Multi-Agent based Functional Testing in the Distributed Environment

Muhammad Fraz Malik
Shaheed Zulfikar Ali Bhutto Institute of Science and Technology, Islamabad, Pakistan

Uzma Bibi
Shaheed Zulfikar Ali Bhutto Institute of Science and Technology, Islamabad, Pakistan

M. N. A. Khan
Shaheed Zulfikar Ali Bhutto Institute of Science and Technology, Islamabad, Pakistan

Muhammad Ayaz Malik
Chalmers University of Technology, Gothenburg, Sweden

***Abstract*—Verification and testing are two formal techniques of defect reduction applied on designing and development phases of SDLC to rationalize quality assurance activities. The process of testing applications in the distributed environment becomes too complex. This study discusses a distributed testing framework that consists of many parallel tester components. The idea is based on utilizing client server environment to conduct software testing efficiently and in a short span of time. It is pertinent to mention that this study is restricted to testing of functional aspects of the software while testing of performance and other quality-of-service aspects are outside the scope of the study. An important factor influencing the use of agent technology in software testing is the dynamic nature of events. Since agents are characterized by intelligence and autonomy, their ability to interact with the environment offers added functionality to make decisions based on the needs of the scenarios that are dynamic in nature. This study shows that the use of agents to build a dynamic model for software testing in the distributed environment results in a more robust and efficient design. The proposed framework is based on distribution of test cases among multiple agents deployed across a distributed system which collaborate with each other to perform testing in an efficient manner. The proposed framework also provides an in-depth visibility into the software quality by providing the defect statistics on-the-fly. The experiments have been conducted using Selenium test automation tool. The test cases along with their test scripts and the test run results are described herein.**

Keywords*—Software quality assurance; software testing; distributed environment; input variation testing; test vectors; multi-agents

I. INTRODUCTION

Software testing is a process which is used to ensure quality of the product by assessing software behavior according to the specifications. The abnormal software behavior is generally termed as a bug or defect which could be a fault, error or failure of software that causes it to produce unexpected results or exhibit unwanted behavior. It is important to mention that faults lead to errors and errors lead to failures. The term failure is generally used from user's perspective which means that certain functionality is either missing or is not producing the desired results. The software quality is mostly viewed from customers' perspective mainly the customer satisfaction and is generally termed as fitness for purpose. IEEE and ISO define software quality as meeting the "user needs or expectations" and ability to "satisfy specified or implied needs" respectively.

Testing is a critical phase in designing and development of software and computer systems. In case of distributed software applications, the testing process particularly becomes too complex as the distributed applications inherently are multifaceted and more convoluted than the applications developed in collocated environment. Software testing of distributed systems even becomes a daunting task if manual testing is employed. This paper addresses the issue of automated testing of distributed applications by looking into the common challenges in the distributed systems testing followed by proposing a framework that automates the testing process.

Software testing is one of the most difficult tasks to assure the quality of software and plays a vital role in the SDLC process to ensure that it adheres to the client or customer needs. When it comes to the distributed applications environment, software testing is considered as backbone for applications. Testing the distributed software application is much more difficult as compared to testing standalone software applications, because the distributed system behaviors are dynamically changed with respect to time and platforms. There are several key challenges linked to testing the distributed applications; e.g., the same test run executed frequently on the same scenario with same input, may generate different outputs. This happens due to the non-linear behavior of the distributed systems i.e., event timing can also affect the end results. The functional and non-functional requirements play key role in web applications testing.

Generally, the term Software under Test (SUT) refers to the application that needs to be tested to determine correctness of its operations/functionality. For this purpose, a correctness-centered approach needs to be employed to ensure that software meets the software quality assurance requisites. For this purpose, the "design for testability" rules are applied on the specifications to prepare a suitable test run. Test harness is an umbrella term used to refer to collection of software and input data variation datasets to be used for software testing. Test harness is usually applicable at the level of unit testing

and is based on the concept of executing software under varying conditions of input followed by observing the correctness of the produced output. Test harness consists of two key components known as test script repository and test execution engine. Multi-agent is a predestined technology which synergizes the power of autonomous computational components that have control over their behavior and mutually work to achieve their specific individual objectives.

Within this context, multi-agent systems impart more functionality and intelligence to computer systems as agents operate in a flexible and rationale manner through interaction with other agents or even humans. With agents having the capacity to operate in parallel, multi-agent systems not only speed up efficiency but also enhance reliability and robustness. Also, since multi-agent systems follow a more modular structure with each agent performing independently of the other, it addresses the problem of scalability as programmers can simply add new agents to enhance functionality of a program.

Current research shows that multi-agent systems are less costly than most of the centralized systems as they are composed of subsystems of low unit cost. These agents can be reused in different scenarios without the hassle of coding new programs from the scratch for newly emerging scenarios. Agents act autonomously in order to solve complex problems in real time that are beyond the scope of human capability. Multi-agent systems make use of the expertise of individual agents coupled with their ability to collaborate, cooperate and also interact with one another to formulate powerful systems beyond the scope of individual agents alone. It is this very feature of agent technology that forms the basis of multi-agent systems [14]-[16].

II. Related Work

A software agent is a computer code or program that operates in the dynamic environment on behalf of an additional entity either human or computational. Software agents are designed to be autonomous, proactive, collaborative and operate asynchronously as well as in parallel fashion. These agents communicate through the message passing instead of method invocation. Since agents run autonomously and cooperatively among the other agents, so they may run properly by themselves and may perform inaccurately while working together. For the specific nature of the software agents, it is tricky to apply the normal software testing and debugging technique to software agents. Software agents require special techniques dedicated for testing.

Collins et al. [1] combine the advantages of Distributed Software Development and agile software development methods and state that the old works did not cover the scenario where the tasks related to testing in distributed environments with individual work groups that are located on different spatial locations. DiLucca et al. [2] performed analysis of different testing method for web applications with respect to functional and non-functional requirements. The research highlights that functionality testing of a web application relies on the following basic aspects: testing models, testing levels, testing strategies, test cases and test processes.

Clune et al. [3] state that systematic testing is crucial for the complex scientific software systems. The objective of this study was to analyze testing techniques for scientific software so that they can be maintained and evolve in a systematic way. The complex scientific software evolves due to the growing requirements and the developers introduce new line of codes in the software to meet the additional changes. Every new line in the system carries risk of introducing bugs and may result in performance degradation. Identifying and fixing such bugs require additional cost, time and effort. Early detection of bugs could considerably reduce the efforts needed to implement a correction. The authors suggest that scientific software should be covered with systematic testing.

Chu et al. [4] state that it might become more informative to perform the testing inside live situations. The vivo testing focuses on easing the burden by simply sharing load across a number of multiple instances of the software application. This approach elevates the scope in vivo testing from a single instance to a couple of instances. A central server coordinates this effort by monitor the size of the community and collecting the test results. This approach extends to presented in vivo testing framework which is called Invite. Applying this distributed method to in vivo testing technique help amortizing the workload above many instances cause higher performance impact lacking of sacrificing the quantity of tests being conduct. In addition, in vivo testing supports testing as many permutation of states as possible, in the hope that it would encounter the ones that are not correctly handled by the code. Testing software applications that use nontrivial databases are increasingly being outsourced to test centers for reducing cost and achieving higher quality [5]. However, for security reasons the sensitive information is not shared with the test centers to perform testing with the real data. The author introduced a novel approach called PISTIS for minimizing database for software testing tasks. PISTIS used on a weight-based information clustering formula that divide the test data by making using of pertinent information obtained by way of program evaluation. This way a large database is reduced to a few centric objects. This main benefits of this testing is that tests are not dependent able the designer and the tester.

Agent oriented software engineering methodologies provide us a platform to develop agents based systems. These methodologies mainly focus on development rather than the testing. It is not possible to map all agent properties e.g. autonomy, reactivity etc. to object oriented constructs. Therefore, a proper testing technique for agent-based software solutions is needed. Sivakumar et al. [6] propose an effective and specialized testing technique for agent-based systems. The proposed technique focuses the main attribute of an agent which is role. It follows a v-based model which starts from requirements and ends at role-based acceptance testing. The proposed approach provides better solution for industrial, commercial, medical, networking and educational applications related problems. The purpose of software product lines is to create efficient products in a systematic manner, and Uzuncaova et al. [7] build on one such systematic technique referred to as “scope-bounded testing” in order to develop a novel specification based methodology far efficiently creating tests regarding products within a software product line.

Lv et al. [8] propose hybrid approach that uses Adaptive Testing and Random Partition Testing is an alternating manner. The motivation for this approach is that both strategies are employed such that the underlying computational complexity of Adaptive Testing is reduced by introducing Random Partition Testing into the testing process without affecting the defect detection effectiveness. A case study with seven real-life subject programs is presented in the study. The Adaptive Random Testing is to enhance the failure-detection ability of Random Testing. Chen et al. [9] consider and compare towards the performance regarding adaptive randomly testing from code coverage perspective.

Eassa et al. [10] introduce a dynamic testing tool that use a temporal logic assertion language for detecting run time errors in agents and agent-based systems. The proposed technique is based on the syntax and semantic of the temporal logic assertion language. A dynamic testing tool has been built and tested for ascertaining its effectiveness against its use as a dynamic testing tool.

Serrano et al. [11] propose a framework to record and order interactions among agents in a MAS. To capture the interactions in the distributed MAS, the proposed solution uses a generic registration layer based on aspect oriented programming. In MAS, the distributed events are ordered using vector clocks which are combined with graph theory to produce abstract graphs. The framework is based on debugging errors in different testing environments and removes the matching errors.

Hao et al. [12] introduce the technique of performance assessment and offered a platform of agent-based functionality testing about web based services. The study includes some specific features Test Flow Generator, Scenario Creator, Test Manager and Load Generator Agent. Communication and the coordination between distributed testing components are more complex features. The typical reactions of such systems are the generation of errors such as time outs, locks, observability, controllability and synchronization problem. Azzouzi et al. [13] show how to cope with these problems by using a distributed testing method including timing constraints. Afterwards, a multi-agent architecture is proposed to describe behavior of testing a distributed chat group application on high level of abstraction. The study focuses on the temporal properties that specify the time required for exchanging messages between the various components of the distributed test applications.

Based on the literature review, we observed that there is a need for speedy execution of the testing activities to curtail the testing costs and save time. In this regards, testing of application using the distributed environment can be a viable and speedy solution. Multi-agent based framework is proposed to address the issue of performing software testing in a robust manner. In view of this, a suitable proposition could be to formulate a network of multi-agents that possess learning capabilities and are intelligent as well as collaborative. The cooperative nature of the multi-agents in this study is expected to enhance the multi-agent based software testing frameworks.

III. Functional Testing Framework

In view of the problem statement mentioned in the last paragraph of the previous section, we propose a multi-agent based functional testing framework within the distributed environment. The proposed framework only encompasses the functional testing and does not cater for non-functional or quality attribute testing. The functional testing is primarily based on input-output relationship. While testing a software, the obtained results are matched against the expected/desired results and based on this comparison the decision whether the test has passed or failed is taken. Our proposed framework is shown in Fig. 1. The detail description of the various components/artifacts that constitute our framework is described below.

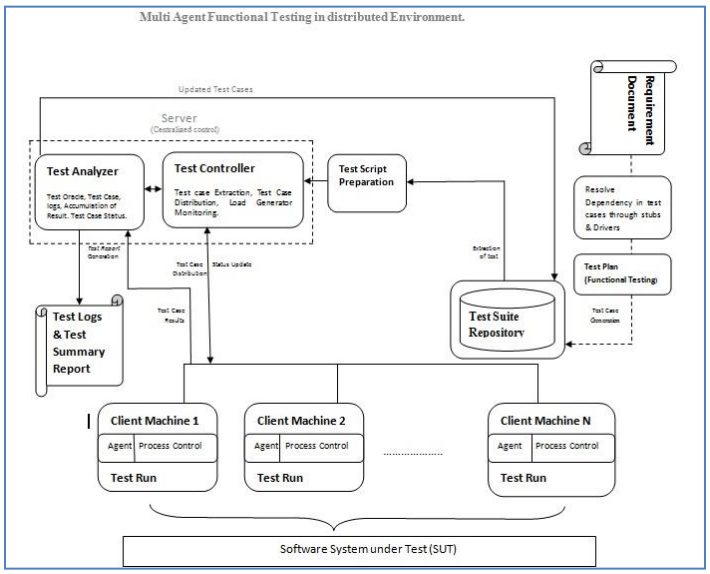

Fig. 1. Framework for multi-agent based functional testing in distributed environment.

A. Centralized Client-Server Environment

Network server is a key component to support the distributed environment (i.e. Client Server Environment) by providing a centralized control. The purpose of a server is to share data or hardware and software resources among the clients. This architecture is called the client–server model. In our framework, we imply that the client server environment is already in place and we do not deal with any of its hardware configuration or network protocol improvement. We are using distributed environment as a test bed. Our proposed framework supports parallel testing. It is used for speedily performing the testing activities by different distributed bunches of test cases across the multiple client machines. Parallel testing means testing multiple applications or subcomponents of an application concurrently to reduce the time required to test the entire system. Parallel tests consist of two or more parts (projects or project suites) that check different parts or functional characteristics of an application.

B. Test Suite Repository

Test Suite Repository maintains all the test cases corresponding to different software functionalities. It is assumed that test cases have already been prepared/generated by the test engineers either automatically or manually from the software specifications. In this study, we assume that the necessary test cases to be run for an application have already been made available in the test repository. And the dependency among different test cases has already been resolved using appropriate measures. The test plan used in this study merely corresponds to the functional testing.

C. Test Oracle

In software testing paradigm, the Test Oracle that determines whether a test has passed or failed based on certain criteria bears three capabilities: a generator, a comparator and an evaluator. The generator furnishes the expected resulted for a test case which is examined by the comparator against the obtained result. Finally, the evaluator then determines whether the comparison was successful or not. The key limitation of the oracles is that they can be applied only on a small subset of all the possible inputs and outputs pairing. Thus, it makes them suitable for small scale testing activities.

D. Test Script Generator

The task of test script generator in our framework is to prepare an executable test scripts (e.g. in HTML Format) for each and every test case. We can also assign/add priority with the test cases in order to decide upon their order of execution. A test script in software testing is a set of instructions that will be performed on the system under test (SUT) to verify that the system functions as expected. The test scripts can be either generated automatically or prepared manually. There are several testing tools such as Selenium and QTP (Quality Testing Professional) that can generate test scripts.

E. Multi-Agents

Software agents are programs or code snippets that are placed across the network and have their own control and goals. There are several types of agents and among them Interface agent, Information agents, Heterogeneous agent, Mobile agent, Reactive agents, Collaborative agents are commonly used. Agents are generally categorized based on their properties. Agent should be message passing, collaborative, proactive and autonomous. In this study, agents are supposed to receive, execute and send results of the test cases. Intelligent agents deployed on the client machines will decide to load the relevant software component/artifact based on the received test case and the procedure or plan to execute them.

F. Test Controller

Test controller is an important module of our framework. The main task of test controller is to extract the test cases from the test suite repository and prepare the test scripts accordingly followed by distributing test cases on different client machines. Test controller continuously receives the status of test cases from different client machines and updates their status accordingly. Test controller is also connected with another important artifact known as test analyzer. Test Controller has a two way communication channel with the test analyzer and client machines, which enables it to send and receive data across both the modules. Therefore, we can safely assume that Test Controller is the core module of our testing framework which initiates, executes and monitors the whole testing process.

G. Test Analyzer

After the Test Controller, Test Analyzer is the next important part of our framework. It keeps record of the number of test cases passed and failed. Such results are used to determine the level and quality of a software product. It is connected with test controller and different client machines. It analyzes the output of client machines and accumulates results and then it sends failed and deferred test cases back to the test suite repository so that test controller can fetch them again and try their re-run. It also helps test controller to update the test cases in test suite repository so that the successfully executed test cases should not be executed again. Test Analyzer also maintains test case logs which are needed by the Test summary module to generate different summary and analytic reports.

IV. Implementation And Experimentation

To validate our proposed framework, we prepared a test bed consisted of one server machine and one another machine which hosts multiple client machines in the form of virtual machines. For this purpose, we created four virtual machines on the computer using VMware Workstation 10.0. All these virtual machine were connected to the server and we installed soft bots (i.e., agents) on each of the client machine as well as on the server side. These agents are in fact a piece of code to communicate and coordinate with other agents deployed on other client machine as well as server machine.

A. Case Study

As a case study to perform the testing activities, we used a web-based application for "Employee Management". This web-based application was prepared for a company "**Cafedunord**". Employee management is a shift management platform to prepare a shift roaster for different employees of an organization. The manager can create different shifts and can allocate them to different employees. Employees receive emails about their whole weekly or monthly working schedule. Manager can also generate shifts related work report for employees. We prepared 50 functional test cases which correspond to different functionalities provided by the software such as login (authentication), adding employee, and assigning tasks to the employee, preparing shift schedule, making duty rostrum etc. Some of the selected test cases are appended as Annex-1 to this report. The prepared test cases were then converted into test scripts using the Selenium testing tool. The test cases were run on the Selenium using the "record" option, for which Selenium prepared the test scripts accordingly. We saved these test scripts for future use. The test scripts, which were in executable form, were then passed on to the test controller for distribution to the client machines for their execution.

TABLE I. DESCRIPTION OF TEST CASES BASED ON SYSTEM FUNCTIONALITY

Sr #	Functionality	Module/ Webpage	Test cases	Description of the selected test cases provided in Annex-1
1	Employee Authentication	Login.aspx	6	a. Check for valid Username and Password. b. Test with invalid Username and Password.
2	Registering New Employee	Add Employee.aspx	8	Valid user name and particulars for New employee.
3	Shift Management	Shift Template.aspx	13	Different shifts allocated different staffs members.
4	Scheduling	Employee Scheduling.aspx	17	Scheduling/rotation of employees by admin
5	Generate Reports	Reports.aspx	6	Generate the reports when required.

These sample test scripts generated by the Selenium are also attached as Annexure-II to this report.

B. Test Case Execution

We selected a web-based application named "Cafedunord employee management system" to test its functionalities for validation of our framework. It is actually a shift management application for the employees. We can create different shifts e.g. day-shift or night-shift and then we can allocate them to different employees. Employee receives their whole weekly or monthly schedule by email. We have created 50 test cases in the beginning from which we selected specific test cases to perform testing which are described in Table 1. Our framework is a distributing functional testing with multi-agent in which we have a server and a small bunch of client machines. In our test bed, the client machines are not physical machines but are virtual machines created using VM-ware Workstation. For the testing purpose, we use Selenium automation testing tool. Selenium is a suite of tools to automate web browser across many platforms. This testing tool is free and open source software.

Running tests cases in parallel calls for two things: an infrastructure to spread the tests and a framework which will run these kinds of tests with parallel in the given infrastructure. So, we can first make a distributed infrastructure and then create several tests cases, which will be executed in this distributed test environment. Selenium is powerful tool which can work with distributing environment and we can also record a test script for a particular test case. When we have to verify a test case, we will run its correspondent test script. Selenium automation testing tool and multi-agents are be deployed on server and client machines. Software agent is in fact a piece of code snippet that monitors and controls all the work related to communication and collaboration among the network nodes. To begin with the testing, all of our test cases are placed in the Test Suite Repository. Test Controller fetches the test cases from the repository. In Test Controller milieu, we use Selenium to create test scripts for those cases and agents will distribute those test scripts among the client machines depending upon the load on each machine. Every client machine will verify the test script using testing tool and will provide the result. Agents can share those results with each other through message passing which can create a speedy execution of test cases as well as reliable and robust testing environment. Every test result is sent to the Test Analyzer. Test Analyzer updates the repository with the test case status whether the test has been passed or failed. Test Controller can run the failed test cases again at a later stage. The execution summary of passed and failed test cases is provided in Table 2 whereas, statistics about the percentage of passed and failed test cases in shown in Table 3.

TABLE II. TEST CASES EXECUTION SUMMARY

Functionality	Test cases	Passed	Failed	Remarks
Employee Authentication	6	**5**	**1**	The logoff functionality was not working properly.
Registering New Employee	8	**6**	**2**	Change password option on the first login attempt was not active. Also employee's roles modification was not being carried out.
Shift Management	13	**13**	**0**	All the test cases passed in this module.
Scheduling	17	**16**	**1**	Erroneous behavior observed while assigning off days. Same off day could be assigned to all the employees.
Generate Reports	6	**4**	**2**	Two of the summary reports in the menu list did not generate anything.

TABLE III. PASSED AND FAILED TEST CASE EXECUTION SUMMARIES (PERCENTAGE)

Test Cases	Quantity	Percentage
Passed	42	84 %
Failed	6	12 %
Deferred (Error in test script)	2	4 %
Total	50	100 %

Graphical representation of Table 3 is shown in Fig. 2.

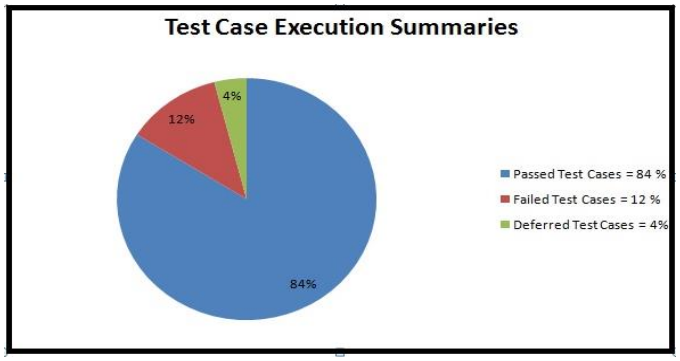

Fig. 2. Test execution summary.

A time based comparison of the test execution on monolithic and multi-agent based system in the distributed environment is shown in Table 4.

TABLE IV. EXECUTION TIME ANALYSIS USING MONOLITHIC AND MULTI-AGENT SYSTEM

Functionality (Module)	**Input fields**	**Test Cases**	**Test Run (Execution) Time on Monolithic Environment (*in seconds*)**	**Time of Execution using multi-agents (i.e., JADE agents in Distributed Environment)** (in seconds)		
				2 Client machines	*3 Client machines*	*4 Client machines*
Employee Authentication	2	6	12.86	8.21	6.91	6.72
Registering New Employee	10	8	150.41	84.7	70.13	48.12
Shift Management	4	13	34.12	23.78	16.37	15.93
Scheduling	5	17	180.24	104.89	75.12	60.13
Generate Reports	3	6	41.23	26.36	19.52	18.12

A graph showing the time of execution using multi-agents on multiple machines and on a standalone system which had no agents deployed on it is illustrated in Fig. 3.

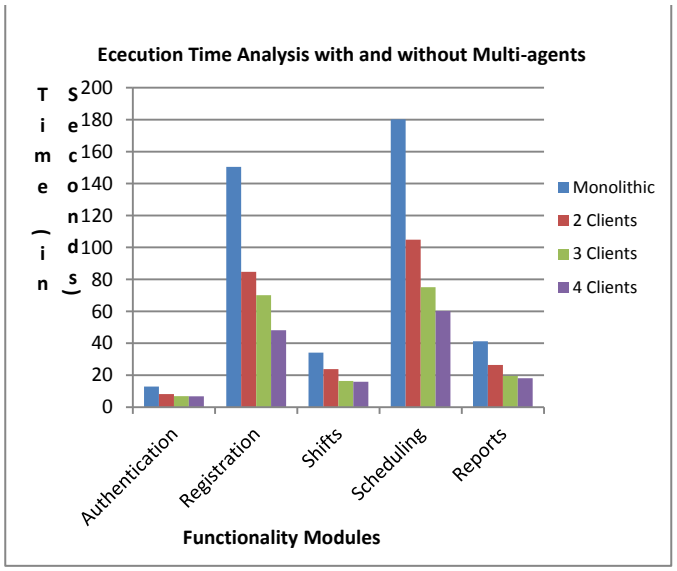

Fig. 3. Execution Time Analysis using monolithic and multi-agent system.

V. DISCUSSION

The framework proposed in this study will facilitate the regression testing of the applications which undergo several releases/builds by automating the testing process. The core reason for using collaborative multi-agent in this research was to make the testing activities faster and economical. The multi-agent approach has been widely used in the domain of computational intelligence as it has been proved to be an adequate approach where cooperative traits of specialized agents (or bots) are required. Since multi-agent are themselves distributed in natural and operate autonomously in different environment therefore, they support better utilization of computational resources.

In this study, we have employed a three-layered multi-agent architecture. We used JADE (Java Agent Development framework) to deploy multi-agents. A comparative analysis of the similar studies shows that our framework supports robustness and enhances test execution speed besides supporting controllability. Further, our model is scalable as well. Hence, our model supports the key performance measures as reported by other researchers. The distinctive part of our study is that it focuses on reducing Test execution time by utilizing more and more resources. A comparative analysis of our framework with other studies is provided in Table 5.

TABLE V. COMPARATIVE STATEMENT OF CONTEMPORARY STUDIES

Ref	**Purpose**	**Evaluation Parameters**	**Benefits/Strengths**
[6]	Agent oriented software testing approach is presented to enhance efficiency and quality of software products.	Efficiency	Agent based approach helped achieve better management of the software testing process.
[11]	Different methods for debugging the multi-agent system for software testing.	Debugging helps enhance the quality of software.	This paper proposes the methodology to test a relational database server as a central storage mechanism.
[12]	This approach enhances performance testing on distributed agent based web services.	Reliability, Accuracy, Dynamicity	It also combines the features of performance testing as well as functional testing and improves the system with respect to reliability and accuracy.
[13]	The proposed approach is used to improve the correctness of testing in distributed systems.	Coordination, Communication, Controllability, Observablility	Several problems influencing fault detection during the conformance testing process arise. So this approach reduces the problems of coordination and improves the controllability of system and enhances the fault detection.
[10]	Build a temporal logic assertion language to help detect the identified errors as well as build a dynamic analyzer based on temporal assertion language for testing agents.	Reliability, Communication, Fault Detection	The main advantage of using agent based testing is that it can generate test cases automatically and it can run continuously. This framework is more scalable in dealing with the distributed environment.
Ours	Our approach use collaborative multi-agents the core	Controllability, Efficiency (Speed)	Our framework supports robustness and enhances test

	reason for using collaborative multi-agent in this research was to make the testing activities faster and economical. Our study is focuses on reducing Test execution time by utilizing more and more resources.	Scalability, Observability	execution speed besides supporting controllability. Further, our model is scalable as well. Hence, our model supports the key performance measures as reported by other researchers.

VI. CONCLUSION

In this research we presented a multi-agent based framework to perform the functional testing in the distributed environment. The core reason for performing testing activities in the distributed environment was to reduce the cost, time and efforts ordinarily required to perform functional testing. We choose to deploy multi-agents on the client machine and server side to better coordinate the testing activities. To validate our proposed framework, we created 50 test cases for a web application called "**Cafedunord**". All the test cases were passed through Selenium automation testing tool to generate their test scripts which were also run through Selenium testing tool using the agents deployed on different client machines. The experimental results show that time to execute test cases was reduced by a proportional factor depending on the number of client machines.

REFERENCES

[1] E. Collins, G. Macedo, N. Maia., and A. Dias-Neto, "A Industrial Experience on the Application of Distributed Testing in an Agile Software Development Environment", In *Global Software Engineering (ICGSE) on IEEE Seventh International Conference, pp-190-194,2012.*

[2] G.A. Di Lucca, and A, R Fasolino, "Testing Web-based applications: The state of the art and future trends"on*Information and Software Technology, vol 48, pp-1172-1186, 2006.*.

[3] T. Clune, M. Rilee, and D. Rouson, "Testing as an essential process for developing and maintaining scientific software".on In *The 2nd Workshop on* Sustainable Software for Science: Practices and Experiences, 2014.

[4] M. Chu, C. Murphy, and G. Kaiser, "Distributed in vivo testing of software applications. In *Software Testing Verification, and Validation", on 1st International Conference on*, pp. 509-512, 2008

[5] B. Li, M. Grechanik, and D. Poshyvanyk, "Sanitizing and minimizing databases for software application test outsourcing. In *Software Testing, Verification and Validation (ICST)" on IEEE Seventh International Conference on*, pp. 233-242,2014

[6] N. Sivakumar, and k. Vivekanandan, "Agent Oriented Software Testing–Role Oriented approach" on *International Journal of Advanced Computer Science and Applications, vol 3, 2012*

[7] E. Uzuncoava, S.khurshid,, and D. Batory, "Incremental test generation for software product lines" on *Software Engineering, IEEE Transactions, vol 36, pp.* 309-322,2010.

[8] J. Lv, H. Hu, K. Y. Cai, and T. Y Chen," Adaptive and Random Partition Software Testing",2014.

[9] T. Chen, F., Kuo, H., Liu and E. Wong, "Code coverage of adaptive random testing", *on IEEE Transactions on Reliability, vol 62, pp.* 226-237.2013.

[10] F.E, Eassa., L.J Osterweil, M. A, Fadel, S. Sandokji and A Ezz," DTTAS: A Dynamic Testing Tool for Agent-based Systems" on *Pensee Journal, vol 76. 2014.*

[11] E. Serrano., A. Munoz. And J. Botia. "An approach to debug interactions in multi-agent system software tests." On *Information Sciences, vol 205, pp.* 38-57,2012.

[12] D. Hao, Y., Chen., F. Tang, and F. Qi. "Distributed agent-based performance testing framework on Web Services" In *Software Engineering and Service Sciences (ICSESS)* on *International Conference on*, pp. 90-94, 2010.

[13] S., Azzounzi., M., Benattou, and M.E.H Charaf, "A temporal agent based approach for testing open distributed systems." on *Computer Standards & Interfaces, vol 40, pp.* 23-33.2015.

[14] G.,Weiss, "Multi-Agent Systems." MITPress, Cambridge, MA.1999.

[15] M. F. Malik & M.N.A. Khan, "An Analysis of Performance Testing in Distributed Software Applications." International Journal of Modern Education and Computer Science, 8(7), 53, 2016.

[16] M. Wooldridge, "An Introduction to Multi Agent Systems." Wiley Second Edition.ISBN 978-0-470-51946-2.2009.