Special issue on new generations of UI testing

SOFTWARE TESTING, VERIFICATION AND RELIABILITY
Softw. Test. Verif. Reliab. 2021;**31**:e1770
Published online 17 March 2021 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/stvr.1770

Editorial

Special issue on new generations of UI testing

Market demands for faster delivery and higher software quality are progressively becoming more stringent. A key hindrance for software companies to meet those demands is how to test the software due to the intrinsic costs of development, maintenance and evolution of testware, especially since testware should be defined and aligned, with all layers of the system under test (SUT), including all user interface (UI) abstraction levels. UI-based test approaches are forms of end-to-end testing. The interaction with the system is carried out by mimicking the operations that a human user would perform. Regarding graphical user interfaces (i.e., GUIs), different GUI-based test approaches exist according to the layer of abstraction of the GUI that is considered for creating test locators and oracles: specifically, first generation, or coordinate-based, tests use the exact position on the screen to identify the elements to interact with; second generation, or layout-based, tests leverage GUI properties as locators; and third generation, or visual, tests make use of image recognition.

The three approaches provide various benefits and drawbacks. They are seldom used together because of the costs mentioned above, despite growing academic evidence of the complimentary benefits. User interfaces are, however, not limited to GUIs, especially with the recent diffusion of innovative typologies of user interfaces (e.g., conversational, voice-recognition, gesture-based and textual UIs) that are still rarely tested by developers; testing techniques can also be distinguished based on the way the test scripts are generated, i.e., if they are written inside JUnit-like test scripts or obtained through the capture of interactions with the SUT, or automatically obtained traversing a model of the user interface, as modern model-based testing tools do it. Test automation is a well-rooted practice in the industrial environment. However, there are software development domains, e.g., web and mobile apps, where UI testing is still not adopted on a systematic basis. The results of many investigations in literature highlighted many reasons for this lack of penetration of the most evolved UI testing techniques among developers:

1 Scarce documentation of the available testing tools;
2 Significant maintenance effort when keeping the test scripts aligned with the evolution of the AUT, e.g., for performing regression testing;
3 Limited perception of the benefits that advanced UI testing techniques yield when confronted with traditional manual testing.

The present special issue is focused on the concept of software testing in general, since it will not take into account other forms of testing (e.g., unit, integration, performance testing) that are at lower layers than E2E testing and that, in general, do not involve the final UI of the application under test. On the other hand, the proposed special issue will not have a focus on a specific application domain. The goal is to provide the reader with a broad perspective of the UI-based E2E testing process automation regardless of the domain in which an application falls.

The reviewing process of the received papers followed the same process as for regular papers. At least three reviewers reviewed each paper, and after a rigorous selection process, five papers were proposed for publication.

The first paper, 'Functional Test Generation from UI Test Scenarios using Reinforcement Learning for Android Applications' by Yavuz Koroglu and Alper Sen, presents a methodology to

generate GUI test scenarios for Android applications, named FARLEAD-Android (Fully Automated Reinforcement LEArning-Driven Specification-BAsed Test Generator). The test generator defines test sequences based on formal specifications as linear-time temporal logic formulas. The authors' evaluation proved the technique more efficient than three other state-of-the-art testing engines, Random, Monkey and QBEa.

The second paper, 'TESTAR – Scriptless Testing through Graphical User Interface' by Tanja Vos, Pekka Aho, Fernando Pastor Ricos, Olivia Rodriguez Valdes and Ad Mulders, describes an open-source tool, TESTAR, which complements scripted testing with scriptless test automation. The manuscript provides a comprehensive description of the characteristics and features of TESTAR, along with an overview of the state of the research and experimentation agenda with the tool.

The third paper, 'Comparing the Effectiveness of Capture and Replay against Automatic Input Generation for Android GUI Testing,' by Porfirio Tramontana, Sergio Di Martino, Anna Rita Fasolino and Luigi Starace, describes two experiments conducted to compare the effectiveness of capture & replay vs. freely available automated testing tools. The experiments that involved a sample of computer engineering students showed that the generation of test cases with capture & replay techniques outperformed the automated tools in terms of achieved coverage, especially for complex execution scenarios.

The fourth paper, 'Generating and selecting resilient and maintainable locators for Web automated testing' by Vu Nguyen, Than To and Gia-Han Diep, defines an approach to generate and select resilient maintainable locators for automated web GUI testing, relying on the semantic structure of web pages. The approach outperformed state-of-the-art tools (namely, Selenium IDE and Robula+) in the capability of locating target elements and avoiding wrong locators.

Finally, the fifth paper, 'SIDEREAL: Statistical Adaptive Generation of Robust Locators for End-to-End Web Testing,' by Maurizio Leotta, Filippo Ricca and Paolo Tonella, tackles the issue of generating robust XPath locators by interpreting it as a graph exploration problem instead of relying on ad-hoc heuristics as the state-of-the-art tool Robula+. The authors describe a tool, SIDEREAL, which outperforms Robula+ in robustness to broken XPath locators.

Hereby, we would also like to thank all the authors who considered the special issue on New Generations of UI Testing in the *Software Testing, Verification and Reliability Journal* for publishing their research articles, and also all the reviewers whose review comments and recommendations helped us to ensure the quality of the special issue and also helped the authors to improve their papers. A special thanks to the STVR chief editors Robert M. Hierons and Tao Xie for their guidance and support throughout this process.

Emil Alégroth
Department of Software Engineering, Faculty of Computing, Blekinge Institute of Technology, Karlskrona, Sweden

Luca Ardito
Department of Control and Computer Engineering, Politecnico di Torino, Corso Duca degli Abruzzi, 24, Turin 10129, Italy

Riccardo Coppola
Department of Control and Computer Engineering, Politecnico di Torino, Corso Duca degli Abruzzi, 24, Turin 10129, Italy

Robert Feldt
Department of Computer Science and Engineering, Chalmers University of Technology, Sweden

DOI: 10.1002/stvr