# *MODEL-BASED TESTING UPDATE*

*ROBERT V. BINDER*

Email*:* rbinder@ieee.org

## What is Model-based Testing?

Automated testing of software usually means writing small programs that attempt to check some aspect of a system. This is error-prone and turns to drudgery for a large system. But suppose that you could instead encode the same aspects in a way that your test programs could be automatically generated and run – that's model-based testing.

What is the essential difference between MBT and other kinds of testing? In other approaches, even if using an automated test harness, the test engineer first conceptualizes each test case. The next step is to document it for manual input, or to write and then run a test driver program. In MBT, the test engineer develops a model, which supports automated production and evaluation of test cases. A program that embodies a testing strategy produces test cases from the model, which are usually run and evaluated in an automated test harness.

For example, suppose we're developing a sports watch that simultaneously records time, distance, and heart rate. Under a traditional testing approach, we'd probably prepare a test plan organized around features and usage modes, and then choose specific scenarios and user actions as test cases. We'd probably have both lab tests (the watch would be embedded in a test fixture to ease control and observation) and field tests, where the tester would wear the watch and note its response in defined test cases.

With MBT, the test engineer produces a test model that specifies the required response to user inputs under all operational modes. Then, a test generation program would identify each event sequence that starts and ends the same state, and determine the necessary user inputs and expected result at each step.

The particular test case generation program determines the specifics of how the model is represented. For example, some systems use an event/state table or similar specification language; others accept source code with certain tags. There are many variations, and MBT is typically used in concert with traditional test approaches. Some tools have the ability to generate expected results along with the test inputs – a capability critical for industrial use.

## Model-based Testing and Reliability

Software reliability engineering (SRE) has been extensively studied and applied to good effect. SRE requires test suites where the relative frequency of operations in a test suite is a statistically accurate representation of actual field use. However, most applications of SRE simply tweak traditional testing strategies by requiring that the number of tests of a certain feature is in proportion to the expected field use of that feature.

To support SRE with MBT, the test model must represent the operations of interest and their expected relative frequency. If there are sequential constraints on input sequence, the model must also support a Markov chain or something like it to represent the relative frequency of each allowed operation sequence.

## Who Uses MBT?

Model-based testing has been studied for several decades, resulting in hundreds of research reports. There have been many industrial applications as well.

The first commercial tool to support model-based testing was offered in 1996. Leading IDE providers (e.g. IBM's Rational and Microsoft's Visual Studio) provide integrated model-based testing. Several firms specialize in general and special-purpose MBT tools.

Microsoft has made extensive use of model-based testing in its open protocols initiative. This project uses MBT to validate hundreds of Microsoft API specifications published to meet regulatory obligations. The tool used, *Spec Explorer*, is a Visual Studio plug-in and may be downloaded and used free of charge.

Many development organizations producing high reliability systems have developed in-house MBT systems, which are typically domain-specific and integrated with other engineering and manufacturing systems.

## Outlook for MBT

Although its technology has some inherent differences, the digital logic/device industry provides a broad roadmap for MBT. Sophisticated, model-driven, and automated verification has been a mainstay of chip development over all its vertical and horizontal segments. Integrated logic design and verification tools are commonly used. It is hard to imagine how today's complex multi-chip systems could have been developed without this. However, the situation for software systems is generally less evolved.

The cost for sustained use of MBT is often perceived to be excessive relative to traditional testing. Although its economic payback has been proven many times, to be effective, test engineering with MBT has to be on equal footing with systems engineering and take place upstream of the actual testing effort. Also, the recent "agile" emphasis on short-term and immediate results is often seen to be in conflict with the once-removed artifacts of MBT. This "cultural" friction can result in the exclusion of MBT or its abandonment after an initial experiment. Also, for firms not willing to develop their own MBT systems, gaps in the tool marketplace can be a deterrent.

Recent interest in model-driven architecture (MDA) has renewed interest in MBT, as MBT provides another way to leverage MDA models and tools, and requires a similar design-first discipline. MDA repositories and program-generation technologies can also support MBT, but time will have to tell how much real synergy will occur.

If you use or are looking to use SRE to support testing and release, MBT can be very useful. Once in place, a suitable MBT system can generate a statistically realistic yet unique test suite in minutes. In contrast, it isn't unusual to take weeks or months to manually produce a statistically realistic and unique test suite. In practice, this means manual reliability test suites are produced once and re-run, repeating exactly the same experiment over and over. With MBT, the input space can be re-sampled for each test run resulting in higher coverage, more effective testing, and greater confidence in reliability estimates.