LDRA

Software Technology

The Development of Object Code Verification

Bill StClair of LDRA looks at the increasing need for verifying object code in software development

Article by Bill StClair originally published in Automotive Electronics Magazine Feb/Mar 2006

automotive electronics

The Development of Object Code Verification

Bill St. Clair of LDRA

Bill StClair of LDRA looks at the increasing need for verifying object code in software development

An ever increasing reliance upon software control has meant that many companies from the automotive and other business sectors that do not have a traditional requirement for sophisticated software analysis now find themselves compelled to undertake safety-critical/ safety related testing by the nature of the applications they now develop.

With this increased requirement for software testing across different industries a tendency has emerged for companies to look outside their own market sector when seeking best practice techniques or standards. Examples of such industry crossover have been seen in the automotive and avionics industries with the adoption of elements of the DO-178B standard in the former and a similar adoption of the MISRA standard in the latter.

With out of sector testing standards comes the potential for unfamiliar testing techniques. This is illustrated by, amongst others, the object code verification requirements of the DO-178B standard. While a key testing element of many avionics programmes it has been a relatively un-used technique outside this industry.

The increasing sophistication and safety-critical nature of many modern embedded control applications, however, mean that as non-avionics based suppliers adopt DO-178B then object code verification is one of the key elements that they have to sit up and take notice of.

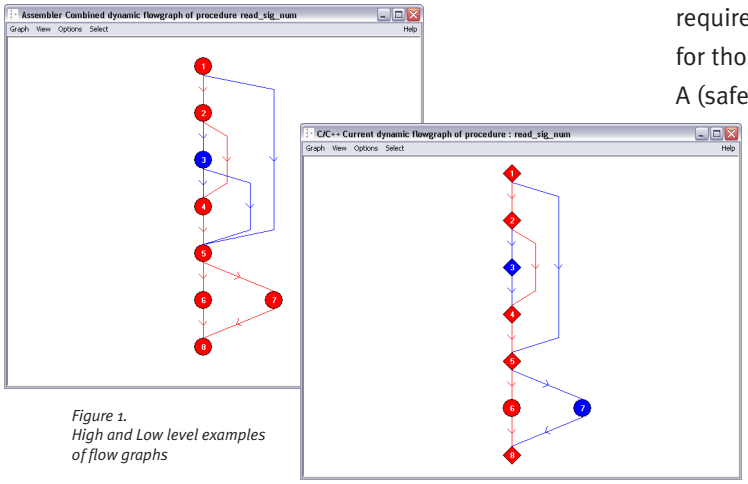

Figure 1.
High and Low level examples of flow graphs

Object Code Verification

So what is object code verification?

The relevant section of the DO-178B standard (6.4.4.2 Structural Coverage Analysis) describes the requirement as follows:

"The structural coverage analysis may be performed on the source code, unless the software is level A and the compiler generates object code that is not directly traceable to source code statements. Then, additional verification should be performed on the object code to establish the correctness of such generated code sequences. A compiler generated array bound check in the object code is an example of object code that is not directly traceable to the source code."

In a nutshell, object code verification is concerned with how much the control flow structure of the compiler generated object code differs from that of the application source code from which it was derived. Such differences may occur for a number of reasons, e.g. compiler interpretation, optimisation, etc. Given, however, that traditional structural coverage techniques are applied at the source code level whereas it is actually the object code that executes on the processor, differences in control flow structure between the two can make for significant gaps in the testing process.

The demands of DO-178B are such that developers of applications that are subject to the standard are required to implement object code verification facilities for those elements of the application that have a Level-A (safety-critical) classification.

While this is often a sub-set of the application as a whole, it can nevertheless represent a significant amount of testing effort and hence require considerable resources in terms of time and money.

As such, opportunities to implement automated, compiler- independent processes can help to reduce overall development costs by considerable margins.

Object Code Verification Solutions

The software development market has recognised and responded to the increasing requirement for object code verification test facilities from differing industry sectors and many software tool vendors can now provide either partial or complete structural coverage analysis solutions for both source and object code from unit to system and integration levels.

The differing solutions on the market tend to utilise combinations of both high and object level (assembler) source code variants of tool suites with the object level tool variant being determined by the target processor that the application is required to run on. A typical example might see a combination of C/C++ as a high-level language and TMS320C25x Assembler at the object level with copies of appropriate tools teamed together to provide the necessary structural coverage facilities. Many other high level/assembler language combinations are supported by a variety of tool vendors and examples of the well known coverage metrics that these solutions typically support are listed below.

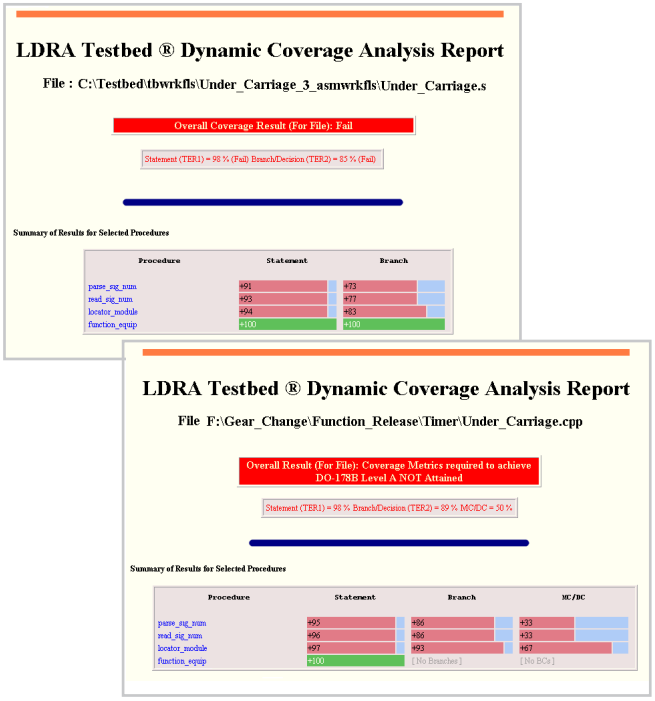

Figure 2: High and Low level examples of dynamic analysis coverage reports

- Statement
- Branch
- Test path
- Procedure/Function Call
- Boolean Expression Coverage
- Branch Decision Condition
- Branch Condition Combination
- Modified Condition/Decision (DO-178B)*

(*Language dependent)

Object Code Verification at the Unit Level

Some tool vendors have taken a significant step further by extending their object code verification solutions to provide partial or fully automated facilities that are targeted at the unit test level and hence enable this sophisticated analysis technique to be applied at a much earlier stage of the software development life-cycle.

This Object-box Mode, as the unit test object code verification facility is referred to by some vendors, enables users to create test cases for structural coverage of high-level source and apply these exact same test cases to the structural coverage of the corresponding object code.

Key to this facility is the generation of an enhanced driver program which, depending on the sophistication of the vendor solution, is either automatically created or created by manual or partially automated means.This driver encapsulates the entire test environment, defining, running and monitoring the test cases through initial test verification and then subsequent regression analysis. In Object-box Mode this driver may be linked with either the high-level source unit or the associated object code. In so doing users can ensure that a uniform test process may be applied and compared in order to determine any discrepancies / deficiencies.

If structural coverage discrepancies / deficiencies are identified at the object level users are then presented with an opportunity to define additional test cases to close any gaps in the test process. The obvious advantage of being able to identify and apply corrective action at such an early software development stage is that it is much easier and cheaper. It also significantly increases the quality of the code and the overall test process with the latter reaping benefits at the later stages of integration and system testing and then onward in the form of reduced failure rates / maintenance costs when the application is in the field.

While the code is still under development, together with satisfying the necessary object code verification requirements in a highly automated and cost-effective manner, developers can also benefit from the considerable additional test feedback that is provided by software testing tools in the form of sophisticated Code Review and Design Review elements. The results of these analysis facilities can be fed back to the development team with the possibility that further code and design deficiencies may be identified and rectified, further enhancing the quality of the application as a whole.

Conclusion

There is no doubt that object code verification presents a significant challenge to those software development projects that are required to undertake it.With the right tools and facilities, however, the scope of these challenges may be greatly reduced thus enabling developers to realise the full potential and benefits that such analysis may bring in terms of increased code quality and reliability.

Bill St Clair is a Technical Evangelist at LDRA

LDRA Headquarters
Portside, Monks Ferry,
Wirral, CH41 5LH
Tel: +44 (0)151 649 9300
e-mail: info@ldra.com

LDRA Technology Inc. (US)
Lake Amir Office Park
1250 Bayhill Drive Suite # 360
San Bruno CA 94066
Tel: (650) 583 8880

www.ldra.com