Empirical Software Engineering Journal manuscript No.
(will be inserted by the editor)

Seeing Confusion Through a New Lens: On the Impact of Atoms of Confusion on Novices' Code Comprehension: Additional Analysis

José Aldo Silva da Costa · Rohit Gheyi · Fernando Castor · Pablo Roberto Fernandes de Oliveira · Márcio Ribeiro · Baldoino Fonseca

Abstract Code comprehension is crucial for software maintenance and evolution, but it can be hindered by tiny code snippets that can confuse the developers, called atoms of confusion. Previous studies investigated how atoms impact code comprehension through the perspectives of time, accuracy, and opinions of developers. However, we need more studies evaluating other perspectives and the combination of these perspectives on a common ground through experiments. In our study, we evaluate how the eye tracking method can be used to gain new insights when we compare programs obfuscated by the atoms with functionally equivalent clarified versions. We conduct a controlled experiment with 32 novices in Python and measure their time, number of attempts, and visual effort with eye tracking through fixation duration, fixations count, and regressions count. We also conduct interviews and investigate the subjects' difficulties with the programs. In our results, the clarified version of the code with *Operator Precedence* reduced the time spent in the region that contains the atom to the extent of 38.6%, and the number of answer attempts by 28%. Most subjects found the obfuscated version more difficult to solve than the clarified one, and they reported the order of precedence to be difficult to validate. By analyzing their visual effort, in the obfuscated version, we observed an increase of 47.3% in the horizontal regressions count in the atom region, making its reading more difficult. The additional atoms evaluated revealed other interesting nuances. Based on our findings, we encourage researchers to consider eye tracking combined with other perspectives to evaluate atoms of

J.A. Silva da Costa and R. Gheyi and P. R. Fernandes de Oliveira
Federal University of Campina Grande
E-mail: josealdo@copin.ufcg.edu.br, rohit@dsc.ufcg.edu.br, pablo@copin.ufcg.edu.br

F. Castor
Utrecht University
E-mail: f.j.castordelimafilho@uu.nl

M. Ribeiro and B. Fonseca
Federal University of Alagoas
E-mail: marcio@ic.ufal.br, baldoino@ic.ufal.br

confusion and educators to favor patterns that do not impact the understanding and visual effort of undergraduates.

Keywords atoms of confusion · code comprehension · eye tracking

1 Summary of Data

We found that the clarified version of the code containing the atom *Operator Precedence* reduced the time spent in the AOI and in the entire code to the extent of 38.6% and 20.1%, respectively. The metrics for the visual effort were also impacted to a considerable extent. The fixation duration in the AOI and fixations count reduced as well. The most impacted metric was the regressions count in the AOI, with a reduction by 50%. These results can be associated with less visual effort. In addition, the number of attempts was reduced by 28.3%. On the other hand, the clarified version of the code containing *Multiple Variable Assignment* increased the time and the regressions count. We did not observe consistent reduction for the other atoms, however, they revealed other interesting nuances.

1.1 Multiple Variable Assignment

Answering our RQs, in the clarified version of code with *Multiple Variable Assignment*, there is an increase in the time in the AOI (RQ_1) and in the number of vertical regressions between the two lines (RQ_5). While in the obfuscated version, subjectively, the sources of confusion concentrate on the first line with the atom, while in the clarified version, they concentrate on the number of variables.

1.2 True or False Evaluation

Answering our RQs, in the clarified version of code with *True or False Evaluation*, there is a reduction in the number of horizontal regressions in the AOI (RQ_5). The additional metrics in RQ_1–RQ_4 did not present a substantial impact. In the obfuscated version, subjectively, the sources of confusion are more concentrated in `not`, while in the clarified version, the sources are more diverse.

1.3 Conditional Expression

Answering our RQs, in the clarified version of code with *Conditional Expression*, there is an increase in the number of vertical regressions in the AOI concerning the RQ_5. While there were substantial reductions in the fixation duration and fixations count (RQ_4–RQ_5), time and submissions

		In the AOI					In the Code				
Atoms	**Metrics**	O	C	PD %	PV	ES	O	C	PD %	PV	ES
Multiple Variable Assignment	Time (sec)	8.3	10.9	↑30.1	**0.03**	0.31	41.1	44.3	↑7.6	0.85	n/a
	Attempts	n/a	n/a	n/a	n/a	n/a	1.19	1.07	↓10.0	0.16	n/a
	Fix. Duration (sec)	4.2	5.1	↑22.9	0.15	n/a	19.4	18.2	↓6.1	0.93	n/a
	Fix. Count	13.0	18.0	↑38.4	0.17	n/a	59.5	59.5	0.0	0.86	n/a
	Reg. Count	2.5	4.0	↑60.0	**0.04**	0.29	26.0	25.0	↓3.8	0.67	n/a
	Horiz. Reg. Count	2.5	2.0	↓20.0	0.98	n/a	11.5	10.0	↓13.0	0.95	n/a
	Vert. Reg. Count	0.0	2.0	↑Inf	0.000	–	14.0	14.5	↑3.5	0.42	n/a
True or False Evaluation	Time (sec)	20.3	16.3	↓19.6	0.24	n/a	61.2	55.9	↓8.5	0.93	n/a
	Attempts	n/a	n/a	n/a	n/a	n/a	1.25	1.37	↑9.6	0.36	n/a
	Fix. Duration (sec)	10.1	10.6	↑4.7	0.47	n/a	35.4	27.0	↓23.7	0.81	n/a
	Fix. Count	30.5	28.0	↓8.2	0.22	n/a	92.0	79.0	↓14.1	0.77	n/a
	Reg. Count	9.5	5.0	↓47.3	**0.03**	-0.34	41.5	35.0	↓15.6	0.74	n/a
	Horiz. Reg. Count	9.5	5.0	↓47.3	0.03	–	23.5	15.0	↓36.1	0.58	n/a
	Vert. Reg. Count	0.0	0.0	n/a	n/a	n/a	17.0	16.0	↓5.8	0.87	n/a
Conditional Expression	Time (sec)	30.7	32.1	↑4.3	0.84	n/a	71.6	62.7	↓12.3	0.24	n/a
	Attempts	n/a	n/a	n/a	n/a	n/a	1.22	1.14	↓8.8	0.46	n/a
	Fix. Duration (sec)	18.9	14.0	↓25.5	0.44	n/a	34.3	26.9	↓21.5	0.18	n/a
	Fix. Count	59.0	41.0	↓30.5	0.33	n/a	107.0	78.0	↓27.1	0.21	n/a
	Reg. Count	15.0	14.0	↓6.6	0.71	n/a	43.0	35.0	↓18.6	0.38	n/a
	Horiz. Reg. Count	14.0	8.0	↓42.8	0.01	–	26.0	14.0	↓46.8	0.02	–
	Vert. Reg. Count	0.0	6.0	↑Inf	0.000	–	17.0	21.0	↑23.5	0.45	n/a
Operator Precedence	Time (sec)	20.2	12.4	↓38.6	**0.009**	-0.37	43.5	34.7	↓20.1	**0.04**	0.29
	Attempts	n/a	n/a	n/a	n/a	n/a	1.62	1.16	↓28.3	**2x10^{-4}**	-0.46
	Fix. Duration (sec)	11.0	7.3	↓34.1	**0.02**	-0.33	19.9	17.1	↓14.1	0.08	n/a
	Fix. Count	32.5	22.0	↓32.3	**0.02**	-0.32	57.5	49.0	↓14.7	0.07	n/a
	Reg. Count	10.0	5.0	↓50.0	**0.02**	-0.33	25.0	19.0	↓24.0	0.06	n/a
	Horiz. Reg. Count	10.0	5.0	↓50.0	0.02	–	14.0	10.0	↓28.5	0.04	–
	Vert. Reg. Count	0.0	0.0	n/a	n/a	n/a	11.5	9.0	↓21.7	0.09	n/a
Implicit Predicate	Time (sec)	26.4	17.3	↓34.4	0.29	n/a	71.2	47.6	↓33.1	0.10	n/a
	Attempts	n/a	n/a	n/a	n/a	n/a	1.42	1.18	↓16.9	0.16	n/a
	Fix. Duration (sec)	16.1	11.0	↓31.5	0.42	n/a	35.1	25.9	↓26.2	0.33	n/a
	Fix. Count	43.0	29.0	↓32.5	0.45	n/a	86.0	64.5	↓25.0	0.28	n/a
	Reg. Count	10.0	8.0	↓20.0	0.88	n/a	40.0	28.5	↓28.7	0.26	n/a
	Horiz. Reg. Count	10.0	8.0	↓20.0	0.88	n/a	22.0	14.5	↓34.0	0.44	n/a
	Vert. Reg. Count	0.0	0.0	n/a	n/a	n/a	21.0	12.0	↓42.8	0.13	n/a
Augmented Operator	Time (sec)	7.2	5.9	↓17.6	0.18	n/a	45.7	36.5	↓20.0	0.30	n/a
	Attempts	n/a	n/a	n/a	n/a	n/a	1.13	1.20	↑6.1	0.69	n/a
	Fix. Duration (sec)	4.4	2.8	↓35.4	0.09	n/a	20.4	16.4	↓19.3	0.29	n/a
	Fix. Count	11.5	8.0	↓30.4	0.22	n/a	58.5	47.5	↓18.8	0.41	n/a
	Reg. Count	2.0	1.0	↓50.0	0.36	n/a	24.0	17.0	↓29.1	0.31	n/a
	Horiz. Reg. Count	2.0	1.0	↓50.0	0.36	n/a	9.0	6.0	↓33.3	0.08	n/a
	Vert. Reg. Count	0.0	0.0	n/a	n/a	n/a	13.0	11.0	↓15.3	0.71	n/a

Table 1: Results for all metrics for all atoms. O = obfuscated code; C = clarified code; PD = percentage difference; PV = p-value; ES = effect size (Cliff's delta). Columns O and C are based on the median as a measure of central tendency, except for attempts, which is based on the mean.

were not so affected (RQ_1–RQ_2). In the obfuscated version, subjectively, confusion sources are more concentrated in the conditional expression. In the clarified version, they concentrate in variables and the condition.

1.4 Operator Precedence

Answering our RQs, in the clarified version of code with *Operator Precedence*, there were reductions in the time in the AOI (RQ_1), number of attempts (RQ_2), duration of fixations (RQ_3), fixations count (RQ_4), and horizontal regressions count (RQ_5). In the obfuscated version, subjec-

tively, sources of confusion are more concentrated in `if` condition, while in the clarified version, still in `if` condition, however, less often.

1.5 Implicit Predicate

Answering our RQs, in the clarified version, we observe reductions in the time in the AOI (RQ_1), number of attempts (RQ_2), fixation duration (RQ_3), fixations count (RQ_4), and regressions count (RQ_5). The highest impact was observed for the time in the AOI. In the obfuscated version, subjectively, the sources of confusion are more concentrated in the condition of the `if` statement involving the modulo operator, while in the clarified version, still in the `if` statement but with more diverse sources.

1.6 Augmented Operator

Answering our RQs, in the clarified version, we observed reductions in the time spent in the AOI (RQ_1) as well as in the fixation duration (RQ_3), fixations count (RQ_4), and regressions count (RQ_5). The number of attempts slightly increased (R_2). In the obfuscated version, subjectively, the sources of confusion are more concentrated in the lack of knowledge of the `*=` symbol, while in the clarified version, the sources were concentrated in the values.

2 Additional Analyses

All atoms. We present the data distribution in Appendix 1. We analyze all atoms combined in Table 2. The idea of combining all atoms follows the Latin Square design methodology. We assign two subjects to each square. Each subject solves six programs obfuscated by the atoms from one set of programs, and six clarified programs from another set of programs. We can combine the atoms in two ways. We can perform combinations of the individual atoms, but not pairing the subjects in the squares, and we can combine the atoms by pairing the subjects. In the former, if we have a reduction in time in code for all six individual atoms, we will have a reduction across all atoms as well. However, in the latter one, which we used, we can better control for differences among the subjects and we may have slight increases in the combination. Combined, the differences were not so evident for the number of horizontal and vertical regressions in the AOI clarified. The results revealed that, across all atoms, the average number of horizontal regressions reduced by 31.6% while the average number of vertical reduced from zero to 6.5. For *Multiple Variable Assignment*, *True or False Evaluation*, and *Conditional Expression*, we have shorter lines of code in the clarified version, however, for the *Operator Precedence*, *Implicit*

Predicate, and *Augmented Operator*, even with more elements, we observed reductions.

		In the AOI					In the Code				
Atoms	**Metrics**	O	C	PD %	PV	ES	O	C	PD %	PV	ES
All atoms	Time (sec)	83.5	81.4	↓2.5	0.67	n/a	268.4	267.5	↓0.3	0.70	n/a
	Attempts	n/a	n/a	n/a	n/a	n/a	6.53	6.25	↓4.2	0.29	n/a
	Fix. Duration (sec)	50.9	45.8	↓10.0	0.32	n/a	124.5	128.1	↑2.8	0.77	n/a
	Fix. Count	143.5	127.5	↓11.4	0.46	n/a	367.0	392.5	↑6.9	0.86	n/a
	Reg. Count	39.5	34.5	↓12.6	0.35	n/a	159.0	164.0	↑3.1	0.89	n/a
	Horiz. Reg. Count	39.5	27.0	↓31.6	0.03	–	84.5	69.5	↑17.7	0.34	n/a
	Vert. Reg. Count	0.0	6.5	↑Inf	$\mathbf{3x10^{-11}}$	–	73.0	87.0	↑19.1	0.37	n/a

Table 2: Results for all metrics for all atoms. O = obfuscated code; C = clarified code; PD = percentage difference; PV = p-value; ES = effect size (Cliff's delta). Columns O and C are based on the median as a measure of central tendency except for attempts, which is base on the mean.

In Table 3, we present each of the null-hypotheses with its respective confirmation or rejection. The two most sensitive metrics were the regressions count, which showed significant differences in three out of six atoms, followed by time, which showed differences in two atoms. The most noticeable effect size was observed for the code containing the *Operator Precedence* with respect to the clarified version in RQ_2 (Cliff's delta of -46). The other effects observed for clarified versions of the code containing the *Multiple Variable Assignment*, *True or False Evaluation*, and *Operator Precedence* were also noticeable, but to a smaller degree.

Atoms	**Time**	**Attempts**	**Fix. Duration**	**Fix. Count**	**Reg. Count**
Multiple Var. Assignment	Reject				Reject
True or False Evaluation					Reject
Operator Precedence	Reject	Reject	Reject	Reject	Reject

Table 3: Summary of the rejection of the null hypothesis in isolated atoms in the AOIs. Null-Hypothesis consists of no difference between control and treatment groups with respect to the mentioned metric.

As shown in Table 3, we found consistent reductions in the time, number of attempts, and visual metrics only for the clarified version of the code with *Operator Precedence* and partially increases the *Multiple Variable Assignment*. However, for the other four atoms, we did not observe such differences in our study. According to Gopstein et al. [1], measuring time and answer correctness, in the C language, the atoms *Assignment as Value*, *Conditional Operator*, *Operator Precedence*, and *Implicit Predicate* cause confusion. However, we added

the new perspective of visual metrics in our study which complements time and answer correctness. The new atom added in our study, the *True or False Evaluation*, did not affect time and number of attempts consistently such as *Operator Precedence*. However, we found evidence for the reductions in the number of visual regressions within the line of code containing the atom. We need more studies with other demographic groups and more atoms.

Possible explanations for the lack of consistent differences across our metrics can be due to that fact that we varied the programming language or the demographic group. For instance, the syntax for the *Conditional Operator* in the C language includes the symbol "?:", which led to 31% more errors than `if` statements. In the syntax of Python, the *Conditional Operator* includes the test of a condition in a single line with the if-else. In Python syntax, we did not observe an impact in the number of attempts, which can be easier to understand compared to the syntax of C. These differences can shed light on the differences in the results. In addition, since we focused on novices, we had to resort to simple programs, which can also explain the lack of differences in the number of attempts. While the differences in language and syntax may have an impact on the results, there are also methodological differences compared to Gopstein et al. [1] such as the sample size and statistical method. For instance, they had 72 subjects evaluating 84 paired programs using the McNemar's test. They had 16x more data points and used a more powerful statistical test to find significance.

Interaction of the atoms. We tested the interaction of the atoms by defining them as independent variables with the six levels. We found statistically significant differences for the time spent in the AOI for the obfuscated (p-value $< 8\text{x}10^{-11}$) and clarified (p-value $< 1\text{x}10^{-12}$) versions, according to atom types. The post-hoc test revealed that the time the subjects spent in the AOI obfuscated by *Multiple Variable Assignment* was significantly lower than in the AOIs of the other atoms, except for *Augmented Operator*. Similarly, the time in the *Multiple Variable Assignment* clarified was significantly lower than in the AOIs of the other atoms except for *Operator Precedence* and *Implicit Predicate*. The time in the AOI of the code obfuscated by *Augmented Operator* was significantly lower than in the other atoms, except for *Multiple Variable Assignment*, while in the clarified, the time was lower than any other AOI evaluated. In addition, for clarified, the time in the AOI with *Condition Expression* was higher than any other atom. These results indicate more difficulties with *Condition Expression* and less difficulties with *Multiple Variable Assignment* and *Augmented Operator*. We performed the same analysis on duration of fixation, fixations count, and regressions count, and we observed similar results.

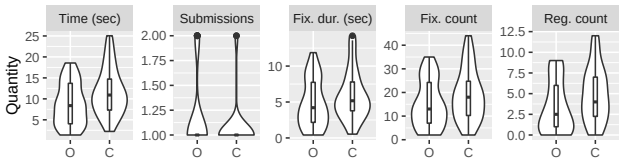
With respect to the number of attempts, we found statistically significant differences for the obfuscated (p-value < 0.003) but not for the clarified version, according to atom types. The *Operator Precedence* with the obfuscated version presented significant lower number of attempts compared to every other atom. These results indicate more difficulties with *Operator Precedence*.

Sets of programs. As an additional analysis, we compared between the two distinct programs with same atoms instantiated trying to find possible differences. The most relevant differences were observed for the *Operator Precedence* in the clarified versions. For instance, the novices spent 41% less time in the AOI and 33% fewer regressions examining `if True or (True and False)` compared to `if (False and True) or False`. The explanation might be that, since the priority is in the beginning of the expression, it can help the novices to make sense of it faster and do not need to go back many times. For the *Implicit Predicate*, we observed that the novices spent 39% less and regressed 42% fewer times in `if (elem % 5 != 0)` compared to `if (elem % 4 != 0)`. A possible explanation might be that it is easier to compare with odd numbers since the rest is different from zero.

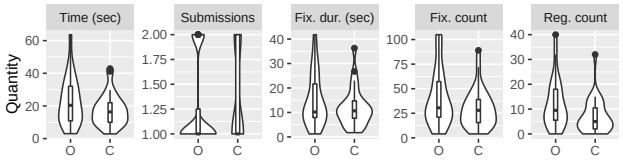
For the *Augmented Operator*, the novices spent 55% less time in the AOI, fixated 58% less, and made 200% fewer regressions examining `total += 1` compared to `valor *= 10`. A possible explanation might be that "+=" operator is seen more frequently than "*=" for novices or that multiplication might be more complex and requires more memory than sum operation. For the clarified version, the novices spent 40% less time in the AOI, fixated 51% less, and made 50% fewer regressions examining `valor = valor * 10` compared to `total = total + 1`.

References

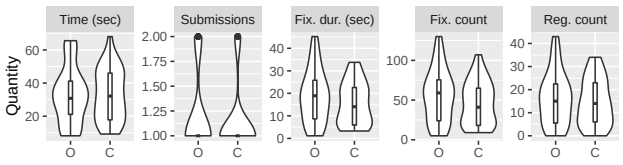
1. Dan Gopstein, Jake Iannacone, Yu Yan, Lois DeLong, Yanyan Zhuang, Martin K.-C. Yeh, and Justin Cappos. Understanding Misunderstandings in Source Code. In *Proceedings of the Joint Meeting on Foundations of Software Engineering*, ESEC/FSE'17, pages 129–139, 2017.



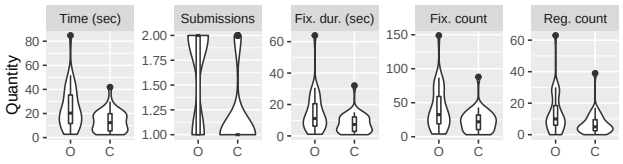
(a) Distribution of data for *Multiple Variable Assignment*



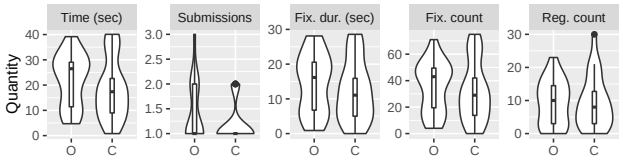
(b) Distribution of data for *True or False Evaluation*



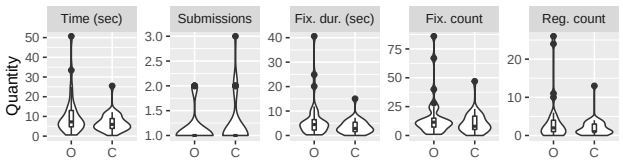
(c) Distribution of data for *Conditional Expression*



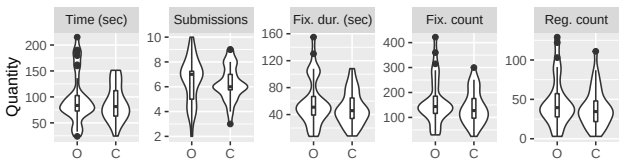
(d) Distribution of data for *Operator Precedence*



(e) Distribution of data for *Implicit Predicate*



(f) Distribution of data for *Augmented Operator*



(g) Distribution of data for all atoms combined

Fig. 1: Distribution of the data in the AOI. Obfuscated (O) and Clarified (C) versions of all atoms for ST_1–ST_2 together.