Automating and Optimizing Software Testing using Artificial Intelligence Techniques

Minimol Anil Job
Faculty of Computer Studies, Arab Open University
Kingdom of Bahrain

***Abstract*—The final product of software development process is a software system and testing is one of the important stages in this process. The success of this process can be determined by how well it accomplishes its goal. Due to the advancement of technology, various software testing tools have been introduced in the software engineering discipline. The use of software is increasing day-by-day and complexity of software functions are challenging and there is need to release the software within the short quality evaluation period, there is a high demand in adopting automation in software testing. Emergence of automatic software testing tools and techniques helps in quality enhancement and reducing time and cost in the software development activity. Artificial Intelligence (AI) techniques are widely applied in different areas of Software engineering (SE). Application of AI techniques can help in achieving good performance in software Testing and increase the productivity of the software development firms. This paper briefly presents the state of the art in the field of software testing by applying AI techniques in software testing.**

Keywords*—Software testing; artificial intelligence; testing automation; software engineering; software quality

I. Introduction

Software engineering is the creation and application of sound engineering principles to produce cost-effective software which is both reliable and functional on real machines. A well-managed development process is needed to produce a high-quality software product. Software development is a human endeavor which involves various activities. The activities are; analysis, design, implementation and testing and each of them contributes to the creation of the final product. These activities continue in the development process, and thus producing a working version of the system can be time-consuming. Software testing is one of the principal activities in software development for verifying and validating a software system. Testing assists software developers in ensuring that the developed software fulfills its intended function, as well as determining whether or not the identified problems have been solved. Since the software development life-cycle is a complex process with a crucial need to deliver a new product within the allocated time, the software testing process should be efficient and effective.

In the software industry, automation plays a critical role in increasing testing performance. There are various automation tools available to support the testing activity. Newer technologies like Artificial Intelligence (AI) and Machine Learning (ML) are constantly being used to speed up the software development process. With the advancement of AI technologies, various business domains are accepting and using AI based software. AI systems are developed based on machine learning models and techniques. AI is used to promote automation and reduce the amount of routine activities to create testing phases by applying logic, problem solving, and ML. The purpose of this paper is to present the application of AI techniques in automation of software testing and the impacts. The paper is closed with a conclusion and a viewpoint of future work to enhance the practical aspects of the AI automation tool.

II. Literature Review

Testing is an activity takes place throughout the software development life cycle. The major aim of the software testing is find errors and to ensure that the developed system satisfies the requirements of customers. Software testing a way of evaluating a system by discovering the differences between the identified requirements at the requirement engineering activity and the archived results. There are various techniques used for testing a software to ensure quality. Testing can be done as manual or automated using specific tools [20] [7]. In manual testing, the software tester follows a test plan and complete a set of test cases manually. Manual testing is time consuming and cost effective. The alternative of the manual testing is the automated execution in which a software program runs a pre-defined test cases and shows and saves the results. Automatic testing executes the test-case automation, test-case generation and result verification. [2]. Automation testing tools are used in automatic software testing and the reliability as well as performance are more than manual testing [10] [19]. Software testing using AI techniques has been adopted by software development companies all over the world. The application of AI is wide-ranging branch of computer science that deals with building smart machines which are capable of performing smart tasks with assistance of human intelligence [9].

AI has become an emerging technology which can be applied in ensuring quality assurance (QA) in organizations [26] [15]. AI developments require an approach to validation and verification of software. AI has becoming an important factor in quality assurance and testing in future. [14]. AI is a branch of science and engineering concerned with the computational analysis of intelligent behavior and the creation of intelligent machines. The term AI refers to a collection of tools, techniques, and algorithms [3] [11] [21]. Application of AI techniques are creating impacts in various domains like health, manufacturing etc. Emergence of big data and its management requires strong computing tools and AI helps in

improving the efficiency of these activities. AI techniques are applied in different areas of software engineering such as requirement engineering, design and testing [24]. AI in software testing aims to make testing smarter and more efficient. AI and machine learning apply reasoning and problem solving to automate and improve testing. AI in software testing helps reduce time-consuming manual testing, so teams can focus on more complex tasks, like creating innovative new features. Faulty software is mission critical as software becomes an increasingly important component in a wide variety of systems. Since automated test generation techniques are free of the cognitive biases that have been found in human testers, they can be used for software testing [4].

Automating the execution process of the software testing cycle is the most common approach in the automation field. Automated software testing is important and adding more toolkits to make testing phases fully automated by generating test scripts is also equally important. Test cases can be generated to complete the automated testing [22]. The test execution speed will be increased by this tools and the testing process can be applied repeatedly. In manual testing method, developing test scripts is a time consuming process but if the test cases are ready the human tester can complete the testing process quickly [27] [25]. In manual testing, code visibility does not affect test code coverage and fault detection rate12]. It is possible to facilitate a process for eliciting testing requirements and creating test-suites [1]. Various algorithms are used in AI tools. Use of Bayesian probabilistic reasoning to model software reliability is an example probabilistic AI technique. This is used in the discipline of Software Engineering [18] [5] [21]. AI techniques can be applied in supporting automation and decreasing the amount tedious tasks in the software development and testing phase [17].

III. Software Testing

Software testing is one of the crucial stage in the software development life cycle and it is an important method to assess the developed software to ensure quality. Before releasing the final product, it is important to ensure that the requirements identified during the analysis stage have been accomplished as well as the product is defect free. In the context of software development, bug is a name given for the defect which means the developed system is not producing accurate results as per the requirements. Testing plays an essential part in the software development process. Various techniques can be used during the software development process to facilitate testing. Prototyping is one of the approaches in which an experimental software artefact will be developed and that will be discarded after evaluation. Early software artefacts are built on rather than discarded in the Iterative approach [23]. Another approach is using frameworks such as the dynamic systems development method (DSDM) in which best practice processes for iterative and incremental development are documented [13].

A. Categories of Testing

Software developers need to be aware and to be able to use a range of other testing methods, concepts and practices. The four distinct categories of testing are summarized in Fig. 1:

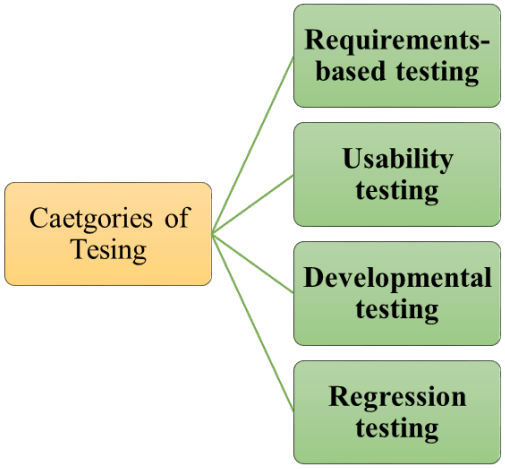

Fig. 1. Categories of Testing.

Table I presents the details of the four categories of testing shown in the above figure.

TABLE I. Types of Testing

Requirements-based testing
➢ Checks that a system meets the customer's requirements using previously gathered or formulated testable requirements. ➢ Acceptance testing is the final stage to check that the user requirements have been satisfied. ➢ The customer formally accepts the software by ensuring the correct functioning of the acceptance tests
Usability testing
➢ User interface (UI) is an integral part in every software system. ➢ Various users other than the developers are using the system and the usability testing is essential. ➢ Due to the problems of user interface, the systems may fail. ➢ Usability testing comprises systematically trying out the user interface with intended users
Developmental testing
➢ Refers to the testing carried out by the entire software development team. ➢ The developmental testing is done in three different levels which are unit testing, integration testing and system testing. ➢ Integration testing involves checking that all the tested units are interfaces together correctly.
Regression testing
➢ Any form of testing done during the development or maintenance of a system to ensure that fixing one bug does not result in the introduction of new ones. ➢ are needed at unit, integration and system levels depending on the particular development method adopted ➢ Regression tests are critical during the developmental testing and in system maintenance.

B. Software Testing Techniques

Strategies for creating test cases is an important part of validation and verification. The two major strategies are black-box testing and white-box testing. The main objective of black box testing is to ensure each aspect of the customer's requirements is handled correctly by an implementation. The test cases are designed by looking at the specification of the system. The specification includes the details of the systems intended functions.

Grey-Box Testing is another method in which the application is tested with complete knowledge of the overall aspects of the system and limited knowledge of the internal functioning of the system. [12]. The major objective of white box testing is to check that the details of an implementation of the system are correct. The test cases are designed by looking at the detail of the implementation of the system to be tested to check that the system performs its intended functions correctly [8].

While comparing the possibility of automation of black-box and while-box testing techniques, automating white-box testing is easier than black-box testing. The reason is that box in black-box testing, programmer and the tests are dependent and this will affect the automation. Since grey-box testing provides the features of both black-box and white-box testing techniques, it's features and techniques are not considered in detail in this paper.

Fig. 2 depicts the five strategies used in black-box testing and four strategies used in white-box testing.

The different black-box testing techniques are summarized in Table II [17, 22].

The different approaches of White box testing are listed in Table III [17, 22].

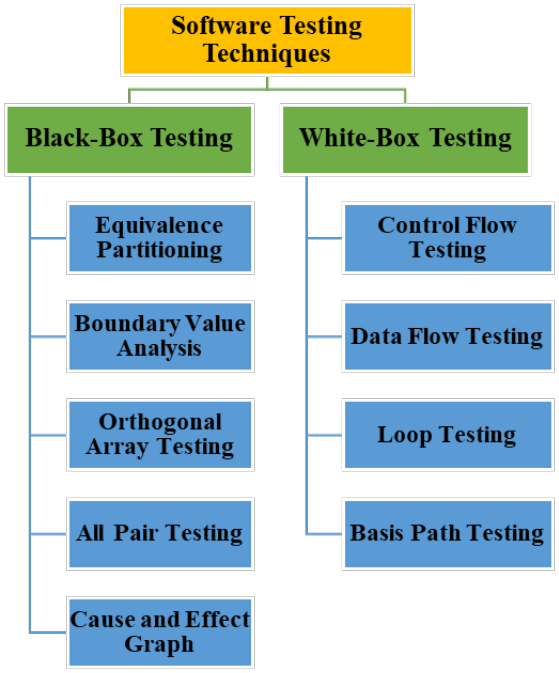

Fig. 2. Black-Box and White-Box Testing Techniques.

TABLE II. BLACK-BOX TESTING TECHNIQUES

Black-Box Testing Techniques
Equivalence partitioning
➢ This technique divides the input domain of the software unit into groups or partitions and generates test cases thus helps in reducing the number of test cases.
Boundary value analysis
➢ Errors at the boundaries of the input domain are tested in this technique. ➢ The boundary values are usually taken the minimum and maximum values in the boundaries and values just inside and outside of the boundaries.
Orthogonal Array Testing
➢ It's a strategy for problems with a limited input domain that are relatively small and cannot be tested thoroughly
All pair Testing
➢ In this method, test cases are created using all possible combinations of each pair of input parameters.
Cause and effect Graph
➢ In this approach, a graph is generated and a cause-and-effect relationship is formed. The cause is the input condition that leads to internal change in the system and the effect is the output condition.

TABLE III. WHITE-BOX TESTING TECHNIQUES

White-Box Testing Techniques
Control Flow Testing
➢ This technique uses a testing strategy in which control flow of the program is involved in the test coverage. Branch coverage, statement coverage and condition coverage are the three methods used in test coverage.
Data Flow Testing
➢ Data movement within the program is focused in this testing strategy and the test paths are determined from the positions of variable definition in the program. This technique helps to detect errors such as undefined variables etc.
Loop Testing
➢ Different types of loops such as simple loops, nested loops are used in most of the programs. Validity of the loops in the programs are tested using this technique.
Basic Path Testing
➢ Basis-path testing is based on the cyclomatic-complexity metric. The number of independent paths in a method body is computed using this metric.

IV. TEST AUTOMATION

The need to make sure that users are really receiving the value promised by the software, many powerful testing tools have emerged, which make it easy to do black box testing in an automated and repeatable manner. The automation is important because the testing process is repetitive and it is recommended to test all possible scenarios. Automating tests will increase the test coverage and improve efficiency. When the testing process is automated, the tests can be executing repeatedly, test different input values and various conditions. The testing resources and time will be reduced. To automate functional, system and acceptance tests, numerous tools are available. Some of them are Selenium, Watir and JMeter. Selenium and Watir are used to test local files using the file:// protocol supported by web browsers, however, JMeter needed the target files to be hosted on a web server.

V. Application of AI in Software Testing

Due to the increasing complexity and features, modern applications require various features in order to achieve the functional and non-functional requirements of the applications. Requirements are the information about what a system will be and do that needs to be known before development starts. Once the requirements are identified and agreed by the developer and the customer, they will be implemented in the software system. Converting the requirements into an application using appropriate tools and techniques is the role of the system developer. The application should be developed without errors and during the coding stage, the developer needs to write down and execute test cases. A software tester is a person who is in charge of putting an application through its paces and ensuring that it performs as intended. Software testers play a crucial role same as the developer in the software engineering activity as they are part of the quality assurance of the software. The software tester sends a report to the development team detailing the bugs found and the sequence of events that contributed to the mistake. As shown in Fig. 3, the Developer's responsibility of writing code and writing and executing test cases. The testers will follow the test plans and complete the testing tasks.

Software testing is an important process which ensures the developed system's satisfaction from the customers. Testing helps to protect the system against any failure which may lead potential impact in organizations' performance during the operational period. In manual testing, the software is tests are executed by the software tester to discover bugs in the system. Even though exploratory testing is possible in manual testing, it is a time-consuming process by involving human resources and test cases are executed by a human tester and software. Automated testing is considerably faster than manual testing and faster than a manual approach and uses automation tools to execute test cases. As testing is moving more and more towards automation, AI techniques are applied in software testing. AI techniques supports the automation of the testing processes in an efficient way. AI algorithms are supporting the testing environment to be more productive. The AI algorithm encourages the process and helps software testers to find the maximum number of bugs within less period of time. The system can be route to market as reliable and accurate. Fig. 4 and Fig. 5 show the functions of a tester and a developer in manual and AI enables testing environments.

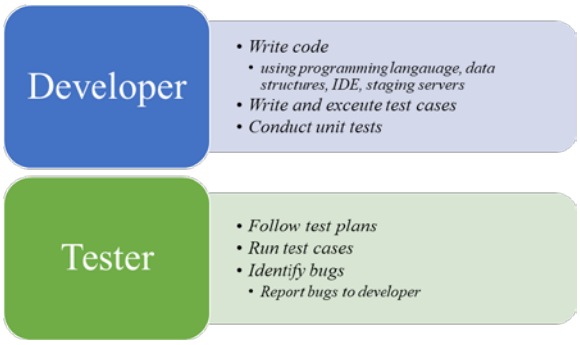

Fig. 3. Tasks: Software Developer Vs Software Tester.

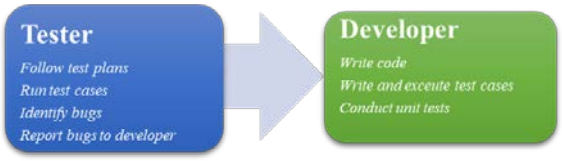

Fig. 4. Functions in a Manual Testing Environment.

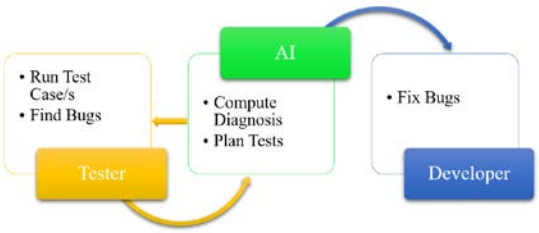

Fig. 5. Functions in an AI Enables Testing Environment.

In a manual testing environment, the tester runs the test cases according to the test plans, finds the bugs and reports them to the developer. In the AI based testing environment, the tester runs test cases and finds the bugs. The AI tool do the diagnosis and send notification to the developer to fix the bugs. The AI tool plans further tests automatically and the testers will continue running the test cases and identify bugs.

The functional and non-functional requirements of the product need to be tested by ensuring the full test coverage. In manual testing this will take time and resources and application of AI techniques will reduce the time and also will help to identify poorly designed requirements. Requirements will be thoroughly verified using AI and it will be transferred to the design stage, the next stage in the software development life-cycle. Based on the experience of the software testers, the test selection and quality will vary and the tests are created based on the requirements, use cases and user stories. Application of AI helps to automatically generate test cases and identify the tests for verification and validation of the system. Ai also help to understand the test coverage as well. Missing or inappropriate test can be identified based on the models. Ain addition to this, automatic generation of the test cases will also reduce any eluded bugs and defects also.

VI. Application of AI in Test Automation

Business organizations in various sectors are adopting AI techniques makes their operations efficient. In software development process, automation testing is widely used to improve the test efficiency. There are various ways AI techniques can support the software testing process [6] [16]. The most benefit takes from AI is the black box testing. Applications of AI techniques in software testing is listed.

Fig. 6 depicts the test automation in pyramid shape according to the test types. UI testing is placed in the highest level because and this is the most expensive testing. Test automation can be done in the lower levels both in API and Unit layers to achieve best test coverage and reduce time and cost.

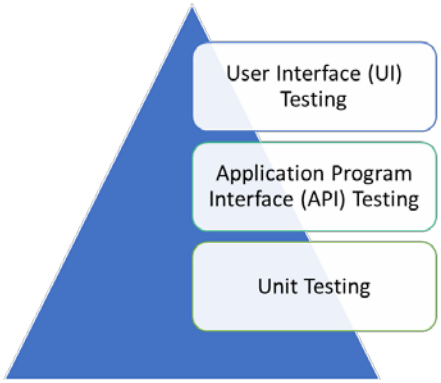

Fig. 6. Test Automation Levels.

User Interface (U) testing: Usability tests are conducted to identify any design inconsistencies or usability issues in the user interface. To build UI tests, AI techniques use image recognition techniques to navigate through the application and visually validate UI objects and components. The test automation tools decode the Document Object Model (DOM) and related code to determine object properties in AI-based UI testing. The UI tests are done after the design and development of the software system. In the pyramid, UI tests are in the top level of the pyramid and the automation is challenging and they are the hardest to automate.

Application Program Interface (API) Testing: In API testing, validation of Application Programming Interfaces (APIs) is done by testing various software quality factors of the programming interfaces such as functionality, performance, reliability and security. Nonfunctional requirements like look-and-feel are not relevant in API testing and focuses on the business logic instead. Without any reference to the user interface, API automation can be used to validate the business logic. Automation of the API allows the software development team to start the testing activity early which helps to identify and fix errors and reducing the efforts in the AI testing.

Creating and updating unit tests: Unit tests are used to ensure that small pieces of code, such as functions or object methods, are correctly executed before they are combined with other units to form a larger functional feature. Developers spend a significant amount of time writing and maintaining unit tests, which is much less enjoyable than writing application code AI-based products for automated unit test development can be beneficial and they can be generated promptly. Automation strategy starts with unit tests. Unit tests are written by the developers using the same application development programming language. Early detection and correction of errors in the unit testing will prevents regression defects.

In addition to the above mentioned testing automation strategies AI can be applied in the following areas of the software testing also.

- AI is used as an adaptive method for detecting element-level changes that will improve the testing suites' robustness.
- aid in the prediction of incorrect test cases that result in total test failures, as well as include recommendations for resolving those issues.
- used in simulating Behavioral patterns.
    - AI techniques helps to simulate the behaviors of people in using the system by geography, devices and demographics as inputs to build test suites.
- AI algorithms are efficient in prediction and automation of test suites.
- used in script Automation.
    - Automating a test script is not required in AI because it is executed from the AI algorithm automatically.
- Helping in mining defects and based on this test suite can be defined. This speeds up the process of making informed decisions about test coverage and test suite optimization.
- Enhances number of tests and their scopes.
- AI techniques are applied in Automation Test Maintenance, Test Data Generation, early feedback in testing etc.

VII. AI Based Test Automation Tools

A. AI Bases Automation Tools

Table IV summarizes 5 AI based automation tools uses in software industry recently.

TABLE IV. AI BASED AUTOMATION TESTING TOOLS

Testing Tool	Application
Testim	This AI tool is used to automate functional testing by using artificial intelligence and machine learning. This automated testing tool speeds up the authoring, execution, and maintenance of automated tests.
Functionize	A cloud based automated testing technology used for functional, performance and load testing. This automated testing tool speeds up the test creation, diagnosis, and maintenance.
Appvance	This AI enabled tool used for automating functional, performance and security testing. The AI integrated tool helps codeless test creation.
Applitools	This AI enabled tool used for User Interface (UI) testing. It provides an end-to-end software testing platform powered by Visual AI and used in Quality Assurance and test automation.
Testcraft	This AI-powered test automation platform supports regression testing and also for monitoring of web applications. The AI technology helps in eliminating maintenance time and cost.
Watir	This is an open Source tool, uses Ruby Libraries to automate tests. Watir is used for testing websites and uses Selenium for browser automation. In addition to this Watir also supports in writing stable and maintainable test scripts.

Functional tests are part of integration tests and designed to evaluate specific tasks. For example, user registration in an online system, the registration function should validate the data entry from the user. Other examples are catalogue search and online payment in an e-commerce application etc. A wide range of test values are required to perform these tests. Performance testing is used for measuring the ability of servers to respond to user demand in a networking environment. A load test is a method of evaluating a server's performance by applying a given load to the application in order to evaluate the response time for individual functions. The aim of security testing is to ensure that the application has authentication and authorization controls in place and is not vulnerable to attacks. User interfaces are usually tested for Navigation errors, presentation errors and control usage problems. Regression testing can reveal defects that recur as a result of software changes that were not planned.

B. *Pactical Application: Using Watir*

Watir (Web Application Testing in Ruby) has been selected as a software experimenting too to support the technical aspects of the research. l. The Watir supports almost all of the web browsers. The websites and user interfaces can be tested using Watir. The test framework by using Watir follows the following order as shown in Fig. 7.

After installing the required drivers, test scripts for opening browsers have been written in the RubyMine IDE. The test scripts are written in Ruby language. Watir is selected for the practical application because first of all, it is very easy to use open source tool. This tool is developed using Ruby and any web application can be automated irrespective of the browser it is running. There are in-built libraries in Watir to support various activities such as page performance. Fig. 8 depicts the test script for opening Chrome browser.

Fig. 9 demonstrates another sample of creation and execution of test case using Watir. A simple test has been used to test the accuracy of the code. The test script runs successfully.

Fig. 10 shows an example of a test script for the user interface testing. The sample HTML and JavaScript code is shown in Fig. 11 for the page.

The presentation of the web page is done using HTML and a JavaScript function is used for the static feature of the page code using HTML and Fig. 11 shows the code.

Fig. 12 shows the debugging screen of another sample test script with another browser, Firefox using Watir automation tool.

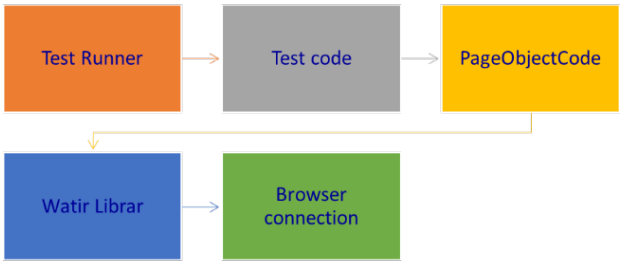

Fig. 7. The Order of Test Framework in Watir.

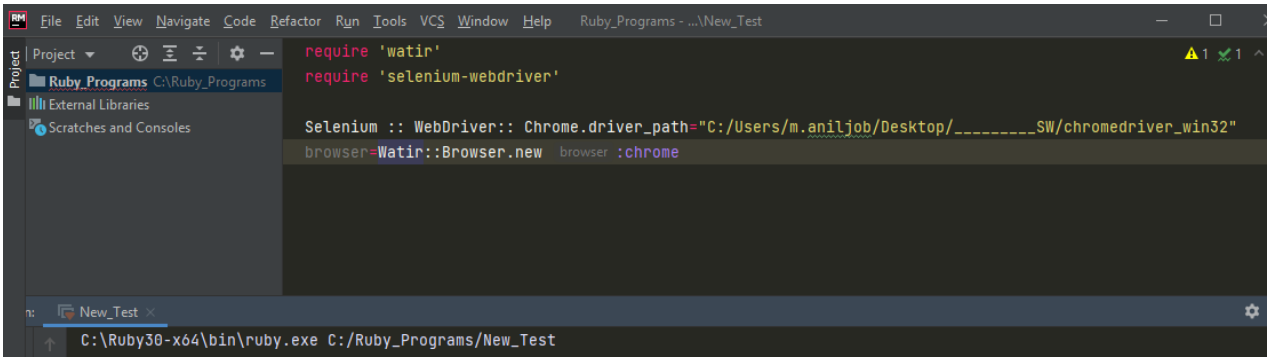

Fig. 8. Test Script for Chrome Browser.

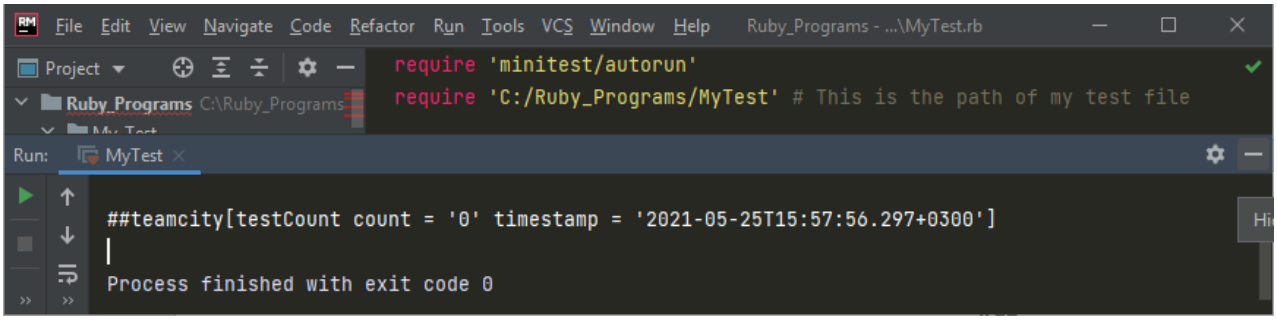

Fig. 9. Sample Test Script.

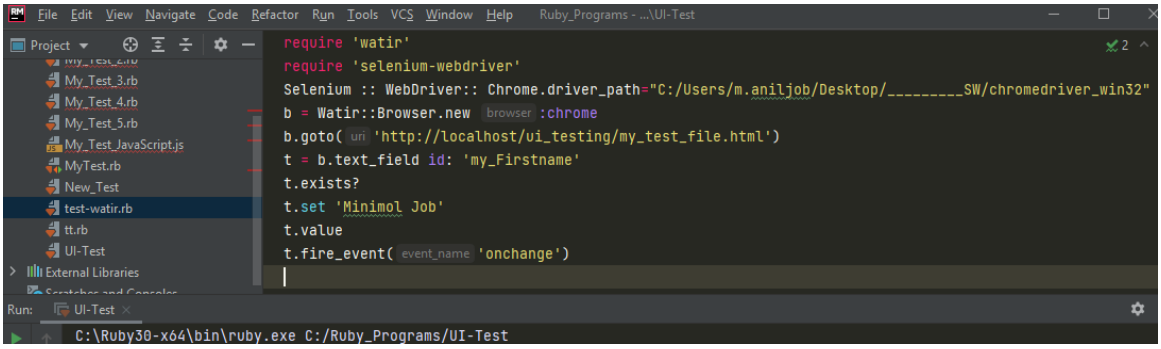

Fig. 10. Test Script for user Interface Testing.

```
<html>
   <head>
      <title>A sample test of UI using Watir</title>
   </head>

   <body>
      <script type = "text/JavaScript">
         function value_test() {
            console.log("value_test");
            var my_Firstname = document.getElementById("my_Firstname");
            if (my_Firstname.value!= "") {
               document.getElementById("my_Firstname").innerHTML =
               "My First Name is  : " + my_Firstname.value;
                document.getElementById("disp_my_Firstname").
                style. display = "";
                }
            }
      </script>
<p>
<div id = "div_my_Firstname">
   Please Enter Your First Name :
   <input type = "text" id = "my_Firstname" name = "my_Firstname"
    onchange = " value_test ()" />
 </div></p>
<p>
<div style = "display:none;" id = "display_my_Firstname">
</div></p>
</body>
</html>
```

Fig. 11. HTML and JavaScript Code for the Sample UI.

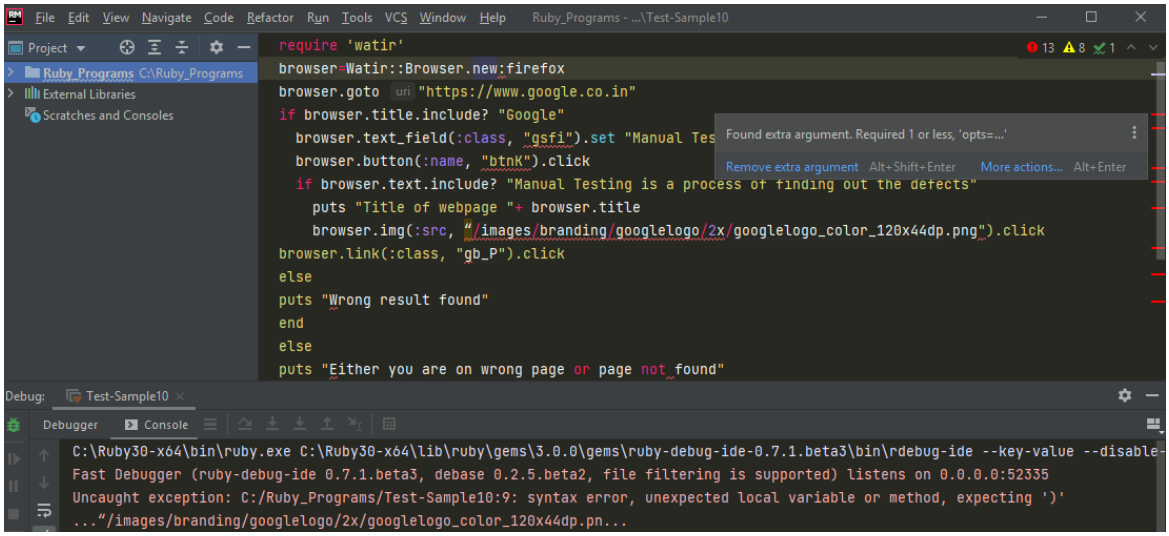

Fig. 12. Debugging Screen.

After running the test cases using it can be concluded that Watir allows easy to test file download for the user interface of a web application. Popup test alerts are provided at times as well. Another feature in Watir is, Page Performance can be measured using the performance object. Navigation, timing and memory performance can be measured by getting these details when the application is connected to the browser. Page object feature available in Watir helps reusability of the code basically in the form of classes. This feature of Watir helps the automation of the application without code redundancy. Without opening the browser, the details are obtained in the command line and thus supports the execution of the user interface test cases at the command line prompt.

VIII. Advantages of using AI in Software Testing

Fig. 13 lists the advantages of using AI tools in automating software testing. When the testing of a software system is done manually by even by highly skilled and experienced software testers there is a possibility of making mistakes and experience tiredness. Artificial Intelligent tools are constantly performing tasks effectively even repetitive tasks. Automated testing tools cab ne used by both software developers and testers. If the modifications ad editing of the program source code are cheeked thoroughly software tests will run automatically. For any unsuccessful tests, the developers will be notified and thus saving the time of the developers. AI enables automatic testing can help in improving the overall test coverage and ensure quality of the developed software. The performance of the system can be tested against the expected requirements in an efficient manner. AI enables testing can perform image and pattern recognition. This feature will help in detecting visual errors and make sure all visual elements are working properly. Software testing is a repeating process and source code may change time to time, manual testing will be a time consuming task. AI tools can perform the testing and detect errors with lesser time and AI tools are not expected to make errors and will help to generate more accurate results. Test automation can be done efficiently AI enables testing tools.

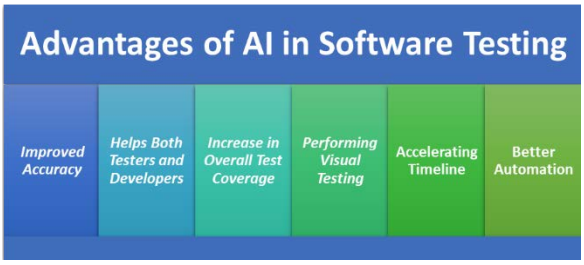

Fig. 13. AI Enabled Testing: Advantages.

IX. Conclusion and Future Work

Producing a quality software to the clients within the specified time by incorporating all the requirements is a crucial task. By using annual testing approach this will not be easy. There are various automation tools available to perform software testing. Artificial Intelligence techniques have significant impact in various stages of software development activity including software testing. Application of AI in test automation is an appropriate solution for software testing activity to produce a defect free software application. This paper presented the role of artificial Intelligence tools in software testing.

The paper also looked into various types of testing techniques for validation and verification of the software system. Selection of an appropriate AI based test automation tools is important based on the type of testing and this paper discusses the features of popular AI enables testing tools. Finally, the paper is presented the advantages of AI testing tools applications in software testing. An extension of this research paper will be presented by evaluating AI based test automation tools by taking into consideration of more technical details. Enhanced practical aspect will be covered by real implementation of the test automation of white-box testing using a mobile application.

References

[1] Agrafiotis, I., Creese, S., Goldsmith, M.: Developing a Strategy for Automated Privacy Testing Suites. In: Camenisch J., Crispo B., Fischer-Hübner S., Leenes R., Russello G. (eds.) Privacy and Identity Management for Life. Privacy and Identity 2011. IFIP Advances in Information and Communication Technology, vol. 375, pp. 32–44. Springer, Berlin, Heidelberg (2012). doi:10.1007/978-3-642-31668-5_3 2.

[2] Andreas Leitner, Ilinca Ciupa, Bertrand Meyer, Mark Howard "Reconciling Manual and Automated Testing: the AutoTest Experience". ETH Zurich CH-8092 Zürich, AXA Rosenberg Investment Management LLC Orinda, California 94563.

[3] A. M. Turing. "Computing machinery and intelligence". In: Parsing the Turing Test. Springer, 2009, pp. 23–65. [2] J. McCarthy. "Artificial intelligence, logic and formalizing common sense". In: Philosophical logic and artificial intelligence. Springer, 1989, pp. 161–190.

[4] Berndt, D.J., Fisher, J., Johnson, L., Pinglikar, J., and Watkins, A., "Breeding Software Test Cases with Genetic Algorithms," In Proceedings of the Thirty-Sixth Hawaii International Conference on System Sciences (HICSS-36), Hawaii, January 2003.

[5] B. Littlewood and J. L. Verrall, "A Bayesian reliability growth model for computer software," Applied Statistics, vol. 22, no. 3, pp. 332–346, 1973.

[6] E. Horvitz, J. Breese, D. Heckerman, D. Hovel, and K. Rommelse, "The Lumiere project: Bayesian user modeling for inferring the goals and needs of software users," in Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence. San Mateo: Morgan Kaufmann, Jul. 1998, pp. 256–265.

[7] Er. Rajender Bathla, Er. Shallu Bathla. "Innovative Approaches of Automated tools in Software testing & current technology as compared to Manual testing ". Global Journal of Enterprise Information System, Vol-1(1) , 2009.

[8] Glenford J. Myers, Corey Sandler, Tom Badgett , The Art of Software Testing, 3rd Edition, 2015.

[9] Harman, M. (2012, June). The role of artificial intelligence in software engineering. In 2012 First International Workshop on Realizing AI Synergies in Software Engineering (RAISE) (pp. 1- 6). IEEE.

[10] Jagdish Singh, Monika Sharma." A Comprehensive Review of Web-based Automation Testing Tools". International Journal of Innovative Research in Computer and Communication Engineering, Vol. 3(10), October 2015.

[11] J. McCarthy. Artificial intelligence: a paper symposium: Professor Sir James Lighthill, FRS. Artificial Intelligence: A General Survey. In: Science Research Council, 1973. 1974.

[12] Khan, M. E., & Khan, F. (2012). A comparative study of white box, black box and grey box testing techniques. Int. J. Adv. Comput. Sci. Appl, 3(6).

[13] Lee Copeland, A Practitioner's Guide to Software Test Design, 2003.

[14] M. Buenen and A. Walgude, "World quality report 2018–19," Paris, France, Tech. Rep., 2018.

[15] Mariani, L., Hao, D., Subramanyan, R., Zhu, H.: The central role of test automation in software quality assurance. Software Quality Journal 25(3), 797–802 (2017). doi:10.1007/s11219-017-9383-5.

[16] Meziane, F. and Vadera, S., (2010). Artificial Intelligence in Software Engineering Current Developments and Future Prospects, In "Artificial Intelligence Applications for Improved Software Engineering Development: New Prospects", IGI Global.

[17] Narayan, Vaibhav, The Role of AI in Software Engineering and Testing (June 22, 2018). International Journal of Technical Research and Applications, 2018, Available at SSRN: https://ssrn.com/abstract= 3633525.

[18] N. E. Fenton, M. Neil, W. Marsh, P. Hearty, L. Radlinski, and P. Krause, "On the effectiveness of early life cycle defect prediction with Bayesian Nets," Empirical Software Engineering, vol. 13, no. 5, pp. 499–537, 2008.

[19] Prof. (Dr.) V. N. MAURYA , Er. RAJENDER KUMAR . "Analytical Study on Manual vs. Automated Testing Using with Simplistic Cost Model ". International Journal of Electronics and Electrical Engineering, vol .2 (1), 2012.

[20] R. M. Sharma." Quantitative Analysis of Automation and Manual Testing." International Journal of Engineering and Innovative Technology (IJEIT), Vol. 4(1), 2014.

[21] S. Russell et al. Artificial Intelligence: A Modern Approach. 3rd ed. Prentice Hall, 2010.

[22] S. H. Trivedi, "Software Testing Techniques," International Journal of Advanced Research in Computer Science and Software Engineering, vol. 2, no. 10, pp. 433439, 2012.

[23] Stuart Feldman, Quality Assurance: Much More than Testing, ACM Digital Library, Queue – Quality Assurance, February 2005, Vol 3, Issue 1.

[24] Tadapaneni, N. R. (2017). Different Types of Cloud Service Models. Available at SSRN 3614630.

[25] T. Kosa, M. Mernikb and T. Kosarb, "Test automation of a measurement system using a domain-specific modelling language," Journal of Systems and Software, vol. 111, p. 74–88, 2016.

[26] Van De Ven, T. et al., 2018. World Quality Report 2018-19 - Artificial Intelligence - World Quality Report 2018-19 Findings:. [Online] Available at: https://www.sogeti.com/globalassets/global/wqr-201819/wqr-201819_secured.pdf.

[27] V. Sangave and V. Nandedkar, "A Review on Automating Test Automation," International Journal of Advance Research in Computer Science and Management Studies, vol. 2, no. 12, 2014.