

Human-Centered Evaluation of REST API Fuzzing Tools: Bridging Academia and Industry

Fanny Febriani Susilo

School of Economics, Innovation and Technology

Kristiania University of Applied Sciences

Oslo, Norway

fanny.susilo@kristiania.no

Abstract—As software systems grow in complexity—especially in cloud-based microservice architectures, automated testing has become crucial for reliability and security. While academic REST API fuzzing tools (e.g., EvoMaster, Schemathesis) show strong fault detection, their adoption in industry is limited due to insufficient human-centered evaluation focusing on usability, learnability, and integration. This PhD project will address the gap through a mixed-methods, human-centered evaluation approach. Phase 1 will review current empirical methods in automated testing. Phase 2 will run controlled lab studies with students to evaluate tool usability and cognitive load. Phase 3 will bring the evaluation to real-world industrial settings. By triangulating technical performance with cognitive, behavioral, and perceptual data, this project aims to foster more usable and effective testing tools. It will contribute empirical methods and design recommendations to align academic advancements with real-world development needs.

Index Terms—fuzzing, automated testing, human factors, empirical software engineering, human computer interaction.

I. INTRODUCTION

Modern software systems, underpinning nearly every societal aspects, increasingly leverage cloud platforms (e.g., AWS, Azure) and microservice architectures for scalability and rapid deployment [1]. Composed of modular, independently deployable services [2], these systems typically communicate via REST APIs [3], offering simpler integration than older protocols like SOAP [4] and accelerating adoption across enterprises.

However, as digital infrastructure becomes more complex, the potential impact of software failures rises, particularly in terms of security vulnerabilities [5]. The Knight Capital software error in 2012 caused a \$440 million loss in just 45 minutes [6], while the Log4shell vulnerability in 2021 exposed millions of systems worldwide to remote code execution attacks through a flaw in a logging library [7]. Such incidents underscore how software bugs can lead to financial and reputational damage, making software testing a critical yet costly [8] quality assurance component that ensures expected system behavior and early fault detection.

Manual testing remains valuable, but it often involves repetitive and time-consuming tasks. Its limitation in scalability and

efficiency for modern software ecosystems have driven the adoption of automated testing [9]. Among these techniques, fuzzing excels at finding security vulnerabilities by injecting invalid, unexpected, or random inputs into software and observing its behavior [10].

Numerous academic tools exist for REST API fuzzing [11], including EvoMaster [12], Schemathesis [13], and RESTler [14], with some like EvoMaster seeing industrial adoption in large-scale microservice environments such as Meituan [15] and Volkswagen AG [16]. However, most evaluations prioritize technical metrics (e.g. fault detection, code coverage) while neglecting crucial human factors like developer usage, learning curves, and integration challenges [11], [17], [18]. This oversight limits our understanding of adoption barriers, as the practical success of automated test generation tools depends not just on technical performance but also on usability, learnability, and seamless integration into real-world development processes.

This PhD project aims to bridge the gap between academic innovation and industry adoption by advancing human-centered, empirical evaluation methods for REST API fuzzing tools in cloud-based microservice environments.

II. BACKGROUND AND RELATED WORK

A. Technical Landscape of REST API Fuzzing Tools

Fuzz testing has advanced from random input generation [19] to sophisticated techniques like coverage-guided [20] and grammar-based input generation [21], now widely applied to REST APIs for enhanced test input generation and fault detection. Modern REST API fuzzers typically use OpenAPI schemas to generate valid HTTP requests and employ diverse test generation strategies. For instance, EvoMaster [22] uses search-based heuristics for both black-box and white-box testing, while RESTler [14], RestTestGen [23], and Schemathesis [13] operate as black-box fuzzers, employing diverse generation approaches (e.g., producer-consumer dependencies, dependency graphs, property-based testing). Newer tools like EmRest [24] leverage error message analysis, while LlamaRestTest [25] uses fine-tuned large language models for realistic input generation and dependency inference. Despite these advances, many tools remain research-oriented with limited widespread integration into daily development practice. While some tools like RESTler and Schemathesis show

This project is a part of EAST, which is a larger project funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (EAST project, grant agreement No. 864972).

growing adoption (e.g., 2000+ GitHub stars), broader usability and integration studies are still scarce, especially across diverse developer populations.

B. Barriers to Adoption in Industry

Despite strong detection capabilities, REST API fuzzers face industrial adoption barriers beyond their technical performance. Recurring challenges across fuzzing tools include compatibility, functionality, performance, reliability, reproducibility, usability, and implementation effort [17]. Real-world observations of EvoMaster’s use reveal developer difficulties such as steep learning curves, limited framework compatibility, and unclear value propositions [15]. These findings suggest that even well-engineered tools may fail to gain traction if they do not align with the practical realities of developer workflows.

C. Human Factors in Developer Tools

Human factors are crucial for tool adoption yet remain underrepresented in software testing research. Studies on REST API fuzzing involving human participants are notably absent [11]. Only a small fraction of web testing studies include human-subject evaluations [18], often using students, raising generalizability concerns. As a result, little is known about how developers truly learn, evaluate, and integrate fuzzing tools into their daily workflows. This contrasts with broader software engineering trends, where human-centered methods have gained prominence [26]–[28], reflecting growing attention to developer cognition and tool interaction. This vital shift has not yet impacted automated testing research, leaving usability and workflow integration largely unexamined.

D. Gaps in Current Evaluations

Current evaluations of REST API fuzzers primarily focus on narrow metrics like code/schema coverage and fault detection (e.g., HTTP 5xx errors) [11]. For example, EvoMaster [22] demonstrated strong code coverage and fault detection, while RESTler [14] and RestTestGen [23] showed improvements in robustness and schema coverage, respectively highlighting how test strategies affect different quality dimensions. Such metrics, however, overlook critical adoption factors like usability, onboarding experience, and developer satisfaction. Furthermore, a substantial portion of REST API test generation studies introduce novel techniques, often validated only in confined lab settings with limited real-world application [11]. Methodological flaws are also prevalent, including vague experimental procedures, inconsistent demographic reporting, and inconsistent use of statistical tests [26], [29]. Evaluations frequently rely on either fully aggregated or task-specific data, obscuring insights and introducing validity risks [29]. Tool feature analyses are often subjective and misaligned with developer needs [30]. Although robust frameworks for empirical rigor exist [28], [31], [32], their infrequent application undermines the reliability and generalizability of findings.

In summary, despite their strong technical potential, REST API fuzzing tools are underutilized due to usability issues, integration barriers, and a lack of human-centered evaluation.

The persistent disconnect between academic innovation and industrial adoption necessitates rigorous, human-centered research that considers not just what tools detect, but also how developers learn, use, and adopt these tools in practice.

III. PROPOSED APPROACH

This PhD project uses a mixed-methods approach to evaluate the usability, integration, and adoption of automated testing tools, specifically REST API fuzzers. It aims to address a critical gap in human-centered evaluation at the intersection of software testing and human-computer interaction (HCI).

The project is guided by the following research questions:

- RQ1: How do developers with different experience levels (e.g., students vs. professionals) perceive and interact with automated testing tools?
- RQ2: What usability and integration barriers limit the effective use of automated testing tools for novice and experienced developers?
- RQ3: How do lab-based findings on usability and effectiveness differ from those observed in real-world, industrial contexts?
- RQ4: What methodological strategies are needed to evaluate automated test generation tools in realistic, industrial settings?
- RQ5: How do interface design, onboarding materials, and documentation affect the learnability and adoption of fuzzing tools?

Throughout all empirical phases involving human participants (Phases 2 and 3), ethical and privacy compliance will be maintained through informed participant consent, data anonymization, adherence to GDPR compliance [33], and obtaining all necessary ethics board approvals.

The research is structured in three phases, each building on the findings of the previous.

A. Phase 1: Systematic Literature Review (SLR)

Following Kitchenham et al.’s [34] guidelines, the SLR will examine how automated test generation tools have been empirically evaluated, focusing on human-centered concerns like usability, tool adoption, and developer experience.

It will investigate:

- Evaluation strategies involving human participants
- Human factors, tool characteristics, and technical performance metrics
- Methodological rigor, reporting practices, and study design
- Practical challenges and implications for adoption

We target empirical studies involving human participants or human-centric concerns in automated testing. Database searches will be conducted in IEEE Xplore, ACM DL, ScienceDirect, Wiley, Web of Science, MIT Libraries, and SpringerLink. Additionally, backward and forward snowballing [35] will ensure search comprehensiveness. Metadata (e.g., evaluation methods, participant demographics, tool characteristics) will be extracted and analyzed thematically and

quantitatively. The results of this study will lay the foundation for Phases 2 and 3 by identifying research gaps and informing study design.

B. Phase 2: Controlled Lab Experiments with Students

This phase will investigate early-stage interaction, usability, and cognitive effort. A controlled lab experiment will use automated testing tools as the independent variable. EvoMaster is the primary tool due to its open-source availability, active development, documented industrial use, and high code coverage performance in recent evaluation [36], supporting both black-box and white-box testing via search-based heuristics. Schemathesis will also be considered for its popularity (2000+ GitHub stars), user-friendly design, comprehensive documentation, and competitive black-box fuzzer performance in empirical evaluations [36]. Task instructions, environment setup, and time constraints will be standardized to ensure internal validity.

Participants will be upper-level undergraduate and graduate Computer Science students, with a target sample size of 25–35 to balance statistical power with qualitative depth. Recruitment will occur through course invitations, mailing lists, and lab announcements.

Quantitative dependent variables could include:

- Effectiveness: code coverage, faults detected
- Efficiency: task completion time, valid test cases generated
- Usability: System Usability Scale (SUS) scores [37], task success/error rates
- Cognitive load: NASA Task Load Index (NASA-TLX) ratings [38]

Qualitative data sources could include:

- Observational data: screen recordings, usage logs (e.g., tool interactions and command sequences), field notes, and optional think-aloud protocols
- Physiological data: eye tracking (visual attention), facial expression analysis (emotional state), GSR or EEG (stress and cognitive load)
- Interviews: semi-structured, exploring perceptions, pain points, comprehension, and improvement suggestions
- Questionnaires: Likert-scale and open-ended items on background (e.g., experience level, prior tool use, skill self-assessment) and subjective experience

Note: The final selection and implementation of all data collection methods are subject to feasibility assessments and full ethical approval.

A mixed-methods design will balance statistical generalizability with contextual insight, using quantitative metrics (e.g., coverage, SUS) for baselines and qualitative data (e.g., interviews, think-aloud protocols) to explain and contextualize observed behaviors. Physiological data will interpret cognitive and affective responses in relation to tool use and task difficulty. This triangulated approach enhances validity and practical relevance for researchers and practitioners.

Data analysis will involve descriptive statistics to summarize core measures and inferential statistics to compare user

groups and task conditions. Selective aggregation [29] will be applied to maintain statistical power while retaining task-specific insights. Qualitative data will undergo thematic analysis, guided by frameworks like Cognitive Load Theory [39], the Technology Acceptance Model [40], or ISO 9241-210 on human-centered design [41].

Our study design, evaluation, and reporting will adhere to methodological frameworks from Sadler [31], Kitchenham [30], Vos [32], and Ko [28] to ensure rigor, reproducibility, and alignment with real-world workflows. Tools like NVivo, SPSS, iMotions will be used for data analysis.

This multi-theoretical, multi-modal approach will enable us to identify usability barriers with nuance and informs context-aware tool improvements. Tasks will simulate realistic developer workflows, including tool setup, test generation, environment configuration, and result interpretation. The results of this study will RQ1, RQ2, and RQ5.

C. Phase 3: Field Studies in Industrial Settings

This phase evaluates real-world usability, integration, and adoption of fuzzing tools by professional developers. Industry partners will integrate selected tools into ongoing projects for a defined period. Participants will primarily be QA or API development roles. Recruitment will occur through existing research partnerships and industrial contacts, targeting 10–20 developers across 2–4 partner organizations to ensure generalizability while enabling comparative analysis.

Data collection will mirror Phase 2 but be adapted for the industrial context and conducted in situ, with tasks reflecting participants' daily workflows, collaboratively defined with stakeholders to ensure validity and alignment with actual testing practices. This phase will address RQ2, RQ3, and RQ4, enabling comparison between controlled lab findings and real-world application outcomes.

IV. EXPECTED RESULTS AND CONTRIBUTIONS

This project will contribute to empirical software engineering and human-computer interaction through:

- A human-centered evaluation framework for assessing automated test generation tools, incorporating behavioral, perceptual, and physiological measures.
- Empirical insights into how usability, cognitive load, and contextual factors (e.g., developer experience, lab vs. field setting) influence the adoption of automated testing tools.
- Design recommendations for tool onboarding, documentation, and interface improvements, for different developer experience levels and real-world workflows.
- Methodological contributions through a replicable, mixed-methods approach that combines controlled experiments with field studies.

V. CURRENT STATUS

The project began in May 2025 and is currently in Phase 1 (SLR). An initial 959 papers are being filtered based on predefined criteria, with snowballing for comprehensiveness.

This PhD project is a vital component of the ERC-funded EAST project. As EAST approaches its final phase, our human-centered evaluation of REST API fuzzing will provide crucial insights, bridging the gap between academic advancements and real-world applicability.

Upcoming activities include preparing the annual progress report, engaging in workshops with company partners, and developing science communication pieces. We aim to publish the SLR findings and two empirical papers in top SE/HCI conferences and journals (e.g., TOSEM, IST, ICSE, ESEM, CHI, IUI, and ASE). Thesis defense is planned for Spring 2028.

ACKNOWLEDGMENT

I would like to thank my supervisors Assoc. Prof. Jefferson Moller and Prof. Andrea Arcuri for all their continuous support, help and invaluable advice with this PhD.

REFERENCES

- [1] Villamizar, M., Garcés, O., Castro, H., Verano, M., Salamanca, L., Casallas, R., & Gil, S. (2015, September). Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud. In 2015 10th computing colombian conference (10ccc) (pp. 583-590). IEEE.
- [2] Newman, S. (2015). Building microservices. oreilly media. Inc...
- [3] Fielding, R. T. (2000). Architectural styles and the design of network-based software architectures. University of California, Irvine.
- [4] Curbera, F., Duftler, M., Khalaf, R., Nagy, W., Mukhi, N., & Weerawarana, S. (2002). Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI. *IEEE Internet computing*, 6(2), 86-93.
- [5] McGraw, G. (2012). Software security: Building security in. *Datenschutz und Datensicherheit-DuD*, 36(9), 662-665.
- [6] Saltapidas, C., & Maghsood, R. (2018). Financial risk the fall of knight capital group. University of Gothenburg, 1-8.
- [7] Oxford Analytica. (2021). Apache software flaw could result in major breaches. *Emerald Expert Briefings*, (oxan-es).
- [8] Beizer, B. (2003). Software testing techniques. dreamtech Press.
- [9] Hynninen, T., Kasurinen, J., Knutas, A., & Taipale, O. (2018, May). Software testing: Survey of the industry practices. In 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO) (pp. 1449-1454). IEEE.
- [10] Oehlert, P. (2005). Violating assumptions with fuzzing. *IEEE Security & Privacy*, 3(2), 58-62.
- [11] Golmohammadi, A., Zhang, M., & Arcuri, A. (2023). Testing restful apis: A survey. *ACM Transactions on Software Engineering and Methodology*, 33(1), 1-41.
- [12] Arcuri, A., Zhang, M., Seran, S., Galeotti, J. P., Golmohammadi, A., Duman, O., ... & Ghianni, H. (2025). Tool report: EvoMaster—black and white box search-based fuzzing for REST, GraphQL and RPC APIs. *Automated Software Engineering*, 32(1), 4.
- [13] Hatfield-Dodds, Z., & Dygalo, D. (2022, May). Deriving semantics-aware fuzzers from web api schemas. In Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Companion Proceedings (pp. 345-346).
- [14] Atlidakis, V., Godefroid, P., & Polishchuk, M. (2019, May). Restler: Stateful rest api fuzzing. In 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE) (pp. 748-758). IEEE.
- [15] Zhang, M., Arcuri, A., Li, Y., Liu, Y., Xue, K., Wang, Z., ... & Huang, W. (2025). Fuzzing microservices: A series of user studies in industry on industrial systems with evomaster. *Science of Computer Programming*, 103322.
- [16] Poth, A., Rjollli, O., & Arcuri, A. (2025). Technology adoption performance evaluation applied to testing industrial REST APIs. *Automated Software Engineering*, 32(1), 5.
- [17] Nourry, O., Kashiwa, Y., Lin, B., Bavota, G., Lanza, M., & Kamei, Y. (2023). The human side of fuzzing: Challenges faced by developers during fuzzing activities. *ACM transactions on software engineering and methodology*, 33(1), 1-26.
- [18] Kertusha, I., Assress, G., Duman, O., & Arcuri, A. (2025). A Survey on Web Testing: On the Rise of AI and Applications in Industry. *arXiv preprint arXiv:2503.05378*.
- [19] Miller, B. P., Fredriksen, L., & So, B. (1990). An empirical study of the reliability of UNIX utilities. *Communications of the ACM*, 33(12), 32-44.
- [20] Zalewski, M. (2015). American fuzzy lop (AFL) fuzzer (2015). URL: <http://lcamtuf.coredump.cx/afl/>(visited on 05/12/2019), 33.
- [21] Godefroid, P., Kiezun, A., & Levin, M. Y. (2008, June). Grammar-based whitebox fuzzing. In Proceedings of the 29th ACM SIGPLAN conference on programming language design and implementation (pp. 206-215).
- [22] Zhang, M., Belhadi, A., & Arcuri, A. (2022, April). JavaScript instrumentation for search-based software testing: A study with RESTful APIs. In 2022 IEEE Conference on Software Testing, Verification and Validation (ICST) (pp. 105-115). IEEE.
- [23] Viglianisi, E., Dallago, M., & Ceccato, M. (2020, October). Resttestgen: automated black-box testing of restful apis. In 2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST) (pp. 142-152). IEEE.
- [24] Xu, L., Wu, H., Pan, Z., Xu, T., Wang, S., Niu, X., & Nie, C. (2025). Effective REST APIs Testing with Error Message Analysis. *Proceedings of the ACM on Software Engineering*, 2(ISSTA), 1978-2000.
- [25] Kim, M., Sinha, S., & Orso, A. (2025). Llamaresttest: Effective rest api testing with small language models. *Proceedings of the ACM on Software Engineering*, 2(FSE), 465-488.
- [26] Sjøberg, D. I., Hannay, J. E., Hansen, O., Kampenes, V. B., Karahasanovic, A., Liborg, N. K., & Rekdal, A. C. (2005). A survey of controlled experiments in software engineering. *IEEE transactions on software engineering*, 31(9), 733-753.
- [27] Buse, R. P., Sadowski, C., & Weimer, W. (2011, October). Benefits and barriers of user evaluation in software engineering research. In Proceedings of the 2011 ACM international conference on Object oriented programming systems languages and applications (pp. 643-656).
- [28] Ko, A. J., LaToza, T. D., & Burnett, M. M. (2015). A practical guide to controlled experiments of software engineering tools with human participants. *Empirical Software Engineering*, 20, 110-141.
- [29] Siegmund, J., Peitek, N., Apel, S., & Siegmund, N. (2020). Mastering variation in human studies: The role of aggregation. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 30(1), 1-40.
- [30] Kitchenham, B. A., & Jones, L. (1997). Evaluating software engineering methods and tool part 5: the influence of human factors. *ACM SIGSOFT Software Engineering Notes*, 22(1), 13-15.
- [31] Sadler, C., & Kitchenham, B. A. (1996). Evaluating software engineering methods and tool—part 4: the influence of human factors. *ACM SIGSOFT Software Engineering Notes*, 21(5), 11-13.
- [32] Vos, T. E., Marín, B., Escalona, M. J., & Marchetto, A. (2012, August). A methodological framework for evaluating software testing techniques and tools. In 2012 12th international conference on quality software (pp. 230-239). IEEE.
- [33] Regulation, P. (2016). Regulation (EU) 2016/679 of the European Parliament and of the Council. *Regulation (eu)*, 679, 2016.
- [34] Kitchenham, B., & Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering.
- [35] Wohlin, C. (2014, May). Guidelines for snowballing in systematic literature studies and a replication in software engineering. In Proceedings of the 18th international conference on evaluation and assessment in software engineering (pp. 1-10).
- [36] Zhang, M., & Arcuri, A. (2022). Open problems in fuzzing restful APIs: a comparison of tools (2023).
- [37] Brooke, J. (1996). SUS-A quick and dirty usability scale. *Usability evaluation in industry*, 189(194), 4-7.
- [38] Hart, S. G., & Staveland, L. E. (1988). Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. In *Advances in psychology* (Vol. 52, pp. 139-183). North-Holland.
- [39] Sweller, J. (2011). Cognitive load theory. In *Psychology of learning and motivation* (Vol. 55, pp. 37-76). Academic Press.
- [40] Davis, F. D. (1989). Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS quarterly*, 319-340.
- [41] International Organization for Standardization. (2010). *Ergonomics of Human-system Interaction: Part 210: Human-centred Design for Interactive Systems*. ISO.