



8 Key Aspects of ED-12C / DO-178C Compliance



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

8 Key Aspects of ED-12C / DO-178C Compliance

1. Traceability Management
2. Coding Standards
3. Model Based Development and Verification
4. Subclass Verification for Local Type Consistency
5. Data & Control Coupling Coverage
6. Object Code Verification
7. Compliance Knowledgebase
8. Compliance Management System

Aspect #1: Traceability Management

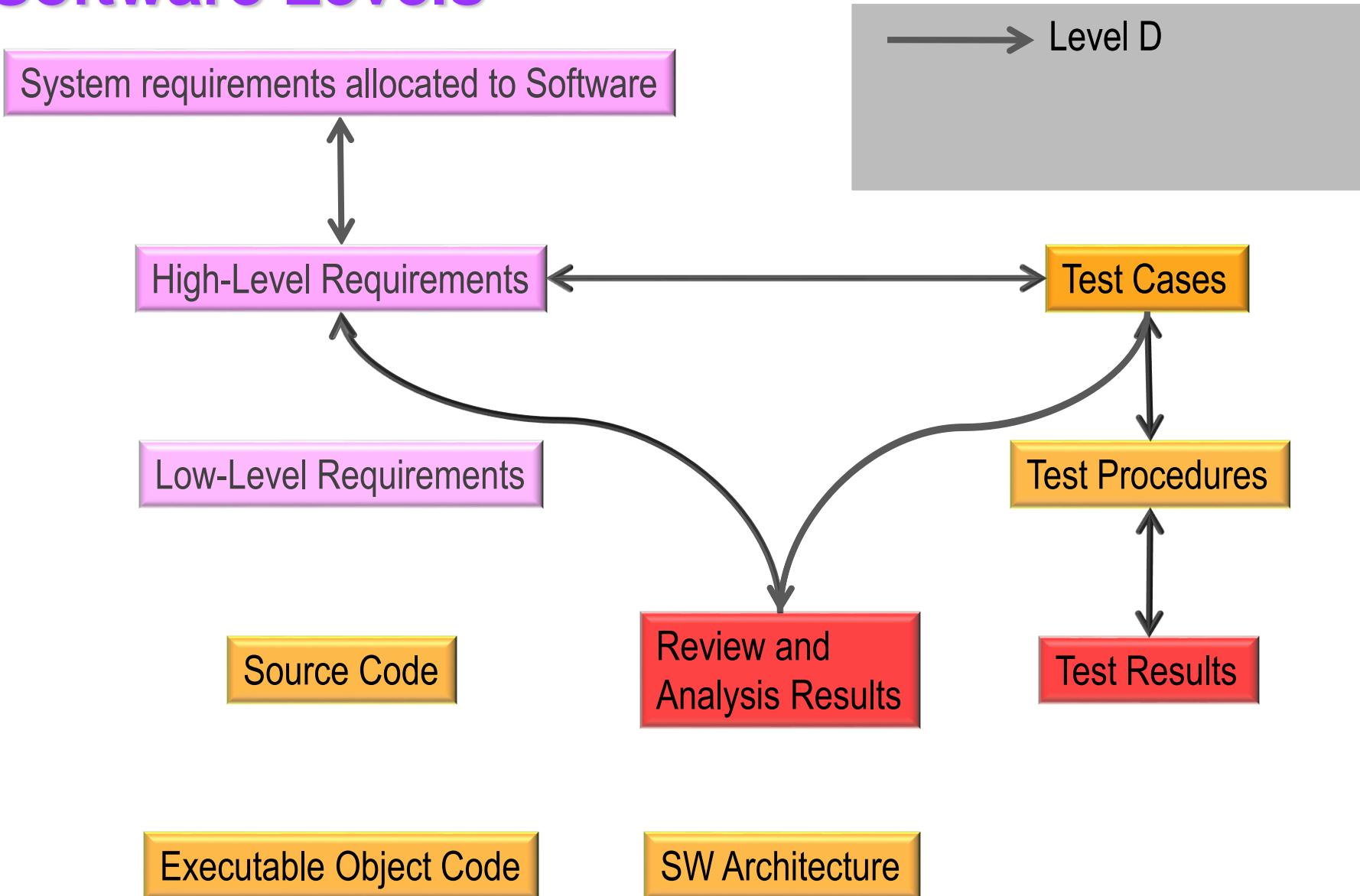


Delivering Software Quality and Security through
Test, Analysis & Requirements Traceability

Traceability Management

- Production and maintenance of a traceability matrix that shows:
 - The bidirectional relationship between requirements and their immediate higher-level requirements
 - The bidirectional relationship between source code and its associated low-level requirements
 - The bidirectional relationship between requirements and the test cases and procedures that verify those requirements
 - The relationship of objectives with development assets and verification artifacts

Requirements Traceability Across Software Levels



Requirements Traceability Across Software Levels

LDRA

System requirements allocated to Software

High-Level Requirements

Low-Level Requirements

Source Code

Review and Analysis Results

Executable Object Code

SW Architecture

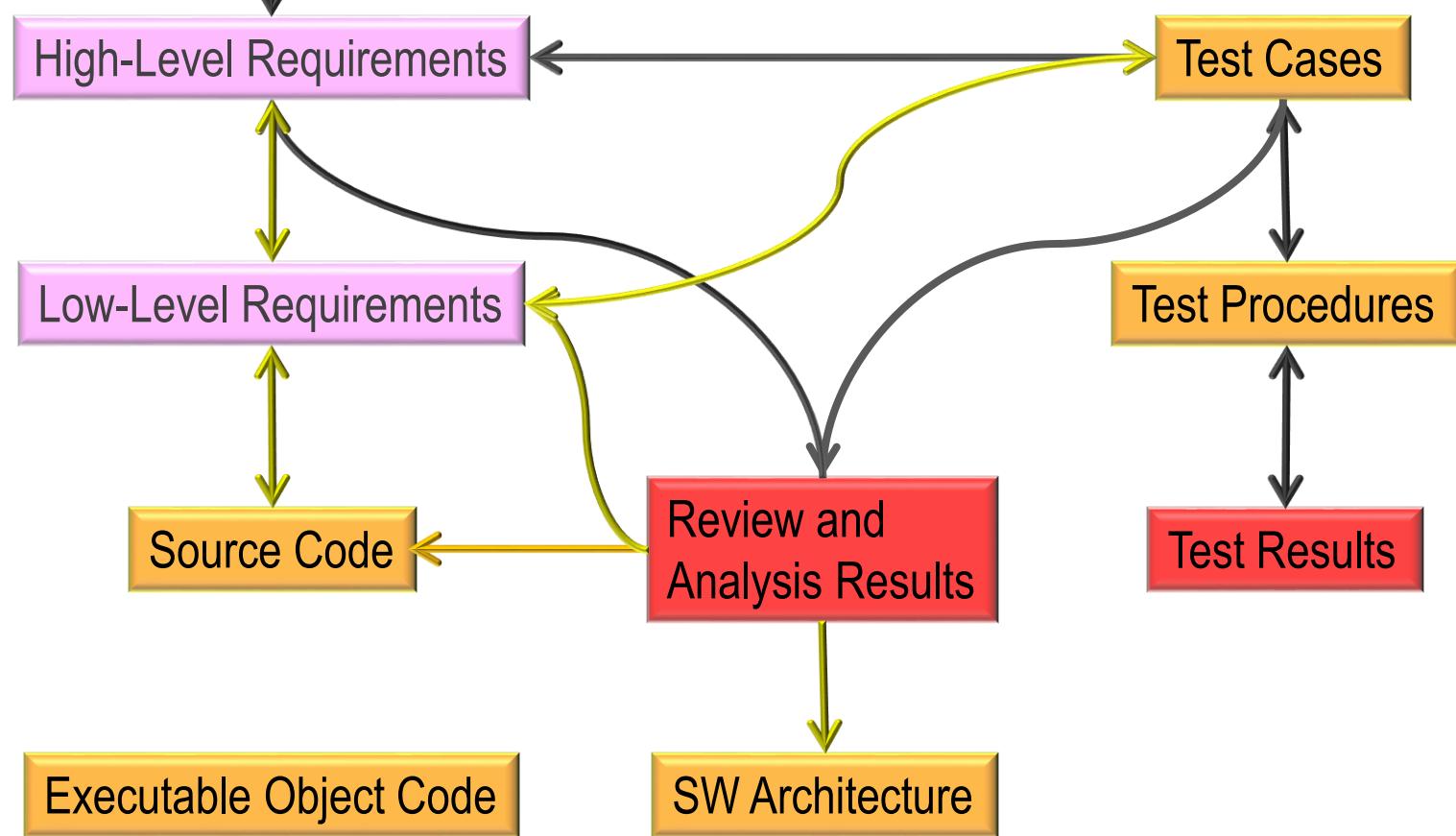
Test Cases

Test Procedures

Test Results

Level D

Levels B and C



Requirements Traceability Across Software Levels

LDRA

System requirements allocated to Software

→ Level D
→ Levels B and C
→ Level A

High-Level Requirements

Low-Level Requirements

Source Code

Executable Object Code

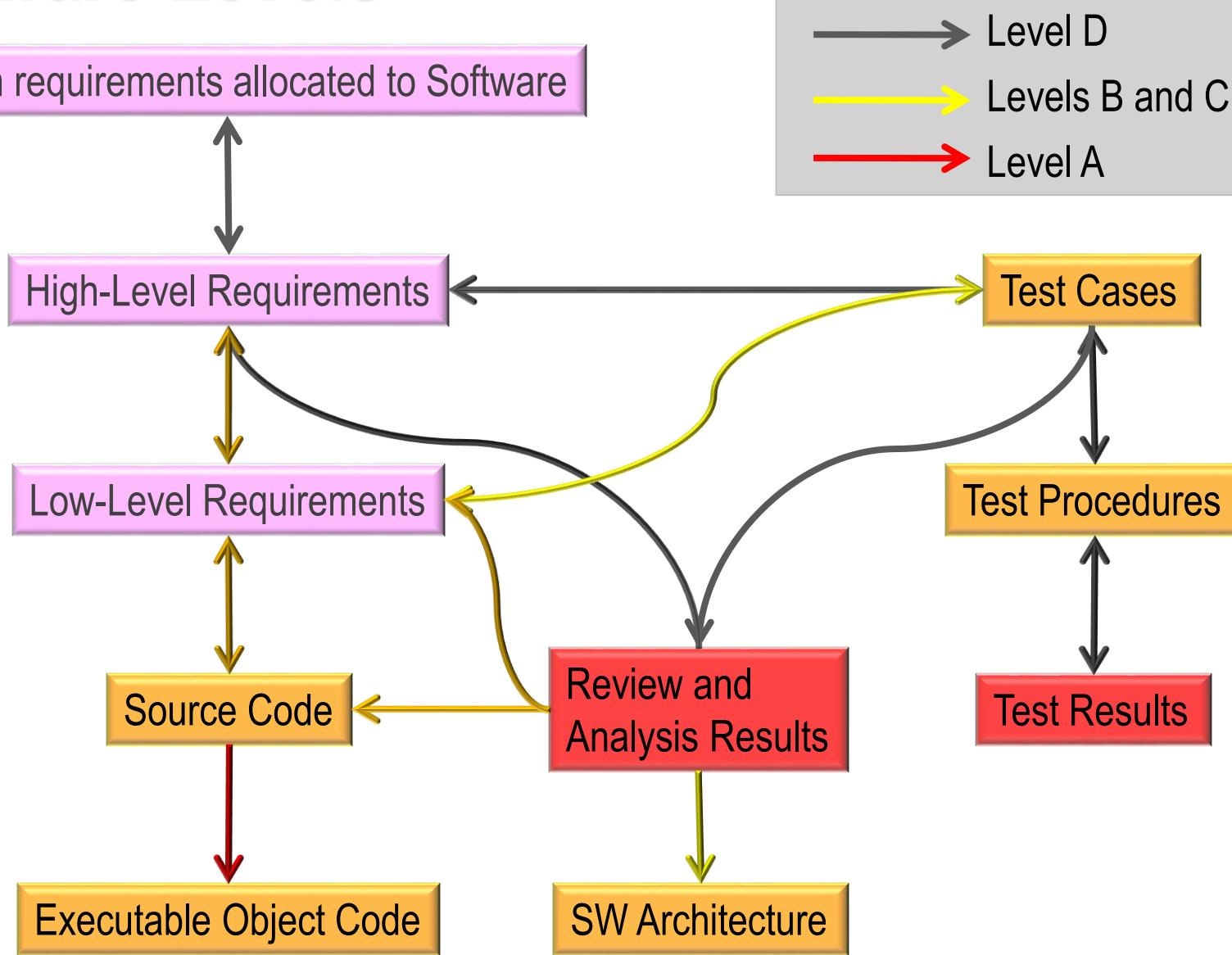
Review and Analysis Results

Test Cases

Test Procedures

Test Results

SW Architecture



DO-178C Objectives

Design Assurance Level	Objectives	Objectives that must be verified with independence
A - Catastrophic	71	30
B - Hazardous	69	18
C - Major	62	5
D - Minor	26	2
E - No Effect	-	-

Identifier

Table A-5 1

Name [²]

Source Code complies with low-level requirements

Standard

DO-178C

Standard Level

A - SI, B - SI, C - S, D -

References

6.3.4.a

DO-178C Objectives

- ▷ ✘ Table A-1 - Software Planning Process - Unfulfilled
- ▷ ✘ Table A-2 - Software Development Process - Unfulfilled - 1 asset
- ▷ ✘ Table A-3 - Verification of Outputs of Software Requirements Process - Unfulfilled
- ◀ ✘ Table A-4 - Verification of Outputs of Software Design Process - Unfulfilled
 - ✘ Table A-4 1 - Low Level requirements comply with high level requirements - Unfulfilled
 - ✘ Table A-4 2 - Low-level requirements are accurate and consistent - Unfulfilled
 - ✘ Table A-4 3 - Low-level requirements are compatible with target computer - Unfulfilled
 - ✘ Table A-4 4 - Low-Level requirements are verifiable - Unfulfilled
 - ✘ Table A-4 5 - Low-level requirements conform to standards - Unfulfilled
 - ✘ Table A-4 6 - Low-level requirements are traceable to high-level requirements - Unfulfilled
 - ✘ Table A-4 7 - Algorithms are accurate - Unfulfilled

- ✿ ✘ Table A-4 6 - Low-level requirements are traceable to high-level requirements - Unfulfilled
- ✿ ✘ Table A-4 7 - Algorithms are accurate - Unfulfilled
- ✿ ✘ Table A-4 8 - Software architecture is compatible with high-level requirements - Unfulfilled

- Table A-5 - Verification of Outputs of Software Coding and Process - Unfulfilled
 - ✘ Table A-5 1 - Source Code complies with low-level requirements - Unfulfilled
 - ✘ Table A-5 2 - Source Code complies with software architecture - Unfulfilled
 - ✘ Table A-5 3 - Source Code is verifiable - Unfulfilled
 - ✘ Table A-5 4 - Source Code conforms to standards - Unfulfilled
 - ✘ Table A-5 5 - Source Code is traceable to low-level requirements - Unfulfilled
 - ✘ Table A-5 6 - Source Code is accurate and consistent - Unfulfilled
 - ✘ Table A-5 7 - Output of software integration process is complete and correct - Unfulfilled
 - ✘ Table A-5 8 - Parameter Data Item File is correct and complete - Unfulfilled
 - ✘ Table A-5 9 - Verification of Parameter Data Item File is achieved - Unfulfilled
- ▷ ✘ Table A-6 - Testing of Outputs of Integration Process - Unfulfilled
- ▷ ✘ Table A-7 - Verification of Verification Process Results - Unfulfilled
- ▷ ✘ Table A-8 - Software Configuration Management Process - Unfulfilled
- ▷ ✘ Table A-9 - Software Quality Assurance Process - Unfulfilled
- ▷ ✘ Table A-10 - Certification Liaison Process - Unfulfilled

Objectives Tracking

- Input assets and output artifacts can come from across the entire development lifecycle and must be mapped to objectives
- Evidence should be accumulated as it is produced, so inconsistencies or gaps may be easily identified
- Completed objectives tracking provides the audit evidence of standards adherence

-  Table A-5 3 - Source Code is verifiable - Unfulfilled
 -  Output: Software Verification Results (Artifact)
-  Table A-5 4 - Source Code conforms to standards - Fulfilled - 2 artifacts
 -  Design and coding guidelines document fulfilled by 1 item
 -  Misra-c_2004_coding_guidelines.pdf
 -  Software Verification Results fulfilled by 1 item
 -  Coding Standard Report.html

Traceability Reports

TBmanager Traceability Summary

File Edit View History Bookmarks Tools Help

file:///L:/LDRA_Demos/Cashregister_5.0/TBreq/TBmanager/report_storage/Traceability Su

LLR_01000: generateTicket

- Verification Status: Verified
- Type: Subordinated
- Body: generateTicket shall produce a list of all scanned items, their price and the total cost of the items.
- Mapped Procedures
 - void generateTicket(); - L:\LDRA_Demos\Cashregister_5.0\Sourcery_CodeBench_ARM_Cashregister\Src\Cashregister.c

LLR_01100: Start Cashregister

- Verification Status: Not Verified
- Type: Subordinated
- Body: Cashregister_start shall ensure that the Cashregister application is in the active state
- Mapped Procedures
 - void Cashregister_start(); - L:\LDRA_Demos\Cashregister_5.0\Sourcery_CodeBench_ARM_Cashregister\Src\Cashregister.c

LLR_01125: Start Cashregister

- Verification Status: Not Verified
- Type: Subordinated
- Body: Cashregister_start shall be ignored if the session is already active
- Mapped Procedures
 - void Cashregister_start(); - L:\LDRA_Demos\Cashregister_5.0\Sourcery_CodeBench_ARM_Cashregister\Src\Cashregister.c

LLR_01150: Start Cashregister

- Verification Status: Not Verified
- Type: Subordinated
- Body: Cashregister_start shall ensure that the barcode generation algorithm seed is returned to "zero"
- Mapped Procedures
 - void Cashregister_start(); - L:\LDRA_Demos\Cashregister_5.0\Sourcery_CodeBench_ARM_Cashregister\Src\Cashregister.c

LLR_01200: End Session

Fulfilled Objectives

Table A-5 4 - Source Code conforms to standards

- Status: Fulfilled
- Body:
- Standard: DO-178C - Populated
- Standard Level: A - S, B - S, C - S, D -
- References: 6.3.4.d
- Identifier: Table A-5 4
- Keywords:
- Grouping:
- Document:

Input Assets

- Placeholder - Design and coding guidelines
 - Document - Acme_coding_standards

Output Artifacts

- Placeholder - Code Review Report Artifacts
 - Document - Sourcery_CodeBench

Unfulfilled Objectives

Table A-1 - Software Planning Process

- Status: Unfulfilled
- Body:
- Standard: DO-178C - Populated
- Standard Level:
- References:

Aspect #2: Coding Standards



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

DO-178C Objectives

- ▷ ✘ Table A-1 - Software Planning Process - Unfulfilled
- ▷ ✘ Table A-2 - Software Development Process - Unfulfilled - 1 asset
- ▷ ✘ Table A-3 - Verification of Outputs of Software Requirements Process - Unfulfilled
- ▷ ✘ Table A-4 - Verification of Outputs of Software Design Process - Unfulfilled
- ▷ ✘ Table A-4 1 - Low Level requirements comply with high level requirements - Unfulfilled
- ▷ ✘ Table A-5 - Verification of Outputs of Software Coding and Process - Unfulfilled
 - ✿ ✘ Table A-5 1 - Source Code complies with low-level requirements - Unfulfilled
 - ✿ ✘ Table A-5 2 - Source Code complies with software architecture - Unfulfilled
 - ✿ ✘ Table A-5 3 - Source Code is verifiable - Unfulfilled
 - ✿ ✘ Table A-5 4 - Source Code conforms to standards - Unfulfilled**
 - ✿ ✘ Table A-5 5 - Source Code is traceable to low-level requirements - Unfulfilled
 - ✿ ✘ Table A-5 6 - Source Code is accurate and consistent - Unfulfilled
 - ✿ ✘ Table A-5 7 - Output of software integration process is complete and correct - Unfulfilled
 - ✿ ✘ Table A-5 8 - Parameter Data Item File is correct and complete - Unfulfilled
 - ✿ ✘ Table A-5 9 - Verification of Parameter Data Item File is achieved - Unfulfilled
- ✿ ✘ Table A-5 4 - Source Code conforms to standards - Unfulfilled
- ✿ ✘ Table A-5 5 - Source Code is traceable to low-level requirements - Unfulfilled
- ✿ ✘ Table A-5 6 - Source Code is accurate and consistent - Unfulfilled
- ✿ ✘ Table A-5 7 - Output of software integration process is complete and correct
- ✿ ✘ Table A-5 8 - Parameter Data Item File is correct and complete - Unfulfilled
- ✿ ✘ Table A-5 9 - Verification of Parameter Data Item File is achieved - Unfulfilled
- ▷ ✘ Table A-6 - Testing of Outputs of Integration Process - Unfulfilled
- ▷ ✘ Table A-7 - Verification of Verification Process Results - Unfulfilled
- ▷ ✘ Table A-8 - Software Configuration Management Process - Unfulfilled
- ▷ ✘ Table A-9 - Software Quality Assurance Process - Unfulfilled
- ▷ ✘ Table A-10 - Certification Liaison Process - Unfulfilled

Standard	DO-178C
Standard Level	A - S, B - S, C - S, D -
References	6.3.4.d

Coding Standards

- Roughly 80% of software defects when using the C or C++ language, are attributable to the incorrect usage of 20% of the language constructs
- If the usage of the language can be restricted to avoid this subset that is known to be problematic, then the quality of the ensuing software is going to greatly increase

If you don't want defects in your code,
then don't put them there!

Why use a Coding Standard?

- In many languages such as C, there are various constructs such as “goto” that when used incorrectly often lead to defects
- Other constructs such as operator precedence are not always fully understood by programmers
- It is easy to write code that is difficult to read
- Can help to prevent developers from trying to develop “*clever*” code
- The use of a Coding Standard can also help ensure that the code is written in a particular style; for example to ensure that the “tab” character is not used

Coding Standards

- DO-178C requires that developed code be “Uniformly designed and implemented”
- DO-178C does not mandate any specific coding standard and as such any Coding Standard such as *MISRA C:2012*, *MISRA C++:2008*, *JSF++ AV*, ... can be used
- Alternatively companies often create their own coding standard generally based on standards such as the ones stated above

Configure Standard

- Common sense rules
 - Don't mix signed and unsigned types
- Reduced language subset
 - Ensure “goto” or “malloc” are not used
- Style guidelines
 - Ensure the “tab” character is not used
- Naming conventions
 - Ensure all public functions start with <filename>_
- Quality & complexity metrics
 - Ensure all functions have a low complexity

Configure Standard: Example

LDRA Report File Editor - C:\LDRA_Toolsuite\c\creport.dat*

File Edit View Help

Static	Complexity	Data Flow	Cross Ref	Info Flow	Qual Report	Qualsys	LCSAJ	Hungarian Notation	User Def	Section			
Rule Number	Default Strength	Description			MISRA-AC	MISRA	MISRA-C 2004	MISRA C:2012	NETRINO	Standard	ACME Standard		
184	C	LDRA Report File Editor - C:\LDRA_Toolsuite\c\creport.dat*											
185	O	File Edit View Help											
186	O	Static Complexity Data Flow Cross Ref Info Flow Qual Report Qualsys LCSAJ Hungarian Notation User Def Section											
187	O	Rule Number	Default Strength	Description			MISRA-AC	MISRA	MISRA-C 2004	MISRA C:2012	NETRINO	Standard	ACME Standard
188	C	1	M	Cyclomatic complexity greater than ***.			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
189	C	2	O	Procedure is not reducible in terms of intervals.			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
190		LDRA Report File Editor - C:\LDRA_Toolsuite\c\creport.dat*											
191		File Edit View Help											
Report	Static	Complexity	Data Flow	Cross Ref	Info Flow	Qual Report	Qualsys	LCSAJ	Hungarian Notation	User Def	Section 3	Section	
	Rule Number	Default Strength	Description			MISRA-AC	MISRA	MISRA-C 2004	MISRA C:2012	NETRINO	Standard	ACME Standard	
	1	M	Global Variable does not conform to style g_<name>.			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
	3	M	Enum Element does not conform to style e_<name>.			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>					
	6	M	Pointer does not conform to style p_<name>.			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
	7	M	Enum Name does not conform to style E<name>.			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>					
	8	M	Global Func Name does not conform to style <file>_<name>.			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
	9	M	Global Var Name does not conform to style <file>_<name>.			<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>					

Report File (PC): Section 1 read (25 Models)(27 Modifiers)...Section 2 read (825 Rules)...Section 3 read...Section 4 read...Penalty File (PC): read

Checking Compliance

- Check the code manually
 - Needs to be done on the “undecidable” rules
 - But don’t really want to do it on all the code!
- Use a heavyweight tool
 - Slow (*Deep analysis*)
 - Detects the easy and **hard** to find defects
 - Tends to be “Pessimistic” – **False Positives**



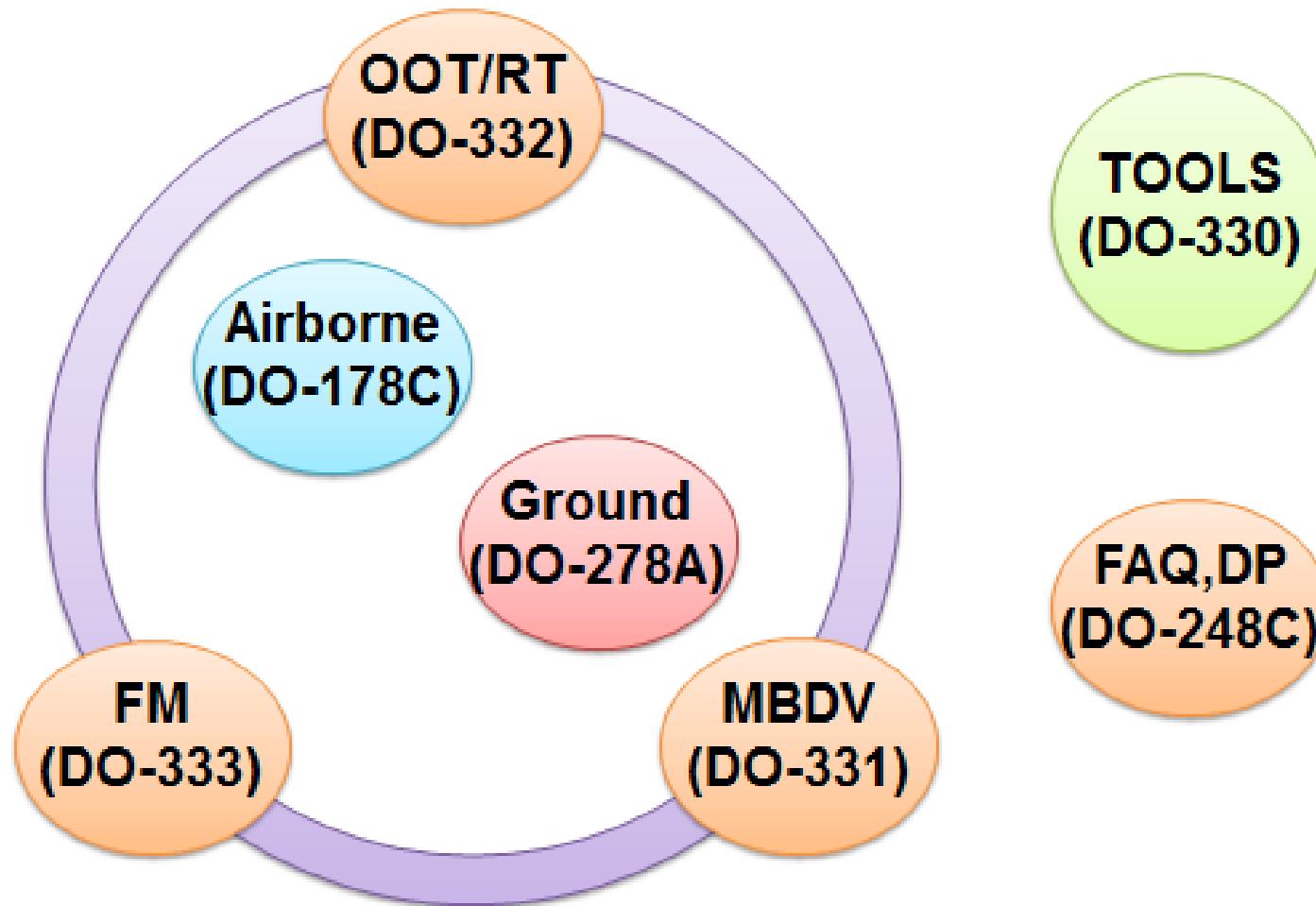
Aspect #3: Model Based Development and Verification



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

DO-178C & Supplementary Documents

LDRA



DO-331: "Model-Based Development and Verification Supplement to DO-178C and DO-278"

- MB.6.7: Model Coverage Analysis for Design Models
 - “Model coverage analysis is different than structural coverage analysis and therefore model coverage analysis **does not eliminate the need to achieve the objectives of structural analysis** per DO-178C section 6.4.4.2”
- MB.6.8.2: Model Simulation for Verification of the Executable Object Code
 - “The software testing objective, Executable Object Code is compatible with the target computer, cannot be satisfied by means of simulation (see DO-178C 6.4e). In particular tests dedicated to hardware/software integration verification **always need to be performed in the target computer environment** as defined in DO-178C section 6.4.3.a”

Accruing Coverage Credit

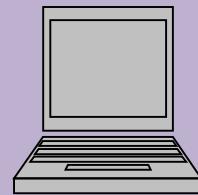
Simulation

Test Cases



Host Computer

Test Cases



Additional Test Cases

Target H/W

Test Cases



Additional Test Cases

Model Coverage

+ 100 %

C Level Coverage

< 100 %

C Level Coverage

+ 100 %

C Level Coverage

+ 100 %

ASM Level (Level A)

+ 100 %

Independent Back to Back Testing

LDRA - LDRA Testbed Dynamic Overview Report - Code Composer Studio

File Edit Navigate Search Project LDRA Tools Run Window Help

C/C++ P... Project E... Navigator

TMS570_Launchpad_BSP TMS570_Launchpad_Dice TMS570_Launchpad_QuickBrownFox TMS570_Launchpad_Simulink_CruiseControl [Active]

- Binaries
- Includes
- Debug
- LDRA
- targetConfigs
- main.c
- model_reference_types.h
- rtwtypes.h
- sldvdemo_cruise_control2_harness_private.h
- sldvdemo_cruise_control2_harness_types.h
- sldvdemo_cruise_control2_harness.c
- sldvdemo_cruise_control2_harness.h
- sldvdemo_cruise_control2_private.h
- sldvdemo_cruise_control2_types.h
- sldvdemo_cruise_control2.c
- sldvdemo_cruise_control2.h
- sys_link.cmd
- sysppvar.dat
- target.cxml
- TMS570_Launchpad_BSP.lib

LDRA Testbed Dynamic Overview Report file:///C:/LDRA_Workarea/TMS570_Launchpad_Simulink_CruiseControl_tbwrkfls/TMS570_Launchpad_Simulink_CruiseControl.ms3.htm

Combined Coverage Results

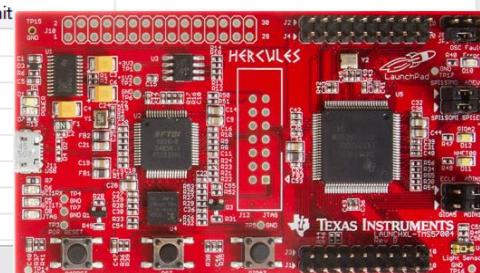
Procedure	Statement	Branch	MC/DC
TMS570_Launchpad_Simulink_CruiseControl [Set]	78	67	75
sldvdemo_cruise_control2_harness_step	+62	+47	[No BCs]
sldvdemo_cruise_control2_harness_initialize	+100	[No Branches]	[No BCs]
sldvdemo_cruise_control2_harness_terminate	+100	[No Branches]	[No BCs]
sldvdemo_cruise_control2	+91	+85	+75
sldvdemo_cruise_cont_initialize	+100	[No Branches]	[No BCs]
main	+100	+100	[No BCs]

[Top of Report | Contents]

LDRA Tool Suite Output LDRA Code Violations LDRA Code Coverage Problems Console Progress

Double click a Function to Open in Editor.

	Percentage	Success Limit
L:\LDRA_Demos\TICCS60\Workspace\TMS570_Launchpad_\ Combined Coverage Run	91	100
Statement Coverage	91	100
Branch/Decision Coverage	85	100
Modified Condition / Decision Coverage	75	100
sldvdemo_cruise_control2		
sldvdemo_cruise_cont_initialize		
L:\LDRA_Demos\TICCS60\Workspace\TMS570_Launchpad_\		



Done

Aspect #4: Subclass Verification for Local Type Consistency

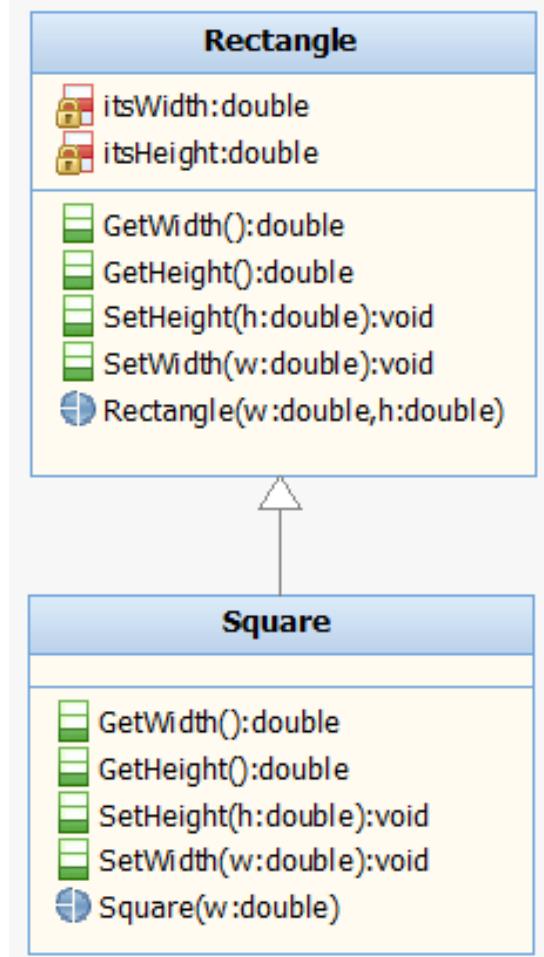


Delivering Software Quality and Security through
Test, Analysis & Requirements Traceability

Design Verification Testing Scenarios

LDRA

- Assume a subclass *Square* is being added to an existing parent C++ class *Rectangle*
- Override parent class virtual methods with new implementations in the subclasses
- In compliance with the DO-332, ensure that the low level requirements of the parent class are met by the subclasses, thereby verifying local type consistency



DO-332: "Object-Oriented Technology and Related Techniques Supplement to DO-178C and DO-278A"

Parent & Subclass Implementation

- virtual void Rectangle::SetWidth (double w) {
 itsWidth=w;
}

Post condition: itsHeight does not change

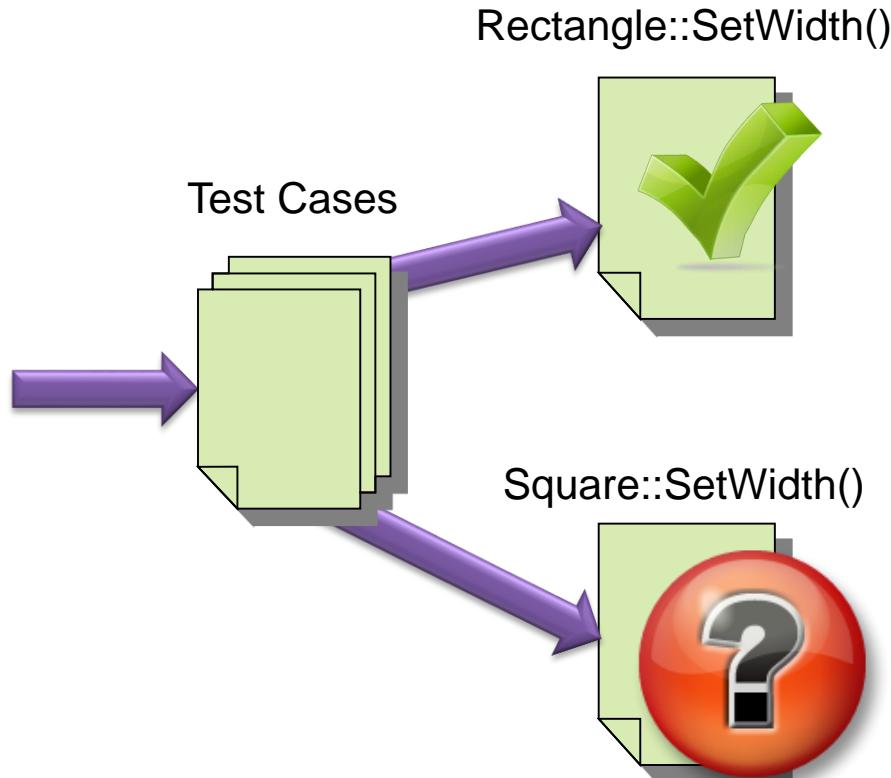
- void Square::SetWidth (double w) {
 Rectangle::SetWidth(w);
 Rectangle::SetHeight(w);
}

Post condition: itsHeight set to w

Traceability

- Low level requirements map to the parent class
- Subclasses must be proven to be type consistent by meeting the same verification criteria for redefined functions

Requirement Name:	Set Width
Requirement Number:	LL_3_u001
Body:	
	The software shall correctly set the width of the shape.
Post Condition:	
	<pre>assert((itsWidth == w) && (itsHeight == old.itsHeight));</pre>
Invariant:	
	<pre>assert(itsWidth > 0);</pre>



Parent Class (Rectangle) Results

LDRA

- Use negative testing to verify that `SetWidth()` does not modify the member variable `itsHeight`

Test Case	Regression P / F	Procedure
▲ 1	PASS	<code>Rectangle::Rectangle</code>
▲ 2 Negative Testing		<code>Rectangle::SetWidth</code>

Variable	Type	Initial Value	Final Value	Expected	Status
itsHeight	Double	1.000000e+000	1.000000e+000	No Change	PASS
itsWidth	Double	2.000000e+000	4.000000e+000	Change	PASS

Subclass (Square) Results

- The same tests, run against the subclass, demonstrate type inconsistency

Test Case	Regression P / F	Procedure
 1		C:\Square.tcf
 2		C:\Square SetWidth.tcf

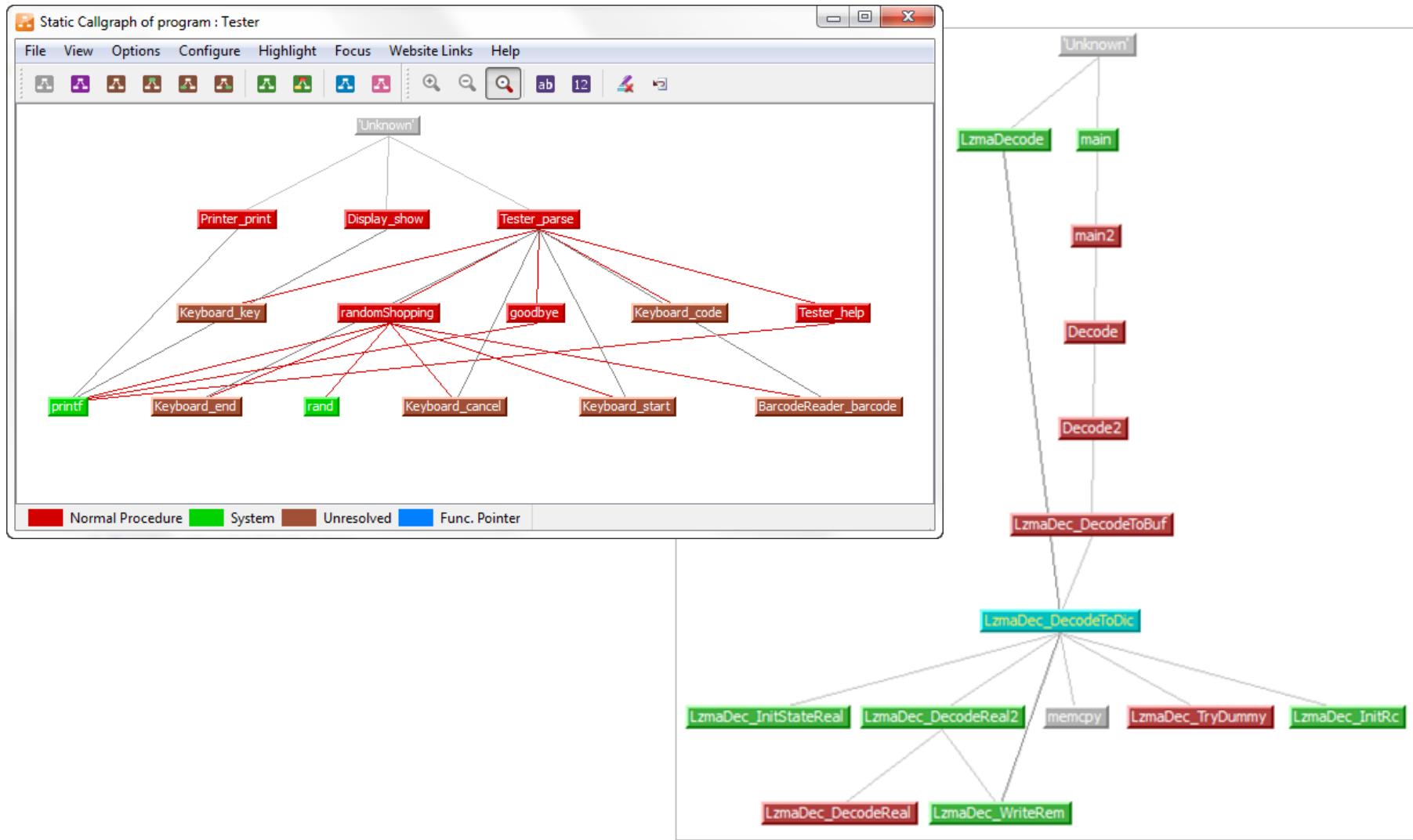
Variable	Type	Initial Value	Final Value	Expected	Status
itsHeight	Double	1.000000e+000	4.000000e+00	No Change	FAIL
itsWidth	Double	2.000000e+000	4.000000e+000	Change	PASS

Aspect #5: Data and Control Coupling Coverage



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

Control Flow Analysis



Data Flow Analysis

- How is data used?

```
169  #pragma vector=ADC12_VECTOR
170  __interrupt void ADC12ISR (void)
171 {
172     switch(__even_in_range(ADC12IV, 34))
173     {
174         case 0: break;
175         case 2: break;
176         case 4: break;
177         case 6:
178             adc12_result = ADC12MEM0;
179             adc12_data_ready = 1;
180             _BIC_SR_IRQ(LPM3_bits);
181             break;
182         case 8: break;
```

adc12_result (Pri)	u16	adc12.c	<Global scope>	G	E	60					
		adc12.h/adc12.c		G	E	62					
			ADC12ISR	G	D	178					
			adc12_single_conversion	G	R	157					

Data and Control Coupling Coverage

LDRA

- DO-178C section 6.4.4.2 c states:
“Analysis to confirm that the requirements-based testing has **exercised** the data and control coupling between code components”
- Control coupling **coverage** is ensuring that every invocation of a function has been exercised
- Data coupling **coverage** is ensuring that we have exercised every access to the data



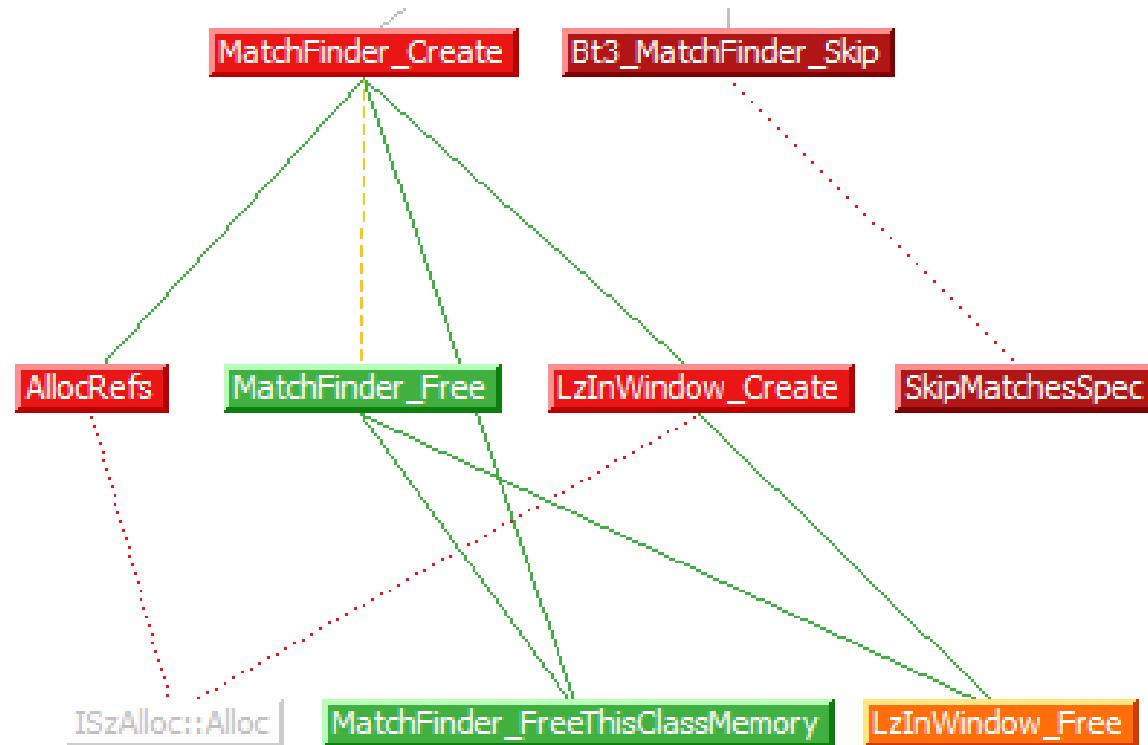
Data Coupling and Control Coupling

Technical Note

Author: Professor M. A. Hennell

Control Coupling Coverage

- ▷ ✖ Table A-7 5 - Test coverage of software structure (modified condition/decision coverage) is achieved - Unfulfilled
- ▷ ✖ Table A-7 6 - Test coverage of software structure (decision coverage) is achieved - Unfulfilled
- ▷ ✖ Table A-7 7 - Test coverage of software structure (statement coverage) is achieved - Unfulfilled
- ▷ ✖ Table A-7 8 - Test coverage of software structure (data coupling and control coupling) is achieved - Fulfilled - 2 assets
 - ✖ Software Verification Results fulfilled by 2 items
 -  Callgraph - Pass/Fail Coverage
 -  DataCouplingReport.html
- ▷ ✖ Table A-7 9 - Verification of additional code that can not be traced to Source Code, is achieved - Unfulfilled



Data Coupling Coverage

- ▷ X Table A-7 5 - Test coverage of software structure (modified condition/decision coverage) is achieved - Unfulfilled
- ▷ X Table A-7 6 - Test coverage of software structure (decision coverage) is achieved - Unfulfilled
- ▷ X Table A-7 7 - Test coverage of software structure (statement coverage) is achieved - Unfulfilled
- X Table A-7 8 - Test coverage of software structure (data coupling and control coupling) is achieved - Fulfilled - 2 assets
 - X Software Verification Results fulfilled by 2 items
 - A Callgraph - Pass/Fail Coverage
 - D DataCouplingReport.html
- ▷ X Table A-7 9 - Verification of additional code that can not be traced to Source Code, is achieved - Unfulfilled

	Call Depth / Parameter Name	data access not exercised!						
Variable Name	Alias	File	Procedure	Type Code	Attribute Code	Used on lines...		
OutBound.itsIException		Calculator.cpp	Calculator::calc_C::setItsIException	P	D			
	1 itsIException	Calculator.cpp	Calculator::calc_C::OutBound_C::setItsIException	P	D	118		
p_		Calculator.cpp	Calculator::calc_C::calc_C	P	D	128		
		Tester.cpp	Tester::calc_C::calc_C	P	D	130		
arg		TesterBuilder.cpp	TesterBuilder::cancelTimeout	P	E	58		
				P	R	60 *****		
arg1		Calculator.cpp	Calculator::add	P	E	177		
				P	R	180		
arg1			Calculator::calc_C::InBound_C::add	P	E	26		
				P	R	29		
	1 arg1		Calculator::add	P	E	177		
				P	R	180		
arg1			Calculator::calc_C::InBound_C::divide	P	E	34		
				P	R	37		

Aspect #6: Object Code Verification



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

- In 6.4.4.2, b, there is a Level A Objective that is “*hidden*” in the Structural Coverage Analysis guidance

6.4.4.2 Structural Coverage Analysis

The objective of this analysis is to determine which code structure was not exercised by the requirements-based test procedures. The requirements-based test cases may not have completely exercised the code structure, so structural coverage analysis is performed and additional verification produced to provide structural coverage. Guidance includes:

- a. The analysis should confirm the degree of structural coverage appropriate to the software level.
- b. The structural coverage analysis may be performed on the Source Code, unless the software level is A and the compiler generates object code that is not directly traceable to Source Code statements. Then, additional verification should be performed on the object code to establish the correctness of such generated code sequences. A compiler-generated array-bound check in the object code is an example of object code that is not directly traceable to the Source Code.
- c. The analysis should confirm the data coupling and control coupling between the code components.

DO-178B 6.4.4.2.b

- 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

In DO-178C, EOC Verification Is Official!

- In DO-178C, Executable Object Code verification is now added to Table A-7, the Objectives table for testing
 - “Verification of additional code, that cannot be traced to Source Code, is achieved.”
 - The Activity associated with this new Objective is in the same paragraph, 6.4.4.2 b, as in DO-178B

Table A-7 - Verification of Verification Process Results - Unfulfilled

- Table A-7 1 - Test procedures are correct - Unfulfilled
- Table A-7 2 - Test results are correct and discrepancies explained - Unfulfilled
- Table A-7 3 - Test coverage of high-level requirements is achieved - Unfulfilled
- Table A-7 4 - Test coverage of low-level requirements is achieved - Unfulfilled
- Table A-7 5 - Test coverage of software structure (modified condition/decision coverage) is achieved - Unfulfilled
- Table A-7 6 - Test coverage of software structure (decision coverage) is achieved - Unfulfilled
- Table A-7 7 - Test coverage of software structure (statement coverage) is achieved - Unfulfilled
- Table A-7 8 - Test coverage of software structure (data coupling and control coupling) is achieved - Unfulfilled
- Table A-7 9 - Verification of additional code that can not be traced to Source Code, is achieved - Unfulfilled

Standard	DO-178C
Standard Level	A - SI, B - , C - , D -
References	6.4.4.2.b

Object Code Verification

- 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 optimize 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

Sample “C” Code

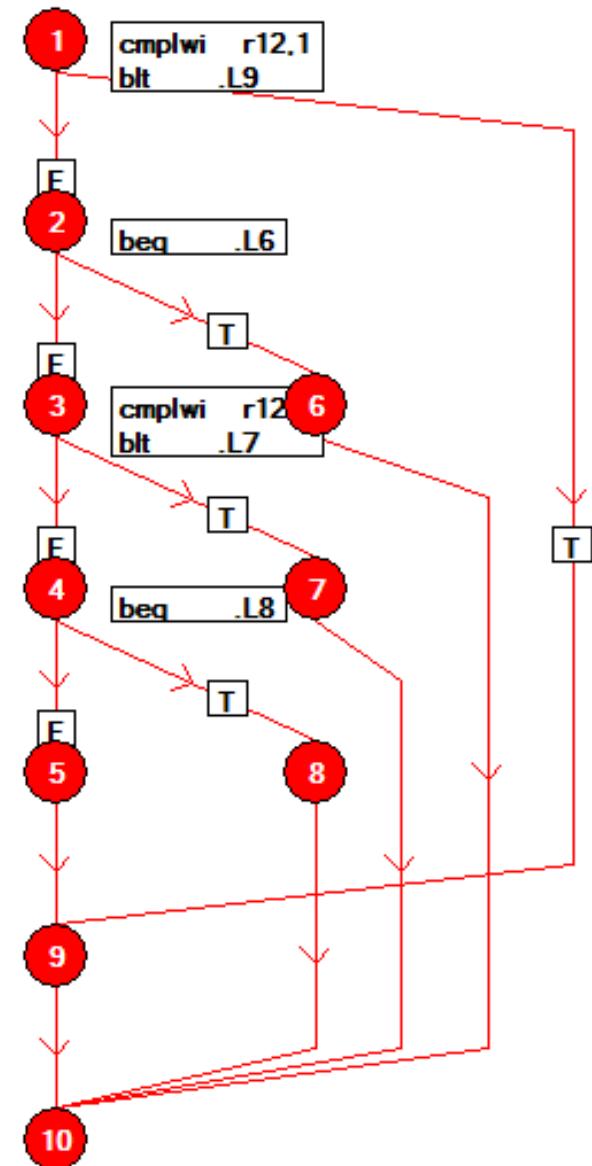
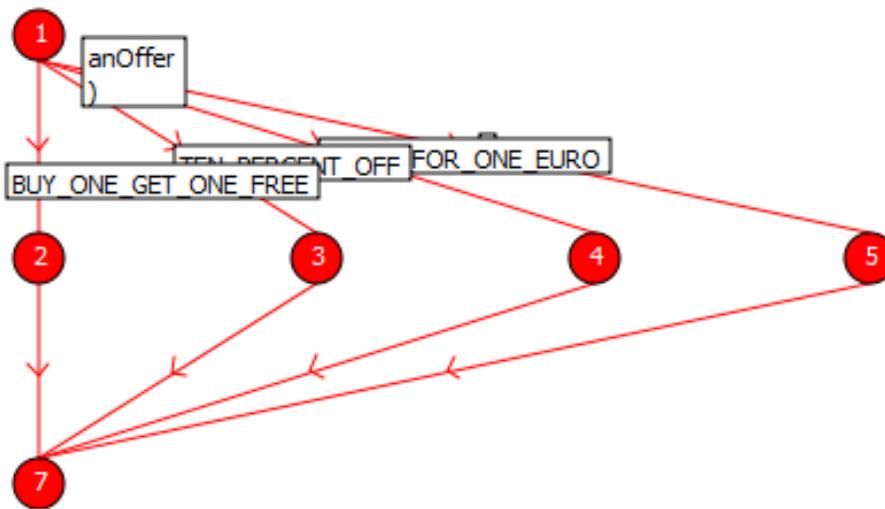
- Consider the following simple C code:

```
#include "misrac_types.h"
#include "specialoffer.h"

/*
 * 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);
            break;
        case TEN_PERCENT_OFF:
            price = (aUnitPrice * aQuantity * 9U) / 10U;
            break;
        case THREE_FOR_ONE_EURO:
            price = ((aQuantity / 3U) * 100U) + ((aQuantity % 3U) * aUnitPrice);
            break;
            /* no offer */
        default:
            price = aUnitPrice * aQuantity;
            break;
    }
    return price;
}
```

Assembler Code Structure

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



C Code Structural Coverage

- In order to get 100% statement and 100% decision coverage of the “C” code, we need to create four test cases:

Test Case	Regression P / F	Procedure	Value	Name
1	PASS	SpecialOffer_getPrice	I 1	aQuantity
2	PASS	SpecialOffer_getPrice	I 10	aUnitPrice
3	PASS	SpecialOffer_getPrice	I NO_OFFER	anOffer
4				

Test Case	Regression P / F	Procedure	Value	Name
1	PASS	SpecialOffer_getPrice	I 2	aQuantity
2	PASS	SpecialOffer_getPrice	I 10	aUnitPrice
3	PASS	SpecialOffer_getPrice	I BUY_ONE_GET_ONE_FREE	anOffer
4	PASS	SpecialOffer_getPrice	> 10	%

Test Case	Regression P / F	Procedure	Value	Name
1	PASS	SpecialOffer_getPrice	I 2	aQuantity
2	PASS	SpecialOffer_getPrice	I 50	aUnitPrice
3	PASS	SpecialOffer_getPrice	I TEN_PERCENT_OFF	anOffer
4	PASS	SpecialOffer_getPrice	O 90	%

Test Case	Regression P / F	Procedure	Value	Name
1	PASS	SpecialOffer_getPrice	I 3	aQuantity
2	PASS	SpecialOffer_getPrice	I 50	aUnitPrice
3	PASS	SpecialOffer_getPrice	I THREE_FOR_ONE_EURO	anOffer
4	PASS	SpecialOffer_getPrice	O 100	%

C Code Coverage

LDRA



LINE NUMBER REF. (SOURCE)	STATEMENT	PREVIOUS RUNS	CURRENT RUN	COMBINED
150 (16) LDRA_uint32_t		-	-	-
151 SpecialOffer_getPrice (0	4	4
152 const LDRA_uint32_t aQuantity ,		-	-	-
153 (17) const LDRA_uint32_t aUnitPrice ,		-	-	-
154 const tSpecialOffer anOffer)		-	-	-
155 (18) {		-	-	-
156 (19) LDRA_uint32_t		-	-	-
157 price ;		-	-	-
158 (20) switch (0	4	4
159 anOffer		0	4	4
160)		0	4	4
161 (21) {		0	4	4
162 (22) case BUY_ONE_GET_ONE_FREE :		0	1	1
163 (23) price = aUnitPrice * (0	1	1
164 (aQuantity + 1U) >> 1U) ;		0	1	1
165 (24) break ;		0	1	1
166 (25) case TEN_PERCENT_OFF :		0	1	1
167 (26) /* LDRA_INSPECTED 96 S		-	-	-
168 */		-	-	-
169 (27) price = (aUnitPrice * aQuantity * 9U) /		0	1	1
170 10U ;		0	1	1
171 (28) break ;		0	1	1
172 (29) case THREE_FOR_ONE_EURO :		0	1	1
173 (30) /* LDRA_INSPECTED 96 S		-	-	-
174 */		-	-	-
175 (31) price = (0	1	1
176 (aQuantity /		0	1	1
177 3U) * 100U) + (0	1	1
178 (aQuantity % 3U) * aUnitPrice) ;		-	-	-
179 (32) break ;		-	-	-
180 (33) /* no offer */		-	-	-
181 (34) default :		-	-	-
182 (35) price = aUnitPrice * aQuantity ;		-	-	-
183 (36) break ;		-	-	-
184 (37) }		-	-	-
185 (38) return		-	-	-
186 price ;		-	-	-
187 (39) }		-	-	-

LDRA Testbed ® Dynamic Coverage Analysis Report

File : C:\LDRA_Demos\GHS_MULTI4_Workspace\GHS_MULTI4_C_CashRegister\src\specialoffer.c

Overall Result (For File): Coverage Metrics required to achieve DO-178B Level A Attained

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

Assembler Structural Coverage

- 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\specialoffer.s

Overall Coverage Result (For File): Fail

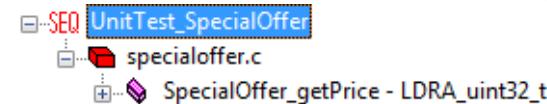
Statement (TER1) = 98 % (Fail) Branch/Decision (TER2) = 83 % (Fail)

Assembler Code Coverage

- 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
73 (66)	addi r12,r11,0	1	1	2

100% Assembler Coverage



- With the addition of one extra test, we can get 100% coverage of the assembler code:

Test Case	Regression P / F	Procedure	Value	Name
1	PASS	SpecialOffer_getPrice	I 3	aQuantity
2	PASS	SpecialOffer_getPrice	I 50	aUnitPrice
3	PASS	SpecialOffer_getPrice	I (tSpecialOffer)(THREE_FOR_ONE_EURO+1)	anOffer
4	PASS	SpecialOffer_getPrice	O 150	%
5	PASS	SpecialOffer_getPrice		

LDRA Testbed ® Dynamic Coverage Analysis Report

File : C:\LDRA_Workarea\GHS_MULTI4_C_CashRegister_tbwrkfls\GHS_MULTI4_C_CashRegister_asmwrkfls\specialoffer.s

Overall Coverage Result (For File): Pass

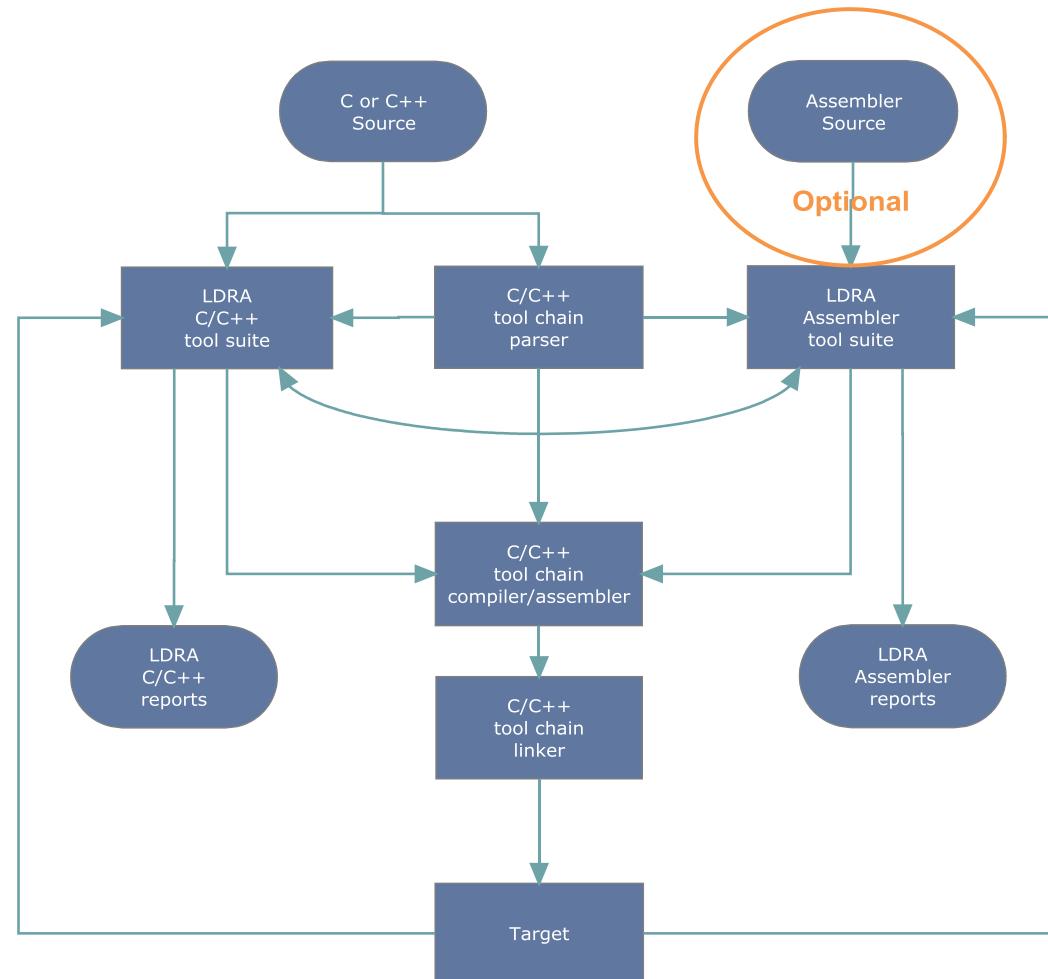
Statement (TER1) = 100 % (Pass) Branch/Decision (TER2) = 100 % (Pass)

LDRA Object-Box

LDRA

- Links the LDRA C/C++ tool suite with the appropriate LDRA assembler tool suite
- Tests can be run in:
 - Black Box
 - White Box
 - Red Box

LDRA Object-Box flow diagram

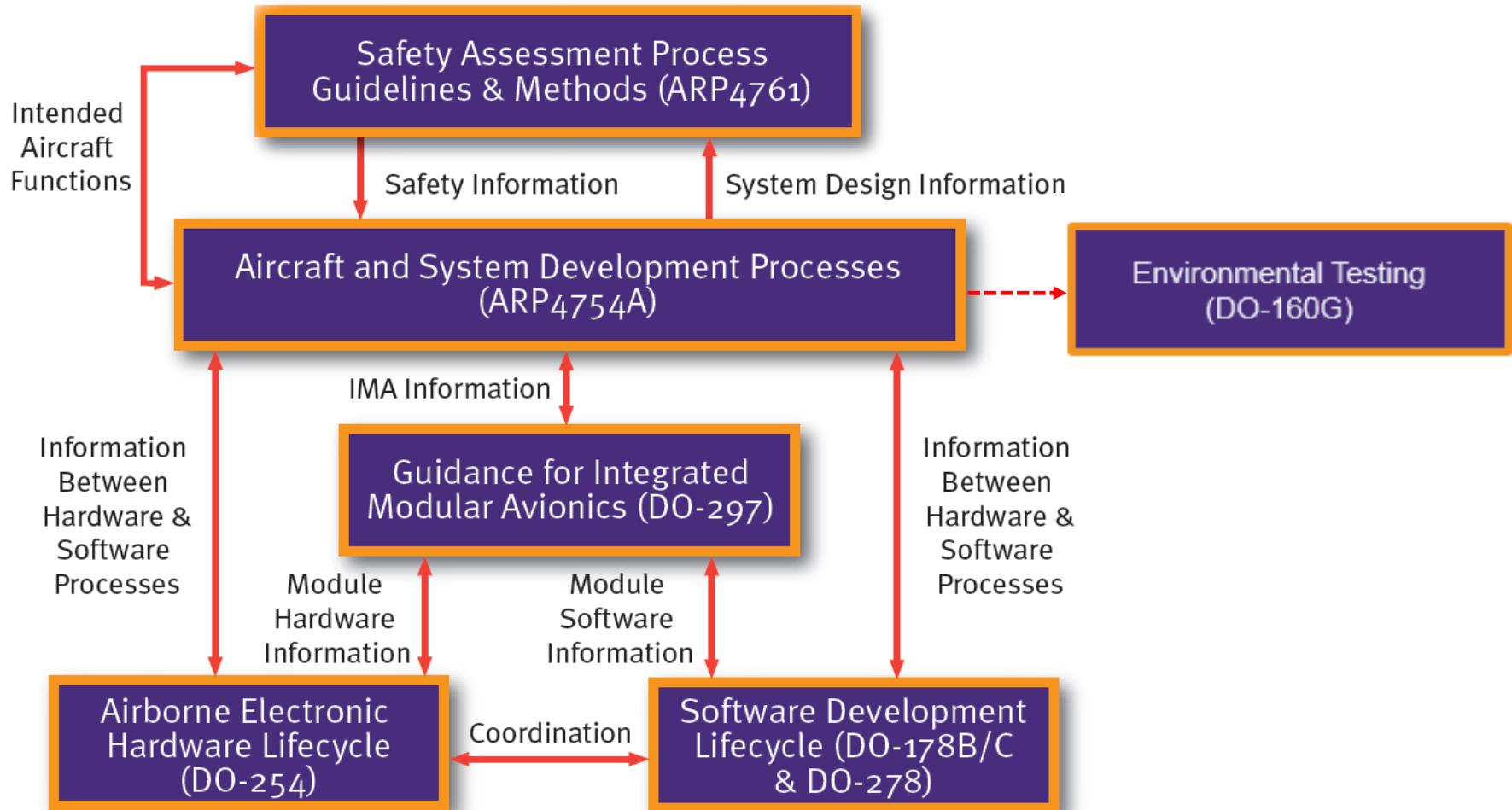


Aspect #7: Compliance Knowledgebase



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

Required Standards Coverage



Convert Standards Into Compliant Process

- But complicated and ambiguous guidance across multiple standards makes it hard!
 - In addition to DO-178C/DO-278A and DO-254, there are:
 - DO-178C/DO-278A supplements:
 - Model Based, OOT and Formal Methods
 - DO-254: Scope of standard has recently been expanded by EASA certification memo
 - DO-330 Tool Qualification
 - DO-248C— Rationale (FAQ's)
 - System & safety standards: ARP-4754A and ARP-4761
 - Each standard requires detailed interpretation

Comprehensive FAA/EASA Knowledge Base

- Process Compliance Documents (PCDs)
 - 90% complete
 - Reduces project planning costs by 50%
 - Used by FAA to train Aircraft Certification Engineers
- Checklists
 - Assurance that ***all*** project transition criteria have been met
 - Checklists ask everything you need to know about applying DO-178 and DO-254 guidance
- LCMS PCDs and checklists have been used by hundreds of successful DO-178 and DO-254 FAA and EASA applicants over the past 15 years

Process Compliance Documents (PCDs)

- PCDs for all project deliverable documents
 - ARP 4754A (Aircraft systems)
 - DO-297 (Integrated modular avionics)
 - DO-178C (Airborne software)
 - DO-278A (Ground control software)
 - DO-254 (Complex electronic hardware)
- All planning, development & verification documents
- Comprehensive, up to 90% complete
- Approved by the FAA, EASA, TCCA
- Approved by all major OEM's

Aspect #8: Compliance Management System



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

Compliance Management System

- DO-178C, DO-278A and DO-254 Lifecycle processes, not just the product, must be compliant
 - Though standards do not prescribe a particular process, your lifecycle processes must be:
 - Consistent
 - Repeatable
 - Transparent
 - You must have an infrastructure in place to support these objectives
- LDRA Compliance Management System (LCMS)

LDRA Compliance Management System

LDRA

- Everything you need to organize and prepare for the Stages of Involvement (SOI) audits and to “show compliance”
- Used by Designated Engineering Representatives (DERs) and subject matter experts worldwide
- LCMS tools:
 - Security management system
 - Project management
 - Document review management
 - Reviews & analysis management
 - Problem reporting system
 - Archive management system

Security Management

- Named user tied to 32-bit secure login
- Supports project, admin and guest users
- Login and usage records automatically maintained

User Fields **LCMS Password** **Login Records**

Web Security Management System									
Account: seema				Actions					
Registered User List				Add User	Modify Passwords	Set Max Programs	View My Profile	Home Page	Logout
First Name	Last Name	Company	User Name	Password	Last Login	Admin	Team	Guest	Edit
Todd	White	Qualtech Consulting	sqa	*****	20/03/2011 7:51:39 PM	✓	✓	✗	✗
ICMS	ICMS	TBD	admin	*****	26/11/2012 4:55:47 PM	✓	✓	✗	✗
Anitha	Poojar	Affinity	pujar	*****	08/11/2012 4:35:28 PM	✓	✓	✗	✗
abc	xyz	Faac	faac	*****	09/11/2012 12:12:03 PM	✓	✓	✗	✗
XYZ	XYZ	affinity software	xyz1	*****	17/11/2012 11:33:32 AM	✗	✗	✓	✗
Seema	Seema	ABC Systems LTD	seema	*****	10/03/2013 8:04:47 PM	✓	✓	✗	✗
abc	abc	CARLSBERG	abc123	*****	06/01/2013 10:57:35 PM	✓	✓	✗	✗
todd	smith	xmp	toddsmit	*****	28/12/2012 3:47:39 PM	✓	✓	✗	✗
Vasanth	Vasanth	INFOSYS	vasanth	*****		✗	✗	✓	✗
john	smith	xmp	johnsmith	*****		✓	✓	✗	✗
john	john	xmp	johndacc	*****		✓	✓	✗	✗
test	test	ABC SYSTEMS	test123	*****	01/01/2013 6:26:21 PM	✗	✗	✓	✗
srinibas	das	xyz	srinibas	*****	02/01/2013 3:46:54 PM	✓	✓	✗	✗
raman	raman	Yahoo Systems	raman	*****	29/12/2012 6:33:57 PM	✗	✗	✓	✗
peter	dacc	xmp	peterdacc	*****		✗	✗	✓	✗
peter	parker	abc	peterparker	*****		✓	✓	✗	✗

[Next] [Last]

Number of Records: 21

Project Management

- Select project
- Select activity
 - Document review
 - Action item review
 - Problem reporting
- Confirm project
 - Aero standard
 - DAL
 - Start date

Project ID

Activity

Program Management Interface

Select Records: Program Id Get Reset

Account: seema

Registered User List

Prog Id	Program Name	Type	DAL	Rec Date	Docs	Review	PRs	Edit	Arch
P001	AIR FRANCE	DO-178	C	07/03/2013	View	View	View		
P002	Curtiss C-46	DO-178	B	01/02/2013	View	View	View		
P003	Cornell	DO-178	A	01/02/2013	View	View	View		
P004	Vought F4U	DO-178	A	01/02/2013	View	View	View		
P005	Expeditor	DO-178	A	01/02/2013	View	View	View		
P006	Jet Airways	DO-178	B	01/02/2013	View	View	View		
P007	Invader	DO-178	C	01/02/2013	View	View	View		
P008	Lightning	DO-178	A	01/02/2013	View	View	View		
P009	Nighthawk	DO-178	A	01/02/2013	View	View	View		
P010	Mitchell	DO-178	A	01/02/2013	View	View	View		
P011	Aardvark	DO-178	A	01/02/2013	View	View	View		
P012	Bronco	DO-178	A	01/02/2013	View	View	View		
P013	Delta Dagger	DO-178	B	01/02/2013	View	View	View		
P014	Flying Boxcar	DO-178	A	01/02/2013	View	View	View		
P015	PROG04NEW1	DO-178	A	05/02/2013	View	View	View		
P016	PROG04NEW2	DO-178	B	01/03/2013	View	View	View		

[Next] [Last]

Number of Records: 58

Document Review Management

LDRA

- Review document
 - View / modify checklist
 - Comment on document
 - Track comments
 - Document approval status
 - Reviews performed by individuals
 - SCM category

Document ID	Checklist	Comment/Status									
Document Review Management System											
Select Records: Standards Document Title Get Reset											
Account: seema	Add Doc Manage Doc Master List Manage Cls All Com Open Com MY Com Archives Project List Home	Logout									
List of Requirements Lifecycle Data with Checklists and Comments											
Program: P003 Cornell (DO-178) A	Click on Item Number to get into documents										
Item Num	Doc Title	SCM	Check List	Pass	Comment	Open	Closed	Rel Date	Rel	Edit	Del
800-SYS-01	System Requirements Document	CC1	View Modify			001	000	05/03/2013	✓	✓	✗
800-PSAC-01	Plan for Software Aspects of Certification	CC1	View Modify	—		002	000	29/01/2013	✓	✓	✗
800-SDP-01	Software Development Plan	CC1	View Modify			001	000	31/12/2099	✓	✓	✗
800-SVP-01	Software Verification Plan	CC1	View Modify	—		000	000	11/03/2013	✓	✓	✗
800-SCMP-01	Software Configuration Management Plan	CC1	View Modify			000	000	31/12/2099	✓	✓	✗
800-SQAP-01	Software Quality Assurance Plan	CC1	View Modify	—		000	000	31/12/2099	✓	✓	✗
800-SRS-01	Software Requirements Standards	CC1	View Modify			000	000	31/12/2099	✓	✓	✗
800-SDS-01	Software Design Standards	CC1	View Modify	—		000	000	31/12/2099	—	✓	✗
800-SCS-01	Software Code Standards	CC1	View Modify			000	000	02/05/2013	✓	✓	✗
800-SRD-01	Software Requirements Document	CC1	View Modify	—		000	000	31/12/2099	✓	✓	✗
800-SDD-01	Software Design Description	CC1	View Modify			000	000	31/12/2099	✓	✓	✗
800-SVCP-01	Software Verification Cases and Procedures	CC1	View Modify	—		000	000	31/12/2099	✓	✓	✗
800-SVR-01	Software Verification Results	CC1	View Modify			000	000	04/02/2013	✓	✓	✗
800-SECI-01	Software Life Cycle Environment Configuration Index	CC2	View Modify	—		000	000	31/12/2099	—	✓	✗
800-SCI-01	Software Configuration Index	CC1	View Modify			000	000	06/02/2013	—	✓	✗
800-SAS-01	Software Accomplishment Summary	CC1	View Modify	—		001	000	31/12/2099	✓	✓	✗

Number of Records: 16

Reviews and Analysis Management

- Transition reviews
 - Planning
 - Development
 - Verification
- Peer reviews
 - Requirements
 - Design
 - Code
 - Verification

Document Review Management Interface

The screenshot shows a web-based application titled "Document Review Management Interface". At the top, there is a search bar with fields for "Select Records", "Review Title", and buttons for "Get" and "Reset". Below this, the title "Program: P001 Braking System (EN 50128) 4" and the account "Account: admin" are displayed, along with links for "Logout", "Add Review", "Manage Inputs", "Signature", "All AIs", "Open AIs", "Closed AIs", "Archives", "Projects", and "Home Page". The main area is a table with the following columns: Review Title, Review Topic, Review Date, Review Id, Signin Sheet, Checklist, Pass, AIs, Open, Closed, Folder, Edit, and Del. The table lists ten rows of review records, each with a unique ID and status indicators (Pass: green, AIs: yellow, Open: orange, Closed: grey). At the bottom of the table, a summary row displays: Reviews: 10, Action Items: 4, Open: 3, Impl: 0, Veri: 0, Closed: 1.

Review Title	Review Topic	Review Date	Review Id	Signin Sheet	Check List	Pass	AIs	Open	Closed	Folder	Edit	Del
Software Planning Process Transition Review	Transition Compliance	01/01/2099	20	Create View	View Modify		003	001				
Software Requirements Process Transition Review	Transition Compliance	01/01/2099	40	Create View	View Modify		000	000				
Software Architecture & Design Process Transition Review	Transition Compliance	01/01/2099	60	Create View	View Modify		000	000				
Software Component Design Process Transition Review	Transition Compliance	01/01/2099	80	Create View	View Modify		000	000				
Software Component Implementation Process Transition Review	Transition Compliance	01/01/2099	100	Create View	View Modify		000	000				
Software Component Testing Process Transition Review	Transition Compliance	01/01/2099	120	Create View	View Modify		000	000				
Software Integration Process Transition Review	Transition Compliance	01/01/2099	140	Create View	View Modify		000	000				
Software Application Process Transition Review	Transition Compliance	01/01/2099	160	Create View	View Modify		000	000				
Software Validation Process Transition Review	Transition Compliance	01/01/2099	180	Create View	View Modify		000	000				
Software Assessment Process Transition Review	Transition Compliance	01/01/2099	200	Create View	View Modify		000	000				

Reviews: 10 Action Items:- 4 Open: 3 Impl: 0 Veri: 0 Closed: 1

Action Item Review

- Document meetings, review issues and non-compliances
- System generated action item ID
- Review comment text entry
- Essential compliance records
- Automated attendee and sign-off lists

Action Item ID	Reviewer	Review Comment
<input type="text" value="System Generated"/>	<input type="text" value="Helene"/>	<input type="text" value="Peer Review - Test Procedures"/> Special Test Procedure List System Requirements Robustness needs to be checked. Conformity needs to be checked. Regression tests needs to be specially carried out. System test needs to be carried out.

Submit Review Record

Use this form to document meetings & non compliance related reviews.

Program: P6 P6 Jet Airways (DO-178) B Account: seema

Home Page Return to List

Required Optional

Review Id: Review Date:

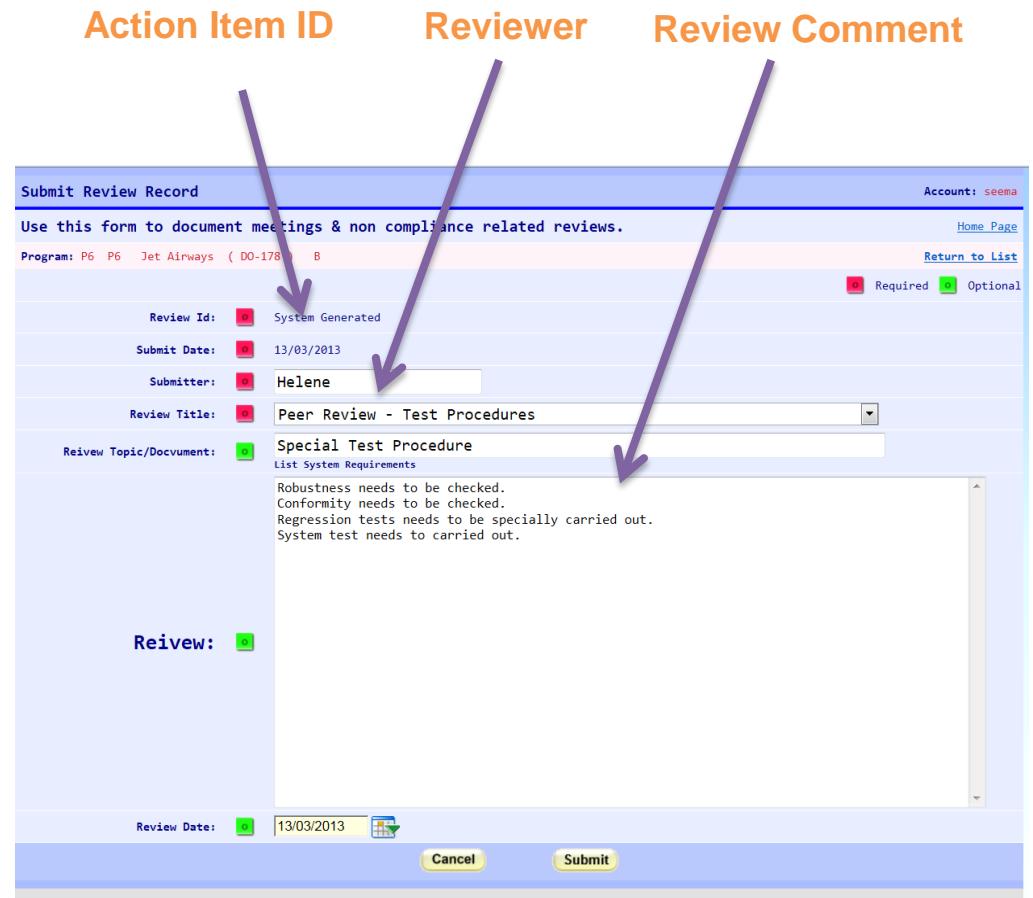
Submit Date: Submitter:

Review Title:

Review Topic/Document:
List System Requirements

Review:

Cancel Submit



Review Attendee List

- Documents all attendee and contact info
- Identifies independent reviewers
- Links to signature sheet (sign ID)
- Prints hard copy signature sheet
(Required by regulators)

Signature ID

Attendee Details

Hard Copy

Attendee Sign-In Sheet Details
Software Planning Review
Program: P001 AIR FRANCE (DO-178) C

Review Id: 0
Program: AIR FRANCE
Review Topic: Transition Compliance
Evaluation Date: 08/03/2013 12:00:00 AM
Evaluation Title: Jhon Pierr

Sign Id	User Id	Name	Signature Date	Phone	e-Mail	Print/Edit
84	sqa	Todd White	09/02/2013	941-321-5651	info@faaconsultants.com	Prt Edit
	admin	ICMS Administrator	09/02/2013	999-999-9999	TBD@TBD.com	Prt Edit
	faac	abc xyz	09/02/2013	9876662542	faac@facc.com	Prt Edit
	xyz1	XYZ abcd	09/02/2013	26743957	aff@gmail.com	Prt Edit
	abc	anz IT	09/02/2013	abc123	t8WKYfwK4WEf1fURa8ccbg==	Prt Edit
	toddsmit	todd smith	09/02/2013	222-222-2222	tsmith@gmail.com	Prt Edit
	srinibas	srinibas das	09/02/2013	9898989989	sri@gmail.com	Prt Edit
	catharine	Catharine Berrizuga	09/02/2013	3846678932	cathb@bull.net	Prt Edit
86	shima	Shimadri Rao	09/02/2013	222-222-2222	shima@bull.net	Prt Edit
	sqa	Todd White	04/03/2013	941-321-5651	info@faaconsultants.com	Prt Edit
	sqa	Todd White	04/03/2013	941-321-5651	info@faaconsultants.com	Prt Edit
	admin	ICMS Administrator	04/03/2013	999-999-9999	TBD@TBD.com	Prt Edit
	sqa	Todd White	04/03/2013	941-321-5651	info@faaconsultants.com	Prt Edit
	admin	ICMS Administrator	04/03/2013	999-999-9999	TBD@TBD.com	Prt Edit
	faac	abc xyz	04/03/2013	9876662542	faac@facc.com	Prt Edit
	xyz1	XYZ abcd	04/03/2013	26743957	aff@gmail.com	Prt Edit
100	abc	anz IT	04/03/2013	abc123	t8WKYfwK4WEf1fURa8ccbg==	Prt Edit
	toddsmit	todd smith	04/03/2013	222-222-2222	tsmith@gmail.com	Prt Edit

Number of Records: 38

Problem Reporting Management

- Provides SC1/CC1/HC1 problem reporting
- Review comment text entry
- **Required** and **Optional** fields
- Automated email notifications
- Automated search

System Fields		Submitter Fields		Recommended	
PR No:	0	1			
Impact Area:	0	Airborne Software			
Submit Date:	0	06/02/2013			
Submitter:	0	RAJAM			
Status:	0	Open	▼		
Last Modified:	0	06/03/2013			
Title:	0	PROBLEM1			
Product / Doc:	0	PROBLEM 1 DOCUMENT			
Criticality:	0	Type 0	▼	Definitions	
Urgent/Safety Related:	0				
PR Source:	0	Internal	▼		
Department Assigned:	0	Engineering	▼		
Person Assigned:	0	Todd	▼		

Home | Return To List | Logout

Required Optional

Cancel Submit

Detailed description: This screenshot shows a 'Problem Report Detail' form. It is divided into three main sections: System Fields, Submitter Fields, and Recommended. Arrows point from the section headers to specific fields: a purple arrow points to the 'PR No:' field, another points to the 'Submitter' field, and a third points to the 'PR Source' dropdown. The 'System Fields' section contains fields like PR No, Impact Area, Submit Date, Submitter, Status, Last Modified, Title, Product / Doc, Criticality, Urgent/Safety Related, PR Source, Department Assigned, and Person Assigned. The 'Submitter Fields' section contains fields like Impact Area, Submit Date, Submitter, Status, Last Modified, Title, Product / Doc, Criticality, Urgent/Safety Related, PR Source, Department Assigned, and Person Assigned. The 'Recommended' section contains fields like PR No, Impact Area, Submit Date, Submitter, Status, Last Modified, Title, Product / Doc, Criticality, Urgent/Safety Related, PR Source, Department Assigned, and Person Assigned.

Archive Management

- Facilitates archiving of directories and files
- Also supports archive restore
- Provides “view” of archived files

Archive Data

View

Restore

ICMS Archive Viewer											
Select Records:		Program Id	Get		Res.						
User Name: seema											
Program Id	Archive Id	Program Name	Standard	Description	Arch Date	View	Restore	Download	Edit	Del	
P000	P0_20130131_03195811.8	P000	DO-178	DEVELOPement 121 001	31/01/2013 03:20						
P001	P1_20130204_11452611.8	P001	DO-178	D0999	04/02/2013 11:45						
P001	P1_20130131_03190011.8	P001	DO-178	arch001_12EV	31/01/2013 03:19						
P001	P1_20130306_04560511.8	P001	DO-178		06/03/2013 04:56						
P001	P1_20130314_03205111.8	P001	DO-178	1	14/03/2013 03:22						
P001	P1_20130314_03234211.8	P001	DO-178	b1k1	14/03/2013 03:23						
P001	P1_20130314_03402911.8	P001	DO-178	1	14/03/2013 03:44						
P001	P1_20130314_03402911.8	P001	DO-178	1	14/03/2013 03:44						
P001	P1_20130314_03492111.8	P001	DO-178	y	14/03/2013 03:49						
P001	P1_20130114_12185611.8	P001	DO-178	ARCHIVE 100	14/01/2013 12:19						
P001	P1_20130115_12114011.8	P001	DO-178	ASAP	15/01/2013 12:11						
P001	P1_20130115_03171311.8	P001	DO-178	ASAP	15/01/2013 03:17						
P001	P1_20130115_03580411.8	P001	DO-178	ASAP	15/01/2013 03:58						
P001	P1_20130123_03303411.8	P001	DO-178	Arch01_02ER	23/01/2013 03:30						
P001	P1_20130123_03310511.8	P001	DO-178	arch1 oi	23/01/2013 03:31						
P001	P1_20130123_03554811.8	P001	DO-178	block	23/01/2013 03:55						
P001	P1_20130125_11132311.8	P001	DO-178	ARCHIVE 100	25/01/2013 11:13						
P001	P1_20130201_04023011.8	P001	DO-178	CAF name for a CH-46 Sea Knight development. French word for	01/02/2013 04:03						

Number of Records: 55

Summary

- In order to meet the new DO-178C objectives and activities, we have looked at the following 8 aspects:
 1. Traceability Management
 2. Coding Standards
 3. Model Based Development and Verification
 4. Subclass Verification for Local Type Consistency
 5. Data & Control Coupling Coverage
 6. Object Code Verification
 7. Compliance Knowledgebase
 8. Compliance Management System

Questions & Answers

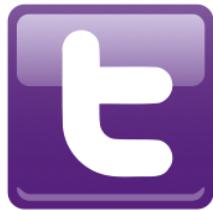


Delivering Software Quality and Security through
Test, Analysis & Requirements Traceability

For further information:

www.Idra.com

info@Idra.com



@Idra_technology



LDRA Software Technology



LDRA Limited



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