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) [**?**]. 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 [**?**].

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 [**?**], [**?**], [**?**], [**?**], [**?**], [**?**], [**?**].

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 [**?**] 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.

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 [**?**].

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 [**?**].

C. Abbreviations and Acronyms

Define abbreviations and acronyms the first time they are used in the text, even after they have been defined in the abstract. Abbreviations such as IEEE, SI, MKS, CGS, ac, dc, and rms do not have to be defined. Do not use abbreviations in the title or heads unless they are un

D. Figures and Tables

a) Positioning Figures and Tables: Place figures and tables at the top and bottom of columns. Avoid placing them in the middle of columns. Large figures and tables may span across both columns. Figure captions should be below the figures; table heads should appear above the tables. Insert figures and tables after they are cited in the text. Use the abbreviation "Fig. 1", even at the beginning of a sentence.

TABLE I
TABLE TYPE STYLES

Table Head	**Table Column Head**		
	Table column subhead	***Subhead***	***Subhead***
copy	More table copy[a]		

[a]Sample of a Table footnote.

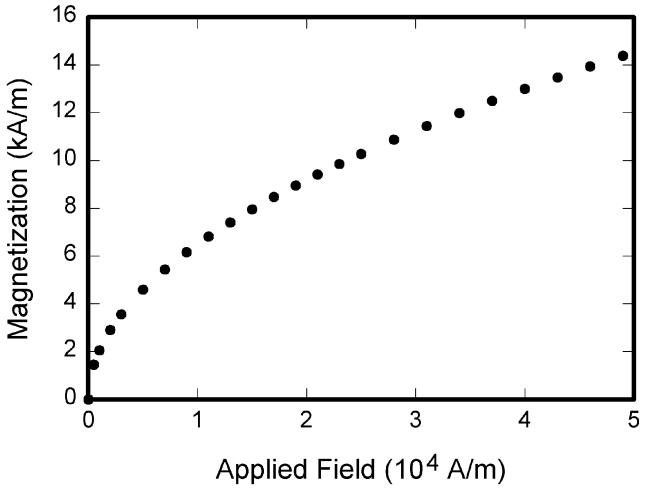

Fig. 1. Example of a figure caption.

Figure Labels: Use 8 point Times New Roman for Figure labels. Use words rather than symbols or abbreviations when writing Figure axis labels to avoid confusing the reader. As an example, write the quantity "Magnetization", or "Magnetization, M", not just "M". If including units in the label, present them within parentheses. Do not label axes only with units. In the example, write "Magnetization (A/m)" or "Magnetization

{A[m(1)]}", not just "A/m". Do not label axes with a ratio of quantities and units. For example, write "Temperature (K)", not "Temperature/K".

ACKNOWLEDGMENT

The preferred spelling of the word "acknowledgment" in America is without an "e" after the "g". Avoid the stilted expression "one of us (R. B. G.) thanks ...". Instead, try "R. B. G. thanks...". Put sponsor acknowledgments in the unnumbered footnote on the first page.

REFERENCES

Please number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in [3]—do not use "Ref. [3]" or "reference [3]" except at the beginning of a sentence: "Reference [3] was the first ..."

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the abstract or reference list. Use letters for table footnotes.

Unless there are six authors or more give all authors' names; do not use "et al.". Papers that have not been published, even if they have been submitted for publication, should be cited as "unpublished" [4]. Papers that have been accepted for publication should be cited as "in press" [5]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [6].

REFERENCES

[1] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.

[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[3] I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[4] K. Elissa, "Title of paper if known," unpublished.

[5] R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[7] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

[8] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," 2013, arXiv:1312.6114. [Online]. Available: https://arxiv.org/abs/1312.6114

[9] S. Liu, "Wi-Fi Energy Detection Testbed (12MTC)," 2023, gitHub repository. [Online]. Available: https://github.com/liustone99/Wi-Fi-Energy-Detection-Testbed-12MTC

[10] "Treatment episode data set: discharges (TEDS-D): concatenated, 2006 to 2009." U.S. Department of Health and Human Services, Substance Abuse and Mental Health Services Administration, Office of Applied Studies, August, 2013, DOI:10.3886/ICPSR30122.v2

[11] K. Eves and J. Valasek, "Adaptive control for singularly perturbed systems examples," Code Ocean, Aug. 2023. [Online]. Available: https://codeocean.com/capsule/4989235/tree