short-term memory is large): *low* recalculation cost is NOT needed to obtain the values.

Therefore, the total cost to understand the program (i.e., reading speed) is *less* affected by human memory performance when the values of the metrics are *lower*. Similarly, the reading speed is affected by human memory performance *more* when the values of the metrics are *higher*.

Thus, the metrics denote the extent of the influence of human memory performance on reading speed. Assume that the reading speed of senior developers is higher in programs where the extent of the influence is smaller and the reading speed is lower in programs where the extent of the influence is larger. In this case, the reading speed of the senior developers is considered to be affected by human memory performance to some extent, and we should make a new development support tool for senior developers that helps human memory.

3. Experiment

3.1 Overview

The purpose of the experiment is to analyze understanding speed of senior subjects, when they read a program in which the influence of human memory performance (i.e., the number of variables remembered in the short-term memory) is larger. We prepared programs in which the influence of human memory performance on the reading speed is different, and measured time for subjects to understand the programs.

We called them as program a0, a1, b0 and b1, which were used in the study [4]. The size of the programs is about 20 to 30 lines of codes. Program a0 (easy to read) and b0 (difficult to read) are shown in Fig. 1. We asked subjects a value of a variable after program execution. For example, about program b0, subjects answered the value of the variable “g” after execution. The subjects understood the

```
int i , t;

t = 11;
t = t - 1;
i = 2;
if(i < t){
    i = i + 2;
    if(i < t){
        i = i + 2;
    }
    if(i < t){
        i = i + 2;
        if(i < t){
            i = i + 2;
        }
    }
    if(i < t){
        i = i + 2;
    }
}
System.out.println("i = "
+i);
```

(a0)

```
int a, b, c, d, e, f, g;

a = 2;
b = 4;
c = 3;
d = 6;

c = c + 4;
d = d - 2;
if(c < 5)
    e = d + 5;
else
    e = d + 3;
a = a * 2;
b = b + 6;
if(a > 7)
    f = b - 3;
else
    f = b - 5;
g = e + f;

System.out.println("g =
"+g);
```

(b0)

Fig. 1 Program a0 and b0 [4]

programs based on the mental simulation (i.e., we did not allow for subjects to use memos).

To measure the extent to which the reading speed of a program is affected by human memory performance, we used six metrics explained in Sect. 2. Table 1 shows the metrics of each program. The ASSIGN, BT_CONST, and SUM_UPD metrics are larger on programs b0 and b1, while VAR_UPD is smaller. As explained in Sect. 2, BT_VAR has some drawbacks (e.g., the metric is too sensitive to the change order of lines of codes). Therefore, we disregarded BT_VAR.

Based on the metrics, the reading speed of programs b0 and b1 is relatively more affected by human memory performance. We should consider the validity of the understandability metrics. There are many variables to be remembered in program b0 and b1, and hence they are regarded to be difficult to understand. Therefore, the assumption about the programs is correct if the metrics are not very accurate.

We classified subjects into the young group and the senior group. Subjects of the young group were students who study information science at Kindai University, the first author's university. They were fourth year undergraduate students and master's students, and the number of the subjects was 24. Their age was around 22 to 24. Subjects of senior group were faculty members (professors) of Kindai University (Department of informatics). The average age of the senior group was 45. Minimum age of them was 33 and maximum age was 64. The number of the subjects was 8.

To avoid the influence of the order of reading programs, we changed the order for each subject in the experiment. For example, a subject reads the programs in program a0, a1, b0, b1 order. Another subject reads the programs in program b1, a0, b0, a1 order.

3.2 Results

Table 2 shows the average and median of answering time of the senior group and the young group. Figure 2 shows boxplots illustrating the time difference between groups. In Table 2, when focusing on program a0 and a1, the time of the senior group was shorter than the young group, except for average of program a0. In contrast, when focusing on program b0 and b1, the time of the young group was shorter than the senior group.

Table 3 shows the percentile rank of each senior subject (s1 ... s8 indicate each subject), when they are compared with young subjects. Bold type signifies the rank was larger than 50%. On program a0 and a1, answering time of senior subjects was shorter than the median (i.e., the rank is smaller than 50%) of the young group on five or six out of eight subjects. In contrast, on program b0 and b1, answering time of senior subjects was larger than the median of the young group on six out of eight subjects. The result suggests there is the relationship between code reading speed and the age of subjects, when they read programs in which the influence of human memory performance is larger.

Next, to analyze the reading speed relatively, we set answering time of program a0 as the benchmark, since, in the program, the influence of human memory performance is the smallest. To calculate the ratio of answering time of program a0 and others, we divided answering time of each program by the time of program a0. The result is shown in Table 4. The boxplots in Fig. 3 illustrate their difference between groups. In Table 4, focusing on the young group, the average and the median of b0 / a0, and b1 / a0 (i.e., relative answering time of b0 and b1) were lower than 2.0, except for the average of a0 / b0. In contrast, on senior group, all

Table 1 Understandability of programs in the experiment [4]

	ASSIGN	RCL	BT_ CONST	BT_ VAR	SUM_ UPD	VAR_ UPD
a0	12	6	0	80	7	1.25
a1	12	6	0	48	7	0.25
b0	18	8	1	30	11	0.24
b1	18	6	1	54	11	0.24

Table 2 Answering time of each group (seconds)

		a0	a1	b0	b1
Young	Average	65	82	105	89
	Median	58	75	100	70
	S.D.	37	44	52	70
Senior	Average	59	91	141	115
	Median	51	51	144	109
	S.D.	22	86	65	62

Table 3 Percentile rank of time of senior subjects

	s1	s2	s3	s4	s5	s6	s7	s8
a0	**81**	**80**	33	9	**79**	33	42	46
a1	**100**	48	4	43	38	30	8	**93**
b0	**100**	**72**	2	28	**91**	**87**	**72**	**89**
b1	**79**	**88**	**87**	**62**	48	27	**88**	**97**

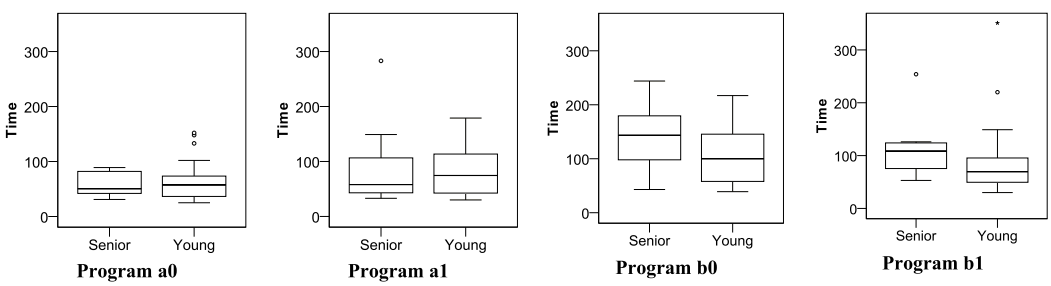

Fig. 2 Answering time of each program

Table 4 Ratio of answering time to program a0

		a1 / a0	b0 / a0	b1 / a0
Young	Average	1.6	2.0	1.4
	Median	1.2	1.5	1.4
	S.D.	1.4	1.5	0.7
Senior	Average	1.5	2.4	2.2
	Median	1.0	2.4	2.4
	S.D.	1.0	0.9	1.3

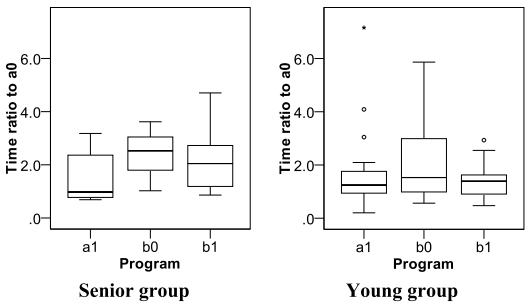

Fig. 3 Ratio of answering time to a0

Table 5 Percentile rank of relative time of senior subjects

	s1	s2	s3	s4	s5	s6	s7	s8
a1	**92**	17	18	**82**	15	43	19	**90**
b0	**74**	**59**	26	**67**	**71**	**84**	**75**	**78**
b1	29	**63**	**99**	**97**	23	37	**96**	**100**

Table 6 The number of incorrect answers

		a0	a1	b0	b1
Young	Average	1.0	0.5	1.3	0.8
	Median	0.0	0.0	0.0	0.0
	S.D.	2.5	0.8	2.2	2.3
Senior	Average	0.1	0.4	1.1	0.5
	Median	0.0	0.0	1.0	0.0
	S.D.	0.4	0.7	1.0	0.8

of them were larger than 2.0.

Table 5 shows the percentile rank of each senior subject. On program b0 (took the most time to read), the ratio of senior subjects was larger than the median of the young group on seven out of eight subjects. On program b1, answering time of senior subjects was larger than the median of the young group on five out of eight subjects, and the percentile rank was larger than 90% on four subjects. So, understanding speed of senior subjects is considered to be relatively slow, when they read programs which need much memory to understand them.

Basic statistics of the number of incorrect answers are shown in Table 6. The boxplots in Fig. 4 illustrate their difference between groups. In Table 6, the median of the number was 0, except for program b0, and there were few subjects who input incorrect answers many times. Focusing on program b0, the median of senior group was larger than 1.0. So, the program was especially difficult for the senior subjects. Note that in the young group, the standard deviation was larger than 2.0, except for program a1. This is because few subjects input incorrect answers many times. However, the average and the median of the young group were not

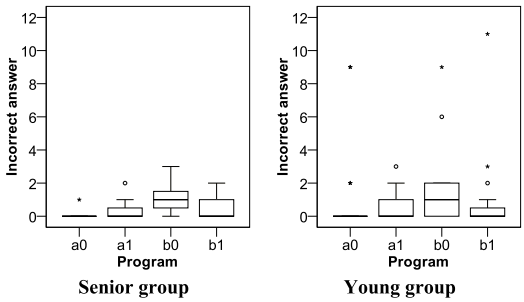

Fig. 4 Incorrect answers

Table 7 Correlations between frequency of reading codes and answering time

	a0	a1	b0	b1	a1 / a0	b0 / a0	b1 / a0
ρ	0.06	0.01	-0.21	0.20	-0.08	-0.19	0.08

very different from senior group. So, the percentage of correct answers is regarded as almost same on each group.

3.3 Discussion

The frequency of reading source codes on subjects may affect the results. So, we asked daily code reading/writing time to subjects. The average of senior subjects was 5.8 hour per week, and that of junior subjects was 13.9. To analyze the influence of the frequency, we used Spearman's rank correlation coefficient. The correlation between answering time in the experiment and the frequency is shown in the Table 7. In the table, the definition such as b0 / a0 is same as Table 4. The frequency did not have strong relationships to answering time. So, the influence of the frequency to the result is considered to be small.

It is not probable that subjects of the senior group coped with the experiment carelessly. This is because although the order of program a0 was different (i.e., sometimes appeared first, and sometimes appeared last) for each subject, the answering time of program a0 was shorter than the young group on average. Even if the subjects of senior group read all programs carelessly and slowly, understanding speed was relatively slow on senior group (see Table 4), when they read programs which need much memory.

The programs used in the experiment are very simple, compared with programs of actual software. This is because we used them to clarify the influence of subjects' memory. Similarly, other tests to measure the fundamental performance of people such as physical fitness tests and intelligence tests are often far from actual situations. If we used more realistic programs in the experiment, subjects' experience may affect understanding speed. The purpose of the analysis is not to analyze total performance of senior subjects, but to analyze the weak points of senior subjects.

Most subjects in our experiment were not professional software developers. Past studies have shown that students can be used instead of professionals in experiments (i.e., the differences between students and professionals are small) [9]. Therefore, we think that the results of our study

will not be very different when professionals join our experiment.

The number of subjects was not exceptionally large. Subjective experiments are time consuming, and therefore it is not easy to recruit subjects. For example, there were only ten subjects in the study [6]. Additionally, we used small programs and specifications in our experiment. This was done because of experimental time limitations [13]. However, we must consider the number of subjects and the size of programs when interpreting the results.

4. Related Work

Some studies suggested the performance of software developers is very different for each person [8], [12]. For example, the study of Sackman et al. [8] showed performance of debugging program was very different for each person. However, there were very few studies which analyzed the relationship between developers' age and their performance, and analyzed which ability (e.g., memory) is affected by age, as long as we know. The study of Morrison et al. [5] is one of such studies.

Morrison et al. [5] analyzed the relationship between developers' knowledge and their age, using data collected from Stack Overflow, a question-and-answer (Q&A) site. In the analysis, they showed that evaluation of developers was higher, when their age was higher. Although Morrison et al. focused on developers' knowledge, they did not focus on developers' memory. Note that considering all abilities of developers such as knowledge, total performance of senior developers may be higher than younger developers.

In gerontology, several studies [2], [11] showed that the age does not affect the performance of short-term memory very much. However, our experiment suggested that program understanding speed of senior developers was slow, when they read programs in which the influence of performance on the short-term memory (i.e., the influence of the number of variables remembered in the short-term memory) is larger. This may be because understanding programs is not only using memory but also calculation.

5. Conclusion

We analyzed the relationship between subjects' age and code reading speed. In the analysis, we assumed that the code reading speed of senior subjects gets slow, when they read programs in which the influence of human memory performance (i.e., the number of variables remembered in the short-term memory) on the reading speed is larger. The analysis suggested that subjects' age is not ignorable, especially when they read programs in which the influence of human memory performance is larger. Based on our results, showing histories of values of variables and method calling on IDE is effective to enhance code reading performance of senior developers. Although they are important to understand codes, popular IDEs do not show them.

Note that the performance of developers is very different from each other, as explained in Sect. 1. As shown in the analysis (see Table 3 and 5), not all senior subjects read programs slowly, when they read programs in which the influence of the human memory performance is larger. When we cope with age-centered software engineering, we should aware the variance of the performance of developers.

Acknowledgments

This research was partially supported by the Japan Society for the Promotion of Science (JSPS) [Grants-in-Aid for Scientific Research (A) (No.17H00731)]

References

[1] A. Aho, M. Lam, R. Sethi, and J. Ullman, Compilers: Principles, Techniques, and Tools, Pearson Education, 2013.
[2] F. Craik, N. Anderson, S. Kerr, and K. Li, "Memory changes in normal ageing," Handbook of memory disorders, John Wiley & Sons, pp.211–241, 1995.
[3] A. Dunsmore, M. Roper, and M. Wood, "The role of comprehension in software inspection," Journal of Systems and Software, vol.52, no.2–3, pp.121–129, 2000.
[4] T. Ishiguro, H. Igaki, M. Nakamura, A. Monden, and K. Matsumoto, "Evaluating the Cost of Program Mental Simulation Based on the Number and Variance of Variable Updates," Technical report of IEICE, SS-104 (466), pp.37–42, 2004. (in Japanese)
[5] P. Morrison and E. Murphy-Hill, "Is programming knowledge related to age? an exploration of stack overflow," Proc. of Working Conference on Mining Software Repositories (MSR), pp.69–72, 2013.
[6] S.C. Müller and T. Fritz, "Using (bio)metrics to predict code quality online," Proc. of International Conference on Software Engineering (ICSE), pp.452–463, 2016.
[7] M. Nakamura, A. Monden, T. Itoh, K. Matsumoto, Y. Kanzaki, and H. Satoh, "Queue-based Cost Evaluation of Mental Simulation Process in Program Comprehension," Proc. of International Software Metrics Symposium, pp.351–360, 2003.
[8] H. Sackman, W.J. Erikson, and E.E. Grant, "Exploratory experimental studies comparing online and offline programming performance," Communications of the ACM, vol.11, no.1, pp.3–11, 1968.
[9] I. Salman, A.T. Misirli, and N. Juristo, "Are students representatives of professionals in software engineering experiments?" Proc. of International Conference on Software Engineering (ICSE), pp.666–676, 2015.
[10] I. Stuart-Hamilton, The Psychology of Ageing: An Introduction, Jessica Kingsley Publishers, 2012.
[11] H.A. Taub, "Coding for short-term memory as a function of age," Journal of Genetic Psychology, vol.125, no.1, pp.309–314, 1974.
[12] T. Thelin, C. Andersson, P. Runeson, N. Dzamashvili-Fogelström, "A Replicated Experiment of Usage-Based and Checklist-Based Reading," Proc. of International Symposium on Software Metrics, pp.246–256, 2004.
[13] P.M. Uesbeck, A. Stefik, S. Hanenberg, J. Pedersen, and P. Daleiden, "An empirical study on the impact of C++ lambdas and programmer experience," In Proc. of International Conference on Software Engineering (ICSE), pp.760–771, 2016.
[14] V. Wadhwa, Silicon Valley's Dark Secret: It's All About Age, 2010, http://techcrunch.com/2010/08/28/silicon-valley%E2%80%99s-dark-secret-it%E2%80%99s-all-about-age/.
[15] P. Zaphiris, M. Ghiawadwala, and S. Mughal, "Age-centered research-based web design guidelines," Proc. of CHI '05 Extended Abstracts on Human Factors in Computing Systems (CHI EA), pp.1897–1900, 2005.