

January 7, 2005

Company: iTKO

LISA smiles on J2EE app testers

iTKO's regression and load testing tool makes steep learning curve worth the effort

By **Rick Grehan**

January 07, 2005

Capable of prying into just about any nook or cranny where one might apply J2EE technology, Interactive TKO's (iTKO) LISA (Load-bearing Internet-based Simulator Application) is a QA engineer's dream.

This pure-Java tool for regression- and load-testing J2EE components runs anywhere you can plant and execute a JDK 1.3.1+ environment, shaking down individual classes, EJB, Web services, Web pages, databases, and more to find their weaknesses.

Although I tested LISA with **JBoss** ([Profile](#), [Products](#), [Articles](#)), its documentation claims it can test J2EE applications running on any J2EE container or Web server backed by any JDBC-compliant database. That wide-ranging coverage, combined with its solid performance and versatility, means LISA just might supersede the QA arena and have a few developers seeing visions of grandeur, too.

LISA's Structure

LISA's documentation touts the tool as a "no-code" solution. This is largely true: You can create and execute batteries of tests by promenading through LISA's crowd of wizards.

Given that it's a Java testing tool, however, you won't get far without employing some Java — if only to identify which methods need exercising, what their input arguments should be, and what the methods' outputs must be. Nevertheless, LISA dispatches much of your test-creation work via its numerous, dense dialogs.

LISA is a simulator tool, as noted in its acronym, and the product executes its simulation duties well. For example, when testing a hypothetical user's access to a Web page, LISA allows you to add "think time" to the activity, thereby mimicking the time delay a user will exhibit while scanning the page before actually entering data into any controls.

In addition, LISA simulates multiple users calling into the same Web site and applies the effect of "think time" across all the user simulations so that you can adjust the tests to create a more realistic usage profile.

Projects in the LISA environment are organized into "test cases." A test case is a collection of nodes, in which each node typically defines an individual test action within the test case.

LISA recognizes more than a dozen different node types, each geared for testing a specific component type. For example, you can build a Web page test by creating an HTTP/HTML Request node. After you specify the URL you want to call, the host name and port, and the Get and Post parameters you want included, LISA will send the request "live" to the Web server, allowing you to examine the response.

If the response is as you like it, you set that as the response that LISA should expect when the test is run noninteractively. Once you've established the correct result of a test, you "teach" LISA the correct response by setting an "assertion." Typically, an assertion will point to a target node; if LISA finds that the response is not as expected, it will transfer control to that target. Usually, the target node's job is to post an error message in a log, but it can also halt the testing completely, depending on your preference.

LISA's steep learning curve is not so much the fault of the product as a reflection of the fact that J2EE applications are composed of lots of dissimilar entities.

Because of that fact, creating a node to test a database is very different from creating a node to test an EJB, which is very different from creating a node to test an HTML page, and so on. The wizards help a great deal, but a new user will have to invest considerable time exploring all the options available.

Still, LISA reduces the complexity of an otherwise knotty undertaking to something manageable. For example, setting up an EJB test requires little more than dropping the EJB JAR file in LISA's hotdeploy directory. LISA will unpack the contents of that EJB so that the appropriate wizard counsels you on the EJB's available methods, their input parameters, and their return types.

After everything is unpacked and laid out, you select the test inputs, and the test node is more or less built for you. You can even run the specific test interactively so that you can tell LISA what the test's proper outcome should be.

A Price Worth Paying

LISA's pricing structure incorporates a discounting schedule based on team size (those who actually create tests with LISA) and "virtual users" (those who run the tests). John Michelsen, principal and CEO of iTKO, estimates that "a small team will cost about \$20,000 to start." In my estimate, a small team has four members, so that's about \$5,000 per member — a bit steep, but worth the climb.

Automated testing systems for J2EE continue to improve, which bodes well for developers anxious to stomp every airpocket of error out of increasingly elaborate applications. LISA is the best-looking J2EE tester I've seen yet.