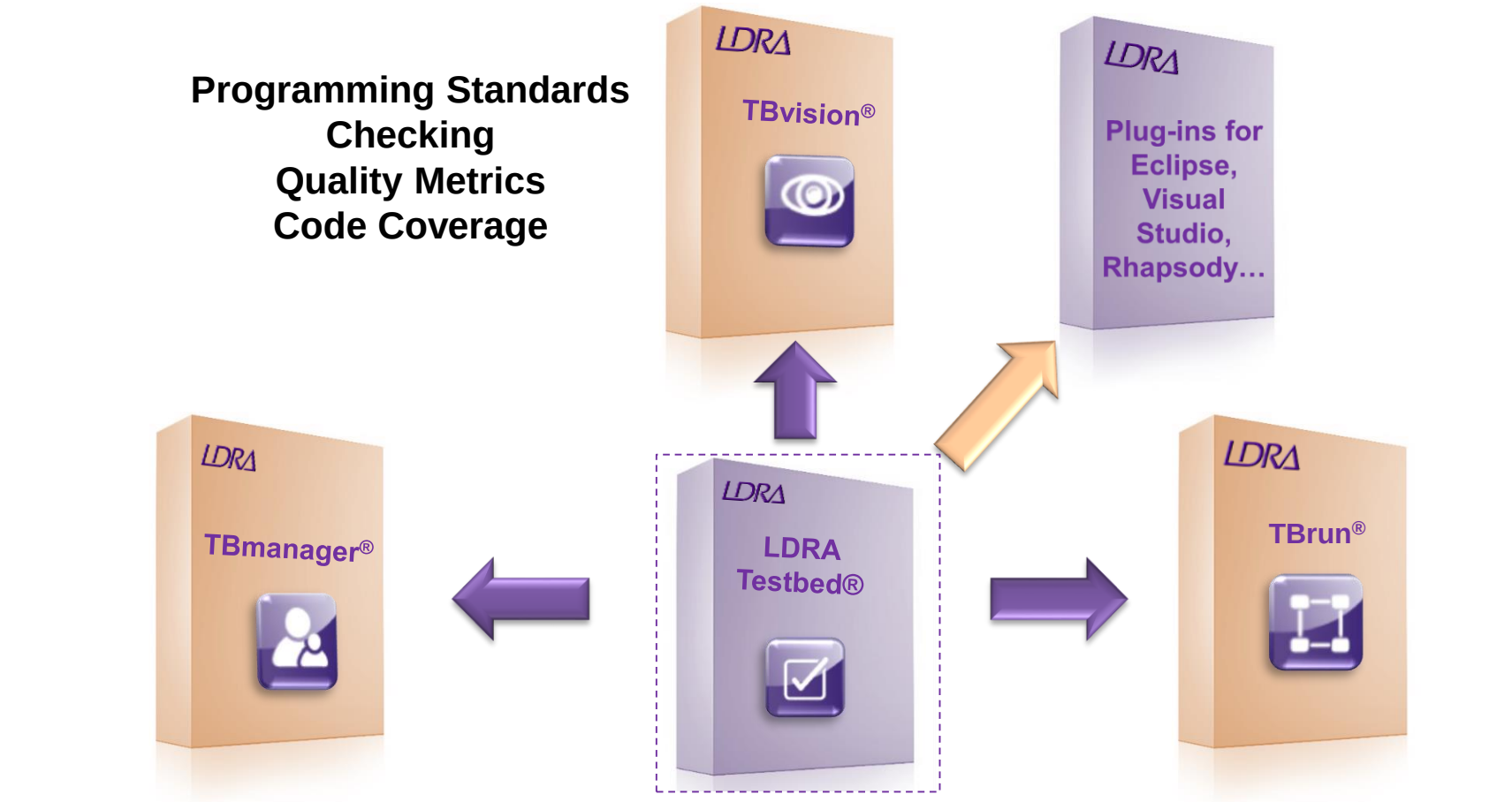
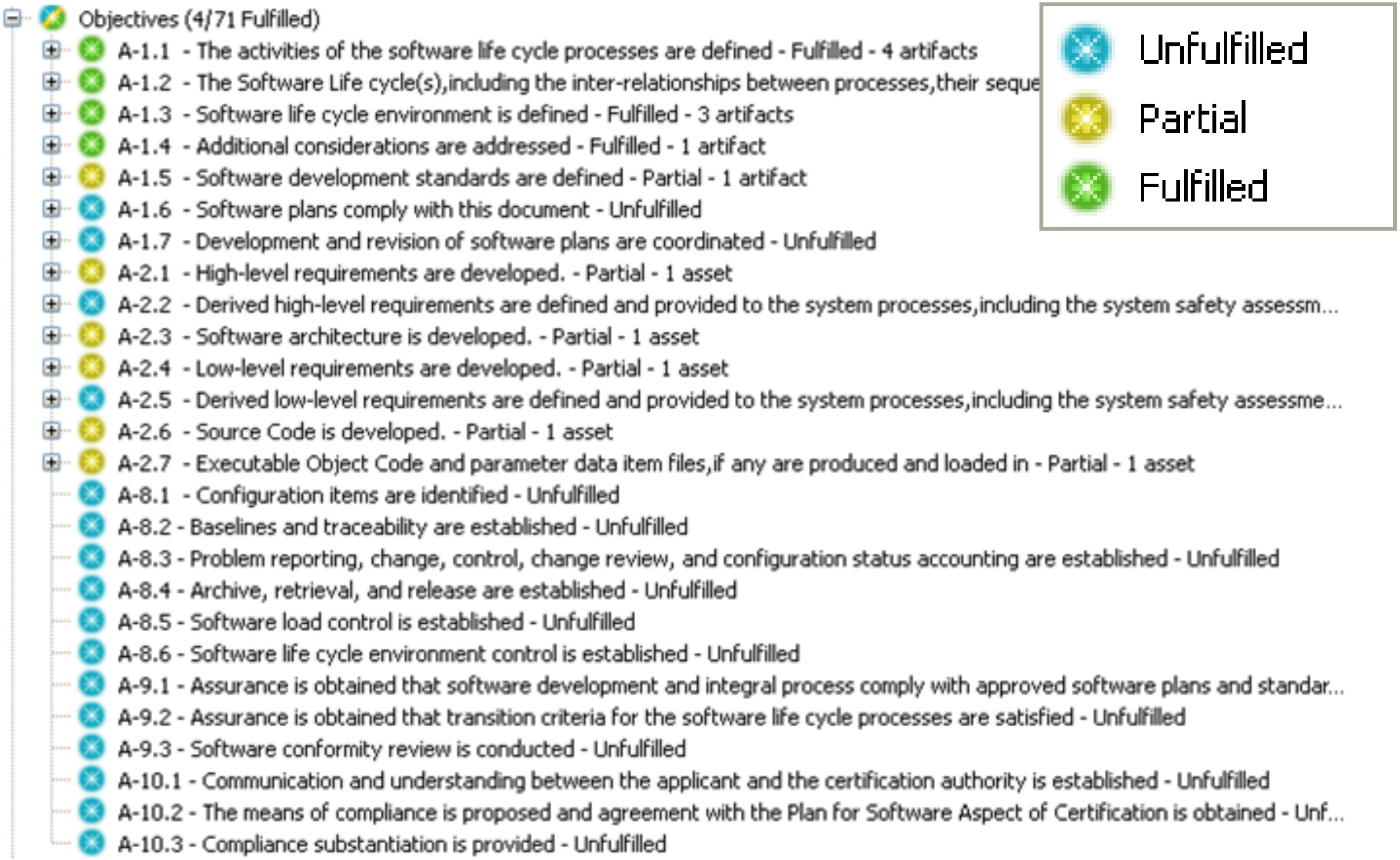
Meeting DO-178C Objectives with the LDRA tool suite

LDRA tool suite® Integrated Solution



DO-178C Objectives

LDRA



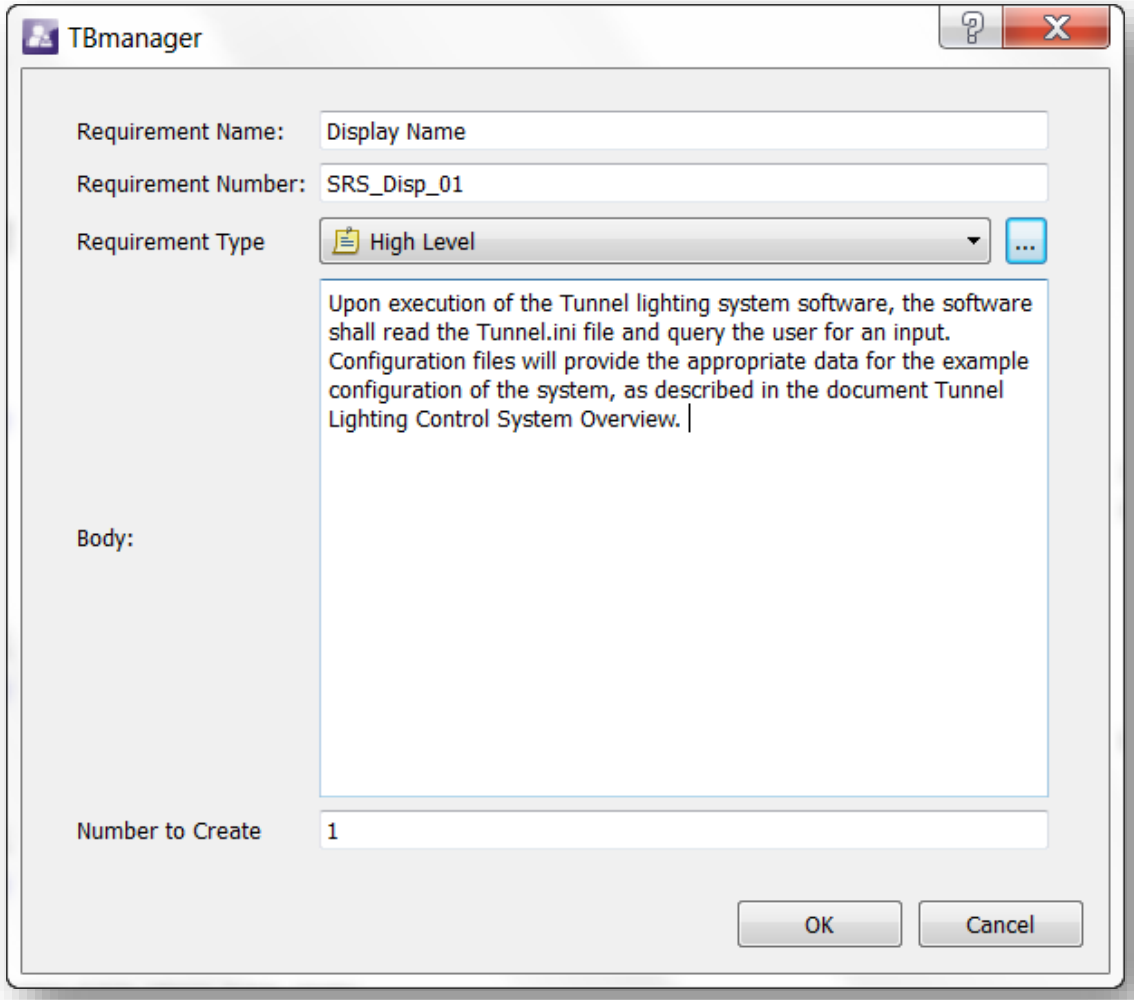
Objectives - Requirements

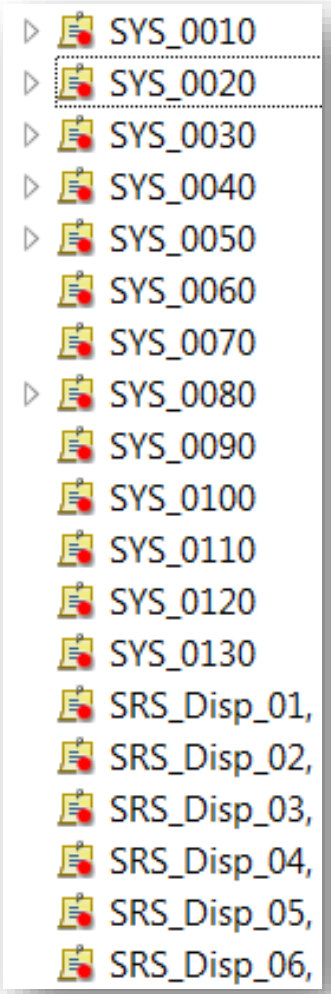
LDRA

- 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

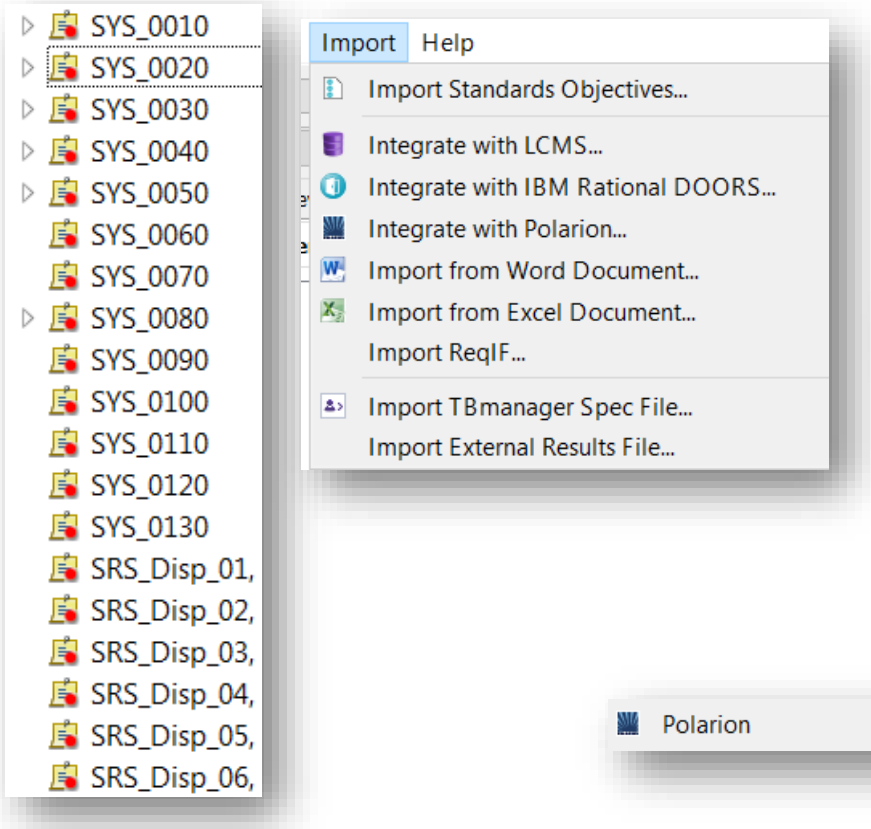
- In TBmanager, we can create High Level Requirements.



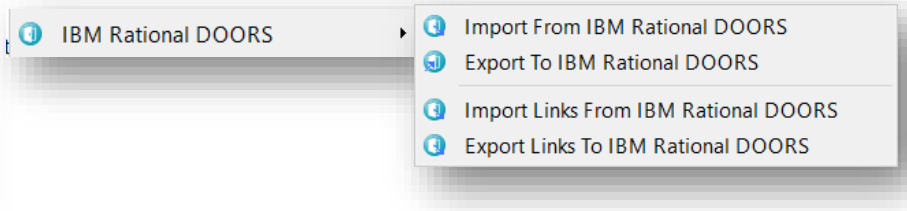


A 2.1 - High-level requirements are developed

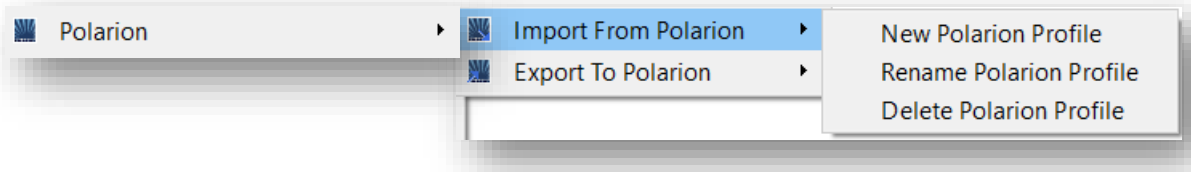
- TBmanager can import High-level requirements from Requirement management tools like DOORS, Polarian,Word,Excel,ReqIF



Importing requirements from DOORS

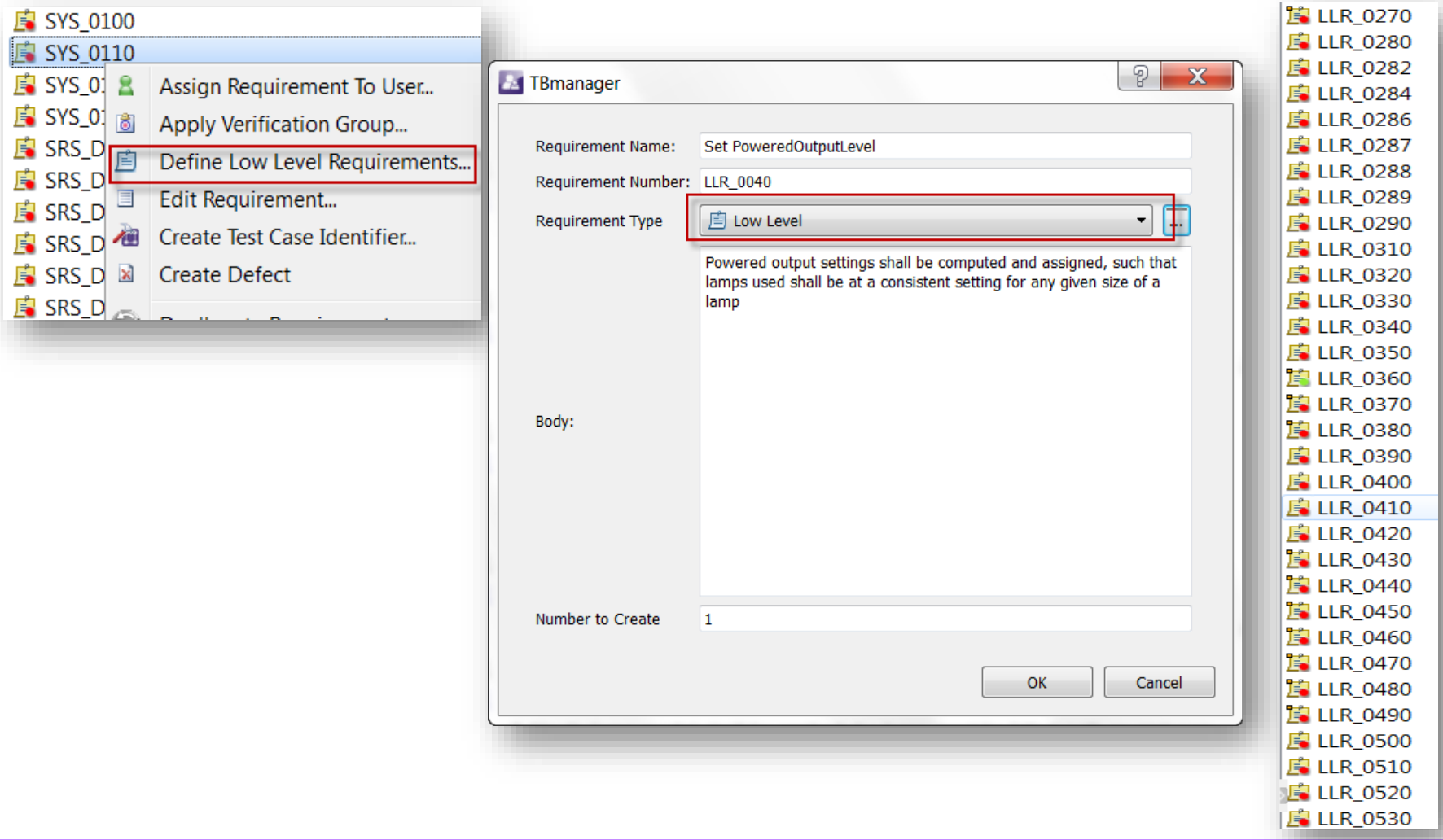


Importing requirements from Polarion



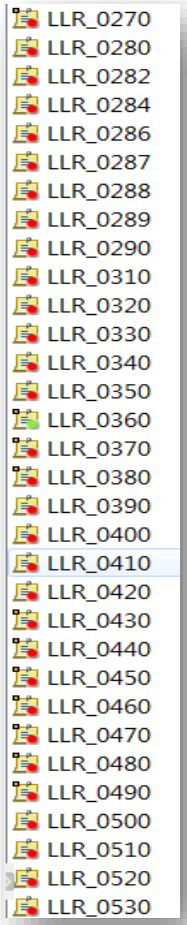
A 2.4 - Low-Level requirements are developed

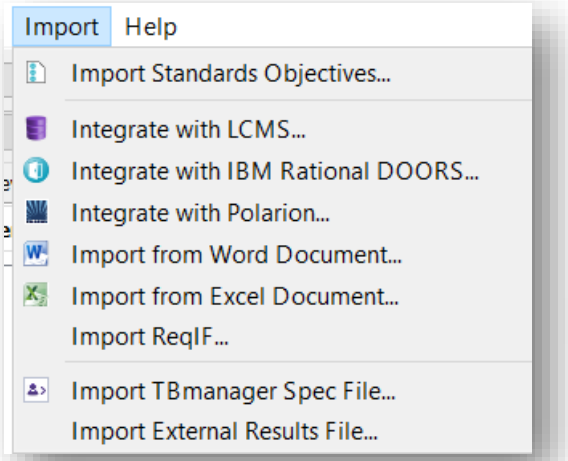
- In TBmanager, Low Level Requirement can be created for corresponding High level Requirement



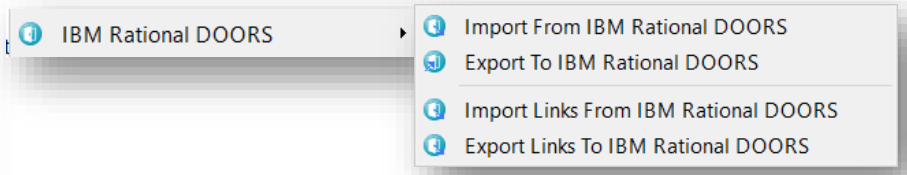
A 2.4 - Low-Level requirements are developed

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

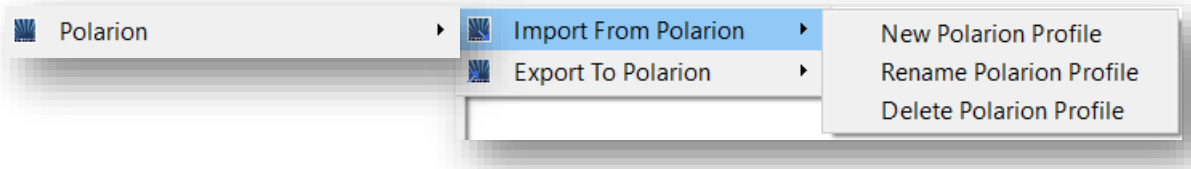




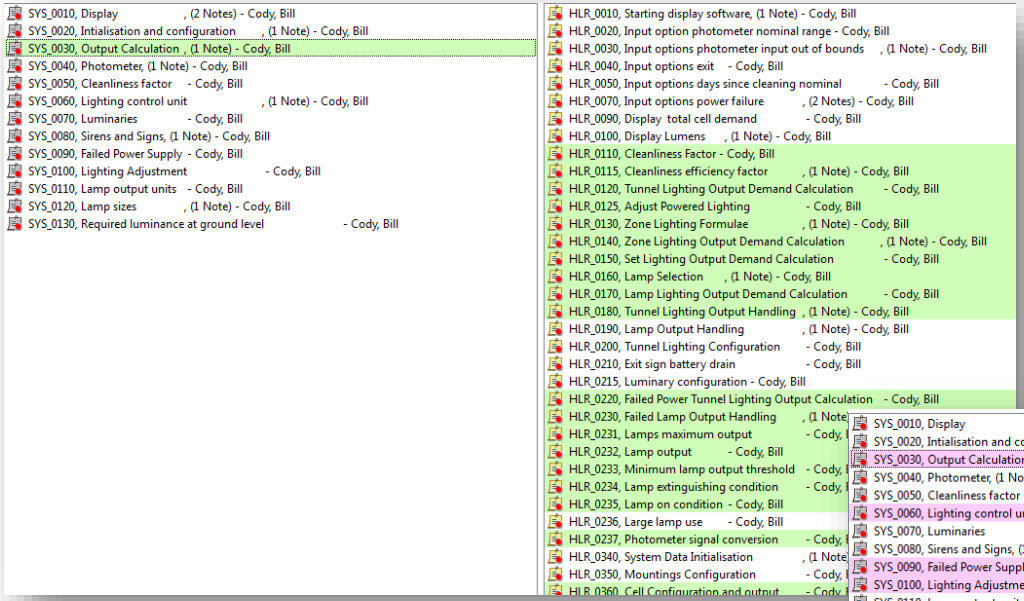
Importing requirements from DOORS



Importing requirements from Polarion



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



HLR ->System Req

System Req ->HLR

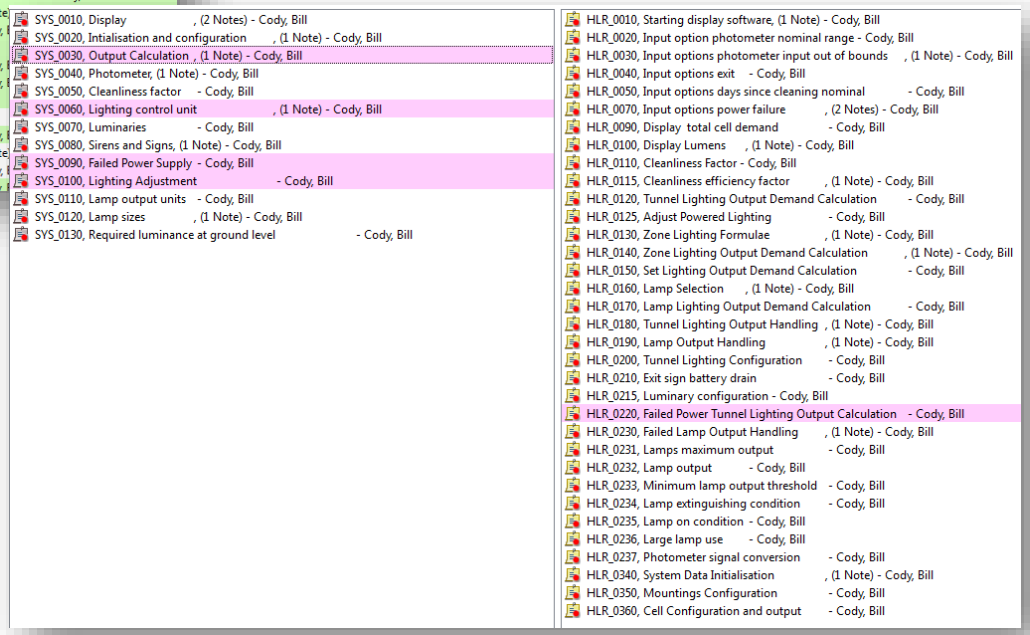
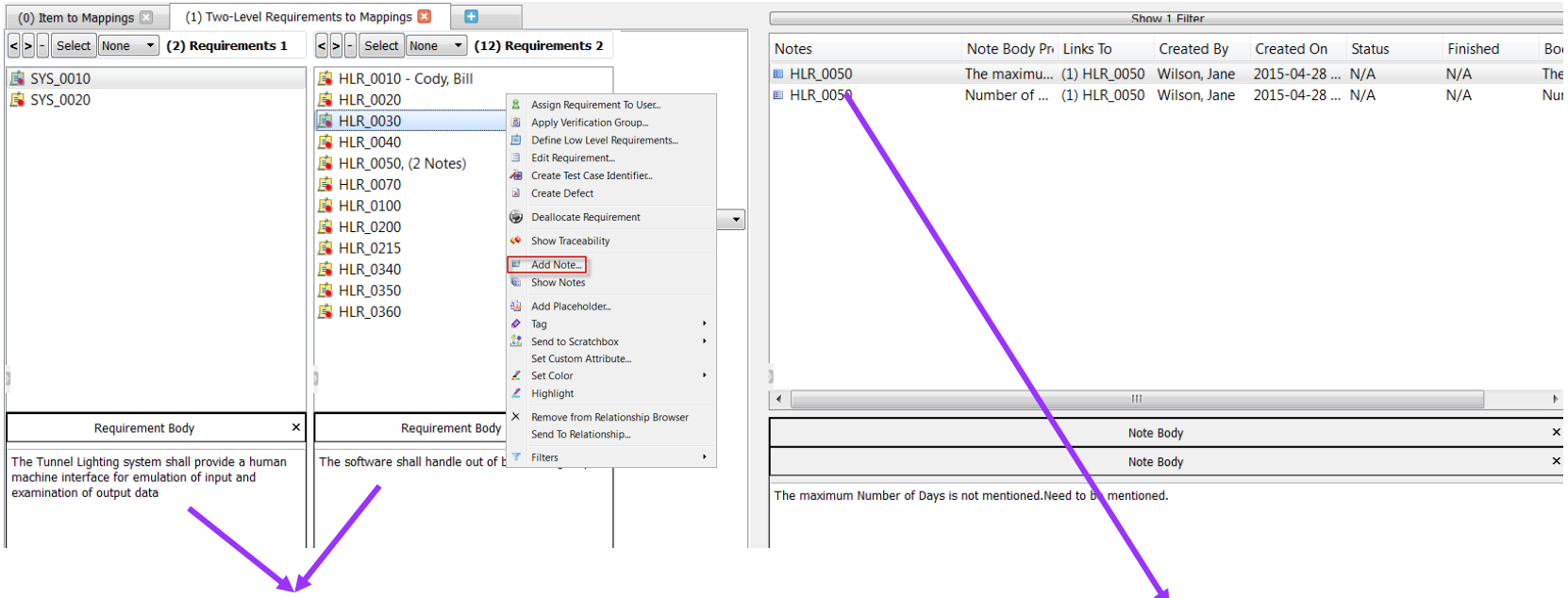


Table A- 3 2 - High-level requirements are accurate and consistent

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

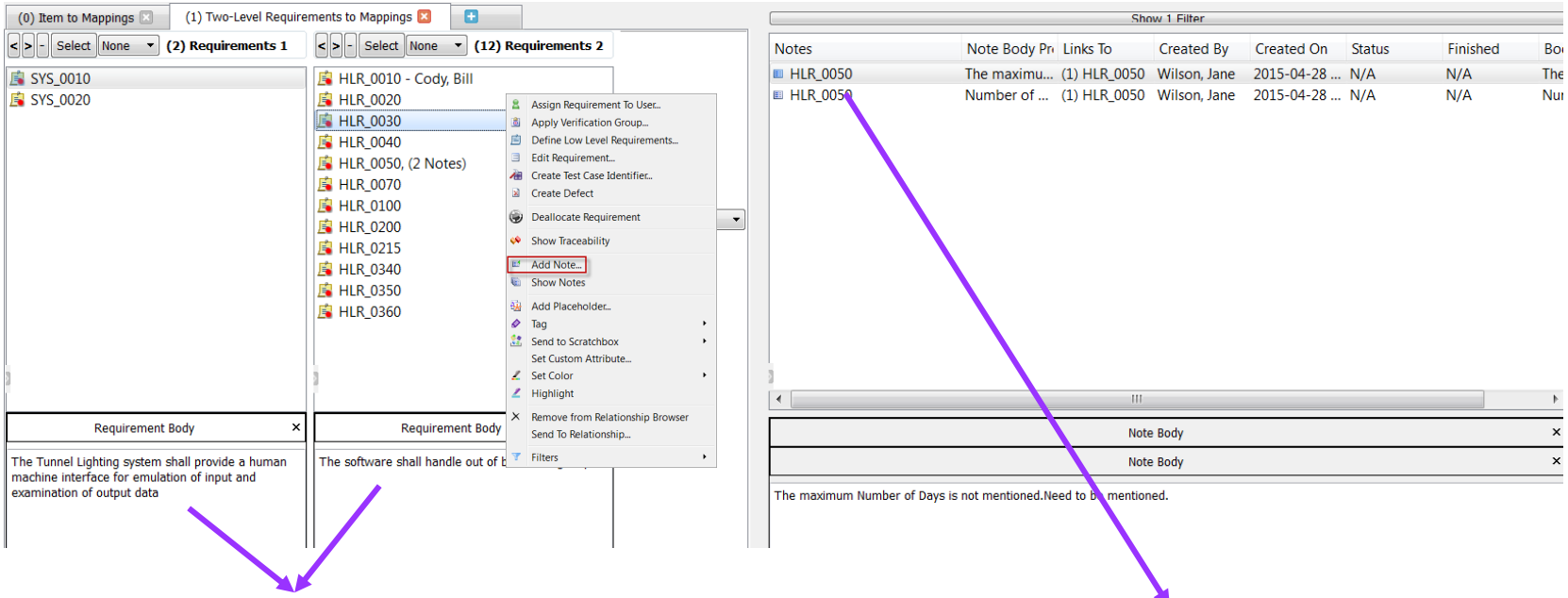


A review can be done on each requirement as body of requirement is visible in TBmanager

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

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

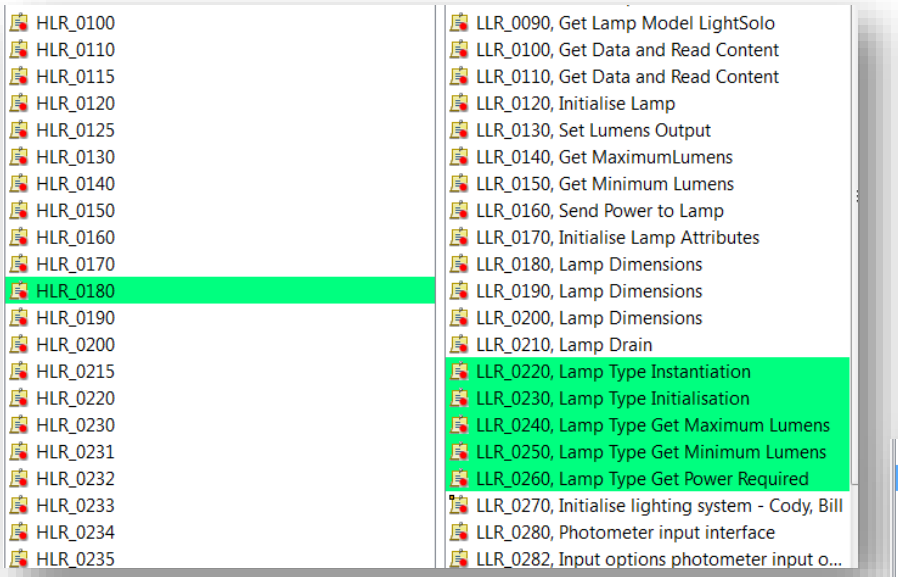
- TBmanager allows to review Low Level requirements and add corresponding Notes/Defects for each of Requirement.



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

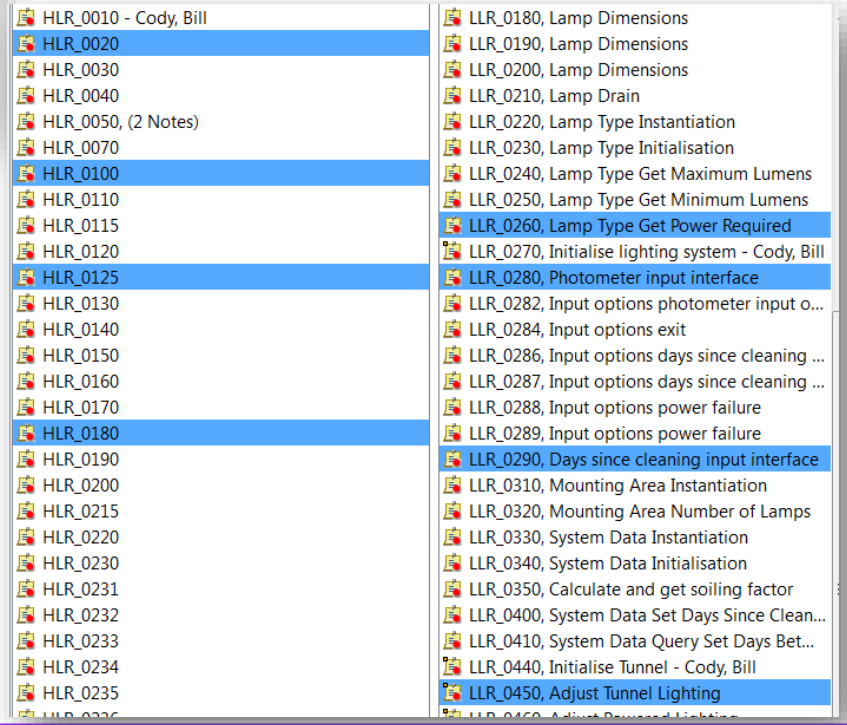
Corresponding NOTES can be added if requirements are not correct/consistent

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



HLR -> LLR

LLR ->HLR



Objectives – Source Code

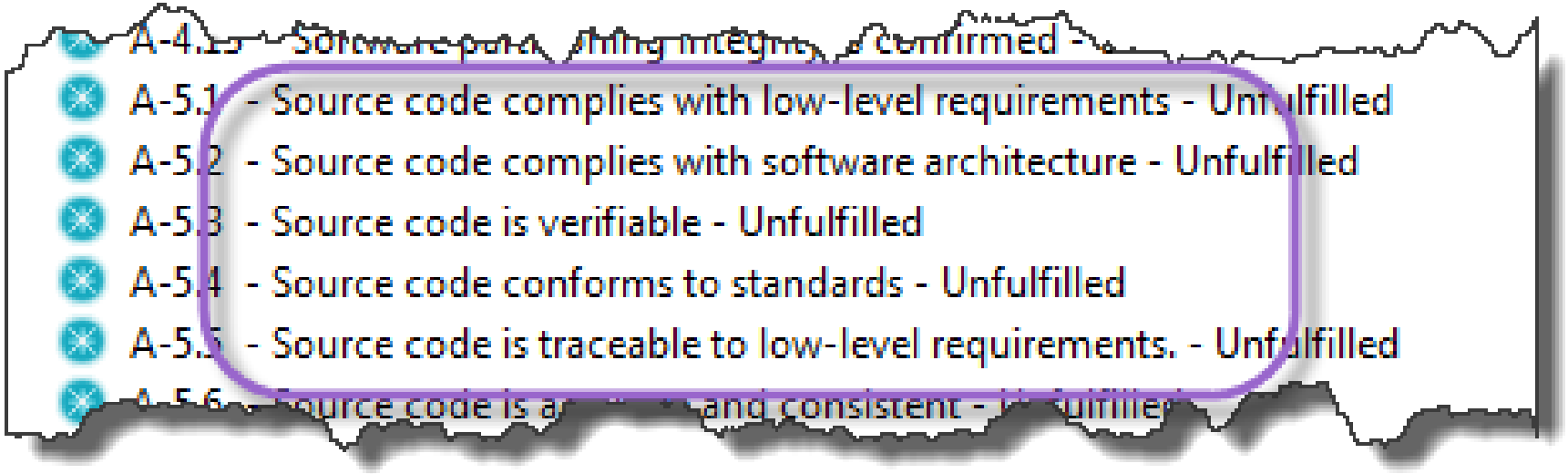


Table A-5 4 - Source Code conforms to standards

- TBvision performs code conformance with industry programming standards.

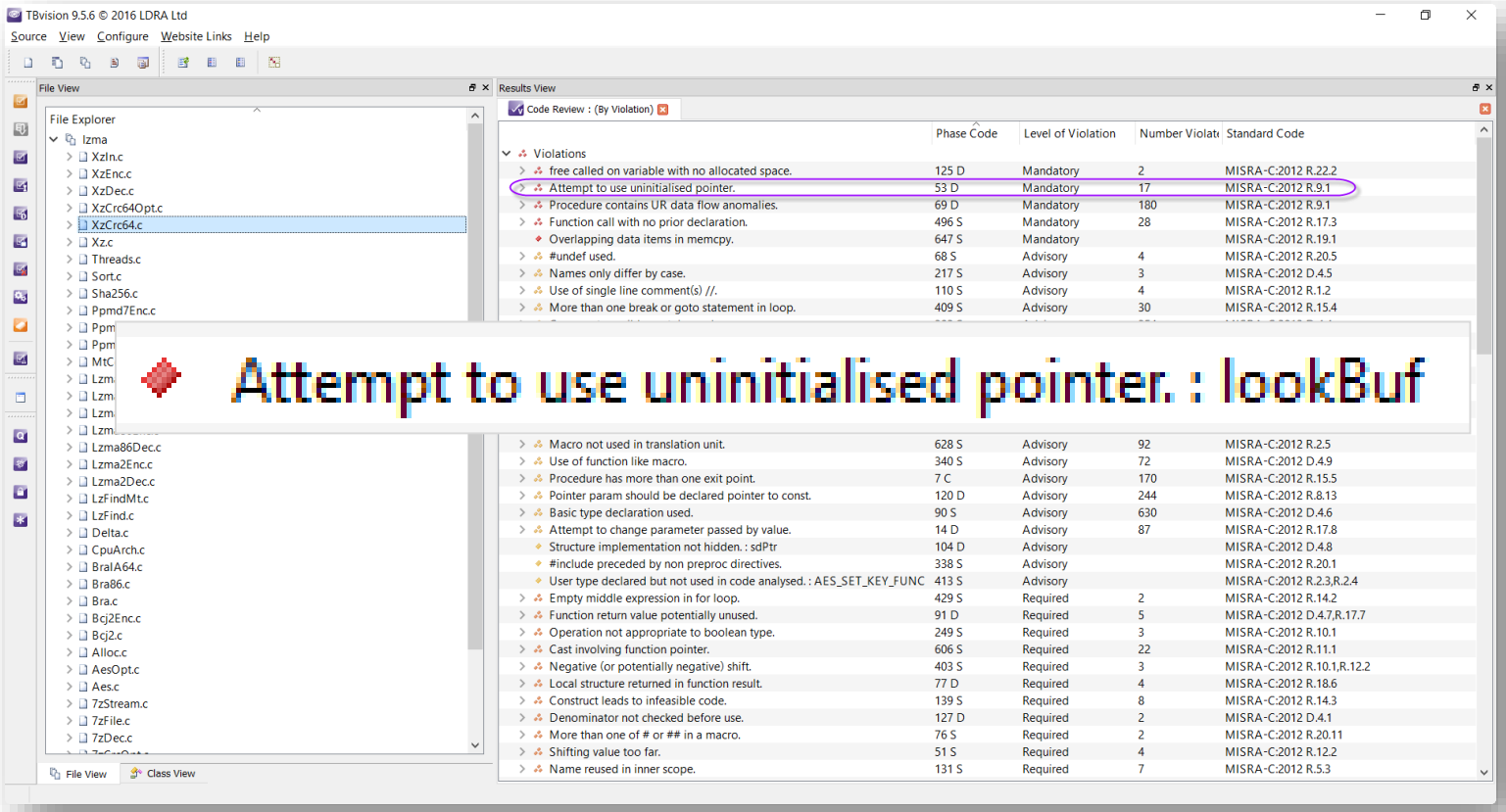
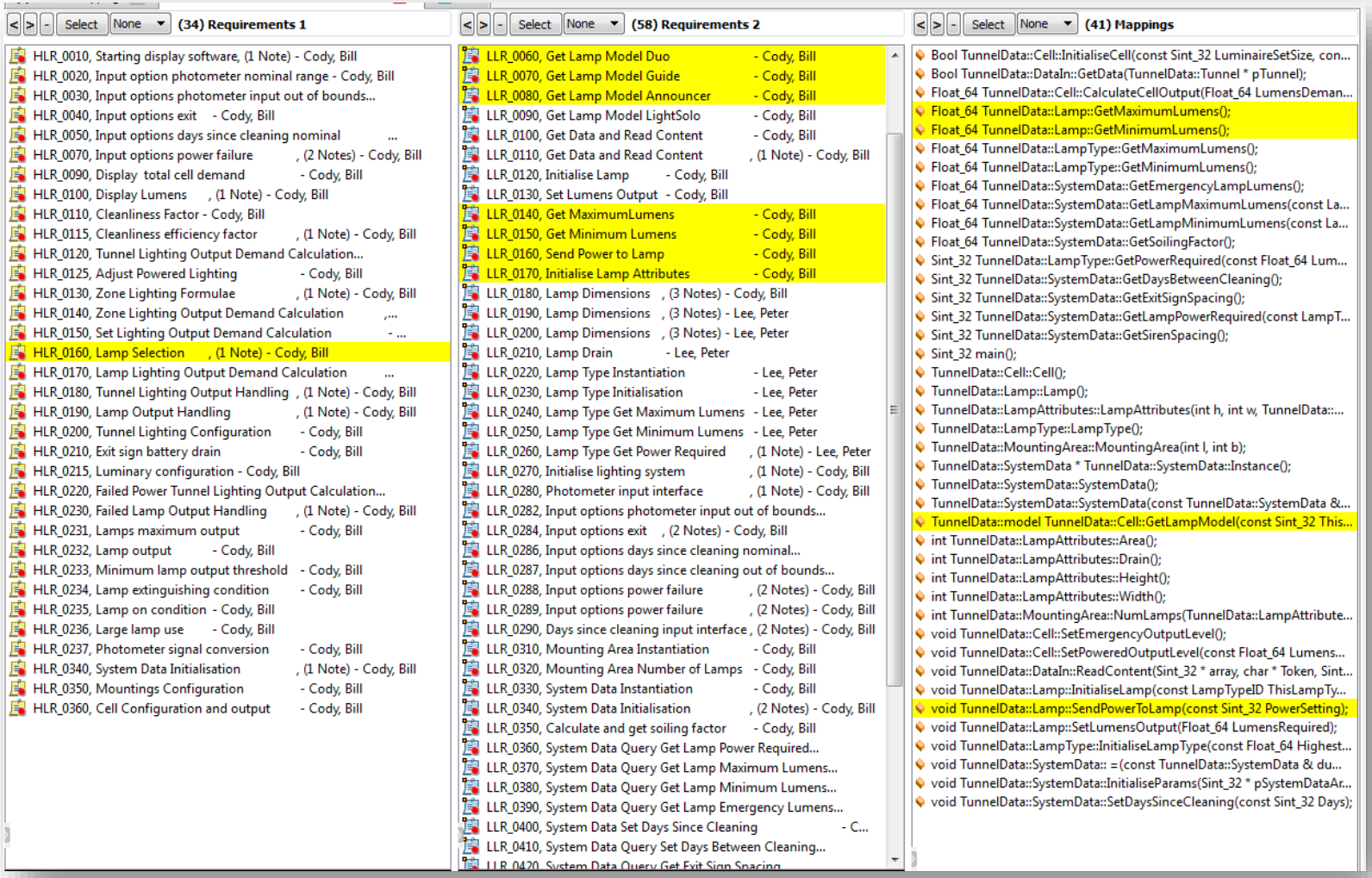
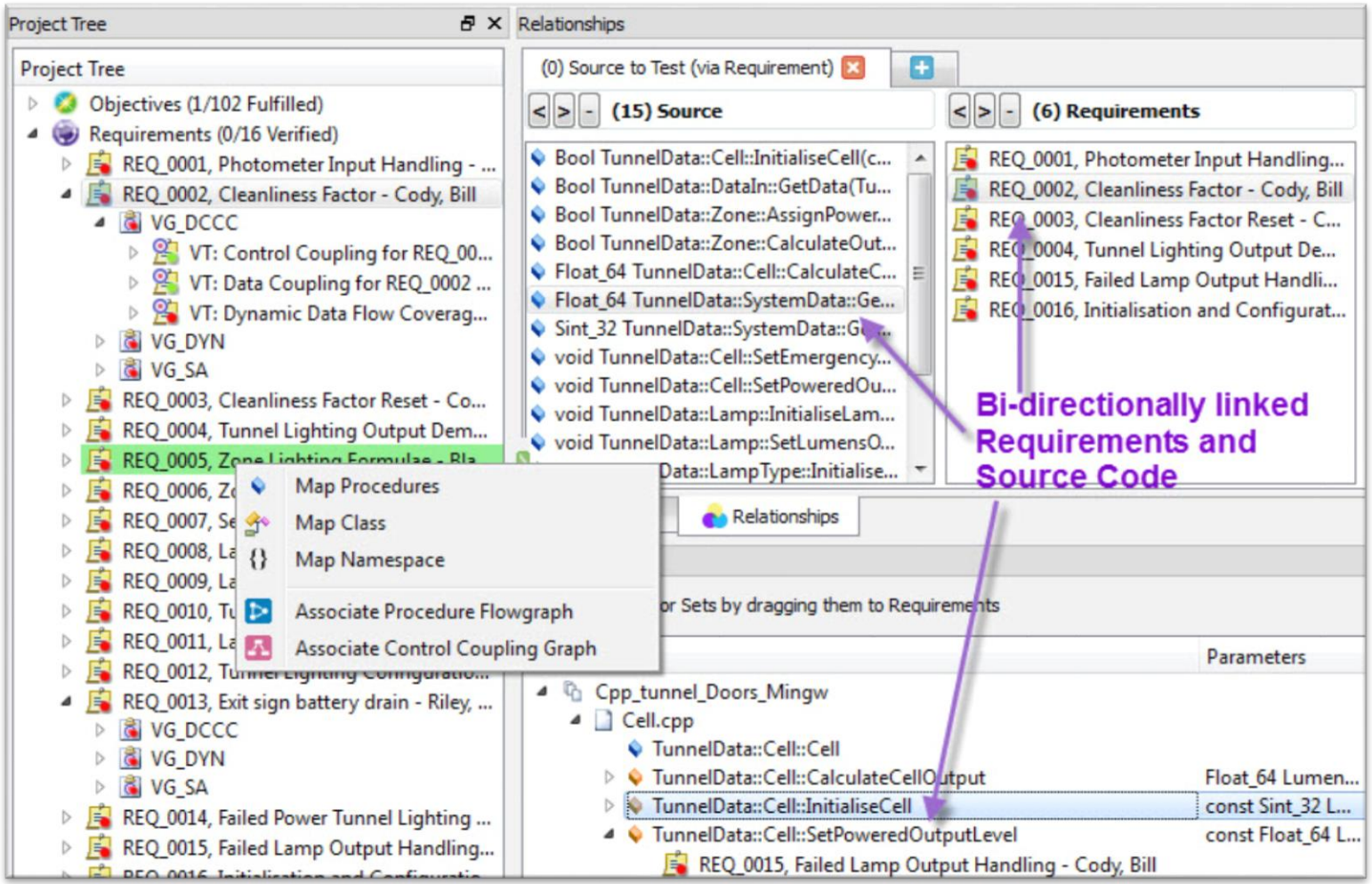


Table A-5 5 - Source Code is traceable to low-level requirements

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



Building and Maintaining Links



VISUAL LINKING AND LIVE BI-DIRECTIONAL TRACEABILITY

Objectives – Test

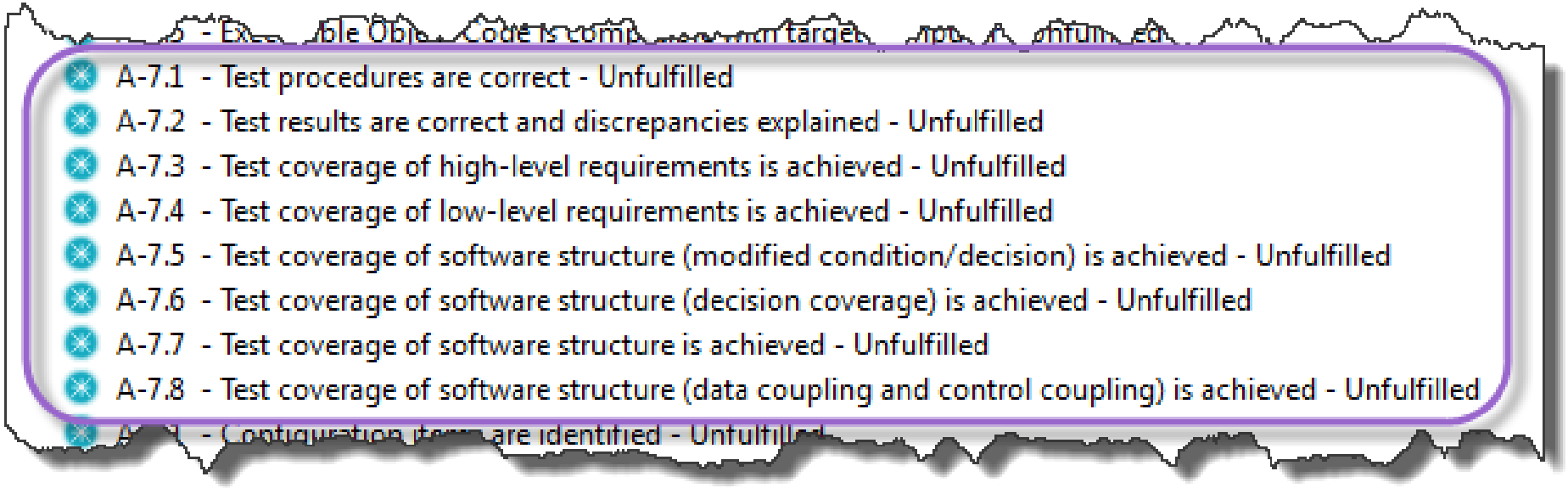


Table A-7 5 - Test coverage of software structure (MC/DC) is achieved

LDRA

- Generates a Test Case Planner

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

```
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 )
```

```
=====================================================================
                             EXPECTED
INDEX      C1 C2 C3          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 )  =   T     .........|..|..C3          #
4          f( T, F, F )  =   F     .........|..C2.C3          #*
5          f( F, T, T )  =   T     .........|.......C3        #
6          f( F, T, F )  =   F     .........C1.......C3       #*
7          f( F, F, T )  =   T     ....................C3
8          f( F, F, F )  =   F     ....................C3
```


```
=====================================================================
                             EXPECTED
INDEX      C1 C2 C3          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 )  =   T     .........|..|..C3          #
4          f( T, F, F )  =   F     .........|..C2.C3          #*
5          f( F, T, T )  =   T     .........|.......C3        #
6          f( F, T, F )  =   F     .........C1.......C3       #*
7          f( F, F, T )  =   T     ....................C3
8          f( F, F, F )  =   F     ....................C3
```

```
=====================================================================
                             EXPECTED
INDEX      C1 C2 C3          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 )  =   T     .........|..|..C3          #
4          f( T, F, F )  =   F     .........|..C2.C3          #*
5          f( F, T, T )  =   T     .........|.......C3        #
6          f( F, T, F )  =   F     .........C1.......C3       #*
7          f( F, F, T )  =   T     ....................C3
8          f( F, F, F )  =   F     ....................C3
```

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	.....C1.C2	#*
3	f(T, F, T) =	T	NO	NO	NO	.....|..|..C3	#
4	f(T, F, F) =	F	NO	YES	YES	.....|..C2.C3	#*
5	f(F, T, T) =	T	NO	NO	NO	.....|.....C3	#
6	f(F, T, F) =	F	NO	YES	YES	.....C1.....C3	#*
7	f(F, F, T) =	T	NO	NO	NO	.....C3	
8	f(F, F, F) =	F	NO	YES	YES	.....C3	

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

Detailed reporting on which branches/decisions are executed

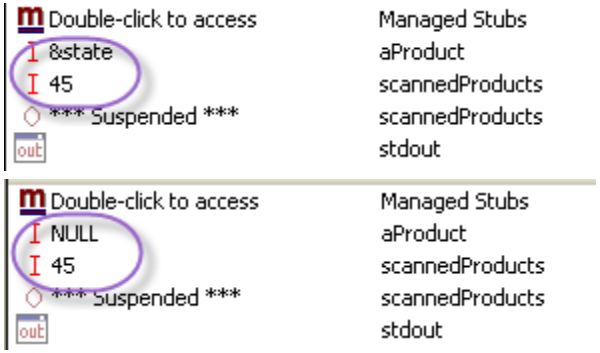
Table A-7 6 - Test coverage of software structure (decision coverage) is achieved

```
/*
 * 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

2	PASS	addProduct
3	PASS	addProduct



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

LDRA

```
/*
 * 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)	}	-	-	-

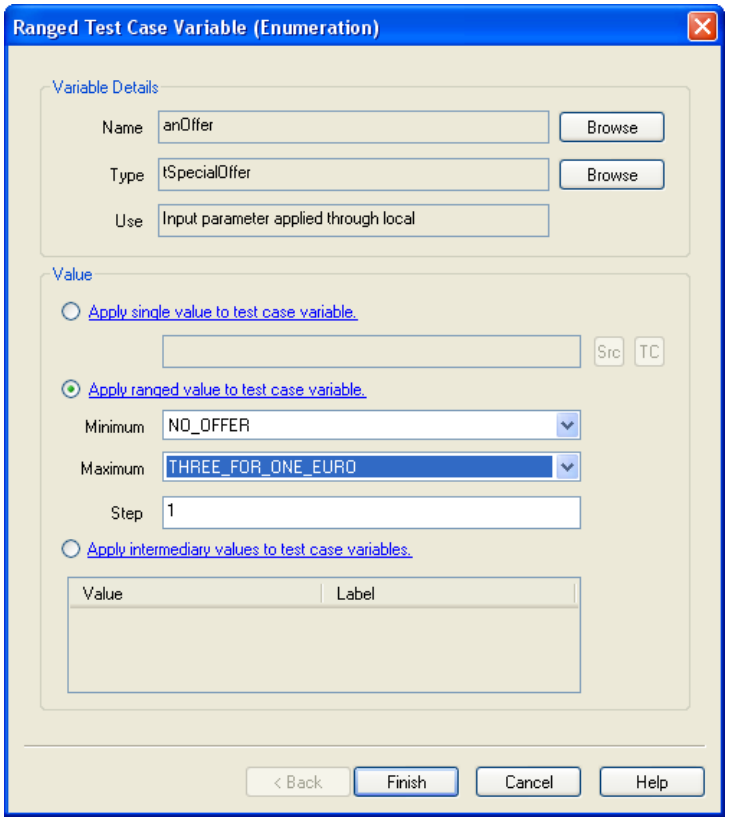


Table A-7 8- Test coverage of software structure (data coupling and control coupling) is achieved

- 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				
sum_total		Cashregister.c	generateTicket	L	E	117				
				L	R	152	164			
				L	D	117	152			
theBarcode		Cashregister.c	Cashregister_code	G	R	265 *****				
				G	D	266 *****				
	1 aBarcode	Cashregister.c	identifyProduct	P	E	174				
				P	R	177 *****				
theBarcode		Cashregister.c	Cashregister_key	G	R	290 *****				
				G	D	290 *****				
			Cashregister_start	G	D	301				
theCP		Productdatabase.c	Productdatabase_getSpecificCountedProduct	L	E	96				
				L	R	106				
				L	D	99				
						103 *****				
theChar		Main.c	main	L	E	25				

Use Coverage Summary

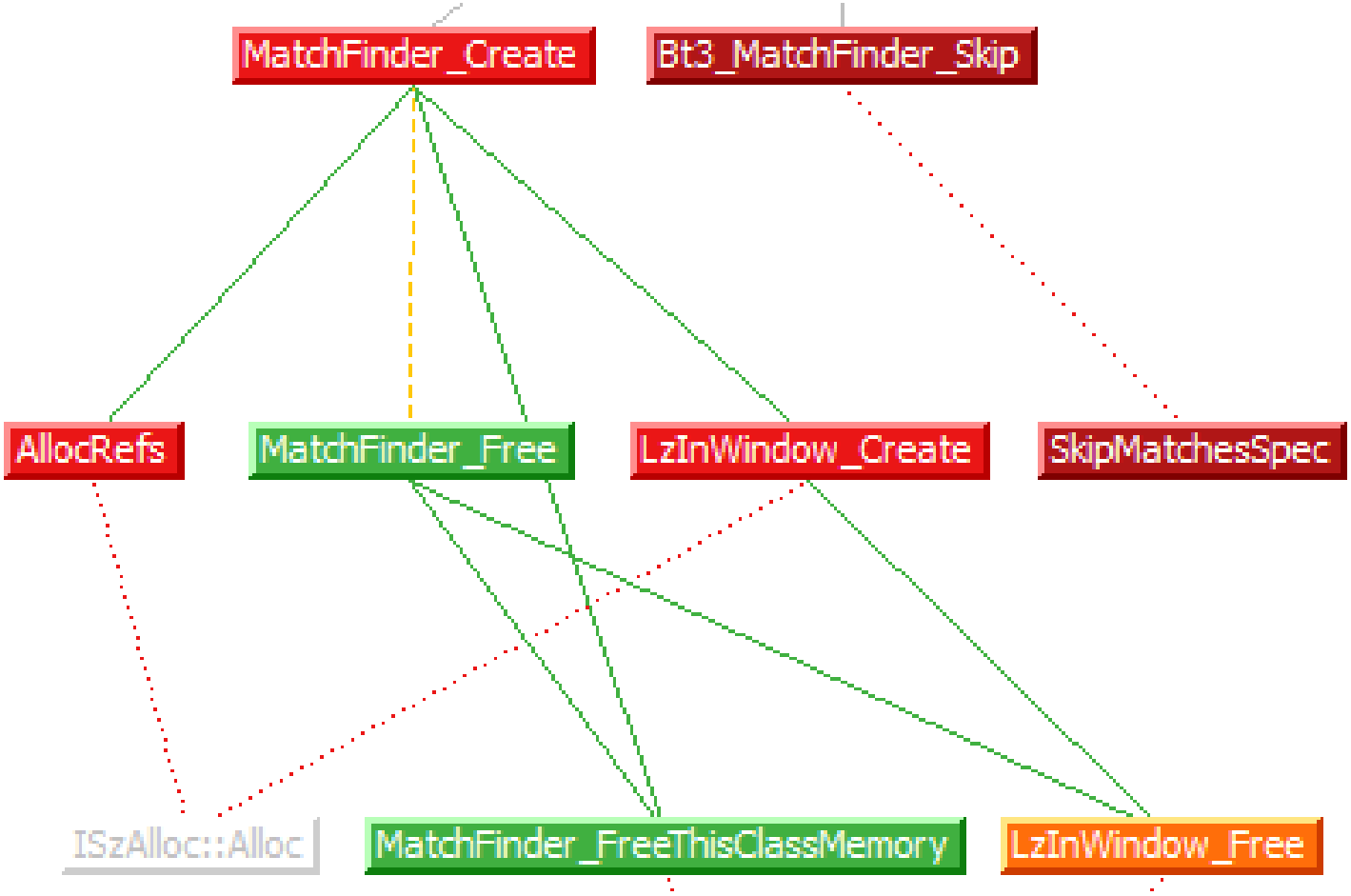
Summary	Uses
Total Uses	177
Uses Covered	166
Uses Not Covered	11
Overall Use Coverage (%)	**94**

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



The Benefits of using LDRA tools for DDCC

- 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 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 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)

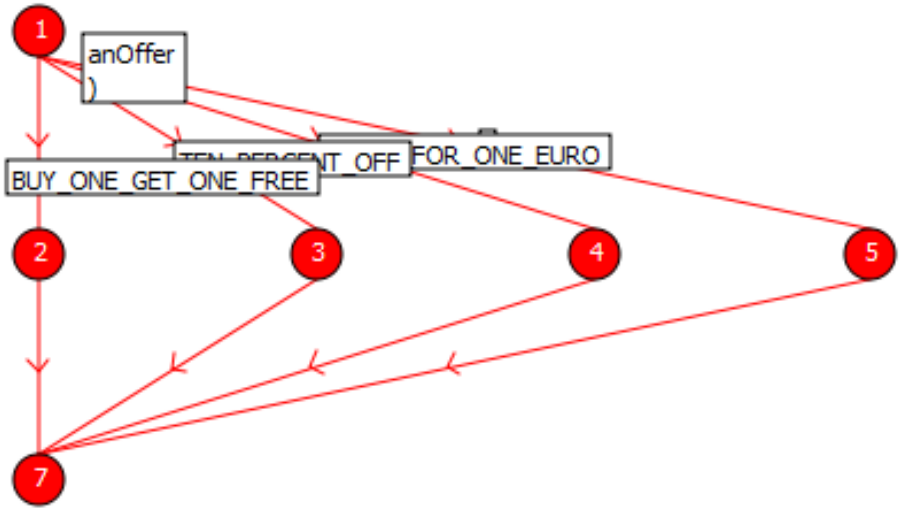
- Even if no options are set, then the control flow of the generated code will differ from the source code

- 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 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



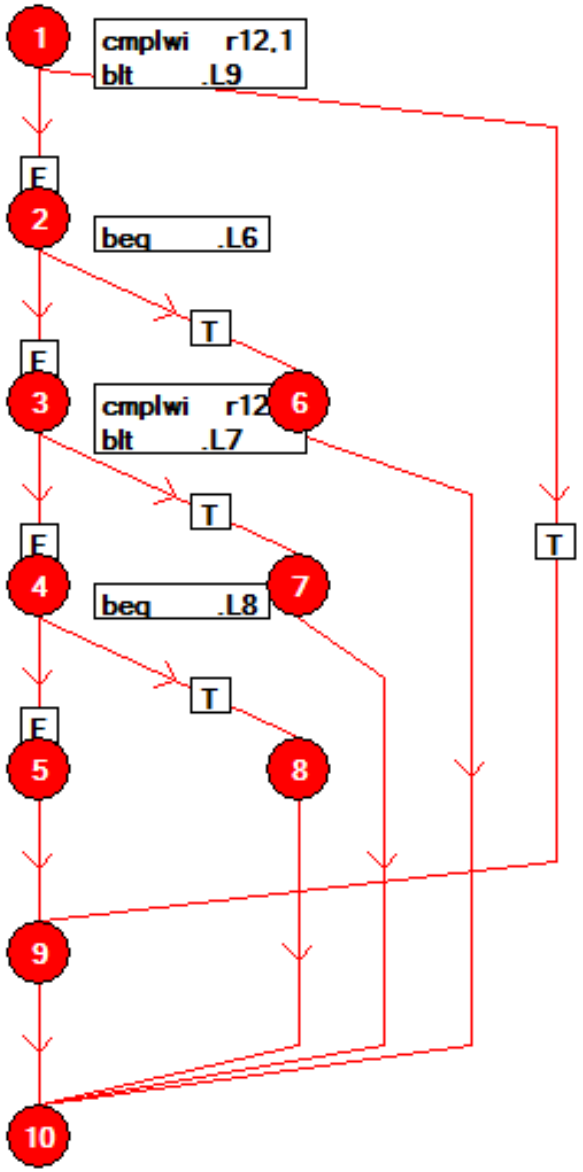


Table A-7 9 - Verification of additional code that can not be traced to Source Code, is achieved

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

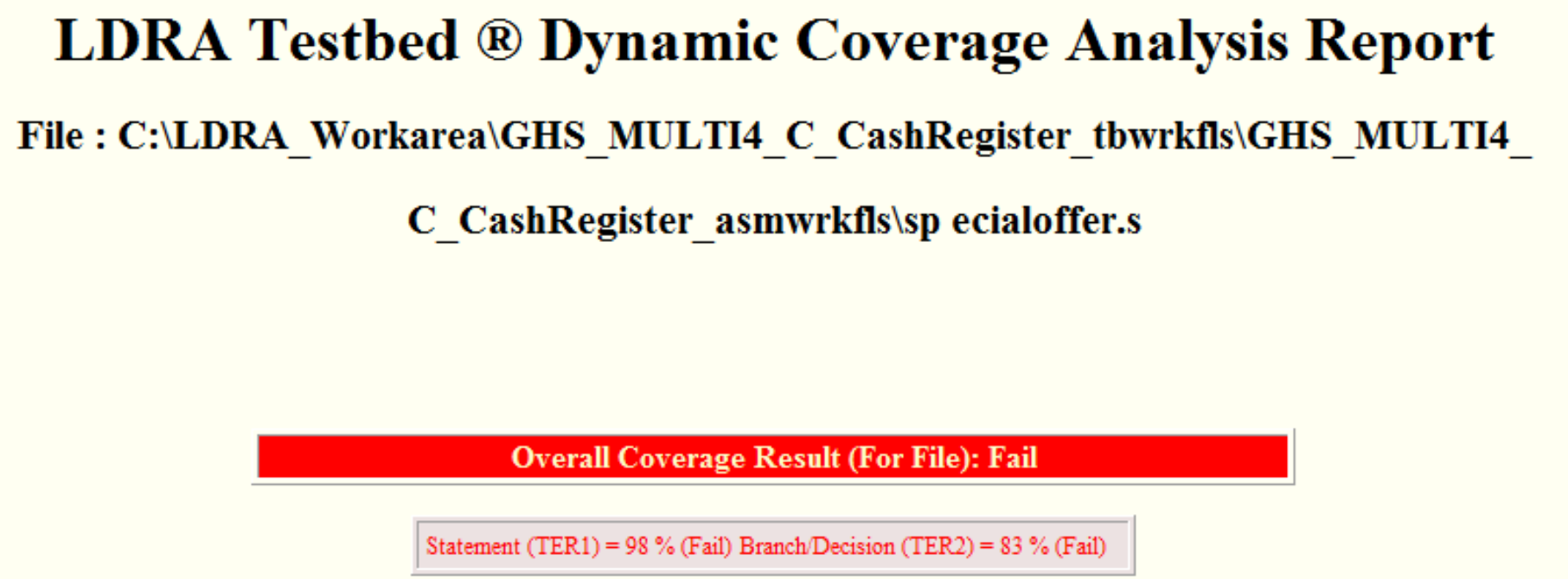
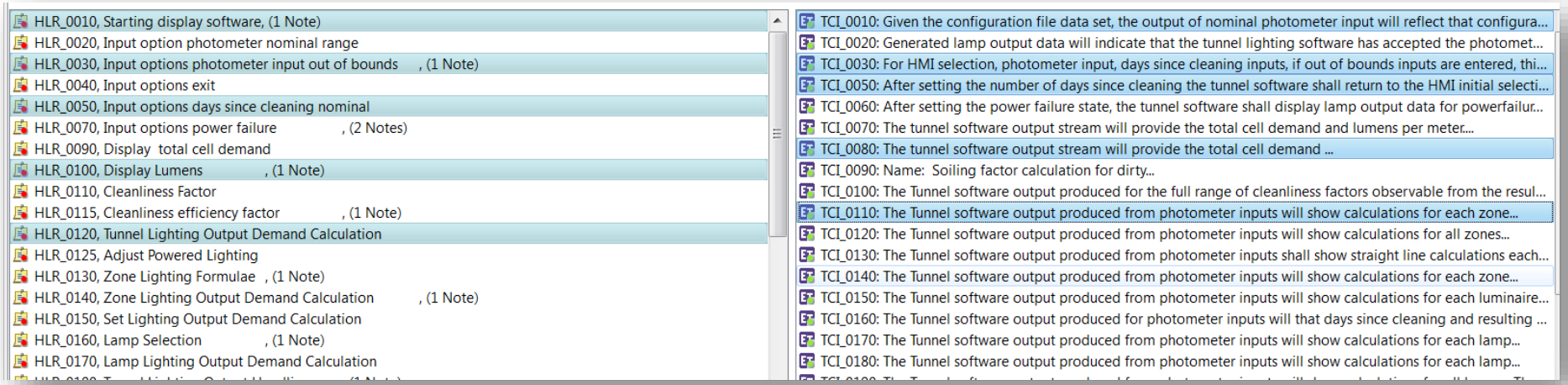


Table A-7 9 - Verification of additional code that can not be traced to Source Code, is achieved

- 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.5CGHS_MULTI4_C_CashRegister_tbwrkfls.5CGHS	-	-	-
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

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

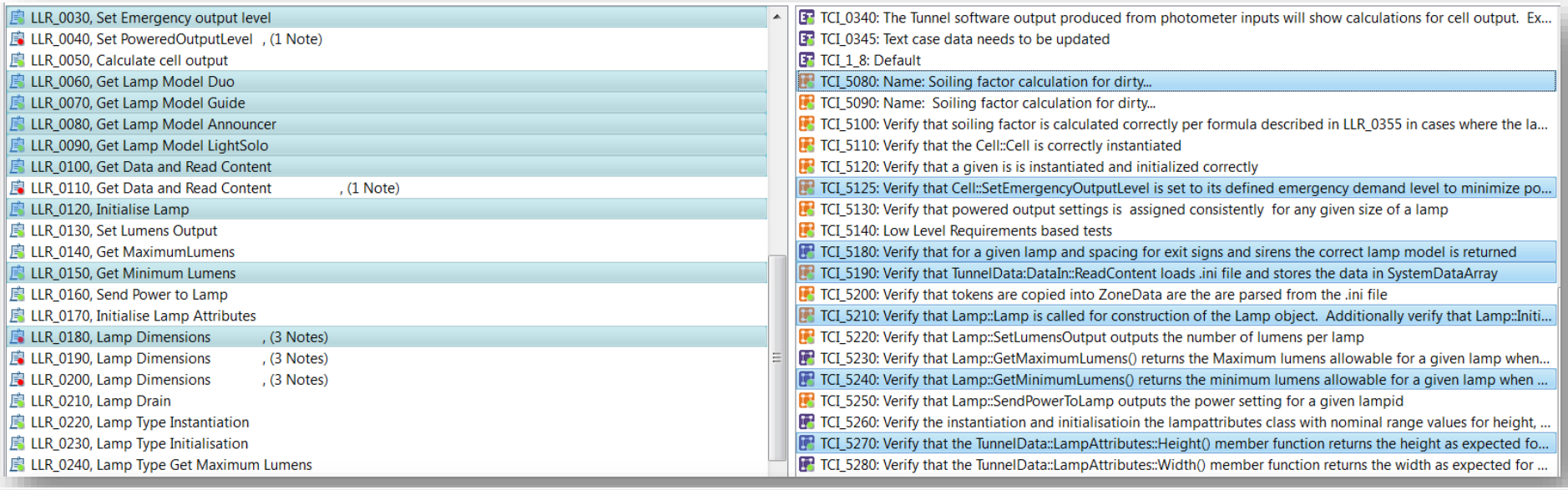


High Level Requirements Traceability Table Show/Hide

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	TCI_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	TCI_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	TCI_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	TCI_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 obscurity 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	TCI_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 mAps 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 mAps 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	TCI_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

LDRA

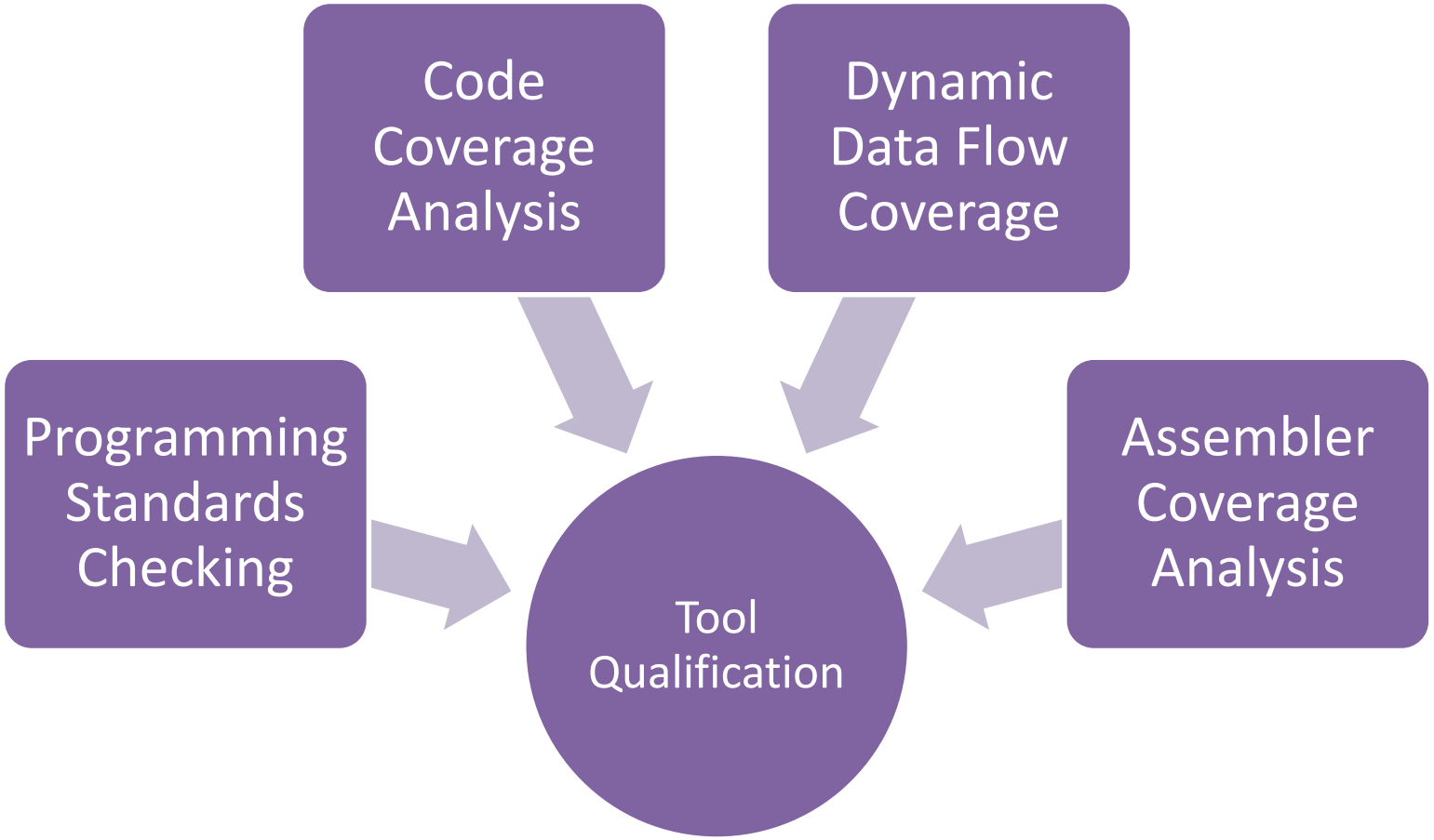


Low Level Requirements Traceability Table

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	TCI_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	TCI_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	TCI_5260: Verify the instantiation and initialisation the lampattributes class with nominal range values for height, width, and tunndata::model	Verify the instantiation and initialisation the lampattributes class with nominal range values for height, width, and tunndata::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	TCI_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	TCI_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	TCI_5480: Verify that TunnelData::SystemData::GetDaysBetweenClearning returns the mDaysBetweenCleaning correctly	Verify that TunnelData::SystemData::GetDaysBetweenClearning 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	TCI_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	TCI_5120: Verify that a given is instantiated and initialised correctly	Verify that a given is instantiated and initialised correctly

Tool Qualification



Show Compliance

LDRA

Comprehensive Process Control Documents

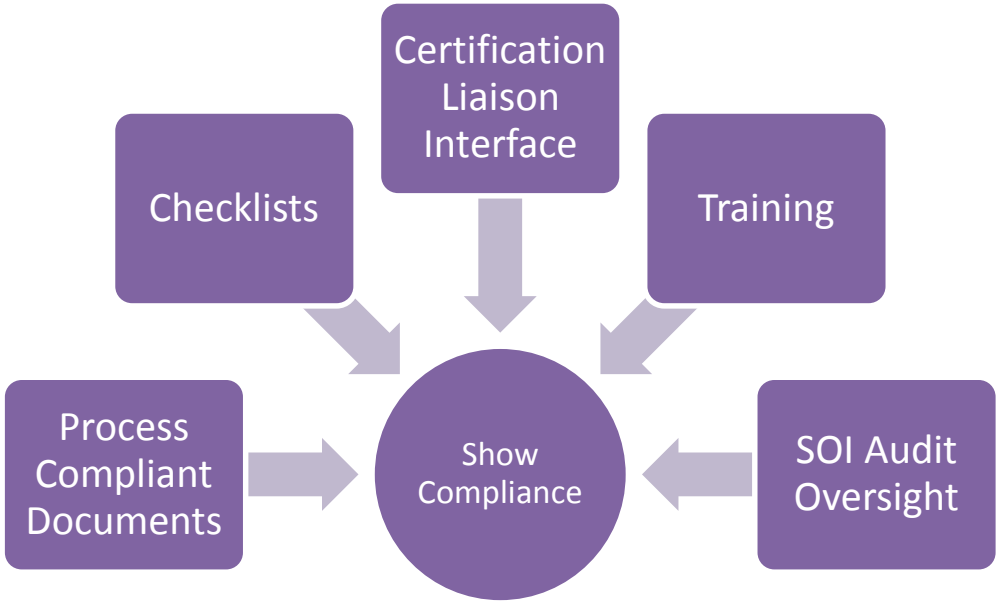
- 90% complete
- FAA & EASA
- DO-178 & DO-254

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