# Automated Testing of GUI Applications: Models, Tools, and Controlling Flakiness

Atif M. Memon
University of Maryland
Department of Computer Science
College Park, MD 20742
atif@cs.umd.edu

Myra B. Cohen
University of Nebraska-Lincoln
Department of Computer Science & Eng.
Lincoln, NE 68588-0115
myra@cse.unl.edu

***Abstract*—System testing of applications with graphical user interfaces (GUIs) such as web browsers, desktop, or mobile apps, is more complex than testing from the command line. Specialized tools are needed to generate and run test cases, models are needed to quantify behavioral coverage, and changes in the environment, such as the operating system, virtual machine or system load, as well as starting states of the executions, impact the repeatability of the outcome of tests making tests appear *flaky*.**

**In this tutorial, we present an overview of the state of the art in GUI testing, consisting of both lectures and demonstrations on various platforms (desktop, web and mobile applications), using an open source testing tool, GUITAR. We show how to setup a system under test, how to extract models without source code, and how to then use those models to generate and replay test cases. We then present a lecture on the various factors that may cause flakiness in the execution of GUI-centric software, and hence impact the results of analyses and experiments based on such software. We end with a demonstration of a community resource for sharing GUI testing artifacts aimed at controlling these factors.**

**This tutorial targets both researchers who develop techniques for testing GUI software, and practitioners from industry who want to learn more about model-based GUI testing or who run and rerun GUI tests and often find their runs are flaky.**

## I. Tutorial Overview

System testing of applications on which the user interacts through a graphical user interface (GUI) such as common desktop applications, web clients or mobile apps, requires a more complex test harness than traditional command line interfaces. This *event-driven* paradigm creates several challenges. Models are needed to evaluate behavioral coverage, and specialized tools are required to generate and replay tests in an automated fashion [1]. And these tools and techniques may be platform dependent. Test cases are sequences of events, instead of simple inputs; actions that are performed on the interface in a specific order. The starting state of the application and its environment (e.g. Java version, OS and system load) can both impact the outcome of a test, and as events are executed, threads may be spawned to perform calculations, which will change the system state, and may incur delays before they are complete. All of these factors, if not carefully controlled lead to *flakiness* – the inability to repeat execution in a reliable manner, and can affect both test results and experimentation on new techniques.

This tutorial aims to de-mystify testing in event-driven systems by presenting an overview on the state of the art in GUI testing, with lectures on modeling, test generation and replay, as well as a discussion of the important factors that should be considered for controlling flakiness. We include demonstrations on the basics of automated GUI testing using the open source framework GUITAR [2], with examples from various domains such as desktop, web and mobile applications. A second demonstration on benchmarking and experimentation uses artifacts from the COMET (*COMmunity Event-based Testing*) website [3] with a focus on repeatability of test results. COMET is a community resource that the presenters of this tutorial have been developing to help alleviate some of the challenges faced when comparing and repeating experiments on event-driven testing.

## II. Audience

The expected audience for this tutorial includes students, industry practitioners, and researchers who work on software testing of systems with graphical interfaces. The tutorial includes fundamental material that is applicable to both research and industry as well as more advanced topics related to benchmarking and experimentation which are of interest to researchers or industrial practitioners who want repeatable test results. Recent research that leverages GUI testing tools to build models for predicting human performance [4] suggests that the tutorial may also be of interest to some researchers from the HCI community.

## III. Tutorial Description and Plan

The tutorial will consist of a combination of lectures intermingled with demonstrations. Within each lecture or demonstration period, sufficient time will be left open for questions or clarifications of key concepts.

The tutorial begins with an overview of GUI testing covering techniques for modeling the event space using state machine and graph models [1], [5]. We then demonstrate how the graph models can be automatically extracted and used for test case generation. To make this concept concrete, we leverage our tools (the *GUI Testing frAmewRk*, GUITAR) [2] for extracting, generating and running GUI tests. Materials will be provided for tutorial attendees so that they can repeat these demonstrations later at their own leisure.

In the second part of this tutorial, we highlight the challenges of reducing flakiness. We demonstrate how varying some factors may cause the software to change behavior in unexpected ways. We discuss some of these factors and how to control them to improve execution results reproducibility. In a second demonstration, we introduce participants to COMET, and the shared GUI benchmarks that they can use for experimentation.

**Planned Tutorial Structure**

This is a half-day tutorial with the following structure:

• **Lecture:** Introduction to GUI Testing and Models

1) Overview of GUI testing
2) State machines and event flow graphs
3) Ripping (model extraction) and replaying

• **Lecture:** Test Generation and Coverage

1) Overview of event-based coverage criteria
2) Test generation

• **Demonstration:** Using GUITAR to Automate the GUI Test Process

• **Lecture:** Extending GUITAR with New Test Generation Methods and Models of Interaction

• **Lecture:** Controlling Factors

1) Factors that impact repeatability
2) Introduction to benchmarking for experimentation

• **Demonstration:** Using COMET benchmarks

1) Downloading and configuring the benchmarks
2) Executing the benchmark subjects

## IV. Presenters

Myra Cohen and Atif Memon are the architects of this tutorial. They have expertise in testing graphical user interfaces, testing configurable software, and in experimental software engineering, all of which form the core content of the tutorial, and they have collaborated on recent papers on the state of the in GUI testing [6]–[8]. Atif Memon developed the GUITAR framework that is used for demonstration in this tutorial. Together they host COMET [3] that serves as a repository for GUI software subjects used in experimentation for increased repeatability and comparison between experiments.

**Atif Memon** is an Associate Professor in the Department of Computer Science, University of Maryland, where he founded and heads the Event Driven Software Lab (EDSL). He designed and developed the model-based GUI testing software GUITAR. He has published over 100 research articles on the topic of event driven systems, software testing, and software engineering. He is the founder of the TESTBEDS workshop. He also helped develop the workshop on Experimental Evaluation of Software and Systems in Computer Science (EVALUATE). He has given numerous talks on model-based testing, experimentation, and testing of event-driven software including 2 at the Google Test Automation Conference (GTAC) with videos available online.[1]

[1] see (*http://youtu.be/6LdsIVvxISU*) and ( *http://youtu.be/OiE9zRPD6ps*)

**Myra Cohen** is an Associate Professor in the Department of Computer Science and Engineering at the University of Nebraska-Lincoln where she is a member of the Laboratory for Empirically based Software Quality Research and Development (ESQuaReD). She is a recipient of a National Science Foundation early CAREER development award and an Air Force Office of Scientific Research young investigator program award. Her research expertise is in testing highly configurable software, GUI testing, combinatorial interaction testing, applications of combinatorial designs to software engineering, and search based software engineering.

## V. Links to Tutorial Materials

The tutorial materials will be hosted on the COMET website [3] and can be found at http://comet.unl.edu/tutorial.php. COMET includes software subjects combined with artifacts such as models and coverage adequate test suites and oracles, process descriptions, and open source tools. We also use the GUITAR [2] to demonstrate how to extract models, generate and replay tests. Other material comes from courses and papers on testing event-driven systems from the presenters' respective websites (see: http://www.cse.unl.edu/~myra and http://www.cs.umd.edu/~atif).

## Acknowledgments

This tutorial is supported in part by the National Science Foundation through award CCF-1161767, CNS-1205472 and CNS-1205501 and by the Air Force Office of Scientific Research, award FA9550-10-1-0406. The views and conclusions in this manuscript and tutorial are those of the authors and do not necessarily reflect the position or policy of NSF or AFOSR.

## References

[1] A. M. Memon, M. L. Soffa, and M. E. Pollack, "Coverage criteria for GUI testing," in *Proceedings of the European Software Engineering Conference and ACM SIGSOFT International Symposium on Foundations of Software Engineering (ESEC/FSE)*, 2001, pp. 256–267.

[2] "GUITAR – a GUI Testing frAmewoRk," website, 2009, http://guitar.sourceforge.net.

[3] "COMET - COMmunity Event-based Testing," website, 2012, http://comet.unl.edu/.

[4] A. Swearngin, M. Cohen, B. John, and R. Bellamy, "Easing the generation of predictive human performance models from legacy systems," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, (CHI)*, 2012, pp. 2489–2498.

[5] A. Memon, I. Banerjee, and A. Nagarajan, "GUI ripping: Reverse engineering of graphical user interfaces for testing," in *Proceedings of the 10th Working Conference on Reverse Engineering*, ser. WCRE '03, 2003, pp. 260–269.

[6] X. Yuan, M. Cohen, and A. Memon, "GUI interaction testing: Incorporating event context," *IEEE Transactions on Software Engineering*, vol. 37, no. 4, pp. 559 –574, 2011.

[7] S. Huang, M. B. Cohen, and A. M. Memon, "Repairing GUI test suites using a genetic algorithm," in *International Conference on Software Testing (ICST)*, April 2010, pp. 245–254.

[8] M. B. Cohen, S. Huang, and A. M. Memon, "AutoInSpec: Using missing test coverage to improve specifications in GUIs," in *International Symposium on Software Reliability Engineering (ISSRE)*, November 2012, pp. 245–254.