Rooting Out Atoms Of Confusion – A Call To Action

Justin Cappos
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

Dan Gopstein
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

Renata Vaderna
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

Yanyan Zhuang
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

***Abstract*—A few years ago, Gopstein et al. discovered that many bugs in software are caused by small, confusing patterns in code called Atoms of Confusion. This finding has been replicated to different domains and validated in a variety of ways, including brain wave scans with EEGs. ...**

We are a growing community of academics which are bridging the industry / academic gap to put these ideas in practice. We are currently adding data sharing and visualization tools, which are working with the Linux Foundation to host in a neutral, industry facing manner. We have been working to deploy Atoms checking in the Linux Kernel and are also building a broader community through deep connections into large open source ecosystems. We invite you to join us!

***Index Terms*—component, formatting, style, styling, insert.**

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) [3]. 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 in an ad hoc manner.

Our research has delved into the causes of code confusion for human readers. We have identified and empirically validated small C code patterns, termed *atoms of confusion*, which programmers often misinterpret. 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 [5].

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 [2], [6]–[11].

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

In our previous experience, organizing community meetings has proven to be an effective method for gathering individuals who are passionate about specific fields or projects. Since the beginning of this year, we have initiated bimonthly meetings to cultivate a strong sense of community among researchers interested in this field. During these gatherings, we encourage members to introduce themselves and to share both past and recent research. We ensure that lists of all published papers are readily accessible, and distribute notes and slides from presentations. Our primary aim is to establish a supportive platform where every participant can showcase their work and engage in constructive feedback exchanges within the community.

Furthermore, we've initiated discussions on challenges particularly relevant to those studying code confusion, such as conducting user studies. We hope that members of the community who are new to this area can learn from and seek guidance from experienced researchers who have conducted several user studies.

Another challenge we've tackled is the varied interpretations of what constitutes an atom of confusion. We've 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.

A. A Framework for Classifying Atoms of Confusion

We have identified a core issue arising from the absence of a formal definition of what constitutes an atom of confusion. While our previous classifiers [5] have shown correlations between atoms of confusion, bugs, and confusing code, they have not succeeded in providing clear definitions. Our goal is to develop a precise, easily extendable, and reproducible methodology for classifying specific code patterns as atoms. To this end, we are actively developing a framework that includes mathematical guidelines to determine what is and isn't an atom. This framework is not only designed to support hypothesis testing among community members but also to pave the way for future applications to explore atoms in various programming languages. This effort builds upon our findings that increased frequencies of comments and bug fixes often indicate the presence of atoms, reinforcing their significant impact on software reliability.

The main idea behind this ongoing research is to count occurrences of all AST subtrees within large open source projects, as well as in bugfix commits and commented code snippets, looking for patterns with high frequencies of bugfixes and comments. The next step involves applying a formal statistical method (link to method here) to clearly distinguish atom candidates from non-atoms. This approach aims to provide a more precise and objective answer to questions about questionable patterns, potentially uncovering additional confusing patterns that were previously unrecognized. Additionally, researchers should be able to apply different metrics, adding new dimensions or substituting the measures of bugfix and comment frequencies.

B. Collaboration with the Industry

We aim for our research on atoms of confusion to surpass academic borders and have a tangible impact in the industry. Our previous research has led to the detection and fix of a bug in the Linux kernel caused by one of the patterns we consider to be an atom of confusion. This success demonstrates the potential for the practical impact of our work and highlights the importance of ongoing collaboration with industry.

C. Coccinelle

The tool widely used to detect issues and refactor certain parts of the Linux kernel code is Coccinelle. Coccinelle is a program matching and transformation engine that provides the Semantic Patch Language (SmPL) for specifying desired matches and transformations in C code. It was initially designed to document and automate collateral evolutions in device driver code. Unlike traditional patches, which are limited to specific instances, Coccinelle's semantic patches can modify hundreds of files across thousands of code sites by abstracting away specific details and variations [1].

Given its extensive use, and in direct collaboration with the maintainers of Coccinelle, we are implementing the detection of atoms 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 [4].

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.

D. 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 fosters innovation and standardization in open-source security. 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] Coccinelle. https://coccinelle.gitlabpages.inria.fr/website/. Accessed: August 5, 2024.

[2] 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):81, 2023. Epub 2023 May 18. PMID: 37220598; PMCID: PMC10193347.

[3] 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 5, 2024.

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

[5] 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. ACM, 2018.

[6] C. Langhout and M. Aniche. Atoms of confusion in java. In *2021 IEEE/ACM 29th International Conference on Program Comprehension (ICPC)*, pages 25–35, Delft, The Netherlands, 2021. Delft University of Technology, Software Engineering Research Group.

[7] 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.

[8] 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)*, pages 106–116, Limassol, Cyprus, 2022.

[9] W. Mendes, W. Viana, and L. Rocha. Bohr - uma ferramenta para a identificação de Átomos de confusão em códigos java. In *Anais do IX Workshop de Visualização, Evolução e Manutenção de Software*, pages 41–45, Joinville, Brasil, 2021. SBC, Porto Alegre, Brasil.

[10] 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.

[11] 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. IEEE, 2017.