LDRA
Software Technology

Testing Automotive Software Applications

Working with the automotive industry to meet the challenges of achieving software quality and reliability

www.ldra.com

Background

Modern motor vehicles make greater use of software than ever before with applications ranging from simple climate control to sophisticated anti-lock braking systems and engine control units (ECUs). Much of this software is of a safety-critical or safety-related nature and, as such, must be designed, developed and tested to the highest possible standards.

The automotive industry has recognised and responded to the need to ensure best practice in software development. This is demonstrated by the introduction of new initiatives and the development of new software standards, most notably the MISRA (Motor Industry Software Reliability Association) standard for the C programming language. The MISRA standard was published in 1998 and was developed in response to the increasing use of C in safety-related automotive applications. Today the standard has been widely adopted throughout the software industry as a basis for encouraging good programming practice.

LDRA Tool Suite Functionality

The source code analysis features provided by the ***LDRA tool suite*** are designed to promote best practice for the development and testing of software applications. Such features include the checking of source code against industry recognised coding rules, such as those stipulated under the MISRA C standard and more recently MISRA-C: 2004 . Such automated source code checking takes many forms including the reporting of complexity metrics and the measuring of code coverage via Dynamic Coverage Analysis techniques. This combination of analysis techniques is successfully used in the automotive industry for the development of well designed and tested code, which is safe in service.

The purpose of this document is to detail the source code language specific analysis capabilities of the tool suite.

Static Code Analysis

Static code analysis is a cost effective technique to monitor the quality of source code, augmenting or replacing traditional manual peer code reviews. The techniques listed below enable software developers and QA professionals to identify and remove errors and improve code structure. The end result is code of improved quality, testability and maintainability.

The ***LDRA tool suite*** provides powerful static code analysis capabilities, all of which are available at unit, module or system level (across file boundaries):

- Control flow analysis
- Data flow analysis
- Information flow analysis
- Structured programming verification
- Programming standards checking (includes MISRA C / MISRA-C: 2004)
- Static array bounds checking
- Infinite loop detection
- Divide by zero detection

Code Coverage

The LDRA dynamic analysis features measure the effectiveness of software testing. Through graphical and textual reports the tool highlights areas of source code that have been tested, and more importantly, source code that has not been tested. Utilising this information enables testers to improve their tests to achieve desired code coverage standards and thus greatly increase confidence in the tested code. From unit test to system test, the LDRA tool suite may be used for implementing an efficient, effective test process.

Specific features include:

- Statement coverage
- Branch coverage
- LCSAJ coverage (test path)
- Subcondition coverage (including MC/DC)
- Data flow relationship coverage
- Array bounds violation detection
- Change impact coverage
- Data set analysis for efficient maintenance testing
- Profile analysis for reducing regression testing costs

Exact Semantic Analysis

By utilising the exact semantics of the target processor the ***LDRA tool suite*** provides the user with powerful, annotation based, conformance criteria that can be checked rapidly and with minimal user intervention. Important applications are:

- Divide-by-zero detection
- Pre and post condition checking
- Specification checking
- Code proving

Host/Target Testing for Embedded Software Systems

Source code analysis and dynamic testing with the ***LDRA tool suite*** may be performed in a host environment (e.g. on a Windows, or UNIX machine), in simulated environments or utilising the embedded target processor to ensure the application under test will perform as specified in its native environment.

Language Specific Features

The ***LDRA tool suite*** provides analysis and reporting facilities for the following language specific features:

- Pointer analysis
- Class inheritance handling
- Polymorphism
- Template handling
- Register data flow analysis

Unit Testing

Unit testing is a proven, cost effective, analysis process to ensure that systems are thoroughly tested at the earliest possible opportunity during the software development lifecycle. Traditionally seen as expensive and time consuming, the ***LDRA tool suite*** automates key manual processes and makes unit testing a realistic technique to be used by all software developers. The key facilities provided are as follow:

- Automatically generates test harnesses and driver programs
- Runs tests on single units, sets of units and subsystems
- Detects changes in source code and documents the associated required changes to tests
- Stores test data and results
- Performs regression tests automatically as part of a repeatable batch process
- Runs in host/host and host/target environments

Cost Implications

The LDRA tool suite has a proven capacity to reduce software production costs by:

- The use of fast and effective fault finding technologies
- The provision of a wide spectrum of software assessment metrics
- Superimposing faults on to the actual source code for instant visual impact
- Simple and direct user interface with extensive point-and-click facilities
- Command-line mode which provides off-line analysis and automated regression testing

Through the implementation of an automated unit test process costs may be further reduced by:

- Removing the need to manually write and maintain scripts in another language
- Removing the requirement for specifically qualified personnel to perform the unit test process
- Minimising the number of essential tests for regression testing to reduce time and cost
- Reducing maintenance costs by documenting the relationship between test data sets and code components

Both the user interface and core functionality are largely common for all languages (C/C++, Assemblers, etc) and platforms. Thus experience and skills gained from using the tool for the analysis of one source code language are transferable to another.

Pedigree

LDRA Ltd was established in 1975 and has unparalleled experience in the provision of static and dynamic analysis test solutions and consultancy. Much of this experience has been gained through close links with the automotive industry either through the provision of specialist software scrutineering services to motorsport series such as the F1 World Championship and British Touring Car Championship, or through the supply of test tools and test solutions to mainstream automotive manufacturers and their suppliers. Most recently LDRA's association with the automotive industry has included membership of the Motor Industry Reliability Association committee where LDRA's Mike Hennell participated in the design and implementation of the latest MISRA-C:2004 software standard.

LDRA Ltd. Portside, Monks Ferry,
Wirral, CH41 5LH, UK.
t: +44 (0)151 649 9300
f: +44 (0)151 649 9666
w: www.ldra.com
e: info@ldra.com