AI for Testing Today and Tomorrow: Industry Perspectives

Tariq M. King
Ultimate Software Group, Inc.
Weston, FL 33028
tariq_king@ultimatesoftware.com

Dionny Santiago
Ultimate Software Group, Inc.
Weston, FL 33028
dionny_santiago@ultimatesoftware.com

Wendy Chin
Intel Corporation
Penang, Malaysia
wendy.chin@intel.com

Jason Arbon
Test.ai
San Francisco, CA 94107
jason@test.ai

David Adamo
Ultimate Software Group, Inc.
Weston, FL 33028
david_adamo@ultimatesoftware.com

Ram Shanmugam
AutonomIQ
Palo Alto, CA 94303
ram@autonomiq.io

***Abstract*—With modern advances in artificial intelligence (AI) and machine learning and their applications to software testing, the intersection of AI and testing is receiving close attention. The 2018 Annual Western Conference on Software Testing Analysis and Review featured a two-session panel on AI for Software Testing (AIST). The panel brought together six industry experts with experience developing AIST products, services, and research prototypes. Questions sourced from the industrial testing community were used to provoke thought, stimulate conversation, and guide panel discussions. This paper provides a review of the industry panel, which includes discussions on the visions, ideas, thoughts, strategies, directions, and lessons learned developing systems that use AI to test software, applying methods to test AI systems, and designing self-testing systems. Both the testing community survey and the expert panel yielded insightful perspectives on AIST in practice.**

***Index Terms*—Artificial Intelligence, Machine Learning, Software Testing, Industry, Panel, Survey**

I. Introduction

Artificial intelligence (AI) and machine learning (ML) are transforming multiple sectors of the economy, and impacting several aspects of our daily lives [1]. Workplaces such as those in finance, healthcare, retail, education, and technology are leveraging AI to automate tasks, reduce costs, and make data-driven decisions. In our homes, AI is powering television and movie recommendations, personal digital assistants, security cameras, and home automation. As AI continues to permeate our world, it is becoming more and more critical to validate that these types of systems are functional, safe, secure, performant, available, and resilient. In other words, AI needs testing.

Industries everywhere are being disrupted by the application of AI and ML, and the testing industry is no exception. Testing involves evaluating a product by learning from it through experimentation, which includes studying, questioning, modeling and inferencing [2]. However, the current state of the art in automated testing still requires significant manual effort [3]. Researchers and practitioners are recognizing the potential for AI and ML to bridge the gap between human-present and machine-driven testing capabilities. As a result, several start-ups are attempting to develop AI-powered automated testing tools [4]–[8], and these tools are generating what some consider to be a much needed industry buzz in testing innovation. Hence, it seems that just like AI needs testing, testing needs AI.

Artificial Intelligence for Software Testing (AIST) is an emerging field aimed at developing AI tools to test software, devising methods to test AI systems, and designing software that is capable of self-testing and/or self-healing [3]. The 2018 Annual Western Conference on Software Testing Analysis and Review featured a two-session panel on AIST [9]. The panel brought together industry experts actively working on AIST products, services, and research projects to share their thoughts and practical experiences on the subject. This paper presents a review of the panel for the purpose of sharing industrial perspectives, opinions, thoughts, and expertise on AIST. The major contributions of this paper include:

1) Presents and discusses the results of a community survey on the impact of AI-driven testing, conducted as a precursor to the panel.

2) Provides a detailed set of organizer-seeded and community-sourced questions related to the current state and future direction of AIST.

3) Summarizes the panel discussions on AI-driven testing, testing AI systems, and self-testing that occurred during the sessions, and provides references that link to recommended resources.

The remainder of this paper is organized as follows: Section II contains background information on AIST and motivation for its increased attention in the software industry. Section III presents the community survey on the impact of AI on testing. Section IV describes the panel. Section V summarizes the panel discussions, and Section VI concludes the paper.

II. AI for Software Testing

Figure 1 depicts AIST as an intersection of three sub-fields: *AI Testing*, *Testing AI*, and *Self-Testing* [3]. This section describes each sub-field and provides motivating rationale for its increased attention in the software industry.

A. AI Testing

In software testing, the term *test automation* is often used to refer to manually encoding a predefined set of program input actions and output verification steps into a script, which can in turn be executed by a machine [10]. Upon execution, a log of the results is generated, stored, and associated with the run. The only aspects of this process that are automated are test execution and logging. Human testers are needed to define testing goals, acquire the knowledge necessary to adequately test the software, design and specify detailed test scenarios, write the test automation scripts, execute scenarios that could not be automated, and analyze the test results to determine any threats to the project.

With the mainstream of testing being focused on manual testing, and the manual creation of test scripts [3], researchers and practitioners have started investigating how AI and ML can be leveraged to build the next generation of testing tools. The idea is to utilize advances in AI/ML, cloud computing, and big data technologies to help bridge the gap between human-present and machine-driven testing. The use of AI/ML technologies in the performance of automated testing activities is referred to as *AI-driven testing* throughout this paper.

B. Testing AI

The adoption of AI and ML techniques creates software quality and testing problems. AI-based systems tend to be complex, non-deterministic, and self-adaptive [11]. In addition, they introduce a new set of testing challenges related to feature computation, sampling quality, outlier tolerance, label quality, quality drift and traceability [12]. ML-based products may have to be re-trained from scratch with each new version, and are difficult to predict and debug. As a result, there is a growing need for applied testing techniques, standardized toolsets, and best practices for testing AI. However, there are few industrial case studies and experience reports that share the results of testing AI-based systems in practice [12].

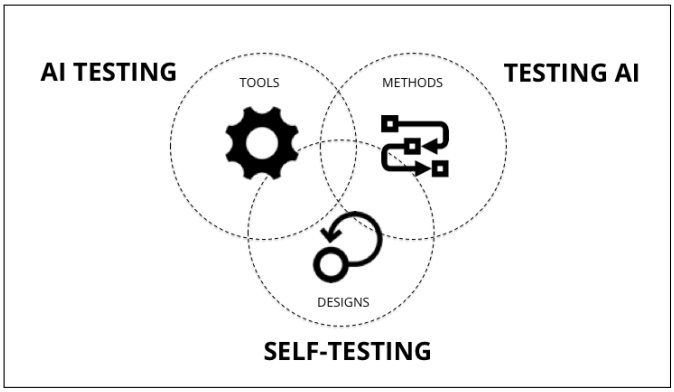

Fig. 1. Artificial Intelligence for Software Testing: Intersecting Sub-Fields.

C. Self-Testing

Dynamic adaptation in AI-based systems can either be: stochastic – resulting from random selection during learning, or environmental – learning based on feedback passed into the system from end users or other systems [11]. However, regardless of the source, adaptation causes the internal logic of the system to change at runtime.

Self-testing refers to the ability of a system or component to monitor its dynamically adaptive behavior and perform runtime testing prior to, or as part of the adaptation process [13]. Existing research on self-testing and runtime testing of dynamically adaptive systems is therefore applicable to AI-based software. However, few of the approaches found in the literature are performed in the context of AI-based systems [13]–[15]. To the best of our knowledge, there have been little to no industrial applications of self-testing in AI-based systems. As a result this area appears to be wide open for research investigation.

III. Survey: Impact of AI-Driven Testing

This section discusses the objectives, setup, and results of a 2017 survey on the impact of AI on software testing [16]. The survey was conducted online as a precursor to the expert panel and other community-based activities. Survey recipients included an e-mail distribution list of members of the Artificial Intelligence for Software Testing Association (AISTA). Participants included a cross-section of software professionals who are: (a) interested in how AI applies to testing, and/or (b) currently practicing and publishing important work that applies AI to testing.

A. Objectives and Setup

The objectives of conducting the survey were to collect information that could be used to gain insights on participants': (a) familiarity with AI and machine learning; (b) beliefs on whether AI will have a significant impact on, or even replace manual testing; (c) interest in attending in-person and/or online AIST conferences; and (d) preferences for receiving updates and educational content on the subject.

The survey was distributed to approximately 500 participants, out of which 328 provided responses. Most of the participants held the job title of test automation engineer, test manager/director, and manual software tester. Typeform [17], a survey tool which attempts to keep users engaged by presenting questions one at a time, was used to create and distribute the survey.

B. Results and Discussion

Figures 2-7 contain the questions and responses for each section of the community survey. The results suggest that although only 22% of the participants are familiar with AI/ML, 76% of them believe that this technology will have a significant impact on manual testing within 3 years. Approximately

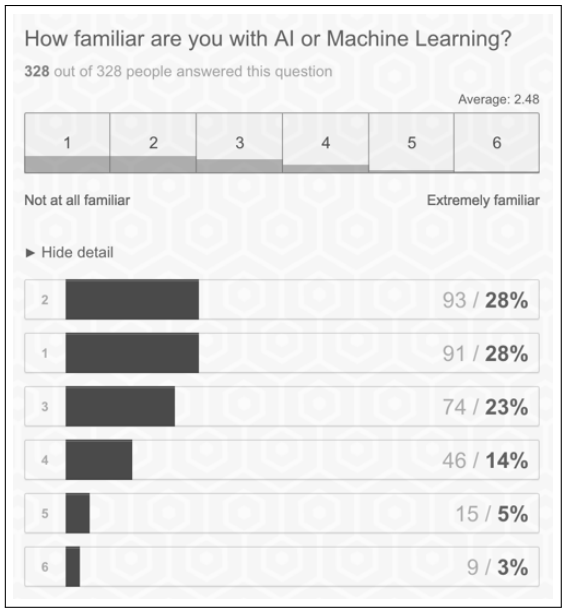

Fig. 2. Familiarity with AI or Machine Learning.

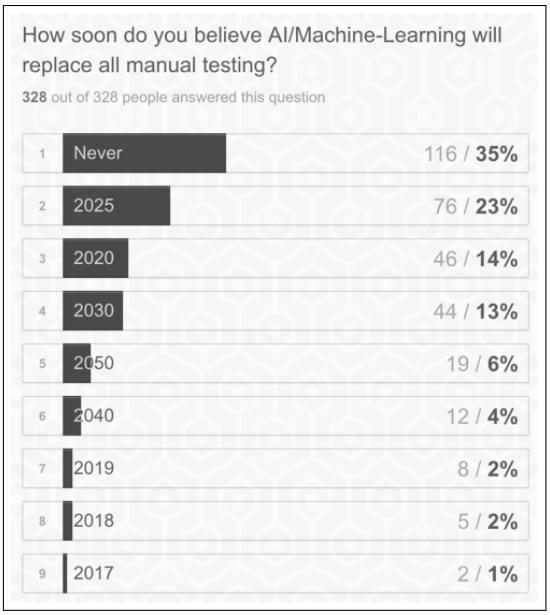

Fig. 4. Proximity to Replacement of Manual Testing.

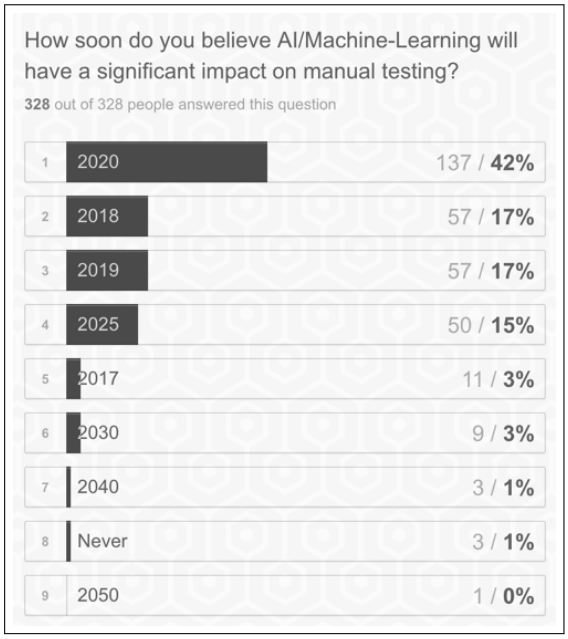

Fig. 3. Proximity to Significant Impact on Manual Testing.

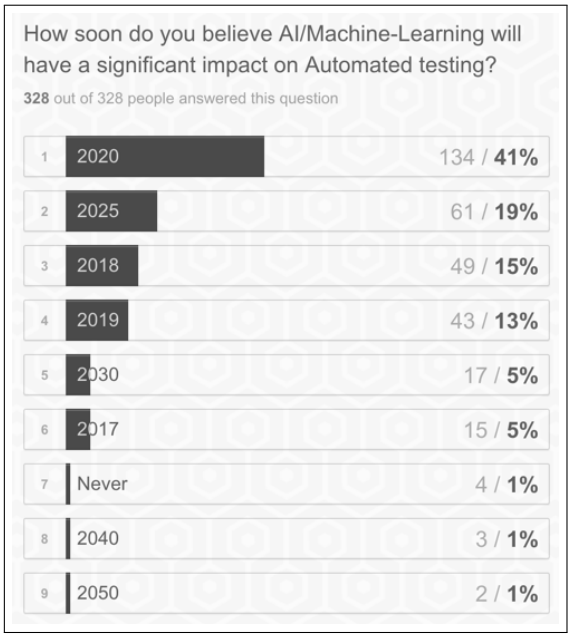

Fig. 5. Proximity to Significant Impact on Automated Testing.

a third of the participants do not think that AI will replace manual testing. However, the data appears to be a bit polarized as the next largest vote was for AI to replace manual testing as soon as 2025, with votes trailing to 2050. Very few of the participants think that AI will replace manual testing in the next couple of years.

Looking at the types of participants behind the aforementioned numbers shows that half of the manual testers believe AI will replace their job by 2025. On the other hand, test automation engineers made up almost all of the *Never* votes. Perhaps manual testers are hopeful, or consider their job is ripe for the machines, while automation engineers are thinking of how difficult and daunting the effort would be to fully automate testing.

On the subject of automated testing, 74% of the participants believe that AI will have a significant impact on test automation which will occur between 2017 and 2020. If this is to be the case, it means that test automation engineers should start learning AI soon as it will likely be overwhelming to do so later. 59% of participants think the job of test automation

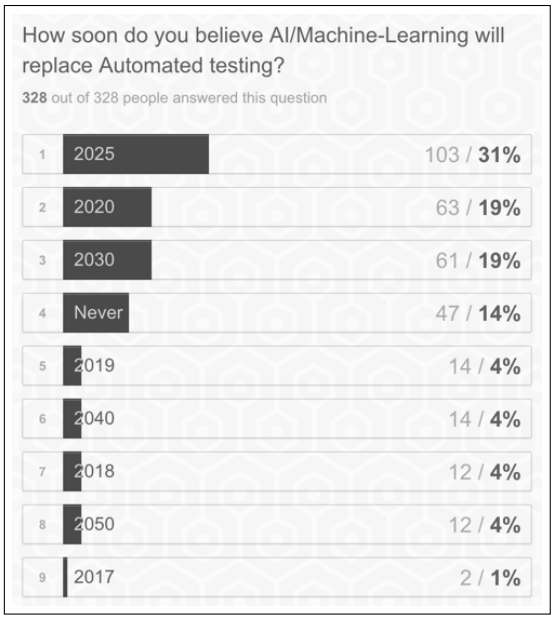

Fig. 6. Proximity to Replacement of Automated Testing.

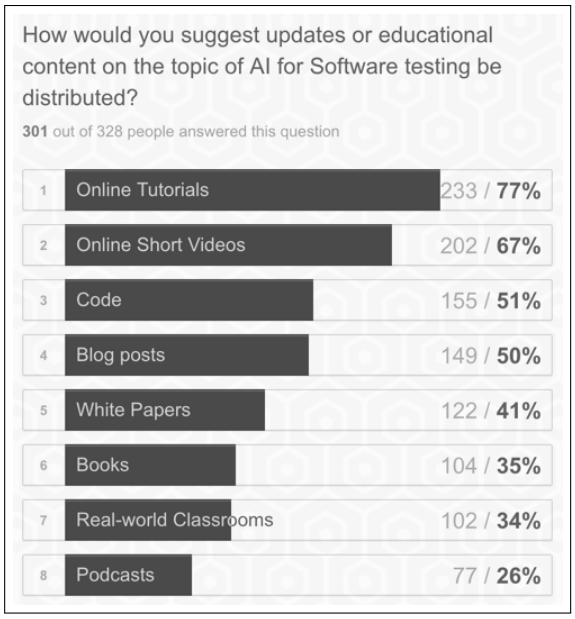

Fig. 7. Distribution of Updates and Educational Content.

engineers will be replaced in the next 5 to 10 years, which are not great odds to keep making a career of it. 86% of participants believe that test automation jobs will be replaced by AI at some point.

The most popular communication mechanisms for updates and educational content are online tutorials, short videos, code/tools, blog posts, and research/white papers. In contrast, books, real-world classrooms and podcasts were the least favorite. In a short follow-up survey, participants were asked if they would be willing to attend a one-day conference including an expert panel on AIST, and 98% of participants responded positively.

IV. Panel Description

The panel was hosted by the AI for Software Testing Association (AISTA) [3], in conjunction with the Techwell Corporation [9]. The live audience was made up of a total of 144 conference delegates, with 101 attending the first session and 43 attending the second session. Most, if not all, of the delegates were software testing professionals, and each session was approximately one-hour long. This section contains abstracts that describe each panel session, biographies of the panelists, and pools of questions used to stimulate panel discussions.

A. Abstracts

The abstracts were used by the organizers prior to the conference to motivate delegates to attend, and by the host at the beginning of each session to help guide discussions.

AI for Testing Today: How can we apply AI to the testing problems of today? What works and what doesn't work? Are there any critical lessons that have been learned? This panel discusses AI frameworks and approaches that are most useful today for testing real world systems. Hear advice from the folks that have had successes and failures applying AI to real world testing problems so you can avoid the same mistakes and bring back proven techniques for applying AI to software testing challenges.

AI for Testing Tomorrow: What does AI mean for the future of testing? What aspects of testing will the machines replace? What things will AI soon be better than humans at and what things will humans always do better than AI? This panel explores the future of AI for testing including thoughts on how humans can prepare for a future of testing where we work alongside AI. Hear experts discuss their views on the future impact of AI in testing and where the boundary between human and AI-powered testing truly lives.

B. Panelists

The panel was comprised of six industry experts across four different companies working on AIST research and development projects. These included:

Tariq King is the Head of Quality at Ultimate Software. He leads Ultimate's Quality Center of Excellence, which includes an AIST research and development team. Tariq holds a Ph.D. in Computer Science from Florida International University. His primary research interest is engineering autonomous, self-testing computing systems.

Jason Arbon is the CEO at test.ai where his mission is to test all the world's apps. Jason previously worked on: web search at Google and Bing, the web browsers Chrome and Internet Explorer, operating systems such as Windows and ChromeOS, and crowd-sourced testing at uTest.com. He is the co-author of How Google Tests Software.

PANEL PART I: AI FOR TESTING TODAY	
Seeded Questions (Organizers)	
Impacts Challenges Problems	Why are there so many different answers to how AI is impacting testing? What are the easiest testing problems for AI to solve today? What are some of the hardest testing problems for AI to solve today?
Testing AI	What are the best approaches to testing AI-based systems today? What are the biggest challenges to testing AI-based systems?
Sourced Questions (Community)	
Tools Frameworks	Are there any open-source tools for implementing AI in testing today? Any tools for object identification without having to manually construct selectors? Do you know of any AI tools that can be used to analyze user stories?
Strategies Methods	Why is there less attention on testing AI compared to AI-driven testing? Is there a high-level strategy for testing AI systems?
Benefits Advances	Can AI testing reduce human effort when testing multiple apps in a domain? Can AI solve the problem of automating photo\video quality testing? Can AI be used to find defects or potential defects before coding starts?
Impacts Challenges Problems	What are common challenges a tester is facing when it comes to AI? What challenges do we face today in AI for testing? What are the top three problems that we should focus on today?
Trust Security	How can we build confidence in AI-driven testing tools? Is it possible to have your own AI and train it because you need more security? How?
Adoption Transition	Why is AI important for a tester? What is the simplest AI functionality to introduce into conventional automation? How do we incorporate AI-driven testing into current traditional processes? If a company wanted to get started today, what are the first steps to take? In what timeframe after investing should companies expect some minimal return? What will the initial costs be?

(a)

PANEL PART II: AI FOR TESTING TOMORROW	
Seeded Questions (Organizers)	
Takeover	Will AI replace all testing jobs? If so when? What is the first testing role that will be eliminated by AI? What will our jobs be when AI for testing becomes mainstream?
Advances	What AI for testing advances do you foresee in the next 1-2 years? What about 4-5 years into the future, and beyond? What is self-testing and when will it be widely deployed?
Adoption	How should test engineers approach the transition? How can we start to prepare for a future where AI tests software?
Sourced Questions (Community)	
Benefits Advances	Will AI eventually be used to test all layers of the tech stack? Where will it provide the greatest advantage? Will AI be able to generate models by themselves? Will that also lead to AI building the application? Will AI bring us closer to bridging the coverage gap?
Roadmap Planning	What steps are needed for AI to understand systems like a human? What is the roadmap for building a new AI-driven testing tool? Is there a plan to develop tools that will help the AI testing industry? Can you give three points/milestones along the path to AI testing?
Adoption Transition	Will testers need programming skills? Should testers learn languages like Java, JavaScript, and Python? How can a tester improve their skills to work with AI for testing? For experienced manual testers, what is the first step to adapt to the AI shift?
Community	What is the plan for AI for testing as a community? Does AISTA intend to facilitate sharing tips, code, and models?

(b)

Fig. 8. Seeded and community-sourced questions on issues surrounding the (a) present state, and (b) future direction of AI for Software Testing.

Dionny Santiago is the Principal Architect for Quality at Ultimate Software. He is focused on research and development efforts to apply AI and machine learning to web application testing. Dionny is currently pursuing a Ph.D. in Computer Science at Florida International University. His research applies machine learning approaches to test generation.

David Adamo is a Quality Engineer at Ultimate Software working in the AIST Research Division. David holds a Ph.D. in Computer Science from the University of North Texas. His areas of research and development include reinforcement learning and event sequencing for automated GUI testing of Android applications.

Wendy Chin received her Ph.D. in Electrical & Electronic Engineering from The University of Nottingham with a specialization in computer vision and machine learning. She is currently a software engineer in the Internet of Things group at Intel. Her research interests include machine learning, computer vision, signal processing and data analytics.

Ram Shanmugam is a serial entrepreneur and CEO of multiple start-ups in machine learning, artificial intelligence, deep learning, cloud and infrastructure technologies. He has experience in product management, general management, engineering and operations at several Fortune companies, and has run business units with $15M - 500M$ in revenue.

C. Question Pools

In preparation for the panel, organizers used the feedback from the community survey in Section III to prepare a set of seeded questions for the panelists in order to stimulate initial discussions. A set of community-sourced questions were also collected and made available to both the panelists and the audience. Figures 8(a) and 8(b) provide the complete set of seeded and sourced questions for the panel sessions – *AI for Testing Today* and *AI for Testing Tomorrow*, respectively.

In the AI for Testing Today session, most of the questions addressed were related to current advances and benefits of AI-driven testing; open-source tools and data to support AI-driven testing; and approaches and challenges when testing AI systems in practice. During the AI for Testing Tomorrow session, the discussion focused on the possible AI takeover of automated testing; its adoption and transition; and the vision, approaches and directions behind self-testing systems.

V. Panel Discussion

This section provides a summary of the panel discussions, including references to the resources mentioned by the panelists and live audience.

A. AI-Driven Testing: Advances and Benefits

Since 2014, there has been a spike in the number of vendors offering AI-driven testing services [4]–[7], [18]. The majority of these vendors are start-up companies targeting system-level testing of mobile applications. The current state of the practice uses autonomous and intelligent agents, referred to as "test bots", to automate activities such as application discovery, modeling, test generation, and failure detection.

Categories of AI-driven testing approaches found in practice include: (a) *Differential* – comparing application versions over builds, classifying the differences and learning from feedback on the classification; (b) *Visual* – leveraging image-based learning and screen comparisons to test the look and feel of an application; (c) *Declarative* – specifying the intent of a test in a natural or domain-specific language [19], and having agents figure out how to carry out the test; and (d) *Self-maintaining* – auto-correcting tests when the UI changes.

Key advantages of AI-driven testing are that it is general purpose, reusable, robust, adaptive, and scalable. Machine learning approaches are applicable to many types of problems. For example, a simple feed forward neural network with a single hidden layer can be used to approximate almost any mathematical function [20]. As a result, AI-driven testing can be applied to several types of testing, applications, or domains.

AI-driven tests tend to be independent of any specific application and hence may be reused across applications in the same domain (e.g., add item to cart), or across domains (e.g., login). New tests can be generated each time, thereby eliminating the pesticide paradox, i.e., a lack of breadth in fault detection due to running the same tests repeatedly. Lastly, a major benefit of AI-driven testing is accelerated test coverage by combining AI-based test generation with large-scale test execution in the cloud.

B. AI-Driven Testing: Open-Source Tools and Data

General purpose tools commonly used to support the development of AI-driven testing systems include: TensorFlow, Keras, Scikit-Learn, Stanford NLP, spaCy, PyTorch and PyDataLog [21]–[27]. There are not many domain-specific, open-source tools and datasets that can directly be leveraged for AI-driven testing. However, during the panel discussion three such tools and their associated datasets were highlighted [4], [28], [29].

Element location using AI has been incorporated into the open-source mobile testing framework Appium [4], [30]. The feature finds web elements in mobile applications such as search text boxes, login buttons, among others. Leveraging AI for web element location makes the tests quicker to create and less brittle [31]. Furthermore, the testing community can improve the AI by adding new training data, alternative training methods, more rigorous relevance testing, or new labels. The training data is open and available on Kaggle [28], thereby allowing individuals or organizations to fork the data, clean it up, add it to their proprietary test frameworks, or compete to improve the classifiers.

AI-based Generation and Exploration iN Test (AGENT) [29] is an ML-based, multi-agent prototype for web application testing. The prototype consists of a collection of intelligent agents that work in conjunction with abstract test generation and web UI classification subsystems [32]. The agents incorporate long short-term memory units trained by human testers. For any given subsequence of steps through the application, the agents predict sequences of the next possible set of actions and assertions that a human tester might make, and automatically execute them on the system under test [32].

DrQA [33] is an open-domain question answering system that combines document retrieval with machine comprehension. It uses Wikipedia as its knowledge source for documents but does not rely on its graph structure. The repository includes code, data, and pre-trained models for querying Wikipedia as described in [34]. A number of datasets used for DrQA training and evaluation are also freely available [33]. Although DrQA is not necessarily specific to the testing domain, its machine comprehension capabilities make it a good candidate for implementing automated test oracles in AI-driven systems.

Two additional datasets to support web and mobile UI testing were mentioned [35], [36]. The Rico data set [35] is an open collection of 72K mobile UI screenshots mined from 9.7K Android applications. The latest version of the dataset augments the original collection with annotations on component categories, text button concepts, and icon classes. ClueWeb12 [36] was developed for research on information retrieval and related human language technologies. The dataset consists of 733 million English web pages, collected between February and May, 2012. It is a companion or successor to the ClueWeb09 web dataset.

C. Testing AI Systems: Practical Approaches and Challenges

Testing AI-based products is different than traditional software testing. For example, during development, testing a supervised machine learning system is more focused on data and analytics than formulating creative test cases, manually exploring the application, or writing automated test scripts.

Testing supervised ML can be divided into two major phases: training validation and relevance testing. Training validation is about testing the training data. During the validation phase, some key questions should be asked about the training data. For example, are labels correct and relevant? Is the sample representative and proportional? Are there sufficient samples? Training sets are often large, and labeling may have been done manually. As a result, samples are sometimes mislabeled and hence it is important to check all the labels for correctness.

Labels must be relevant to the context for which the machine is being trained. For example, if an AI-based search engine is trained using only non-engineers, the results it produces may not work well for engineers. Samples should represent the

types of users and data expected in production to avoid training a system that does not work in the real world. Relevance of the artificial brain should be tested in isolation and then again when the AI is integrated into the final product.

Two AI-based testing challenges that are gaining much attention in industry are: (1) non-determinism – the same inputs can produce different outputs as the system learns and adapts, and (2) fuzzy oracles – it may be very difficult to determine whether or not responses are right or wrong, and instead correctness may need to be measured on a continuous scale [37]. Other challenges include performance, security, and systems integration testing [38].

D. AI-Driven Testing: Takeover, Adoption, and Transition

A hypothesis on the possible AI takeover of automated testing was that it is already underway and will progress gradually. First, AI will be used to enhance existing tools and frameworks that target specific testing problems in isolation. Examples at this time of writing include functional testing of web and mobile applications, visual testing of user interfaces, and UI element location and auto-correcting element selectors.

Next, AI will start to replace entire technology stacks for automated testing. At all testing levels, AI will take over automation tasks that require decisions that a human could make in less than a second. Initially, higher order testing tasks will still require human input or intervention. However, over time as AI technology progresses, and as the machines are trained on the actions of the higher order tasks, AI will take over those activities as well.

So what can testers do to prepare for a world in which AI tests software? The initial step is to determine whether you are interested in designing and developing AI-driven systems, or being an end user of these tools. An engineer who is building these types of systems needs to have a good understanding of AI and machine learning. Both MIT and Stanford University have freely available introductory courses on machine learning [39], [40]. King and Arbon [41] have open-sourced the materials to an introductory course on AI and ML skills for the testing world, including the slides and tools used during the exercises [29], [42].

It is unlikely that end users of AI-driven testing tools will need to have deep knowledge of machine learning. Vendors are likely to provide easy to use interfaces and APIs for using AI-driven testing features and customizing pre-trained AI models. This trend is already being followed with the advent of AI tools and platforms like Google's AutoML Vision [43], H2O.ai [44] and DataRobot [45].

E. Self-Testing: Vision, Benefits, Approaches, and Directions

The autonomous and adaptive nature of AI-based software means that self-testing will likely have to be an inherent part of these types of systems. King [13] describes two approaches to self-testing in dynamically adaptive systems – *Replication with Validation* (RV) and *Safe Adaptation with Validation* (SAV). RV tests changes on copies of the component under test, and SAV tests changes in-place directly on the component under test. Each approach comes with its own set of advantages and disadvantages. For example, RV has the advantage that testing can be distributed to a separate computational node, thereby minimizing disruptions to the actual services in production. However, a disadvantage of RV is that copies of components need to be maintained specifically for testing purposes.

Some promising directions associated with overcoming some of the challenges with designing self-testing systems include the rapid adoption of microservices and event sourcing [46]. Building a system using small, independently deployable and scalable microservices makes it easy to clone and distribute components under the RV strategy. In addition, microservices promote the development of robust systems that can still operate when some services go offline. SAV benefits from robust designs since the live system generally has to be placed in a *safe* state, where the components targeted for testing have to be temporarily blocked from receiving user requests. Event sourcing facilitates bringing the system or a component to a specific state prior to test execution, thereby increasing its overall runtime testability.

VI. Conclusion

There is a buzz in the software testing industry around topics related to the intersection of AI and software testing. A preliminary survey of testing professionals showed that test engineers, manual testers, and test managers are all curious about the impacts of AI on testing. Survey participants expressed interest in attending virtual and physical conference events to engage in conversations related to AI-driven testing, testing AI systems and self-testing. Hosting one such event in the form of an expert panel attracted a relatively large audience, especially considering the panel occurred during time slots with several competing presentation sessions.

Both the survey and the panel were informal but insightful. Panelists provided a wealth of information including tips, benefits, challenges, strategies, training materials, open-source tools, datasets and more. A threat to the validity of our results is the small size of the survey and number of panelists. Future work involves expanding the survey and panel to a broader audience and conducting formal studies on various areas of AIST. We hope that our work will encourage others to perform their own AIST studies in both academia and industry, and conclude with these recommendations:

- *Support Tool and Method Transitions* – A quick search of the literature reveals multiple academic works at the intersection of AI and testing [11], [32], [38]. However, only a few of these works are backed by real-world case studies, or result in industrial tools and methods. However, we anticipate that migration of academic AIST research to industry will play a key role in its success, and therefore encourage these communities to support it.
- *Enable Data Collection* – For those leveraging AI to perform automated testing, it is important to collect data in the form of images for application screens and widgets, requirements documentation, test cases, among others. Investing in the development of tools to automatically

collect training data, and freely sharing those tools and datasets will help to make AI-driven testing a reality.

- *Invest in Testable and Explainable AI* – Since AI-based systems are complex and dynamically adaptive [11], the system should be designed for testability and explainability [13], [47]. In other words, building trust in these types of systems involves designing and developing AI systems that are capable of: demonstrating or explaining their rationale, characterizing their present behavior, conveying their future behavior and intent, and testing themselves as they adapt to changing conditions [47].

ACKNOWLEDGMENTS

The authors would like to thank Alison Wade, Raj Rao, Philip Daye, and the STARWEST 2018 program and organizing committees for their contributions to this work. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors, and do not necessarily reflect the views of Ultimate Software, Test.ai, Intel, or AutonomIQ.

REFERENCES

[1] R. Stanley. 16 Examples of Artificial Intelligence Across 6 Industries. [Online]. Available: https://callminer.com/blog/16-examples-of-artificial-intelligence-across-6-industries/

[2] J. Bach and M. Bolton, "Rapid Software Testing Class Materials," http://www.satisfice.com/rst.pdf, 2018, accessed: 2018-12-21.

[3] T. M. King and J. Arbon, "AI for Software Testing Association," https://www.aitesting.org/, 2018, accessed: 2018-12-21.

[4] Test.ai, "AI-Powered Test Automation," https://test.ai/, 2018, accessed: 2018-12-21.

[5] Mabl, "Advancing QA using ML," https://www.mabl.com/, 2018, accessed: 2018-12-21.

[6] Applitools, "AI-Powered Visual Testing and Monitoring," https://applitools.com, 2018, accessed: 2018-12-21.

[7] TestIM, "Agile, Self-Healing End-to-End Automation that Everyone Can Use," https://testim.io/, 2018, accessed: 2018-12-21.

[8] AutonomIQ, "Autonomous Testing, Data Generation, Scripting, Execution and Maintenance," https://autonomiq.io/, 2018.

[9] Techwell Corporation, "STARWEST Software Testing Conference," https://starwest.techwell.com/, 2018, accessed: 2018-12-21.

[10] D. J. Mosley and B. Posey, *Just Enough Software Test Automation*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2002.

[11] T. Menzies and C. Pecheur, "Verification and validation and artificial intelligence," *Advances in Computers*, vol. 65, pp. 154–203, 2005.

[12] J. Arbon, "AI for Software Testing," in *Pacific NW Software Quality Conference*. PNSQC, 2017.

[13] T. M. King, A. E. Ramirez, R. Cruz, and P. J. Clarke, "An integrated self-testing framework for autonomic computing systems." *JCP*, vol. 2, no. 9, pp. 37–49, 2007.

[14] E. M. Fredericks, A. J. Ramirez, and B. H. C. Cheng, "Towards run-time testing of dynamic adaptive systems," in *Proceedings of the 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, ser. SEAMS '13. Piscataway, NJ, USA: IEEE Press, 2013, pp. 169–174. [Online]. Available: http://dl.acm.org/citation.cfm?id=2487336.2487363

[15] E. Diebelis and J. Bicevskis, "Test points in self-testing," in *Databases and Information Systems VI*, 01 2010, pp. 309–321.

[16] J. Arbon and T. M. King, "AI for Software Testing Survey," https://jason535.typeform.com/to/IhLnOQ, 2017.

[17] Typeform, "Turn data collection into an experience," https://www.typeform.com/, 2018, accessed: 2018-12-21.

[18] Functionize, "Automation Testing with Machine Learning," http://functionize.com/, 2018, accessed: 2018-12-21.

[19] J. Arbon, C. Navrides, K. Toley, R. Bedino, V. Fan, and M. Petersen, "Abstract Intent Test (AIT) Syntax," https://goo.gl/XbwgkL, 2018, accessed: 2018-12-21.

[20] S. J. Russell, P. Norvig, J. F. Canny, J. M. Malik, and D. D. Edwards, *Artificial Intelligence: A modern approach*. Prentice hall Upper Saddle River, 2003, vol. 2, no. 9.

[21] TensorFlow, "An open source machine learning framework for everyone," "https://tensorflow.org/", 2018, accessed: 2018-12-21.

[22] Keras, "The python deep learning library," "https://keras.io/", 2018, accessed: 2018-12-21.

[23] SciKit-Learn, "Machine learning in python," "https://scikit-learn.org/stable/", 2018, accessed: 2018-12-21.

[24] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky, "The Stanford CoreNLP natural language processing toolkit," in *Proceedings of 52nd annual meeting of the association for computational linguistics*, 2014, pp. 55–60.

[25] spaCy, "Industrial-Strength Natural Language Processing," "https://spacy.io/", 2018, accessed: 2018-12-21.

[26] PyTorch, "An open source deep learning platform that provides a seamless path from research prototyping to production deployment." https://pytorch.org/, 2018, accessed: 2018-12-21.

[27] Pierre Carbonnelle, "pyDatalog," https://sites.google.com/site/pydatalog/ home, 2018, accessed: 2018-12-21.

[28] Kaggle, "Open Datasets for Any Projects," https://www.kaggle.com/, 2018, accessed: 2018-12-21.

[29] AGENT, "AI-Based Generation and Exploration in Test," https://github.com/UltimateSoftware/AGENT, 2018.

[30] Appium, "Mobile App Automation Made Awesome," http://appium.io/, 2018, accessed: 2018-12-21.

[31] J. Arbon, "Adding AI to Appium," https://medium.com/testdotai/adding-ai-to-appium-f8db38ea4fac, 2018, accessed: 2018-12-21.

[32] D. Santiago, P. J. Clarke, P. Alt, and T. M. King, "Abstract flow learning for web application test generation," in *Proceedings of 9th ACM SIGSOFT International Workshop on Automating TEST Case Design, Selection, and Evaluation*. ACM, 2018, pp. 49–55.

[33] DrQA, "PyTorch implementation of the DrQA system," https://github.com/facebookresearch/DrQA, 2018.

[34] D. Chen, A. Fisch, J. Weston, and A. Bordes, "Reading wikipedia to answer open-domain questions," in *ACL*, 2017.

[35] Data Driven Design Group, "Rico - Interaction Mining," http://interactionmining.org/rico, 2018, accessed: 2018-12-21.

[36] Lemur, "The ClueWeb12 Dataset," https://lemurproject.org/clueweb12/, 2018, accessed: 2018-12-21.

[37] Y. Makedonov, "Improve Testing of AI Systems with Grey-Box Testing Technique," https://conferences.techwell.com/archives/starcanada-2018/program/concurrent-sessions/improve-testing-ai-systems-grey-box-testing-technique-starcanada-2018.html, 2018.

[38] M. G. Kukkuru, "Testing Imperatives for AI Systems," https://www.infosys.com/insights/digital-future/Pages/testing-imperative-for-ai-systems.aspx, 2018, accessed: 2018-12-21.

[39] A. Ng, "Lecture 1 — Machine Learning," https://goo.gl/ra8pGN, 2018, accessed: 2018-12-21.

[40] L. Fridman, "MIT 6.S094: Introduction to Deep Learning and Self-Driving Cars," https://goo.gl/osH95h, 2018.

[41] T. King and J. Arbon, "AI+ML - Skills for the Testing World," https://www.slideshare.net/tariqking1/ai-and-ml-skills-for-the-testing-world-tutorial, 2018, accessed: 2018-12-21.

[42] AGENT-X, "AI-Based Generation and Exploration in Test Chrome Extension," https://github.com/UltimateSoftware/AGENT-X, 2018.

[43] Google, "Cloud Vision," https://cloud.google.com/vision/, 2018, accessed: 2018-12-21.

[44] H2O.ai, "H2O.ai is Democratizing Artificial Intelligence," https://www.h2o.ai/, 2018, accessed: 2018-12-21.

[45] DataRobot, "Automated Machine Learning for Predictive Modeling," https://www.datarobot.com/, 2018, accessed: 2018-12-21.

[46] S. Newman, *Building Microservices: Designing Fine-Grained Systems*, 1st ed. O'Reilly Media, Inc., 2015.

[47] T. Ha, S. Lee, and S. Kim, "Designing explainability of an artificial intelligence system," in *Proceedings of the Technology, Mind, and Society*, ser. TechMindSociety '18. New York, NY, USA: ACM, 2018, pp. 14:1–14:1. [Online]. Available: http://doi.acm.org/10.1145/3183654.3183683