Automated Testing

OVERVIEW
As experienced mainframe personnel retire, next-generation developers must have access to tools and capabilities that enable them to manage mainframe programs with the same level of confidence and quality as other languages. **Topaz® for Total Test**, Compuware’s automated testing tool, enables developers and testers at all skill levels to test programs and subprograms almost immediately after updating a piece of code to ensure unintended changes weren’t introduced into the codebase.

UNIT AND FUNCTIONAL TESTING IN ONE SOLUTION
Topaz for Total Test automates the creation of unit tests saving developers valuable time that would normally be spent manually creating tests and test assets. Topaz for Total Test also automates the execution of unit and functional tests within the Jenkins automated build-test-deploy process.

- **Unit testing** enables small portions of a program to be tested, as many times as necessary, to immediately ensure the program functions as it should prior to performing more complex, time consuming and costly tests. Unit tests work to isolate the program by virtualizing external calls and data access and making assertions on internal states of the program.

- **Functional testing** ensures programs run successfully on live systems with live data just as they would in production. Functional tests verify that the external behavior of a program meets business requirements without any visibility into the code of the program.

By creating and combining unit and functional tests of a program or a group of programs, developers will ensure they have a regression test suite that covers all foreseeable failure scenarios.

EASILY CREATE UNIT TESTS
A developer can automatically create a unit test for a program in just a few steps from within a **Compuware Xpediter** debugging session. A right click at the breakpoint in the source code automatically generates a unit test and collects the necessary test assets—including program stubs and test data from Db2, IMS, CICS and datasets—to help execute the test. Test assets are saved so they can be used repeatedly in later testing. Topaz for Total Test virtualizes (stubs out) program-to-data and program-to-subprogram calls and returns data for only what is programmed into the test.

By verifying that programs consistently return the same data before and after code changes, Topaz for Total Test helps facilitate regression testing, enabling developers to determine if any bugs were inadvertently introduced when code changes were made.

TEST ASSERTIONS
Default test result assertions enable comparisons between the expected results the developer or tester defines in the test and the actual results returned by the program. After the test executes, test assertions are evaluated to determine if they pass or fail. Test assertions not only provide test results but also include the reason for the failure, greatly speeding test failure resolution.
Program Entry and Write Assertions

For unit tests, program entry assertions ensure that the parameters passed into a subprogram are correct. They are similar to write assertions, which confirm the data that is to be written to VSAM or QSAM files, to Db2 tables, IMS segments and CICS commands, are correct. Db2 SQL statement write assertions can be used to verify data in UPDATE, INSERT or DELETE statements. IMS write assertions include ISRT, REPL, DLET. Write assertions support many CICS commands that move data, for example WRITEQ. The expected values for all of these assertions are automatically captured during program execution when the unit tests are created.

TEST EXECUTION FLEXIBILITY

Developers can execute tests on different LPARs based upon available test environments. If a developer is using test assets and wants to execute the test on another LPAR, the test assets are moved to the other LPAR as part of the test case for execution. Alternatively, you can also choose from a simple list of LPARs with test runners. Simply pick the LPAR and run the test.

TEST DATA VIRTUALIZATION

Unit Testing

Test data virtualization simplifies testing by eliminating dependencies on submodules, Db2, IMS and CICS on the mainframe. With test data virtualization, data stubs are used to virtualize (stub out) live data and programs on the mainframe. These test assets move with the unit tests, so tests run independently of live data, eliminating the need to establish live data separately. Virtualized data can be edited to drive specific test cases with stubbing support available for Db2 SQL statements, IMS DL/I commands and CICS commands in addition to VSAM and QSAM data files. Virtualized data (data stubs) is created automatically and does not require re-compiling, making it far easier for developers to create repeatable tests.

The subprogram testing feature of Topaz for Total Test enables developers to test individual mainframe subprograms without requiring a call from the main program. Likewise, the external stored procedure testing feature enables developers to test individual stored procedures without a call from the program.

Topaz for Total Test also allows a subprogram or external stored procedure to be called directly using parameters collected by Xpediter. Parameters can be easily edited to create additional subprogram/ stored procedure test cases. Users can automate test setup and simplify external stored procedure testing by virtualizing Db2 calls, eliminating the need to have a Db2 subsystem.

IMS Batch Message Processing (BMP), Message Processing Programs (MPP) and Batch DL/I stubbing support allows developers to test an IMS program without an IMS system. Because the data has been virtualized in the program, an IMS database isn’t required.

Additionally, Topaz for Total Test allows developers and testers to virtualize live CICS data so CICS tests are isolated from actual CICS regions. CICS tests run as batch programs, which means there is a lot of flexibility in where the test can be executed.

Functional Testing

Developers use a test scenario to execute functional tests for live program testing. Test scenario elements can also be used to create a sequence of steps (e.g. set up data in datasets and Db2) before testing programs; executing live programs by providing input data and verifying output data; and finally verifying data by the program in Db2 and datasets.
Test scenarios containing unit and functional tests can be combined in a test suite and automatically executed by CI/CD pipelines like Jenkins. Test reports are kept in a shared repository where detailed information and history statistics can be found.

**DEVOPS TOOLCHAIN INTEGRATIONS**

You can easily integrate test assets into a DevOps toolchain to enable unit and functional testing within the Jenkins automated build-test-deploy process. A Jenkins plugin facilitates seamless configuration of test execution with Jenkins workflow for enterprise DevOps efficiency.

Test code coverage results can be automatically fed into SonarSource SonarQube for analysis, enabling IT teams to understand the scope and effectiveness of their testing as code is promoted towards production. (Code coverage metrics provide insight into which lines of code have or have not been executed, and what percentage of an application has or has not been tested.) Using SonarQube, developers can also see test results along with code quality trends across multiple platforms throughout the DevOps process.

A Jenkins plugin automatically generates Topaz for Total Test pass/fail results into a format SonarQube can display, enabling DevOps teams to see how testing progresses after every build. SonarQube Quality Gates on test results can be used to determine whether to proceed with a workflow or stop and fix the code before continuing.

**THE VALUE OF TOPAZ FOR TOTAL TEST**

- Combines unit and functional testing into one easy-to-use tool.
- Uses data virtualization to simplify test data management.
- Can be easily integrated into a DevOps toolchain to automate unit and functional testing within the Jenkins automated build-test-deploy process.
- Reduces dependency on the specialized knowledge of mainframe veterans by empowering novice developers to immediately validate mainframe code changes with the same speed and confidence as other code.
- Provides developers with the confidence to make large mainframe code changes, knowing their codebase can be tested to maintain good quality.
- Empowers teams to put code changes into production more quickly to better respond to changing business needs.
- Ensures continuous code quality management across platforms through an integration with SonarQube.

Learn more at compuware.com/topaz-for-total-test.

For more details on the full Topaz suite, visit compuware.com/topaz.