



# Testing, Debugging, and Log Analysis With Modern AI Tools

Miroslaw Staron<sup>ID</sup>, Silvia Abrahão<sup>ID</sup>, Gregory Gay<sup>ID</sup>, and Alexander Serebrenik<sup>ID</sup>

**TESTING AND LOG** analysis can be effort intensive, just to say the least. Finding bugs in complex components (such as neural networks for image analysis) requires new methods for finding the right data and new processes for doing so. At the same time, generative artificial intelligence (AI)-based tools such as Stable Diffusion, ChatGPT, and autoencoders can be very useful for generating new data and new test scenarios, either freeing up valuable testing resources or increasing the amount of testing performed with the same resources. A similar observation holds for log analysis: this activity is time consuming and sometimes even difficult to perform with traditional methods but not with generative AI.

Therefore, in this month's edition of the "Practitioner's Digest," we summarize recent research on testing, debugging, and log analysis from two conferences: the 38th IEEE/ACM International Conference on Automated Software Engineering (ASE 2023) and the 16th IEEE International Conference on Software Testing, Verification and Validation. We hope the papers will inspire you to use generative AI



©SHUTTERSTOCK.COM/GIRAFCHIK

in a safe, sustainable, and responsible way. Feedback or suggestions are welcome. In addition, if you try or adopt any of the practices included in the column, please send us and the authors of the paper(s) a note about your experiences.

## Generative AI and Fuzzifying Image Data

Effective training and use of deep learning techniques depends heavily on the data used to train the networks. The general principle is that

higher data volume, more data diversity, and more truthful data lead to better network performance. However, the reality is that the data are scarce and often not as diverse as we would have liked. This is where data augmentation techniques come into play, often using simple shape or color transformations, sometimes adding a bit of noise to the data. In the paper "Semantic Data Augmentation for Deep Learning Testing Using Generative AI," Sondess Missaoui, Simos Gerasimou, and Nicholas

Matragkas present a new technique, GenFuzzer, that uses generative AI to augment images. The augmented images are still within the domain space of the dataset but differ significantly from the existing images. The authors found that GenFuzzer with Stable Diffusion generative AI is able to effectively generate high-fidelity synthetic test inputs. The presented approach increased the testing effec-

the use of ChatGPT in finding a failing test input and its expected output for failure-inducing test cases. They experimented using ChatGPT with both buggy and correct Python programs in QuixBugs, a common benchmark for studying the use of large language models for software engineering tasks.

The experiment shows that ChatGPT has a relatively low success rate (28.8%) in finding failure-inducing

-prompting.github.io/ to facilitate future research. The paper was presented at the Research Papers Track of ASE 2023. Access it at <https://arxiv.org/pdf/2304.11686.pdf>.

### Practitioners' Expectations on the Readability of Log Messages

Although logging is an inherent part of software development, it is not exactly clear what developers expect from log messages. This is the challenge faced by Zhenhao Li, An Ran Chen, Xing Hu, Xin Xia, Tse-Hsun (Peter) Chen, and Weiyi Shang, in their paper "Are They All Good? Studying Practitioners' Expectations on the Readability of Log Messages." In their study the authors first conducted a series of interviews with software practitioners to ask them about aspects of log messages that they consider important or examples of confusing or unhelpful log messages.

From these interviews, the authors identified structure, information, and wording as the three main facets of practitioners' expectations. Among the three facets, information is considered the most important. The authors then manually analyzed the structure, information, and wording of a sample of log messages from nine large open source projects, noting that the readability of nearly 40% of the log messages is inadequate with respect to one or more facets. The adequacy of the log messages was highest when the messages were between six and 10 words; for shorter or longer messages, it was lower. Next, the authors used the sample messages to survey practitioners asking them to evaluate practices that can help improve the structure, information, and wording of the messages. These practices included, for example, having clear boundaries and

Finding bugs in complex components (such as neural networks for image analysis) requires new methods for finding the right data and new processes for doing so.

tiveness of the studied networks by up to 26%. In conclusion, whatever we think of generative AI, it seems effective in improving mundane tasks such as fuzzing images. Thanks to community-driven repositories such as Hugging Face, these models are becoming increasingly accessible. The paper was presented at the New Ideas and Emerging Results Track of ASE 2023. Access it at <https://tinyurl.com/bdsauwez>.

### ChatGPT and Differential Prompting

Finding failure-inducing test cases is a primary objective in software testing. It is, however, challenging in practice. In the paper "Nuances Are the Key: Unlocking ChatGPT to Find Failure-Inducing Tests With Differential Prompting," Tsz-On Li, Wenxi Zong, Yibo Wang, Haoye Tian, Ying Wang, Shing-Chi Cheung, and Jeff Kramer present the first study to investigate

test cases for buggy programs. Nevertheless, the authors make an observation that the program intent inferred by ChatGPT is insensitive to buggy nuances in program code. It points out that ChatGPT can mostly infer the correct intention of a buggy program for common coding tasks on which ChatGPT has been well trained. Leveraging the observation, an automated technique, called differential prompting, is devised. It automatically deduces the failing test inputs and their expected outputs for failure-inducing test cases by generating alternative program implementations that fulfill the inferred intention using ChatGPT and applying the deduced test cases to the original program.

The evaluation result shows that differential prompting significantly outperforms the state-of-the-art baselines in finding failure-inducing test cases. The authors provide a replication package at <https://differential>

distinctions among items and using parameterized logs to present variables, writing log messages that are self-explanatory and independent of other log messages, following the convention of written language and using impartial and neutral wording.

Finally, the authors examined ways in which the adequacy of log messages can be automatically assessed and showed that both deep learning and machine learning approaches achieve promising results, suggesting that practitioners could use techniques such as bidirectional long-short term memory, random forest, and decision tree to support the automatic assessment of log message quality. The paper was presented at the Research Papers Track of ASE 2023. Access it at [https://ginolzh.github.io/papers/ASE2023\\_Log\\_Message\\_Readability.pdf](https://ginolzh.github.io/papers/ASE2023_Log_Message_Readability.pdf).

### ChatGPT and Log Parsing

Log files constitute an important part of debugging and maintaining large-scale software systems. They provide rich and pervasive information but can also be overwhelming as they are inherently unstructured because developers usually record logs using free text for convenience and flexibility. In the paper “Log Parsing: How Far Can ChatGPT Go?,” Van-Hoang Le and Hongyu Zhang study how we can use ChatGPT to analyze log files. In particular, they studied whether it is possible to use so-called few-shot learning to teach ChatGPT what to look for in the log file.

The presented study used the GPT-3.5-turbo model through the OpenAI application programming interface (API). The results show that with zero-shot prompts, i.e., with the vanilla ChatGPT prompt, the model is on par with the best existing baselines. With four examples (four-shot

learning), ChatGPT outperforms all other models, with perfect results for one of the systems studied and with an average general accuracy of 0.76. This means that when provided with four examples of good answers, the model can correctly replicate the same pattern in new log files. In terms of editing distance, the four-shot model

space to be explored due to the large number of operations, possible execution orders, dependencies between parameters, and constraints associated with the values of the input parameters.

In the paper “Adaptive REST API Testing With Reinforcement Learning,” Myeongsoo Kim, Saurabh

With four examples (four-shot learning), ChatGPT outperforms all other models, with perfect results for one of the systems studied and with an average general accuracy of 0.76.

performed best, with an average editing distance of 3.2. This is impressive for such a generic model as ChatGPT, which was not designed for this task.

The only limitation, however, is the fact that ChatGPT cannot recognize domain-specific elements of log files, which means that for these cases, the output would need additional processing. In conclusion, ChatGPT can help debug and analyze large-scale software systems, thus easing the tedious task of software maintenance. The paper was presented at the New Ideas and Emerging Results Track of ASE 2023. Access it at <https://arxiv.org/pdf/2306.01590.pdf>.

### Adaptive REST API Testing

APIs act as the bridge between different software applications. When it comes to testing web applications, REST API testing is a key strategy to evaluate the efficiency of RESTful APIs. However, it can be challenging because of the large search

Sinha, and Alessandro Orso present a black-box testing approach (adaptive REST API testing with reinforcement learning [ARAT-RL]) that leverages reinforcement learning to prioritize operations and parameters for exploration, dynamically construct key-value pairs from response and request data, analyze these pairs to inform dependent operations and parameters, and use a sampling-based strategy to efficiently process dynamic API feedback and adapt its exploration based on the gathered information.

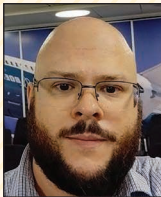
To evaluate ARAT-RL, the authors conducted a set of empirical studies using 10 RESTful services and compared its performance with that of three state-of-the-art REST API testing tools: RESTler, EvoMaster, and Mostest. These tools were compared in terms of effectiveness, efficiency, and fault detection capability. The results show that the proposed approach and tool outperformed the three tools considered in terms of the branch, line,



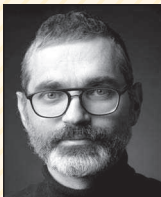
**MIROSLAW STARON** is a professor in the Interaction Design and Software Engineering Division, Computer Science and Engineering Department, Chalmers University of Technology and the University of Gothenburg, SE-412 96 Gothenburg, Sweden. Contact him at <https://www.staron.nu> or [mirosław.staron@cse.gu.se](mailto:mirosław.staron@cse.gu.se).



**SILVIA ABRAHÃO** is a full professor of software engineering in the Department of Computer Systems and Computation, Universitat Politècnica de València, 46022 Valencia, Spain. Contact her at <https://sabrahao.wixsite.com/dsic-upv> or [sabrahao@dsic.upv.es](mailto:sabrahao@dsic.upv.es).




**GREGORY GAY** is an associate professor in the Interaction Design and Software Engineering Division, Computer Science and Engineering Department, Chalmers University of Technology and the University of Gothenburg, SE-412 96 Gothenburg, Sweden. Contact him at <http://greggay.com> or [greg@greggay.com](mailto:greg@greggay.com).



**ALEXANDER SEREBRENİK** is a professor at Eindhoven University of Technology, 5600 MB Eindhoven, The Netherlands. Contact him at [a.serebrenik@tue.nl](mailto:a.serebrenik@tue.nl).

suite, and *program repair*, in which tools attempt to produce patches that fix faults. The two areas are often studied in isolation. However, in the paper “The Inversive Relationship Between Bugs and Patches: An Empirical Study,” Jinhao Kim, Jongchan Park, and Shin Yoo make the observation that both introducing faults and patching them are fundamentally code changes. Thus, they hypothesize that these two actions do not differ syntactically.

Specifically, the authors empirically assess this hypothesis by comparing patches and faulty commits using clustering and pattern analysis. They found that up to 70% of patches and faults can be clustered based on the change patterns they contain and that 44% of code changes can be mapped into patterns. In other words, automated practices that introduce and fix faults are intrinsically related, and advances in one or the other field can inform the other.

This finding is especially relevant for machine-learning-based tools, which infer patterns (for fixing or mutating code) from real-world software commits. Instead of learning only from one or the other source, tools could infer patterns from both (e.g., a repair tool could be trained on both fault-fixing commits and additional patches created by reversing fault-introducing commits). The authors demonstrate this by inverting the use of mutation and automatic program repair tools and find that, for example, the TBar repair tool can produce more fault couplings than state-of-the-art mutation tools. The study was presented at the 2023 IEEE International Conference on Software Testing, Verification and Validation Workshops. Access it at <https://arxiv.org/pdf/2303.00303.pdf>. 

and method coverage achieved, requests generated, and faults detected. The authors also conducted an ablation study to assess the individual effects of prioritization, dynamic feedback analysis, and sampling on the overall effectiveness of ARAT-RL. The results indicate that each of the components contributes to the overall effectiveness of the tool and that the prioritization mechanism plays an important role in improving the performance of the tool in terms of code coverage achieved and faults detected. Overall, these results suggest that by using ARAT-RL, developers and testers can improve their

REST API testing strategies and deliver high-quality APIs. ARAT-RL experiment infrastructure and data are available at <https://github.com/codingsoo/ARAT-RL>. The paper was presented at the Research Papers Track of ASE 2023. Access it at <https://arxiv.org/pdf/2309.04583.pdf>.

### Introducing Bugs and Patching: Same, Same but Different

Extensive research has been conducted on *mutation testing*, a practice in which faults are introduced into code to assess the robustness of a test