MULTI-AGENTS BASED SOFTWARE TESTING AS A SERVICE ON CLOUD

Sabih Jamal, Muhammad Aslam, Tauqir Ahmad
Department of Computer Science & Engineering,
University of Engineering & Technology, Lahore
sabih.jamal@uet.edu.pk, maslam@uet.edu.pk, tauqir_ahmad@hotmail.com

ABSTRACT: *Software testing plays an important role to make any software project reliable and successful. Owing to human effort and deviated results, manual testing is being replaced by automated testing tools. But the automated solution is much costly due to licensing and as it still requires human interaction. Hence, such a testing solution is required, in which on demand testing services based on 'pay as you go' model, can be provided. This framework is intended to highlight the role of intelligent agents in providing such a multi-agents based testing framework, where different agents collaborate to provide software testing as a service on cloud. Different agents' groups are introduced, to support different testing types within an integrated framework.*

Keywords: Multi-agents, Testing as a service, Cloud, Framework

1. INTRODUCTION

In entire software development life cycle, testing plays an important role in assuring quality of software applications. SQA Engineers are responsible for making sure that the software which is being delivered is error free, reliable, robust, secure and validated against required functionality. If all the efforts have been done in developing, but still somehow lacks in testing, then it may result in software failure, after its deployment. Hence, due to much effort and chances of errors/deviated results, currently manual testing is being replaced by automated testing tools.

Though automation gives rise to better results, but it is very costly, as tools are licensed based, also manual effort is still required to operate tools, which costs double and most of tools provide only single testing type. Hence, purchasing different tools for different testing types is not economical, especially for small and medium organizations. Thus, most organizations are looking for testing solutions, just like 'pay as you go model'. So that they only need to pay based on testing services they use, instead of purchasing tools / keeping their licensing. As cloud/ on demand services are evolving like (X as a service) Xaas, hence also there is need of 'testing as a service'.

In order to replace manual testing by automation and providing in form of on demand scalable services, there is need of such entities which can intelligently test applications on behalf of testers and can negotiate for cost while providing testing services. Hence, intelligent agents can play a key role in providing such on demand testing services, as agents are autonomous, social, proactive, reactive and also having learning capabilities.

The proposed system of Multi-agents based Software Testing as a Service on Cloud (MSTAS) highlights the role and significance of intelligent agents in providing such testing services. It is intended to conceptualize the multi-agents based framework, that emphasis that how different groups of agents can collaborate to provide software testing as a service on cloud.

2. RELATED WORK

Khalid A.M. [5] presents MARTS Multi-Agents based Regression Testing Suite that makes use of intelligent agents for test cases versioning, report and result analyzing. It provides interface for users to access testing history. It only covers regression testing and lacks in covering other testing types.

Nicholas R. *et a*l. [7] provides verification of the system into two classes. Logic gives axiom for every statement and semantic approach checks model for testing. But overall, it does not provide any concrete testing strategy.Yu Qi et al. [13] highlights an approach, specific test agents are generated from abstract classes. All the test agents test a specific type of web document or object. Web application testing is done by cooperation of a set of agents. These test agents follow the BDI (Belief- Desire- Intention) model agents

Qingning Huo and Hong Zhu [8] present agent based environment for web applications' testing. The system consists of a light weight agent platform which supports agents' communication ontology that provides integration of different agents and XML represents concepts of ontology. The agents can be distributed to different computers.

K. Mong Sim [19] proposes an agent-based approach to compose services in multi-cloud environments for different types of cloud services. N. Mehrotra [21] elaborates different aspects that how cloud testing differs from testing the cloud. S. Vilkomir [22] provides the review of cloud testing. Taas (testing as a service) or cloud testing includes testing the cloud and testing using the cloud.

[14,17,23] discuss cloud computing, its uses through a service-oriented interface to offer on demand services, gives overview of intelligent agents in cloud computing, the role of agents in cloud computing and how the Intelligent Agent can help in facilitating better services to cloud computing.

[11,15,24] specify mobile agents play vital role in implementation of cloud computing and network security testing. M Kim *et al*. [26] proposed an intelligent multi-agent model based on virtualization rules for resource virtualization to automatically allocate service resources suitable for mobile devices.

Yin et al. [27] proposed cloud testing platform based on virtualization technology. This platform is capable of providing testing resources, and these resources are assigned dynamically on demand, which reduces the cost of development and testing.

[12] addresses the service oriented testing, as services based on agents, while [28] only presents web security testing as a service. Impact of web 2 and cloud computing on software engineering and cloud computing related issues and implementations have been specified in [18,20]. Framework based on domain specific languages for software development and deployment on cloud, has been presented in[25].

3. MULTI-AGENTS BASED SOFTWARE TESTING AS A SERVICE (MSTAS)

MSTAS is multi-agents based integrated framework which provides different types of on demand testing services, just as 'pay as you go' model. It comprises of below specified modules as shown in Figure 1.

3.1 Request Analyzer Module

On browsing the MSTAS utility by the end user, the User Interface Agent (UIA) prompts the user to enter test types and url of the application to be tested. Then, UIA passes the user request to Coordinator Agent (CA). CA keeps knowledge base related to other agents' task and is responsible to send the input request or output data to corresponding agent.

Then CA passes the request to Request Repository /Analyzer Agent (RRAA), this agent keeps the request in its DB and based on KB of its selected test types, it sends more input parameters to UIA, which need to be asked from end user.

For example:

- If selected test type is 'Load', UIA asks how many virtual users are required to put the load.(Hence number of v-users is such an input which is dependent on which test type is selected)
- For 'Functional Test', UIA may prompt end user to input some requirements document.

Once detailed input is provided by the end user, then CA passes it to RRAA, which is stored in its repository. Then CA activates the Cost Quote Agent (CQA) to determine the cost for the test service.CQA determines the cost based on its KB and estimated number of resources as specified in implementation section. Through CA, the estimated cost is sent to UIA which is shown to end user. Now the end user is given option to proceed, negotiate or cancel, based on estimated cost.

- **Cancel Option**
  On cancelling, the end user is out of system (redirected to home page).
- **Negotiate Option**
  On selection of negotiation option, the request is passed again to CQA through CA.As CQA keeps on updating its KB, then CQA provides some more reduced costs, based on its available flexibility. Then end user is left with two options either to proceed or cancel.
- **Proceed Option**
  On proceeding the UIA prompts the user to enter payment mode related information, along with selection of test result option (either by email or at run time for short interval tasks).Once all inputs are given by user and kept in repository of RRAA, then this agent analyzes the request and decomposes each test request based on its test type. E.g. if there is test request from same user for performance test and organized in First in First out data structure then for same user, test service is broken down into two ids based on selected test types. Hence for above test, two ids are tagged against same user email.
  Rq1_userone
  Rq2_userone

RRAA then creates Monitor Agents (MA) against each request id, for above example

two agents MA1 and MA2 are being created and test service request input data is passed to monitor agents.

3.2 Test Type Selection and Test Executor Module

Based on requested test type kept by MA, the CA activates the corresponding test type agents group. MA is responsible for providing corresponding test related input data to Type Selector as well as Resource Scalability Provider module.

On activation of any test type, the corresponding agents group collaborates to generate the test (test script). Based on generated scripts, the test is executed.

For each test type, the corresponding agents have been defined in below section "Agents Groups based on Test Type".

Agents Groups based on Test Type

Functional Testing

The following agents collaborate to accomplish the functional testing:

- **Info-Extractor Agent (IEA):** It receives test input from MA and based on inputs (urls, source code or documents) extracts useful information for either black box or white box testing [6].
- **Test Data Generator Agent (TDGA):** TDGA generates test data from extracted sources based on formalism [1], [10].
- **Test Case Generator Agent (TCGA):**It generates the test cases, based on test requirements determined by IEA
- **Optimized Test Case Generator Agent (OTCGA):** It finds the optimized test sequences and based on sequences, it generates the optimized test cases [3].
- **Test Executor Agent (TEA):** TEA executes the test cases based on outputs of TDGA and OTCGA. On individual bases, the above defined agents perform their tasks based on mechanism as specified in detailed in [4], [9].

Load Testing

The below group of agents performs the required load testing based on method [2].

- **Recorder Agent:** It records the user test actions by capturing the client side user logs.
- **Virtual Test Replay Agent (VRA):** The virtual VRA replays the recorded actions, based on number of times of users specified by MA for creation of virtualized simulation.

Compatibility Testing

This group of agents collaborates to perform compatibility testing of web applications regarding to both browsers and operating systems.

- **Observer-Agent:** It gets the testing input parameters from corresponding MAs and passes to Computability Run Agent.

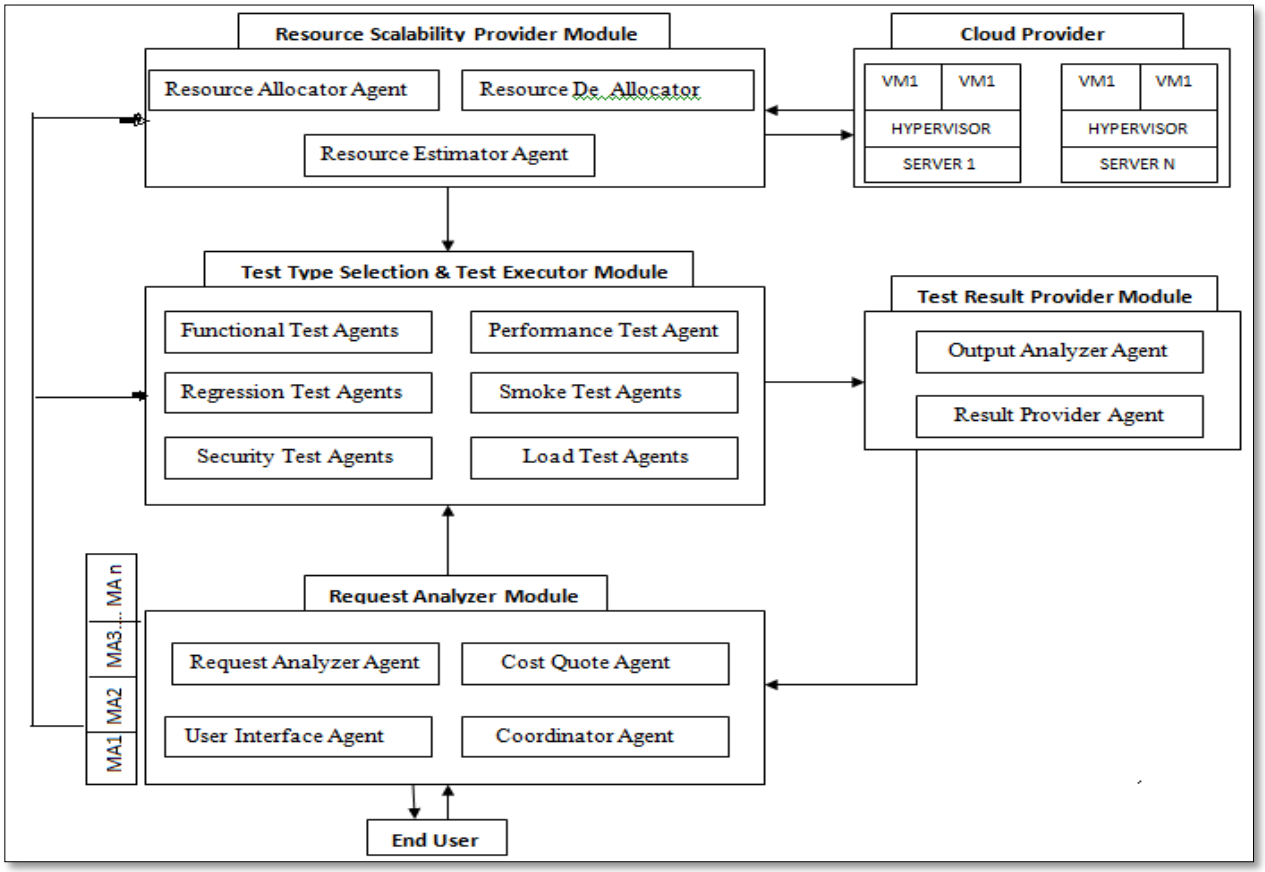

Figure 1: MSTAS Framework

- **Compatibility Run Agent (CRA):** It makes use of open source tools like Cross Browser and provide execution mode for running applications in different browsers and operating systems.

Performance Testing

Here, the Page Timer Agent (PTA) is responsible for calculating the load time of web page. It makes use of different plug-ins available through browsers like Firefox.

Security Testing

The basic theme of security testing is to uncover all the risky areas of the web. The below agents collaborate to perform security testing of agents.

- **Risk Finder Agent (RFA):** The RFA finds all risky areas of web, which need to be tested against security measures, embedded in its KB.
- **Authenticator Agent (AA):** It makes sure about authority and authenticity of users, accessing the web based on user roles provided by MAs.
- **Injector Agent (IA):** It injects different SQL injections to prevail unsecure areas of web.
- **Cross Site Script Checker Agent (XSCA):** It performs cross site scripting testing by adding XSS.

Smoke Testing

To uncover any high level change or major show stopper defects, smoke testing is performed by Smoke Test Run Agent. It makes use of broken link checker open source tool by integrating to its API.

3.3 Resource Scalability Provider Module

As different users can demand same or different types of test services, hence cloud providers provide resources pool, so that all users are facilitated over the cloud in such a way that SLA violations are avoided and resources are used in efficient way. In RSP Module, three agents are defined. Whenever any MA is created, to monitor any test request, it passes parameters to Resource Estimator Agent (REA), for each request the REA sets flags by analyzing that which resources are required with what capacity.

For example Load Test is based on Virtual users input by user. REA estimates the resources like for Load Test number of 50 V-users.

R1_CPU = 1RU (Required Resource Units)
R1_RAM = 4 RU

Now the estimated resources info is passed to either Resource Allocator Agent (RAA) or Resource Deallocator Agent (RDA). RAA and RDA communicates with hypervisors of data centers to either allocate or deallocate the estimated resources.MA keeps record of resources occupied by each test request.

3.4 Test Result Provider Module

Test Execution Module's output serves as input to Result Provider Module. The module consists of two agents: Output Analyzer Agent (OAA) and Result Provider Agent (RPA). OAA extracts the inputs from executor module; it analyzes those based on inputs by executor and forwards to RPA, which adjusts the output in formatted results based on

selected test type like load testing graphs or if detailed reports required for functional issues.

Once results are extracted, CA passes those to UIA and these are displayed to end user. Once test is done, the CA notifies RRAA and this agent in turn destroys the corresponding MA, and before destruction the MA activates the RSP Module and RDA is notified to release the resources held by above agent.

4. IMPLEMENTATION

The MSTAS is being implemented in JADE, as it is an open source framework that implements FIPA specifications in Java. As JADE provides library of classes in the form of packages, which are being imported during the development. Agents creation, communication and tasks based actions are being implemented. Communication messages are exchanged among agents by using FIPA Agent Communication Language, ACL. In MSTAS, SL is being used as content language, as it makes use of 1st and 2nd order logic along with Conjunction, Disjunction operators.

Implementation details regarding some areas of framework have been specified as below:

4.1 User Interaction with MSTAS

The system user is provided testing types options for interaction with system, as per shown in Figure 2

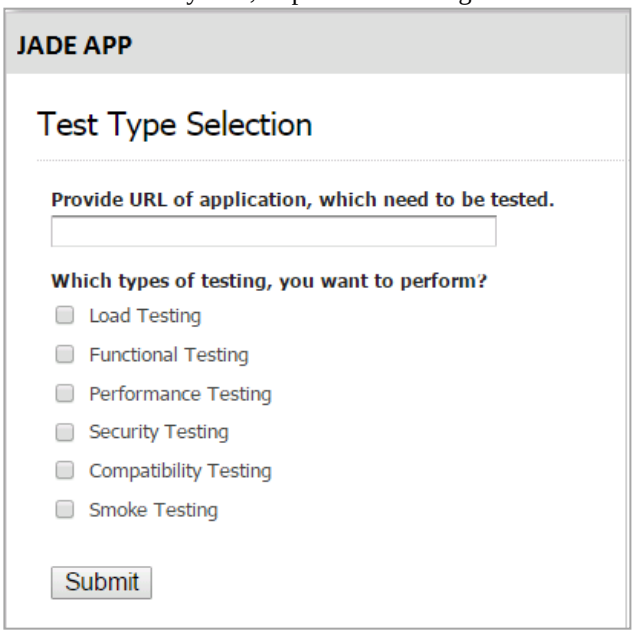

Figure 2: Test Type Selection UI

4.2 Ontology and Content Language for Agents Communication

In order to communicate, agents must agreed on some common language usage e.g. in order to perform load or any different type of testing, related terms must be specified and updated in that testing domain. For content based communication SL is being enabled in **JADE** by **JSA** which is JADE base **Semantics Add On** [29].

The below piece of code, taken from MSTAS Framework shows that how an agent sends messages to another one. The UI Agent sends information to Coordinator Agent (CA), updating about number of virtual users required in load testing.

```
{
ACL Message msg = new ACL Message(ACL Message.Inform);
msg.addReceiver(new AID ("Coordinator-Agent", AID.ISLOCALNAME);
msg.setLanguage(" FIPA SL");
msg.setOntology("Load-Testing");
msg.setContent(" 50 Virtual Users are required");
send (msg);
}
```

4.3 Prompt for detailed test inputs

Based on selected test type, the RRAA infers for more detailed inputs based on its knowledge base KB. The following pseudo code shows that how system user is prompted for detailed inputs based on selected test type.

```
public prompt-detailed-input( )
 {
   if test-type is Load then
   input number of virtual-users && response-time
   else if test-type is Compatibility then
   input browsers-name & operating-system-name
   else if test-type is Security then
   input sample authorized-user-roles
   }
```

4.4 Cost determination and Negotiation

CQA determines cost based on inputs provided by REA as number of estimated resources. For example if CQA has to determine cost for Load Testing.

Let a web page has to be tested for Load Testing , then how much resources may be required, let D be the database, and S be the set of rules for the knowledge base and A be the selected action, then by using 1st order logic, predicates are defined:

Load-Testing-> LT
No-of-Vusers ->NU
Client Static Page ->CSP
Client Active Page -> CAP
Serve Static Page ->SSP
Server Active Page -> SAP
Server Dynamic Page ->SDP

If there is load need of 50 virtual users to be applied on a client static web page then it may require resources units as shown below transformation function of *input parameters to resources.*

i. LT ∧ 50 U ∧ CSP → 25 CPU units → 25 CU
   → 10 Network Bandwidth units → 10 NB
   → 5 Memory Storage Units → 5 SC

Based on above set of rules for the above knowledge base, now REA can estimate resources units for load testing of 100 users accessing the client server page.

ii. LT ∧ 100 U ∧ CSP → 50 CPU units → 50 CU
   → 20 Network Bandwidth units → 20 NB
   → 10 Memory Storage Units → 10 SC

iii. Like above many more rules are available in KB of REA
   LT ∧ 50 U ∧ SSP → 30 CPU units → 30 CU
   → 20 Network Bandwidth units → 12 NB

➔ 10 Memory Storage Units → 7 SC

Now CQA determines the cost for the testing service to system user. Based on inputs for estimated resources by REA, the CQA utilizes these inputs and transform these to service charges.

E.g. if user wants to have Load Testing of 50 virtual users on client static page and same for static server page, then CQA determines total cost as shown below:

Let: P(x):25 CU ∧ 10 NB ∧ 5SC → 200 AU

Q(x):30 CU ∧ 12 NB ∧ 7SC → 250 AU

Here each rule must be checked where it validates, then prescribes an action with predicate, these rules with next function (conversion of resources units to amount) generates required behavior of the agent.

Hence P(x) ∪ Q(x) → 450 AU

If prescribed amount is not agreed by user then negotiation takes place by following formalism. The CQA has cost reduction parameter stored in its DB say X AU, it applies it on amount calculation rules.

As per function named ***new***

New = pres-amount – reduction-param →neg-amount

(negotiated amount is calculated by subtracting the reduction parameter from prescribed amount)

4.5 User Requests Repository

- **Data Structure for requests**
  RRAA keeps internal data structure to have repository of all user requests. The requests are organized in First in First out (FIFO) data structure. Though as cloud based pool of resources is available and thus there is no much time required to serve any request but still all requests are served in FIFO structure.
- **Request Control Block**
  Each user request is split and monitored by MAS, as MA keeps information related to each request id, type, number of occupied resources. Like in operating system, all information related to any process is stored in Process Control Block. Hence MA keeps info of each test request in Request Control Block.
- **MAs Destruction**
  In order to destroy MA, the RRAA invokes the *takeDown()* method [29].

4.6 Integration with Cloud Platform

As it is being implemented in JADE and Java based, so it is integrated with all those platforms that support Java Runtime Environments JRE and provide such APIs.

5. EVALUATION

The framework is evaluated based on below specified parameters:

5.1 Avoiding SLA Violation

Based on testing service needs, the resources are estimated by agents and these agents interact with hypervisors of cloud providers, hence whenever any request is received, the resources are allocated by host machine. If any host having 4 Virtual machines (VMs), then if request arrives by agent .If the first three VMs are already allocated then the 4th VM is allocated. If all the VMs of a host are allocated, then request is passed to next host. VM migration detailed algorithms based on resources need, are specified in [16]. As, resources are allocated based on shared pool, hence cloud based resources are used efficiently without any deficiency. So, SLA violation is avoided in terms of availability, performance (maximum responses), and location independent access.

Let say the number of resources required in terms of VMs depend on the user requests, it can be simply validated by the regression method. Let X is the number of test requests arrived simultaneously and Y represents the allocated resources (e.g. CPU in MIPS) scaled automatically based on MSTAS.

Table 1: Allocated CPUs for user request

X (No. of user req)	30	50	80
Y (Allocated CPU MIPS)	400	600	800

Equation of regression line becomes,

Y= a + bX

Y = 178.947 + 7.895 X

We are interested in finding the correlation coefficient.

$$r = \frac{n \cdot \sum XY - \sum X \cdot \sum Y}{\sqrt{\left[n\sum X^2 - (\sum X)^2\right] \cdot \left[n\sum Y^2 - (\sum Y)^2\right]}}$$

$$r = \frac{3 \cdot 106000 - 160 \cdot 1800}{\sqrt{\left[3 \cdot 9800 - 160^2\right] \cdot \left[3 \cdot 1160000 - 1800^2\right]}} = 0.9934$$

Hence, the correlation coefficient value is almost near about to 1, which shows that as per need the resources are allocated dynamically and SLA violation is avoided.

5.2 Replacement of testers' expertise

As intelligent agents collaborate throughout the framework, even by taking inputs from users to process that input requests, testing and allocating resources on cloud, hence there is lesser human interaction required and testing expertise are not required.

5.3 Lower costs (Economical approach)

Currently, all the automated testing approaches are highly costly and licensed based. Suppose any automated testing tool serving any specific testing type costs θ, also human effort in form of SQA Engineers required too which may also cost θ too. Hence, overall it costs 2θ. So, for more services the cost becomes number of times the services are required. But for MSTAS, as user only needs to pay on usage basis, so cost becomes even $\frac{\theta}{2}$.Say licensed based test approach costs θ and multiple testing types required then it costs $\sum_{i=1}^{n} \theta$

While MSTAS based testing costs $\frac{\theta}{2}$ or $\frac{\theta}{3}$, as it allows user to get services, based on pay as you use model.

$\sum_{i=1}^{n} \frac{\theta}{r}$ (Where r is any real number)

Then from above equations,

P(x): $\sum_{i=1}^{n} \theta < \sum_{i=1}^{n} \frac{\theta}{r}$

The above equation can be proved by mathematical induction.

Basis Step: P(x) holds for n=1.

Induction Step: If P(x) holds for k then it must hold for k+1.

$$\sum k + (k+1) < \sum k + (k+1) / r$$

5.4 Flexibility

One of the leading aspects of this framework is that it provides flexibility, both in terms of testing types and integration with cloud platform.

- **Flexibility w.r.t. testing types**

As framework is Java based, hence using Adapters design pattern allows support of introducing many more different test types by using polymorphism in agent classes.

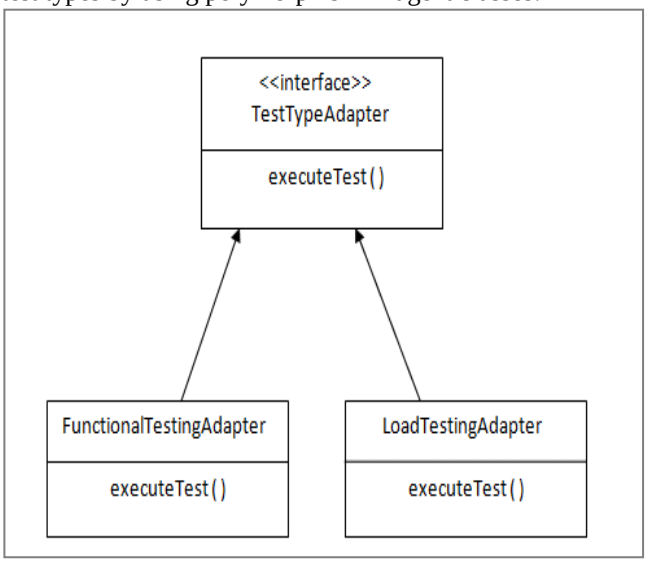

Figure 3: Flexibility in Test Types

- **Flexibility w.r.t. integration with cloud platform**

As already specified in implementation section that it is implemented in JADE, hence, the MSTAS can be integrated with all those platforms that supports JRE and provides APIs for integration. Like, Google Apps Engine, Amazon's Web Services, Open Stack and all such platforms.

6. CONCLUSION

The proposed framework MSTAS provides software testing as service by utilizing shared pool of resources, over the cloud. It makes use of multi-agents which collaborate intelligently on behalf of testers and interact with cloud providers' platforms for allocation of resources, required dynamically based on testing requests. Pay as you use model of testing service, makes it most useful for small and medium businesses, where instead of purchasing licensed based testing tool, users can simply pay as they use.

Though MSTAS provides an integrated framework for testing different types, but in future it can be further extended for incorporating features like test plan development too. Currently, it is being implemented in JADE, so it is also under consideration to make framework compatible to any agent development framework.

REFERENCES

[1] Anjaneulu Pasala and Manuel R., "An Approach to Generate Test Input Data from UCAD Model", in *SetLabs Briefing*, Vol. **9** No. 4, 2011

[2] Baowen Xu and Lei Xu, "Applying Agent into Web Testing and Evolution" in *National Natural Science Foundation of China (NSFC),* pp794-798, 2004

[3] D.Jeya Mala and V. Mohan, "Intelligent Tester Test Sequence Optimization Framework using Multi-Agents", in *Journal of Computers.* Vol **3**, No. 6. 2008

[4] Junyi Li, Lulu Sun and Shenglan Liu, "Web Applications Testing Framework based on Multi-Agent" in *International Journal of Advancements in Computing Technology (IJACT),* Vol. **3**,No. 10, 2011

[5] Khalid A.Muhammadi., "Multi-Agent based Regression Testing Suite" in *Athabasca University Thesis*, April 2009.

[6] M.A.B Junior, F.B. de Lima Netoi and J.C.S. Fort, "Improving Black Box Testing By using Neuro-Fuzzy Classifiers And Multi-Agent Systems", in *Proceedings of 10th International* Conference on *Hybrid Intelligent System* 2010

[7] Nicholas R. Jennings and M. Wooldridge, "Agent-Oriented Software Engineering", in *13th International Workshop on Agent Oriented Conference*, June 2012.

[8] Qingning Huo and Hong Zhu, "A Multi-Agent Software Environment for Testing Web based Applications", in *Proceedings of the 27th Annual International Conference on Computer Software and Applications*, pp 210-215, 2003

[9] Samad Paydar and Mohsen Kahani, "An Agent-Based Framework for Automated Testing of Web-Based Systems", in *Journal of Software Engineering and Applications*, pp 86-94, February 2011

[10] Siwen Yu and Jun Ai, "Software test Data Generation based on Multi-Agent" in *International Journal of Software Engineering and its Applications*, Vol. **4** No. 1, January 2010.

[11] Athanasios Tao Kaygianis, "Network Security Testing using Mobile Agents", in *conference of Practical Applications of Intelligent Agents and Multi-Agents Technology*, March 1998.

[12] Xiaoying Bai, Guilan Dai,D. Xu and W.T Tsai, "A Multi-Agent Based Framework for Collaborative Testing on Web Services", *in Proceedings of the Fourth IEEE Workshop on Software Technologies*, 2006

[13] Yu Qi, David Kung and Eric Wong, "An Agent based Testing Approach for Web Applications", In *Proceedings of the 29th Annual international Computer Software and Applications Software*, 2005.

[14]Kwang. Mong Sim, "Agent based Cloud Computing",*IEEE Transactions on Services Computing*, Vol. **5**.pp 564-577, 2012

[15]Aarti Singh and M. Malhotra, "Analysis for exploring scope of mobile agents in cloud computing",*In International Journal of Advancements in Technology*,Vol.**3**. Issue 3., 2012.

[16] M. A. Ayyub,Yaser and M Daragmeh "Multi-agent based dynamic resource provisioning in cloud

computing Systems", *In International Journal of Science & Technology*, 2012.

[17] D Kumar and Ashwin R, "Multi-agent based Cloud Services",*International Conference on EGovernance & Cloud Computing Services, 2012*

[18] Radha Guha and David Al-Dabass, "Impact of web

2 & Cloud Computing platform on software engineering" In *IEEE International Symposium onElectronic System Design,2010*

[19] J Octavio and K. Mong Sim, "Agent based cloud service composition", in *Intenational Journal of Artificial Intelligence* ,Vol **22** No.2, 2012

[20] Mladen. A. Vouk, "Cloud Computing- Issues, Research and Implementations",*in Proceedings of the 30th International Conference on ITI,* 2008

[21] Neha Mehrotra, "Cloud Testing Vs Testing a Cloud",in *10th International Software Testing Conference,* 2010

[22] S. Vilkomir, "Cloud Testing: A state of the art review", *In International journal of information & security,* Vol **28**,No.2, pp 213-222, 2012

[23] A. K. Srivastava,V. Srivastava and Richa Bhargava, Towards developing an intelligent agent for cloud computing",in *International Conference on Cloud,Big Data and Trust* Nov, 2013

[24]A. Ali, A. Abdullah and Okba Kazar, "Implementation of cloud computing approach based On mobile agents",In *International Journal of Computer & IT*, Vol. **2**,Issue No. 6, 2013

[25] Bezad Bordbar, Krzysztof S. and R. Anane"A DSL based approach to software development & deployment on cloud" In 24th *IEEE International Conference on Advanced Information Networking & Application,* 2010

[26] M Kim, Hanku Lee, H. Yoon, Jee-In Kim and H Seok Kim, "An Intelligent Multi-agent model Based on cloud Computing for resource virtualization", In *International Conference on Information and Electronics Engineering*, Vol . **6**, 2011

[27] Lei Yin, Jin Zeng and Bo Li, "CTPV: A cloud testing platform based on virtualization" In *IEEE 7th International Symposium on SOSE,* pp 425-428, 2013

[28] H Ling Shan, "Test as a service: A framework for web Security Taas Service in cloud environment",In *IEEE 8th International Symposium on SOSE,* pp 212-217, 2014

[29] Fabio B, G.C and D.Greenwood, "Developing multi-agent systems with JADE".