Testing software: The new frontier

As the designs created in the aerospace and defense industry have grown more sophisticated, the resulting systems have become quite complex. This complexity starts with the mission and permeates the system being built, including the control system that runs, executes, or guides it. As a result, software development has become ever more critical to the development process.

An aerospace system today seldom has fewer than a dozen computers or microprocessors embedded in it. Each of these microprocessors has thousands of lines of computer code that must be designed, written, validated, and tested before the computer goes online. At the same time, software engineers are very difficult to find. This conundrum has motivated organizations and individuals to help automate this process.

Improving software code

To improve the quality of software code, several standard processes have been created. Although some of these processes, such as ISO 9000, are seen at a corporate or major division level, some can be applied at the level of a program or project. Two in particular are related to the aerospace and defense industry: CMM and DO-178B.

CMM, or Capability Maturity Model, is a method for evaluating the maturity of programs or organizations on a scale of 1 to 5. Originally developed by the Software Engineering Institute at Carnegie Mellon University, CMM has been used extensively for avionics software and for government projects since it was created in the mid-1980s. A revised version was subsequently released, known as CMMI, shorthand for Capability Maturity Model Integration.

The second process, DO-178B, classifies software into five levels of criticality related to atypical behavior that could cause or contribute to failure of a system function. RTCA, a private, not-for-profit organization that develops consensus-based recommendations regarding communications, navigation, surveillance, and air traffic management system issues, is responsible for the establishment and implementation of the DO-178B process.

There are several companies developing tools that will help reduce the huge disconnect between the amount of software being used and the engineering manpower needed to design, test, validate, and verify it. One is LDRA [http://www.ldra.com] in Wirral, U.K., and San Francisco, Calif. Its tools assist software engineers in the testing, validation, and verification stages of software development.

LDRA Testbed

The company's key product is the LDRA Testbed, a quality control tool that provides source code testing and analysis facilities for validating and verifying software applications. It is used primarily where software is required to be reliable, rugged, and as error free as possible. In providing those functions, LDRA brings time, cost, and efficiency savings to the development and implementation stages of the engineering process. Testbed's analysis facilities may be applied in the two main testing domains, static analysis and dynamic analysis.

Static Analysis is the primary source of information for the automatic documentation of software. It generates all of the vital control flow information for each procedure and the interprocedural links. The interfaces are accurately delineated, the loop structure is exposed, and complexity metrics are generated. LDRA Testbed's Static Analysis is approved for the analysis of safety-critical code by many regulatory authorities for whom static analysis is the sole requirement.

Dynamic Analysis uses test data sets to execute software in order to observe its behavior and produce test coverage reports. This assessment of source code ensures consistent levels of testing and correct use of capture/playback tools. LDRA Testbed Dynamic Analysis provides the facilities to achieve quality standards for critical code, improve code efficiency, minimize regression test costs, and detect software defects.

When used during software development and maintenance, Dynamic Analysis techniques can contribute significantly to a software program's robustness and reliability. The product explores the semantics of the application under test via test data selection. Control and data flow models constructed from the static analysis of the software application are compared with the actual control flow and data flow that occur at run time. This enables checks that show errors in either the static or dynamic analysis.

Such analysis is particularly effective for software applications that must achieve high levels of reliability. It is the primary requirement for the testing of safety-critical avionics software and is widely used in all military, safety, and mission-critical software.

The results of Dynamic Analysis may vary considerably depending on the design requirements for which the results are intended. The testbed can use several types of coverage:

- Statement coverage is the simplest form of measuring the level of code coverage attained with a test data set. It is the measurement of how many executable statements are exercised by a particular test data set. Measuring statement coverage attained during the test process helps to ensure that no areas of code are left unexercised, and hence reduces the likelihood of the occurrence of certain classes of software errors.
- Branch/decision coverage, which is

used throughout the software industry as an essential level for all software testing, measures how many branches/decisions are exercised by test data.

•Linear code sequence and jump (LCSAJ) coverage is a language independent, "third-level" code coverage testing mechanism. Pioneered by LDRA, it currently yields the highest attainable code coverage testing standards. LCSAJ forms a finite set of paths through the source code application under test. The finite nature of this set of paths means it is feasible to test all LCSAJs.

By contrast, attempting to test all control flow paths, although it may be desirable, is not technically possible for anything but the most trivial of programs. This is because even simple constructs, when used in combination, give rise to an exponentially large number of paths, and any nondeterministic loop constructs will yield a potentially infinite number.

•Modified condition decision/branch condition (MC/DC) testing and BCC (branch condition combination) testing are closely related, as are the associated coverage measures. MC/DC requires test cases to show that each Boolean operand (A, B, and C) can independently affect the outcome of a decision. Although this tests less than all the possible combinations, it is a pragmatic compromise that requires fewer test cases than BCC. It is widely used in the development of avionics software, as required by the DO-178B standard.

Automating with TBrun

In response to the industry's software testing needs, LDRA has also developed TBrun. Using the static and dynamic analysis facilities of the LDRA tool suite, this product provides an automated unit test solution. TBrun automatically generates test harnesses for the unit under test. This can save time, free up highly qualified staff, increase test efficiency, and improve motivation to test through a repeatable and less error-prone process.

As TBrun is fully integrated with LDRA Testbed, it enables the use of a wide range of static and dynamic analysis techniques at a variety of levels. The code being analyzed (the "unit") therefore may be a single function, a set of functions, a source file, a subsystem, or even a complete system. This feature allows TBrun to be used for unit, module, subsystem, and integration testing. Studies have shown that, compared to manual techniques, testing efficiency in the unit code and test arena can be improved by as much as 76% using TBrun.

Unit testing may be defined as the process of verification and validation of an individual module or unit of software. In its simplest form, a unit may be a single function or method that has been isolated from the main body of application code. The analysis of this unit, in isolation, is normally made possible when the developers write additional driver modules or test harnesses, which "manage" the necessary unit inputs and outputs.

The major strength of unit testing is that it enables developers to apply analysis techniques much earlier in the development life cycle than might otherwise be possible. Studies, however, indicate that such testing is underutilized by up to 90% of software developers. This is because traditional techniques are labor intensive, costly, and dependent on expert knowledge; thus they are unattractive to developers and testers alike. Nevertheless, there is general agreement that it is much more cost effective to identify and resolve software errors early in the development cycle; hence the application of source code analysis techniques at the unit level will yield significant long-term cost benefits.

DO-178B examples

The LDRA tool suite has been specifically enhanced to enable simple measurement of conformance to the various levels of DO-178B. Reports are specifically tailored to give users DO-178B information quickly and concisely, speeding up the testing procedure. Reports can be produced in either ASCII or HTML. Either format can be easily incorporated into a word processor. HTML has the added advantage of links and the ability to be published on the Internet or an organization's intranet.

The LDRA tool set enables analysis for the Structural Coverage Analysis requirements that are outlined in section 6.4.4.2 of the DO-178B specification. In addition, a wide variety of programming languages can be tested. These include C, C++, Ada, Fortran, Java, Visual Basic, and many assemblers.

One of the big questions with regard to the DO-178B specification is, who has successfully used the tool set before? The list includes Airbus, Bell Helicopter, Boeing, Honeywell, Lockheed Martin, Northrop Grumman, Sener, Smith, and Thales Avionics, to name just a few.

John Binder
jbinderaero@rcn.com
OnePIN: 2447-6000-4391

AEROSPACE AMERICA

Let ***Aerospace America*** help you with your recruitment efforts.

Our Career Opportunities pages reach over 40,000 aerospace professionals every month.

Send all contracts, insertion orders, and artwork for space ads to:

Howard O'Brien,
AIAA
1801 Alexander Bell Drive,
Suite 500
Reston, VA 20191-4344;
Phone: 703/264-7535;
Fax: 703/264-7551
e-mail: howardo@aiaa.org