

**InDetail** Paper by Bloor

Authors **Daniel Howard and Philip Howard**

Publish date **February 2017**

---

## **Compuware Topaz for Total Test**

“

**Unit testing is ubiquitous when working with relatively modern programming languages such as Java and C#. Compuware's Topaz for Total Test brings comparable capabilities to the mainframe environment and the COBOL that runs on it.**

”

Authors  
**Daniel Howard  
and Philip Howard**

# Executive summary

**M**ainframes have been around for as long as businesses have been using computers. Although in terms of technology they are often perceived to have been superseded by distributed systems, the fact of the matter is that many of the world's largest companies (at the last count: around 4,500 of them) rely on mainframes and the legacy software that runs on them. While the death of the mainframe has been predicted for more than twenty years, the truth is that so much mission critical software is mainframe dependent that the mainframe is unlikely to ever truly go out of fashion. What's more, although there are several good choices of programming language for writing new software on a mainframe, the reality is that the vast majority of software already written for the mainframe is in COBOL. As a result, COBOL is here to stay – in the same way that mainframes are – because there is so much legacy software written in it.

This poses a problem. It's not because of the language itself. COBOL is extremely performant and not a difficult language to learn. But, legacy COBOL code – like most legacy code – needs to be maintained and kept up-to-date. But – again, like most legacy code – even if it is not outright poorly written by modern standards, it is often monolithic, complex, undocumented and untested. This makes it exceptionally hard to understand the code and makes it even harder to change it without breaking something unexpectedly. Given particularly that mainframe programs tend to be the oldest and most important parts of the business, this is a very big deal.

The solution is to build confidence in the code, and the way to do that is via unit testing. Unit testing is the practice of examining small pieces of code functionality – units – separately from the rest of your codebase and subjecting them to rigorous, fast, automatic, and repeatable tests, usually via a unit testing framework. Such frameworks typically automate unit testing processes and support agile development methodologies. By using a framework,

whenever a developer makes changes to the code, they can run the unit tests attached to the program they've changed and make sure nothing is broken before their code either goes live or is submitted for further testing. Unit testing will not find every issue and it does not make higher levels of testing obsolete, but it massively improves the efficiency of such tests by ensuring that the submitted code is of consistently high quality.

Unit testing is ubiquitous when working with relatively modern programming languages such as Java (using the JUnit unit testing framework) and C#. Compuware's Topaz for Total Test brings comparable capabilities to the mainframe environment and the COBOL that runs on it.

## Fast facts

Compuware's Topaz for Total Test is a new addition to Compuware's Topaz suite of products – tools meant to help modernise mainframe development – and facilitates unit testing for COBOL. In particular, it can automatically generate unit tests for existing COBOL programs, complete with attached input data and check conditions.

## Key findings

In the opinion of Bloor Research, the following represent the key features of Topaz for Total Test:

- It automatically generates unit tests, including input data and check conditions (test assertions), based on how your program currently works. This means that you can very quickly create tests for a functional program and be sure that any future changes to that program do not break the core functionality of your application.
- It doesn't require any changes to your COBOL code in order to work. Combined with the automatic generation of tests and associated data, this means that it is feasible to generate unit tests for a much larger number of COBOL programs than could reasonably have tests written for them manually.



**So much mission critical software is mainframe dependent that the mainframe is unlikely to ever truly go out of fashion.**





Unlike JUnit, in which only the test execution is automated, Topaz for Total Test automates both test creation and test execution.



- When a program is updated, it is easy to validate those updates by editing the test cases and test case data.
- It can test subprograms in isolation from their main program, and in addition can stub out subprograms when testing a main program. This means that each piece of functionality in the code can be encapsulated and tested individually. The stubbing process is automated and does not require manual intervention: it is a major differentiator for Compuware compared to its competition.
- While the product does not currently offer features with respect to coverage (what percentage of paths through the code have been tested?) we are assured that this will be in a near-future release of the product.
- No invasive code changes are required.
- Integration with tools such as Jenkins enables continuous deployment and testing, and supports agile development methodologies.

### The bottom line

Topaz for Total Test can be likened to “JUnit for COBOL”, but this does not really do it justice. Unlike JUnit, in which only the test execution is automated, Topaz for Total Test automates both test creation and test execution. Compuware has delivered a product that automates a large amount of the unit testing lifecycle and allows you to enjoy a correspondingly huge gain in efficiency. Although Topaz for Total Test can technically be used on greenfield projects (for example, supporting test-driven development), it has been designed specially to support brownfield projects and, specifically, it leverages the existing codebase as a part of its automation process. As a result, the product loses a large amount of functionality if used for greenfield COBOL projects. However, new COBOL projects are few and far between – if they exist at all – so this is not much of a downside.

The big advantage that Compuware has brought to the table with Topaz for Total Test is that there is significantly more automation in this product than in other tools (whether proprietary or open source) targeted at the same market. This means that working with Topaz for Total Test is going to create an environment that enables the maintenance and development of legacy code in a way that is more productive and efficient.

# The product

**T**opaz for Total Test, released in January 2017, is a unit testing framework for COBOL that runs as an extension of Compuware's Topaz Workbench, and it continues Compuware's efforts to bring agile methodologies to the mainframe.

The general way in which Topaz for Total Test is illustrated in **Figure 1**.

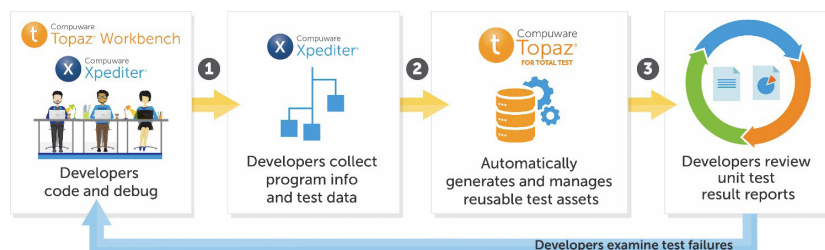
Topaz for Total Test will automatically generate unit tests – complete with input data and check conditions (also known as test assertions) – based on the current state of your COBOL program. In other words, it will assume that your programs are working correctly now, and will create unit tests to that effect. In this way, you can be sure that any further changes you make to your codebase do not break the core functionality of your application. In addition, the tests and associated data are suitably easy to edit if changes to the core functionality do need to be made. As an aside, this is why the product loses functionality in greenfield environments: because there is no existing code to introspect and therefore you cannot automatically capture test assertions.

The process begins in Xpediter, Compuware's mainframe application debugging tool. Once you have opened a suitable debug configuration in Xpediter and navigated to the program you'd like to test, generating a unit test is as simple as selecting the appropriate option from the right-click menu. This is shown in **Figure 2**.

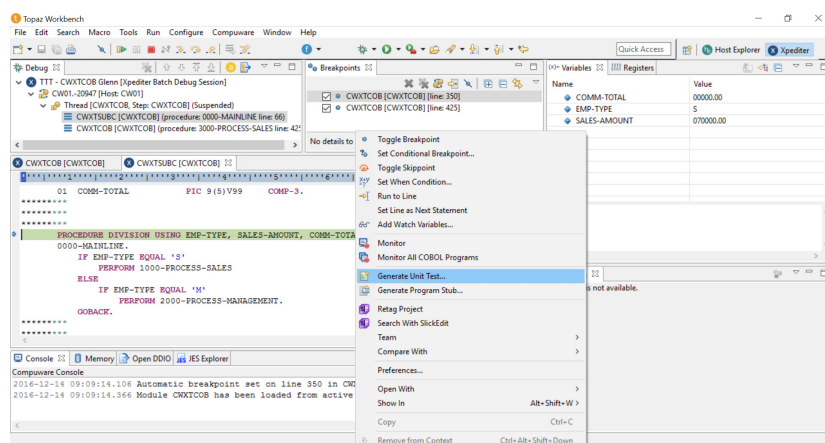
Once the unit test options have been selected, the only step that's left is to run the debugger on the program in question. Xpediter will automatically capture input data and check conditions from the debugger and feed them through to Topaz for Total Test. Note that at no point have changes been made to the COBOL code itself. Moving on to the Project Explorer tab in Topaz Workbench, you'll be able to see that a new test project has been created for your unit test – alternately, your unit test has been added to an existing project – and it has been automatically populated with input data and check conditions. A generated unit test can be seen in **Figure 3**.

It's important to understand how the automatically generated input data and check conditions have come about: they have been captured based on the current inputs and outputs of your program.

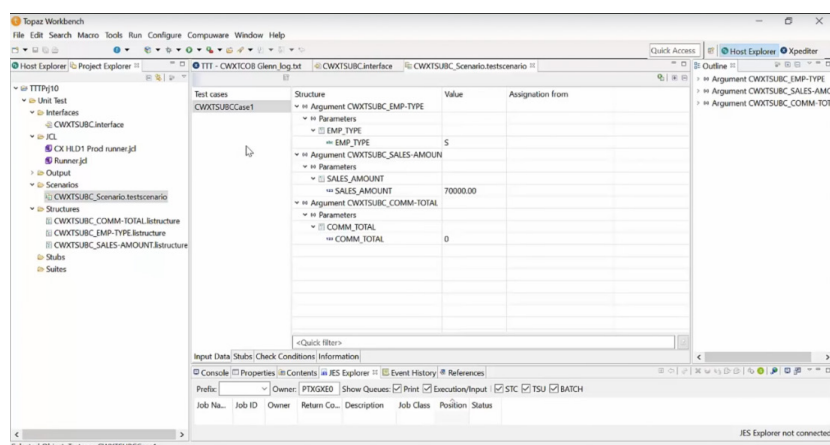
**Figure 1 – Process flow from Xpediter to Topaz for Total Test**



**Figure 2 – Generating a unit test in Xpediter**

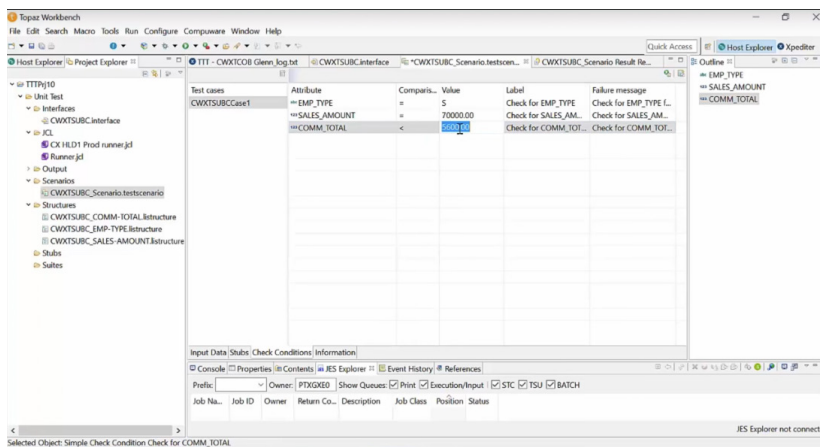


**Figure 3 – A test scenario with attached input data in Topaz for Total Test**



If your program is already broken, the tests will check that it remains broken! For this reason, or otherwise, the input data and check conditions can be easily changed, exactly as you might expect, by clicking on them and, depending on the field, either selecting a new value from a dropdown list or by entering it manually. This can be seen in **Figure 4**.

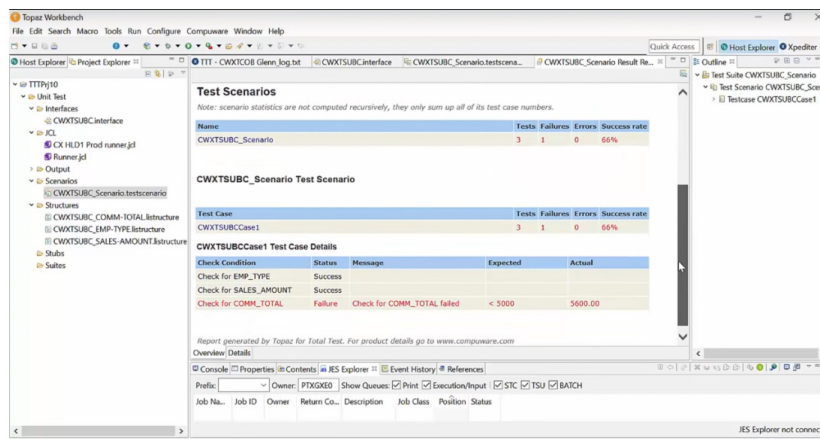
**Figure 4 – Editing a check condition in Topaz for Total Test**



The final step is to actually run your tests. This can be done manually with Topaz for Total Test or continuously by integrating with continuous testing tools such as Jenkins. In either case, Topaz for Total Test will proceed to upload tests to the mainframe, run them, then return a results report displaying exactly which tests passed and which tests failed. In the latter case, additional information is provided, telling you which check condition failed, what the value in question was expected to be, and what the value really was. This is all shown in **Figure 5**.

There is some advanced functionality available, as well as some more subtle points to be made. When generating unit tests, you have a few different options for encapsulation: that is, isolating the specific program you want to test from its dependencies in the rest of your application. To begin with, it's not just programs you can generate unit tests for: you can also generate unit tests for subprograms. Correspondingly, when you generate tests for a program, you can choose to stub out any subprograms it contains. What this means is that, when the program is tested and it tries to call a subprogram, Topaz for Total Test will intercept that call and return an appropriate value (as before, this will be set automatically but can be changed manually). This means that the subprogram is never actually run during the testing of the parent program. This prevents situations where, for instance, a test for a program might fail because a subprogram is faulty. Although this might seem fine, from a debugging perspective it is much easier to pinpoint where an error is occurring if the program and subprogram have separate tests, particularly since one program could have many subprograms.

**Figure 5 – A failing test in Topaz for Total Test**



In a similar way, you can also stub out file reads or writes (and in the former case, you can additionally set a maximum number of file reads). This makes the tests in question much more efficient – read and write operations are often very slow compared to the speed unit tests normally run at – and allow you to more finely control your input and output data. They would normally also result in your mainframe being cluttered up by extraneous testing data but, as it happens, Topaz for Total Test takes care of that for you: any temporary files which are created by Topaz for Total Test are cleaned up automatically. In addition, Topaz for Total Test also supports continuous testing (as mentioned above) and deployment, integrating with tools such as Jenkins.

It is further worth noting that both program stubs (for subprograms) and data stubs (for file reads and writes) are created automatically: they do not require manual intervention and, unlike some competitive products, data stubs do not require re-compilation.

Finally, it is appropriate to comment on future plans. Topaz for Total Test is Compuware's ninth successive quarterly release since its original introduction in January 2015. A near-future release will bring code coverage into Topaz for Total Test. This is an important feature, and although its present absence is unfortunate, it is good to know that it will be released relatively soon. In addition, although Topaz for Total Test is currently a unit testing product, Compuware's vision extends beyond just unit tests. Ultimately, it sees Topaz for Total Test as a product that it can build on to deliver higher levels of testing to the mainframe. In the company's opinion – and we agree with them – higher levels of testing substantially benefit by starting at the bottom, with a solid foundation of unit tests, to dramatically increase code quality and improve the efficiency of subsequent testing.



**Topaz for Total Test also supports continuous testing and deployment.**



# The vendor

**C**ompuware was established in 1973 and floated on the NASDAQ exchange in 1992. From the outset, it focused on mainframe environments. However, through the 90s and the first decade of this century the company diversified into other areas of computing, through a variety of acquisitions. However, in 2014 it sold off a number of its subsidiaries and spun-off Covisint. At the end of that year the company was acquired by the private equity company Thoma Bravo, LLC. Subsequent to that acquisition, what was then Compuware was split into two new companies: Compuware and Dynatrace, with Compuware now focusing exclusively on mainframe business, building on existing and well-established products: Abend-AID, File-AID, Hiperstation, Strobe and Xpediter.

Since its reincarnation, Compuware has made four acquisitions (ISPW, Integrations, Standardware COPE and MVS Solutions ThruPut Manager). They have also entered into a variety of technical partnerships. Notable amongst the latter are Atlassian, Jenkins, Splunk, Syncsort, XebiaLabs and SonarSource.

The company's headquarters are in Detroit, Michigan, and the company has a global presence across North America, Europe, Australia and Asia.

Since it became part of Thoma Bravo, Compuware has wholeheartedly embraced a company-wide agile methodology that has led to it publishing either new products or major updates to existing products every quarter since January 2015. This new approach was kicked off with the release of Compuware Topaz, a suite of products that, among other things, aims to modernise and bring agile development to the mainframe. In this contest, it is worth commenting that Topaz was the first new product to be developed by Compuware – as opposed to acquired – this century.

Website: [www.compuware.com](http://www.compuware.com)

# Conclusion

**T**opaz for Total Test is aimed primarily at development where COBOL programs need to be maintained and updated and this is where all of the product's features can come fully into play. In particular, Topaz for Total Test will automate swathes of the test generation process, not only saving significant amounts of time but, given the sheer quantity of COBOL programs present on most mainframes, enabling unit tests to be written on a large scale that otherwise would not

be practical. This really cannot be overstated: COBOL programs on most mainframes are usually measured in the thousands, meaning that writing unit tests manually for each and every program is not just impractical, but impossibly onerous. In our view, automatic test generation is the only practical way to bring unit testing to COBOL, and while other products in this space have some levels of automation, they do not have it to the extent that Topaz for Total Test does.

## FURTHER INFORMATION

Further information about this subject is available from [www.BloorResearch.com/update/2324](http://www.BloorResearch.com/update/2324)



### About the authors

**DANIEL HOWARD**  
Information Management

**D**aniel started in the IT industry relatively recently, in only 2014. Following the completion of his Masters in Mathematics at the University of Bath, he started working as a developer and tester at IPL (now part of Civica Group). His work there included all manner of software and web development and testing, usually in an Agile environment and usually to a high standard, including a stint working at an 'innovation lab' at Nationwide.

In the summer of 2016, Daniel's father, Philip Howard, approached him with a piece of work that he thought would be enriched by the development and testing experience that Daniel could bring to the

table. Shortly afterward, Daniel left IPL to work for Bloor Research as a researcher and the rest (so far, at least) is history.

Daniel primarily (although by no means exclusively) works alongside his father, providing technical expertise, insight and the 'on-the-ground' perspective of a (former) developer, in the form of both verbal explanation and written articles. His area of research is principally DevOps, where his previous experience can be put to the most use, but he is increasingly branching into related areas.

Outside of work, Daniel enjoys latin and ballroom dancing, skiing, cooking and playing the guitar.



**PHILIP HOWARD**  
Research Director/Information Management

**P**hilip started in the computer industry way back in 1973 and has variously worked as a systems analyst, programmer and salesperson, as well as in marketing and product management, for a variety of companies including GEC Marconi, GPT, Philips Data Systems, Raytheon and NCR.

After a quarter of a century of not being his own boss Philip set up his own company in 1992 and his first client was Bloor Research (then ButlerBloor), with Philip working for the company as an associate analyst. His relationship with Bloor Research has continued since that time and he is now Research Director, focused on Information Management.

Information management includes anything that refers to the management, movement, governance and storage of data, as well as access to and analysis of that data. It involves diverse technologies that include (but are not limited to)

databases and data warehousing, data integration, data quality, master data management, data governance, data migration, metadata management, and data preparation and analytics.

In addition to the numerous reports Philip has written on behalf of Bloor Research, Philip also contributes regularly to *IT-Director.com* and *IT-Analysis.com* and was previously editor of both *Application Development News* and *Operating System News* on behalf of Cambridge Market Intelligence (CMI). He has also contributed to various magazines and written a number of reports published by companies such as CMI and The Financial Times. Philip speaks regularly at conferences and other events throughout Europe and North America.

Away from work, Philip's primary leisure activities are canal boats, skiing, playing Bridge (at which he is a Life Master), and dining out.

## Bloor overview

Bloor Research is one of Europe's leading IT research, analysis and consultancy organisations, and in 2014 celebrated its 25th anniversary. We explain how to bring greater Agility to corporate IT systems through the effective governance, management and leverage of Information. We have built a reputation for 'telling the right story' with independent, intelligent, well-articulated communications content and publications on all aspects of the ICT industry. We believe the objective of telling the right story is to:

- Describe the technology in context to its business value and the other systems and processes it interacts with.
- Understand how new and innovative technologies fit in with existing ICT investments.

- Look at the whole market and explain all the solutions available and how they can be more effectively evaluated.
- Filter 'noise' and make it easier to find the additional information or news that supports both investment and implementation.
- Ensure all our content is available through the most appropriate channel.

Founded in 1989, we have spent 25 years distributing research and analysis to IT user and vendor organisations throughout the world via online subscriptions, tailored research services, events and consultancy projects. We are committed to turning our knowledge into business value for you.

## Copyright and disclaimer

This document is copyright © 2017 Bloor. No part of this publication may be reproduced by any method whatsoever without the prior consent of Bloor Research. Due to the nature of this material, numerous hardware and software products have been mentioned by name. In the majority, if not all, of the cases, these product names are claimed as trademarks by the companies that manufacture the products. It is not Bloor Research's intent to claim these names or trademarks as our own. Likewise, company logos, graphics or screen shots have been reproduced with the consent of the owner and are subject to that owner's copyright.

Whilst every care has been taken in the preparation of this document to ensure that the information is correct, the publishers cannot accept responsibility for any errors or omissions.



Bloor Research International Ltd  
20-22 Wenlock Road  
LONDON N1 7GU  
United Kingdom

Tel: **+44 (0)20 7043 9750**  
Web: **[www.Bloor.eu](http://www.Bloor.eu)**  
email: **[info@Bloor.eu](mailto:info@Bloor.eu)**