Advances in Test Automation for Software with Special Focus on Artificial Intelligence and Machine Learning

J. Jenny Li
Kean University
Union, NJ, USA

Andreas Ulrich
Siemens AG
Munich, Germany

Xiaoying Bai
Tsinghua University
Beijing, China

Antonia Bertolino
ISTI-CNR
Pisa, Italy

Software testing is an integral part of the software engineering (SE) discipline. Effective testing with reduced costs can be achieved through automating the testing process. In the past decades, a great amount of research effort has been spent on automatic test case generation, automatic test selection, automatic test oracles, etc. and there has been a rapid growth of practices in using automated software testing tools. Work on this topic has long been published as an important part of the software engineering discipline, and in recent years testing is consistently among the top-most popular topics in submissions to SE conferences.

In the past few years, a large number of software test tools have been developed and made available on the market. The practice of software test automation (TA) has also moved forward significantly, from record-and-replay techniques to support automated testing of graphical user interfaces to unit test frameworks such as JUnit that operate at code level, till test data generation tools as KLEE or Pex. However, further progress in TA is still required. Software systems have become more and more complicated with components developed by different vendors, programmed in different programming languages, and even running on different platforms. Few software testing tools can automatically adapt to support all testing tasks within one environment. The advent of cloud and mobile computing has imposed further grave challenges to software TA. Moreover, the recent progress in artificial intelligence (AI) and machine learning (ML) of self-adaptive and autonomously acting systems, such as self-driving cars, is another contributing factor to these new challenges. There is an urgent need to develop the TA technology for this emerging class of AI/ML empowered software. On the other hand, AI/ML technologies themselves can be applied to support TA in new and intelligent ways, e.g., to support the classification of test outputs in non-functional testing. AI-based testing could further push smart automation and digitization in the society. Further, TA has become an important factor for the development of software testing methodologies for heterogeneous software and contributes to the progress of software testing as a scientific discipline.

This special issue aims at bridging the gap between theory and practice in TA in order to improve the current state of practice and to foster innovative research in the area. It invited extended versions of best papers from the ICSE 2018 Workshop on “Automation of Software Test” (AST 2018), held in Gothenburg, Sweden in June 2018, as well as other relevant works through an open call-for-papers. Among the eight submissions that answered our call, following the rigorous journal reviewing process four papers have been finally selected that focus on recent

but solid work with promising results. Two papers are directly related to the application of AI and ML in test automation, while two other papers address the topic of TA in domains that are anyhow AI-relevant, such as rule-based and mobile systems.

The first paper "An Automated Model-based Testing Solution for Access Control Systems" reports on the systematic testing of access control systems, known as Policy Decision Points, of which the decision rules are specified in XACML, a standardized declarative access control policy language in XML syntax. Rules are often used in AI expert systems to represent the knowledge base of an agent. The paper provides the XACMET approach that translates an XACML specification into an XAC graph, from which access request values are derived to systematically cover the decisions paths contained in the graph. In addition, the approach derives the decision outcome for each request by interpreting the policy rules and policy combination algorithms. This decision outcome serves as an oracle in testing. The approach is evaluated and compared in terms of coverage with the earlier X-CREATE approach by the same research group using several realistic policy specifications.

The second paper "Virtualization of Stateful Services via Machine Learning" proposes an approach to create stateful service mocks using two different machine learning techniques to support testing, which were evaluated and compared on three case studies. The comparison included MINT that is an EFSM inference tool, adopting the classification and sequence-to-sequence based machine learning algorithms. The work addresses interesting research on service virtualization for supporting testing of systems with service-oriented architectures.

The third paper "Planning-based Security Testing of Web Applications with Attack Grammars" contributes to the security testing of web applications, proposing to apply AI techniques to test common XSS and SQL injections vulnerabilities. The idea consists of automatically deriving test cases that mimic unexpected user sequences and input values for detecting those vulnerabilities. In particular, the authors use planning models for both generating potential attacks (which are abstract test cases expressed in PDDL), and then deriving attack vector models that can be translated into concrete test cases using a defined attack grammar as reference. The implemented framework is evaluated on the OWASP Mutillidae 2 Project.

The fourth paper "Sentinel: Generating GUI Tests for Android and Android Wear Sensor Leaks" focuses on the automated generation of test cases for apps that run on the devices powered by either Android or Android Wear. It is somewhat less relevant with AI, however some people consider wearable computing a part of AI. The key goal is to generate tests that can identify parts of a code base that fail to disable sensors that are not needed. Using both Android and Android Wear apps, the authors conducted an experiment with a testing tool called Sentinel. The results suggest that this tool can highlight a small subset of GUI event sequences that are likely to lead to a mobile device excessively draining its battery because its app software did not correctly disable unused sensors. The technique is based on the identification of patterns of callback sequences that could possible lead to a sensor leak.

Overall, we hope that this special issue will stimulate new and continued research on test automation for software in general and the application of AI/ML to test automation and to automate the testing of AI/ML systems in particular.