SHIFTING FROM MANUAL TO AUTOMATED TESTING ON THE MAINFRAME

Overview: Traditional manual testing methods will hold you back in the digital age. Learn what challenges to this Topaz for Total Test can help you overcome through automation.

Many mainframe teams claim they have good reasons for keeping things how they are: Massive COBOL programs are complex and poorly documented, and experts are retiring at an unprecedented rate, so don't touch anything. And if you're going to make changes—always with a "green screen"—make sure you stick to the processes in place.

But it's these tools and processes that must be replaced with modern alternatives if mainframe-powered firms have any hope of keeping up with agile disruptors in a digital age. Change is necessary. Fear of failure is the bane of innovation and the foundation for maintaining a status quo that holds you back from excellence.

Manual unit testing of mainframe code is a glaring example of a process that must change, and one that requires a modern automation tool to do it.

The Need for Automated Testing

In a blog post about automated unit testing in Flow Agile, Jason Bloomberg, President of Intellyx,

writes, “Team members should be focusing on delivering business value, not on the intricacies of manual testing, or the complexities of testing across environments. Everybody knows that testing is essential, but it certainly doesn't have to be manual.”

Manually creating and executing tests is labor-intensive and inefficient. It slows down the development process, which is why many developers forego a process like unit testing or “finish” before completing it. The result is lower-quality code.

As the mainframe continues to support growing customer-facing innovation, there's an urgency to increase unit testing, and it can't happen manually. It requires automating unit test creation and execution.

Compuware Topaz for Total Test helps you achieve mainframe DevOps success by:

- Automating the creation and execution of mainframe unit tests
- Validating code changes immediately
- Maintaining the quality of the codebase
- Leveraging integrations with leading cross-platform DevOps and Compuware tools for continuous code quality management and Continuous Integration/Continuous Delivery

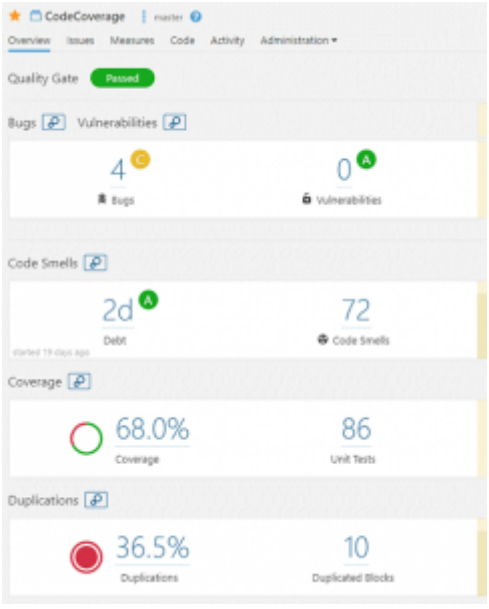
Here, we'll dive into some of the key problems with unit testing today and look at how Topaz for Total Test helps you overcome them.

Key Problems with Unit Testing Today

Through Topaz for Total Test, we've introduced an innovative automated testing approach that helps mainframe teams accelerate development cycles and rest assured code changes won't negatively impact another part of a program. The following are some of the big problems we've designed Topaz for Total Test to solve.

Limited Visibility into COBOL Programs

With Topaz for Total Test, you can verify when data is written to QSAM, VSAM, Db2 or IMS. Generally, the program calculates some type of update to be written to a report in a QSAM or VSAM file or updates a Db2 or IMS Database. The ability to verify this data at the field level provides great visibility into the program. If the expected values are not being written into these items, you know that something changed in the program, since the data used by the program hasn't changed because virtualized data (also known as data stubs) is used. The last step is to verify if the program is incorrect or if the expected value is incorrect; in either case, an update is required.



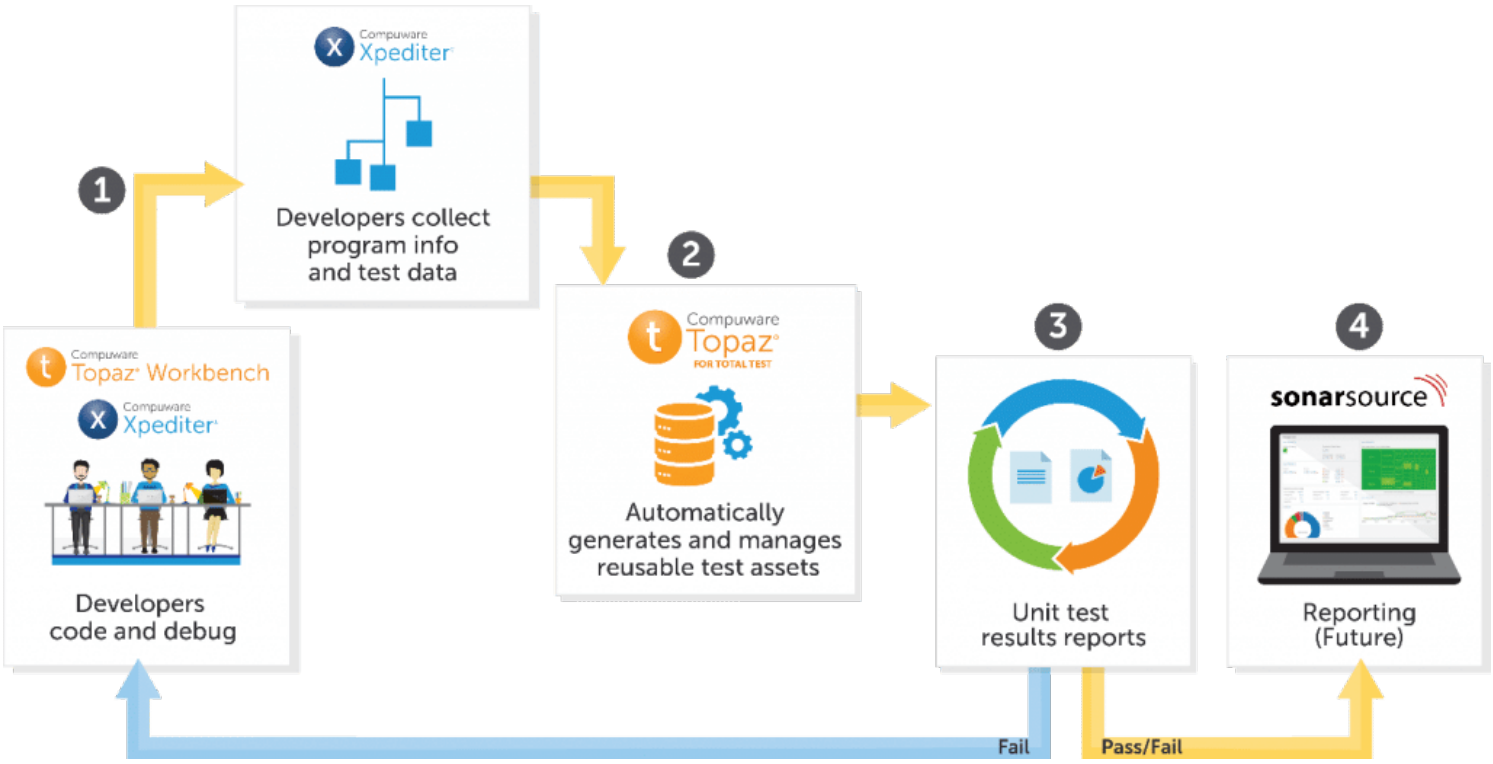
Code coverage results can be fed into SonarSource SonarQube to help you understand the scope and effectiveness of your testing as code is promoted toward production. SonarQube dashboards

display unit testing pass/fail and code coverage results as well as other quality trends. Quality Gates can be defined to determine whether to proceed with the workflow or to fix the code before continuing based on the unit test results.

Time-consuming Test Creation

Topaz for Total Test simplifies test setup and execution on various test systems by automatically generating unit tests and virtualizing program calls and test data access.

From within a Compuware Xpediter debugging session, you can automatically create a unit test in Topaz for Total Test. Necessary test assets, such as virtualized test data and program calls, are generated to help execute the test and can be saved for use in later testing.



Inconsistent Data for Unit Tests

Instead of requiring you to manually create data, Topaz for Total Test automatically virtualizes Db2 data as well as IMS Batch Message Processing (BMP)/Message Processing Programs (MPP), Batch DL/I, VSAM and QSAM data files, making it easier to execute repeatable tests.

Virtualized program calls eliminate the need to call an actual subprogram or Db2 stored procedure. For IMS BMP/MPP and Batch DL/I, data virtualization also eliminates the need for an IMS system when testing an IMS program, and it eliminates the need to have an IMS database.

Virtualized data moves with unit tests, so tests run independently of data files, meaning you don't have to move data files or tables separately to your test system. The virtualized data can be edited to drive specific test cases.

Difficult Sharing and Non-portability of Tests

Because the virtualized data moves with unit tests, developers can easily execute unit tests across various systems by choosing from a list of target systems with test runners installed. When you

choose a target system, the unit tests and necessary test assets are automatically transferred.

Unavailable Test Subsystems

When running a Topaz for Total Test case, there's no need for a Db2 subsystem. You can call external stored procedures independently of the main program and other stored procedures. Virtualizing Db2 SQL calls simplifies stored procedure testing.