

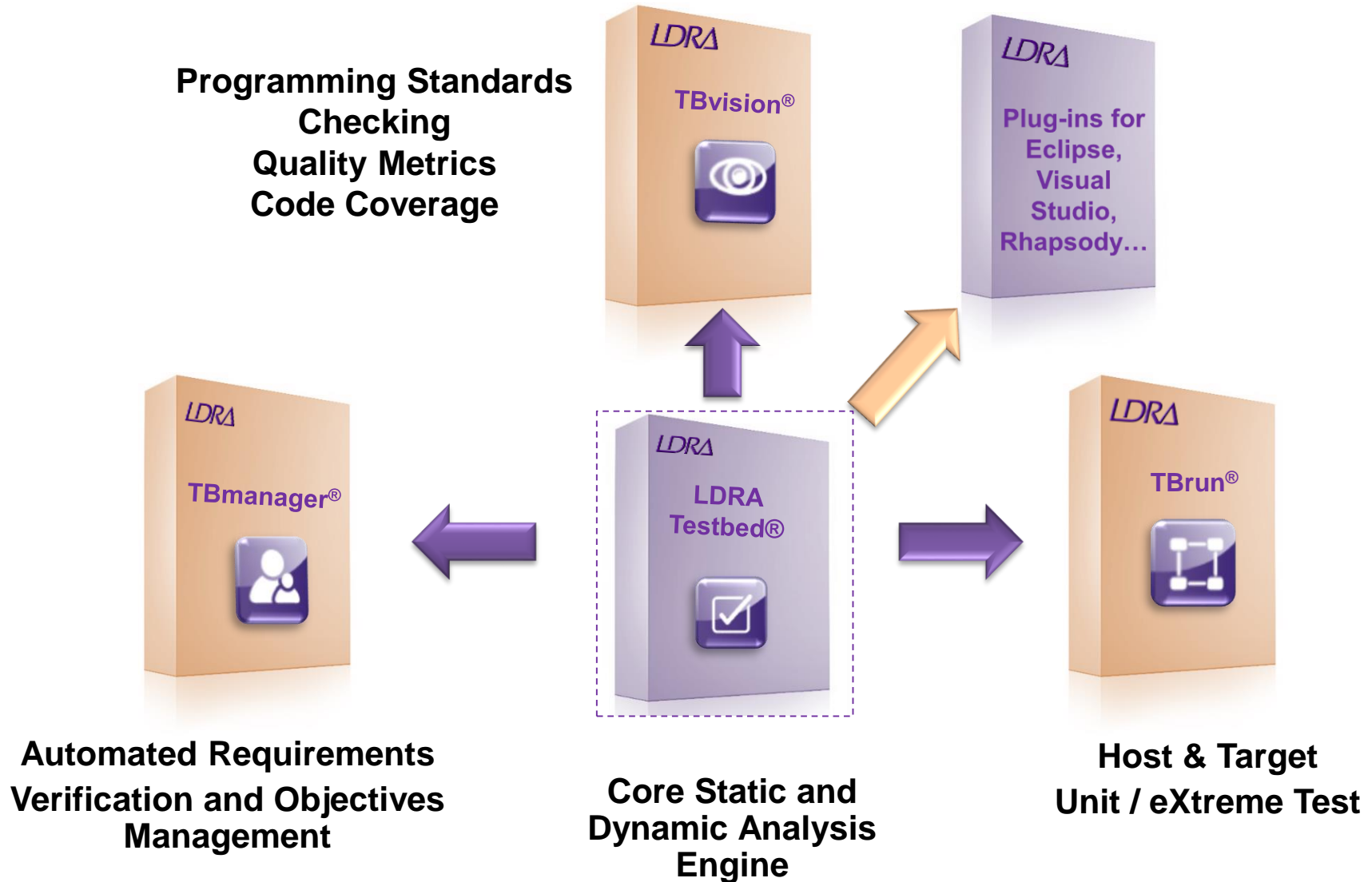


Meeting DO-178C Objectives with the LDRA tool suite



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

LDRA tool suite® Integrated Solution



DO-178C Objectives

Objectives (4/71 Fulfilled)

- ✖ A-1.1 - The activities of the software life cycle processes are defined - Fulfilled - 4 artifacts
- ✖ A-1.2 - The Software Life cycle(s), including the inter-relationships between processes, their sequence, and the software life cycle environment is defined - Fulfilled - 3 artifacts
- ✖ A-1.3 - Software life cycle environment is defined - Fulfilled - 3 artifacts
- ✖ A-1.4 - Additional considerations are addressed - Fulfilled - 1 artifact
- ✖ A-1.5 - Software development standards are defined - Partial - 1 artifact
- ✖ A-1.6 - Software plans comply with this document - Unfulfilled
- ✖ A-1.7 - Development and revision of software plans are coordinated - Unfulfilled
- ✖ A-2.1 - High-level requirements are developed. - Partial - 1 asset
- ✖ A-2.2 - Derived high-level requirements are defined and provided to the system processes, including the system safety assessment - Unfulfilled
- ✖ A-2.3 - Software architecture is developed. - Partial - 1 asset
- ✖ A-2.4 - Low-level requirements are developed. - Partial - 1 asset
- ✖ A-2.5 - Derived low-level requirements are defined and provided to the system processes, including the system safety assessment - Unfulfilled
- ✖ A-2.6 - Source Code is developed. - Partial - 1 asset
- ✖ A-2.7 - Executable Object Code and parameter data item files, if any are produced and loaded in - Partial - 1 asset
- ✖ A-8.1 - Configuration items are identified - Unfulfilled
- ✖ A-8.2 - Baselines and traceability are established - Unfulfilled
- ✖ A-8.3 - Problem reporting, change, control, change review, and configuration status accounting are established - Unfulfilled
- ✖ A-8.4 - Archive, retrieval, and release are established - Unfulfilled
- ✖ A-8.5 - Software load control is established - Unfulfilled
- ✖ A-8.6 - Software life cycle environment control is established - Unfulfilled
- ✖ A-9.1 - Assurance is obtained that software development and integral process comply with approved software plans and standards - Unfulfilled
- ✖ A-9.2 - Assurance is obtained that transition criteria for the software life cycle processes are satisfied - Unfulfilled
- ✖ A-9.3 - Software conformity review is conducted - Unfulfilled
- ✖ A-10.1 - Communication and understanding between the applicant and the certification authority is established - Unfulfilled
- ✖ A-10.2 - The means of compliance is proposed and agreement with the Plan for Software Aspect of Certification is obtained - Unfulfilled
- ✖ A-10.3 - Compliance substantiation is provided - Unfulfilled

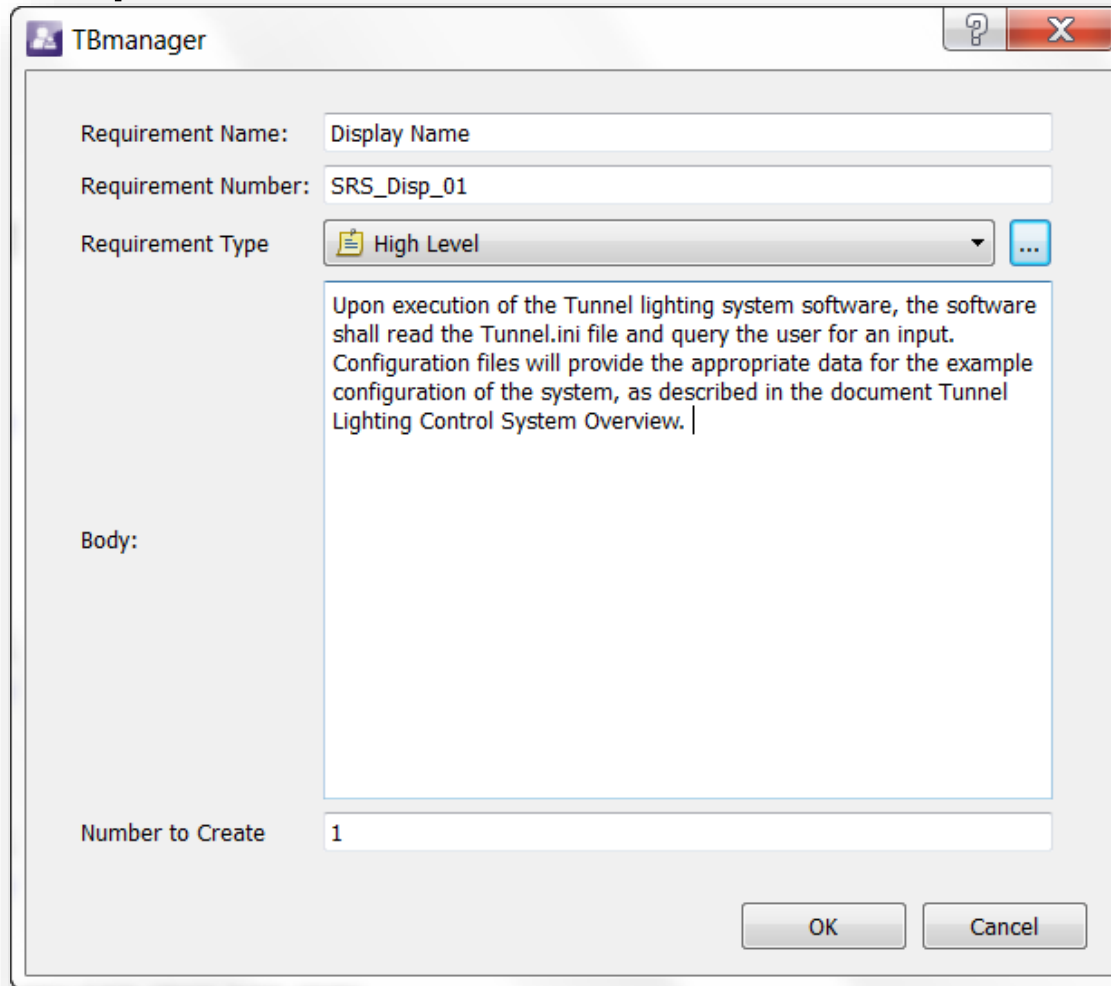


Objectives - Requirements

- + ✖ A-2.1 - High-level requirements are developed.
- ✖ A-2.2 - Derived high-level requirements are defined and provided to the system processes, including the system safety assessment process
- + ✖ A-2.3 - Software architecture is developed.
- + ✖ A-2.4 - Low-level requirements are developed.
- ✖ A-2.5 - Derived low-level requirements are defined and provided to the system processes, including the system safety assessment process
- + ✖ A-2.6 - Source Code is developed.
- + ✖ A-2.7 - Executable Object Code and parameter data item files, if any are produced and loaded in
 - ✖ A-3.1 - High-level requirements comply with system requirements
 - ✖ A-3.2 - High-level requirements are accurate and consistent
 - ✖ A-3.3 - High-level requirements are compatible with target computer
 - ✖ A-3.4 - High-level requirements are verifiable
 - ✖ A-3.5 - High-level requirements conform to standards
 - + ✖ A-3.6 - High-level requirements are traceable to system requirements
 - ✖ A-3.7 - Algorithms are accurate
- ✖ A-4.1 - Low-level requirements comply with high-level requirements
- ✖ A-4.2 - Low-level requirements are accurate and consistent
- ✖ A-4.3 - Low-level requirements are compatible with target computer
- ✖ A-4.4 - Low-Level requirements are verifiable
- ✖ A-4.5 - Low-level requirements conform to standards
- ✖ A-4.6 - Low-level requirements are traceable to high-level requirements
- ✖ A-4.7 - Algorithms are accurate

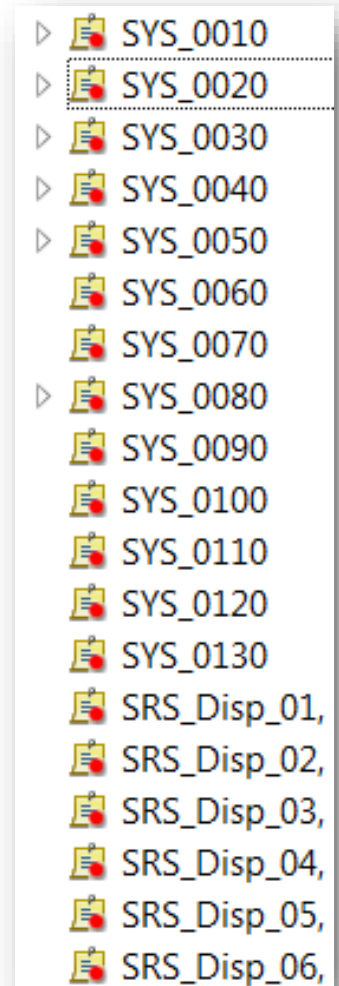
A 2.1 - High-level requirements are developed

- In TBmanager, we can create High Level Requirements.



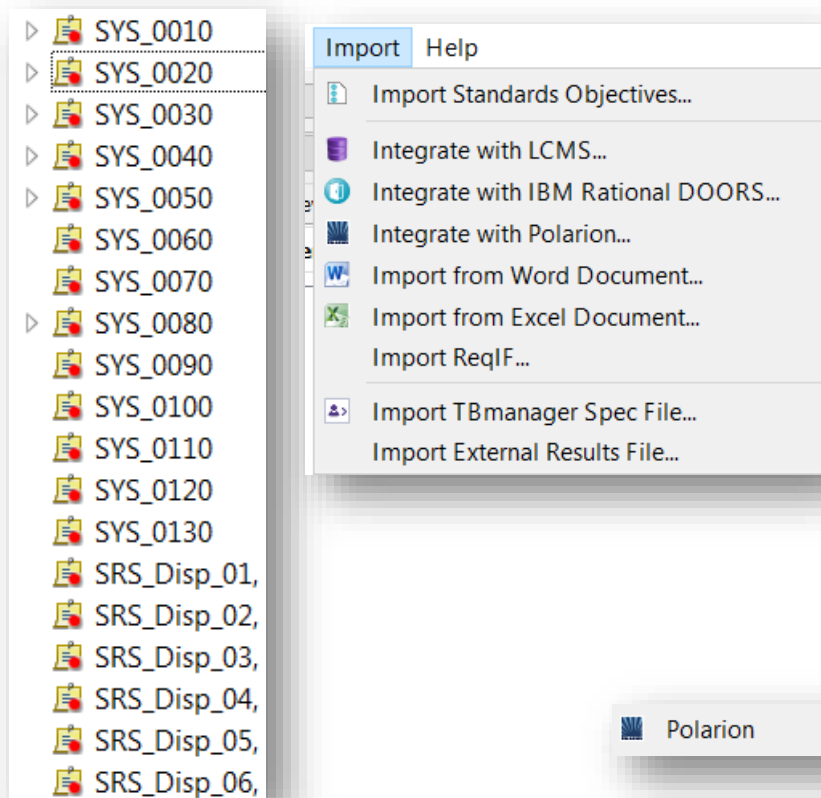
The screenshot shows the TBmanager dialog box with the following fields and content:

- Requirement Name:** Display Name
- Requirement Number:** SRS_Dispatch_01
- Requirement Type:** High Level (selected from a dropdown menu)
- Body:** Upon execution of the Tunnel lighting system software, the software shall read the Tunnel.ini file and query the user for an input. Configuration files will provide the appropriate data for the example configuration of the system, as described in the document Tunnel Lighting Control System Overview.
- Number to Create:** 1
- Buttons:** OK, Cancel

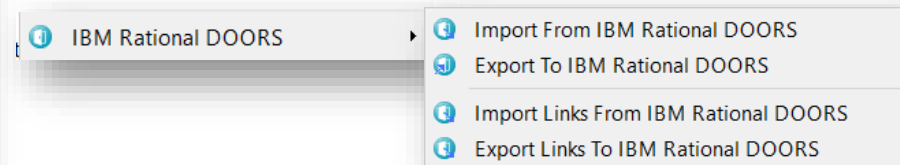
- 
- The screenshot shows a list of requirements in TBmanager, with the following items:
- SYS_0010
 - SYS_0020
 - SYS_0030
 - SYS_0040
 - SYS_0050
 - SYS_0060
 - SYS_0070
 - SYS_0080
 - SYS_0090
 - SYS_0100
 - SYS_0110
 - SYS_0120
 - SYS_0130
 - SRS_Dispatch_01,
 - SRS_Dispatch_02,
 - SRS_Dispatch_03,
 - SRS_Dispatch_04,
 - SRS_Dispatch_05,
 - SRS_Dispatch_06,

A 2.1 - High-level requirements are developed

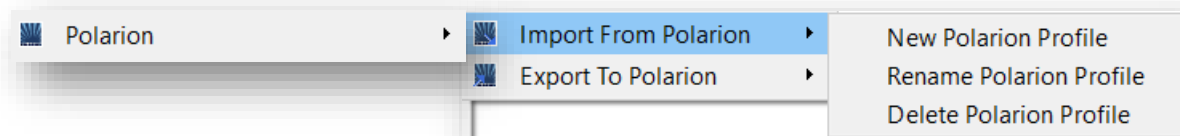
- TBmanager can import High-level requirements from Requirement management tools like DOORS, Polarian, Word, Excel, ReqIF



Importing requirements from DOORS

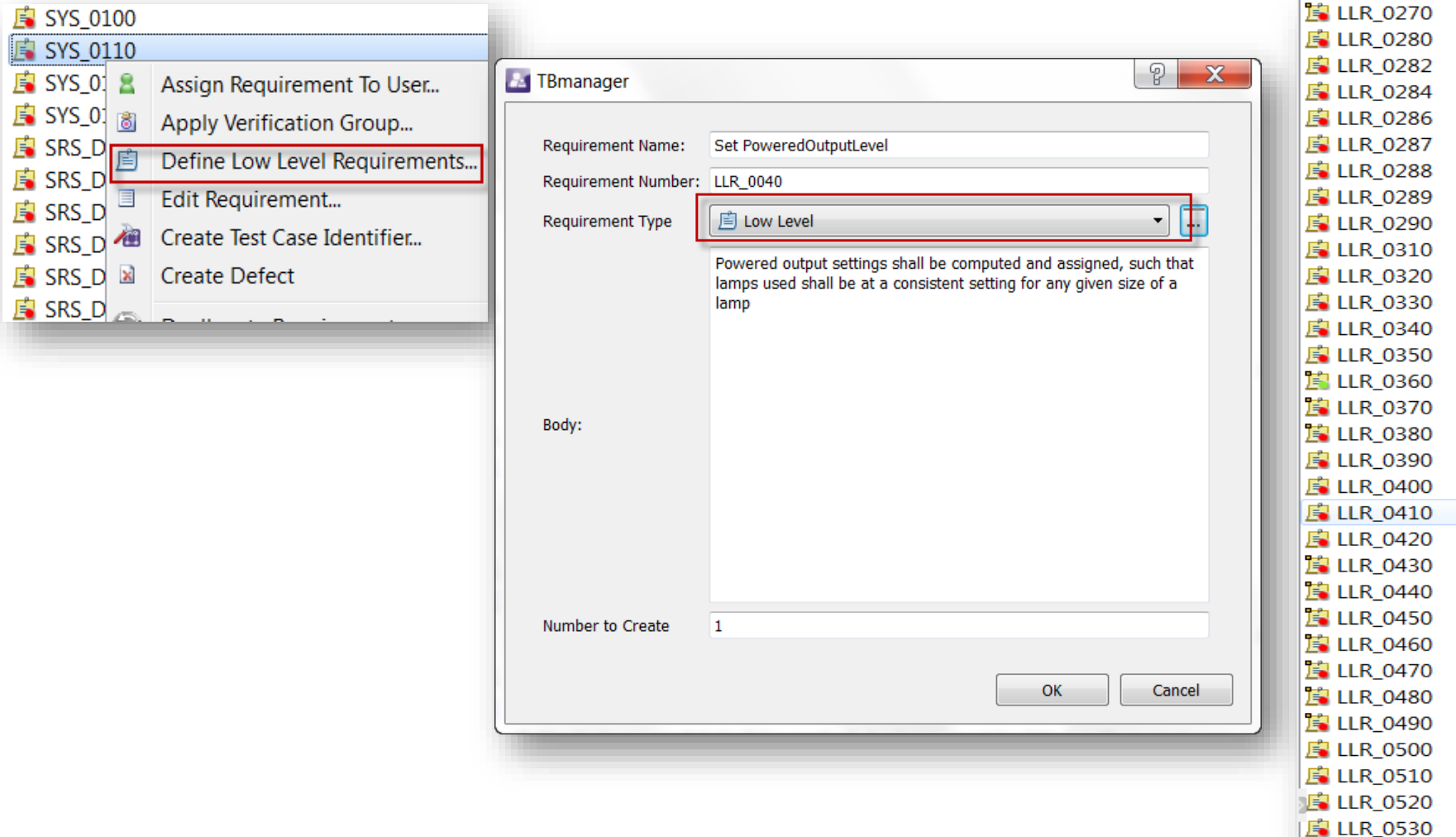


Importing requirements from Polarian



A 2.4 - Low-Level requirements are developed LDRA

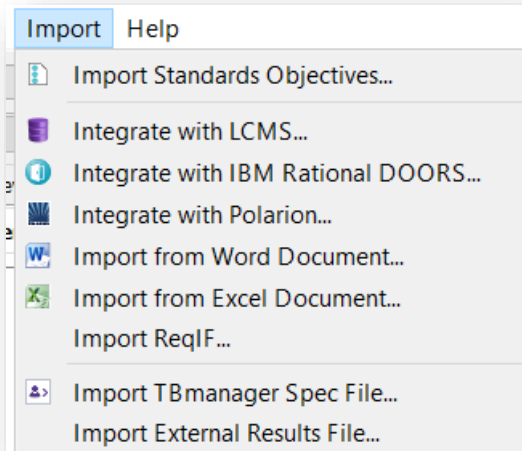
- In TBmanager, Low Level Requirement can be created for corresponding High level Requirement



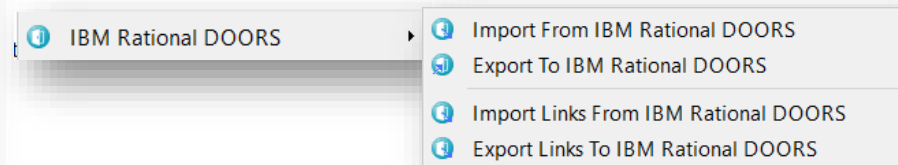
A 2.4 - Low-Level requirements are developed

- TBmanager can import Low-Level requirements from Requirement management tools like DOORS, Polarian, Word, Excel, ReqIF

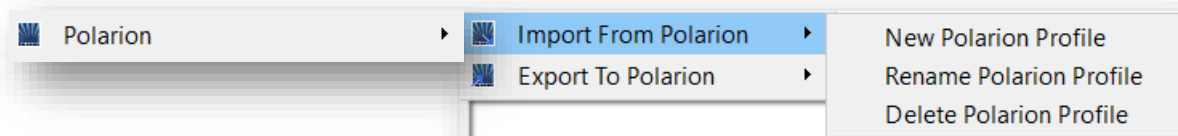
LLR_0270
LLR_0280
LLR_0282
LLR_0284
LLR_0286
LLR_0287
LLR_0288
LLR_0289
LLR_0290
LLR_0310
LLR_0320
LLR_0330
LLR_0340
LLR_0350
LLR_0360
LLR_0370
LLR_0380
LLR_0390
LLR_0400
LLR_0410
LLR_0420
LLR_0430
LLR_0440
LLR_0450
LLR_0460
LLR_0470
LLR_0480
LLR_0490
LLR_0500
LLR_0510
LLR_0520
LLR_0530



Importing requirements from DOORS



Importing requirements from Polarian



A 3.6 - High-level requirements are traceable to *LDRA* system requirements

SYS_0010, Display , (2 Notes) - Cody, Bill
SYS_0020, Initialisation and configuration , (1 Note) - Cody, Bill
SYS_0030, Output Calculation , (1 Note) - Cody, Bill
SYS_0040, Photometer, (1 Note) - Cody, Bill
SYS_0050, Cleanliness factor - Cody, Bill
SYS_0060, Lighting control unit , (1 Note) - Cody, Bill
SYS_0070, Luminaries - Cody, Bill
SYS_0080, Sirens and Signs, (1 Note) - Cody, Bill
SYS_0090, Failed Power Supply - Cody, Bill
SYS_0100, Lighting Adjustment - Cody, Bill
SYS_0110, Lamp output units - Cody, Bill
SYS_0120, Lamp sizes , (1 Note) - Cody, Bill
SYS_0130, Required luminance at ground level - Cody, Bill

HLR_0010, Starting display software, (1 Note) - Cody, Bill
HLR_0020, Input option photometer nominal range - Cody, Bill
HLR_0030, Input options photometer input out of bounds , (1 Note) - Cody, Bill
HLR_0040, Input options exit - Cody, Bill
HLR_0050, Input options days since cleaning nominal - Cody, Bill
HLR_0070, Input options power failure , (2 Notes) - Cody, Bill
HLR_0090, Display total cell demand - Cody, Bill
HLR_0100, Display Lumens , (1 Note) - Cody, Bill
HLR_0110, Cleanliness Factor - Cody, Bill
HLR_0115, Cleanliness efficiency factor , (1 Note) - Cody, Bill
HLR_0120, Tunnel Lighting Output Demand Calculation - Cody, Bill
HLR_0125, Adjust Powered Lighting - Cody, Bill
HLR_0130, Zone Lighting Formulae , (1 Note) - Cody, Bill
HLR_0140, Zone Lighting Output Demand Calculation , (1 Note) - Cody, Bill
HLR_0150, Set Lighting Output Demand Calculation - Cody, Bill
HLR_0160, Lamp Selection , (1 Note) - Cody, Bill
HLR_0170, Lamp Lighting Output Demand Calculation - Cody, Bill
HLR_0180, Tunnel Lighting Output Handling , (1 Note) - Cody, Bill
HLR_0190, Lamp Output Handling , (1 Note) - Cody, Bill
HLR_0200, Tunnel Lighting Configuration - Cody, Bill
HLR_0210, Exit sign battery drain - Cody, Bill
HLR_0215, Luminary configuration - Cody, Bill
HLR_0220, Failed Power Tunnel Lighting Output Calculation - Cody, Bill
HLR_0230, Failed Lamp Output Handling , (1 Note) - Cody, Bill
HLR_0231, Lamps maximum output - Cody, Bill
HLR_0232, Lamp output - Cody, Bill
HLR_0233, Minimum lamp output threshold - Cody, Bill
HLR_0234, Lamp extinguishing condition - Cody, Bill
HLR_0235, Lamp on condition - Cody, Bill
HLR_0236, Large lamp use - Cody, Bill
HLR_0237, Photometer signal conversion - Cody, Bill
HLR_0340, System Data Initialisation , (1 Note) - Cody, Bill
HLR_0350, Mountings Configuration - Cody, Bill
HLR_0360, Cell Configuration and output - Cody, Bill

HLR ->System Req

System Req ->HLR

SYS_0010, Display , (2 Notes) - Cody, Bill
SYS_0020, Initialisation and configuration , (1 Note) - Cody, Bill
SYS_0030, Output Calculation , (1 Note) - Cody, Bill
SYS_0040, Photometer, (1 Note) - Cody, Bill
SYS_0050, Cleanliness factor - Cody, Bill
SYS_0060, Lighting control unit , (1 Note) - Cody, Bill
SYS_0070, Luminaries - Cody, Bill
SYS_0080, Sirens and Signs, (1 Note) - Cody, Bill
SYS_0090, Failed Power Supply - Cody, Bill
SYS_0100, Lighting Adjustment - Cody, Bill
SYS_0110, Lamp output units - Cody, Bill
SYS_0120, Lamp sizes , (1 Note) - Cody, Bill
SYS_0130, Required luminance at ground level - Cody, Bill

HLR_0010, Starting display software, (1 Note) - Cody, Bill
HLR_0020, Input option photometer nominal range - Cody, Bill
HLR_0030, Input options photometer input out of bounds , (1 Note) - Cody, Bill
HLR_0040, Input options exit - Cody, Bill
HLR_0050, Input options days since cleaning nominal - Cody, Bill
HLR_0070, Input options power failure , (2 Notes) - Cody, Bill
HLR_0090, Display total cell demand - Cody, Bill
HLR_0100, Display Lumens , (1 Note) - Cody, Bill
HLR_0110, Cleanliness Factor - Cody, Bill
HLR_0115, Cleanliness efficiency factor , (1 Note) - Cody, Bill
HLR_0120, Tunnel Lighting Output Demand Calculation - Cody, Bill
HLR_0125, Adjust Powered Lighting - Cody, Bill
HLR_0130, Zone Lighting Formulae , (1 Note) - Cody, Bill
HLR_0140, Zone Lighting Output Demand Calculation , (1 Note) - Cody, Bill
HLR_0150, Set Lighting Output Demand Calculation - Cody, Bill
HLR_0160, Lamp Selection , (1 Note) - Cody, Bill
HLR_0170, Lamp Lighting Output Demand Calculation - Cody, Bill
HLR_0180, Tunnel Lighting Output Handling , (1 Note) - Cody, Bill
HLR_0190, Lamp Output Handling , (1 Note) - Cody, Bill
HLR_0200, Tunnel Lighting Configuration - Cody, Bill
HLR_0210, Exit sign battery drain - Cody, Bill
HLR_0215, Luminary configuration - Cody, Bill
HLR_0220, Failed Power Tunnel Lighting Output Calculation - Cody, Bill
HLR_0230, Failed Lamp Output Handling , (1 Note) - Cody, Bill
HLR_0231, Lamps maximum output - Cody, Bill
HLR_0232, Lamp output - Cody, Bill
HLR_0233, Minimum lamp output threshold - Cody, Bill
HLR_0234, Lamp extinguishing condition - Cody, Bill
HLR_0235, Lamp on condition - Cody, Bill
HLR_0236, Large lamp use - Cody, Bill
HLR_0237, Photometer signal conversion - Cody, Bill
HLR_0340, System Data Initialisation , (1 Note) - Cody, Bill
HLR_0350, Mountings Configuration - Cody, Bill
HLR_0360, Cell Configuration and output - Cody, Bill

Table A- 3 2 - High-level requirements are accurate and consistent

- TBmanager allows to review High Level requirements and add corresponding Notes/Defects for each of Requirement.

The screenshot displays the TBmanager interface. On the left, a tree view shows a hierarchy of requirements: (0) Item to Mappings, (1) Two-Level Requirements to Mappings, (2) Requirements 1, and (12) Requirements 2. Under (2) Requirements 1, a list of requirements is shown, including HLR_0010, HLR_0020, HLR_0030, HLR_0040, HLR_0050 (2 Notes), HLR_0070, HLR_0100, HLR_0200, HLR_0215, HLR_0340, HLR_0350, and HLR_0360. A context menu is open over HLR_0030, with 'Add Note...' highlighted. Below the tree, the 'Requirement Body' for HLR_0030 is visible, showing the text: 'The Tunnel Lighting system shall provide a human machine interface for emulation of input and examination of output data'. On the right, a 'Notes' table is displayed, showing a list of notes for HLR_0050. A purple arrow points from the 'Add Note...' menu item to the 'Notes' table. Another purple arrow points from the 'Requirement Body' text to a text box at the bottom left.

Notes	Note Body	Links To	Created By	Created On	Status	Finished	Bo
HLR_0050	The maximum...	(1) HLR_0050	Wilson, Jane	2015-04-28 ...	N/A	N/A	The
HLR_0050	Number of ...	(1) HLR_0050	Wilson, Jane	2015-04-28 ...	N/A	N/A	Nur

A review can be done on each requirement as body of requirement is visible in TBmanager

Corresponding NOTES can be added if requirements are not correct/consistent

Table A- 4 2 - Low-level requirements are accurate and consistent

- TBmanager allows to review Low Level requirements and add corresponding Notes/Defects for each of Requirement.











































The screenshot displays the TBmanager interface. On the left, a tree view shows a hierarchy of requirements: (0) Item to Mappings, (1) Two-Level Requirements to Mappings, (2) Requirements 1, and (12) Requirements 2. Under (2) Requirements 1, a list of requirements is shown, including HLR_0010, HLR_0020, HLR_0030, HLR_0040, HLR_0050 (with 2 Notes), HLR_0070, HLR_0100, HLR_0200, HLR_0215, HLR_0340, HLR_0350, and HLR_0360. A context menu is open over HLR_0030, with the 'Add Note...' option highlighted. Below the tree, the 'Requirement Body' for HLR_0030 is visible, showing the text: 'The Tunnel Lighting system shall provide a human machine interface for emulation of input and examination of output data'. On the right, a table titled 'Notes' lists notes for HLR_0050. The table has columns: Note Body, Links To, Created By, Created On, Status, Finished, and Bo. The first note for HLR_0050 has a body of 'The maximum...' and links to (1) HLR_0050, created by Wilson, Jane on 2015-04-28. The second note has a body of 'Number of ...' and also links to (1) HLR_0050, created by Wilson, Jane on 2015-04-28. Below the table, the 'Note Body' for the second note is visible, showing the text: 'The maximum Number of Days is not mentioned. Need to be mentioned.' Two purple arrows point from the 'Add Note...' option in the context menu to the 'Notes' table, and another purple arrow points from the 'Requirement Body' of HLR_0030 to the first note in the table.

Note Body	Links To	Created By	Created On	Status	Finished	Bo
The maximum...	(1) HLR_0050	Wilson, Jane	2015-04-28 ...	N/A	N/A	The
Number of ...	(1) HLR_0050	Wilson, Jane	2015-04-28 ...	N/A	N/A	Nur

A review can be done on each requirement as body of requirement is visible in TBmanager























































Corresponding NOTES can be added if requirements are not correct/consistent

A 4.6 - Low-level requirements are traceable to *LDRA* high-level requirements

 HLR_0100	 LLR_0090, Get Lamp Model LightSolo
 HLR_0110	 LLR_0100, Get Data and Read Content
 HLR_0115	 LLR_0110, Get Data and Read Content
 HLR_0120	 LLR_0120, Initialise Lamp
 HLR_0125	 LLR_0130, Set Lumens Output
 HLR_0130	 LLR_0140, Get MaximumLumens
 HLR_0140	 LLR_0150, Get Minimum Lumens
 HLR_0150	 LLR_0160, Send Power to Lamp
 HLR_0160	 LLR_0170, Initialise Lamp Attributes
 HLR_0170	 LLR_0180, Lamp Dimensions
 HLR_0180	 LLR_0190, Lamp Dimensions
 HLR_0190	 LLR_0200, Lamp Dimensions
 HLR_0200	 LLR_0210, Lamp Drain
 HLR_0215	 LLR_0220, Lamp Type Instantiation
 HLR_0220	 LLR_0230, Lamp Type Initialisation
 HLR_0230	 LLR_0240, Lamp Type Get Maximum Lumens
 HLR_0231	 LLR_0250, Lamp Type Get Minimum Lumens
 HLR_0232	 LLR_0260, Lamp Type Get Power Required
 HLR_0233	 LLR_0270, Initialise lighting system - Cody, Bill
 HLR_0234	 LLR_0280, Photometer input interface
 HLR_0235	 LLR_0282, Input options photometer input o...

HLR -> LLR

LLR ->HLR

 HLR_0010 - Cody, Bill	 LLR_0180, Lamp Dimensions
 HLR_0020	 LLR_0190, Lamp Dimensions
 HLR_0030	 LLR_0200, Lamp Dimensions
 HLR_0040	 LLR_0210, Lamp Drain
 HLR_0050, (2 Notes)	 LLR_0220, Lamp Type Instantiation
 HLR_0070	 LLR_0230, Lamp Type Initialisation
 HLR_0100	 LLR_0240, Lamp Type Get Maximum Lumens
 HLR_0110	 LLR_0250, Lamp Type Get Minimum Lumens
 HLR_0115	 LLR_0260, Lamp Type Get Power Required
 HLR_0120	 LLR_0270, Initialise lighting system - Cody, Bill
 HLR_0125	 LLR_0280, Photometer input interface
 HLR_0130	 LLR_0282, Input options photometer input o...
 HLR_0140	 LLR_0284, Input options exit
 HLR_0150	 LLR_0286, Input options days since cleaning ...
 HLR_0160	 LLR_0287, Input options days since cleaning ...
 HLR_0170	 LLR_0288, Input options power failure
 HLR_0180	 LLR_0289, Input options power failure
 HLR_0190	 LLR_0290, Days since cleaning input interface
 HLR_0200	 LLR_0310, Mounting Area Instantiation
 HLR_0215	 LLR_0320, Mounting Area Number of Lamps
 HLR_0220	 LLR_0330, System Data Instantiation
 HLR_0230	 LLR_0340, System Data Initialisation
 HLR_0231	 LLR_0350, Calculate and get soiling factor
 HLR_0232	 LLR_0400, System Data Set Days Since Clean...
 HLR_0233	 LLR_0410, System Data Query Set Days Bet...
 HLR_0234	 LLR_0440, Initialise Tunnel - Cody, Bill
 HLR_0235	 LLR_0450, Adjust Tunnel Lighting
HLR_0236	LLR_0460, Adjust Powered Lighting

Objectives – Source Code

- ✗ A-4.1 - Software patching integrity confirmed -
- ✗ A-5.1 - Source code complies with low-level requirements - Unfulfilled
- ✗ A-5.2 - Source code complies with software architecture - Unfulfilled
- ✗ A-5.3 - Source code is verifiable - Unfulfilled
- ✗ A-5.4 - Source code conforms to standards - Unfulfilled
- ✗ A-5.5 - Source code is traceable to low-level requirements. - Unfulfilled
- ✗ A-5.6 - Source code is accurate and consistent - Unfulfilled

Table A-5 4 - Source Code conforms to standards

- TBvision performs code conformance with industry programming standards.

File View

Results View

Code Review : (By Violation)

Violations	Phase	Code	Level of Violation	Number Violations	Standard Code
free called on variable with no allocated space.	125	D	Mandatory	2	MISRA-C:2012 R.22.2
Attempt to use uninitialised pointer.	53	D	Mandatory	17	MISRA-C:2012 R.9.1
Procedure contains UR data flow anomalies.	69	D	Mandatory	180	MISRA-C:2012 R.9.1
Function call with no prior declaration.	496	S	Mandatory	28	MISRA-C:2012 R.17.3
Overlapping data items in memcopy.	647	S	Mandatory		MISRA-C:2012 R.19.1
#undef used.	68	S	Advisory	4	MISRA-C:2012 R.20.5
Names only differ by case.	217	S	Advisory	3	MISRA-C:2012 D.4.5
Use of single line comment(s) //.	110	S	Advisory	4	MISRA-C:2012 R.1.2
More than one break or goto statement in loop.	409	S	Advisory	30	MISRA-C:2012 R.15.4

Attempt to use uninitialised pointer. : lookBuf

Violations	Phase	Code	Level of Violation	Number Violations	Standard Code
Macro not used in translation unit.	628	S	Advisory	92	MISRA-C:2012 R.2.5
Use of function like macro.	340	S	Advisory	72	MISRA-C:2012 D.4.9
Procedure has more than one exit point.	7	C	Advisory	170	MISRA-C:2012 R.15.5
Pointer param should be declared pointer to const.	120	D	Advisory	244	MISRA-C:2012 R.8.13
Basic type declaration used.	90	S	Advisory	630	MISRA-C:2012 D.4.6
Attempt to change parameter passed by value.	14	D	Advisory	87	MISRA-C:2012 R.17.8
Structure implementation not hidden. : sdPtr	104	D	Advisory		MISRA-C:2012 D.4.8
#include preceded by non preproc directives.	338	S	Advisory		MISRA-C:2012 R.20.1
User type declared but not used in code analysed. : AES_SET_KEY_FUNC	413	S	Advisory		MISRA-C:2012 R.2.3.R.2.4
Empty middle expression in for loop.	429	S	Required	2	MISRA-C:2012 R.14.2
Function return value potentially unused.	91	D	Required	5	MISRA-C:2012 D.4.7.R.17.7
Operation not appropriate to boolean type.	249	S	Required	3	MISRA-C:2012 R.10.1
Cast involving function pointer.	606	S	Required	22	MISRA-C:2012 R.11.1
Negative (or potentially negative) shift.	403	S	Required	3	MISRA-C:2012 R.10.1.R.12.2
Local structure returned in function result.	77	D	Required	4	MISRA-C:2012 R.18.6
Construct leads to infeasible code.	139	S	Required	8	MISRA-C:2012 R.14.3
Denominator not checked before use.	127	D	Required	2	MISRA-C:2012 D.4.1
More than one of # or ## in a macro.	76	S	Required	2	MISRA-C:2012 R.20.11
Shifting value too far.	51	S	Required	4	MISRA-C:2012 R.12.2
Name reused in inner scope.	131	S	Required	7	MISRA-C:2012 R.5.3

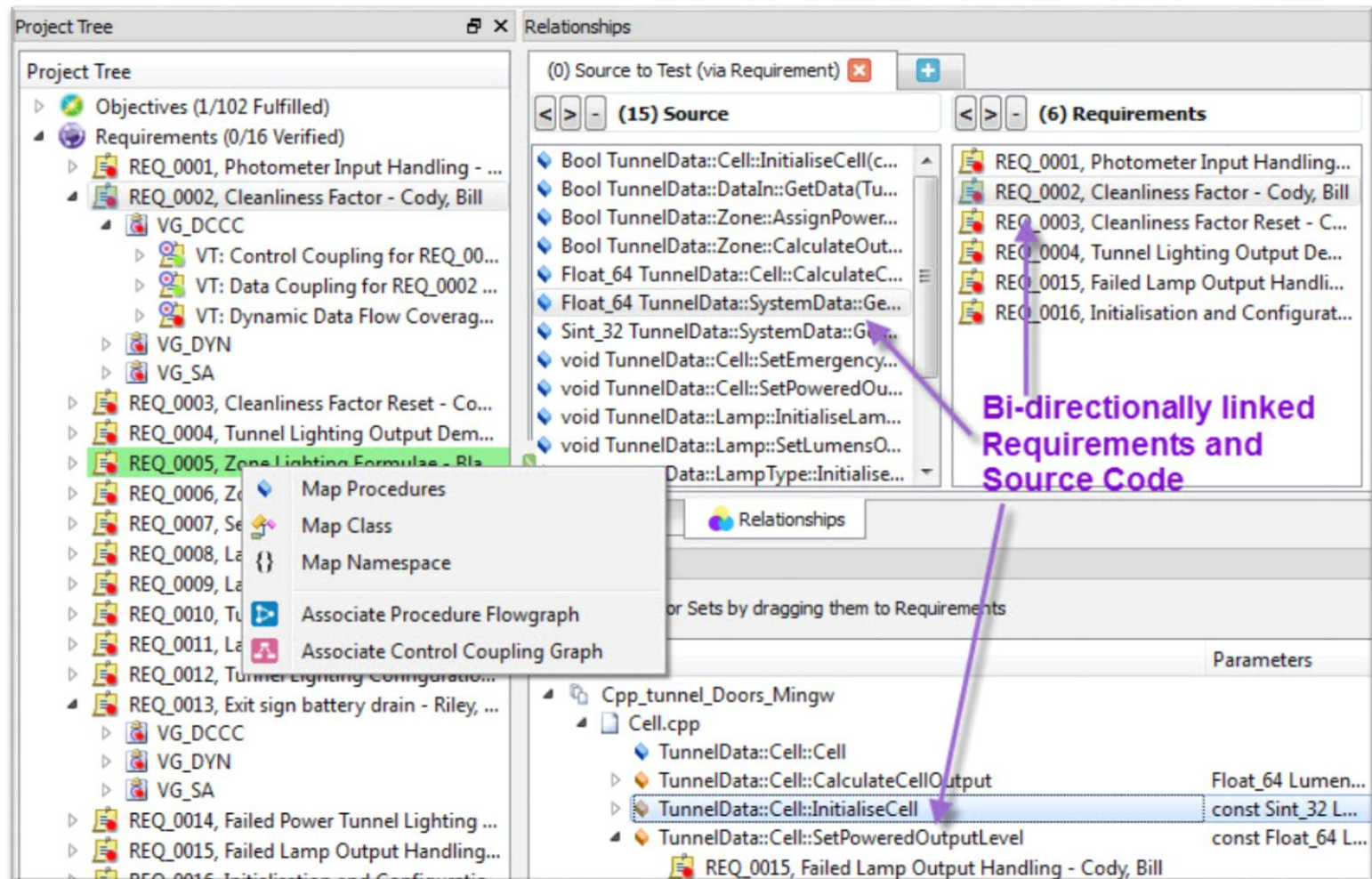
CAST
 CERT-C
 CERT-C:2014
 CMSE
 CONFORM
 CWE
 Customer Sample
 DERA
 EADS
 FSB582-C
 GJB
 GJB_8114
 HIS
 JPL
 Legacy
 MISRA-AC
 MISRA-C:1998
 MISRA-C:2004
 MISRA-C:2012
 MISRA-C:2012/ADD2
 MISRA-C:2012/AMD1
 MISRA-C:2012/AMD1/ADD2
 NETRINO
 RUNTIME
 SEC-Cv1
 SEC-Cv2
 secureC
 Standard
 TBrn Requires
 UML
 VSOS
 No Standards Model

Table A-5 5 - Source Code is traceable to low-level requirements

- HLR->LLR->Source Code(Function) Traceability

(34) Requirements 1	(58) Requirements 2	(41) Mappings
HLR_0010, Starting display software, (1 Note) - Cody, Bill	LLR_0060, Get Lamp Model Duo - Cody, Bill	Bool TunnelData::Cell::InitialiseCell(const Sint_32 LuminaireSetSize, con...
HLR_0020, Input option photometer nominal range - Cody, Bill	LLR_0070, Get Lamp Model Guide - Cody, Bill	Bool TunnelData::DataIn::GetData(TunnelData::Tunnel * pTunnel);
HLR_0030, Input options photometer input out of bounds...	LLR_0080, Get Lamp Model Announcer - Cody, Bill	Float_64 TunnelData::Cell::CalculateCellOutput(Float_64 LumensDeman...
HLR_0040, Input options exit - Cody, Bill	LLR_0090, Get Lamp Model LightSolo - Cody, Bill	Float_64 TunnelData::Lamp::GetMaximumLumens();
HLR_0050, Input options days since cleaning nominal ...	LLR_0100, Get Data and Read Content - Cody, Bill	Float_64 TunnelData::Lamp::GetMinimumLumens();
HLR_0070, Input options power failure, (2 Notes) - Cody, Bill	LLR_0110, Get Data and Read Content, (1 Note) - Cody, Bill	Float_64 TunnelData::LampType::GetMaximumLumens();
HLR_0090, Display total cell demand - Cody, Bill	LLR_0120, Initialise Lamp - Cody, Bill	Float_64 TunnelData::LampType::GetMinimumLumens();
HLR_0100, Display Lumens, (1 Note) - Cody, Bill	LLR_0130, Set Lumens Output - Cody, Bill	Float_64 TunnelData::SystemData::GetEmergencyLampLumens();
HLR_0110, Cleanliness Factor - Cody, Bill	LLR_0140, Get MaximumLumens - Cody, Bill	Float_64 TunnelData::SystemData::GetLampMaximumLumens(const La...
HLR_0115, Cleanliness efficiency factor, (1 Note) - Cody, Bill	LLR_0150, Get Minimum Lumens - Cody, Bill	Float_64 TunnelData::SystemData::GetLampMinimumLumens(const La...
HLR_0120, Tunnel Lighting Output Demand Calculation...	LLR_0160, Send Power to Lamp - Cody, Bill	Float_64 TunnelData::SystemData::GetSoilingFactor();
HLR_0125, Adjust Powered Lighting - Cody, Bill	LLR_0170, Initialise Lamp Attributes - Cody, Bill	Sint_32 TunnelData::LampType::GetPowerRequired(const Float_64 Lum...
HLR_0130, Zone Lighting Formulae, (1 Note) - Cody, Bill	LLR_0180, Lamp Dimensions, (3 Notes) - Cody, Bill	Sint_32 TunnelData::SystemData::GetDaysBetweenCleaning();
HLR_0140, Zone Lighting Output Demand Calculation, ...	LLR_0190, Lamp Dimensions, (3 Notes) - Lee, Peter	Sint_32 TunnelData::SystemData::GetExitSignSpacing();
HLR_0150, Set Lighting Output Demand Calculation - ...	LLR_0200, Lamp Dimensions, (3 Notes) - Lee, Peter	Sint_32 TunnelData::SystemData::GetLampPowerRequired(const LampT...
HLR_0160, Lamp Selection, (1 Note) - Cody, Bill	LLR_0210, Lamp Drain - Lee, Peter	Sint_32 TunnelData::SystemData::GetSirenSpacing();
HLR_0170, Lamp Lighting Output Demand Calculation ...	LLR_0220, Lamp Type Instantiation - Lee, Peter	Sint_32 main();
HLR_0180, Tunnel Lighting Output Handling, (1 Note) - Cody, Bill	LLR_0230, Lamp Type Initialisation - Lee, Peter	TunnelData::Cell::Cell();
HLR_0190, Lamp Output Handling, (1 Note) - Cody, Bill	LLR_0240, Lamp Type Get Maximum Lumens - Lee, Peter	TunnelData::Lamp::Lamp();
HLR_0200, Tunnel Lighting Configuration - Cody, Bill	LLR_0250, Lamp Type Get Minimum Lumens - Lee, Peter	TunnelData::LampAttributes::LampAttributes(int h, int w, TunnelData::...
HLR_0210, Exit sign battery drain - Cody, Bill	LLR_0260, Lamp Type Get Power Required, (1 Note) - Lee, Peter	TunnelData::LampType::LampType();
HLR_0215, Luminary configuration - Cody, Bill	LLR_0270, Initialise lighting system, (1 Note) - Cody, Bill	TunnelData::MountingArea::MountingArea(int l, int b);
HLR_0220, Failed Power Tunnel Lighting Output Calculation...	LLR_0280, Photometer input interface, (1 Note) - Cody, Bill	TunnelData::SystemData * TunnelData::SystemData::Instance();
HLR_0230, Failed Lamp Output Handling, (1 Note) - Cody, Bill	LLR_0282, Input options photometer input out of bounds...	TunnelData::SystemData::SystemData();
HLR_0231, Lamps maximum output - Cody, Bill	LLR_0284, Input options exit, (2 Notes) - Cody, Bill	TunnelData::SystemData::SystemData(const TunnelData::SystemData &...
HLR_0232, Lamp output - Cody, Bill	LLR_0286, Input options days since cleaning nominal...	TunnelData::model TunnelData::Cell::GetLampModel(const Sint_32 This...
HLR_0233, Minimum lamp output threshold - Cody, Bill	LLR_0287, Input options days since cleaning out of bounds...	int TunnelData::LampAttributes::Area();
HLR_0234, Lamp extinguishing condition - Cody, Bill	LLR_0288, Input options power failure, (2 Notes) - Cody, Bill	int TunnelData::LampAttributes::Drain();
HLR_0235, Lamp on condition - Cody, Bill	LLR_0289, Input options power failure, (2 Notes) - Cody, Bill	int TunnelData::LampAttributes::Height();
HLR_0236, Large lamp use - Cody, Bill	LLR_0290, Days since cleaning input interface, (2 Notes) - Cody, Bill	int TunnelData::LampAttributes::Width();
HLR_0237, Photometer signal conversion - Cody, Bill	LLR_0310, Mounting Area Instantiation - Cody, Bill	int TunnelData::MountingArea::NumLamps(TunnelData::LampAttribute...
HLR_0340, System Data Initialisation, (1 Note) - Cody, Bill	LLR_0320, Mounting Area Number of Lamps - Cody, Bill	void TunnelData::Cell::SetEmergencyOutputLevel();
HLR_0350, Mountings Configuration - Cody, Bill	LLR_0330, System Data Instantiation - Cody, Bill	void TunnelData::Cell::SetPoweredOutputLevel(const Float_64 Lumens...
HLR_0360, Cell Configuration and output - Cody, Bill	LLR_0340, System Data Initialisation, (2 Notes) - Cody, Bill	void TunnelData::DataIn::ReadContent(Sint_32 * array, char * Token, Sint...
	LLR_0350, Calculate and get soiling factor - Cody, Bill	void TunnelData::Lamp::InitialiseLamp(const LampTypeID ThisLampTy...
	LLR_0360, System Data Query Get Lamp Power Required...	void TunnelData::Lamp::SendPowerToLamp(const Sint_32 PowerSetting);
	LLR_0370, System Data Query Get Lamp Maximum Lumens...	void TunnelData::Lamp::SetLumensOutput(Float_64 LumensRequired);
	LLR_0380, System Data Query Get Lamp Minimum Lumens...	void TunnelData::LampType::InitialiseLampType(const Float_64 Highest...
	LLR_0390, System Data Query Get Lamp Emergency Lumens...	void TunnelData::SystemData::=(const TunnelData::SystemData & du...
	LLR_0400, System Data Set Days Since Cleaning - C...	void TunnelData::SystemData::InitialiseParams(Sint_32 * pSystemDataAr...
	LLR_0410, System Data Query Set Days Between Cleaning...	void TunnelData::SystemData::SetDaysSinceCleaning(const Sint_32 Days);
	LLR_0420, System Data Query Get Exit Sign Spacing...	

Building and Maintaining Links



VISUAL LINKING AND LIVE BI-DIRECTIONAL TRACEABILITY

Objectives – Test

- ✖ A-7.1 - Test procedures are correct - Unfulfilled
- ✖ A-7.2 - Test results are correct and discrepancies explained - Unfulfilled
- ✖ A-7.3 - Test coverage of high-level requirements is achieved - Unfulfilled
- ✖ A-7.4 - Test coverage of low-level requirements is achieved - Unfulfilled
- ✖ A-7.5 - Test coverage of software structure (modified condition/decision) is achieved - Unfulfilled
- ✖ A-7.6 - Test coverage of software structure (decision coverage) is achieved - Unfulfilled
- ✖ A-7.7 - Test coverage of software structure is achieved - Unfulfilled
- ✖ A-7.8 - Test coverage of software structure (data coupling and control coupling) is achieved - Unfulfilled
- ✖ A-7.9 - Configuration items are identified - Unfulfilled

Table A-7 5 - Test coverage of software structure (MC/DC) is achieved

- Generates a Test Case Planner

```
if ( ( (GetVoltage() > 5) &&
      (Liquidlevel(water_level) > 10) ) ||
      (Enable_Gates() == 1) )
```

INDEX	C1	C2	C3	EXPECTED OUTCOME	MC/DC INDEPENDENT PAIRS	WAVE
1	f(T, T, T)	=	T		
2	f(T, T, F)	=	T	C1.C2	#*
3	f(T, F, T)	=	TC3	#
4	f(T, F, F)	=	FC2.C3	#*
5	f(F, T, T)	=	TC3	#
6	f(F, T, F)	=	F	C1.....C3	#*
7	f(F, F, T)	=	TC3	
8	f(F, F, F)	=	FC3	

```
f( T, T, T )
f( T, T, F )
f( T, F, T )
f( T, F, F )
f( F, T, T )
f( F, T, F )
f( F, F, T )
f( F, F, F )
```

INDEX	C1	C2	C3	EXPECTED OUTCOME	MC/DC INDEPENDENT PAIRS	WAVE
1	f(T, T, T)	=	T		
2	f(T, T, F)	=	T	C1.C2	#*
3	f(T, F, T)	=	TC3	#
4	f(T, F, F)	=	FC2.C3	#*
5	f(F, T, T)	=	TC3	#
6	f(F, T, F)	=	F	C1.....C3	#*
7	f(F, F, T)	=	TC3	
8	f(F, F, F)	=	FC3	

INDEX	C1	C2	C3	EXPECTED OUTCOME	MC/DC INDEPENDENT PAIRS	WAVE
1	f(T, T, T)	=	T		
2	f(T, T, F)	=	T	C1.C2	#*
3	f(T, F, T)	=	TC3	#
4	f(T, F, F)	=	FC2.C3	#*
5	f(F, T, T)	=	TC3	#
6	f(F, T, F)	=	F	C1.....C3	#*
7	f(F, F, T)	=	TC3	
8	f(F, F, F)	=	FC3	

Table A-7 5 - Test coverage of software structure (MC/DC) is achieved

proc1
Coverage Metrics required to achieve DO-178C Level A Not Attained

Statement (TER1) = 89 % Branch/Decision (TER2) = 88 % MC/DC = 67 %

INDEX	C1 C2 C3	EXPECTED OUTCOME	EXECUTED BY RUNS...	
			PREVIOUS	CURRENT
			COMBINED	MC/DC INDEPENDENT PAIRS WAVE
1	f(T, T, T) = T	NO	YES	YES
2	f(T, T, F) = T	NO	YES	YES
3	f(T, F, T) = T	NO	NO	NO
4	f(T, F, F) = F	NO	YES	YES
5	f(F, T, T) = T	NO	NO	NO
6	f(F, T, F) = F	NO	YES	YES
7	f(F, F, T) = T	NO	NO	NO
8	f(F, F, F) = F	NO	YES	YES

BRANCH CONDITION NUMBER	COMBINATION PREVIOUS	EFFECTIVELY CURRENT	EXECUTED COMBINED
C1			
Truth Table Index: 2	f(T, T, F) = T	NO	YES
Truth Table Index: 6	f(F, T, F) = F	NO	YES
Independently affects result			
C2			
Truth Table Index: 2	f(T, T, F) = T	NO	YES
Truth Table Index: 4	f(T, F, F) = F	NO	YES
Independently affects result			
C3	NOT SHOWN	NOT SHOWN	NOT SHOWN

Detailed reporting on which branches/decisions are executed

Table A-7 6 - Test coverage of software structure (decision coverage) is achieved

```
/*
 * Add product to the list of scanned products
 * Unless there are too many products in which case
 * Just ignore the product
 */
static void addProduct(const struct Product * aProduct)
{
    LDRA_char_t message[MAX_STRING];

    if (scannedProducts < MAX_PRODUCTS_IN_BASKET)
    {
        if (aProduct != NULL_POINTER)
        {
            ShoppingBasket[scannedProducts] = aProduct;
            scannedProducts++;
            sprintf(message, "Adding %s", aProduct->name);
            Display_show(&message[0]);
        }
    }
    else
    {
        Display_show("Basket is full");
    }
}
```


Table A-7 6 - Test coverage of software structure (decision coverage) is achieved



PASS
PASS

addProduct
addProduct

Double-click to access

&state

45

*** Suspended ***

out

Managed Stubs
aProduct
scannedProducts
scannedProducts
stdout

Double-click to access

NULL

45

*** Suspended ***

out

Managed Stubs
aProduct
scannedProducts
scannedProducts
stdout

addProduct
Coverage Metrics required to achieve DO-178C Level A Attained

Statement (TER1) = 100 % Branch/Decision (TER2) = 100 % MC/DC : Not Applicable

LINE NUMBERS: FROM	REFORMATTED (SOURCE) TO	PREVIOUS RUNS	CURRENT RUN	COMBINED	CODE PRECEDING DECISION POINT
403 (63)	404 (64)	3	2	5	scannedProducts < 50U)
403 (63)	418 (74)	1	0 ***	1	
409 (65)	410 (66)	2	1	3	(void *) 0))
409 (65)	416 (72)	1	1	2	
417 (73)	421 (77)	3	2	5	else

Table A-7 7 - Test coverage of software structure (statement coverage) is achieved

```

/*
 * Get the price which depends on which special offer, if any, is used
 */
LDRA_uint32_t SpecialOffer_getPrice(const LDRA_uint32_t aQuantity,
    const LDRA_uint32_t aUnitPrice, const tSpecialOffer anOffer)
{
    LDRA_uint32_t price;
    switch (anOffer)
    {
        case BUY_ONE_GET_ONE_FREE:
            price = aUnitPrice * ((aQuantity + 1U) >> 1U);
            price = Special_Customer_Offer(price);
            price = Get_Day_Discount(price, THURSDAY);
            Display(price);
            break;

        case TEN_PERCENT_OFF:
            price = (aUnitPrice * aQuantity * 9U) / 10U;
            Display(price);
            break;

        case THREE_FOR_ONE_EURO:
            price = ((aQuantity / 3U) * 100U) + ((aQuantity % 3U) * aUnitPrice);
            Display(price);
            break;

        /* no offer */
        default:
            price = aUnitPrice * aQuantity;
            Display(price);
            break;
    }
    return price;
}

```

Table A-7 7 - Test coverage of software structure (statement coverage) is achieved

Specialoffer_getPrice
Coverage Metrics required to achieve DO-178C Level A Attained

Statement (TER1) = 100 % Branch/Decision (TER2) = 100 % MC/DC : Not Applicable

LINE NUMBER REF. (SOURCE)	STATEMENT	PREVIOUS RUNS	CURRENT RUN	COMBINED
163 (21)	LDRA_uint32_t	-	-	-
164	SpecialOffer_getPrice (15	4	19
165	const LDRA_uint32_t aQuantity ,	-	-	-
166 (22)	const LDRA_uint32_t aUnitPrice ,	-	-	-
167	const tSpecialOffer anOffer)	-	-	-
168 (23)	{	-	-	-
169 (24)	LDRA_uint32_t	-	-	-
170	price ;	-	-	-
171 (25)	switch (15	4	19
172	anOffer	15	4	19
173)	15	4	19
174 (26)	{	15	4	19
175 (27)	case BUY_ONE_GET_ONE_FREE :	6	1	7
176 (28)	price = aUnitPrice * (6	1	7
177	(aQuantity + 1U) >> 1U) ;	6	1	7
178 (29)	price = Special_Customer_Offer (price) ;	6	1	7
179 (30)	price = Get_Day_Discount (price , THURSDAY) ;	6	1	7
180 (31)	Display (price) ;	6	1	7
181 (32)	break ;	6	1	7
182 (34)	case TEN_PERCENT_OFF :	4	1	5
183 (35)	price = (aUnitPrice * aQuantity + 9U) /	4	1	5
184	10U ;	4	1	5
185 (36)	Display (price) ;	4	1	5
186 (37)	break ;	4	1	5
187 (39)	case THREE_FOR_ONE_EURO :	2	1	3
188 (40)	price = (2	1	3
189	(aQuantity /	2	1	3
190	3U) * 100U) + (2	1	3
191	(aQuantity % 3U) * aUnitPrice) ;	2	1	3
192 (41)	Display (price) ;	2	1	3
193 (42)	break ;	2	1	3
194 (44)	/* no offer */	-	-	-
195 (45)	default :	3	1	4
196 (46)	price = aUnitPrice * aQuantity ;	3	1	4
197 (47)	Display (price) ;	3	1	4
198 (48)	break ;	3	1	4
199 (50)	}	-	-	-
200 (51)	return	15	4	19
201	price ;	15	4	19
202 (52)	}	-	-	-

Ranged Test Case Variable (Enumeration)

Variable Details

Name:

Type:

Use:

Value

☐ Apply single value to test case variable.

☒ Apply ranged value to test case variable.

Minimum:

Maximum:

Step:

☐ Apply intermediary values to test case variables.

Value	Label

< Back Finish Cancel Help

Table A-7 8- Test coverage of software structure (data coupling and control coupling) is achieved

• Data Coupling Coverage

state		Cashregister.c	Cashregister_barcode	G	R	233			
			Cashregister_cancel	G	R	244			
				G	D	249 *****			
			Cashregister_code	G	R	263 *****			
			Cashregister_end	G	R	275			
				G	D	279			
			Cashregister_key	G	R	288 *****			
			Cashregister_start	G	R	299			
				G	D	303			
				L	R	152	164		
sum_total		Cashregister.c	generateTicket	L	E	117			
				L	D	117	152		
theBarcode		Cashregister.c	Cashregister_code	G	R	265 *****			
				G	D	266 *****			
			1 aBarcode	P	E	174			
theBarcode		Cashregister.c	identifyProduct	P	R	177 *****			
			Cashregister_key	G	R	290 *****			
				G	D	290 *****			
theCP		Productdatabase.c	Cashregister_start	G	D	301			
			Productdatabase_getSpecificCountedProduct	L	E	96			
				L	R	106			
theChar		Main.c		L	D	99			
						103 *****			
			main	L	E	25			

Use Coverage Summary

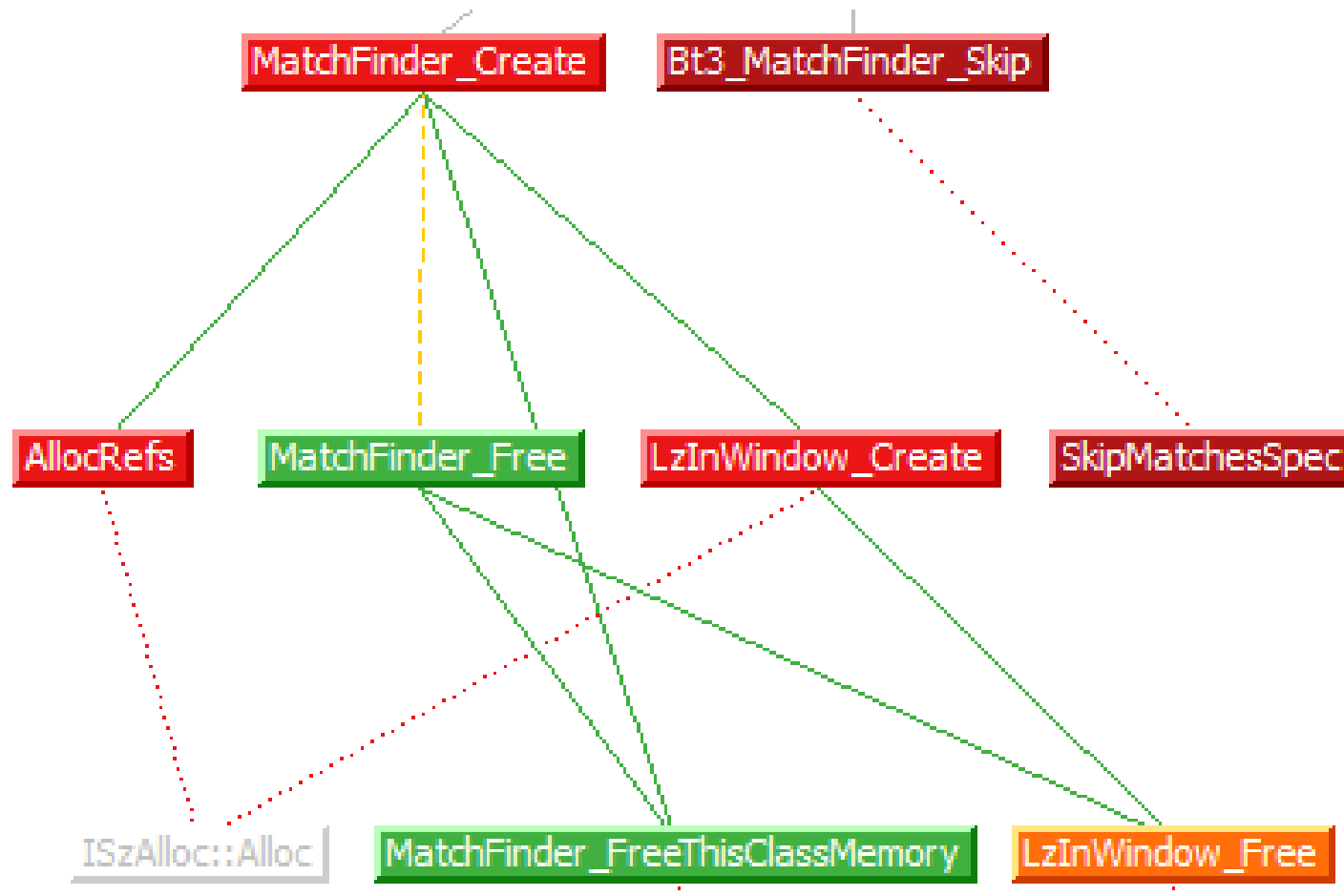
Summary		Uses
Total Uses		177
Uses Covered		166
Uses Not Covered		11
Overall Use Coverage (%)		94

Variable Coverage Summary

Summary		Variables Matching	Variable Uses
Total Variables	73		177
Number Covered	65		160
Number Partially Covered	3		9
Number Not Covered	5		8
Variables Fully Covered (%)		89	90

Table A-7 8- Test coverage of software structure (data coupling and control coupling) is achieved

- Control Coupling Coverage



- Dramatic reduction of time necessary for DDCC analysis
- Clear, repeatable, methodology for DDCC that has been reviewed and accepted by DERs
 - Reduces risks of methodology ambiguities during SOI audits
 - Consistent with the expectations DO-178C as a test measurement exercise
- Defined artifact set for archival and review
- Dramatically reduced cost of DDCC activities during incremental releases

Table A-7 9 - Verification of additional code that can not *LDRA* be traced to Source Code, is achieved

- For level A, it is necessary to show 100% coverage of not just the high-level language, but also verify any additional, non-traceable Executable Object Code (EOC)
- Generally we can't read Object Code, but we can read the generated assembler code and since there must be a one to one relationship between the Object Code and Assembler, we could either:
 - Manually inspect and verify the generated assembler code or;
 - Automatically run the existing test cases and obtain the assembler level code coverage

Table A-7 9 - Verification of additional code that can not *LDRA* be traced to Source Code, is achieved

- Object Code Verification hinges on 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 (therefore raising questions about traceability)
- Even if no options are set, then the control flow of the generated code will differ from the source code
- If options are set for the compiler to optimise the code or to add array bounds checking, then the control flow of the generated code will possibly greatly differ from that of the source code

Table A-7 9 - Verification of additional code that can not *LDRA* be traced to Source Code, is achieve

Assembly Code Structure

As we can see the structure of the generated assembler code is quite different to that of the C code

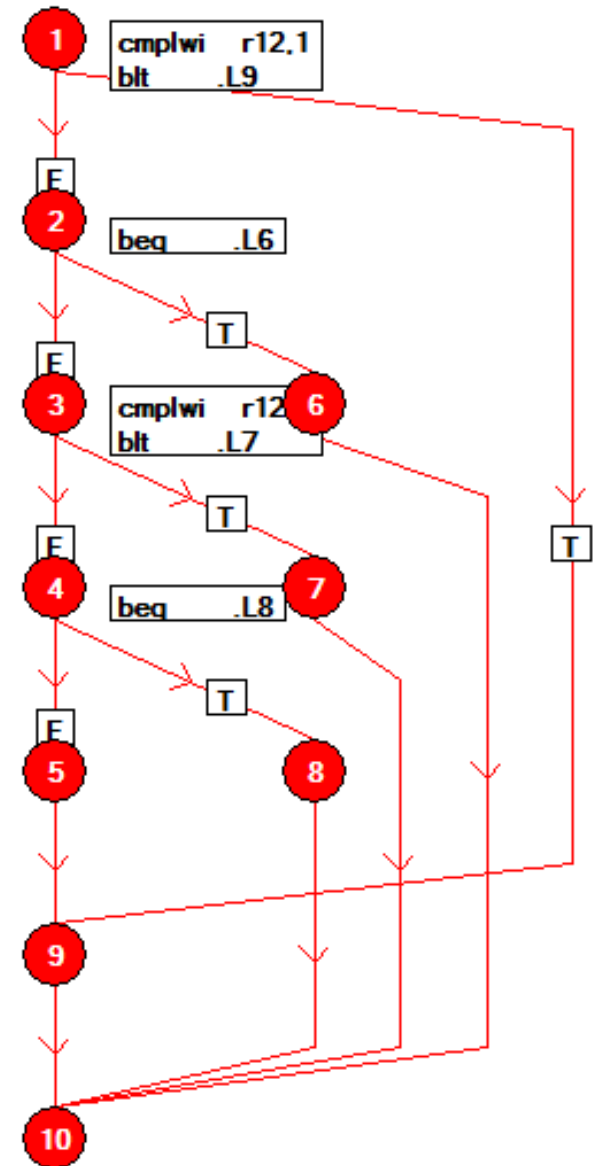
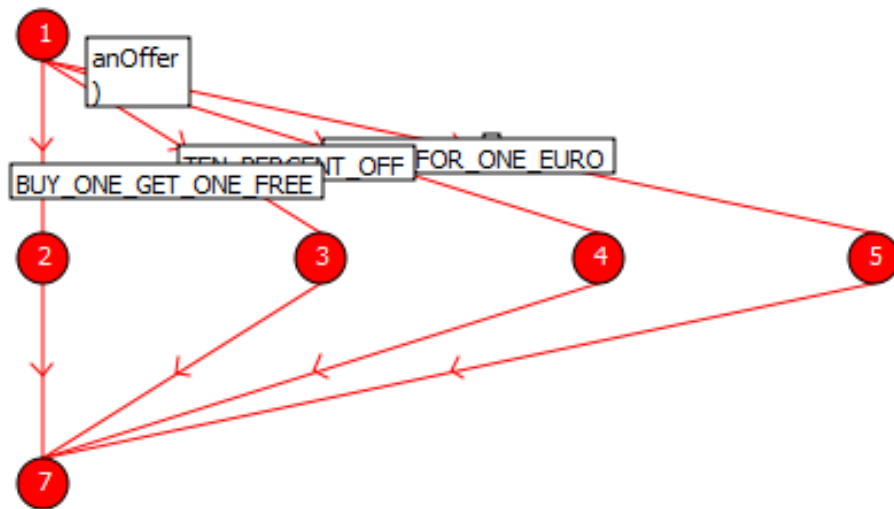


Table A-7 9 - Verification of additional code that can not *LDRA* be traced to Source Code, is achieved

- At the same time, when we inspect the code coverage on the assembler code, we observe that we have not achieved 100% coverage

LDRA Testbed ® Dynamic Coverage Analysis Report

**File : C:\LDRA_Workarea\GHS_MULTI4_C_CashRegister_tbwrkfls\GHS_MULTI4_
C_CashRegister_asmwrkfls\sp ecialoffer.s**

Overall Coverage Result (For File): Fail

Statement (TER1) = 98 % (Fail) Branch/Decision (TER2) = 83 % (Fail)

Table A-7 9 - Verification of additional code that can not *LDRA* be traced to Source Code, is achieved

- There is one **decision** that is not covered

LINE NUMBER REF. (SOURCE)	STATEMENT	PREVIOUS RUNS	CURRENT RUN	COMBINED
34 (31)	SpecialOffer_getPrice:	4	4	8
35 (32)	mr r8,r3	4	4	8
36 (33)	mr r9,r4	4	4	8
37 (34)	mr r12,r5	4	4	8
38 (36)	# .bf	-	-	-
39		-	-	-
40	.LDW01:	4	4	8
41 (38)	#17: const LDRA_uint32_t aUnitPrice, const tSpecialOffer anOffer)	-	-	-
42 (39)	#18: {	-	-	-
43 (40)	#19: LDRA_uint32_t price;	-	-	-
44 (41)	#20: switch (anOffer)	-	-	-
45	cmplwi r12,1	4	4	8
46 (42)	blt .L9	4	4	8
47 (43)	beq .L6	3	3	6
48 (44)	cmplwi r12,3	2	2	4
49 (45)	blt .L7	2	2	4
50 (46)	beq .L8	1	1	2
51 (47)	b .L9	0 ***	0 ***	0 ***
52 (48)		-	-	-
53	.L6:	1	1	2
54 (50)	#21: {	-	-	-
55 (51)	#22: case BUY_ONE_GET_ONE_FREE:	-	-	-
56 (52)	#23: price = aUnitPrice * ((aQuantity + 1U) >> 1U);	-	-	-
57 (53)	#line23	-	-	-
58 (54)	#.lin.C.3A.5CLDRA_Workarea.SCGHS_MULTII4_C_CashRegister_tbwrkfls.SCGHS	-	-	-
59		-	-	-
60	.LDWlin1:	1	1	2
61 (55)	addi r12,r8,1	1	1	2
62 (56)	srwi r12,r12,1	1	1	2
63 (57)	mullw r12,r9,r12	1	1	2
64 (58)	b .L4	1	1	2
65 (59)		-	-	-
66	.L7:	1	1	2
67 (61)	#24: break;	-	-	-
68 (62)	#25: case TEN_PERCENT_OFF:	-	-	-
69 (63)	#26: /*LDRA_INSPECTED 96 S */	-	-	-
70 (64)	#27: price = (aUnitPrice * aQuantity * 9U) / 10U;	-	-	-
71	mullw r12,r9,r8	1	1	2
72 (65)	mr r11,r12	1	1	2
73 (66)	#28: /*LDRA_INSPECTED 96 S */	-	-	-

Table A-7 3 - Test coverage of high-level requirements is achieved

HLR_0010, Starting display software, (1 Note)	TCL_0010: Given the configuration file data set, the output of nominal photometer input will reflect that configura...
HLR_0020, Input option photometer nominal range	TCL_0020: Generated lamp output data will indicate that the tunnel lighting software has accepted the photomet...
HLR_0030, Input options photometer input out of bounds , (1 Note)	TCL_0030: For HMI selection, photometer input, days since cleaning inputs, if out of bounds inputs are entered, thi...
HLR_0040, Input options exit	TCL_0050: After setting the number of days since cleaning the tunnel software shall return to the HMI initial selecti...
HLR_0050, Input options days since cleaning nominal	TCL_0060: After setting the power failure state, the tunnel software shall display lamp output data for powerfailur...
HLR_0070, Input options power failure , (2 Notes)	TCL_0070: The tunnel software output stream will provide the total cell demand and lumens per meter...
HLR_0090, Display total cell demand	TCL_0080: The tunnel software output stream will provide the total cell demand ...
HLR_0100, Display Lumens , (1 Note)	TCL_0090: Name: Soiling factor calculation for dirty...
HLR_0110, Cleanliness Factor	TCL_0100: The Tunnel software output produced for the full range of cleanliness factors observable from the resul...
HLR_0115, Cleanliness efficiency factor , (1 Note)	TCL_0110: The Tunnel software output produced from photometer inputs will show calculations for each zone...
HLR_0120, Tunnel Lighting Output Demand Calculation	TCL_0120: The Tunnel software output produced from photometer inputs will show calculations for all zones...
HLR_0125, Adjust Powered Lighting	TCL_0130: The Tunnel software output produced from photometer inputs shall show straight line calculations each...
HLR_0130, Zone Lighting Formulae , (1 Note)	TCL_0140: The Tunnel software output produced from photometer inputs will show calculations for each zone...
HLR_0140, Zone Lighting Output Demand Calculation , (1 Note)	TCL_0150: The Tunnel software output produced from photometer inputs will show calculations for each luminaire...
HLR_0150, Set Lighting Output Demand Calculation	TCL_0160: The Tunnel software output produced from photometer inputs will that days since cleaning and resulting ...
HLR_0160, Lamp Selection , (1 Note)	TCL_0170: The Tunnel software output produced from photometer inputs will show calculations for each lamp...
HLR_0170, Lamp Lighting Output Demand Calculation	TCL_0180: The Tunnel software output produced from photometer inputs will show calculations for each lamp...
HLR_0180, Tunnel Lighting Output Demand Calculation	TCL_0180: The Tunnel software output produced from photometer inputs will show calculations for each lamp...

High Level Requirements Traceability Table

Show/Hide

Number/Name	Text	Covered	Number/Name	Text
High Level Requirements			High Level Tests	
HLR_0090, Display total cell demand	In the nominal power state after entering a nominal range photometer input or nominal days since cleaning the software shall display Total Cell demand and lumens per metre	Yes	TCL_0070: The tunnel software output stream will provide the total cell demand and lumens per meter.	The tunnel software output stream will provide the total cell demand and lumens per meter.
HLR_0231, Lamps maximum output	A lamp shall provide an output of 120lm/W when used at the maximum output	Yes	TCL_0250: The Tunnel software output produced to drive maximum lm/W levels will generate output levels no greater than 120 lm/w	The Tunnel software output produced to drive maximum lm/W levels will generate output levels no greater than 120 lm/w
HLR_0125, Adjust Powered Lighting	Given the photometer input lighting shall be calculated across all zones	Yes	TCL_0120: The Tunnel software output produced from photometer inputs will show calculations for all zones	The Tunnel software output produced from photometer inputs will show calculations for all zones
HLR_0340, System Data Initialisation , (1 Note)	All software data shall be stored for management tracking and reporting	Yes	TCL_0320: The Tunnel software output produced from photometer inputs, given a set of input configuration files, verified against expected output files will verify that data management capabilities of the software are being met	The Tunnel software output produced from photometer inputs, given a set of input configuration files, verified against expected output files will verify that data management capabilities of the software are being met
HLR_0110, Cleanliness Factor	A percentage cleanliness factor shall be calculated depending on time elapsed since cleaning. 0% shall represent totally obscenity of the luminaires through the build up of grime, 100% shall represent complete cleanliness. The rate of grime accumulation has been calculated by Waveworks Research Labs their figures indicate a reduction to 50% over a period of 182 days.	Yes	TCL_0090: Name: Soiling factor calculation for dirty Description: Verify that soiling factor is calculated correctly per formula described in LLR_0355 in cases where the lamp is not clear Inputs: 1, 19 mAs Expected Output: Total Cell Demand: 173840 at 8692 per metre Lumens: 48000 Power Setting: 400 for lamp ID: 40900 Lumens: 48000 Power Setting: 400 for lamp ID: 40901 Lumens: 19710 Power Setting: 173 for lamp ID: 40902 Lumens: 19710 Power Setting: 173 for lamp ID: 40903 Lumens: 48000 Power Setting: 400 for lamp ID: 40904 Total Cell Demand: 183420 at 9171 per metre The Tunnel software output produced for varying levels of days since cleaning will shows a reduction of 50 percent reduction over a period of 182 days	Name: Soiling factor calculation for dirty Description: Verify that soiling factor is calculated correctly per formula described in LLR_0355 in cases where the lamp is not clear Inputs: 1, 19 mAs Expected Output: Total Cell Demand: 173840 at 8692 per metre Lumens: 48000 Power Setting: 400 for lamp ID: 40900 Lumens: 48000 Power Setting: 400 for lamp ID: 40901 Lumens: 19710 Power Setting: 173 for lamp ID: 40902 Lumens: 19710 Power Setting: 173 for lamp ID: 40903 Lumens: 48000 Power Setting: 400 for lamp ID: 40904 Total Cell Demand: 183420 at 9171 per metre The Tunnel software output produced for varying levels of days since cleaning will shows a reduction of 50 percent reduction over a period of 182 days
HLR_0120, Tunnel Lighting Output Demand Calculation	Lighting demand parameters for the tunnel shall be calculated and allocated to each zone	Yes	TCL_0110: The Tunnel software output produced from photometer inputs will show calculations for each zone	The Tunnel software output produced from photometer inputs will show calculations for each zone

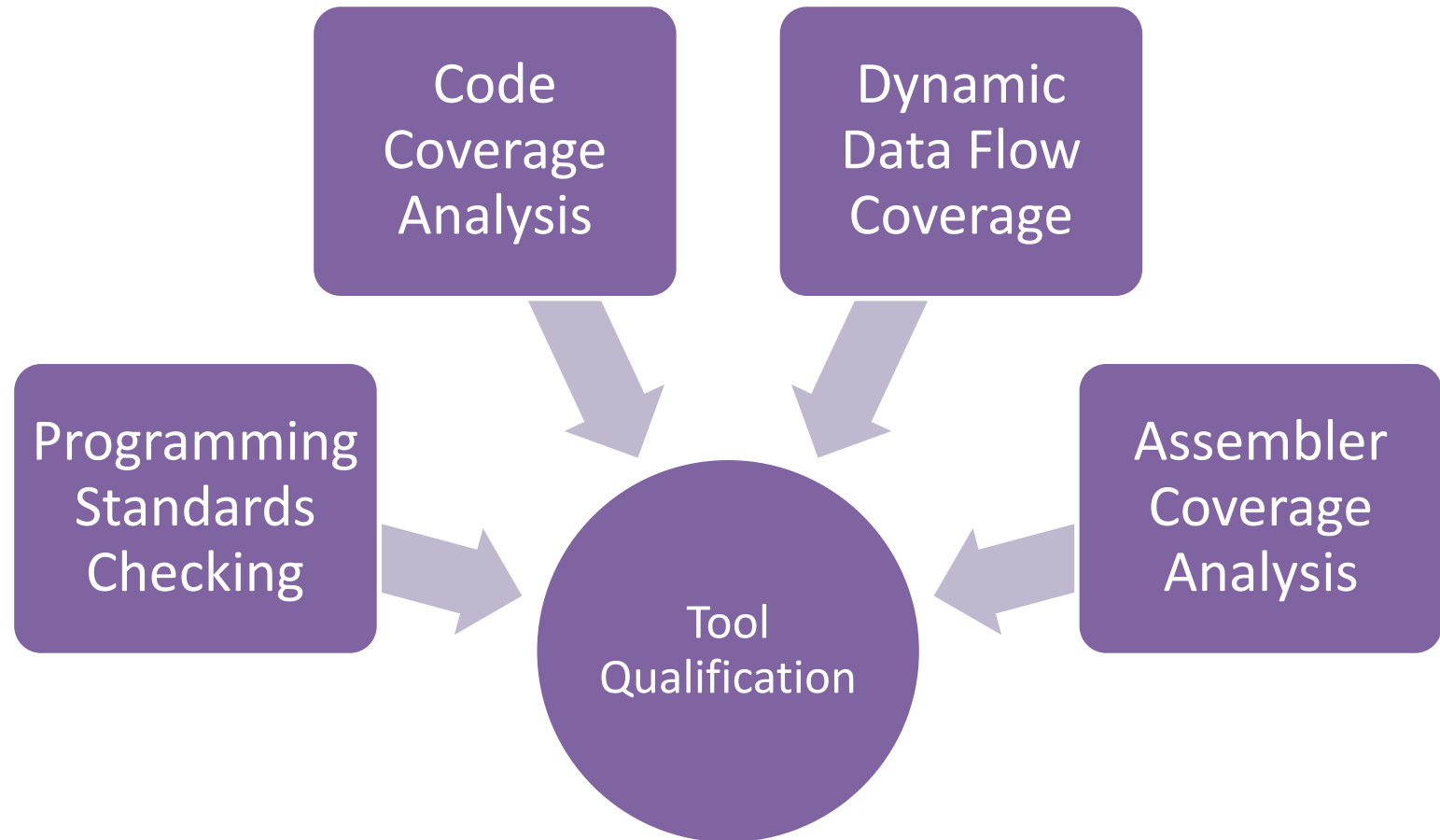
Table A-7 4 - Test coverage of low-level requirements is achieved

LLR_0030, Set Emergency output level	TCL_0340: The Tunnel software output produced from photometer inputs will show calculations for cell output. Ex...
LLR_0040, Set PoweredOutputLevel , (1 Note)	TCL_0345: Text case data needs to be updated
LLR_0050, Calculate cell output	TCL_1_8: Default
LLR_0060, Get Lamp Model Duo	TCL_5080: Name: Soiling factor calculation for dirty...
LLR_0070, Get Lamp Model Guide	TCL_5090: Name: Soiling factor calculation for dirty...
LLR_0080, Get Lamp Model Announcer	TCL_5100: Verify that soiling factor is calculated correctly per formula described in LLR_0355 in cases where the la...
LLR_0090, Get Lamp Model LightSolo	TCL_5110: Verify that the Cell::Cell is correctly instantiated
LLR_0100, Get Data and Read Content	TCL_5120: Verify that a given is is instantiated and initialized correctly
LLR_0110, Get Data and Read Content , (1 Note)	TCL_5125: Verify that Cell::SetEmergencyOutputLevel is set to its defined emergency demand level to minimize po...
LLR_0120, Initialise Lamp	TCL_5130: Verify that powered output settings is assigned consistently for any given size of a lamp
LLR_0130, Set Lumens Output	TCL_5140: Low Level Requirements based tests
LLR_0140, Get MaximumLumens	TCL_5180: Verify that for a given lamp and spacing for exit signs and sirens the correct lamp model is returned
LLR_0150, Get Minimum Lumens	TCL_5190: Verify that TunnelData::DataIn::ReadContent loads .ini file and stores the data in SystemdataArray
LLR_0160, Send Power to Lamp	TCL_5200: Verify that tokens are copied into ZoneData are the are parsed from the .ini file
LLR_0170, Initialise Lamp Attributes	TCL_5210: Verify that Lamp::Lamp is called for construction of the Lamp object. Additionally verify that Lamp::Initi...
LLR_0180, Lamp Dimensions , (3 Notes)	TCL_5220: Verify that Lamp::SetLumensOutput outputs the number of lumens per lamp
LLR_0190, Lamp Dimensions , (3 Notes)	TCL_5230: Verify that Lamp::GetMaximumLumens() returns the Maximum lumens allowable for a given lamp when...
LLR_0200, Lamp Dimensions , (3 Notes)	TCL_5240: Verify that Lamp::GetMinimumLumens() returns the minimum lumens allowable for a given lamp when ...
LLR_0210, Lamp Drain	TCL_5250: Verify that Lamp::SendPowerToLamp outputs the power setting for a given lampid
LLR_0220, Lamp Type Instantiation	TCL_5260: Verify the instantiation and initialisation of the lampattributes class with nominal range values for height, ...
LLR_0230, Lamp Type Initialisation	TCL_5270: Verify that the TunnelData::LampAttributes::Height() member function returns the height as expected fo...
LLR_0240, Lamp Type Get Maximum Lumens	TCL_5280: Verify that the TunnelData::LampAttributes::Width() member function returns the width as expected for ...

Low Level Requirements Traceability Table

Show/H

Number/Name	Text	Covered	Number/Name	Text
Low Level Requirements			Low Level Tests	
LLR_0100, Get Data and Read Content	Get Data shall parse the tunnel lighting system initialisation data using a provided initialisation file	Yes	TCL_5190: Verify that TunnelData::DataIn::ReadContent loads .ini file and stores the data in SystemdataArray	Verify that TunnelData::DataIn::ReadContent loads .ini file and stores the data in SystemdataArray
LLR_0490, Initialise zone , (3 Notes)	The zone class shall be configurable with the specified values	No	<Not Covered>	<Not Covered>
LLR_0010, Instantiate Cell	A Cell shall be instantiation with zero types of lamps, zero maximum lumens, zero minimum lumens, zero for the cell ID, and zero for the cell size	Yes	TCL_5110: Verify that the Cell::Cell is correctly instantiated	Verify that the Cell::Cell is correctly instantiated
LLR_0170, Initialise Lamp Attributes	A lampmodel shall be initialised by a provided height, width, and model	Yes	TCL_5260: Verify the instantiation and initialisation the lampattributes class with nominal range values for height, width, and tunnadata::model	Verify the instantiation and initialisation the lampattributes class with nominal range values for height, width, and tunnadata::model
LLR_0480, Zone default intensity , (2 Notes)	The zone class shall set the default intensity to the brightest level for all the lamps in a luminaire.	No	<Not Covered>	<Not Covered>
LLR_0070, Get Lamp Model Guide	A Guide lamp model shall be applied if a lamp must be fitted with an exit sign	Yes	TCL_5180: Verify that for a given lamp and spacing for exit signs and sirens the correct lamp model is returned	Verify that for a given lamp and spacing for exit signs and sirens the correct lamp model is returned
LLR_0030, Set Emergency output level	For emergency lighting, only the smallest lamp per luminaire shall be set to its defined emergency demand level to minimize power demands on emergency supplies	Yes	TCL_5125: Verify that Cell::SetEmergencyOutputLevel is set to its defined emergency demand level to minimize power demands	Verify that Cell::SetEmergencyOutputLevel is set to its defined emergency demand level to minimize power demands
LLR_0380, System Data Query Get Lamp Minimum Lumens	For each lamp type minimum lamp lumens shall be returned upon query	Yes	TCL_5480: Verify that TunnelData::SystemData::GetDaysBetweenCleaning returns the mDaysBetweenCleaning correctly	Verify that TunnelData::SystemData::GetDaysBetweenCleaning returns the mDaysBetweenCleaning correctly
LLR_0460, Adjust Powered Lighting , (2 Notes)	Given the photometer input in mAmps and powered conditions, the lighting output demands shall be adjusted across all zones	No	<Not Covered>	<Not Covered>
LLR_0110, Get Data and Read Content , (1 Note)	Get Data shall initialise system data using a provided initialisation file.	Yes	TCL_5200: Verify that tokens are copied into ZoneData are the are parsed from the .ini file	Verify that tokens are copied into ZoneData are the are parsed from the .ini file
LLR_0020, Initialise Cell , (1 Note)	A Cell shall be initialised by initialising both the cell parameters as well as the lamps within the cell	Yes	TCL_5120: Verify that a given is instantiated and initialised correctly	Verify that a given is instantiated and initialised correctly



Show Compliance

Comprehensive Process Control Documents

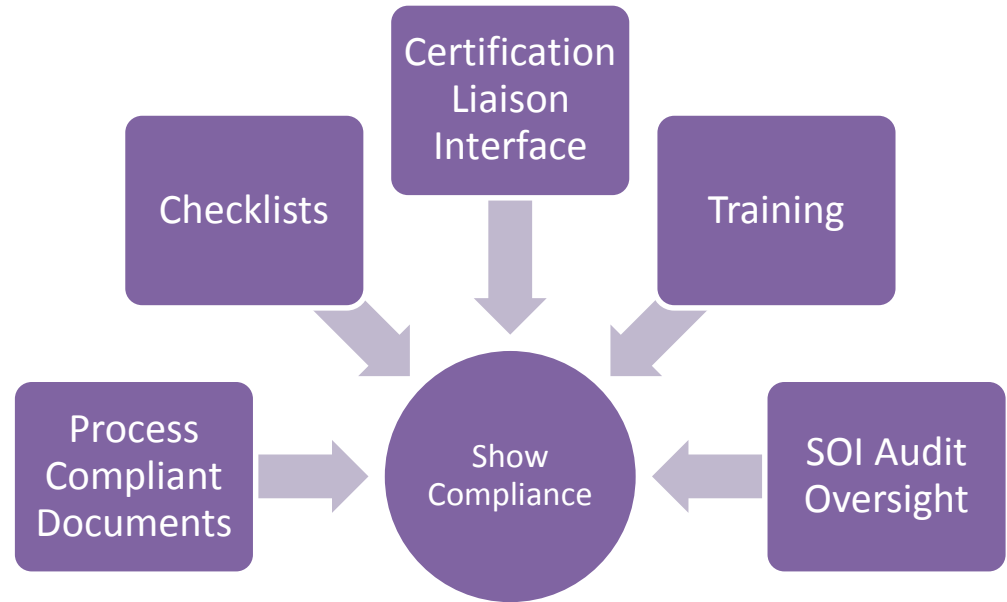
- 90% complete
- FAA & EASA
- DO-178 & DO-254

Project Transition Criteria

- Checklists

Detailed DO-178 & DO-254 Training

Level A DER Audit Support



For further information:

www.ldra.com

info@ldra.com



@ldra_technology



LDRA Software Technology



LDRA Limited



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