# CONVERTING USERS TO TESTERS: AN ALTERNATIVE APPROACH TO LOAD TEST SCRIPT CREATION, PARAMETERIZATION AND DATA CORELLATION*

*Kevin Morrison and Hisham M. Haddad*
*Computer Science Department*
*Kennesaw State University*
*Kennesaw, Georgia 30144, USA*
*kmorri69@students.kennesaw.edu*
*hhaddad@kennesaw.edu*

**ABSTRACT**

This paper proposes an architecture to significantly reduce the time, money and obscurity around "**Load Testing**," aka "**Performance Testing**," aka "**Automated Load Testing**" Oracle E-Business Suite and potentially other Java based Client/Server applications. The focus of this proposal is to identify, debate and recommend alternative methods for script recording and data correlation for purposes of load testing an Oracle E-Business Suite application instance. Alternatives are needed due to the cost and effectiveness of performing load tests. The goal for this architecture proposal is to eliminate 90% of the overhead around script identification, development and data bank design/creation. Oracle Application Testing Suite and Oracle E-Business Suite are used as the basis for this paper's testing and theoretical implementations.

## 1. INTRODUCTION

Most software engineers would like to use some automated testing in their project to ensure high quality and low defects. While it may help, the facts behind automated testing, the different methodologies and the enormous expense setting up and executing an automated test can be overwhelming. Automated testing projects often are stand-alone

projects that must be justified and executed singularly, and are not inclusive of an overall software development cycle.

### 1.1 Background

Load testing really is no different than any software development project. You plan, build, test (implement) and report. The planning process is fairly straight forward and is a smaller subset of planning, which occurs with any software development project. The build process can be costly. The script development and the Infrastructure/Application build cannot only take a lot of resources and time, it can easily cause the project to miss requirements.

### 1.2 Key Concepts and Definitions

- The notation of a Virtual User (**VU**) being emulated via remote agent executable running on one or more systems dedicated to this purpose is fundamental to load testing software. A virtual user is a thread inside of an OATS agent that runs transactions against the application being tested to simulate an actual user. The VU uses the scripts that are developed by a VU Recording IDE Tool.
- The data bank is a file or database that is used by one or more VU's for variable transaction data. For example, if the script wants to look up all orders placed by a customer, the data bank could include a list of customer numbers to be used.
- An aggregate point where all VU's receive commands and report to is an application known as the **Controller**. The Controller tells the Remote Agent how many, when and what transactions the VU's will emulate.
- Within the Controller, an administrator creates a **Scenario**. This scenario identifies how many scripts (think business transactions), how many users should run those scripts and how long they should run those scripts. This scenario needs to emulate what and how many business transactions will run on the system during production.

## 2. E-BUSINESS SUITE LOAD TESTING ARCHITECTURE

The Oracle E-Business Suite (EBS) is a collection of modules that can serve as a company's core transactional system. The EBS applications can be used for operations like financials, manufacturing, distribution and HR. Performing a load test on E-Business Suite always has the following components:

- Load Testing Software Suite
- Application Server
- Oracle Database
- Optional Components
  a. Content switch or Hardware based load balancer for network traffic
  b. Monitoring software/hardware for any given technology stack/component

### 2.1 Typical Oracle E-Business Suite Load Testing Architecture

Figure-1 presents the different layers of a typical Oracle E-Business Suite Load Testing Architecture. The Load Testing Infrastructure and E-Business Suit layers in the architecture are represented logically; Physical segmentation differs between implementations. The Network Load Balancer (e.g., Content Switch) for session aggregation and balancing is a common layer, but optional.

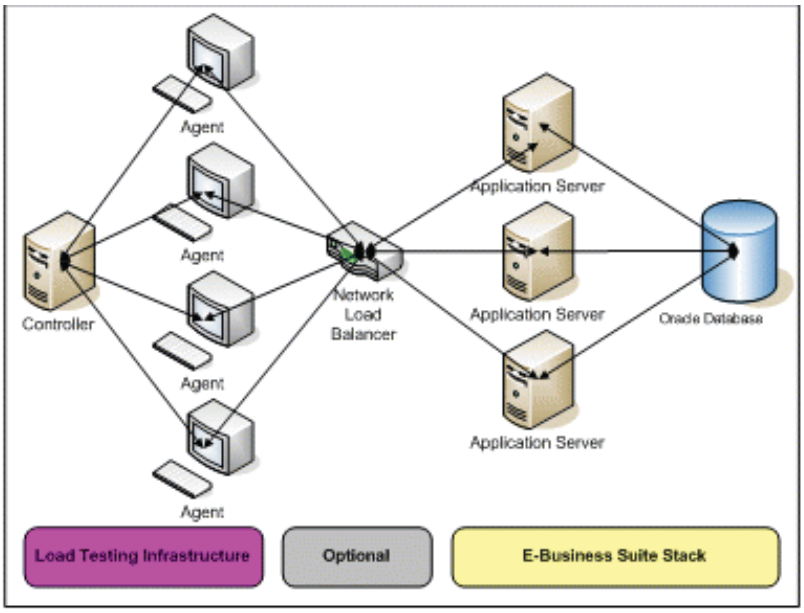


**Figure-1:** Layers of a typical Oracle E-Business Suite Load Testing Architecture.

### 2.2 Load Testing Software

There are several commercial and open-source software testing tools available. The most common load and functional testing tool is HP Quality Center (formally Mercury Interactive). Oracle Application Testing Suite (OATS), (formally Empirix Software) is probably the second most popular package. These two products do very similar things and are capable of producing similar results. The major difference between them is the scripting language: HP has a proprietary development language and Oracle uses Java. OATS can be used to generate thousands of virtual users to simulate actual performance of an application in production. The tool includes script writing capabilities, performance metric gathering capabilities, a Controller to create and run the testing scenario and agents that run the virtual users.

## 3. LOAD TESTING SCRIPTS

A load testing script is truly just that: a script that is interpreted by a runtime compiler, within the load testing software. The script contains commands that simulate what a real user would perform within the application. For example, within Oracle E-Business Suite, a user may enter an expense report. To create a script to simulate that user, the script developer will open client software that allows the developer to "record"

the transaction. The recorded transaction can be played back against an application. OATS has a development suite called OpenScript.

## 4. PROPOSED SOLUTIONS FOR 90% REDUCTION IN SCRIPT DEVELOPMENT EFFORT

Elimination of developers "recording" scripts, developing data that can be used variably in the scripts and the combination of those scripts in a scenario can only be completed by truly recording transactions that users are running in a production system. Only then could the variability and repeatability of users' transactions on a business application be captured and reproduced. Capturing the users' commands could reproduce the exact transactions running at a single point in time. This would require a "**Recording Agent**." Everything is inspected coming through a queue/pipe, but the Recording Agent can replay the inspection. There are multiple layers within the architecture the agent could record:

- Client/User
- Network Load Balancer
- Application
- Database

### 4.1 Potential Recording Agent Implementations

The Recording Agent can be implemented within one or more technology stacks. Just like an OSI model, the technology stack for Testing can be viewed vertically. The major three tiers are: 1) Client, 2) Application and 3) Database.

#### 4.1.1 Client Tier

Within a business enabling packaged application, the client stack is always the end user device. It always has the largest number of devices and is typically more difficult to control and administer. With respect to Oracle E-Business Suite specifically, the client always has a browser for initial application access and a Java JVM, used to instantiate an Oracle Forms session. The client will always have a specific version of browser and JVM resources, meaning there is a consistent development framework installed.

#### 4.1.2 Application Tier

The application tier within Oracle E-Business Suite is comprised of a Web Server and a Forms Server. The Web Server is Apache and runs a website like any other WWW-based application. The Forms Server is a Java application that serves clients on a specific port with information for Forms Sessions. The application code/logic is installed and runs within this stack. This layer is responsible for communicating with devices in both the clients and the database stack.

### 4.1.3 Database Tier

The database stack is just that - the database. There can be one or more servers providing database services. This will typically contain the least amount of devices in all three stacks, meaning that transaction traffic is more consolidated and contained.

Figure-2 visually represents potential Recording Agent implementation tiers. One of the three tiers must be selected to develop and architect a software solution for transaction recordings. The Recording Agent will need to be developed specifically to trap and record the business transactions in one of these tiers. Based upon the analysis, the client tier is most favorable for implementing the Recording Agent. The application tier is also a possible solution, but some of the transactional data will be lost because some of the logic has already occurred on the end user device.

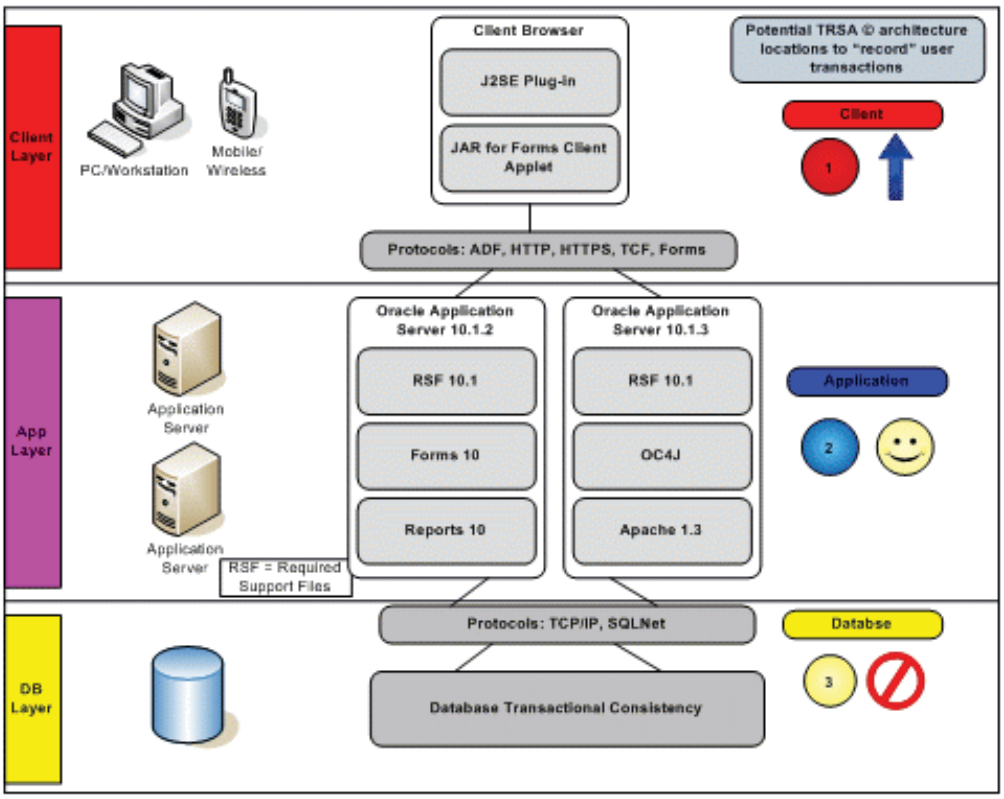


**Figure-2:** Potential Recording Agent Implementation Areas

### 4.2 Recording Agent Stack Location Feasibility Summary

Figure-3 shows a summary of the pros and cons of implementing the Recording Agent within each of the three stacks. The most difficult to manage - but most granular area within the stack to record - is the Client. The client is where all information is sent or received, and where the information is the most raw and precise. The application layer receives information from/to the client and from/to the database, but that information cannot be recorded exactly as the client transaction was generated.

| Legend | Architecture Tier | Pro's | Con's |
|---|---|---|---|
|  | Client | Recording of the user transaction will be per individual and precise | Aggregation of the data from the client to the "controller" will be network and CPU intensive |
|  | Application | Application servers can handle hundreds of users. Data is heavily aggregated. | Identifying the individual user transactions will require a lot of coding and aggregating the data to be used for correlation will be difficult. Certain client specific traffic is lost. |
|  | Database | Most heavily data aggregated point in the architecture | Business logic and client interaction will be lost, as only the SQL and PL/SQL will be recorded at the database layer |

**Figure-3:** Detailed Pro and Con List for Recording Agent Implementation Areas

### 4.3 Client Specific Recordings

All automated test recording occurs at the Client layer. Tools such as HP LoadRunner and Oracle Application Testing Suite all use client-based development tools for script recordings. Our goal is for the Recording Agent to create the ability for client-based recordings, and generate scripts for replay with HTML and Java protocols. This will ensure the maximum amount of information about the transaction will be recorded.

## 5. DISCUSSION

### 5.1 Parameterization: Why you need it and how to eliminate it

The term "parameterization" is a fundamental requirement in performance testing. Parameterization of a load script is performed when a value entered into an application (for example, part number) and the value entered by the user is replaced by a variable. That variable can be used to substitute additional values versus using the same value in the script over and over. This is needed for two reasons:

- It is important to perform varying inputs and queries in the load test to ensure a particular type of data value will not cause issues
- Applications and databases can cache results. When the same value is used, the results will be cached in memory, or on disk, and true performance testing will not be observed

Data correlation is an advanced parameterization activity where the virtual users' input for one field is determined by the value from the previous field. For example, when looking up the price of a vendor's part number, the user requires the number and part number. A part number may not be unique across all vendors so if a vendor number is selected for application input via parameterization, and a part number should be selected that the vendor uses. Data correlation is the concept of ensuring that the data is properly "paired" and that the data entered by the virtual user is meaningful.

Eliminating Parameterization and Data Correlation is impactful and delivers a large amount of savings. Parameterization can be eliminated by recording actual user transactions with a program like the Recording Agent. By using the actual data values the users entered, the performance test is 100% precise. Some statistical random sampling rules should be followed to determine the proper amount of users to record. Additional analysis should also be performed to determine the correct amount of time to record each user. The number of users and recording times will vary depending upon the size of the application user base, the size/configuration of the Infrastructure and the desire to duplicate the exact load accurately.

### 5.2 Client Recording Agent E-Business Suite Implementation

A Recording Agent is nothing more than an add-on to a browser. The add-on would need to launch at the time of application launch and should have an automated execution mechanism for actions such as recording, stop recording, etc. The automated execution could be embedded within the code (HTML, scripting or other) sent to the client browser for run-time execution. Because Oracle E-Business Suite has an HTML and Java (Oracle Forms) based component, the Recording Agent HTML client would need to instantiate a Java based Recording Agent when Java Forms is launched. Figure-4 illustrates the Recording Agent being implemented as sub-processes within the Java and HTML engines (JVM and Browser respectively). The two Recording Agent process will need to communicate with one another and the Script Recording Library, where the scripts will be saved. Both of the Recording Agents can be written in Java to help with the JVM communications, but the browser hook for Recording Agent launch will need to be developed based upon specific browser requirements.

### 5.3 Recording Agent Script Playback

The Recording Agent must record the user transactions in a format that can easily be replayed by a load-testing tool such as OATS. The Recording Agent will need to be intelligent enough to record scripts in native Java and format them in such a way that OATS can replay them. Alternatively, the Recording Agent could record the transactions in the proprietary HP LoadRunner format, and an interpreter could convert the output into different formats.

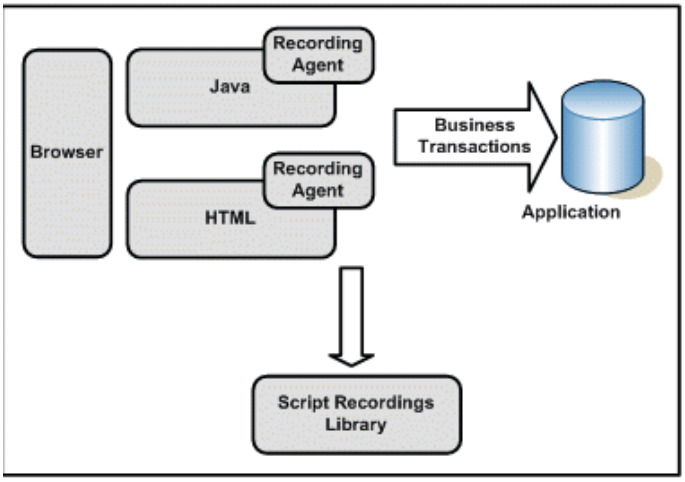


**Figure-4** Architectural view of Recording Agent

## 6. CONCLUSION

The phases used in planning to develop a load test involves considering the Goals, Users, Use Cases, Production Environment, Production Environment and Scenarios, according to Portata [3]. Implementing the Recording Agent eliminates three out of the six phases (Users, Use Cases and Scenarios). These three phases are heavily dependant on the business users for helping to document proper Use Cases and Scenarios. The Development phase of a load testing project would be completely eliminated with the proposed Recording Agent. Activities such as Business User Manual Recording, Parameterization, Data Correlation and Scenario builds are completely eliminated.

Based on the project planning methodology for load testing from PerfTestPlus, Inc. [4], the Recording Agent would eliminate requirements for Determining User Activities, Determine Usage Model, Develop Test Data, Develop Test Scripts and Baseline Exploratory / Planned Scripts. Recording Agent eliminates five out of the first thirteen activities from the Project Plan.

Implementing an agent which is responsible for recording the action of a user within their browser and Java Forms session enables a 90% reduction in effort around script identification, development and data bank design / creation. Additional time savings is gained in the Planning Phase as half of the activities are eliminated. . Implementing the Recording Agent significantly reduces the effort to perform performance / load tests on Oracle E-Business Suite by:

- Eliminating the developer IDE and the effort around recording virtual user scripts
- Eliminating the work around data parameterization
- Eliminating the work around data correlation
- Eliminating the Use Cases and Scenario definition and development

## REFERENCES

[1] Oracle, Oracle E-Business Suite Concepts, 2010, http://docs.oracle.com/cd/E18727_01/doc.121/e12841/T120505T120508.htm, retrieved February 2012.

[2] ACM, Finding Usability Bugs with Automated Tests, 2011, http://queue.acm.org/detail.cfm?id=1925091, retrieved February 2010.

[3] Portata, Load Test Planning Process, 2012, http://www.portata.com/wp/ltpp, retrieved April, 2012.

[4] PerfTestPlus, Inc., Performance Testing Uncovered, 2006, http://www.perftestplus.com/resources/perf_uncovered_ppt.pdf, retrieved March, 2012.