The Effect of Information Content and Length on Name Recollection

Asaf Etgar*
asafetgar@gmail.com
The Hebrew University
Jerusalem, Israel

Ram Friedman*
ram.friedman@mail.huji.ac.il
The Hebrew University
Jerusalem, Israel

Shaked Haiman*
shakedhaiman@gmail.com
The Hebrew University
Jerusalem, Israel

Dana Perez*
danaperez2891@gmail.com
The Hebrew University
Jerusalem, Israel

Dror G. Feitelson
feit@cs.huji.ac.il
The Hebrew University
Jerusalem, Israel

ABSTRACT

Memorable function and variable names are useful for developers: they reduce the need to re-check how objects are named when one wants to use them, and they enhance comprehension when encountered when reading code. We look at the possible interplay between the information contained in names and how memorable they are. We show in two independent experiments involving a total of 190 subjects that informative names are usually easier to recollect than similar-length names which contain less focused information. Interestingly, we find that less-experienced and female participants are better at remembering the less informative names. We also find that short names, which are not just abbreviated but actually contain less information, are significantly more memorable. Hence a good choice would be to use the the shortest name that includes the most focused and pertinent information.

CCS CONCEPTS

• **General and reference** → **Experimentation**; • **Software and its engineering** → **Software organization and properties**.

KEYWORDS

Variable names, function names, name recollection, code comprehension

ACM Reference Format:
Asaf Etgar, Ram Friedman, Shaked Haiman, Dana Perez, and Dror G. Feitelson. 2022. The Effect of Information Content and Length on Name Recollection. In *30th International Conference on Program Comprehension (ICPC '22), May 16–17, 2022, Virtual Event, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3524610.3529159

*These authors contributed equally to the research.

ICPC '22, May 16–17, 2022, Virtual Event, USA

ACM ISBN 978-1-4503-9298-3/22/05…$15.00
https://doi.org/10.1145/3524610.3529159

1 INTRODUCTION

Program comprehension is the dominant activity performed by software developers: two recent studies have found that developers spend no less than 58% and 70% of their time on average on comprehension, and only some 5% actually editing [30, 43]. And names are the most important facilitators of comprehension — without them it is very hard to create a picture of what the code does [35]. Moreover, in large open source projects about a third of the tokens are identifiers, and they account for about two thirds of the characters in the source code [13]. These statistics explain why so much emphasis is placed on using meaningful names for variables, data structures, and functions.

However, it is hard to pin down exactly what makes names "meaningful", and how this can be measured. While there has been significant work on this issue (see Related Work below), our focus is on the possible interplay between information content and memorability. Studies of what developers do in practice indicate that names are becoming longer over the years [19, 24]. But this may involve an implicit tradeoff:

- Names are important for **understanding**. Longer names, which contain more relevant information and present it in a more readable form can be expected to promote the understandability of code [37].
- Working memory is limited [12], so longer names may be harder to **remember**. This means that developers will need to check the names of objects they use, which may cause reduced productivity. It has therefore been suggested that good names should have limited length [9].

However, suggested approaches to creating good names seem to place a premium on understanding, and do not consider length as a constraint [11, 17].

Note that the actual information content of names can be changed in two ways:

- By adding or removing words, which reflects the selection of concepts to include in the name [17].
- By using focused or more general words, which reflects the decision of how to represent these concepts [13].

This allows us to directly study the tradeoff alluded to above, and investigate whether using a long and detailed name will help developers remember the name and recall it when needed, or perhaps this is counterproductive and the extra words in long names actually

make such names harder to recall. We do this with two independent controlled experiments, which adds to the validity of the results. The experiments required subjects to recall names they had seen before, under experimental treatments that differ in the choice of words and in one experiment also in their number. The results indicate that focused and informative names tend to be remembered better than similar names with more general words, but shorter names with fewer words are much easier to recall.

In summary, the contributions of this paper are:

(1) To identify a distinction between name length and information content;
(2) To show that among similar-length names, those with more focused information tend to be more memorable;
(3) To show that short variable names are significantly more memorable despite having a lower information content;
(4) To suggest that comparison metrics used in name recollection studies should discount word order changes.

2 RELATED WORK

A large amount of research recognizes the importance of names for conveying meaning and enabling code comprehension. For example, method names have been evaluated based on their consistency with what the methods actually do [1, 27]. For variables, the equivalent approach is to look for inconsistencies between their definition and use [3, 14]. The renaming of variables has also been studied [4, 33]. If the developers of the software take the trouble to refactor the names, it means that they thought that the names were not as good as they can be. Analyzing the names can then shed light on what the differences mean. There has also been significant work on the structure of names, meaning their composition and the parts of speech used in them, as well as the metaphors they represent [8, 10, 26, 31].

Of special relevance to our work are studies concerned with the length of names and whether this affects comprehension. As in our work, the underlying hypothesis is usually that longer and more detailed names convey more information, and therefore aid comprehension; however, they may also cause clutter and take longer to read, which may have detrimental effects. Several studies have considered the use of abbreviations as a compromise which attempts to convey information but keep it short [6, 18, 22, 23, 36, 41]. In many cases the results were that there was no meaningful effect, but in some it was found that full words indeed lead to better understanding. However, it is hard to compare the different studies, as they use diverse conditions.

A few studies have specifically considered what factors affect the ability to remember names. Lawrie at el. used name recall as an additional metric for the effect of abbreviations [23]. The results indicate that well-chosen abbreviations may be preferable in some situations, as they are easier to remember. Single-letter names, on the other hand, were found to be less memorable despite being the shortest. Binkley et al. also consider the effect of limits on short-term memory [9]. However, their experimental materials are based on chained method calls (e.g. `Thread.currentThread().getName().substring(3)`), and their references to length refer to the lengths of the chains, not the individual named components. While technically the whole chain indeed identifies the final method being called, we feel that it is wrong to call it "the method's name". And when studying how well components of such chains are remembered, noting the dependence on the length of the chain rather than the length of the component itself shifts the focus from the name to the context in which it is embedded. Jones provides an especially extensive discussion of names in his commentary on the C language standard, and includes memorability as one of the most important features [21]. Uniquely, this is discussed in relation to natural language and cognitive processes, including for example being pronounceable.

Note that remembering names is not necessarily the same as learning information. In Bloom's taxonomy of educational objectives, "knowledge"—namely being able to recall facts—is the lowest level. "Comprehension", which reflects actual understanding, is the next level up. But in the case of variables we are concerned with remembering names as labels for code entities (variables and functions). This is closer to the "recognize" activity in the adapted taxonomy used by Oliveira et al. [32]. However, Oliveira et al. actually mean a mastery of the vocabulary of the domain, e.g. being able to identify and name an algorithm, not the vocabulary in which the code is expressed, like variable names. Such names are not learned per se; they are a tool.

An alternative view on the relation between comprehension and memory is that remembering *reflects* comprehension. The justification for this view is the seminal work of Simon and Chase, who showed that expert chess players can easily memorize meaningful chess positions, but are not good at memorizing random placements of chess pieces [40]. Some software engineering studies (from around 40 years ago) therefore also used the ability to memorize and recite a program as a metric for the code's comprehensibility [28, 29, 38, 39]. However, this refers to complete sections of code and not only to names.

3 RESEARCH QUESTIONS

The overarching research question we are concerned with is what makes names good, and what exactly do we mean when we talk about "meaningful names". In this paper we focus on a more concrete question which is what makes names memorable. In this context, our experiments were designed to answer the following research questions:

(1) Are names containing more relevant information more memorable?
(2) Are shorter names more memorable?
(3) Does name recollection depend on demographic factors such as sex and experience?

These questions are applied to the specific cases of variables' and/or functions' names.

4 METHODOLOGICAL CONSIDERATIONS

We start by noting some general methodological considerations. The specific design of each experiment is discussed below in the context of that experiment.

4.1 Experimental Materials

Variable and function names are elements of code. There are many considerations regarding the code to use in experiments [16]. We selected functions that are self-contained and reasonable, in the

sense that they do not perform some contrived operation that is useful only as a programming exercise. Likewise we made an effort to use reasonable names that could occur in real code.

As indicated in our research questions, we are interested in the interplay between two attributes of names, and how they affect developers' ability to recollect these names. These attributes are

- The **length** of the names. Names in computer programs tend to be composed of multiple words. Obviously, the longer the name the more you need to remember.
- The **information content** of the names. It has been conjectured that meaningful names (that is, names that accurately reflect their role in the context of the code) are easier to remember.

The question is how to distinguish between these two effects in experiments. Our approach has two parts. The first, used in both experiments, is to compare pairs of long names of similar length which differ in their information content: one contains focused semantic information, while the other is more general and therefore not as informative. In this our discrimination is less extreme than that used by Binkley et al., who used names either tied to the domain or completely general names with no specific ties [9]. The second part of the approach, used in only one experiment, was to also compare with short names. Importantly, the short names are not derived by abbreviating the words in the long names, but rather by dropping or replacing words. As a result the short names do not just change how information is represented, but explicitly contain *less* information. If they are found to be more memorable this may imply that adding information and making names longer is not a good way to make them more memorable.

Note that we avoid completely meaningless or irrelevant names, as those may create a *standout effect*. For example, if we used the name small_pink_elephant as an alternative to a 3-word informative name, this would be so strange as to be very memorable. Instead, we attempt to use reasonable names, just not focused on the concrete semantics. In addition, we require subjects to recall the names rather than giving them multiple choice questions. This avoids the need to invent alternative wrong answers which might be too transparent. For example, in Binkley et al.'s landmark paper on naming style, the alternatives to the name start_time were smart_time, start_mime, and start_tom [7]. These served to limit the difference to the beginning, middle, or end of the name, but suffered from the confounding factor of being nonsense names. Subjects could then pick start_time not because they remembered it but because it was the only one that made sense.

4.2 Name Similarity Metric

Our experiments are concerned with recalling names. So the dependent variable we need to measure is the degree of similarity between the original name and the name as recalled by the experimental subjects. This is better than using multiple choice questions, where the experimental result is only a right/wrong bit with no indication of *degree* of recollection, and there is a risk of reminding the subjects of the names.

The most commonly used metric for name similarity, which we also use in experiment 2, is the Levenshtein distance [25]. This is an edit-distance metric, which counts the number of single-character add, delete, or change operations that need to be performed to change one name into the other. It provides a crisp mathematical definition of the distance between two names which is intuitive and easy to understand.

Table 1: Comparison of experiments

Factor	*Experiment 1*	*Experiment 2*
Language	Python	C++
Naming style	under_score	camelCase
Named objects	functions	variables
Total functions	5	3
Names to remember	5	6 (2 each)
Number of treatments	2	3
Buffer before recall	unrelated text and question	2 comprehension questions
Recall test	from description	cloze test
Recall metric	Ratcliff-Obershelp	edit distance
Compared treatments	between subjects	within subjects

However, the Levenshtein distance does not necessarily reflect the distance between the names as perceived by human developers. For example, some letters are often confused, while others are very distinct. As a result switching among similar letters can go unnoticed (e.g. simple vs. sinple), but switching among distinct letters would be immediately identified as wrong (e.g. complex vs. compfex). As a possible alternative, we use the default pattern matching algorithm in Python in experiment 1. This is based on the Ratcliff-Obershelp algorithm, which was designed in the context of educational software to be "forgiving and understanding of simple typing mistakes, and allow intelligent responses to erroneous input" [34]. Its score is the relative length of the longest subsequence of identical letters, starting with the longest contiguous subsequence and continuing recursively. However, as there can be more than one such subsequence with the same length, the result depends on the order of the input strings, namely which is first and which is second[1]. To counter this, we use the maximum of both orders.

Another issue is related to the fact that long names are composed of multiple words. One of our findings was that a common form of mistakes in recalling names was to remember the words but change the order in which they appear. For example, if a subject writes the name numElemDiff when the correct name was numDiffElem, the edit distance is 8 (delete 4 letters and re-insert them at a different location), as if only 3 of 11 letters were right. This seems excessively harsh, and we think such an answer should count as "almost right". To accommodate this, we reorder the words manually to the correct order, to see what difference this makes.

4.3 Independent experiments

An important aspect of our methodology is the use of two independent experiments. These experiments were designed and conducted by separate groups of students, working in different contexts, with no interactions or knowledge of each other. This led to multiple methodological differences, as detailed in Table 1. Thus the second

[1] As explained in https://stackoverflow.com/questions/35517353/how-does-pythons-sequencematcher-work.

experiment is not a reproduction of the first one, but rather can be considered a corroborative experiment [15].

The fact that the two groups came up with similar frameworks, used diverse methodologies, and obtained similar results, strengthens our belief in these results. For example, they decided to use different metrics for the quality of the recall: one group used Python's default pattern matching library, and the other used Levenshtein's edit distance. We could discuss the differences between these approaches and what effect they may or may not have. But if the results turn out to be similar, we can simply say that apparently it does not matter very much which metric we use, because the final results are similar. This removes the threat to validity incumbent in relying on a single metric.

However, the differences also mean that the experiments are not directly comparable to each other. For example, this is why results for some of the factors, notably the comparison with short names, come from only one of the experiments.

5 EXPERIMENT 1

In this experiment subjects were presented with 5 functions. They were then distracted by a short text, and then asked to recall the function names.

5.1 Function Selection

As our goal in this project is not to assess comprehension itself, we aimed to make the functions' purpose relatively easy to comprehend without requiring additional context. In this experiment we decided to use implementations of various algorithms from a Python algorithms collection on GitHub[2]. However, we did not want study participants to be able to use auxiliary knowledge of the common names for canonical well-known functions (e.g. binary_search). Therefore we did not choose functions that are well known, but more unusual algorithms with no conventional naming schemes. Additional criteria were that the functions would not be too long, and also that they do not use esoteric external packages, again so that understanding them would not be too challenging.

The functions we chose are listed in Table 2. All the participants were presented with the same 5 functions.

5.2 Choosing Functions Names

In this experiment we needed to give each function two alternative names of similar length. The more informative names were selected as one would write in real life — fully related to the function's goal, descriptive, possibly with some common abbreviations (e.g. "diff" instead of "difference"). As for the alternative names, we performed two pilots before we were satisfied with the results. Our first attempt used completely meaningless names, such as my_function and do_something. In the pilot we found that these names stand out on the background of the code and look suspicious, in that they are obviously not names anyone would use. Our second attempt was therefore to reduce the information in the names by using only the single main concept out of the informative names: name the function multiply instead of multiply_digits, diff instead of sum_square_diff, and so on. However, the results were near perfect recollection, and we realized that these names introduce a confounding factor which might influence memorability, in that they are much shorter. So for the final version we took the good names and stripped them of important information by replacing specific words with more ambiguous words, for example replacing "count" with "process". As a result of this methodology the length of each function's name stayed approximately the same, but the relation to the function's intent was less obvious.

The final informative and more general names we chose for each function are shown in Table 2. In retrospect, some answers we received in the experiment provide testimony that our choices were reasonable. For example, in one case the informative name sum_square_diff was mis-remembered as calc_square_diff, and in another as square_sum_alg. In both cases the participants had unknowingly used the same more general words we had used in the other version of the function name, which was calc_diff_alg.

In the interest of uniformity we created two separate versions of the experiment: one in which all the function names are informative, and another in which all the function names are not indicative of the function's intent. We are interested in the difference in accuracy of recalling functions' names between these two variants. This is therefore a between-subjects comparison. All function names were fully compliant to programming conventions in Python (PEP8), with lowercase letters, words separated by underscores, etc.

5.3 Experiment Execution

The experiment was executed using the Google Forms platform. It consisted of four parts:

(1) An explanation of the experiment, ending with an explicit consent question. This was followed by demographic questions about age, gender, and experience.
(2) Presentation of five different functions. As mentioned above, we have two versions: the first version uses function names that are closely related to the function, and the second version uses more general function names. The functions always appear in the same order. The participants were asked to read the functions carefully. They did not know that we will ask only about their names.
(3) Presentation of a short 3-paragraph text about a casual topic (we selected the lovely topic of penguins!) followed by a comprehension question (why do they waddle?). The goal was to cause an interception in the respondents' thought process between reading the functions and being asked to recollect their names.
(4) An active recall task. The respondents were presented with 5 task descriptions (specifically, the descriptions given in Table 2), and asked to type in the name of the function that performed each task. They were also asked not to go back to previous pages. Note that each respondent is asked about all the functions. The order of the questions was always the same, but it was different from the order in which the corresponding functions were presented.

The experiment was initially distributed by asking friends who are experienced programmers working in the industry to participate and pass it along to more people, in order to expand and diversify the backgrounds of the respondents. Later we also advertised it on

[2] https://github.com/TheAlgorithms/Python

Table 2: Function names used in experiment 1

Informative name	*General Name*	*Description*	*Lines*
multiply_digits	digits_calc	Find the product of the digits of a given number	8
indices_sum_to_target	find_right_numbers	Given an array of integers, return the places of two of them that add up to a specific value	8
words_counter	words_process	Counts words occurrence in a given sentence, and returns a dictionary with these values	6
sum_square_diff	calc_diff_alg	Returns the difference between the sum of the squares of the first n natural numbers and the square of the sum	7
sum_exp_digits	compute_exp_sum	Return the sum of the digits of the number 2^{power}	8

LinkedIn, with a pledge to donate $1 to either of two environmental causes for each filled response we received. The causes were planting trees or cleaning the oceans, and participants could vote which they preferred[3]. There were a total of 131 responses, and about half of them were randomly presented with each version. 82% of the respondents were male. 8% had up to a year of programming experience, 35% had 2 to 4 years, and 57% had 5 or more years of experience. Ages ranged from 20 to 58, with 68% being between 26 and 36 years old.

5.4 Data Processing

All the data entered by the experiment participants was checked manually and classified into two groups:

- Potential names of functions, and
- Non-names, including entries of "I don't know" or equivalent, or nonsense like "yes" or "6".

Names were further subject to the following processing:

- All names were normalized to lower case with words separated by underscores.
- We extracted partial names when participants were unsure of themselves and explained their situation. For example, if a participant wrote "It started with 'count'" we replaced this with just "count". If they wrote "exp_something" we erased the "something" and left only "exp_" (note that underscores were retained, as they were part of the name matching). This was done to avoid losing the answer altogether, and reflect what the participants did remember.
- We corrected the spelling in simple obvious cases. For example, "coubt_words" was changed to "count_words". The justification is that we are interested in name recollection, and not in typing errors.
- Re-ordering words to match the original name had to be done carefully, taking the treatment the participant saw into account. For example, the alternative names of the last function in Table 2 are sum_exp_digits and compute_exp_sum. Consequently the correct order of exp and sum depends on the treatment. We also changed the order to retain word sequences and words that were either the first or the last in the name.

This whole process was performed by one of the authors and then verified by another.

[3]Due to a technical problem all the donations were eventually made to planting trees.

After processing the results as described above we measured the similarity between the given names and the correct answers, using the Python SequenceMatcher class. The distributions of results under the different treatments were then compared to see if they were different from each other. We first used the Shapiro-Wilk test of normality on each distribution. As shown below, all the distributions have strong modes at various places, especially at 0 and at 1, and they therefore failed this test. Consequently we used the Mann-Whitney non-parametric test to test the null hypothesis that the distributions are equal. While we expect the similarities when names are informative to be higher, we nevertheless used the two-sided test, as we cannot be sure our expectation is correct. This was applied separately to the similarities of the original names and the reordered names for each function. The significance level used was $\alpha = 0.05$.

5.5 Results

The similarity results derived by the procedure described above are shown using solid lines in Figure 1. These are CDF plots (cumulative distribution functions). The horizontal axis is the score, where 0 means the answer is completely unrelated to the original name, and 1 means that they are identical. The vertical axis is the cumulative fraction of answers up to a certain score. A CDF that is below and shifted to the right relative to another indicates that the distribution tends to include higher score values.

As can easily be seen, in 4 of the 5 functions using informative names led to a score distribution with higher values than when using more general names. This indicates that the informative names were easier to recollect correctly. In 3 of them the difference was statistically significant according to the Mann-Whitney test, with $p < 0.001$ for the words_counter and sum_square_diff functions, but only $p = 0.043$ for the multiply_digits function. The result for sum_exp_digits was $p = 0.088$, which is pretty low but not enough to qualify under a two-sided test.

Only in one function, the one that finds the indices of two array elements that sum to a given target, the distributions are overlapping. We note that his function has a longer name and involves a task that is more specific and more complex than those featured in the other functions. Therefore we posit that the difference in name recollection might be a result of longer names' length, difference in the number of words in the name, and complexity of the function. Further research is needed to settle these hypotheses.

Some of the graphs in Figure 1 contain large steps in the CDF. Such steps correspond to modes in the distribution of scores, namely

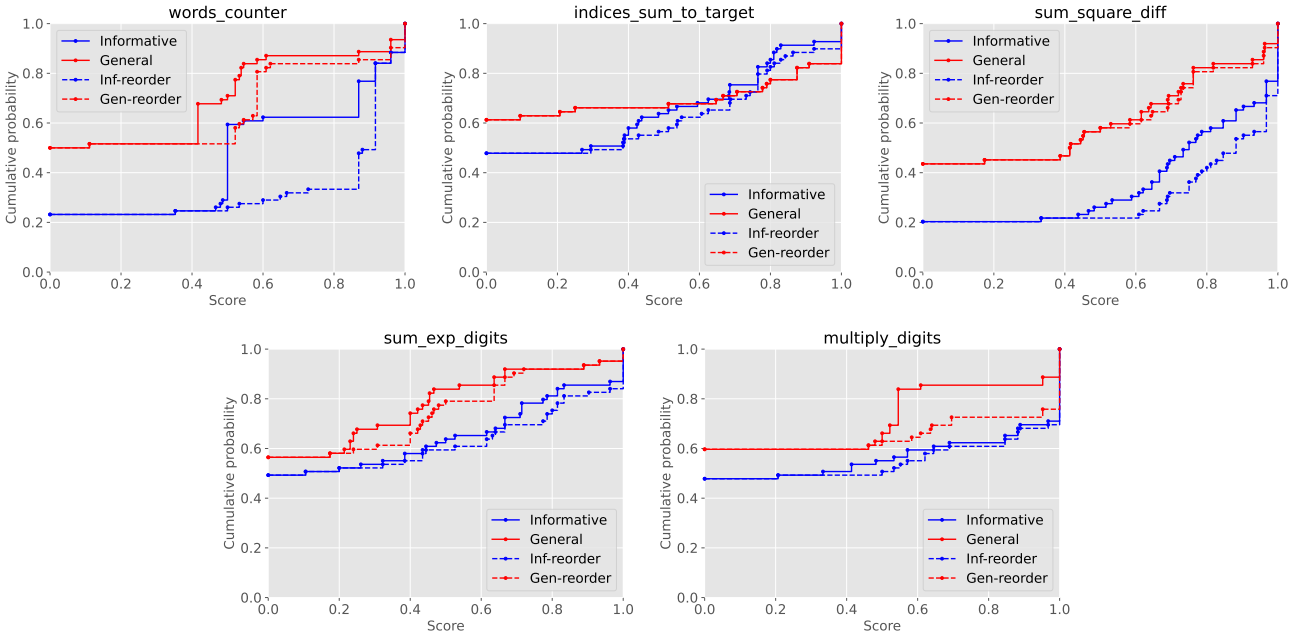

Figure 1: CDFs comparing the distributions of scores achieved with the different names for the 5 functions.

many answers that received exactly the same score, probably because they made exactly the same mistake. The biggest such step occurs in the words_counter function (informative version), where a common mistake was to reverse the order and think that the name was count_words. The same thing happened in digits_calc (general version of multiply_digits), which was mistakenly recalled as calc_digits.

To investigate the influence of word order, we also checked the scores of answers with the words reordered to match the original names. The results are shown with the dashed lines of Figure 1. In all cases but one such reordering led to higher scores, as may be expected. But in all cases the advantage of informative names over the more general names was retained. In other words, informative names were remembered better whether or not word order was considered. For two functions, words_counter and sum_square_diff, the difference was statistically significant under the Mann-Whitney test, with $p < 0.001$ (as had been the case for the similarities of the original names without reordering of words). For sum_exp_digits we got $p = 0.100$, and for multiply_digits it was $p = 0.208$, which is far from significance. This is not surprising given the distribution shown in Figure 1, as one can see that the reordering removed a large mode in the distribution of scores for the more general names, thereby making it much closer to the distribution of scores for the informative names.

The single case in which reordering did not make a difference is the general name of the indices_sum_to_target function. The more general name we had chosen was find_right_numbers, and none of the experiment participants recalled these words in an incorrect order. This is probably due to the fact that alternative orders just do not sound right. Consequently this suggests another rule for good naming: strive for word orders that can not be confused. Considering the previous examples in which reordering did occur, it seems that projects may also benefit from explicit naming rules concerning the order in noun-verb combinations.

The end points — a score of 0 or a score of 1 — were also significant modes in all cases. The bigger mode is at 0, that is, the participants did not remember the names at all (or "remembered" names that were completely wrong). In two of the questions this was 20–50% of the responses, and in the other three is was 50–60%. In all cases the non-informative general names suffered more from such forgetfulness. But the high rates of not being able to recall anything may imply that the task of remembering 5 names you only saw once, after being distracted by penguins, may be too hard.

Another way to look at the results is to pool all the results from all 5 functions, and consider the effect of demographic variables. The results of doing so are shown in Figure 2, for gender and self-reported years of experience. As may be expected, within each demographic group higher scores were obtained when informative function names were given. But the gap between the scores for informative names and for more general names differed. When considering gender, we see that the gap for males was much larger than the gap for females; for females the two CDFs even coincide for the top scores. When considering experience, we see that the gap for experienced participants, those with 5 or more years of experience, was much bigger than the gap for inexperienced participants, who had up to 4 years of experience.

In other words, we observe an interaction between the treatments and the demographic variables. Focusing on the more informative names (the blue lines in Figure 2), we see that in both cases — females vs. males, or inexperienced vs. experienced — the differences are rather slim. But wider differences are observed for the

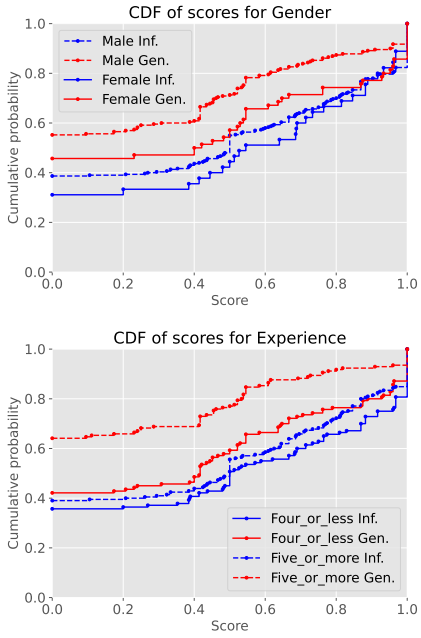

Figure 2: CDFs comparing the distributions of scores achieved by different demographic groups.

more general names. Consequently, when we say that females recollect names better than males, most of the difference comes from their ability to recollect the lower-quality general names better than males. The same holds for inexperienced participants compared with more experienced ones. This is supported by the statistical tests applied to the distributions of scores. Statistically significant differences were found between the distributions of scores for the two name types for males and for experienced developers, and also between the sexes and between the experience groups for the more general names, with $p = 0.036$ for the difference between sexes and $p < 0.001$ in the 3 other cases.

The observed differences mean that it is not the case that females and inexperienced participants just work more diligently, or that males and experienced participants just paid less attention to names. It is that female and inexperienced participants were specifically better at recalling the more general names. We do not have an explanation for this effect, but note that differences between reading patterns of men and women have been observed in other contexts, e.g. code reviews [20].

6 EXPERIMENT 2

In this experiment subjects were presented with 3 functions. For each one they were then requested to answer 2 comprehension-related questions, and to recall 2 variable names.

6.1 Function and Names Selection

The functions used in the second experiment were C++ functions selected from leetCode.com. This site allows programmers to exercise their coding skills and prepare for job interviews. It contains a few thousand problems with dozens of contributed solutions for each. The advantage of functions implementing solutions to such problems is that they are self-contained, and do not need any context. This makes them more comprehensible in an experimental setting.

We initially chose 10 functions which solve diverse programming problems. An important criterion was that the functions solve a realistic challenge, and not a contrived one that was just made up as an exercise. For example, we rejected a function calculating the distribution of candies to children according to some strange rule, because it would be hard to understand why the function does what it does even if the calculation is actually straightforward. In addition we required functions to be from easy or medium difficulty questions (as classified by the site), 8–15 lines long, and with 3–4 variables. Next, we filtered out functions that did not have interesting variables that could justify a longer informative name, as well as functions that could be easily identified as canonical programming problems (e.g. binary search). This left us with 6 functions. However, a pilot study with 3 participants indicated that the average time to answer the questions on all 6 functions was 27 minutes, which was too tedious. We therefore decided to limit the experiment to only 3 functions. Each subject received all 3 functions, but with 3 different treatments.

The variable names in the different treatments were designed to investigate what has a stronger effect on a name's recollection: its length (it is harder to recall a long name) or its content (it is easier to recall a meaningful name). Unlike in experiment 1, in this experiment we also used short names. This led to a total of 3 treatments, as shown in Table 3. The difference between the short and long informative names was added words that add information to the name. The Long general names were used as a control. These were the hardest to derive, as we needed to add words that match the context but do not add pertinent information.

The functions and treatments were organized using a Graeco-Latin square design, so that each function and each treatment had equal probabilities to appear first, second, or third independent of each other. As each subject saw functions in all 3 treatments, the results had an element of within subjects comparison.

6.2 Experiment Execution

The experiment proceeded along the following steps:

(1) The first page explained that this is an experiment with questions about C++ functions, requested subjects to participate only if they have programming experience, requested them not to go back to look at the functions when answering the questions, and notified them that advancing to the questions constitutes informed consent.
(2) The next two pages presented an example function followed by two questions, one about what it does and the other about one of the variables in it. This example was the same for all subjects.
(3) Then came the 3 blocks of the experiment itself. Each block started with a page showing one of the experiment functions in one of the treatments. Then came a separate page with 4 questions, 2 about the function and 2 about variables.

Questions about the functions were multiple choice questions about their functionality, the return value, the number of loops in it, or

Table 3: Functions and variable names used in experiment 2

	Variable names			
Function signature and description	*Informative*	*General*	*Short*	*Lines*
int removeDuplicatesFromSortedArray(vector<int> &nums)	numOfUniqElements	selectionOfObjects	ans	15
Remove duplicates from a sorted array; returns new array size	totalArraySize	objectMaxPoint	size	
int maxSubArray(vector<int> &arr)	currSubArraySum	dataOfArgument	curr	13
Sum elements in all subarrays and return the maximum	maxSubArraySum	resOfArgument	sum	
int firstOccurrence(string string1, string string2)	subStrIdx	uniqueInt	index	10
Return index of first occurrence of second string in first string	currSubString	withinElement	subStr	

which comparison operator was used in a condition. Questions about variable names were presented as short code snippets with the variable name removed (cloze test). This approach identifies the variable in question without giving away any information about it, thereby avoiding a threat to validity that existed in experiment 1, where using a verbal description of the variable's role might give an unfair advantage to informative names that attempt to capture exactly that role. The time to read the functions and answer the questions was not limited, to avoid the possible confounding effect of wrong answers due to lack of time. The questions about the function always appeared before the name recollection questions. Thus they served as a buffer between reading the function and needing to recollect the names.

In total 59 subjects participated in the experiment. Subjects were recruited using invitations in relevant groups, and were either working in the high-tech industry or students majoring in a computing-related area. However, due to an error, no precise demographic data was collected.

6.3 Data Processing

Names were normalized to lowercase with words separated by underscores, and word order was corrected manually, as described above.

As a metric we did not use the edit distance directly, but rather the relative similarity. Edit distance counts the number of letters that need to be changed to transform the given answer to the original name. This naturally handicaps longer names as there are more letters that can change relative to short names. Denoting the edit distance by $edDist$, and the length of the original name by $\text{len}(origName)$, we therefore define the relative edit distance as $edDist/\text{len}(origName)$ — that is, the fraction of letters that need to change[4]. We also invert the scale by subtracting the relative edit distance from 1 so that high values reflect greater similarity. Note, however, that the edit distance may be larger than the original name length, for example when a subject adds a long word that was not present in the original name. This leads to a relative edit distance above 1, and a negative similarity score. To avoid this we define the similarity score according to the following formula:

$$similarity = 1 - \min\left(1, \frac{edDist}{\text{len}(origName)}\right)$$

[4]This is different from Tashima et al.'s normalization [42] in that we give a special role to the original name.

The distributions of results using the different names were then compared to see if they were different from each other. We again used the Shapiro-Wilk test of normality on each distribution. As in experiment 1, all the distributions have strong modes, and they therefore failed this test. We then used the Kruskal-Wallis test, which is the generalization of the Mann-Whitney test when more than 2 distributions are being compared. The significance level used was $\alpha = 0.05$.

6.4 Results

The distributions of similarity scores for the names in all treatments and all functions are shown in Figure 3. As in the first experiment, these are CDFs. The horizontal axis is the similarity score, calculated as described above. The vertical one is the cumulative fraction of answers up to a certain similarity score. Recall that in each function there were questions about the names of 2 variables. The results for both variables are shown together.

In the two left-most graphs we see that the red lines, representing long general (non-informative) names, go up faster than the others. This means that the similarity scores for these names had lower values, like in experiment 1. In the third function, in contradistinction, the general names got higher similarity scores. However, in this case the distance between the distributions was relatively slim. In all cases where there was a difference, the CDFs of the results after reordering the words (represented by the dashed lines) are lower than those for the original results (solid lines), indicating slightly higher similarity scores as expected.

As the number of participants in this experiment was relatively low we analyzed all the results from all the variables in all the functions together. We made two 3-way comparisons: one comparing the achieved similarity to the short names with the achieved similarity to the original informative and more general names, and the other comparing the achieved similarity to the short names to that of the reordered informative and more general names. All the differences between all the compared versions were statistically significant. The differences between the distribution of similarity to short names to the other distributions was very significant, with $p < 0.001$ in all 4 cases. The differences between the distributions of similarity to the informative and more general names had $p = 0.016$, and after reordering the significance was slightly better, with $p = 0.011$. Both these results remain significant after Bonferroni correction for making 3 comparisons.

The gap between the point where each line hits the right end of the graph and the top indicates the fraction of responses that were

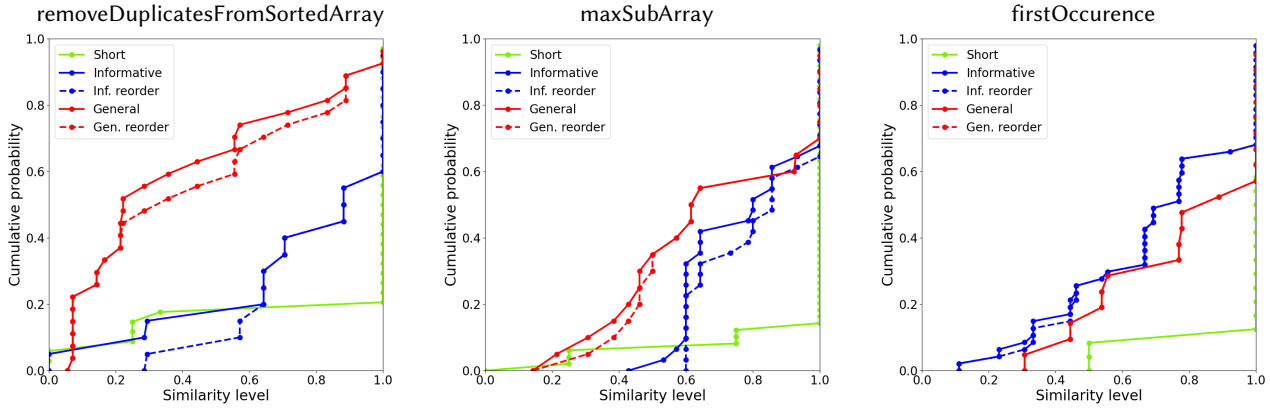

Figure 3: CDFs comparing the distributions of scores achieved with the different variable names in the 3 functions.

Table 4: Descriptive statistics of results in experiment 2

Function	*Questions correctness rate*	*Names average similarity*	*Names completely correct*
Short names			
removeDuplicates	88.2%	0.826	79.4%
maxSubArray	85.2%	0.918	85.9%
firstOccurence	85.7%	0.937	86.4%
Long informative names			
removeDuplicates	78.6%	0.728	40.0%
maxSubArray	88.8%	0.779	32.5%
firstOccurence	96.4%	0.697	31.6%
Long general names			
removeDuplicates	69.2%	0.504	22.5%
maxSubArray	75.6%	0.660	30.0%
firstOccurence	73.3%	0.768	41.7%

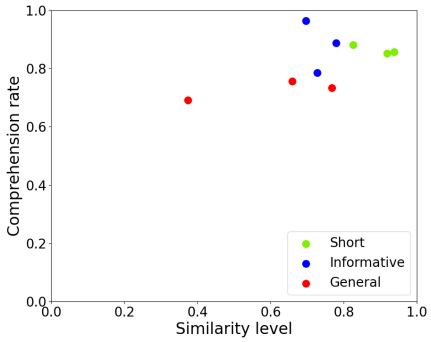

Figure 4: Scatter plot showing relation of recollection and comprehension results.

100% correct (also shown in the rightmost column of Table 4). For the short names (the green lines) this is around 80–86% in the different functions. For the long names it is in the range 22–42%. Thus short names enjoy a much higher fraction of completely correct recollections (as was also observed in one of the pilot studies of experiment 1). This could be simply because there was so little room for failure — if you remember such short names, you remember them fully and correctly.

The above results focus on the name recollection. But we also had 2 comprehension-related questions about each function. We can therefore check for a possible relationship between variable naming and comprehension. The results are shown in Figure 4 (and the data in Table 4). This is a scatter plot. The horizontal axis represents the average similarity score. The vertical axis represents the rate of correct answers to the comprehension questions. Each function is represented by three points, for the three treatments. Interestingly, these points appear to cluster according to the treatment and not according to the function. The red dots, representing general non-informative names, had slightly lower correctness and also slightly lower similarity. The informative names and the short names led to slightly higher correctness, and the short ones had higher similarity than the informative ones.

7 THREATS TO VALIDITY

Construct Validity refers to the degree we can be sure we are measuring what we set out to measure. We were testing the ability to recall the name of a variable or function. In experiment 1 we described the functions and asked the participants to recall the name of each function, so it could be that we were testing their ability to deduce the function's name from its description. Also, we were testing short-term memory. It may be that the distraction paragraph in experiment 1, and the comprehension questions in experiment 2, were not long enough to achieve their purpose.

Another potential problem is that some of the functions were too similar to each other, so there were a few cases where the subjects were confused and mixed the names. For example, multiply_digits and sum_exp_digits both operate on digits of a number. So one subject gave the answer multiply_digits_exponent when asked to recall the function that finds the sum of the digits of an exponent. We had not anticipated such cross-function confusion when we selected the functions and their names.

In experiment 1 we also found that male participants and experienced participants performed particularly badly on the more general names. This could be at least partly because they simply paid less attention to the function names, and preferred to focus on their bodies.

Finally, the platform we used does not prevent participants from going back to look at the names they saw previously. We asked them not to do so, but we cannot be sure they complied. However, the wide distribution of mistakes made testifies that it is unlikely that they looked back.

Internal Validity refers to the degree to which we can be sure our conclusions follow from the measurements. We conducted controlled experiments, and attempted to change only one factor between treatments. However, it is not possible to truly control everything. Numerous additional factors can influence the ability to remember names, including, among others, mood, tiredness, and health at the time of the experiment. Since the experiment was conducted remotely we could not make sure that all participants were focused while answering it or took sufficient time to actually understand the functions.

More concretely, our goal was to study names with different lengths and different information content. There can be no debate regarding the length, namely that the short names are indeed shorter. However, we cannot be so sure about the information content. We tried to distinguish between focused informative names and more general names. However, names can mean different things to different people [5]. Consequently, it may be some other factor which causes the effect. Note, too, that in some of the results we failed to see the main effects. This may testify to differences in the actual characteristics of the names in those instances of the experiments.

The opposite problem can also occur. In experiment 1 we provide text descriptions as a prompt to elicit the function names. While we attempted to keep these descriptions neutral, they may still be closer to the informative names than to the general names, thus giving an unfair advantage to the informative names. However, experiment 2 used a different methodology which does not suffer from this problem and produced similar results, suggesting that this threat did not in fact affect the results. This is an example of how using two experiments with different designs helps improve validity.

Scope may also have an effect. Our experiments involved functions with a relatively short scope, so maybe that is why short variable names were found to be better than long ones. To check this we need to repeat the experiment with a larger scope, for example using class data members.

External Validity refers to the degree to which the research can be generalized, namely whether the results can be expected to hold beyond the specific experimental conditions that were used. We only had 190 subjects in both experiments, which is not a very high number.

In addition, our recall task deals with short term memory, so the result might not be relevant for long term memory of variable and function names. The functions we used also appeared without context and without documentation. Such a setting might not be representative of a "real-life" scenario for software-engineers.

8 DISCUSSION AND CONCLUSIONS

To summarize, our main result is that detailed names with information that is focused on an object's essence appear to be remembered better than more general names of approximately the same length. This was the case for 4 of 5 function names in experiment 1, and for the variable names in 2 of 3 functions in experiment 2. However, in experiment 2 we also found that short names are remembered much better, despite the fact that they contain less information. In addition, in experiment 1 we found that female and inexperienced participants remember names better than male and experienced participants, and the difference is especially pronounced for long general names.

An interesting question is whether naming presents the developer with a tradeoff. Does piling more information on a name, thereby making it longer and harder to remember, nevertheless help to comprehend the code? And conversely, if we want a name to be easy to remember, and strip it from information to make it shorter, does this impair comprehension? Our experiments were not explicitly designed to investigate the effect of the names on comprehension. All we have at present is the comprehension questions of experiment 2, which suggest that there is no significant difference between the comprehension of codes that employ informative long or short names. If true, this would imply that we are not facing a tradeoff: short names are easier to remember, and do not jeopardize comprehension. However, substantial additional work, using diverse code and comprehension metrics, is needed to better support this tentative result.

In any case, the finding that short names are better remembered implies the need to carefully choose what to include in a name. Heaping additional words may have a cost associated with it. Thus the addition of information to a name should be balanced with other considerations, like long names causing clutter, and the scope where the name is used [2]. The recommendation to developers is that they should prefer focused names over general ones, and if they are concerned with memorability, they should prefer short names.

The effect of information content on name memorability appears to be more subtle than the effect of length. We gave initial results on the importance of conciseness, namely that names be precise and focused. More work like this is needed, using memorability as as additional perspective to consider when measuring comprehension and identifier name quality in varying ways (e.g., considering linguistic anti-patterns, grammar patterns, or word usage consistency).

The most interesting future research would be to investigate the link between memory and comprehension. Memorability can probably help comprehension efficiency — if you remember things you do not need to re-check them, so comprehension will be faster and less tedious. Checking this is hard and will require well-designed experiments. One initial idea is to use eye tracking to see how often people look back at declarations as a function of the names' quality.

EXPERIMENTAL MATERIALS

Complete experimental materials and results are available online using the following link:

https://doi.org/10.5281/zenodo.6387422

ACKNOWLEDGMENTS

This research was supported by the ISRAEL SCIENCE FOUNDATION (grant no. 832/18).

REFERENCES

[1] M. Allamanis, E. T. Barr, C. Bird, and C. Sutton, "Suggesting accurate method and class names". In 10th *Joint ESEC/FSE*, pp. 38–49, Sep 2015, DOI: 10.1145/2786805.2786849.

[2] H. Aman, S. Amasaki, T. Yokogawa, and M. Kawahara, "A large-scale investigation of local variable names in Java programs: Is longer name better for broader scope variable?" In 14th *Quality of Inf. & Commun. Tech.*, pp. 489–500, Sep 2021, DOI: 10.1007/978-3-030-85347-1_35.

[3] V. Arnaoudova, M. Di Penta, and G. Antoniol, "Linguistic antipatterns: What they are and how developers perceive them". *Empirical Softw. Eng.* **21(1)**, pp. 104–158, Feb 2016, DOI: 10.1007/s10664-014-9350-8.

[4] V. Arnaoudova, L. M. Eshkevari, M. Di Penta, R. Oliveto, G. Antoniol, and Y.-G. Guéhéneuc, "REPENT: Analyzing the nature of identifier renamings". *IEEE Trans. Softw. Eng.* **40(5)**, pp. 502–532, May 2014, DOI: 10.1109/TSE.2014.2312942.

[5] E. Avidan and D. G. Feitelson, "Effects of variable names on comprehension: An empirical study". In 25th *Intl. Conf. Program Comprehension*, pp. 55–65, May 2017, DOI: 10.1109/ICPC.2017.27.

[6] G. Beniamini, S. Gingichashvili, A. Klein Orbach, and D. G. Feitelson, "Meaningful identifier names: The case of single-letter variables". In 25th *Intl. Conf. Program Comprehension*, pp. 45–54, May 2017, DOI: 10.1109/ICPC.2017.18.

[7] D. Binkley, M. Davis, D. Lawrie, and C. Morrell, "To CamelCase or under_score". In 17th *Intl. Conf. Program Comprehension*, pp. 158–167, May 2009, DOI: 10.1109/ICPC.2009.5090039.

[8] D. Binkley, M. Hearn, and D. Lawrie, "Improving identifier informativeness using part of speech information". In 8th *Working Conf. Mining Softw. Repositories*, pp. 203–206, May 2011, DOI: 10.1145/1985441.1985471.

[9] D. Binkley, D. Lawrie, S. Maex, and C. Morrell, "Identifier length and limited programmer memory". *Sci. Comput. Programming* **74(7)**, pp. 430–445, May 2009, DOI: 10.1016/j.scico.2009.02.006.

[10] S. Butler, M. Wermelinger, and Y. Yu, "A survey of the forms of Java reference names". In 23rd *Intl. Conf. Program Comprehension*, pp. 196–206, May 2015, DOI: 10.1109/ICPC.2015.30.

[11] B. Caprile and P. Tonella, "Restructuring program identifier names". In *Intl. Conf. Softw. Maintenance*, pp. 97–107, Oct 2000, DOI: 10.1109/ICSM.2000.883022.

[12] N. Cowan, "The magical mystery four: How is working memory capacity limited, and why?" *Current Directions Psychological Sci.* **19(1)**, pp. 51–57, Feb 2010, DOI: 10.1177/0963721409359277.

[13] F. Deissenboeck and M. Pizka, "Concise and consistent naming". *Softw. Quality J.* **14(3)**, pp. 261–282, Sep 2006, DOI: 10.1007/s11219-006-9219-1.

[14] S. Fakhoury, D. Roy, Y. Ma, V. Arnaoudova, and O. Adesope, "Measuring the impact of lexical and structural inconsistencies on developers' cognitive load during bug localization". *Empirical Softw. Eng.* **25(3)**, pp. 2140–2178, May 2020, DOI: 10.1007/s10664-019-09751-4.

[15] D. G. Feitelson, "From repeatability to reproducibility and corroboration". *Operating Syst. Rev.* **49(1)**, pp. 3–11, Jan 2015, DOI: 10.1145/2723872.2723875.

[16] D. G. Feitelson, "Considerations and pitfalls in controlled experiments on code comprehension". In 29th *Intl. Conf. Program Comprehension*, pp. 106–117, May 2021, DOI: 10.1109/ICPC52881.2021.00019.

[17] D. G. Feitelson, A. Mizrahi, N. Noy, A. Ben Shabat, O. Eliyahu, and R. Sheffer, "How developers choose names". *IEEE Trans. Softw. Eng.* **48(1)**, pp. 37–52, Jan 2022, DOI: 10.1109/TSE.2020.2976920.

[18] J. C. Hofmeister, J. Siegmund, and D. V. Holt, "Shorter identifier names take longer to comprehend". *Empirical Softw. Eng.* **24(1)**, pp. 417–443, Feb 2019, DOI: 10.1007/s10664-018-9621-x.

[19] G. J. Holzmann, "Code clarity". *IEEE Softw.* **33(2)**, pp. 22–25, Mar/Apr 2016, DOI: 10.1109/MS.2016.44.

[20] Y. Huang, K. Leach, Z. Sharafi, N. McKay, T. Santander, and W. Weimer, "Biases and differences in code review using medical imaging and eye-tracking: Genders, humans, and machines". In 28th *ESEC/FSE*, pp. 456–468, Nov 2020, DOI: 10.1145/3368089.3409681.

[21] D. M. Jones, "The new C standard: An economic and cultural commentary", 2009. URL http://www.knosof.co.uk/cbook/cbook1_2.pdf.

[22] D. Lawrie, H. Feild, and D. Binkley, "Quantifying identifier quality: An analysis of trends". *Empirical Softw. Eng.* **12(4)**, pp. 359–388, Aug 2007, DOI: 10.1007/s10664-006-9032-2.

[23] D. Lawrie, C. Morrell, H. Field, and D. Binkley, "Effective identifier names for comprehension and memory". *Innovations in Syst. & Softw. Eng.* **3(4)**, pp. 303–318, Dec 2007, DOI: 10.1007/s11334-007-0031-2.

[24] O. A. L. Lemos, M. Suzuki, A. C. de Paula, and C. Le Goes, "Comparing identifiers and comments in engineered and non-engineered code: A large-scale empirical study". In 35th *ACM Symp. Applied Computing*, pp. 100–109, Mar 2020, DOI: 10.1145/3341105.3373972.

[25] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals". *Soviet Physics – Doklady* **10(8)**, pp. 707–710, Feb 1966.

[26] B. Liblit, A. Begel, and E. Sweetser, "Cognitive perspectives on the role of naming in computer programs". In 18th *Psychology of Programming Workshop*, pp. 53–67, Sep 2006.

[27] K. Liu, D. Kim, T. F. Bissyandé, T. Kim, K. Kim, A. Koyuncu, S. Kim, and Y. Le Traon, "Learning to spot and refactor inconsistent method names". In 41st *Intl. Conf. Softw. Eng.*, May 2019, DOI: 10.1109/ICSE.2019.00019.

[28] T. Love, "An experimental investigation of the effect of program structure on program understanding". *Softw. Eng. Notes* **2(2)**, pp. 105–113, Mar 1977, DOI: 10.1145/390019.808317.

[29] K. B. McKeithen, J. S. Reitman, H. H. Reuter, and S. C. Hirtle, "Knowledge organization and skill differences in computer programmers". *Cognitive Psychol.* **13(3)**, pp. 307–325, Jul 1981, DOI: 10.1016/0010-0285(81)90012-8.

[30] R. Minelli, A. Mocci, and M. Lanza, "I know what you did last summer: An investigation of how developers spend their time". In 23rd *Intl. Conf. Program Comprehension*, pp. 25–35, May 2015, DOI: 10.1109/ICPC.2015.12.

[31] C. D. Newman, R. S. AlSuhaibani, M. J. Decker, A. Peruma, D. Kaushik, M. W. Mkaouer, and E. Hill, "On the generation, structure, and semantics of grammer patterns in source code identifiers". *J. Syst. & Softw.* **170**, art. 110740, Dec 2020, DOI: 10.1016/j.jss.2020.110740.

[32] D. Oliveira, R. Bruno, F. Madeiral, and F. Castor, "Evaluating code readability and legibility: An examination of human-centric studies". In *Intl. Conf. Softw. Maintenance & Evolution*, pp. 348–359, Oct 2020, DOI: 10.1109/ICSME46990.2020.00041.

[33] A. Peruma, M. W. Mkaouer, M. J. Decker, and C. D. Newman, "Contextualizing rename decisions using refactorings, commit messages, and data types". *J. Syst. & Softw.* **169**, art. 110704, Nov 2020, DOI: 10.1016/j.jss.2020.110704.

[34] J. W. Ratcliff and D. E. Metzener, "Pattern matching: The gestalt approach". *Dr. Dobbs J.* Jul 1988. Https://www.drdobbs.com/database/pattern-matching-the-gestalt-approach/184407970?pgno=5.

[35] F. Salviulo and G. Scanniello, "Dealing with identifiers and comments in source code comprehension and maintenance: Results from an ethnographically-informed study with students and professionals". In 18th *Intl. Conf. Evaluation & Assessment in Softw. Eng.*, art. 48, May 2014, DOI: 10.1145/2601248.2601251.

[36] G. Scanniello, M. Risi, P. Tramontana, and S. Romano, "Fixing faults in C and Java source code: Abbreviated vs. full-word identifier names". *ACM Trans. Softw. Eng. & Methodology* **26(2)**, art. 6, Oct 2017, DOI: 10.1145/3104029.

[37] A. Schankin, A. Berger, D. V. Holt, J. C. Hofmeister, T. Riedel, and M. Beigl, "Descriptive compound identifier names improve source code comprehension". In 26th *Intl. Conf. Program Comprehension*, pp. 31–40, May 2018, DOI: 10.1145/3196321.3196332.

[38] B. Shneiderman, "Exploratory experiments in programmer behavior". *Intl. J. Comput. Inf. Sci.* **5(2)**, pp. 123–143, Jun 1976, DOI: 10.1007/BF00975629.

[39] B. Shneiderman, "Measuring computer program quality and comprehension". *Intl. J. Man-Machine Studies* **9(4)**, pp. 465–478, Jul 1977, DOI: 10.1016/S0020-7373(77)80014-X.

[40] H. A. Simon and W. G. Chase, "Skill in chess". *American Scientist* **61(4)**, pp. 394–403, Jul-Aug 1973.

[41] A. A. Takang, P. A. Grubb, and R. D. Macredie, "The effects of comments and identifier names on program comprehensibility: An experimental investigation". *J. Prog. Lang.* **4**, pp. 143–167, Sep 1996.

[42] K. Tashima, H. Aman, S. Amasaki, T. Yokogawa, and M. Kawahara, "Fault-prone Java method analysis focusing on pair of local variables with confusing names". In 44th *Euromicro Conf. Softw. Eng. & Advanced Apps.*, pp. 154–158, Aug 2018, DOI: 10.1109/SEAA.2018.00033.

[43] X. Xia, L. Bao, D. Lo, Z. Xing, A. E. Hassan, and S. Li, "Measuring program comprehension: A large-scale field study with professionals". *IEEE Trans. Softw. Eng.* **44(10)**, pp. 951–976, Oct 2018, DOI: 10.1109/TSE.2017.2734091.