LDRA

Software Technology

Efficient testing ensures requirements traceability and verification

Integrating requirements with automotive electronics hardware and software testing streamlines development and cuts costs

by Bill StClair of LDRA

Article by Bill StClair originally published on Automotive DesignLine July 2006

AUTOMOTIVE DesignLine

Efficient testing ensures requirements traceability and verification

Bill StClair of LDRA

Integrating requirements with automotive electronics hardware and software testing streamlines development and cuts costs

The need for requirements traceability and verification is typically imposed on automotive electronics suppliers as a contractual requirement. With increasing frequency, vendors are recognizing that requirements-based testing is an essential element of successful software development projects in general.

As a contract deliverable, or more generally, as a work product, the requirements traceability task produces a Test Verification Matrix (TVM)—an artifact that is painfully wrought, consuming valuable resources that are frequently diverted from other more productive activities.

The truly onerous nature of a TVM does not become apparent until one attempts to maintain the TVM through testing, integration, and deployment phases of projects. The inherent inadequacies of the TVM and the manual processes it represents are exposed as defects occur. Specifically, many of these defects are attributed to requirements management, including requirements validation, allocation, and correct implementation. In fact, records indicate that up to 70% of such defects are classified as requirements management related!

The next challenge is to produce a requirements traceability solution that is dedicated to development and test teams that must operate in the context of existing tools and processes. Currently, most customers LDRA sees have a requirements database or flat file capability where they define and maintain system or high-level requirements.

Late mapping

Some customers map these high-level requirements to top-level design; even fewer map these requirements to the as-built design and source code. In the main, customers at least map requirements to the test cases that verify these requirements. However, the possibility of erroneous mappings are very high when customers wait until testing, especially system testing, to perform requirements traceability.

The reason why this very late requirements mapping occurs is the operational constraints imposed by a requirements database located in a project manager's office and the testing environment existing on a developer's workstation or on a target system in a lab. Or perhaps the testing is being performed by a subcontractor in a remote location. At a minimum these operational constraints dictate that a level of integration occur between the requirements database and the test environment in order for an automated solution to be introduced.

A more effective process is to map requirements at least to the as-built (or detailed) design and the embodied source code. Mapping to the as-built system is part of test qualification or the test readiness process that determines the proper correlation of requirements to code; a corollary to this review is the elimination of dead (unreachable) code from the source code listings. Moreover, it can be argued, that infeasible code, or code that cannot be exercised under any combination of test data, should also be remedied or purged as a prerequisite for test readiness.

The optimal solution for requirements traceability includes the mapping of system requirements to the top-level design as a first step, advisably performed while utilizing a design modeling tool. (This option is described in the LDRA white paper "*LDRA tool suite* / Telelogic I-logix Rhapsody Integration.")

Prototyping

The traceability of requirements through the as-built design is further compelled by the existence of low-level and derived requirements. These requirements are commonly defined by the development team in the course of system requirements elaboration (or prototyping) and the construction of a workable and testable system. This pattern of product evolution is most pronounced in the development of software for embedded targets in which target constraints and hardware requirements must also be considered.

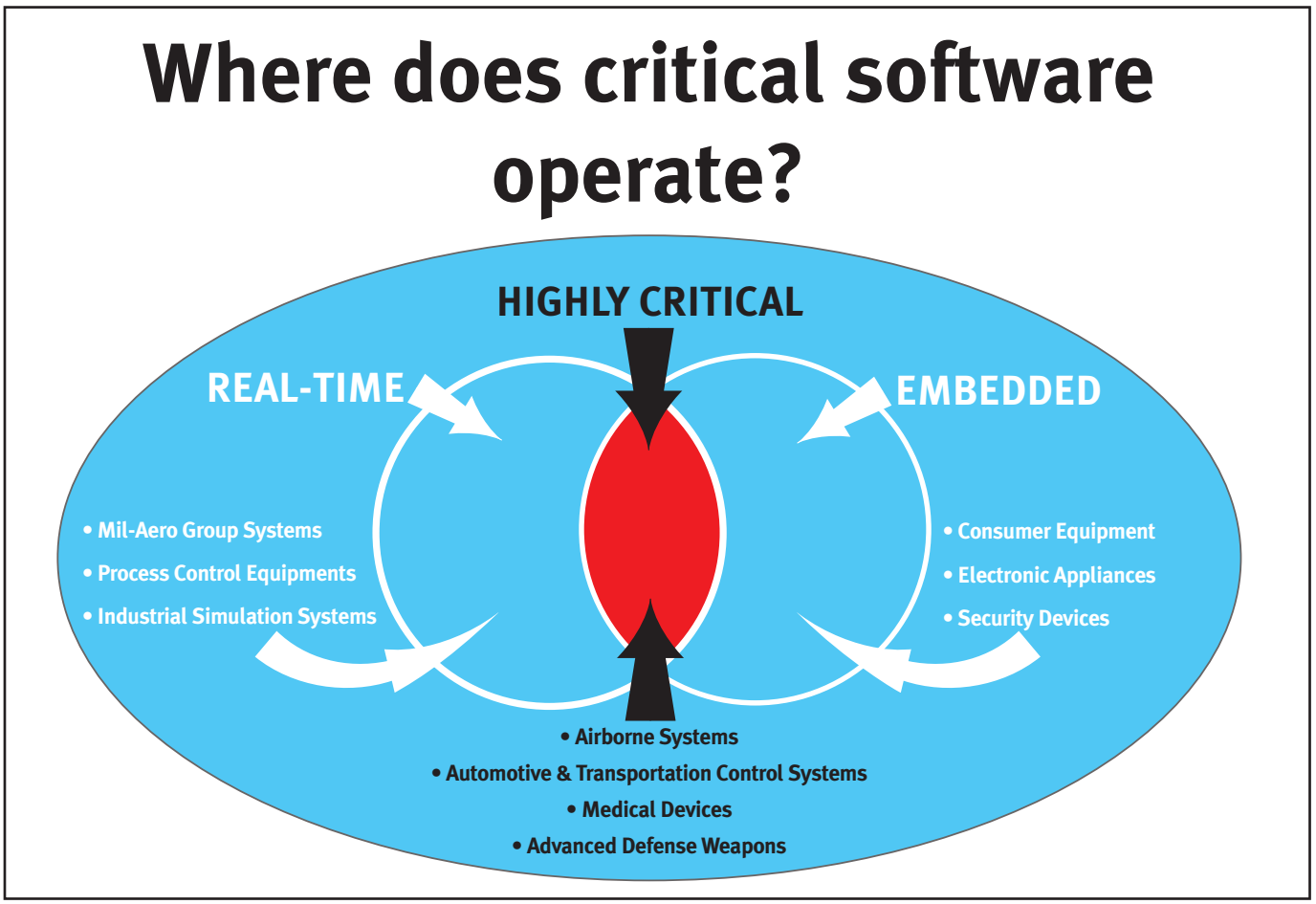
The prevalence and the context of low-level requirements present another significant challenge for traceability. These requirements are not considered system or "customer" requirements; they address the "how" of a software system in contrast with customer requirements that define "what" a system shall do. Consequently, low-level and derived requirements are frequently maintained separately from system requirements. This presents yet another data management demand.

A critical aspect of low-level requirements management, traceability and verification is the dissemination of these requirements to developers and testers. The developer needs to be fully informed of the interface specifications for the code he or she will implement and the procedures this code will call. These specifications must be explicitly coupled to the associated high-level requirements in order for the developers to properly understand the context of the implementation. Properly informed, the developer can design for testability and consider the functionality that must be exercised at multiple levels of testing.

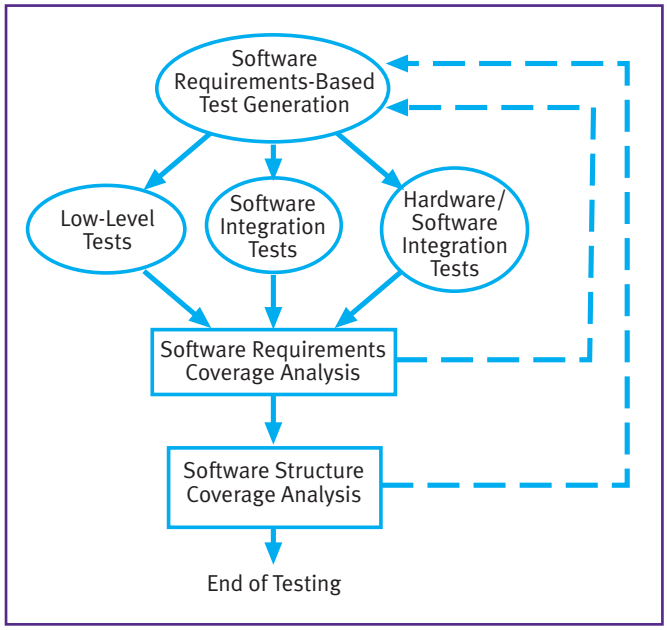
Critical software has many applications across the automotive industry, as well as other commercial and government sectors of the global economy. There are safety critical, mission critical, and business critical applications, to name a few. A common grouping of such applications is presented below:

The breadth of these software applications is even greater if one considers "consumer critical" uses, including ATM and gaming machines (especially if it's your money!). Most of these applications are developed for industries and governmental organizations that define and publish their own software development and testing standards. The following list is representative of such standards:

- **MISRA:** Development Guidelines for Vehicle Based Software, 3.6, "Testing"
- **IEEE 1012:** Standard for software verification and validation
- **IEEE 829:** Standard for software test documentation
- **IEC 61508:** Functional safety of electrical/electronic/programmable safety-related systems
- **FDA:** General principles of software validation, 5.2.5, "Testing by the software developer"
- **EN 50128:** Railway applications, "Software for railway control and protection systems"
- **RTCA DO-178B:** Software considerations in airborne systems and equipment certification requirements, 6.x, "Software Verification Process"
- **Def Stan 00-55:** Requirements for safety related software in (Part 2) defense equipment, Section 5, "Testing and integration"



A common thread among all these standards is an enunciated need to perform requirements-based testing. Prominent among these standards is the standard for airborne systems, DO-178B. This standard identifies two primary activities of requirements based testing as functional or black box testing (as shown below) and structural coverage or white box testing.



The functional testing activity requires that the developer or tester have access to the software requirements that specify the behavior of the code under test. More explicitly, the developer (or test engineer) must define the input values and conditions together with the outputs or expected results in order to create the test specification. This test specification may result in the formation of one or more test cases in order to fully exercise the requirements.

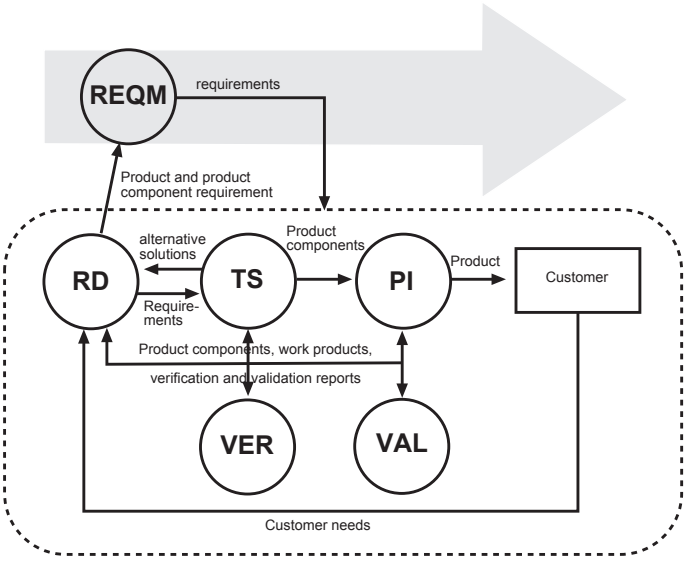
The structural coverage or white box activities help to validate the completeness of the black box testing. Structural coverage will also help determine the correctness of the as-built design; for example, if required software functionality is exercised and yet there is still uncovered code, then the purpose of the additional code comes into question, as does the predictability of the code's run time behavior.

Capability Maturity Model Integration (CMMI)

The Capability Maturity Model Integration standard improves processes in developing automotive electronics software, which leads to unique tools for mapping test information to requirements.

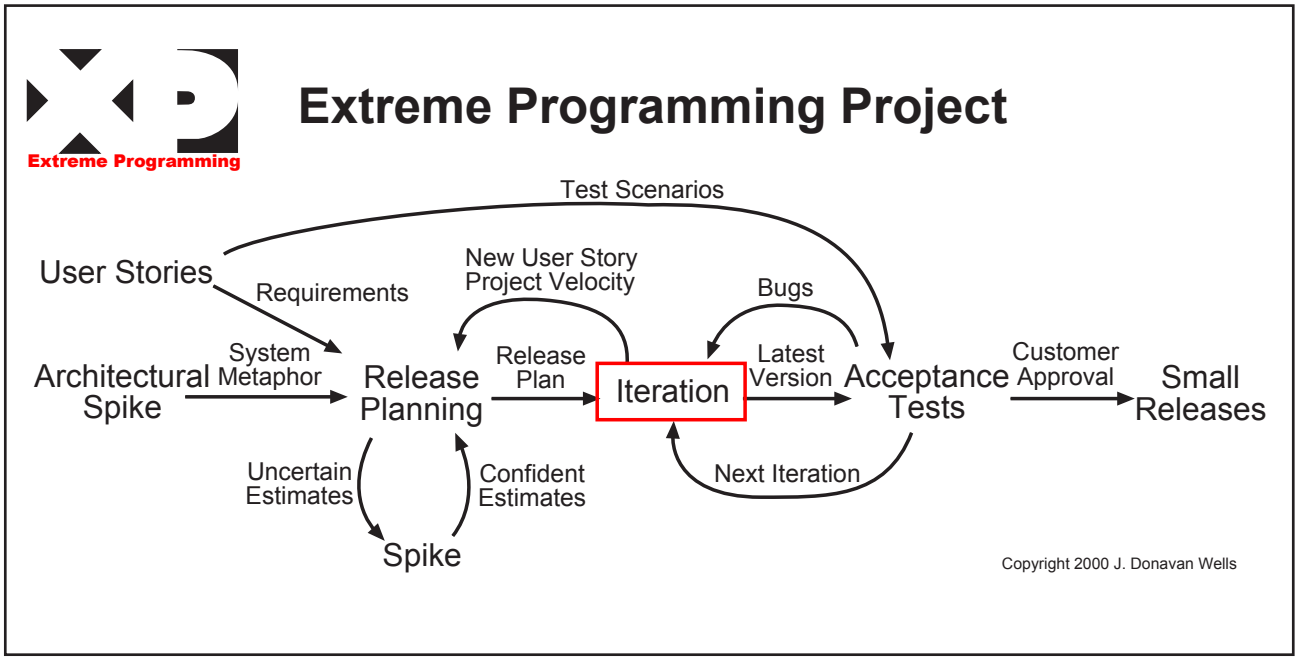
Requirements based testing, and its inherent process of requirements traceability and verification, is widely viewed as a best practice that is promulgated in corporate standards, such as the Capability Maturity Model Integration. CMMI is a process improvement approach that provides organizations with the essential elements of effective processes. It can be used to guide process improvement across a project, a division, or an entire organization. The benefits of CMMI have been established for critical as well as non-critical software.

As shown in the engineering process areas in the figure below, CMMI is predicted on the principals of requirements management (REQM) and requirements development (RD).



In the figure above, the technical solution (TS) is the elaboration of the requirements into prototypes or components. The verification process area (VER) ensures that selected work products meet the specified requirements. The validation process area (VAL) incrementally validates products against the customer's needs. Validation may be performed in the operational environment or a simulated operational environment.

Finally, from the programming standards perspective, processes such as Extreme Programming, along with requirements-based development and testing are integral to all development activities. With Extreme Programming, as illustrated on page 5, user "stories" (i.e. use cases) are prepared in co-operation with the customer as the pretext for the test scenarios before the code is developed (iteration).

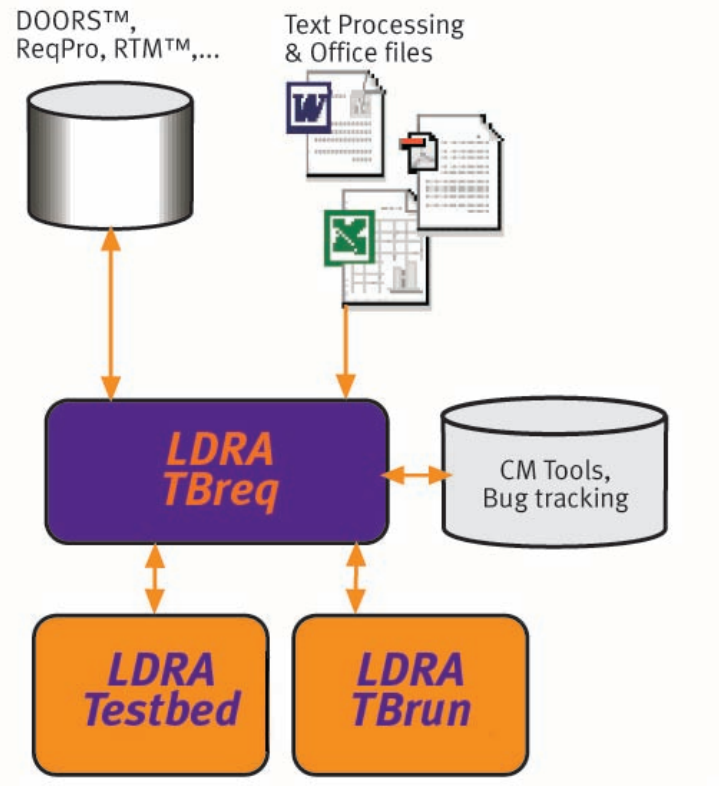


Introduction to TBreq

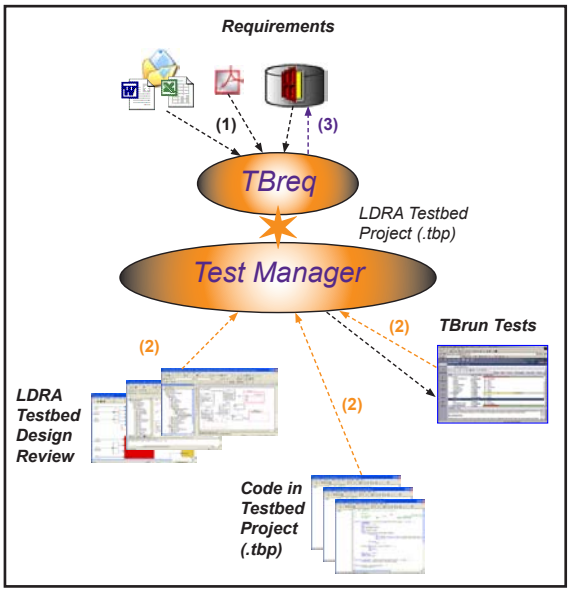
TBreq, through its integration with the LDRA tool suite, which is comprised of LDRA Testbed (which includes Code Review, Quality Review, and Design Review components, as well as code coverage) and TBrun (a unit testing component), is a unique solution that can help overcome the challenges of mapping test specifications, unit test scenarios, test data, and code coverage verification with high level and design requirements.

TBreq interfaces directly with requirements management tools (DOORS, ReqPro, Word or Excel) to ensure traceability across the software lifecycle and the completeness of requirements coverage (see opposite).

Within the LDRA tool suite, TBreq creates test specifications and executable test cases directly from requirements. Test results are automatically returned to the requirements management tool to provide “round-trip” requirements traceability verification.



TBreq operations are depicted below. Requirements can be captured from requirements management tools such as DOORS or ReqPro or from a document or spreadsheet. TBreq acts as a gateway to these requirement sources for the LDRA Testbed Test Manager Dashboard, and interfaces directly with a LDRA Testbed Project and its underlying project directories.



Requirements are captured from any source. They are made available to Test Manager (via LDRA Testbed for traceability and verification). Traceability and Requirements mapping are performed directly in LDRA Testbed and information is captured from Design Review, source code files, and TBrun. Verification results and traceability information can be uploaded into repositories.

The TBreq software performs two basic types of workflow. The first includes requirements traceability and test verification through low-level requirements and the as-built Design Review. The Test Manager supports the mapping of requirements with source code procedures or methods. These mapped requirements are subsequently made available to the developer or tester for the purposes of test specification creation and test verification. Test Manager will also facilitate the automatic creation of test cases from these test specifications. (Subsequent releases will support the automated input of test values from data tables or specifications.) The results of this workflow can then be mapped back to the requirements sources.

The package can also be used for test verification without TBrun. In this workflow scenario, LDRA Testbed is used to instrument source code that is executed by a customer-provided test harness.

TBreq also uses a mechanism called a Requirements Descriptor Thread (or Thread) to facilitate agile traceability and verification capabilities. The properties of the thread are:

- **File Specification** - Source code or skeleton file name
- **Requirement Nomenclature** - Requirement name and number, Requirement source document
- **Requirement Body** - Requirement text
- **Test Configuration** - Associated test case/sequence, Coverage levels, Test case/sequence verification status
- **Test Specification** - Procedure(s) or class interface(s) Test data
- **Test Management** - Project manager name, Developer/tester name, Thread type (RV or DV)

A thread is created for each high-level (system) requirement and for each low-level (design) requirement. The former thread type is called a requirement verification (RV) thread; the latter is called a design verification (DV) thread. A thread contains the requirement name and number as well as the requirement body (text). A thread contains the mapping information including the source code's File Specification and the associated procedure prototype (test specification); the associated test case mapping is provided under test configuration as well as the required coverage level (e.g. statement 100%, branch 80%).

Conclusion

Software TBreq provides a comprehensive, integrated solution for requirements traceability and verification challenges. Moreover, the package, in conjunction with the LDRA tool suite, provides full compliance with the critical software standards discussed. And, with respect to CMMI Level 2 (requirements management) and CMMI Level 3 (requirements development), TBreq offers the process infrastructure mandated by this standard.

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