Encourage, Engage, and Enhance a Community for Atoms of Confusion

Justin Cappos[ny] Dan Gopstein[ms] Renata Vaderna[ny] Yanyan Zhuang[co]
[ny] New York University [co] University of Colorado, Colorado Springs [ms] Microsoft

***Abstract*—The recent work by our team has empirically validated the existence of small patterns in C code that can interfere with program comprehension. These small code patterns, named atoms of confusion (or atoms for short), are correlated with many software bugs. This finding has been replicated by numerous research teams. For example, atoms have been validated in different programming languages including Java and Python, and through various methods such as using an EEG device.**

To engage a growing community, we have set out to put our work into practice by bridging the gap between academia and industry. We have been working to deploy atom parsers in the Linux kernel, and developing data sharing and visualization tools with the Linux Foundation. We plan to build a broader community through academia-industry collaboration, encouraging, engaging, and enhancing community connections to grow into a large open-source ecosystem. We invite you to join us!

***Index Terms*—Atoms of Confusion, Code Comprehension, Community Effort, Open Source**

I. Introduction

Software bugs represent a significant challenge worldwide, with a staggering cost of approximately $2.08 trillion in the US alone in 2020, as reported by the Consortium for Information & Software Quality (CISQ) [5]. Addressing these bugs efficiently before they cause damage is crucial in both industry and academia. Code review serves as a cornerstone of quality assurance in both open source and proprietary projects. Research highlights that inadequate code reviews are linked to diminished software quality, the emergence of anti-patterns, and increased post-release defects. Ensuring that software is understandable to humans is crucial yet challenging, often approached ad hoc.

Our research has delved into the causes of code confusion for human developers. We have empirically validated small C code patterns, termed *atoms of confusion*, which programmers often misinterpret [8]. Analyzing 14 large C projects revealed hundreds of thousands of these atoms, highly correlated with both comments and bug fixes, underscoring their profound impact on software reliability [9]. This body of work has inspired a breadth of subsequent studies across various programming languages, employing methodologies like eye-tracking and EEG to probe deeper into how programmers interact with these confusing elements [3], [11], [12], [16], [17]. Over a dozen research teams, e.g., from the Netherlands and Brazil, have replicated experiments from our prior work and published their work about Java [11], [13], JavaScript [14], Swift [1], Python [4], and the Android platform [15]. A Senior Scientist from the University of Klagenfurt, Austria collaborated with us and published a recent discovery that used the datasets from our work [8], [18] and designed a high-quality C programming ability assessment tool [7].

As this research started to gain momentum, we initiated efforts to enhance community collaboration. By fostering discussions, sharing datasets, and initiating research into unified standardization of what constitutes an atom of confusion, we aim to streamline the process for researchers to conduct future atoms-related experiments. Additionally, we are committed to expanding awareness and fostering interest in this novel approach of examining the causes of code confusion and bugs, not just within academia, but also across the industry.

II. Enhancing Community Engagement

With the community's enthusiasm, however, comes the different definitions and execution in these works that led to divergent uses of our original concept. In this section, we outline our effort to unify the research community and apply our work in collaboration with the industry.

A. Community Meetings

In our previous experience, organizing community meetings has proven to be effective in gathering individuals who are passionate about a research project. Since January 2024, we have initiated bimonthly meetings to cultivate a strong sense of community among researchers. Our goal is to establish a supportive platform where participants can share their work and engage in constructive feedback exchanges within the community. Another challenge is the varied interpretations of what constitutes an atom. We organized meetings specifically focused on this issue, presenting multiple code snippets that have been seen by some as atoms of confusion, while others may not find them confusing. These discussions have sparked significant interest and motivated further exploration into understanding these divisive elements.

B. A Framework for Parsing Atoms of Confusion

While our previous parsers [9] have shown correlations between atoms, bugs, and code comments, they have not succeeded in providing clear definitions. Our goal is to develop a precise, easily extensible, and reproducible methodology for identifying specific code patterns as atoms. To this end, we are actively developing a framework that includes guidelines to determine what is and is not an atom, based on our findings that increased occurrences of comments and bug fixes are often correlated with the presence of atoms.

First, we count the occurrences of all Abstract Syntax Tree (AST) subtrees within a large open-source project, as well as within bug-fix commits and commented code snippets. Next, we look for patterns of code that are correlated (or not correlated) with high frequencies of bug fixes and comments. The former patterns are identified as atoms. To test the significance of these correlations, we will apply a formal statistical method, such as Hotelling's T-Square method, to distinguish atoms from non-atoms.

This approach aims to provide a precise and objective method for identifying atoms, potentially uncovering additional confusing patterns that were previously unrecognized. Researchers will also be able to apply different metrics beyond comments and bug-fix frequencies by adding new dimensions or substituting the measures we have chosen. This framework is designed to support hypothesis testing among community members, assist future projects in exploring atoms in various programming languages, and enhance software reliability.

C. Collaboration with the Industry

We aim for our research to go beyond academic borders and have a greater impact in the industry. Our previous research helped identify and fix several Linux kernel bugs caused by macros, often occurring when developers incorrectly parenthesized one or more arguments. This success demonstrates the potential practical impact of our work and highlights the importance of ongoing collaboration with the industry. Below, we provide two more examples of such collaboration.

Coccinelle is a tool that is widely used to detect issues and refactor the Linux kernel code. It allows a user to specify desired patterns and refactoring in C code. Unlike traditional patches, Coccinelle's semantic patches can modify hundreds of files across thousands of code sites by abstracting away specific details and variations [2].

Given its extensive use, and our direct collaboration with the maintainers of Coccinelle, we are implementing atom parsers using Coccinelle. While our primary focus is on deployment within the Linux kernel, Coccinelle can be used to process any C code. Consequently, the semantic patches we have developed to detect atoms can also be applied to other C codebases. We have implemented patches for the detection of 14 out of 15 identified atoms, excluding only one that is beyond Coccinelle's capabilities. All these semantic patches are available in our GitHub repository [6].

Additionally, to further encourage collaboration between academia and industry, we dedicated one of our community meetings to Coccinelle. We invited a key contributor to Coccinelle to present the tool to our community, with a specific focus on detecting atoms. This initiative brings us a step closer to bridging the gap between academic research and practical industry applications.

OpenSSF and the Linux Foundation. To further solidify our efforts and expand the impact of our research, we are planning to establish a collaboration with the Open Source Security Foundation (OpenSSF) and the Linux Foundation. These organizations provide a collaborative and structured environment that encourages innovation and standardization in open-source security. An important aspect of this collaboration is the sharing of datasets, which allows teams to replicate and build upon our and each other's work. To facilitate the sharing of various large datasets, we are working with the Linux Foundation to host them in a neutral, industry-facing manner. We have successfully collaborated with these organizations on other projects in the past, and with this established reputation, we aim to further promote the adoption of our methodologies and tools. Aligning with OpenSSF and the Linux Foundation will help ensure that our research on atoms of confusion can benefit a wider range of software projects across the industry.

REFERENCES

[1] F. Castor. Identifying confusing code in swift programs. In *Proceedings of the VI CBSoft Workshop on Visualization, Evolution, and Maintenance. ACM*, 2018.

[2] Coccinelle. https://coccinelle.gitlabpages.inria.fr/website/. Accessed: August 5, 2024.

[3] J. da Costa, R. Gheyi, F. Castor, P. de Oliveira, M. Ribeiro, and B. Fonseca. Seeing confusion through a new lens: on the impact of atoms of confusion on novices' code comprehension. *Empirical Software Engineering*, 28(4), 2023.

[4] J. A. S. da Costa, R. Gheyi, F. Castor, P. R. F. de Oliveira, M. Ribeiro, and B. Fonseca. Seeing confusion through a new lens: on the impact of atoms of confusion on novices' code comprehension. *Empirical Software Engineering*, 28(4):81, 2023.

[5] C. for Information & Software Quality. The cost of poorsoftware quality in the us: A 2020 report. https://www.it-cisq.org/cisq-files/pdf/CPSQ-2020-report.pdf, 2020. Accessed: August 6, 2024.

[6] Coccinelle atom finder. https://github.com/AtomsofConfusion/atom-finder-coccinelle. Accessed: August 5, 2024.

[7] C. Glasauer, M. K. Yeh, L. A. DeLong, Y. Yan, and Y. Zhuang. "C"ing the light – assessing code comprehension in novice programmers using C code patterns. *Computer Science Education*, pages 1–25, 2024.

[8] D. Gopstein, J. Iannacone, Y. Yan, L. DeLong, Y. Zhuang, M. K.-C. Yeh, and J. Cappos. Understanding misunderstandings in source code. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, pages 129–139, 2017.

[9] D. Gopstein, H. H. Zhou, P. Frankl, and J. Cappos. Prevalence of confusing code in software projects: Atoms of confusion in the wild. In *Proceedings of the 15th International Conference on Mining Software Repositories*, pages 281–291, 2018.

[10] H. Hotelling. The generalization of student's ratio. *Annals of Mathematical Statistics*, 2(3):360–378, 1931.

[11] C. Langhout and M. Aniche. Atoms of confusion in java. In *2021 IEEE/ACM 29th International Conference on Program Comprehension (ICPC)*, pages 25–35, 2021.

[12] I. M. V. Manor. Átomos de confusão em swift. Master's thesis, Federal University of Pernambuco, Center for Informatics, Undergraduate in Computer Science, 2018. Bachelor's Thesis.

[13] W. Mendes, O. Pinheiro, E. Santos, L. Rocha, and W. Viana. Dazed and confused: Studying the prevalence of atoms of confusion in long-lived java libraries. In *2022 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, Limassol, Cyprus, 2022.

[14] C. Oliveira and A. R. F. Torres. On the impact of atoms of confusion in javascript code. 2019.

[15] D. Tabosa, O. Pinheiro, L. Rocha, and W. Viana. A dataset of atoms of confusion in the android open source project. In *2024 IEEE/ACM 21st International Conference on Mining Software Repositories (MSR)*, pages 520–524, 2024.

[16] A. Torres, C. Oliveira, M. Okimoto, D. Marcílio, P. Queiroga, F. Castor, R. Bonifácio, E. D. Canedo, M. Ribeiro, and E. Monteiro. An investigation of confusing code patterns in javascript. *Journal of Systems and Software*, 203, 2023.

[17] M. K.-C. Yeh, D. Gopstein, Y. Yan, and Y. Zhuang. Detecting and comparing brain activity in short program comprehension using eeg. In *2017 IEEE Frontiers in Education Conference (FIE)*, pages 1–5, 2017.

[18] Y. Zhuang, Y. Yan, L. A. DeLong, and M. K. Yeh. Do developer perceptions have borders? comparing C code responses across continents. *Software Quality Journal*, 2023.