Research Statement

Gregory James Gay
Department of Computer Science and Engineering, University of South Carolina
2247 Storey Innovation and Engineeing Center, 550 Assembly Street, Columbia, SC 29208
(803)-777-9479, greg@greggay.com

1. Research Overview

My research interests lie in the field of **software engineering**, with an emphasis on **automated software testing** and **search-based software engineering**—the application of optimization techniques to software development challenges. Our society is increasingly powered by software, and the growing complexity and cost of development necessitate new advances in the state of the art. In my work, I harness the information content of software project artifacts in order to improve the quality and ease the human burden of the development process. Many of my approaches and innovations are rooted in a **data-centric approach** that intersects search, optimization, data mining, and large-scale empirical investigation.

My research vision is designed be relevant, forward-thinking, and impacting—**relevant** to the challenges faced by the developers that are shaping our connected society, **forward-thinking** in order to ensure the safety and robustness of the software of the future, and **impacting** industry, society, education, and my research community.

A few of my recent contributions include:

- Insight into the efficacy and applicability of the metrics used to guide test case generation, gleaned from some of the largest studies to date on industrial software [5], [10], [11], [13] and real faults [3], [1], [2].
- Invention of new adequacy criteria designed to overcome the limitations associated with commonly-used generation targets [14], [10], [15].
- Exploration of when and how to combine criteria in order to generate multifaceted test suites effective in complex real-world situations [4].
- Enabling the effective use of deterministic test oracles—judges on the correctness of a system—in situations where the system under test behaves non-deterministically [8], [6], [7].

My work to date has resulted in:

- Twenty-five refereed publications—with four more currently under review—in some of the most competitive journals, conferences, and workshops in the software engineering field.
- The prestigious National Science Foundation CISE Research Initiation Initiative (CRII) grant for faculty members in the process of forming promising research programs.
- Election to the steering committees of the Symposium on Search-Based Software Engineering and Workshop on Search-Based Software Testing, and selection as part of the program and organization committees of workshops and conferences such as the International Conference on Software Testing.
- Fruitful research collaborations with NASA's Jet Propulsion Lab and Ames Research Center, as well as researchers at the University of Massachusetts, University of Minnesota, North Carolina State University, University of Sheffield—among others.

I advocate the cause of open science, and make great efforts to disseminate my work to the broader community. My tools, data, and even experimental infrastructure are—when possible—released under open source licenses. I strongly believe that the results of research work should be released publicly and transparently so that society may benefit from my work and so that other researchers may replicate, improve, and even refute my results.

2. Current Research Agenda

Software testing is a crucial development activity—we must be able to rely on the systems we build to produce correct results. The ever-increasing complexity of software is making it both more difficult and expensive to ensure the correctness of system behavior. My research improves the practice of software testing—optimizing result quality, efficiency, and cost—through improvements in the selection, design, and automation of the artifacts of the testing process. Much of my current research is focused on the creation of test cases, either through automated generation of testing artifacts or easing the human burden associated with test creation and interpretation.

2.1. Adequacy Criteria

As we cannot know what faults exist without verification, and as testing cannot—except in simple cases—conclusively prove the absence of faults, a suitable approximation must be used to measure the adequacy of tests. Adequacy criteria—by imposing requirements tests must fulfill to be considered adequate—provide developers with the guidance needed to test effectively. As such criteria can be efficiently measured, they are common targets for test generation. However, the need to rely on approximations leads to two questions—*can adequacy criteria produce effective tests and, if so, which should be used to generate tests?*

To better understand the applicability and efficacy of such criteria, I have conducted large-scale empirical investigation of such criteria as the targets of model-based test generation [10], [11], [13] and search-based generation [3], [1], [2]. My findings have allowed us to better understand the use, applicability, and combination of common criteria and to examine the joint relationship between the fitness function, generation algorithm, and source code in determining the efficacy of test suites.

My work has also explored the sensitivity of criteria to program structure [5] and choice of test oracle [9], [12]. To address these sensitivities, I contributed to the development of a new coverage criterion, Observable MC/DC, which imposes path constraints that test cases must fulfill [14], [15]. Recent work—under submission—extends the idea of *observability* into a form that can enhance any criteria based on Boolean logic.

My findings indicated that combinations of criteria—when used simultaneously as generation targets—can produce test suites that are more effective than those based on single-objective generation. Empirical investigation of this topic revealed that combinations should include criteria that thoroughly explore system structure as primary generation objectives—supported by secondary criteria that explore orthogonal, supporting scenarios such as exceptions [4].

2.2. Regression Test Generation

Regression testing—the practice of re-executing test cases to reveal unintended changes in the behavior of a unit—is an expensive, human-intensive process. However, such effort *may not be necessary*. Each time that code is checked in, tests could be generated using existing code, executed against the newly committed version, and discarded after use. If effective, *disposable generation* could eliminate maintenance effort. However, questions remain about the efficacy of disposable testing—particularly given the strict time constraints necessary to fit within the continuous integration process.

In work currently under submission, I have conducted studies to investigate these questions, and to determine challenges that must be overcome to perform practical disposable regression test generation. I have found that disposable test generation can be effective, the inclusion of coupled classes can improve efficacy, and disposable testing often requires less inspection effort than traditional regression testing. Still, a number of major challenges must be addressed before it can truly replace human-guided regression testing. I am actively exploring such challenges, and am developing new supportive adequacy criteria based on coupling between classes.

2.3. Software Test Oracles

The choice of test oracle - the artifact that determines whether an application under test executes correctly - can significantly impact the efficacy of the testing process. However, despite the prevalence of tools that support test input selection, little work exists for supporting oracle creation.

I have devised a method of supporting test oracle creation that automatically selects the set of variables monitored during testing [9], [12]. This approach uses mutation analysis—the seeding of synthetic faults into the source code—to rank variables in terms of potential fault-finding efficacy. Experimental results obtained by employing this method over six industrial systems indicate that we can automatically produce small, effective oracle variable sets, with fault finding improvements over current industrial best practice.

Specifying test oracles is challenging for some domains, such as real-time embedded systems, where small changes in timing or sensory input may cause large behavioral differences. Models of such systems, often built for analysis and simulation, are appealing for reuse as test oracles. These models, however, typically represent an idealized system, abstracting away certain issues such as nondeterministic timing behavior and sensor noise. Thus, even with the same inputs, the models behavior may fail to match an acceptable behavior of the SUT, leading to many false positives reported by the test oracle. I have invented an automated steering framework that can adjust the behavior of the model to better match the behavior of the SUT to reduce the rate of false positives [8], [6], [7]. This framework allows non-deterministic, but bounded, behavioral differences, while preventing future mismatches by guiding the oracle—within limits—to match the execution of the SUT. Results show that steering significantly increases SUT-oracle conformance with minimal masking of real faults and, thus, has significant potential for reducing testing and debugging costs while improving the quality of the testing process

3. Future Research Plans

Despite advances in automated test generation, the efficacy of the produced test suites has yet to match human-produced test suites [3], [10], [11], [13]. One potential limiting factor is a misunderstanding of the role that the *human* has in an automation-aided process. Much of the existing research assumes—implicitly or explicitly—that automation *replaces* human effort entirely. However, software development is a process made up of dozens of stages, each involving intense human effort and each informing the next. Rather than ignoring the human effort that precedes or succeeds an automated activity, I believe that effective automation must *recognize* its position in the greater process of development and *augment* human efforts. Effective automation helps focus developer attention, taking into account the information when it acts, and striving to leave artifacts behind that inform the next stage of the process. I hypothesize that automation must produce results that are both *human-competitive* and *human-complementing*. Such a feat is—broadly—not possible with current automated generation efforts.

My research agenda over the next several years will be centered around inventing the next generation of automated tools—tools that enable, and take advantage of, human-automation collaboration. I believe that effective automation (1) simultaneously explores *multiple test objectives*, (2) *adapts* its strategy based on information gleaned from project artifacts, and (3), is *usable*—producing artifacts that are understandable. Search-based software engineering (SBSE)—a field I am particularly interested and involved in—is well-suited to addressing each of these items, and I plan to explore the application of SBSE to these principles in a variety of projects over the next several years. To exemplify the type of research I intend to perform, I plan to explore the following aims:

Enable flexible search-based generation of multifaceted test suites: I will investigate generic notions of expressing adequacy criteria and methods of automatically translating their obligations into the format needed for search-based generation—expanding the range of criteria and easing application of both individual and groups of criteria to code segments of varying granularity.

Investigate automated means of optimizing test strategies: For test generation to be effective, a set of strategies must be selected that are appropriate for the system under test. I will explore the use of hyperheuristic search and reinforcement learning to optimize testing strategies.

Discover methods of generating human-usable test cases: Generated test cases should by understandable by human developers. I will investigate techniques inspired by sentiment analysis and text mining to extract input, targets, and expected output from textual project artifacts such as bug reports.

References

[1] H. Almulla, A. Salahirad, and G. Gay. Using search-based test generation to discover real faults in Guava. In *Proceedings of the Symposium on Search-Based Software Engineering*, SSBSE 2017. Springer Verlag, 2017.

[2] G. Gay. Challenges in using search-based test generation to identify real faults in mockito. In *Search Based Software Engineering: 8th International Symposium, SSBSE 2016, Raleigh, NC, USA, October 8-10, 2016, Proceedings*, pages 231–237, Cham, 2016. Springer International Publishing.

[3] G. Gay. The fitness function for the job: Search-based generation of test suites that detect real faults. In *Proceedings of the International Conference on Software Testing*, ICST 2017. IEEE, 2017.

[4] G. Gay. Generating effective test suites by combining coverage criteria. In *Proceedings of the Symposium on Search-Based Software Engineering*, SSBSE 2017. Springer Verlag, 2017.

[5] G. Gay, A. Rajan, M. Staats, M. Whalen, and M. P. E. Heimdahl. The effect of program and model structure on the effectiveness of mc/dc test adequacy coverage. *ACM Trans. Softw. Eng. Methodol.*, 25(3):25:1–25:34, July 2016.

[6] G. Gay, S. Rayadurgam, and M. P. Heimdahl. Improving the accuracy of oracle verdicts through automated model steering. In *Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering*, ASE '14, pages 527–538, New York, NY, USA, 2014. ACM.

[7] G. Gay, S. Rayadurgam, and M. P. Heimdahl. Steering model-based oracles to admit real program behaviors. In *Proceedings of the 36th International Conference on Software Engineering – NIER Track*, ICSE '14, New York, NY, USA, 2014. ACM.

[8] G. Gay, S. Rayadurgam, and M. P. E. Heimdahl. Automated steering of model-based test oracles to admit real program behaviors. *IEEE Transactions on Software Engineering*, 43(6):531–555, June 2017.

[9] G. Gay, M. Staats, M. Whalen, and M. Heimdahl. Automated oracle data selection support. *Software Engineering, IEEE Transactions on*, PP(99):1–1, 2015.

[10] G. Gay, M. Staats, M. Whalen, and M. Heimdahl. The risks of coverage-directed test case generation. *Software Engineering, IEEE Transactions on*, PP(99), 2015.

[11] G. Gay, M. Staats, M. W. Whalen, and M. P. E. Heimdahl. Moving the goalposts: Coverage satisfaction is not enough. In *Proceedings of the 7th International Workshop on Search-Based Software Testing*, SBST 2014, pages 19–22, New York, NY, USA, 2014. ACM.

[12] M. Staats, G. Gay, and M. Heimdahl. Automated oracle creation support, or: how I learned to stop worrying about fault propagation and love mutation testing. In *Proceedings of the 2012 Int'l Conf. on Software Engineering*, pages 870–880. IEEE Press, 2012.

[13] M. Staats, G. Gay, M. Whalen, and M. Heimdahl. On the danger of coverage directed test case generation. In J. de Lara and A. Zisman, editors, *Fundamental Approaches to Software Engineering*, volume 7212 of *Lecture Notes in Computer Science*, pages 409–424. Springer Berlin Heidelberg, 2012.

[14] M. Whalen, G. Gay, D. You, M. Heimdahl, and M. Staats. Observable modified condition/decision coverage. In *Proceedings of the 2013 Int'l Conf. on Software Engineering*. ACM, May 2013.

[15] D. You, S. Rayadurgam, M. Whalen, M. P. E. Heimdahl, and G. Gay. Efficient observability-based test generation by dynamic symbolic execution. In *2015 IEEE 26th International Symposium on Software Reliability Engineering (ISSRE)*, pages 228–238, Nov 2015.