



TBreq™:

**An Integrated Solution for
Requirements-Based Testing**

www.ldra.com

Key Challenges

A common request from LDRA customers, including aerospace, defense, medical, telcom and many other market sectors, entails the production of an automated solution for requirements traceability. The need for requirements traceability is typically imposed on our customers as a contractual requirement. Though now, with increasing frequency, our customers are recognising 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). The TVM is 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 our customers attempt to maintain the TVM through testing, integration and deployment phases of their projects. The inherently brittle nature of the TVM and the manual processes it encapsulates are exposed as defects occur which 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 assessment (or reassessment) of the TVM encompasses what are called Impact and Gap analyses; Impact analysis determines the impact of requirement change on a system. This assumes a hierarchy (or association) of requirements that must be traced in order for change to be properly managed. Gap analysis determines where the mapping of requirements is incomplete or inconsistent. Our customers have learned that these types of analyses are very difficult and expensive to perform when predicated on a manually generated and maintained TVM.

The next challenge is to produce a requirements traceability solution that is dedicated to development and test teams. Our customers stipulate that this solution must operate in the context of existing tools and processes. Currently, most customers have a requirements database or flat file capability where they define and maintain system or high-level requirements. 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, and one that is frequently required of our customers, 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, I would argue, 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.¹

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.

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.

¹ 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 modelling tool. This option is described in another LDRA white paper called LDRA Tool Suite/ Telelogic I-Logix Rhapsody Integration).

Standards Compliance for Critical Software

Critical software has many applications across government and commercial 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 in Figure 1: Critical Software Applications.

The breadth of these software applications is even greater if one considers “Consumer Equipments” as including ATM machines and gaming machines (especially if it’s your money!). Most of these applications are developed for industries and governmental

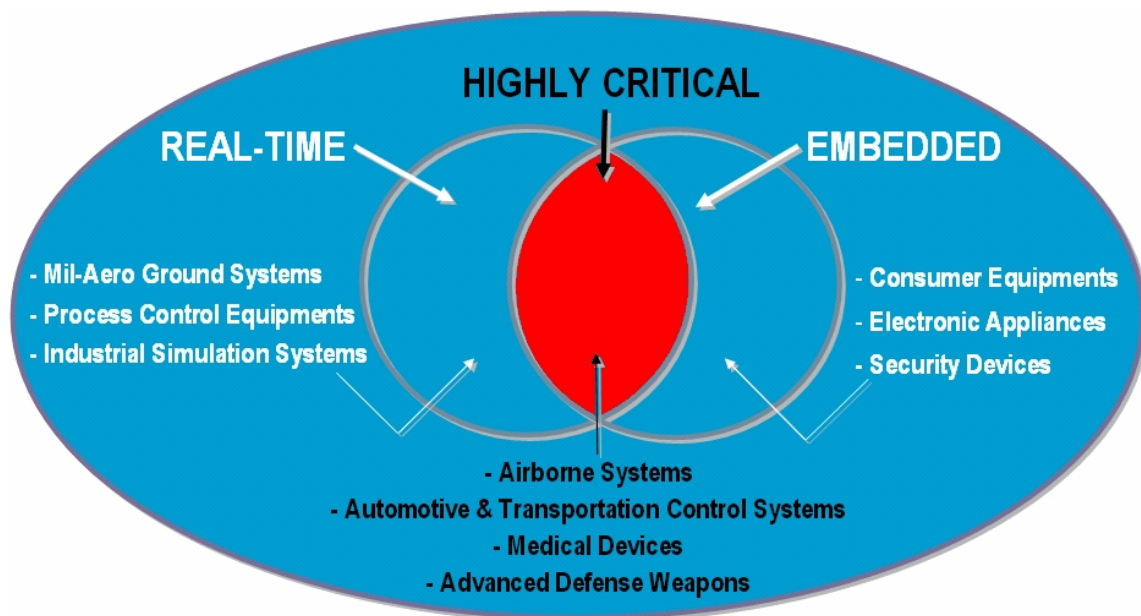


Figure 1: Critical Software Applications

organisations that define and publish their own software development and testing standards. The following list is representative of these standards.

- **FDA** **General principles of software validation, 5.2.5 “Testing by the software developer”**
- **IEC 60880** **Software for computers important to safety for nuclear power plants. Part 2, “Software aspects of defense against common cause failures and use of software tools”**
- **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 (Part2)** Requirements for safety related software in defense equipment, Section 5 “Testing and integration”
- **MIL STD 498** Software Development and Documentation, 5.7 + 5.11 “Software Testing”
- **MISRA** Development Guidelines for Vehicle Based Software, 3.6 “Testing”
- **IEEE 1008** Standard for software unit 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

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 in Figure 2: Testing Activities) and structural coverage or White Box testing.

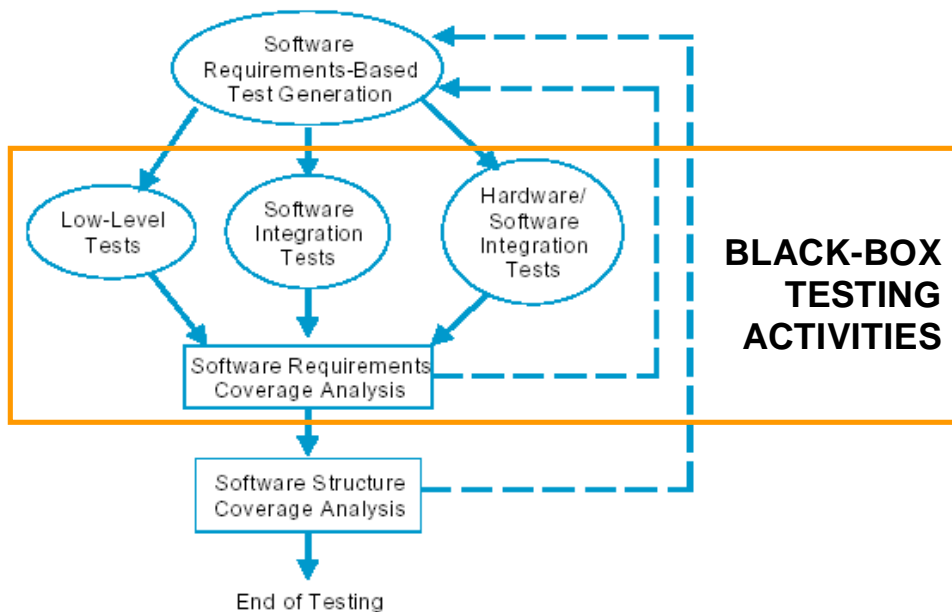


Figure 2: Testing Activities

The functional testing activity requires that the developer or tester have access to the software requirements that specify the behaviour 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 behaviour.

Requirements based testing, and its inherent process of requirements traceability and verification, is also 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 organisation. The benefits of CMMI have been established for critical as well as non-critical software as shown in Appendix A.²

As shown in Figure 3: Engineering Process Areas, CMMI is predicted on the principals of requirements management (REQM) and requirements development (RD).

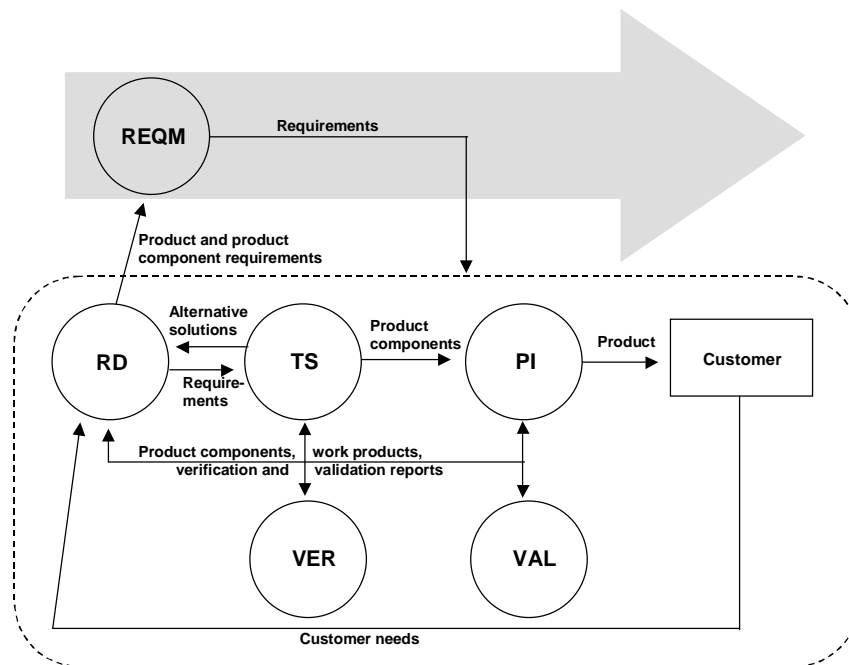


Figure 3: Engineering Process Areas

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

² The terms "Critical software" as defined in this paper applies to software developed for standards that dictate levels of criticality for testing purposes, only.

³ Capability Maturity Model® Integration (CMMISM), Version 1.1, CMMISM for Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing (CMMI-SE/SW/IPPD/SS, V1.1), 2002

validates products against the customer's needs. Validation may be performed in the operational environment or a simulated operational environment.³

Finally, from the programming standards perspective, processes such as Extreme Programming, requirements based development and testing are integral to all development activities. As illustrated in Figure 4: Extreme Programming Project, 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).

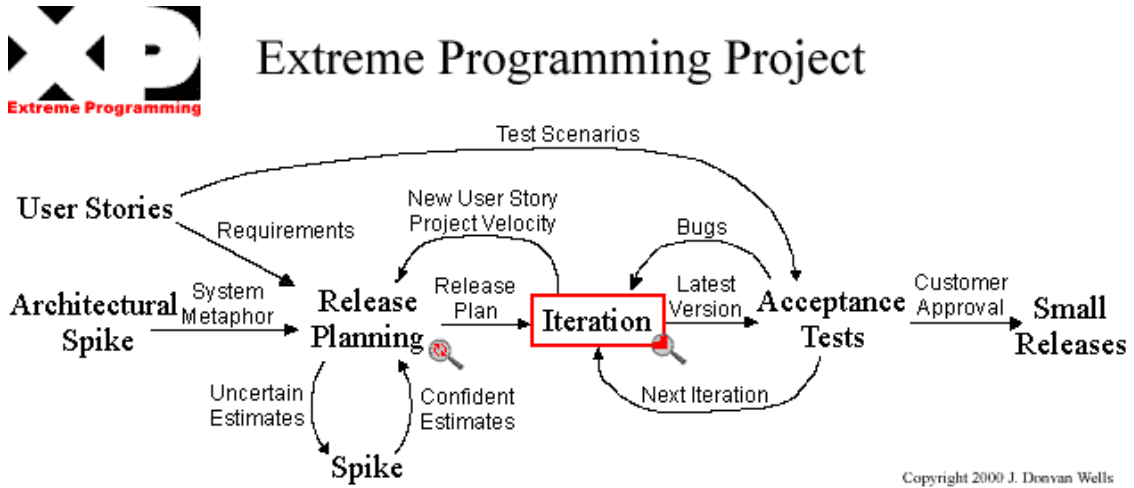


Figure 4: Extreme Programming Project

Introduction to TBreq

TBreq, through its integration with the LDRA tool suite which includes LDRA Testbed and TBrin (unit testing component), is a unique solution that can help your team overcome the challenges of mapping test specifications, unit test scenarios, test data and code coverage verification with your high level and design requirements. **TBreq** interfaces directly with your management tool (DOORS, ReqPro, Word or Excel) to ensure traceability across your software lifecycle and the completeness of your requirements coverage. Refer to Figure 5: TBreq Integration.

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.”

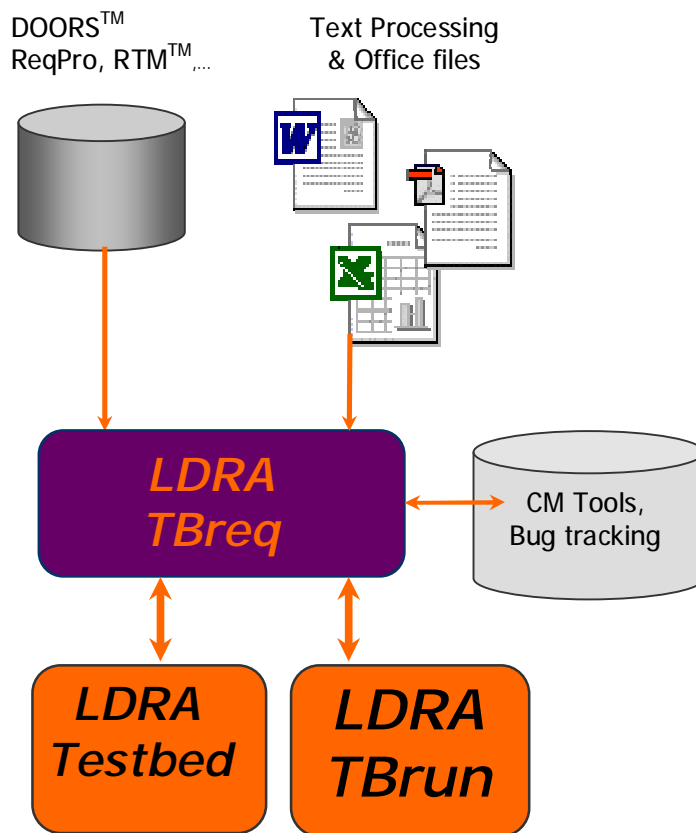


Figure 5: TBreq Integration

TBreq Operations

TBreq operations are depicted in Figure 6. 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. TBreq interfaces directly with a LDRA Testbed Project and its underlying project directories.

TBreq 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 of TBreq 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.

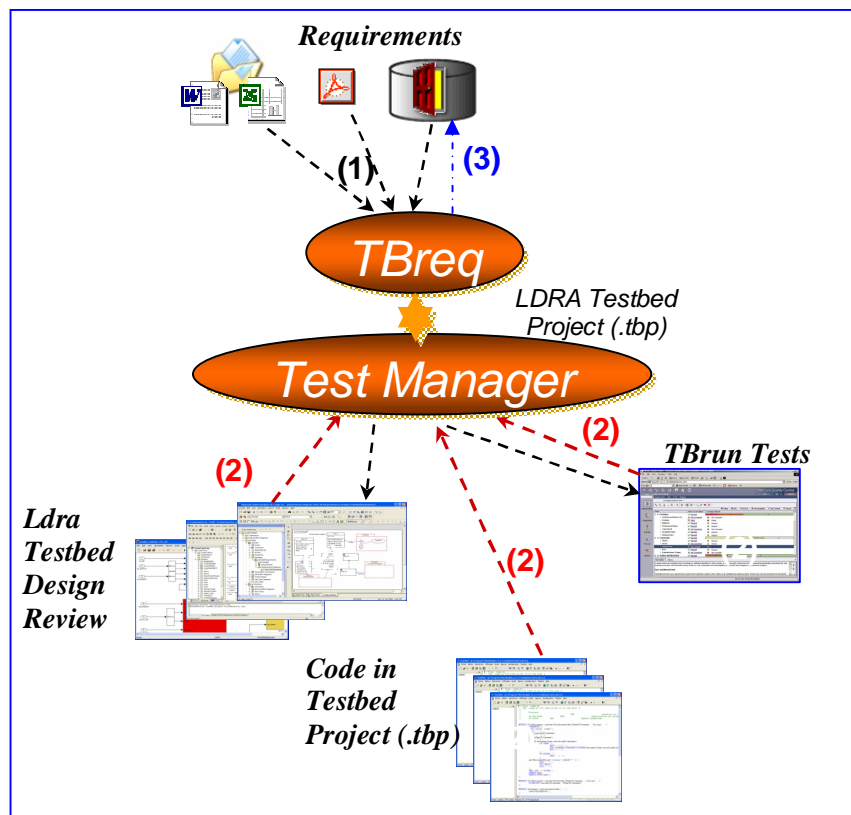


Figure 6: TBreq Operations

Legend:

- (1) Requirements are captured from any source. They are made available to Test Manager (via a LDRA Testbed Project for traceability and verification).

-
- (2) Traceability and Requirements mapping are performed directly in LDRA Testbed; information is captured from Design Review, Source code files and TBrn.
 - (3) Verification results and traceability information can be uploaded into repositories.

A second type of workflow includes the mapping of requirements directly to test cases created in TBrn. In this workflow process, requirements are mapped to previously created test cases and source code mappings are implicit. This process defers requirements traceability to later in the development cycle and can not utilise the automated test specification capabilities of TBreq.

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

This paper will provide an example of the TBreq integration with DOORS. However, to better understand the specifics of this example a short discussion on a key TBreq architectural feature is in order.

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

Requirements Descriptors Thread (RDT)

- **File Specification**
 - Source Code or Skeleton file name
- **Requirement Nomenclature**
 - Requirement Name & 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%).

Access to a thread is role based. A Project Manager accesses the Test Management properties (and thereby allocates requirements to a Developer/Tester.) The Project Manager can also set the Coverage Levels for the Developer/Tester. The Developer/Tester completes the Test Specification; this role also indicates the Test Case/Sequence Verification Status.

Operational Scenario

Selection of the DOORS Module

In the Project Editor of TBreq, select a type of analysis based on DOORS, click in the File field and on the Browse button.

TBreq then presents you with a view of the DOORS database, allowing the selection of the formal module you want to import.

Provide valid login information (User + Password) for DOORS and click OK. (Figure 7) This login and subsequent steps are performed as a Project Manager.

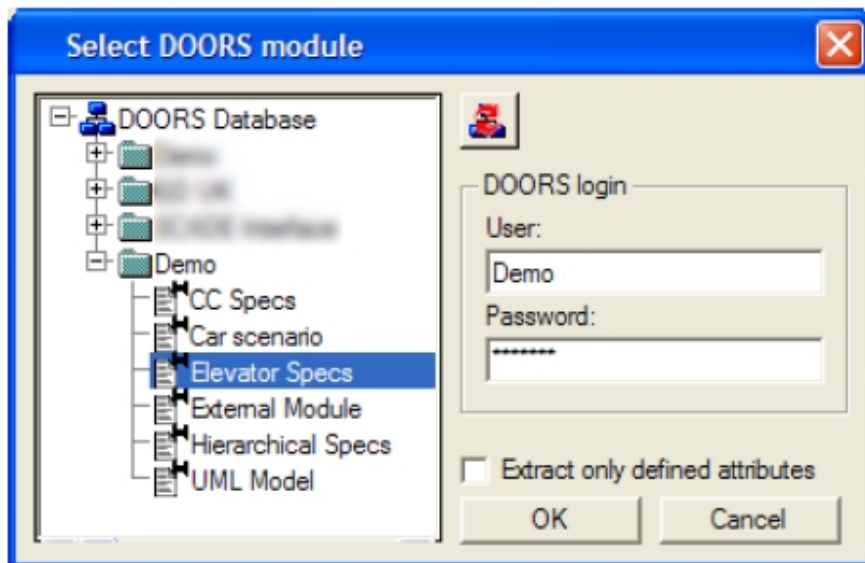


Figure 7: DOORS Login

A depiction of the coupling of TBreq with DOORS is provided in Figure 8. After a successful login, choose the appropriate module from the DOORS database. Create a DOORS requirements document in TBreq and then create an LDRA Testbed project. Associate the LDRA Testbed project with the DOORS requirement and thereby allocate the requirements in the DOORS module to the LDRA Testbed project. All Requirements Traceability and Test Verification activities will be traced by TBreq.

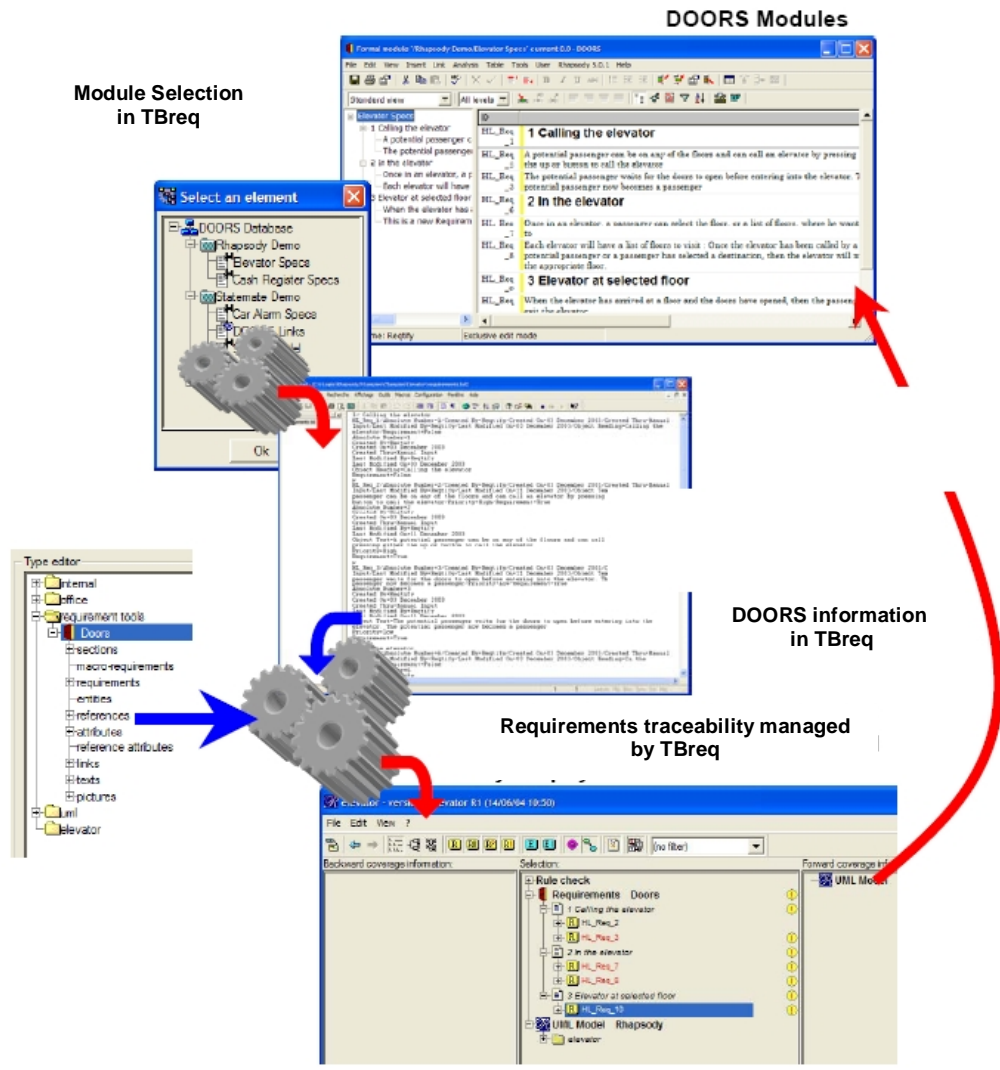
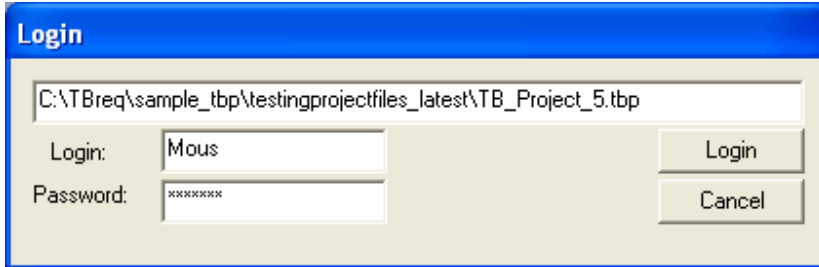


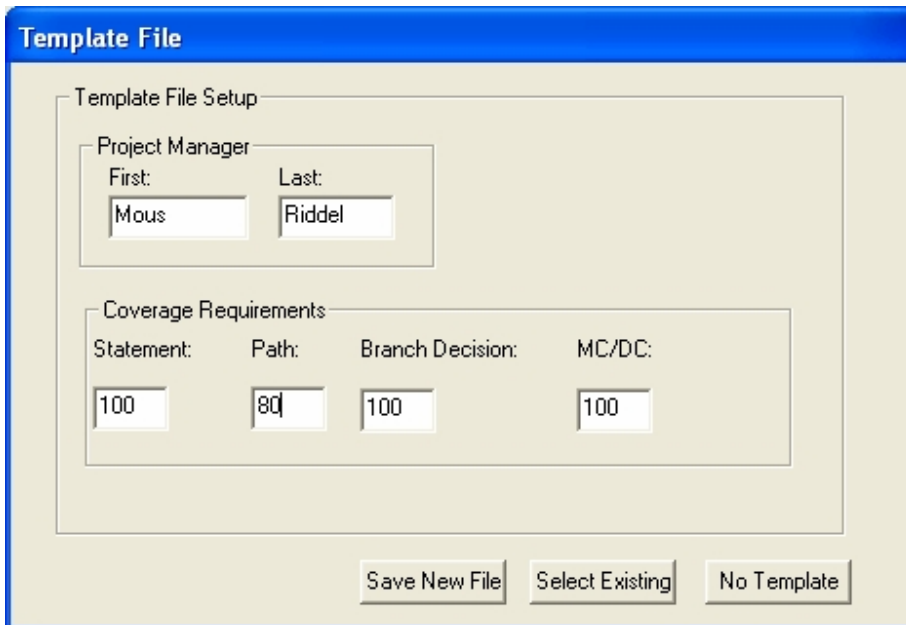
Figure 8: TBreq and DOORS Coupling

From the Test Manager Dashboard (Dashboard) the Project Manager opens the LDRA Testbed project. The Dashboard prompts for a Login



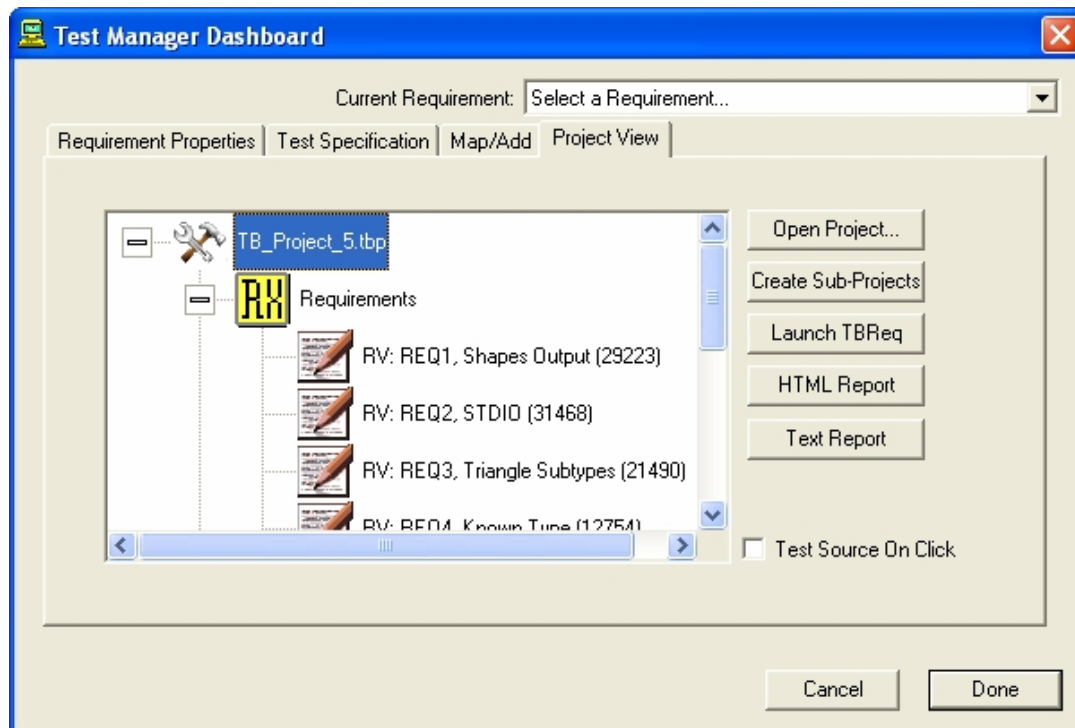
The Login dialog box has a blue title bar with the text "Login". Below the title bar is a text field containing the file path "C:\TBreq\sample_tbp\testingprojectfiles_latest\TB_Project_5.tbp". Underneath are two input fields: "Login:" with the text "Mous" and "Password:" with "*****". To the right of these fields are two buttons: "Login" and "Cancel".

Because this is a new LDRA Testbed project, the Project Manager is prompted to complete a template that will be used to initialise each thread created.

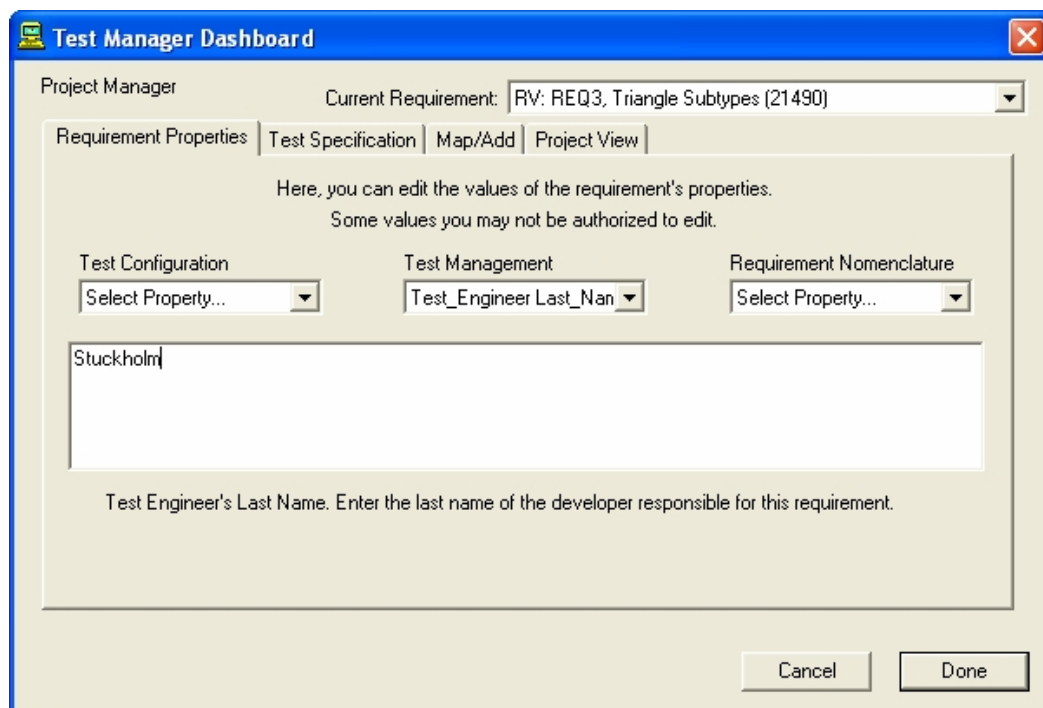


The Template File dialog box has a blue title bar with the text "Template File". The main area is titled "Template File Setup" and contains two sections. The first section, "Project Manager", has "First:" and "Last:" labels with input fields containing "Mous" and "Riddel" respectively. The second section, "Coverage Requirements", has four labels: "Statement:", "Path:", "Branch Decision:", and "MC/DC:", with input fields containing "100", "80", "100", and "100" respectively. At the bottom of the dialog are three buttons: "Save New File", "Select Existing", and "No Template".

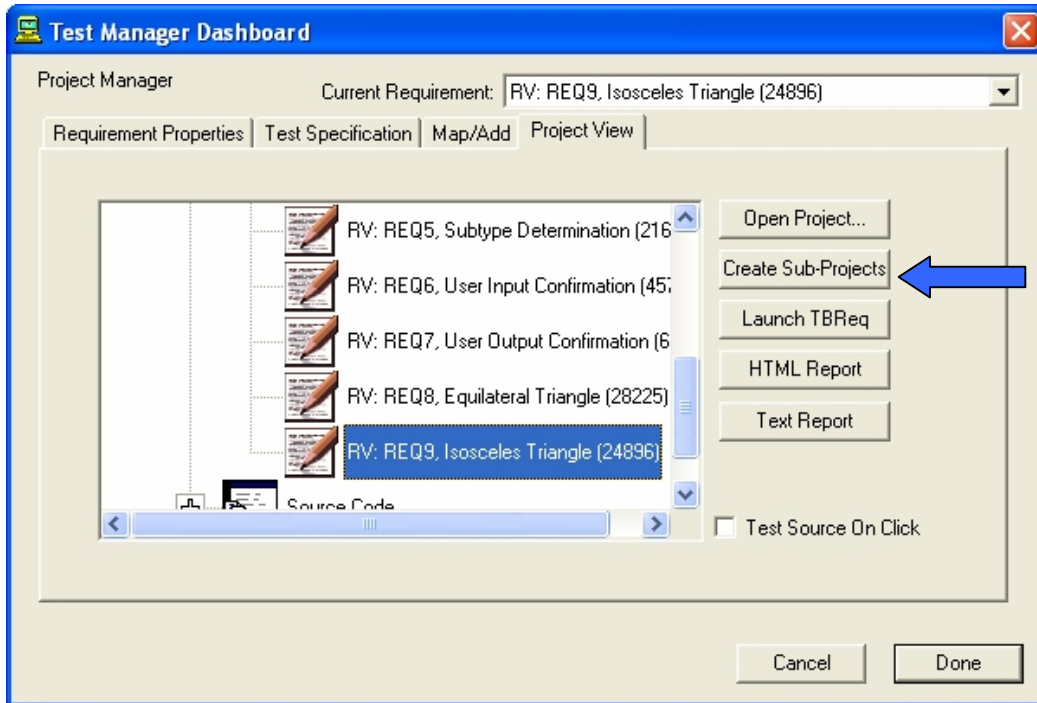
The Project Manager saves the template and the requirements allocated to the project appear in the Project View.



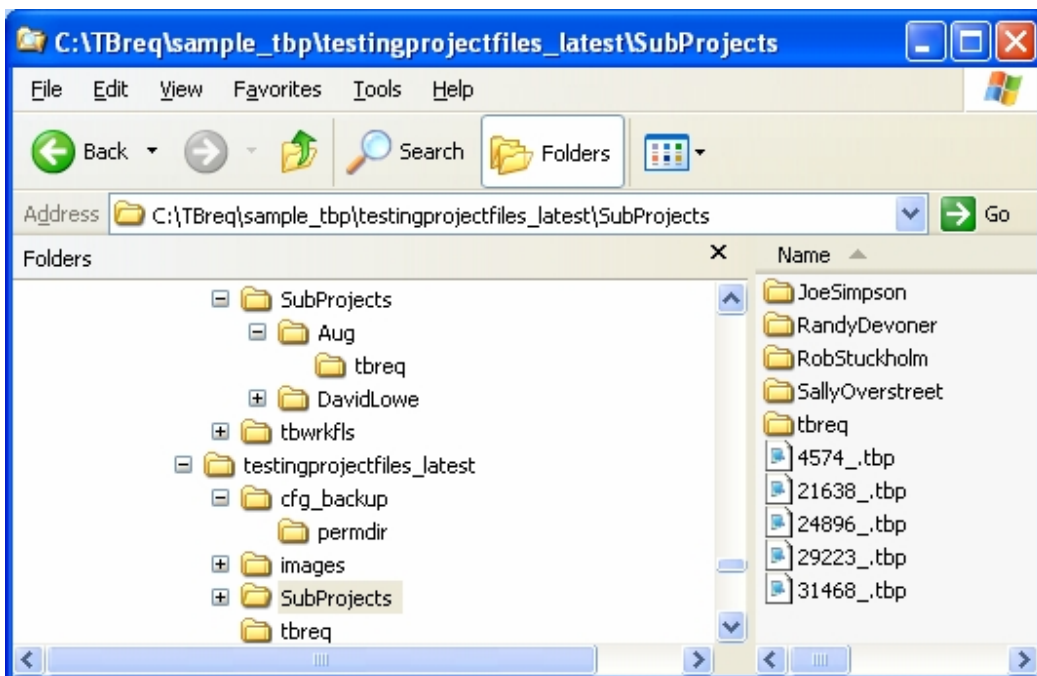
The Project Manager now allocates requirements to individual Developers/Testers. First they select the requirement to be allocated as the Current Requirement and then select the Requirements Properties tab. In this tab they enter the name of the Developer/Tester. (Notice that the Project Manager role is identified by the Test Manger Dashboard in the upper left hand corner.)



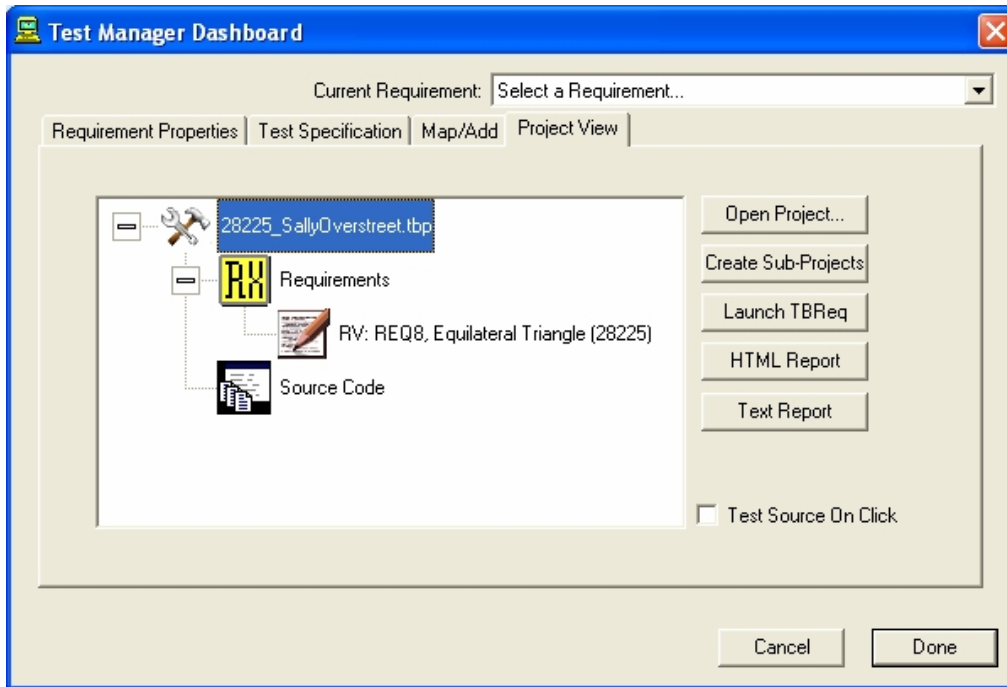
After allocating all requirements to Developers/Testers, the Project Manager creates the Subprojects. A Subproject is the LDRA Testbed workspace for the Developer/Tester.



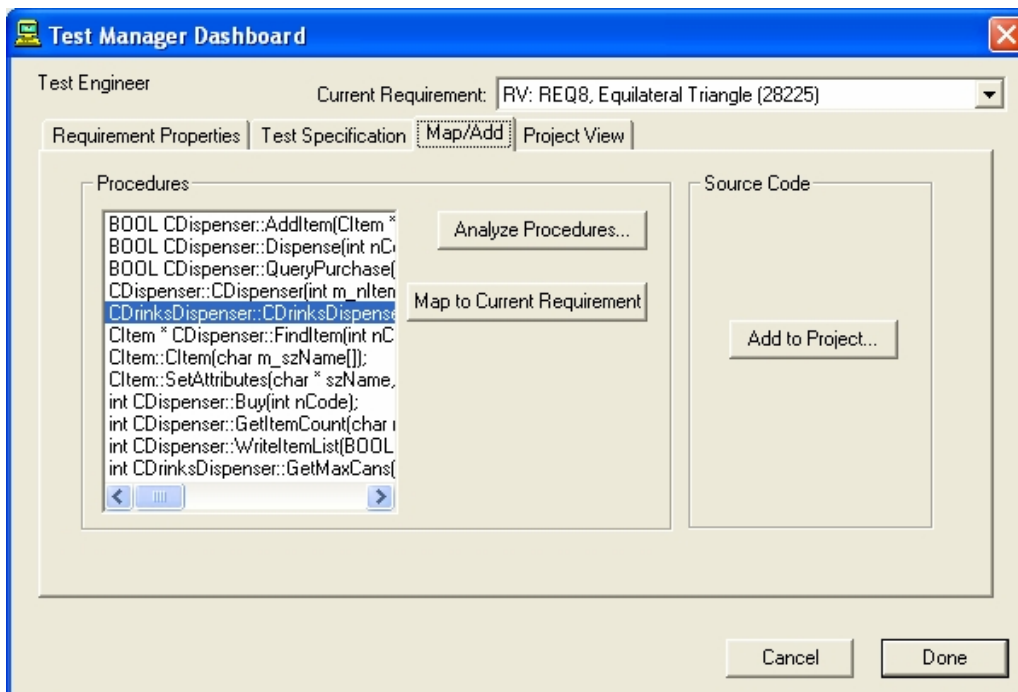
Selecting “Create Sub-Projects” automatically creates Sub-project folders.



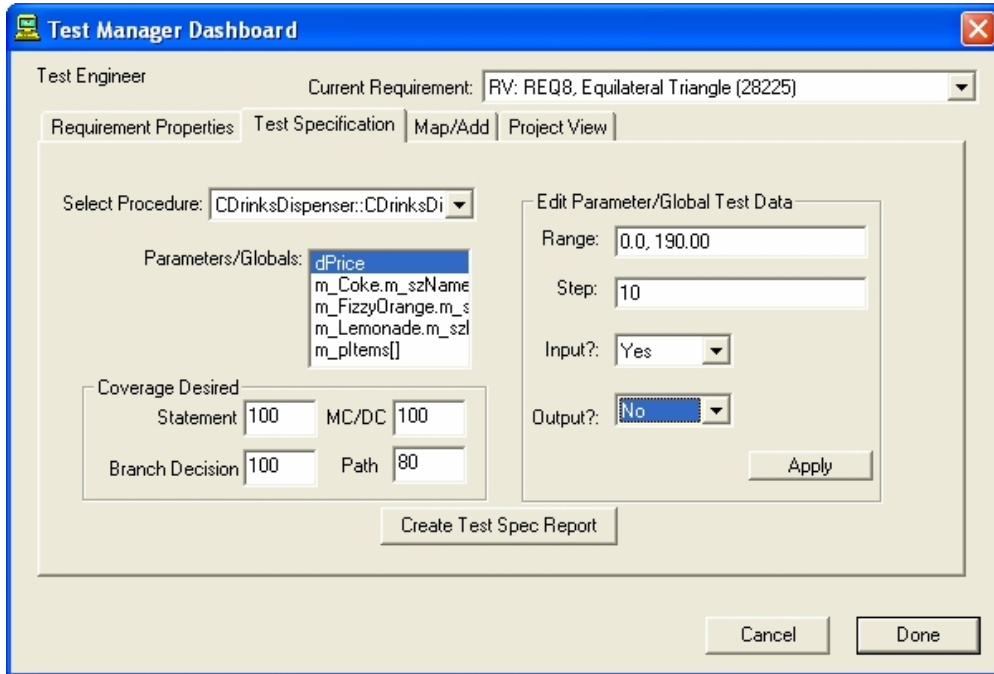
Now the Developer/Tester can open his or her Sub-project. Again, an optional login is provided. In this example, Sally Overstreet has opened her Sub-project.



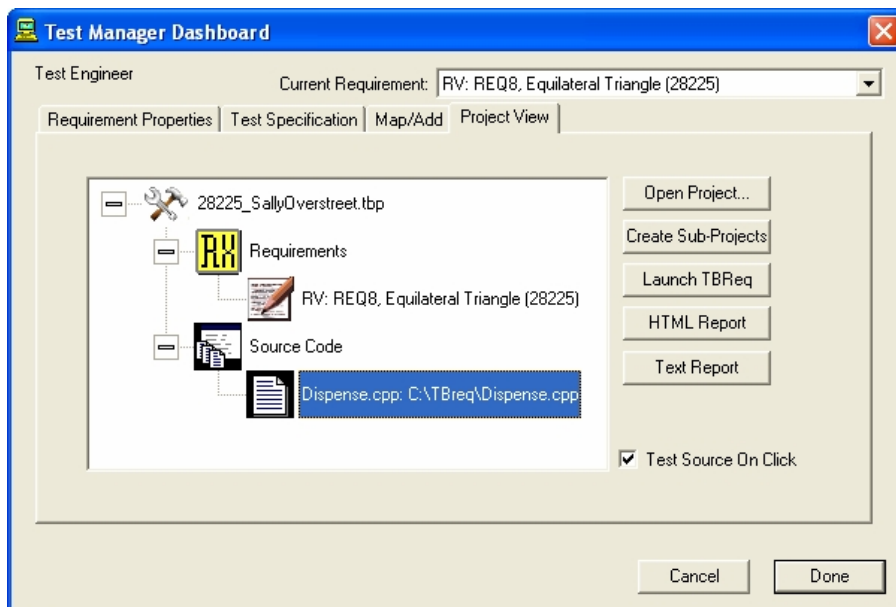
Sally adds her source code to the Sub-project and analyses the procedures. Next, she maps the Current Requirement to the highlighted procedure.



Sally next defines a Test Specification for the mapped procedure. In this snapshot she is defining a range of 0.00 to 190.00, with a step function of 10 for the input parameter, dPrice. The Coverage requirements were inherited from the Thread Template created by the Project Manager. The Test Specification can be used by Sally to automatically create test cases in TBrun.



Having completed her Test Specification, Sally next proceeds to launch LDRA Testbed/TBrun to build and execute the test cases that will be automatically created for her.



Appendix A CMMI Results

Results (reported as of 15 December 2005)

You can view examples of CMMI performance results by organization or by performance category.

The following table contains a summary of the performance results:

Performance Category	Median	Number of Data Points	Low	High
Cost	20%	21	3%	87%
Schedule	37%	19	2%	90%
Productivity	62%	17	9%	255%
Quality	50%	20	7%	132%
Customer Satisfaction	14%	6	-4%	55%
Return on Investment	4.7 : 1	16	2 : 1	27.7 : 1

This table summarises quantitative information from 25 organisations that have reported results that can be expressed as performance changes over time. Additional qualitative results from 5 other organisations are available when you view examples by organisation or performance category.

Contact Details:

LDRA Headquarters
Portside, Monks Ferry,
Wirral, CH41 5LH, UK
Tel: +44 (0)151 649 9300

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

E-mail: info@ldra.com