



LDRA

**Increase Productivity with
Automated Unit/Integration/Low
Level Testing with LDRAunit®**

**Delivering Software Quality and
Security through Test, Analysis
and Requirements Traceability**

Overview

LDRAunit®, LDRA's class leading stand alone unit/integration test tool, provides a complete verification environment for the automated generation and management of test harnesses and unit/integration tests. This solution maximises developer productivity by giving them the ability to focus on implementing correct software functionality versus burdensome and time consuming, low-level manual testing activities.

LDRAunit automates and increases test throughput and repeatability to significantly increase overall test effectiveness. Software development managers seeking to develop the highest quality code in the most cost effective manner are leveraging automated unit/integration testing to avoid the potential delays caused by inefficient manual low-level testing strategies. These traditional techniques often are inadequate and postpone the discovery and correction of defects until late in system test where they are most expensive to fix.

Making use of the comprehensive control/data flow analysis provided by LDRA Testbed®, LDRAunit determines details of the unit interface, parameters, globals (input and output), return values, variable types and usage and procedure calls. Traditionally this level of information could only have been specified by a developer with an expert knowledge of the unit(s) under test. By automating this process LDRAunit frees up highly qualified staff who may then be re-assigned to other modelling, design and development tasks.

LDRAunit facilitates several test scenarios:

- Single procedures, functions, methods (Unit test)
- Files containing many functions, classes (Module/integration test)
- Complete programs (Sub system & system test)

LDRA has revolutionised the traditional “unit testing” activity, which is typically performed on the host and/or target systems with its new automatic testing capability, eXtreme Testing.

This high degree of test automation saves both time and resources, thereby enabling a quicker time to market. LDRAunit's ability to work in a highly distributed environment provides complete visibility into the overall development processes which can be accomplished even if development teams are distributed globally.

Unit/Integration Testing Embedded Systems with LDRAunit

LDRAunit supports the creation and execution of test cases in multiple environments, namely:

- **Host/Host**
- **Host/Target**
- **Host/Simulator**

LDRA's Unit/Integration Testing Features:

- Automated test driver / harness generation with no manual scripting requirement
- High levels of test throughput via the intuitive graphical and command line interface options
- Sophisticated automated analysis facilities which reduce test effort, freeing up developers and empowering testers
- Storage and maintenance of test data and results for fully automated regression analysis
- Automated detection and documentation of source code changes
- Tool driven test vector generation
- Execution of tests in host, target and simulator environments
- Automated generation of test case documentation including pass/fail and regression analysis reports

The screenshot displays the LDRAunit software interface with several key components:

- Sequence Viewer:** A tree view on the left showing a list of test cases (TC 1 to TC 15) under a 'Controls' folder.
- Test Case Detail Viewer:** A table at the bottom left showing test results for 'Is_Brakes_Applied' across multiple instances (10, 11, 12, 13).
- Sequence Diagram:** A central flowchart showing the execution flow of a test case, with nodes representing code blocks and arrows indicating the sequence of execution.
- Code Coverage:** A window on the right showing the source code of the 'Is_Brakes_Applied' function, with colored highlights indicating which lines were executed during the test.
- Legend:** A small legend at the bottom of the diagram window explaining symbols: green for 'Branch/Decision Covered', red for 'Branch/Decision not Covered', diamond for 'Node has Branch/Decision', and circle for 'Node has no Branch/Decision'.

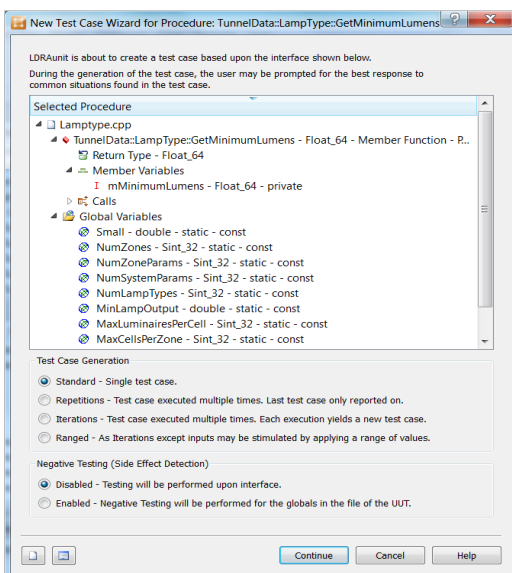
Is_Brakes_Applied	Is_Brakes_Applied	Is_Brakes_Applied	Is_Brakes_Applied	Is_Brakes_Applied
1 amount 0	amount 10	amount 10	amount 10	amount
2 depressed 0	depressed 1	depressed -1	depressed 1	depressed
3 durability	durability	durability 85	durability	durability
4 % 1	% 1	% 1	durability	durability
5 durability	durability	durability	% 1	%
6				

LDRAnuit enables the fully automated creation of test driver programs. The generated driver handles all language features automatically. Key features are detailed below:

Automatically Generated Driver Program/Test Harness

LDRAnuit utilises sophisticated control flow and data flow analysis techniques to document the interface to the unit(s) under test in full. This level of information then enables LDRAnuit to automatically generate test drivers removing the need for manual scripting.

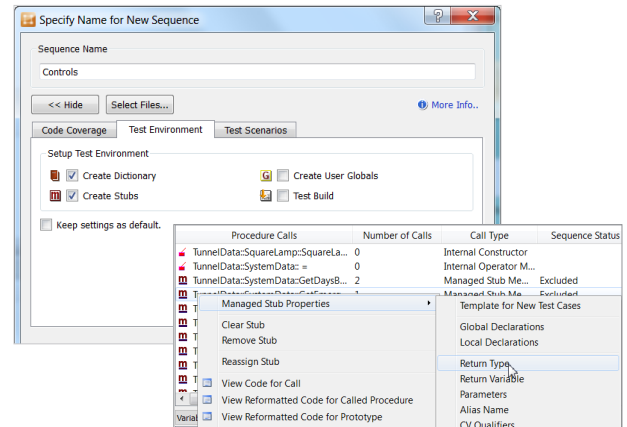
There are no limitations to the automatically generated driver. It is pure C/C++, Ada 83/95 or Java depending on the application code and can be executed in the host, target or simulator environment as required.



Stub Creation

Stubs can be written by hand or generated automatically for functions, methods, constructors, system calls, packages, generics, etc. The automatically generated “managed stubs” are sufficiently complete to allow the test harness to build and execute.

The default behaviour of managed stubs can be modified via an intuitive graphical user interface to tune such items as return and global parameter values. For instance, it is possible to vary return values depending on the number of occasions on which the stubbed function has been called, whilst passed parameter values can become pass/fail criteria for the unit tests themselves.



Exception Handling

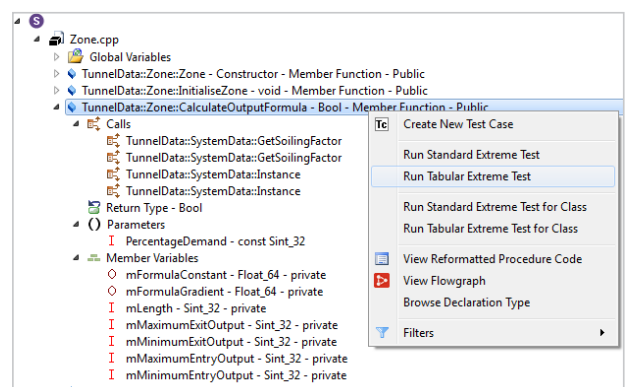
Exceptions can be automatically caught and test cases can be passed or failed dependent on whether such an exception has been raised. The exception handling method is configurable.

The exception handlers themselves can also be subject to unit tests. Such tests can be applied irrespective of whether the exceptions are raised, allowing coverage to be achieved even when the raising of an exception would be impractical.

eXtreme Testing

eXtreme Testing builds on LDRAnuit’s ability to automatically populate unit test cases, extending this to the generation of the test cases themselves. It automates the unit/module/integration testing processes and, by encompassing test harness and test vector production, it eliminates almost all of the overhead associated with bottom-up testing. It is the fastest and simplest way to get started with unit testing.

Features include the ability to automatically fine tune the processes used to create the test vectors to optimise the level of coverage achieved. Vectors generated by means of eXtreme Test can then be complemented by means of manually generated test cases.



Test Case Files/Test Case Management/Storage

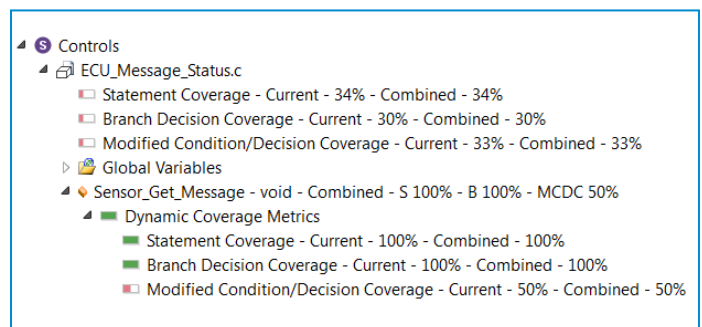
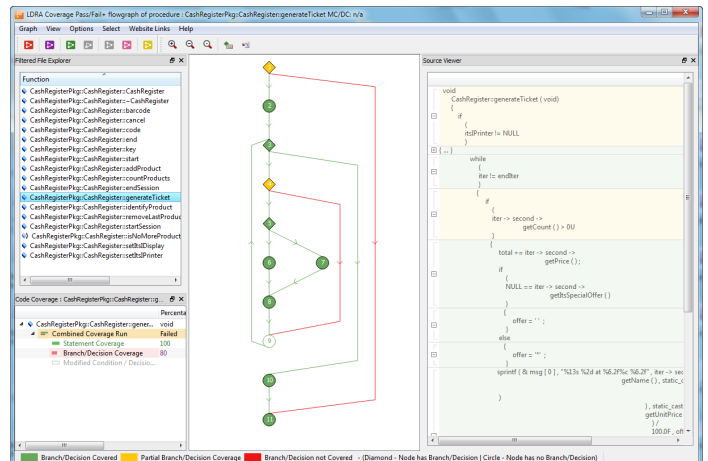
LDRAunit stores groups of test cases as sequences. Users can then export a sequence to a Test Case File (TCF) which contains all of the information required to re-run the test cases. TCF's can be grouped with regression reports and can be stored for regression verification and either saved with the source file, via a software configuration management (SCM) system, or used as an annotation. Requirements based testing documentation, including why particular values were chosen and tags to map to a requirement management system, can be added for storage.

When used as a SCM annotation these files allow managers to determine directly from the SCM system that developers are testing their code on check in. TCF's can also be re-run from the command line and in batch mode so that as the source code changes module interfaces and output can be verified. For companies concerned about managing outsourced development, TCF files can be easily distributed and provide a standard template around the world.

Structural Coverage Metrics

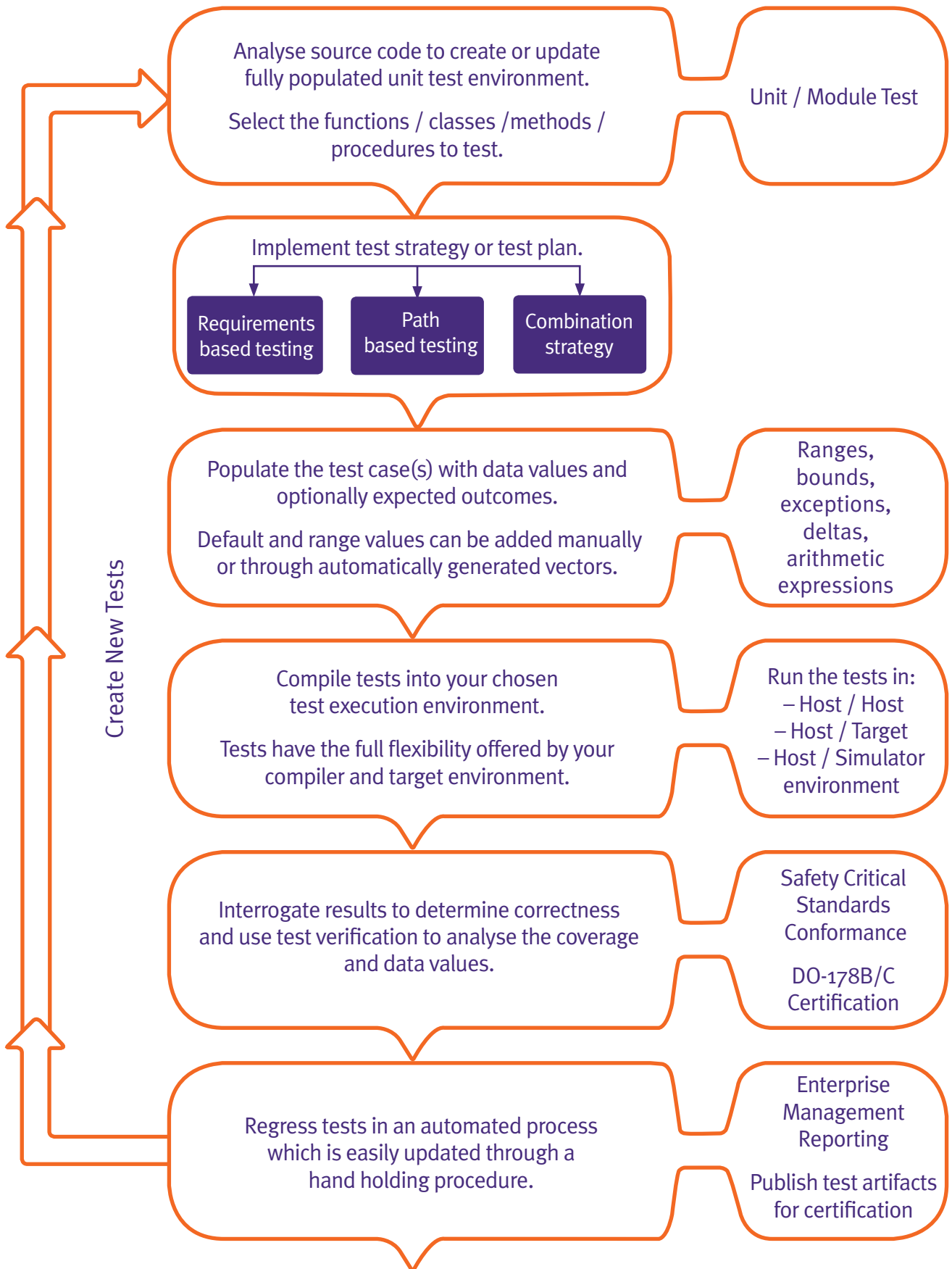
LDRAunit has access to the full range of coverage metrics available in the LDRA tool suite. These include Procedure Call, Statement, Branch/Decision, MC/DC and LCSAJ (Test Path). Users can choose an appropriate metric or set of metrics based on their safety and program constraints. For example, MC/DC coverage is essential to verify results are not masked by condition input conditions and LCSAJ coverage provides a comprehensive metric to evaluate loops. All of these metrics are available graphically, via flow graph displays, call graph displays and the file view of the LDRAunit GUI. Users can directly access compliance reports

to give overall pass / fail metrics for standards such as DO-178B/C. Line by line views indicating which statements, branches and conditions have been executed are also shown in these reports.



Additional Automatically Handled Language Features:

- Abstract Class Testing
- Automatic Creation of Compound Objects in Test
- Access to Private and Protected Data
- Re-use of Tests through Class Hierarchy
- Polymorphism
- Inheritance
- Templates
- Structure/Arrays/Unions
- Automated Resolution of Templated Types
- Classes
- Automatic Creation & Object "Re-Use" (Through Attachment)
- Access Methods & Attributes through the entire Hierarchy
- Exceptions
- Pointers
- Generics (Ada)
- In / Out Parameters (Ada)
- Records (Ada)



LDRA's best in class features are illustrated on real world projects with the client testimonials below.

Client Testimonials

Toyota's GT 86



'LDRA has the ability to work with hardware that has limited resources which is important in the automotive sector in order to meet the demands for cost reduction and downsizing. We use the LDRA tool suite as a benchmark for other third-party and similar software platform products.'

Akihito Iwai, Project Manager DENSO Japan

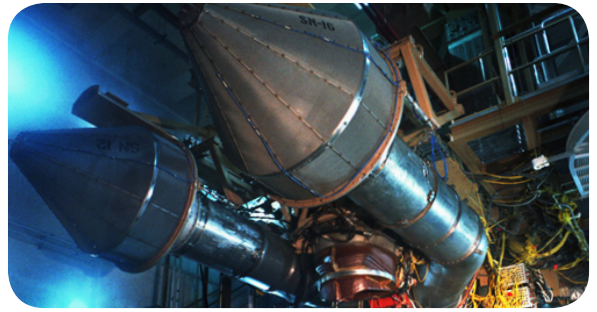
Orion Crew Exploration Vehicle



'Once the LDRA tool suite has analysed the code, it knows the inputs and outputs, expected returns and types, and every function. So, instead of spending months hand-coding and qualifying a test harness, the tool can use this information to automatically generate a test harness that will call every function, feed data in and out, and test every path of the program.'

Trevor Tidwell, Software Development Engineer, USA

Pratt & Whitney F135 Engine for JSF



'We found that the graphical user interface was easy to work with and made developing a rapid, intuitive test process a lot easier than creating it manually. This saving was further increased through the repeatability of tests utilising the automated regression testing facilities. This automated solution made our job a lot easier. The LDRA tool suite resulted in a saving of £2 million.'

Tom Roberts, Engineering Manager, Embedded Software and Systems, Ultra Electronics Datal

F-35 Lightning II



'LDRA has proven they will support us in any way to get the job done especially in meeting demanding milestones. They provided outstanding support for several F-35 teammates: Lockheed Martin (Fort Worth), BAE (Warton), Northrop Grumman (El Segundo), Seaweed, and Honeywell which directly contributed to a successful first flight of the AA-1 aircraft. We continue to work closely with LDRA to develop the needed automated process support to ensure that our software meets program cost, schedule, and quality targets.'

John H. Robb, Air Vehicle Software Senior Manager, LMCO

Languages & Platforms

LDRAunit is available for the following source code languages and host / target platforms:

Languages

C/C++
Ada 83
Ada 95
Java

Host Platforms

Windows 7/8/XP
Solaris
Linux
RHEL 5 32 bit

RHEL 6 64 bit
Ubuntu 10.10 32 bit
Ubuntu 12.04 64 bit

Target Platforms

IDE:

Analog Devices
AONIX
ARM
Cosmic
Eclipse
Freescale
GNU
Green Hills

IAR

iSYSTEM
Keil
QNX
TI
Renesas
TASKING
Wind River

Processor:

ARM
Freescale
Infineon
Intel
MIPS
PowerPC
Renesas
TI

Other languages and host / target platforms are available. Please contact LDRA for more information.



Certificate Number FM 26376

All brand names and product names mentioned herein are trademarks registered trademarks of their respective companies.
Picture acknowledgements: Chrysler, General Electric, Lockheed Mart Sellafield, NASA, Pratt & Whitney, Toyota, United Space Alliance
LDRA Ltd. reserves the right to change any specifications contained within this literature without prior notice.

Designed by Young Greenwood Design (02260) 226541

© 2013 LDRA Ltd

www.ldra.com



Van Horneplein 6
6019 BW Wessem
The Netherlands

Tel: +31 77 307 8438
www.logic.nl
info@logic.nl

Bunsenstrasse 18
81735 Munich
Germany

Tel: +49 89 1436 7945
www.logictechnology.de
info@logictechnology.de