A Multi-Agent System Approach for User-Session-Based Testing of Web Applications

MOHAMMADREZA MOLLAHOSEINI ARDAKANI
Department of computer engineering
Islamic Azad University – Maybod Branch, Iran

MOHAMMAD MOROVVATI
Department of computer engineering
Islamic Azad University – Ardakan Branch, Iran

Abstract: Web applications are becoming increasingly complex and yet important for companies. As web applications become more and more prevalent, their testing and quality assurance become more and more important and crucial. Due to the complexity of the underlying technologies of web applications, testing of such softwares is challenging.

Applying user sessions data as test cases can reduce the cost of regression testing process. In this paper we provide a multi agent software environment in order to support automatic replay of user sessions. In this environment different agents can perform replay of user sessions by being distributed on various platforms and geographic locations.

Key-Words: Web Application, Multi-Agent, User session, Software Testing

1 Introduction

The Internet and web are becoming a distributed platform that provide new ways for developers in order to use softwares. Web applications are typically software systems that are accessible over the web, interacting with the user through an internet browser. Web applications require the presence of web server in simple configurations and multiple servers in more complex settings. Such applications are more precisely named web-based applications. The use of web applications is increasing rapidly. Nowadays most of the business companies trade online. Universities and different academy centers perform most of their activities such as registration, financing, etc through internet. In a sense, applying the web application is to facilitate and speed up the tasks and their utilizations are inevitable in our daily lives.Therefore, test and quality assurance of these softwares is needed and necessary.

Web applications have special characteristics such as evolutionary life cycle and rapid updating. They often use a diversity of information representation formats and execution platforms. Their components can be developed using various techniques and can be written in different languages. These characteristics have caused many problems in their testing and made traditional software techniques and tools ineffective and impractical.

Applying the user session as test cases is one of the useful and inexpensive ways in web application testing. Related data to the user sessions may be relevant to different users in different geographic areas that have used various platforms at similar or different times.

Therefore, by automation of replay user sessions on corresponding software platforms and geographic locations, we can make the newly obtained results more reliable during comparison with previous results. We can also replay these sessions in parallel ordering.

For this purpose in this paper we provide a multi agent software system for automating replay of user sessions. In this system several cooperative agents with coordination of management agents accomplish replay of user sessions.

The remainder of the paper is organized as follows. In the next section we briefly describe related work on validating web software. Section 3 reviews some of the web application testing techniques that are bases of the paper. Section 4 presents details of our proposed approach. Section 5 presents a prototype system of the approach. Section 6 discusses the summary of paper and comments on future work.

2 Related work on validating web software

So far many techniques have been offered for testing of web softwares. Some of these techniques can be categorized in the white box testing techniques such as Ricca and Tonella's [7] approaches. One limiting factor in the use of these techniques is the cost of finding inputs that exercise the system as desired. Selection of such inputs is slow and must be accomplished manually.

In recent years, agent oriented software engineering (AOSE) has emerged new discipline for modeling and designing cooperative, intelligent and autonomous agents. By using software agents in software testing processes in general, and web application testing in particular, we can provide useful and effective solutions in this context, so as to facilitate the testing process and to reduce the costs of test and maintenance phases which are very expensive in software development process.

Kung [6] offered a framework based on BDI agent and unified modeling language (UML). By regarding this point that the web applications consist of heterogeneous documents, he employed different types of test agents for different types of web documents.

Huo et al.[5] proposed a multi agent software environment for testing of web applications. In their proposed environment, each agent with its special capability can perform the given tasks. They have shown this information in a software testing ontology.

3 Web application testing techniques

3.1 User-session-based techniques

A web application generally encompasses a set of static and dynamic web pages. Based on user requests and server state, the web application generates dynamic responses. Changing user profiles and frequent small maintenance changes complicate automated testing. Using the white box testing techniques of web applications cost a lot to produce test cases.

User-session based techniques can help with this problem by transparently collecting user interactions and transforming them into test cases. The techniques capture and store the clients' requests in the form of URLs and name – value pairs, and then apply strategies to these to generate test cases. Each user session is a collection of user requests in the form of URL and name-value pairs (i. e., form field names and values). More specifically, a user session begins when a request from a new IP address reaches the server and ends when the user leaves the web site or the session times out. To transform a user session into a test case, each logged request of the user session is changed into a HTTP request that can be sent to a web server. A test case consists of a set of HTTP requests that are associated with each user session. Different strategies are applied to constructs test cases for the collected user sessions [3]. Capture/replay for web application is relatively cheap Compared to other test domains.

Elbaum et al [2] provide promising results that demonstrate the fault detection capabilities and cost-effectiveness of user session techniques improves as the number of collected sessions increases. However, the cost of collecting, analyzing, and replaying test cases also increases.

Sampath et al [8] presented an approach to achieve scalable user session-based testing of web applications. They view the collection of logged user sessions as a set of use cases where a use case is a behaviorally related sequence of events performed by the user through a dialogue with the system.

A complicating factor for user session-based techniques involves web application state. When a specific user request is made of a web application, the outcome of that request may depend on factors not completely captured in URL and name value pairs alone; for example, the university registration online system depending on the number of registered students in one course unit may function differently.

3.2 Agent-based techniques

In the last few years agent based techniques have become very popular. The agent's capabilities can be used in software testing process and especially in web applications testing. Up to now, few techniques based on agents have been introduced for testing of web applications. The bases of our proposed approach are mentioned at [5], Therefore we are going to review them briefly.

Huo et al [5] formed a multi agent software environment for testing of web applications. In this environment, in general, two types of agents have been implemented for testing of web applications. First, test service agents with special capability each that can perform different testing tasks .The other, broker agents with responsibility of scheduling and assigning of testing tasks to other agents. figure 1 shows the primary structure of such system.

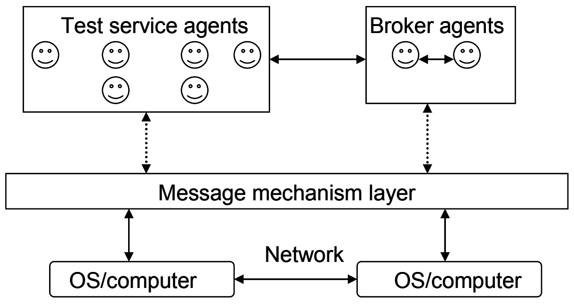

Figure 1. The primary structure of system: proposed by Huo et al.

They have utilized a software testing ontology to enable the flexible integration of agent into the environment and the effective communications between agents and among human testers and agents. The ontology is used as the content language for software agents to register into the system with a capability description, for human testers and agents to make testing requests and report testing results, and also for management agents to allocate tasks to test service agents. They have already presented some relations and concepts in this ontology, such as agent capability and testing task by using BNF representation and XML schema.

In their prototype system, each testing service agent fulfills a specific activity such as test case generation and test case execution. Test case generator agents mostly utilized the extension of traditional white box testing techniques.

4 The proposed approach

As mentioned in section 3, one of the useful and cost effective ways for web applications testing is applying user sessions as test cases. The application of Lehman's theory of software evolution to web-based applications shows that they are by nature evolutionary and, hence, satisfy Lehman's laws of evolution[11]. The essence of web applications implies that supporting their sustainable long term evolution should play the central role in developing quality assurance and testing techniques and tools. Keeping this feature in mind, the regression testing is considered as one of the most highly used test contexts for web applications. Some times, as a result of modifications that made in a web application, the server response (the result of web application code process) shouldn't change when we replay a certain group of previous user sessions. In other words, the newly made change(s) have no effect on the previous user session's operation. Therefore, automation of replay of user sessions and verification of equality or inequality of current results in comparison to the previous or excepted results is one of the major challenges in user session-based testing. The Application of Software agents, especially mobile agents, can provide effective and promising solutions in this field.

We know that user sessions may occur in various platforms and at different times. That is, the different users can utilize different platforms for operating single web application as concurrent or non concurrent. The platform differences are often due to incompatibility in software platforms that are applied by users, i.e. type, configuration and version of user's operating system or internet browser. In addition to platforms differences, some times, time restrictions, as well as geographic location of a user who operates the web application may have an effect on the received web application responses too. Therefore, better comparison results can be obtained if user sessions are replayed on previous user-software- platforms in similar geographic locations and time slots. Moreover, by utilizing agent capabilities we can replay the user sessions as parallel, sequential and any other desired ordering.

For this purpose we can implement some agents that can be distributed on different computers at different geographic locations on intranet or internet. Each agent has especial capability(s). Concepts of agent capability and testing task here differ from those offered at [5].

In our approach the *agent capability* can include the following instances:

- Platform or platforms that an agent will be able to replay the user sessions on top of them. (we name these platforms as supported platforms)
- The geographic area in which agent is located.

We can briefly say that, a *testing task* determines within what platform and even at what time an agent must replay the delivered user session.

5 Prototype system

To demonstrate the feasibility and capability of our proposed approach mentioned above, we designed and implemented a prototype system to automate failure detection of web applications by replaying user sessions.

As shown in figure 2, the system consists of a number of agents. They cooperate with each other to replay user sessions and report failure detection, automatically. The agents can be mobile and agent society is dynamically changing; new agents can be added into the system and old agents can be replaced by a newer version.

Communication mechanism in this system is based on the mailbox scheme [1,4]. Generally, a mailbox is an unbounded buffer of messages. The mailboxes also can be mobile. Each agent has an individual mailbox. An agent can direct a message to another agent's mailbox, and the receiving agent uses a push or pull operation to obtain the message from the mailbox.

Interagent communication thus consists of two distinct steps:

- Transmission of a message from the sender to the receiver's mailbox, and
- Delivery of the message from the mailbox to its owner agent.

We can use the theory of speech act [9,10] in order for agents to clearly express the intention of exchanged messages between themselves. For example, an agent's capability transmission message differs from a replay result transmission message. Among the seven parameters of speech act, three of them are more applicable in our proposed system, which are:

ASSERTIVE: agent claims its capability to Broker or agent sends his results of testing task.

EXPRESSIVE: agent describes a testing task to be implemented by others.

DIRECTIVE: broker assigns a task to an agent.

Generally, there are three groups of agents in this system that the summary discussion on their functionality is as follows:

- Interface agents that perform as user interface and get the replay command from human testers and deliver to tester the results of replay user sessions.
- Failure detection *agents* that themselves divide to two classes:
  1. *Test Oracle agents* generate expected output, which can be utilized to determine if the system is executing properly during testing process. These agents also compare the actual results of the system under test to determine test case success.
  2. *Replay agents* automate the replay of user sessions that are situated in USDB. These agents send URL requests and name-value pairs to the web server and collect the server's responses, which the test oracle agents use.
- *Broker agents* manage other agents and are responsible for the analysis of user sessions, registration of agent's capabilities, task scheduling, and monitoring and recording agent's states. These agents are implemented to negotiate with other agents to assign and schedule testing tasks, e.g. replay command of user sessions. Each broker manages a registry of other agents and keeps a record of their capabilities and performances. Broker agent stores this information in a knowledge base. In this system, each agent registers its capabilities to the brokers when joining the system.

Agents are adaptive, and they can adjust their behaviors based on environment changes. Agents must be able to move within heterogeneous networks of computers. This is only possible if there is a common framework for agent operations across the whole network: a standardized *agent infrastructure*. This infrastructure must offer basic support for agent mobility and communications. *Aglets* can be effective for this goal. It is a java based mobile agent platform and library for building mobile agents based applications. An aglet is a Java agent which can autonomously and spontaneously move from one host to another carrying a piece of code with it.

USDB can be considered as reduced test suite that is proposed at [8] (of course by adding header information such as previous location and platform of user sessions) that are produced by applying the concept analysis tool.

Broker agent also stores some of the relations as basic fact in the knowledge base. These information (agent's capabilities) and relations are bases of the broker's decisions when he wants to choose the most suitable agent for replaying a user session. The most important of these relations consist of:

(A) *Enhancement relation between platforms:* A platform A is an enhancement of platform B, if a testing task can be performed in platform B implies that it can also be performed in platform A. Assume that an enhancement relation is defined on software and hardware components. The enhancement relation between platforms can be defined formally as fallows. Let platforms A and B consist of sets $\{a_1, a_2, ..., a_n\}$ and $\{b_1, b_2, ..., b_m\}$ of hardware and software components, respectively.

A is an enhancement of B, if and only if for all $b_i, i = 1,2,...,m$ there is one component $a_j \in \{a_1, a_2, ..., a_n\}$ such that a_j is an enhancement of b_i.

This relation is partial ordering. That is, this is transitive, reflexive and asymmetric.

(B) Match relation between a testing task and agent capability: In the assignment of a testing task to an agent, a broker agent must answer the question whether the testing task matches the capability of the agent.

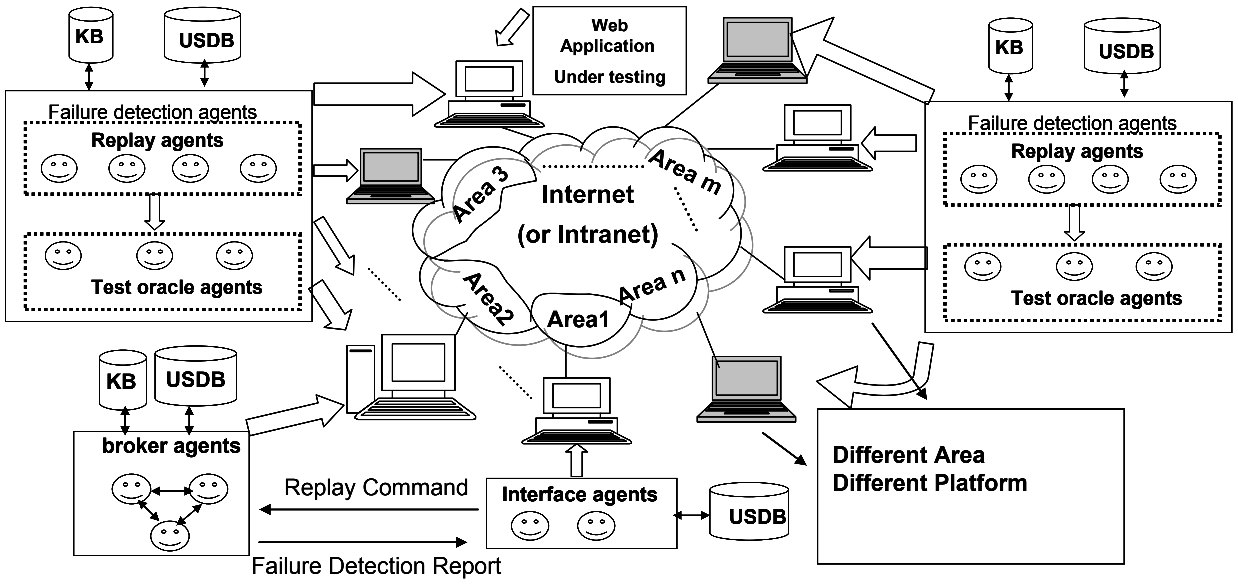

Fig.2 System structure

For example, assume that a replay agent with capability C is registered as capable of replaying user sessions on platforms $p_1, p_2, ..., p_n$. Also assume that a testing task T is requested for replaying a user session US that previously occurred on platform p_i.

The broker agent needs to infer that the replay agent is capable of fulfilling the task. We can say that the replay agent is capable of replaying the user session US, or in other words we can say that capability C matches task T, if and only if both following conditions are true:

- Capability C and testing task T have the same geographic location. (The replay agent is located in the same area that US has occurred previously)
- $p_i \in \{p_1, p_2, ..., p_n\}$

However, if broker agent can't find an agent that satisfies second condition in the match relation, it inevitable tries to find an agent such that:
$< \text{Enhancement of } p_i > \in \{p_1, p_2, ..., p_n\}$.

Now we present a prevalent scenario between replay and broker agents. Broker B is considered as management agent. Assume that agents $A_1, A_2, ..., A_n$ are responsible for replaying user sessions. We are going to replay a user session that is named US.

1) The Agents $A_1, A_2, ..., A_n$ register their capabilities to the broker B when joining the system. This function can be implemented by sending ASSERTIVE messages with <capability> parameters from replay agents to broker B.
2) The broker searches its knowledge about registered agents, and finds that agent A_m $(1 \le m \le n)$ is the best match for replaying US. It then sends a DIRECTIVE message with the <task> parameter to agent A_m.
3) Agent A_m replay this session with regards to its current platform and the available information in <task> parameter, that is:
   - If its current platform is conformed to the platform that is inserted into the <task> parameter, it will replay the US in the same platform. Otherwise,
   - It chooses most suitable platform for migration and will accomplish the replay of US in new platform (new host).

Note that with cooperation of relay and oracle agents a message is sent with <answer> parameter to the broker. This message determines the result of replaying US. If agent A_m was not able to replay the US for any reason, then broker tries to find another agent for replaying US and then assign this task to it.

It is also important to note that broker agent always chooses the most suitable agent with minimal cost for replay of a given user session. Many factors can be involved in this operation, for instance: geographic location, supported platform and current platform of each agent.

Example 1. In the simplest case consider that there are four agents in the system that we name A_1, A_2, A_3, and A_4. We are going to replay a user session named US that previously occurred in platform P_2 and geographic location L_1. Suppose that only three factors mentioned above are interfering in the *replay agent* selection. Table 1 shows the capabilities of these agents.

With referring the table it is obvious that, among these four agents only the agents A_1 and A_2 are suitable for replay US.

Among these two agents, the broker agent chooses agent A_1 as the most suitable agent, because the current platform of agent A_2 is P_3, hence, this agent must migrate to platform P_2 to replay US and agent migration is an overhead. However, if agent A_1 did not exist before or it was not able to replay US for any reason after selection, the broker agent will have to deliver replay of US to agent A_2 .

Table 1. Capability of replay agents.

Agent Name	Geographic Location	Supported platforms	Current Platform
A_1	L_1	P_1,P_2	P_2
A_2	L_1	P_2,P_3	P_3
A_3	L_1	P_3,P_4	P_4
A_4	L_2	P_2,P_4	P_2

6 Conclusion

The essence of evolutionary and the short interval between different versions of web application have caused considerable increase in maintenance phase duration and cost of such softwares.

Replaying subset of user sessions and comparing their results to the expected results is a way to find new faults. Diversity of platforms and geographic locations in which user sessions have occurred previously will motivate us to replay user sessions in similar platforms and geographic areas. Also, the frequent replaying of user sessions in maintenance phase requires an environment to automate this operation.

This paper presented an application of multi-agent system to satisfy the mentioned requirements. It clearly demonstrated that software agents are convenient solution in automating replay of web applications to detect new faults. We have designed and implemented a prototype system based on a multi-agent system to facilitate the replay process. Obtained results of preliminary experiments with the prototype have shown promising success of the system.

As future work, this system can be extended to consider in detail and more precision the state of web application and time dimension of user sessions and other involved factors. We think that this work is possible by adding new responsibilities to the current broker agents or creating new broker agents.

References:

[1] Cao, J. et al, September 2002, Mailbox-based Scheme for Mobile Agent Communication, *Computer,* pp 54-60.

[2] Elbaum, S. et al, 2003, Improving Web Application Testing with User Session data. *In ICSE '03, IEEE transaction on software Society.*

[3] Elbaum, S. et al, May 2005, Leveraging User session data to Support Web Application Testing, *IEEE Transaction on Software Engineering.*

[4] Huo, Q., and Zhu, H., Sep. 2000, A Message Communication Mechanism for Mobile Agents, *In proceeding Of CACSCUK'2000, Loughbrough, UK.*

[5] Huo, Q. et al, 2003, A Multi-Agent Software Environment for Testing Web-based Applications, *In proceeding of COMPSA'03, Dallas,* PP. 210-215.

[6] Kung, D., September 30, 2004, An Agent-based Framework for Testing Web Applications, *In Proceeding of First International Workshop on Quality Assurance and Testing of Web Applications, Hong Kong, IEEE Computer Society Press.*

[7] Ricca, F. and Tonella, T., May 2001, Analysis and Testing of Web Applications. *In Proceeding of the International Conference on Software Engineering,* page 25-34.

[8] Sampath, S. et al, September 2004, A Scalable Approach to User- session based Testing of Web Applications through Concept Analysis, *In Automated Software Engineering Conference.*

[9] Singh, M.P., 1993, A Semantics for Speech Acts, *Annals of Mathematical And Artificial Intelligence 8(II)*, pp 47-71.

[10] Singh, M.P., Dec. 1998, Agent Communication Languages: *Rethinking the principles, IEEE Computer,* pp 40-47.

[11] Zhu, H. et al, September 30 2004, Developing A Software Testing Ontology in UML for A software Growth Environment of Web-based Applications, To appear in Software Evolution with UML and XML, Hongji Yang(eds.)