2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT)

The Impact of Artificial Intelligence on Software Testing

Hussam Hourani
Faculty of Science and IT
Al Zaytoonah University of Jordan
Amman, Jordan
hussam.hourani@gmail.com

Ahmad Hammad
Faculty of Science and IT
Al Zaytoonah University of Jordan
Amman, Jordan
ahmad.hammad94@yahoo.com

Mohammad Lafi
Faculty of Science and IT
Al Zaytoonah University of Jordan
Amman, Jordan
lafi@zuj.edu.jo

***Abstract*—Artificial Intelligence (AI) plays an important role in our life and touch base most of our surrounding applications and systems. A huge amounts of data are created every day from many different sources that need to be monitored and analyzed properly and report results and take actions. A more complex software applications have been built, time is becoming a critical factor to release applications that must be fully tested and comply with Business Requirements. AI plays a key role in Software Testing and can get more accurate results and saves time. This paper discuss the Artificial Intelligence key pillars that can be used in Software Testing. It also open a window on how the future will look like in terms of Artificial Intelligence and the Software Testing. The results show that AI can achieve better results in Software Testing and AI-driven testing will lead the new era of the quality assurance (QA) work in the near future. AI Software Testing will reduce time to market and will increase the efficiency of the organization to produce more sophisticated software and will create smarter automated testing.**

Keywords— Artificial Intelligence, the Software Testing, Test Automation.

I. INTRODUCTION

Artificial intelligence started playing many roles in the applications around us and soon it will be an essential part of our societies and our life. The oxford definition of artificial intelligence is: "The theory and development of computer systems able to perform tasks normally requiring human intelligence, such as visual perception, speech recognition, decision-making, and translation between languages" [1]. The key pillars of AI are: machine learning, deep learning , natural language processing (NLP), expert system and others. AI covers many areas like: data analysis, prediction, decision making, intelligent systems and many others.

In recent years, machine learning, deep learning, NLP and the related algorithms and techniques have achieved a great breakthroughs in many fields and specifically in robots industry. Machines started understanding verbal commands, evaluating information, recognizing images, driving cars, analyzing data and playing games better than we do.

Due to the increasing maturity of AI's algorithms and techniques, and due to the amazing development in the technology and computer hardware that increased the computers speed and provided a huge memory, AI started acting an important role in many areas and one of them is software testing. Software testing is an imperative process that ensures business requirement fulfilment and lead to customer satisfaction and a success journey during the software development lifecycle. Machine learning, deep learning and nature language processing algorithms and techniques are the key players in software testing. In the next section we will give an overview of machine learning and software testing.

A. Mchine Learnig Overview

Machine learning (ML) is the science of getting computers to learn and act like humans do. It uses algorithms and mathematical models to progressively improve their performance on a specific task. ML has the following three main categories and sub categories that are shown in Figure 1:

- Supervised learning*:* is to use an algorithm to learn the mapping function from the input to the output. Classification and regression are examples on the subcategories of supervised learning.
- Unsupervised learning: is trying to find hidden structure in unlabeled data. Clustering and dimensionality reduction are the sub categories for unsupervised learning.
- Reinforcement learning: It allows software agents to automatically determine the ideal behavior within a specific context, in order to maximize its performance.

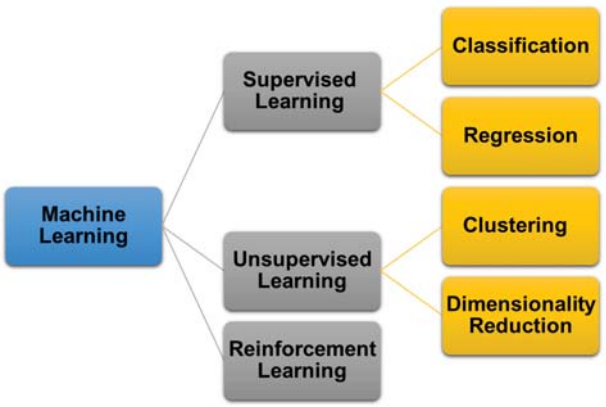

Figure 1. Machine Learning Categires

978-1-5386-7942-5/19/$31.00 ©2019 IEEE

B. *Software Testing Overview*

The definition of testing according to the ANSI/IEEE 1059 standard is the process of analyzing a software item to detect the differences between existing and required conditions (that is defects/errors/bugs) and to evaluate the features of the software item [2].

Software testing is the practice and processes to check whether the software actual results match the expected results as per the requirements and specifications and ensure that the software is defect free. The goals of the software testing are to identify errors, faults, gaps and missing functionalities as per the requirements and specifications. Software testing types are as following:

- *Manual testing:* Testing of the software manually without using any automated tool or scripts [2].
- *Automated testing:* It is also known as "Test Automation", is when the tester writes scripts and uses another software to test the software [2].

Software testing life cycle-phases take place throughout the software development life cycle (SDLC). It is generally divided into a number of distinct phases as following: requirements analysis, test planning, test development, test execution, evaluating exit criteria and test closure as shown in Figure 2 [3].

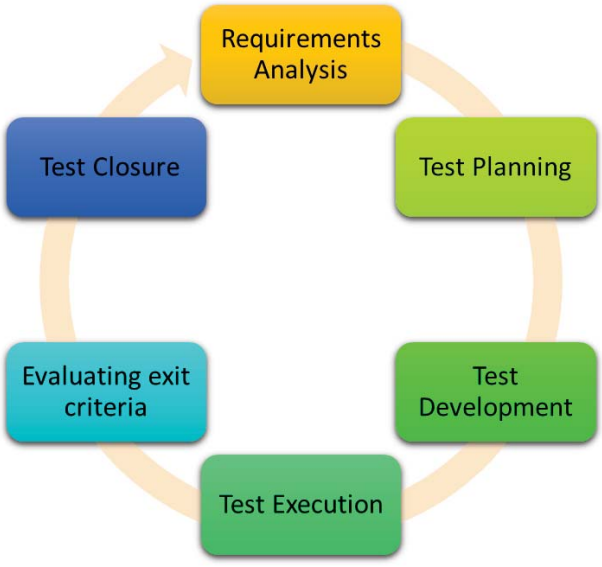

Figure 2 Software Testing Life Cycle-Phases

Testing is done throughout several levels and stages as shown in Figure 3, the following are the main levels:

- *Development Testing:*it consists of the following types:
  - *Unit Testing*: Testing basic units such as method or class and focusing on functionality.
  - *Component Testing*: Integrating software units and testing them, focusing on testing the components interface.
  - *System Testing*: Integrating components from different teams and reusable code and third party code then testing the whole system.
- *Release Testing* : it consists of the following types:
  - *Requirements Testing:* Inventing test case from each requirements.
  - *Scenario Testing*: Inventing scenario of the system and using and testing this scenario.
  - *Performance Testing*: is designed to check that the system can process its intended load.
- *User Testing* : it consists of the following types:
  - *Alpha Testing*: is done in development environment.
  - *Beta Testing*: is done in the user environment.
  - *Acceptance Testing*: is performed by customers.

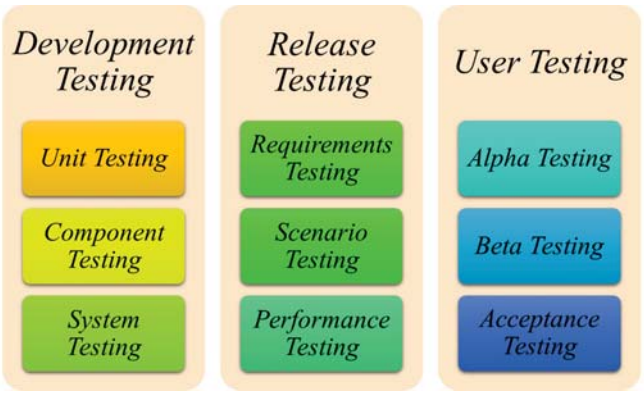

Figure 3 Testing Levels

The key issue is how quality assurance can facilitate the Software Testing and generate more test cases that are accurate and easy to execute with competitive time frame while still meeting the business requirements and the client's expectation. Artificial intelligence and its key pillars like Machine learning and NLP can play a major role in this and can facilitate the software testing in most of the areas.

Automate testing will save time and enhance the accuracy. Auto generating of the test cases and execute them automatically is becoming an important subject to the software development industry. One of the key reasons to automat testing is to ensur that your testing is successful and you get the maximum return on investment (ROI). By using AI in this areas, organizations can enhance the testing quality and generate smart and more accurate test cases for systems and enhance the testing coverage by using machine learning, deep learning and NLP algorithms and techniques.

In the next section we will provide a summary of the literature review of the artificial intelligence and software testing.

II. LITERATURE REVIEW

Machine learning has many techniques and algorithms to be used in the scope of software testing. The algorithms and techniques in AI are defer from each other in terms of: how they work, their mathematical and statistical models, assumptions, characteristics, their accuracy, their strengths and weaknesses and their solving category if they solve classification or regression or other problems. During our review on previous work in this field, we have identified many techniques and algorithms used to integrate the AI with software testing and generate acceptable and good results. This section highlights the main references in this scope as shown hereafter. One note to highlight in this section is that we did not review every single research in this area. We have chosen the most popular researches. However we believe that the selected researches and papers cover most of the testing areas across the software testing lifecycle.

One of the proposed solutions highlighted in [4] is called: MELBA (MachineE Learning based refinement of BlAck-box test specification), a partially automated iterative methodology based on the C4.5 algorithm (C4.5 is an algorithm in machine learning used to generate a decision tree). The resulting test suites were significantly more effective in terms of fault detection while only requiring a modest size increase [4]. Another ML proposed solution is highlighted in [5]. In this research, the goal was to understand which features can be used to train a model able to predict the coverage achieved by automated tools. The features for coverage prediction results show that SVR is the most accurate algorithm among the considered ones. Also to some extent, the study showed that ML algorithms are a viable option to predict the coverage in automated testing [5]. Machine learning has also used in graphical user interface (GUI). In reference [6, 7], they highlighted how to use AI to automatically test GUI. Hybrid genetic algorithms (HGA) have been used in this area. They proposed a framework that includes two important optimizations: test sequence optimization and test case optimization [6, 7]. In [8], Authors proposed a test case classification methodology based on k-means clustering to enhance regression testing. The paper found out that the clustering-based approach performs better when first the statement coverage criterion is utilized [8]. Study [9] has evaluate the machine learning techniques used in 21 fault prediction studies. They concluded that there remains much to be done to improve the quality of machine learning techniques used in software fault prediction [9]. Machine Learning also used to check test case feasibility. In [10], the research has highlighted that classifying test case feasibility is possible. The report highlighted that one advantage to grammar induction is that the induced grammars can show software testers the types of event sequences that cause infeasible test cases [10].

In [11], the study has highlighted the significant analysis in the area's subject to learn and stimulate the association between the metric specifying the object orientation & the concept of change proneness. As a result of this technique, they have proposed a reduction in the efforts that are put in the testing of software [11]. In study [12], they applied SVM to learn a ranked classification model. The results revealed considerable improvements compared to a random and manual prioritization by test experts. The technique was able to find failures earlier and allowing for more efficient Regression Testing [12].

Research [13] developed models for predicting the change proneness for object oriented system. The developed models may be used to predict the change prone classes at early phase of software development. Adaboost is showing highest accuracy with 0.877. Other algorithms like random forest and bagging show a competitive results [13]. Research [14] has covered the application of machine learning tools and variable selection tools to solve the problem of estimating the execution effort of functional tests. In environments where robust databases are available, containing ten or more test process metrics and at least 30 records, the SVR model and the Weighted MLP model are recommended to provide more accurate results, instead of the linear regression model [14]. Research [15] has proposed a framework for value-based software test data generation through genetic algorithms. The framework has the following features: prioritizing testing, electing to fulfill the most valuable testing objectives, devising a more cost-effective way to carry out the remaining testing objectives, and obtaining a graceful degradation when the testing budget is cut back. Genetic algorithms have been applied to test data generation for the following types of testing: structure-based testing, temporal testing, functional testing and safety testing [15]. In [16], the research highlighted how to identify the coincidental correct (CC) test cases, which implement the faulty statement but with a correct output, for single fault version programs. The results showed that the average recall ratio and false positive ratio were 81% and 5% respectively, and the effect of CBFL was improved with three strategies in the different program versions [16].

Research [17], highlighted that in order to detect bugs in early phases, researchers proposed various test case prioritization (TCP) techniques. NLP has been employed to assist the TCP techniques. The result shows that all of these used strategies can help to improve the efficiency of software testing, and the risk strategy achieved the best performance across the subject programs [17]. In research [18], the study proposed a test case failure prediction approach for manual testing that can be used as a non-code/specification-based heuristic for test selection, prioritization, and reduction. The results showed that a simple history-based feature combined with a linear regression model can accurately predict test cases' failure. In addition, the NLP-based approach can provide more improvement on the accuracy of predictions by the baseline approach [18]. In [19], the study proposed a system that deals with automatic generation of test cases from functional requirement using NLP. The goal of the proposed system was to reduce effort and time consumed by software tester to test the product [19]. In [20], the study has proposed an approach to generate test case from software requirements expressed in natural language using a natural language processing technique. The study recommended that the approach needs a tool to be automated, and it suggested to use database such as Hadoop for storing the graphs generated [20]. In research [21], they have presented a novel approach UnitTestScribe that combines static analysis, natural language processing, backward slicing, and code summarization techniques in order to automatically generate expressive NL descriptions concisely documenting the purpose of unit test methods [21]. In [22], the aim was to evaluate the feasibility of using natural language processing techniques to help automate detection of duplicate defect reports. The research has evaluated the identification capabilities on a large defect management system and

concluded that about 40% of the marked duplicates could be found [22].

We have summarized the findings as shown in Table 1 below. The summary highlighted the AI algorithms and techniques used in the selected papers and references along with the software testing scope area and related components.

Ref #	AI Algorithm/ Techniques used	Software Testing Area
4	C4.5 (Decision Tree Algorithms)	Refine Black-Box test specification and improve the category-partition specification
5	Huber Regression, Support Vector Regression (SVR) and multi-layer perceptron	Predicting the coverage in automated testing
6, 7	Hybrid Genetic Algorithms (HGA)	Automatically test GUI, including test sequence optimization and test case optimization
8	K-Means Clustering	Test case classification to enhance regression testing
9	General Classification Methods (SVM and others)	Software Fault Prediction
10	Support Vector Machines (SVM), Induced grammars	Identifying infeasible GUI test cases
11	Ridor, Random trees, Naïve bayes, Ordinal classifier and others	Change Proneness
12	Support Vector Machine (SVM) RANK	Test case prioritization in system-level testing without code access. for black-box testing
13	Logistic regression, Random forest, Adaboost, bagging. and others	Optimizing testing efforts based on change proneness
14	Artificial Neural Network (NN) , Support Vector Machine(SVM) and Linear Regression	Planning and scheduling of testing activities
15	Genetic Algorithms	Test data generation
16	K-Nearest Neighbor	Identify coincidental correct test cases
17	NLP	Test case prioritization
18	NLP, linear regression	Predicting manual test case failure
19	NLP	Generating test cases using
20	NLP	Generation of test cases from software requirements
21	Static Analysis, NLP, Backward Slicing and Code Summarization Techniques	Automatically documenting unit test cases
22	NLP	Detection of duplicate defect reports

Table 1. AI &ST findings summary

III. ARTIFICIAL INTELLIGENCE & SOFTWARE TESTING

As seen in the literature review, AI has played a major role in software testing. Machine learning and NLP cover many testing areas as highlighted in Table 1. Researchers have used and combined many algorithms and techniques to target specific jobs in software testing and achieved a competitive results.

Quality assurance is a journey that QA team and test engineers engaged with thoroughly. There are many challenges throughout this journey specially when there is a manual testing that the QA needs to handle throughout the testing life cycle. Manual testing requires dedicated human resources which is costly and time consuming and less reliable comparing with the smart automated testing. In addition there are many changes in both manual and traditional automated software testing including: understanding requirements, testing coverage, testing planning and time to execute, updating the test scripts and cases, regression test coverage and many others.

As the software market demand grows, organizations need to secure their challenges and be ahead of the competitors. AI is a field that can be used in the software testing to shorten the software development life cycle and reduce time to market. Figure 4 highlights the AI software testing key advantages.

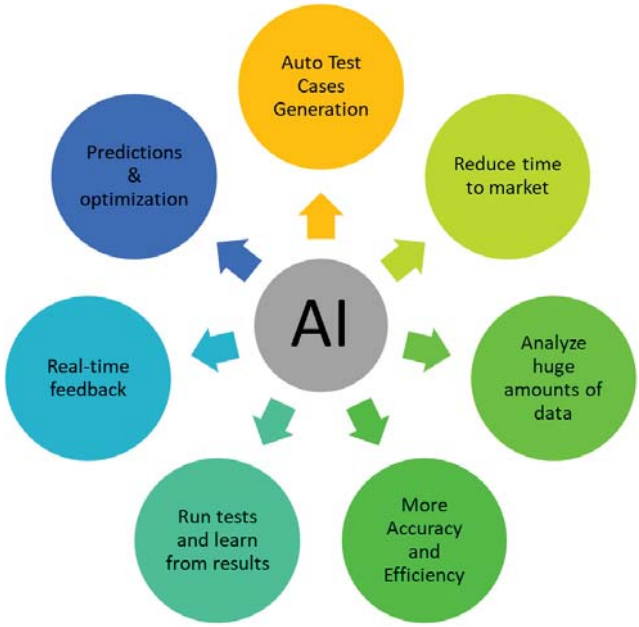

Figure 4 The AI Software Testing Key Advantages

Quality Assurance professionals and test engineers started considering the AI software testing is a key factor for testing their software. By performing quality control checks using AI models, algorithms and techniques, organizations open a new era in software testing and started producing a competitive applications that exceed expectations.

If we look into the software testing, we discover that all related software components are data. The source code is data, the screens, websites, databases, inputs and output are just data. AI can handle huge data easily and effectively by applying its algorithms and techniques and can response to data very effectively comparing to human. AI can apply methods on data for software testing purposes like classifications, regression, clustering and dimensionality reduction. AI can combines different algorithms all together to get better and promising results from analysis and predictions.

As we can see from Figure 5, AI already covers many areas in the software testing from requirements analysis

phase till test execution and closures. This is what currently in the market for the AI contribution in the software testing.

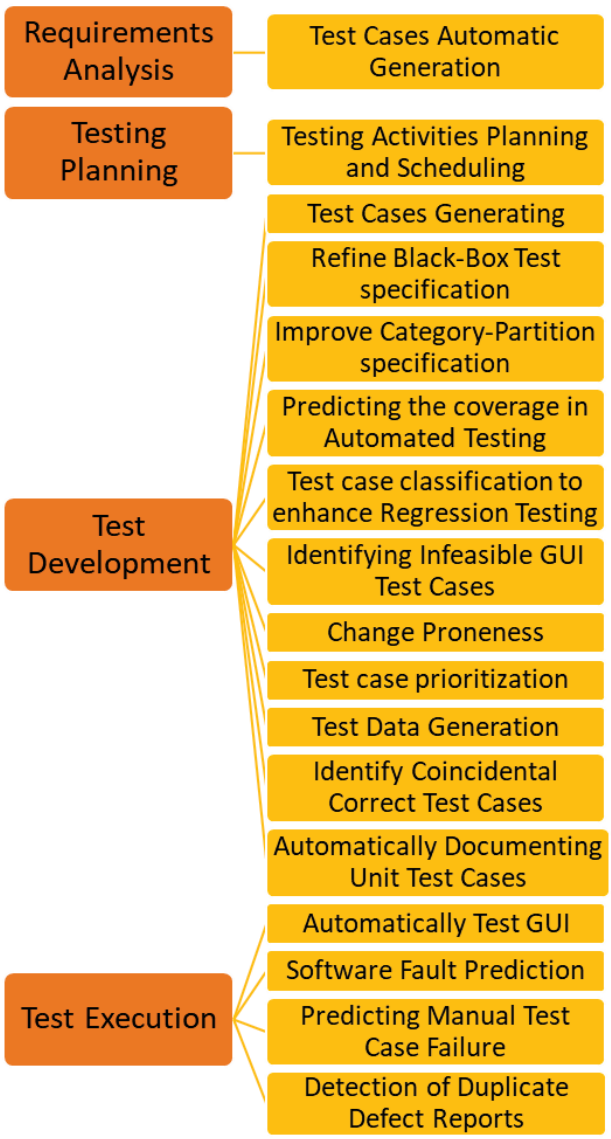

Figure 5.Current AI Software Testing Coverage.

The AI trends underway in the software testing industry are very promising and AI will drive this industry with a great results going forward. This is the future and companies already started investing in this industry. The following are the key expected contributions in the near future (4 to 8 years) for the AI in software testing area, this was based on our research analysis and prediction study:

- The AI software testing will become an independent industry and will play a major role in IT. We expect that AI software testing will replace the QA and testers engineers. QA team and testers engineers will play a new role in tuning and monitoring the AI results.
- AI will drive the software testing and will cover all testing stages from test preparations to planning, execution and reporting without human intervention and errors.
- AI software testing industry will produce more accurate results and will shorten the software development lifecycle than traditional testing techniques. When building software solutions, meeting deadlines will be challenge specially that we might not be able to keep up with the overwhelming software demand, so AI will bridge this gap and will facilitate this challenge by shorten the required testing time.
- AI will eventually have dedicated tools to effectively test the new technology like Cloud Computing, IoT, Big Data and other future technologies. Combining the new technologies will bring innovation to the AI software testing because AI will play the integrator role in generating the required testing data for a specific product.
- We expect that there will be specialized software and hardware solutions that can run AI deep learning and other AI algorithms and techniques to achieve more accurate testing results in a competitive timeframes.
- AI also will play a key role in testing the customer requirements by applying the predictive analysis to check other similar products and services, to better understand what new features the customers need.
- AI Software Testing will reduce time to market and will increase the efficiency of the organization to produce more sophisticated and complex software in a competitive timeframe. AI has the ability to analyze complex data automatically by using smart techniques and algorithms.
- AI will cover most of the software products testing in all areas including: application development, website development, database applications, mobile applications, games industry, real time critical applications, embedded solutions and others.
- The new AI software testing tools will be innovative, agile and smart. They will provide greater results to the beneficiaries and end users.
- By using AI algorithms and techniques, organizations and businesses will improve the customer experience, enhancing their products offering and increase the quality of the provided services and will bring software stability to their products.
- The AI predictive analytics will play a major role in discovering all possible test cases and will make the software products more robust, reliable and will exceed customer expectations.

Machine learning, deep learning, NLP and other AI areas are considered as a leading edge of the most of the technologies around us. As we highlighted and discussed, bringing AI to software testing will release the great power of the smart software testing automation and will move and push the software development and testing industry in a new era focusing on innovation and agility.

IV. Conclusion

Artificial intelligence has the ability to analyze complex data automatically by using smart models and algorithms. AI already showed that it can achieve better results in software testing. AI-driven testing will lead the new era of the QA work in the near future. It will manage and control most of the testing areas and will add great value to the testing outcome and will produce more accurate results in a competitive timeframe. There is no doubt that AI will influence QA and testing industry and will lead this going forward. The smart automation of software testing will improve the quality of the software and will have a major impact on the customers experience through providing a solid defect-free applications and solutions.

It is expected that AI will play a key role in software testing eventually. The new role and scope for the testers will be focusing on truing the AI models, algorithms techniques to become smarter. AI Testing algorithms will also connect to new technologies in the future (like Cloud technology, IoT, Big Data and others) and will extract the best practices techniques that suit the client application to get more accurate and smart test cases and will generate perfect results. Deep learning along with the NLP and other techniques will play a major role in the software testing and will have some specialized tools (Software and Hardware) to use in all software testing lifecycle.

V. Future Work

Future work can look into other areas in the AI and software testing. Deep learning is one promising area in the AI that can provide better results than traditional AI algorithms. This area can be investigated to see how deep learning can play a role in software testing.

Another area is to cover more studies to investigate other testing areas that hasn't been covered in this research.

VI. References

[1] https://en.oxforddictionaries.com/definition/artificial_intelligence

[2] tutorialspoint.com, "Software Testing Tutorial", 2010

[3] Isha, Sunita Sangwan ," Software Testing Techniques and Strategieg ," (Department of computer science) SBMNE College, Rohtak INDI,2014.

[4] L. C. Briand, Y. Labiche, and Z. Bawar, "Using Machine Learning to Refine Black-Box Test Specifications and Test Suites," 2008 The Eighth International Conference on Quality Software, 2008.

[5] G. Grano, T. V. Titov, S. Panichella, and H. C. Gall, "How high will it be? Using machine learning models to predict branch coverage in automated testing," 2018 IEEE Workshop on Machine Learning Techniques for Software Quality Evaluation (MaLTeSQuE), 2018.

[6] A. Rauf and M. N. Alanazi, "Using artificial intelligence to automatically test GUI," 2014 9th International Conference on Computer Science & Education, 2014.

[7] D. J. Mala and V. Mohan, "IntelligenTester –Test Sequence Optimization Framework using Multi-Agents," Journal of Computers, vol. 3, no. 6, Jan. 2008.

[8] Y. Pang, X. Xue, and A. S. Namin, "Identifying Effective Test Cases through K-Means Clustering for Enhancing Regression Testing," 2013 12th International Conference on Machine Learning and Applications, 2013.

[9] T. Hall and D. Bowes, "The State of Machine Learning Methodology in Software Fault Prediction," 2012 11th International Conference on Machine Learning and Applications, 2012

[10] R. Gove and J. Faytong, "Identifying Infeasible GUI Test Cases Using Support Vector Machines and Induced Grammars," 2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops, 2011.

[11] K. Chandra, G. Kapoor, R. Kohli, and A. Gupta, "Improving software quality using machine learning," 2016 International Conference on Innovation and Challenges in Cyber Security (ICICCS-INBUSH), 2016.

[12] R. Lachmann, S. Schulze, M. Nieke, C. Seidl, and I. Schaefer, "System-Level Test Case Prioritization Using Machine Learning," 2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA), 2016.

[13] A. K. Tripathi and K. Sharma, "Optimizing testing efforts based on change proneness through machine learning techniques," 2014 6th IEEE Power India International Conference (PIICON), 2014.

[14] D. G. E. Silva, M. Jino, and B. T. D. Abreu, "Machine Learning Methods and Asymmetric Cost Function to Estimate Execution Effort of Software Testing," 2010 Third International Conference on Software Testing, Verification and Validation, 2010.

[15] D. Zhang, "Machine Learning in Value-Based Software Test Data Generation," 2006 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI06), 2006.

[16] Z. Li, M. Li, Y. Liu, and J. Geng, "Identify Coincidental Correct Test Cases Based on Fuzzy Classification," 2016 International Conference on Software Analysis, Testing and Evolution (SATE), 2016.

[17] Y. Yang, X. Huang, X. Hao, Z. Liu, and Z. Chen, "An Industrial Study of Natural Language Processing Based Test Case Prioritization," 2017 IEEE International Conference on Software Testing, Verification and Validation (ICST), 2017.

[18] H. Hemmati and F. Sharifi, "Investigating NLP-Based Approaches for Predicting Manual Test Case Failure," 2018 IEEE 11th International Conference on Software Testing, Verification and Validation (ICST), 2018.

[19] A. Ansari, M. B. Shagufta, A. S. Fatima, and S. Tehreem, "Constructing Test cases using Natural Language Processing," 2017 Third International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB), 2017.

[20] R. P. Verma and M. R. Beg, "Generation of Test Cases from Software Requirements Using Natural Language Processing," 2013 6th International Conference on Emerging Trends in Engineering and Technology, 2013.

[21] B. Li, C. Vendome, M. Linares-Vasquez, D. Poshyvanyk, and N. A. Kraft, "Automatically Documenting Unit Test Cases," 2016 IEEE International Conference on Software Testing, Verification and Validation (ICST), 2016.

[22] P. Runeson, M. Alexandersson, and O. Nyholm, "Detection of Duplicate Defect Reports Using Natural Language Processing," 29th International Conference on Software Engineering (ICSE07), 2007.