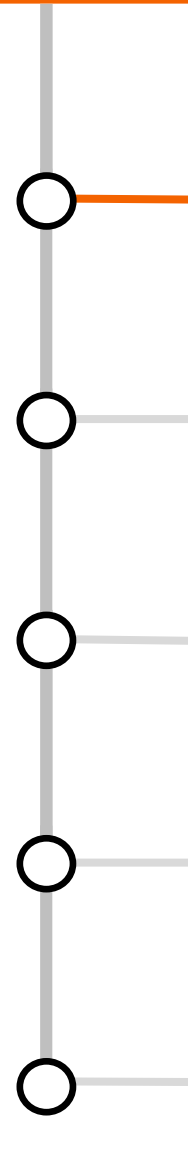




# LDRA

## LDRA Software Technology

- 
- 1 航空業界のベストプラクティス
  - 2 静的解析、コーディング規約チェックなど
  - 3 動的テストとカバレッジ解析
  - 4 要件トレーサビリティ
  - 5 まとめ



1975年設立 => 45年以上の実績とブランド  
航空業界で標準的に採用される  
スタンダード認証プロセスを支援する  
ソフトウェアテスト、管理支援ツールを提供

IEC 61508, IEC 62304, EN 50128,  
ISO 26262, IEC 60880 のツール認定取得

各種スタンダード委員会にも貢献

DO-178C, ISO 26262  
MISRA C/C++, CERT,



Liverpool



**Professor Mike Hennell**

Member of SC-205 /  
WG-71 (DO-178C) formal  
methods subgroup

Member of MISRA C committee  
and MISRA C++ committee

Member of the working group  
drafting a proposed secureC  
annex for the C language  
definition  
(SC 22 / WG14)



**Andrew Banks**

MISRA Committee Chair  
Committee Member for Second  
Edition of ISO 26262

UK Head of Delegation  
to ISO/IEC JTC1/SC7

MISRA representative to BSI  
IST/15 for Software and  
Systems Engineering



**Bill St Clair**

Member of SC-205 /  
WG-71 (DO-178C) Object Oriented  
Technology subgroup



コーディング規約  
チェック

構造化カバレッジ  
解析

データフロー  
コントロールフロー  
解析

あらゆるターゲット  
環境でサポート

要件～検証結果  
のトレーサビリティ

ツール認定

オブジェクティブ  
の追跡・管理

認証機関への  
エビデンス提示




検証作業を自動化



認証プロセスを支援



検証と認証作業を加速

- 
- 1 航空業界のベストプラクティス
  - 2 静的解析、コーディング規約チェックなど
  - 3 動的テストとカバレッジ解析
  - 4 要件トレーサビリティ
  - 5 まとめ

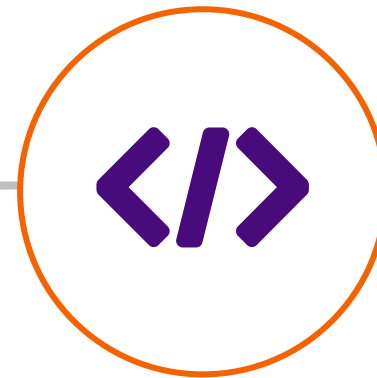
ソフトウェアの欠陥の約80%は、言語の約20%の誤った使用によって引き起こされる

言語の使用を制限することで、問題のあることが分かっているサブセットを避けることができるなら、ソフトウェアの品質は大幅に向上する



処理系定義の動作

コーディングエラー



未規定の動作 / 未定義の動作

ランタイムエラー

- C/C++によるプログラミングで不具合の原因となる根本的な言語要素：
  - 未定義の動作：言語規格で定義されない動作
  - 未規定の動作：規定されない動作
  - 処理系定義の動作：未規定の動作に対して処理系が独自に動作を実装

- アプリケーションの

信頼性や移植性が損なわれるので、

重要度に関わらずこれらの使用は避けるべき



[MISRAのようなガイドライン](#) を適用すること

Code Review : unspec.c : C - MISRA-C:2012 Model

	Number	Level of Viol.	Phase Code	Standard Code
unspec.c - C:\Users\toyama-k\test1\unspec.c				
undef				
Use of shift operator on signed type. : >> used with int: value		Required	50 S	MISRA-C:2012/AM...
noret				
Function has no return statement. : noret		Mandatory	36 S	MISRA-C:2012/AM...
anomaly dead code, var value is unused on all paths : global		Required	105 D	MISRA-C:2012/AM...

```
int32_t undef(void);
int32_t noret(void);

int32_t global = 10;

int32_t undef(void)
{
    int32_t value = -256; // signed int
    value = value >> 5;
    return value;
}

int32_t noret(void)
{
    global += undef();
    // no return exp.
}

int main(void)
{
    printf("%d %d\n", global, noret());
}
```

Programming Standards Call Diagram of program : C:\LDRA\_Workarea\_C\_CPP\_10.0.2\Examples\unspecToyama\unspec.c

Graph Type View Options Highlight Website Links Help

Procedure Calls: Number of Calls Call Type

Procedure	Number of Calls	Call Type
main	0	Internal
printf	1	System
noret	1	Internal
undef	1	Internal

Source Viewer - unspec.c

```
int32_t
undef (void)
{
    int32_t
    value = -256; // signed int
    value = value >> 5;
    return
    value;
}

int32_t
36 S Function has no return statement.: noret (MISRA-C:2012/AMD2 R.17.4)
noret (void)
{
    global ++
    undef ();
    // no return exp.
}

int
main (void)
{
    printf ("%d %d\n", global,
    noret ());
}
```

Call Diagram - unspec\_1

```
graph TD
    main[main] --> noret[noret]
    main --> undef[undef]
    noret --> undef
    undef --> printf[printf]
```

Code Review - noret : C - MISRA-C:2012 - Model

Number Violated	Level of Violation	Phase Code	Standard Code
1	Mandatory	36 S	MISRA-C:2012/AMD2 R.17.4

MISRA-C:2012 Legend: ■ Passed ■ Mandatory ■ Checking ■ Optional ■ Informational ■ Not Applicable

- View Code Review
- View Code Review by Violations
- View Filtered Code Review
- View Quality Qualification
- View Fault Qualification**
- View Security Qualification
- View Memory Qualification
- View Quality Review
- View Quality Review (Filtered)
- View Quality Review (Failures Only)

TBvision

Fault Module	Level 1	Level 2	Level 3
Fault	<input type="checkbox"/> Avoidances	<input type="checkbox"/> Defects	<input checked="" type="checkbox"/> Faults

OK Cancel

Results View

Fault Qualification : demo\_set

	Number Detected	Fault Qualification Status
demo_set		
demo4.c		
demo4_main		
bar		
DD data flow anomaly found. : yy		Fault
DD data flow anomalies found.		Fault
foo		
demo3.c		
main		
Unused procedure parameter. : argv		Fault
Procedure contains UR data flow anomalies. : value		Fault
Procedure contains UR data flow anomalies.		Fault
DD data flow anomalies found.		Fault
DD data flow anomaly found.	2	Fault
alloc_struct		
reassign		
Memory not freed after last reference. : lvar_1		Fault
get_ptr		
finale		
Comparing pointer with zero or NULL. : u16a_ptr		Fault
Shifting value too far. : bitmap << 9U		Fault
Copy source parameter not checked before use. : src_str		Fault
update_value		
Function and prototype param inconsistent (MR). : (unsigned int and No_Type): para...		Fault
Function has no return statement. : update_value		Fault
Identifier name reused. : tolerance		Fault
Incorrect array index validation. : pv		Fault
Procedure contains UR data flow anomalies. : tolerance		Fault
Procedure contains UR data flow anomalies.		Fault
DD data flow anomaly found. : ret		Fault

# CERTなど各種コーディング規約をサポート



- |                    |                    |
|--------------------|--------------------|
| CAST               | CERT               |
| CERT               | CWE                |
| CMSE               | Customer Sample    |
| CONFORM            | DERA               |
| CWE                | EADS               |
| Customer Sample    | FSB582-C++         |
| DERA               | HIC++              |
| EADS               | HIS                |
| FSB582-C           | JPL                |
| GJB                | JSF++ AV           |
| HIS                | LMTCP              |
| JPL                | Legacy             |
| Legacy             | MISRA-AC           |
| MISRA-AC           | MISRA-C:1998       |
| MISRA-C:1998       | MISRA-C:2004       |
| MISRA-C:2004       | MISRA-C:2012       |
| MISRA-C:2012       | NETRINO            |
| NETRINO            | RUNTIME            |
| RUNTIME            | SEC-C              |
| SEC-C              | Standard           |
| Standard           | TBrun Requires     |
| TBrun Requires     | UML                |
| UML                | VSOS               |
| VSOS               | No Standards Model |
| No Standards Model |                    |

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 3	Section 4	PDTMCSFVA	
Rule Number	Default Strength	Description	CAST	CERT	CWE	HIS	JPL	MISRA-AC	MISRA	MISRA-C 2004	MISRA-C 2012	NETRINO	SEC-C
1	C	Procedure name reused.	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
2	M	Label name reused.	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
3	M	More than *** executable reformatted lines in file.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
4	M	Procedure exceeds *** reformatted lines.	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
5	C	Empty then clause.	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
6	O	Procedure pointer declared.	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
7	C	Jump out of procedure.	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
8	C	Empty else clause.	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

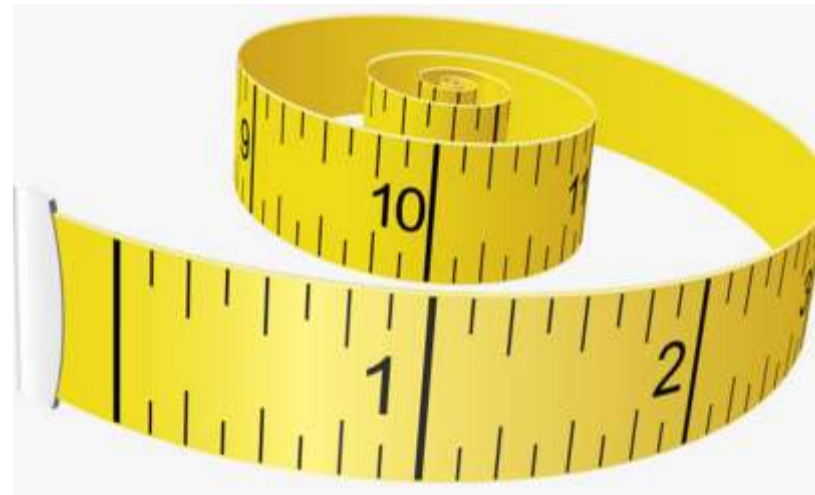
Report File (PC): Section 1 read (24 Models)(24 Modifiers)...Section 2 read (866 Rules)...Section 3 read...Section 4 read...Penalty File (PC): read

可読性が低下してバグや脆弱性が奥に潜んでしまう

テストは容易で無くなり余計な作業が必要になる

保守性が低く新たなバグや脆弱性への迅速な対応が困難

- 複雑になる傾向は避けられないにしても、
  - 複雑性は様々な尺度で測定できるので、
  - 過度に複雑にならないようにリファクタリングで改善する
    - Understandable 可読性
    - Maintainable 保守性
    - Testable テスト容易性



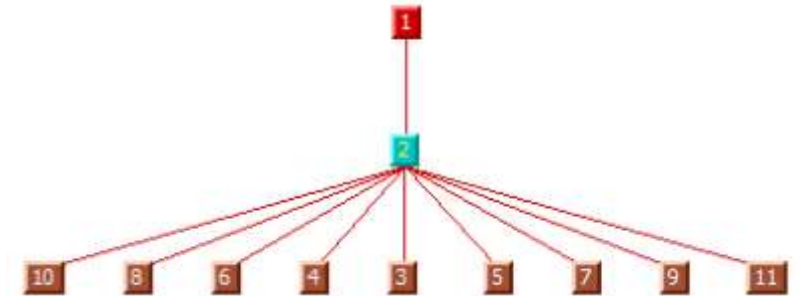
**「測定できないものは制御できない」**  
**“You can't control what you can't measure”**

Certification Standard	Reference	Description
EN 50128 (Railway)	Table A.12.8	Limited size and complexity of functions, subroutines, and methods
ISO 26262 (Automotive)	Section 5, Table 1A	Enforcement of low complexity
IEC 61508 (Industrial)	Annex B. Table B1.7	No unstructured control flow in programs in high-level language
IEC 62304 (Medical)	Section 5.1.4	Software development standards, methods and tools planning
CERT C (Security)	Top 10 Secure Coding Practices	Keep It Simple!

機能安全でなくてもシンプルが望ましいことは等しくあてはまる

多くのメトリクスがある：

- Number of lines of code
- Number of functions / globals
- Fan in / Fan out
- Depth of loop nesting
- Number of comments / Ratio of comments to code
- Halstead Metrics
- Knots
- Linear Code Sequence And Jumps
- McCabe Cyclomatic Complexity
- Essential Cyclomatic Complexity / Essential Knots



- 静的解析の一環としてMISRA等のコーディング規約と同時に解析される

	Value	Lower Limit	Upper Limit
cashregister			
Tester.c			
Clarity	78% Metrics Successful		
Maintainability	90% Metrics Successful		
Testability	93% Metrics Successful		
Metric Groupings			
goodbye			
Clarity	50% Metrics Successful		
Depth of Loop Nesting	0	0	2
Average Length of Basic Blocks	3%	1%	6%
Code Comments/Exe. Lines	0 : (Fail)	5	200
Declaration Comments/Exe. Lines	0 : (Fail)	1	100
Total Comments/Exe. Lines	66	10	200
Blank Lines	0	0	100
Comments in Executable Code	0 : (Fail)	1	100
Comments in Declarations	0	0	100
Comments in Headers	2 : (Fail)	5	50
Total Comments	2 : (Fail)	10	200
Maintainability	100% Metrics Successful		
Testability	100% Metrics Successful		
Metric Groupings			
Procedure Information	100% Metrics Successful		
Comments Associated with Procedures (% of total)	40% Metrics Successful		
Ratio of Comments to Executable lines (%)	60% Metrics Successful		
Complexity Metrics	100% Metrics Successful		
Loop/Interval Analysis	100% Metrics Successful		
Dataflow Information	100% Metrics Successful		
randomShopping			
Clarity	80% Metrics Successful		
Maintainability	100% Metrics Successful		
Testability	87% Metrics Successful		

Category	Value
Clarity	100%
Executable ref. Lines	147
Depth of Loop Nesting	1
Total LCSAJs	46
Unique Operands	105
Average Length of Basic Blocks	4.03%
Comments in Headers	53
Maintainability	100%
Number of Procedures	7
Unreachable Branches	0
Unreachable Lines	0
Maximum LCSAJ Density	4
Unreachable LCSAJs	0
Total LCSAJs	46
Vocabulary	119
Cyclomatic Complexity	9
Knots	23
Essential Cyclomatic Complexity	1
Essential Knots	0
Testability	100%
Fan Out	7
File Fan in	0
Number of Procedures	7
Procedure Exit Points	1
Number of Loops	5
Unreachable Branches	0
Unreachable Lines	0
Maximum LCSAJ Density	4
Unreachable LCSAJs	0
Total LCSAJs	46
Total Operands	286
Number of Basic Blocks	36
Executable reformatted Lines	145
Cyclomatic Complexity	9
Knots	23

# 例：Maintainability（保守性）で視覚化

**System View: Metrics Call Diagram - Maintainability - of set: Cashregister.exe**

Graph Type View Options Highlight Website Links Help

**Procedure Calls** Number of

- goodbye 1
- Productdatabase\_resetCountedProducts 1
- addProduct 1
- Productdatabase\_getCountedProduct 1
- Productdatabase\_getSpecificCountedProduct 1
- randomShopping 1
- identifyProduct 2
- Cashregister\_end 2
- rand 2
- sprintf 4
- Cashregister\_cancel 2
- Cashregister\_barcode 7
- Specialoffer\_getPrice 1
- startSession 1
- Userinterface\_print 7
- Userinterface\_parse 1
- Cashregister\_key 1
- main 0
- countProducts 1
- getchar 1
- Cashregister\_code 1
- generateTicket 1
- Userinterface\_help 2
- Productdatabase\_getProduct 1
- Cashregister\_start 2
- endSession 2
- removeLastProduct 1
- printf 19
- Userinterface\_show 6

**Call Diagram - Cashregister.exe**

**Code Quality View (Maintainability)**

Procedure Calls	Cyclomatic Complexity	Knots	Essential Cyclomatic Complexity	Essential Knots
goodbye	1	0	1	0
Productdatabase_resetCountedProducts	2	4	1	0
addProduct	3	1	1	0
Productdatabase_getCountedProduct	4	2	1	0
Productdatabase_getSpecificCountedProduct	2	1	1	0
randomShopping	3	2	1	0
identifyProduct	2	1	1	0
Cashregister_end	2	0	1	0
Cashregister_cancel	3	1	1	0
Cashregister_barcode	2	0	1	0
Specialoffer_getPrice	4	6	1	0
startSession	2	1	1	0
Userinterface_print	1	0	1	0
Userinterface_parse	16	32	1	0
Cashregister_key	2	0	1	0
main	2	1	1	0
countProducts	4	1	1	0
Cashregister_code	2	0	1	0
generateTicket	5	2	1	0
Userinterface_help	1	0	1	0
Productdatabase_getProduct	3	2	1	0
Cashregister_start	2	0	1	0
endSession	1	0	1	0
removeLastProduct	3	0	1	0
Userinterface_show	1	0	1	0

**Source Viewer - generateTicket**

```
static void generateTicket ( void )  
{  
    char_t  
    msgString [ 64U ];  
    uint32_t  
    sum_total = 0U;  
    char_t  
    theSpecialoffer;  
    uint32_t  
    thePrice;  
}
```

**System View: Metrics Call Diagram - Maintainability - of s**

Graph Type View Options Highlight Website Links

Calls View - System: Cashregist... Maintainability


- 複雑な関数を視覚化して分析

The screenshot displays the LDRA Code Quality View (Maintainability) interface. On the left, a flow diagram for the procedure `Userinterface_parse` is shown, with a red box highlighting a specific code block in the source viewer. The source viewer lists several code blocks, including `code '1' Cashregister_barcode (1034EU) break;`, which is highlighted with a red box. The main window shows a detailed flow diagram with a red path starting from a red circle and branching into multiple paths, each ending in a diamond shape. The 'Code Quality View (Maintainability)' window shows a table of procedure calls and cyclomatic complexity values.

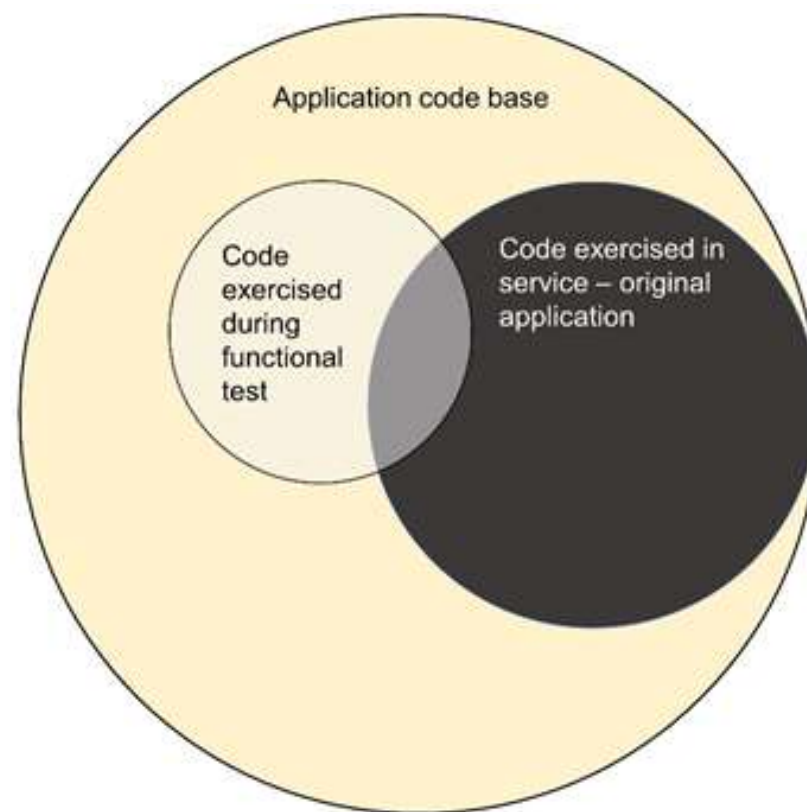
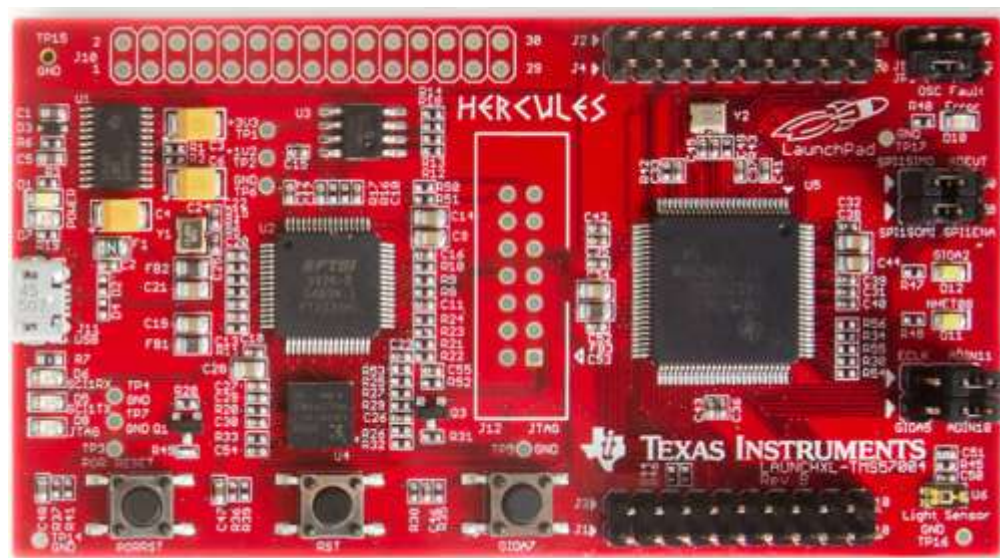
Procedure Calls	Cyclomatic
Userinterface_parse	16
generateTicket	5
Specialoffer_getPrice	4
Productdatabase_getCountedProduct	4
countProducts	4

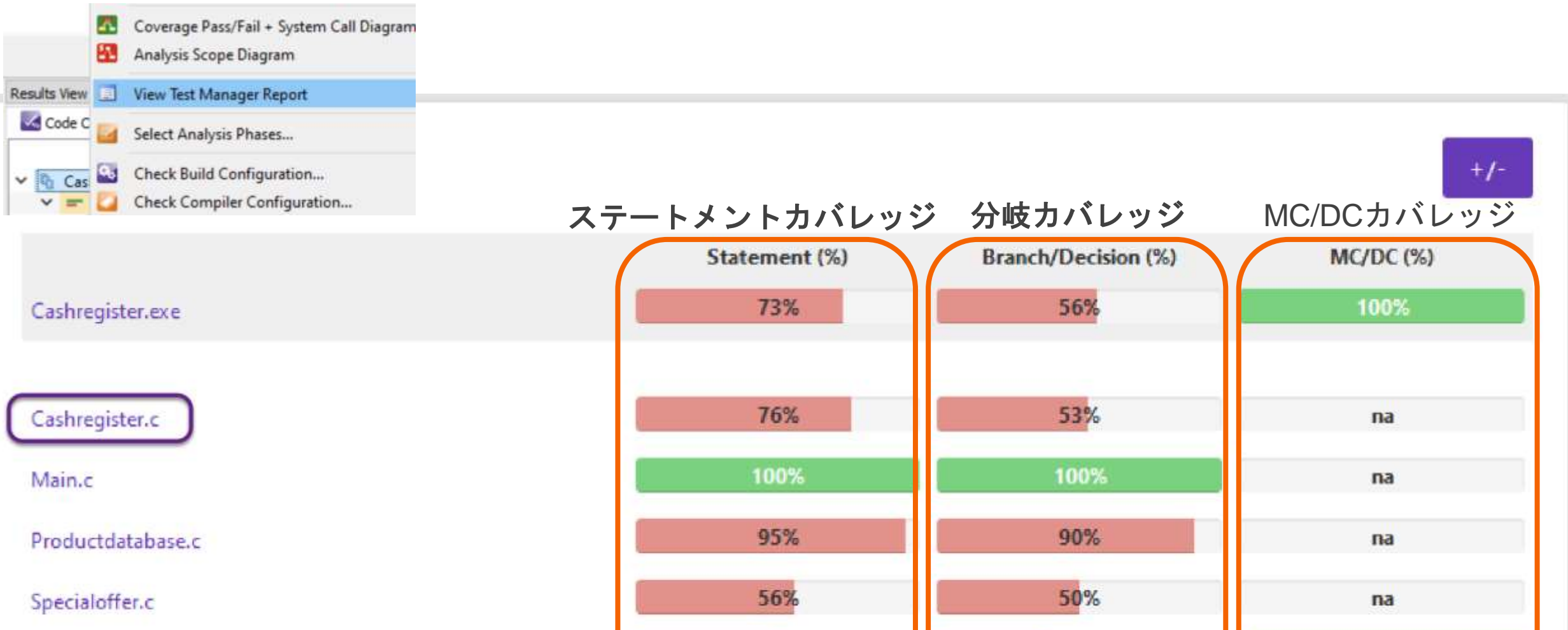
  

Procedure Calls	Number of Calls	Call Type
Cashregister_barcode	7	Internal
Specialoffer_getPrice	1	Internal
startSession	1	Internal
Userinterface_print	7	Internal
Userinterface_parse	1	Internal

- 
- 1 航空業界のベストプラクティス
  - 2 静的解析、コーディング規約チェックなど
  - 3 動的テストとカバレッジ解析
  - 4 要件トレーサビリティ
  - 5 まとめ

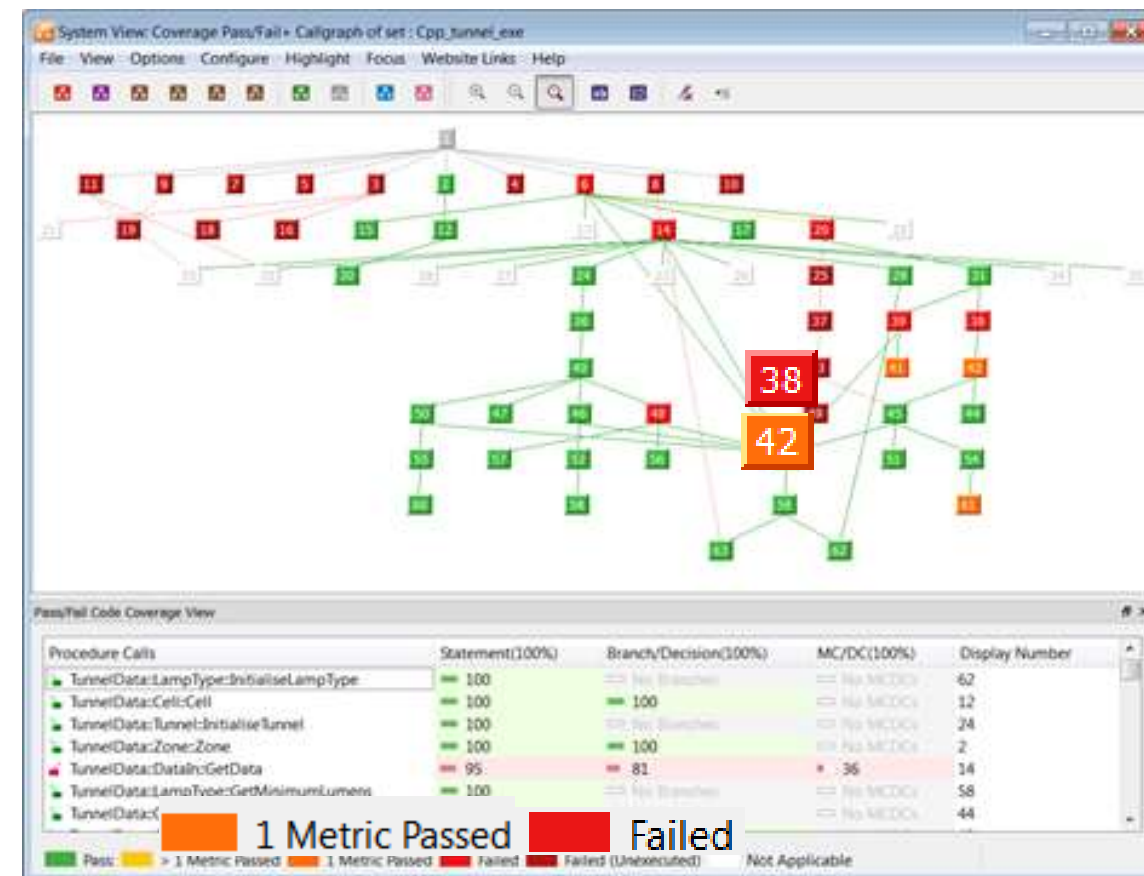
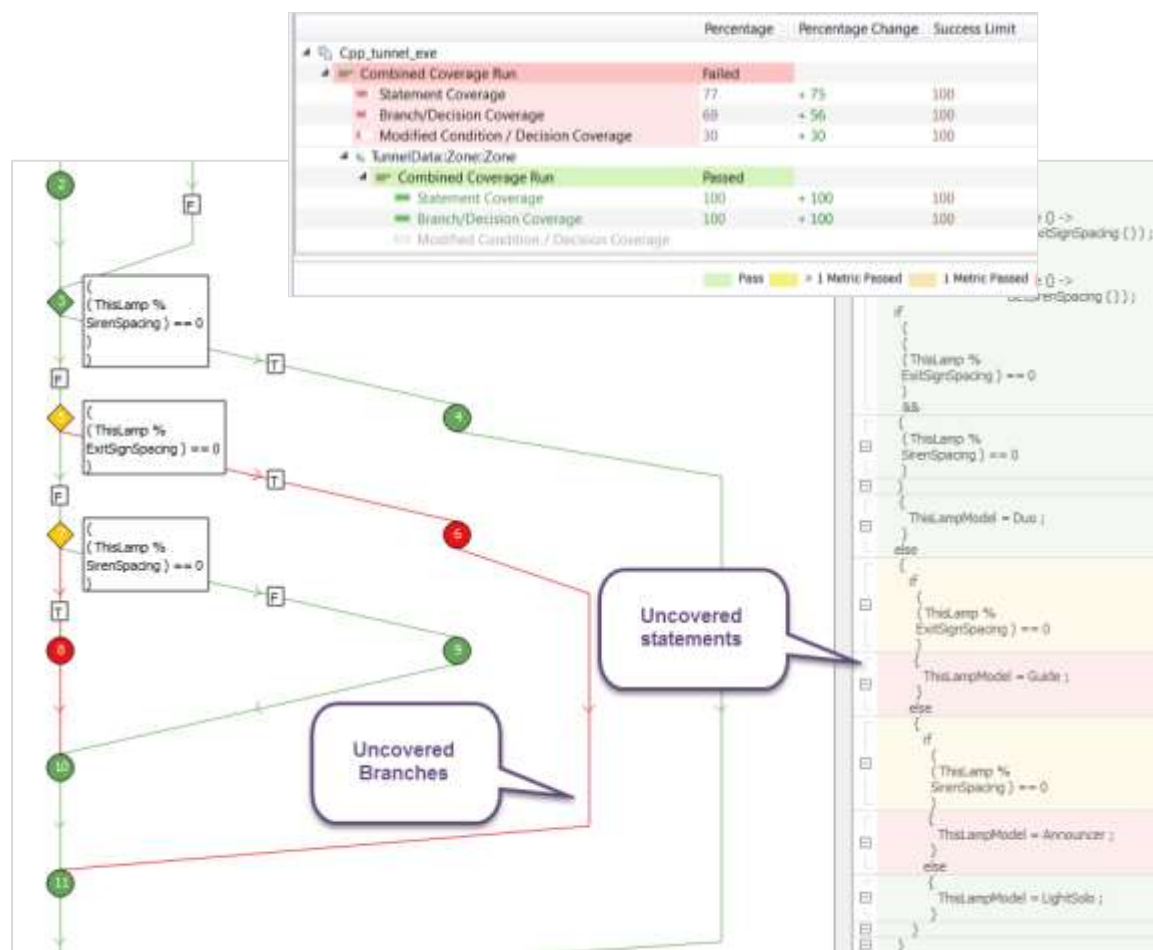
- 実ターゲットレベルのテスト実行なら、静的解析では叶わないソフトウェアハードウェアインターフェースを評価できる
- ただ機能テストだけではコードの一部しか実行されないまま出荷され、予期しない結果が露呈することを否認ない





出荷されて初めて実行されるようなコードを無くし、不要なコードは削除する

Item	Description	DO-178C Reference	DO-178C Level A	DO-178C Level B	DO-178C Level C
5	Test coverage of software structure (MC/DC) is achieved	6.4.4.2	✓	Not Required	Not Required
6	Test coverage of software structure (decision coverage) is satisfied	6.4.4.2a 6.4.4.2b	✓	✓	Not Required
7	Test coverage of software structure (statement coverage) is satisfied	6.4.4.2a 6.4.4.2b	✓	✓	✓
8	Test coverage of software structure (data coupling and control coupling) is achieved	6.4.4.2c	✓	✓	✓
9	Verification of additional code, that cannot be traced to Source Code, is achieved	6.4.4.2d	✓	Not Required	Not Required



未実行パスの分岐条件等进行分析して、追加のテストケース等の考察に役立てる

## LDRA

**Source Code for Decision : ( A | B ) & C**

```
12 | result = (a || b) && c;
```

**Conditions in Decision : ( A | B ) & C**

A	a
B	b
C	c

**Full Truth Table for Decision : ( A | B ) & C**

Index	Conditions			Expected Outcome	Executed	MC/DC Independent Pairs Wave
	A	B	C			
1	F	F	F	= F	YES	
2	T	F	F	= F	NO	# .....C.
3	F	T	F	= F	NO	# ..... ..C.
4	T	T	F	= F	NO	..... .. ..C.
5	F	F	T	= F	YES	#* .A..B.. .. ..
6	T	F	T	= T	YES	#* .A.. ..C.. ..
7	F	T	T	= T	YES	#* ....B.....C..
8	T	T	T	= T	YES	#* .....C.

**Modified Condition/Decision Profile for Decision : ( A | B ) & C**

Condition	Truth Table Index	Combination Eff

File View

First

Sequence File Explorer

- ▼ Mcdctest.cpp - C:\LDRA\_Workarea\_C\_CPP\_10.0.0\Examples\Cpp\_tbrun\_examples\Mcdc\_workshop\
  - > Global Variables
  - ▼ Combined Coverage Run
    - Statement Coverage - 100%
    - Branch/Decision Coverage - 100%
    - Modified Condition / Decision Coverage - 67%
  - ▼ Current Coverage Run
    - Statement Coverage - 100%

## TunnelData::Tunnel::AdjustLighting (const Float\_64,Bool) +/-

### Control Coupling (Calls Out from Procedure)

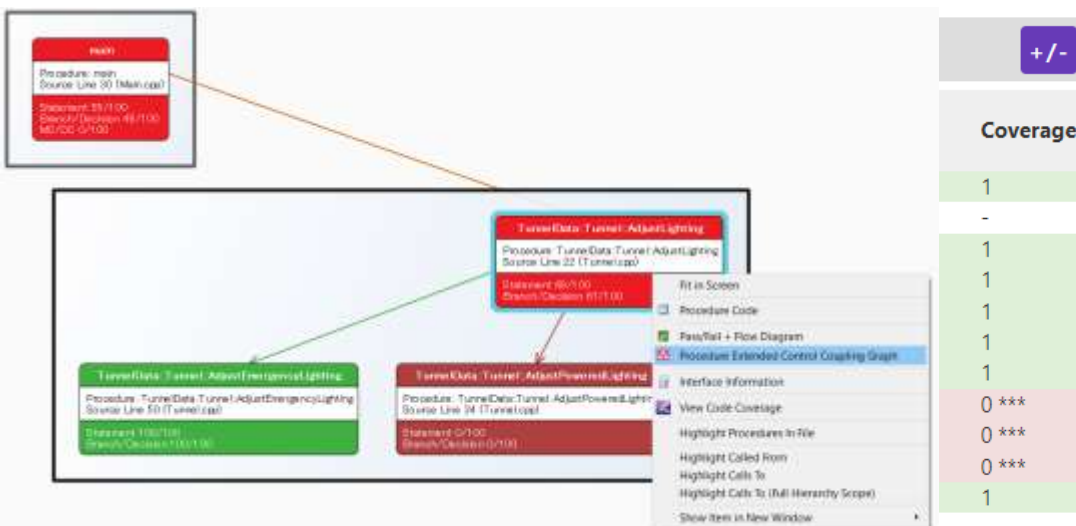
Call From	Call To	Call To File	Called on Line	Coupling Covered
TunnelData::Tunnel::AdjustLighting	TunnelData::Tunnel::AdjustEmergencyLighting	Tunnel.cpp	26	Covered
TunnelData::Tunnel::AdjustLighting	TunnelData::Tunnel::AdjustPoweredLighting	Tunnel.cpp	30	Not Covered ***

### Control Coupling (Calls In to Procedure)

Call From	Call To	Call From File	Called from Line	Coupling Covered
main	TunnelData::Tunnel::AdjustLighting	Main.cpp	90	Not Covered ***
main	TunnelData::Tunnel::AdjustLighting	Main.cpp	103	Not Covered ***
main	TunnelData::Tunnel::AdjustLighting	Main.cpp	110	Covered

### Source Code with Statement Coverage +/-

Source Line	Statement	Coverage
22	<code>void Tunnel::AdjustLighting (const Float_64 PhotometerDemand, Bool PowerFailure)</code>	1
23	<code>{</code>	-
24	<code>if (PowerFailure)</code>	1
25	<code>{</code>	1
26	<code>AdjustEmergencyLighting();</code>	1
27	<code>}</code>	1
28	<code>else</code>	1
29	<code>{</code>	0 ***
30	<code>AdjustPoweredLighting(PhotometerDemand);</code>	0 ***
31	<code>}</code>	0 ***
32	<code>};</code>	1



## TunnelData::Tunnel::AdjustLighting (const Float\_64,Bool) +/-

### Data Coupling (Out from Procedure)

Variable	Line	Type	Coupled To Parameter (Alias)	Procedure	Line	File
const Float_64 PhotometerDemand	30	Parameter	--> const Float_64 PhotometerDemand	TunnelData::Tunnel::AdjustPoweredLighting	34	Tunnel.cpp

### Data Coupling (In to Procedure)

Variable	Procedure	Line	File	Type	Coupled To Parameter (Alias)	Line
Float_64 Photometer	main	90	Main.cpp	Local	--> const Float_64 PhotometerDemand	22
Bool PowerFailure	main	90	Main.cpp	Local	--> Bool PowerFailure	22
Float_64 Photometer	main	103	Main.cpp	Local	--> const Float_64 PhotometerDemand	22
Bool PowerFailure	main	103	Main.cpp	Local	--> Bool PowerFailure	22
Float_64 Photometer	main	110	Main.cpp	Local	--> const Float_64 PhotometerDemand	22
Bool PowerFailure	main	110	Main.cpp	Local	--> Bool PowerFailure	22

### Dynamic Data Flow Coverage for Data Coupling Variables

Variable Name	File	Procedure	Type Code	Attribute Code	Used on lines...
PhotometerDemand	Tunnel.cpp	TunnelData::Tunnel::AdjustLighting	Parameter	Declaration	22
			Parameter	Reference	30 *****
PowerFailure	Tunnel.cpp	TunnelData::Tunnel::AdjustLighting	Parameter	Declaration	22
			Parameter	Reference	24

### Summary

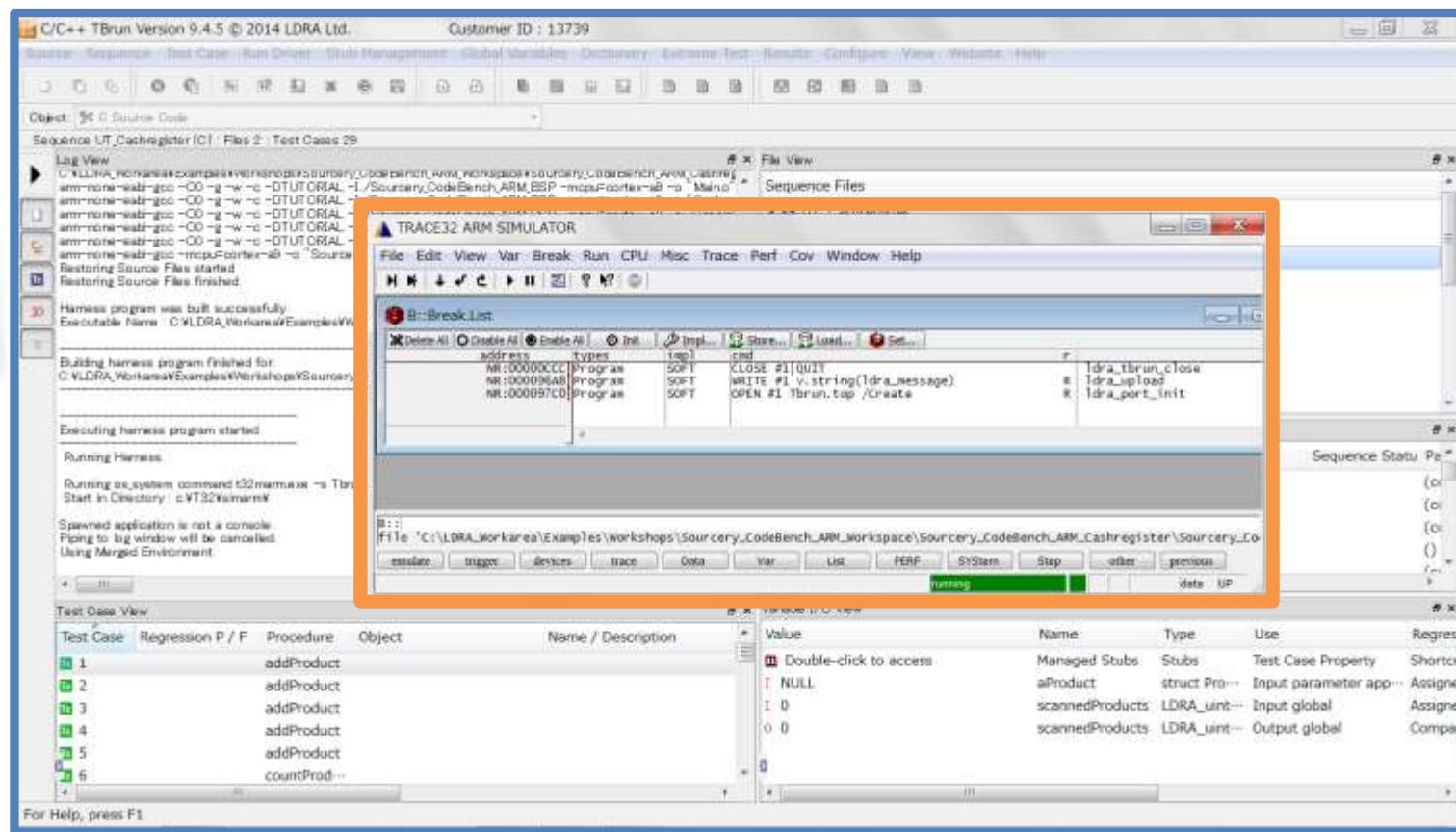
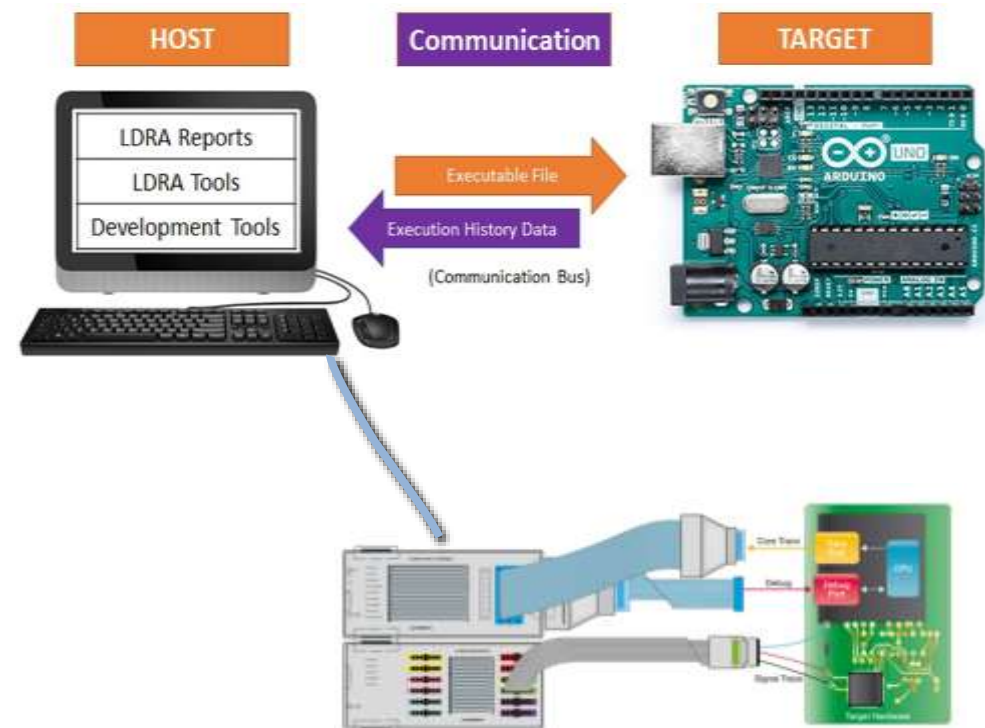
	Uses
Total Data Coupling Variable Uses	2
Data Coupling Variable Uses Covered	1
Data Coupling Variable Uses Not Covered	1
Overall Data Coupling Variable Use Coverage (%)	50%

### Source Code with Statement Coverage +/-

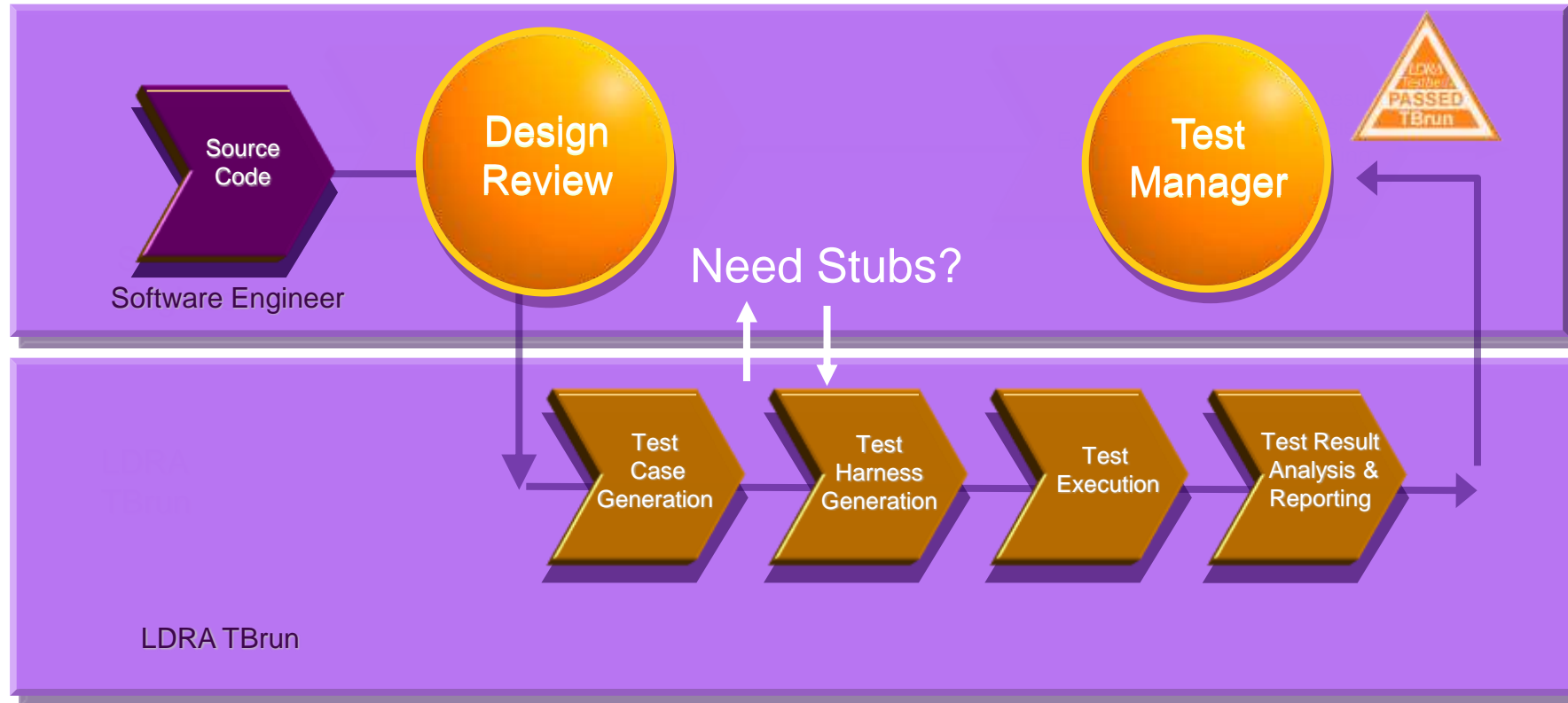
The screenshot displays the LDRA Coverage Pass/Fail+ interface for the procedure `_countProducts`. It features three main panes: a flowgraph on the left, a source code viewer in the center, and an assembly code viewer on the right. The flowgraphs use colored nodes and arrows to represent the execution flow, with green indicating covered branches and red indicating uncovered ones. The source code viewer shows the C code for `_countProducts`, including a `while` loop that iterates over scanned products. The assembly code viewer shows the corresponding assembly instructions, such as `push ebp`, `mov eax, [ebp+8]`, and `inc eax`.

Legend:   
 - Green Circle: Branch/Decision Covered   
 - Yellow Circle: Partial Branch/Decision Coverage   
 - Red Circle: Branch/Decision not Covered   
 - Diamond: Node has Branch/Decision | Circle: Node has no Branch/Decision

ソースコードから実行コードへのトレーサビリティを確保することで、  
DO-178Cのオブジェクトレベルを効率的に達成

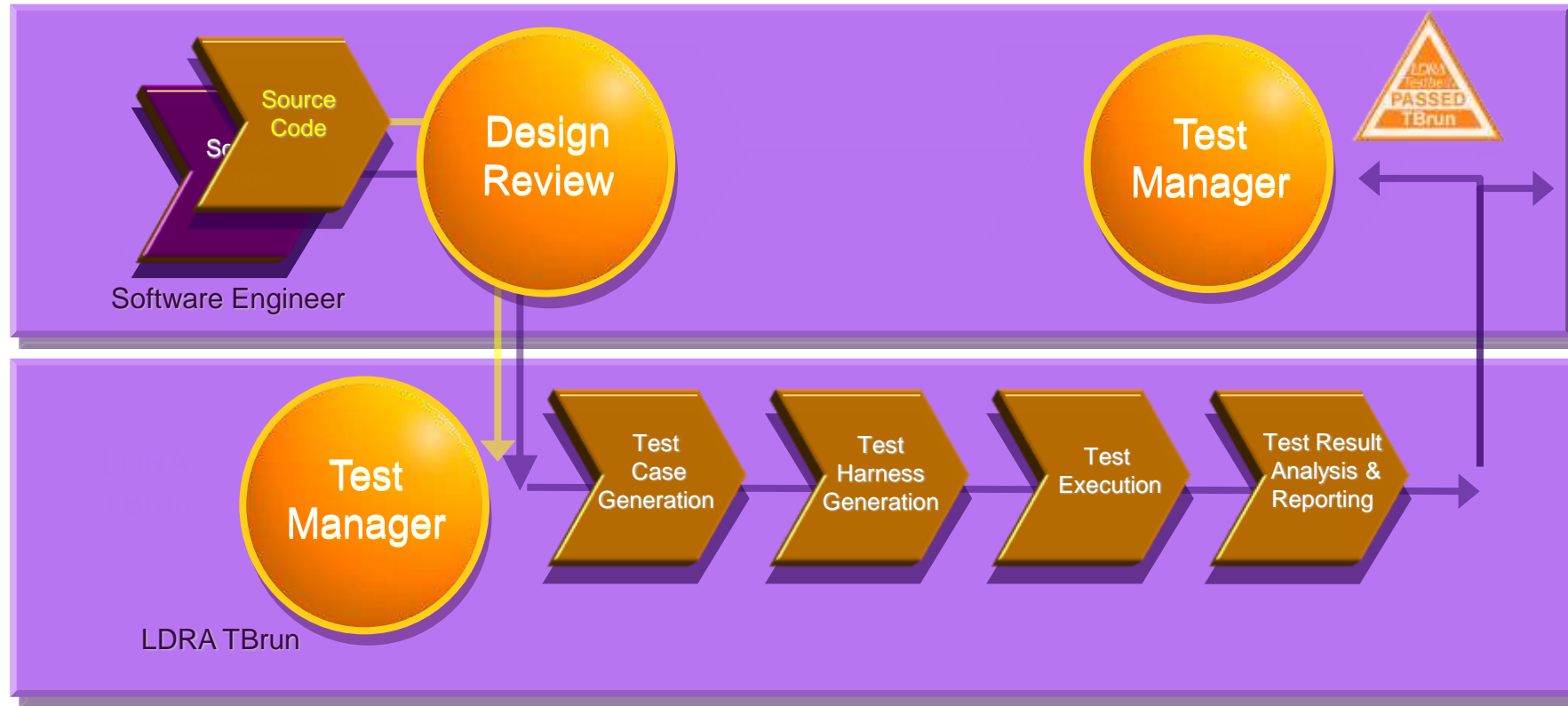


構造カバレッジオブジェクトタイプの達成、低レベル要件の検証、  
検証コストの大幅な削減



静的解析時に得た情報を利用

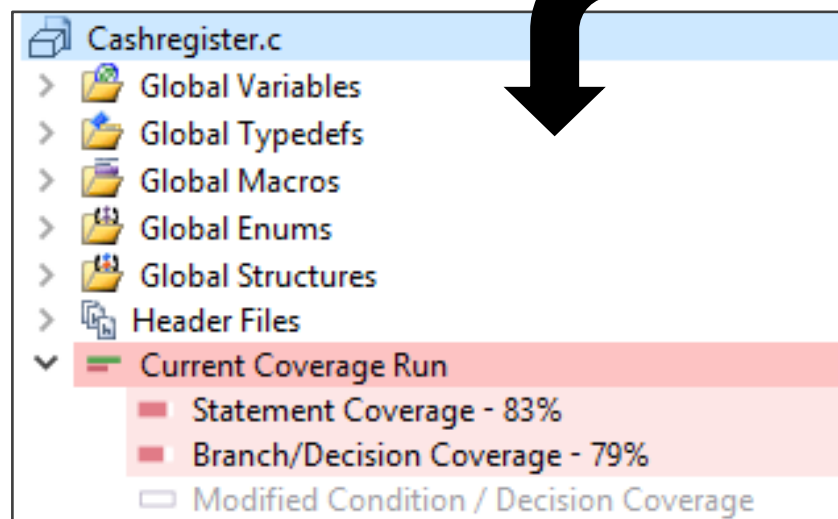
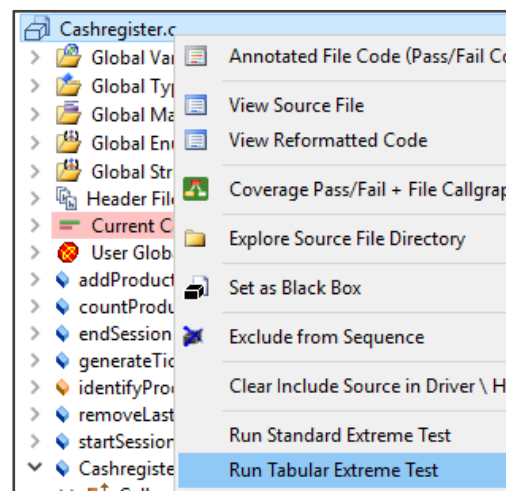
# ソースコードが修正されると



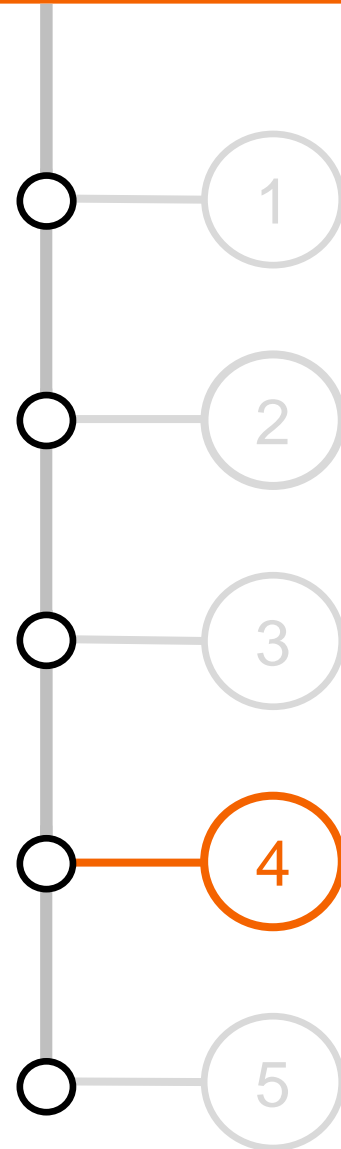
リグレッションテストを支援

# 単体テストのテストケース自動生成

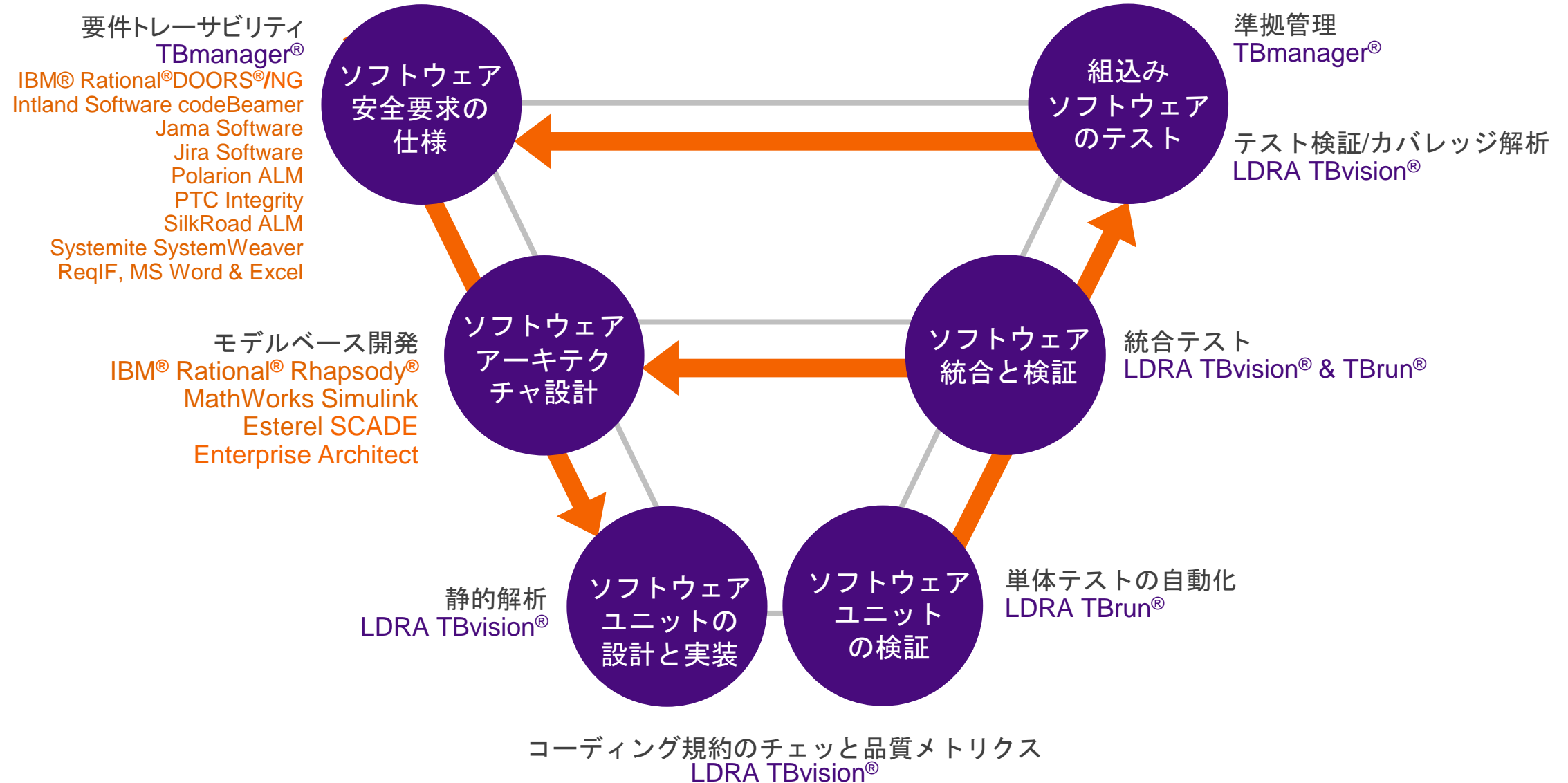
- 堅牢性テストに活用
- 機能テストの補足
- 単体テストのテンプレート
- 派生製品等の開発にも



Test Case	Regression P / F	Procedure
Tc 1	PASS	addProduct
Tc 2	PASS	addProduct
Tc 3	PASS	addProduct
Tc 4	PASS	addProduct
Tc 5	PASS	countProducts
Tc 6	PASS	endSession
Tc 7	PASS	generateTicket
Tc 8	PASS	identifyProduct
Tc 9	PASS	removeLastProduct
Tc 10	PASS	removeLastProduct
Tc 11	PASS	startSession
Tc 12	PASS	Cashregister_barcode
Tc 13	PASS	Cashregister_barcode
Tc 14	PASS	Cashregister_cancel
Tc 15	PASS	Cashregister_cancel
Tc 16	PASS	Cashregister_cancel
Tc 17	PASS	Cashregister_cancel
Tc 18	PASS	Cashregister_cancel
Tc 19	PASS	Cashregister_cancel
Tc 20	PASS	Cashregister_code
Tc 21	PASS	Cashregister_code
Tc 22	PASS	Cashregister_end
Tc 23	PASS	Cashregister_end
Tc 24	PASS	Cashregister_end
Tc 25	PASS	Cashregister_key
Tc 26	PASS	Cashregister_key
Tc 27	PASS	Cashregister_start
Tc 28	PASS	Cashregister_start

- 
- 1 航空業界のベストプラクティス
  - 2 静的解析、コーディング規約チェックなど
  - 3 動的テストとカバレッジ解析
  - 4 要件トレーサビリティ
  - 5 まとめ

## ソフトウェア開発プロセスと自動化ツールチェーン



- 様々なレベルの要件とソースコードを双方向に追跡可能にして、正しく要件を満たしていることの検証作業と証明を支援

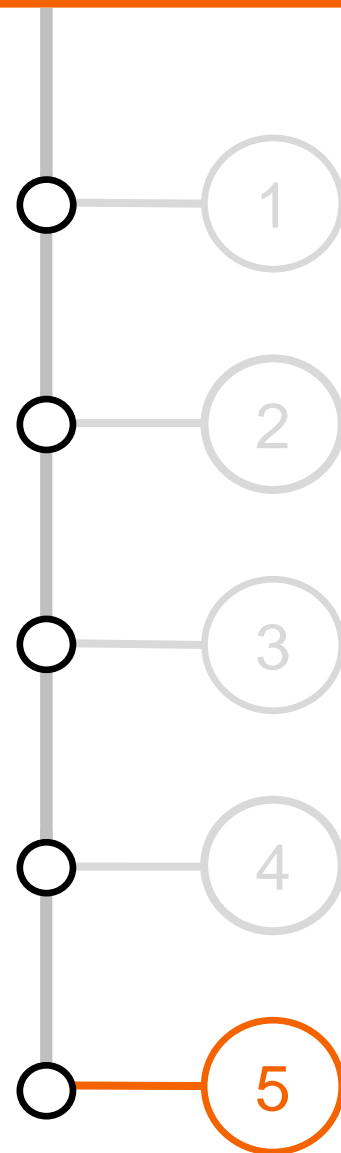
The screenshot displays the TBmanager interface with a hierarchical tree of requirements and tests. The tree is organized into several panes:

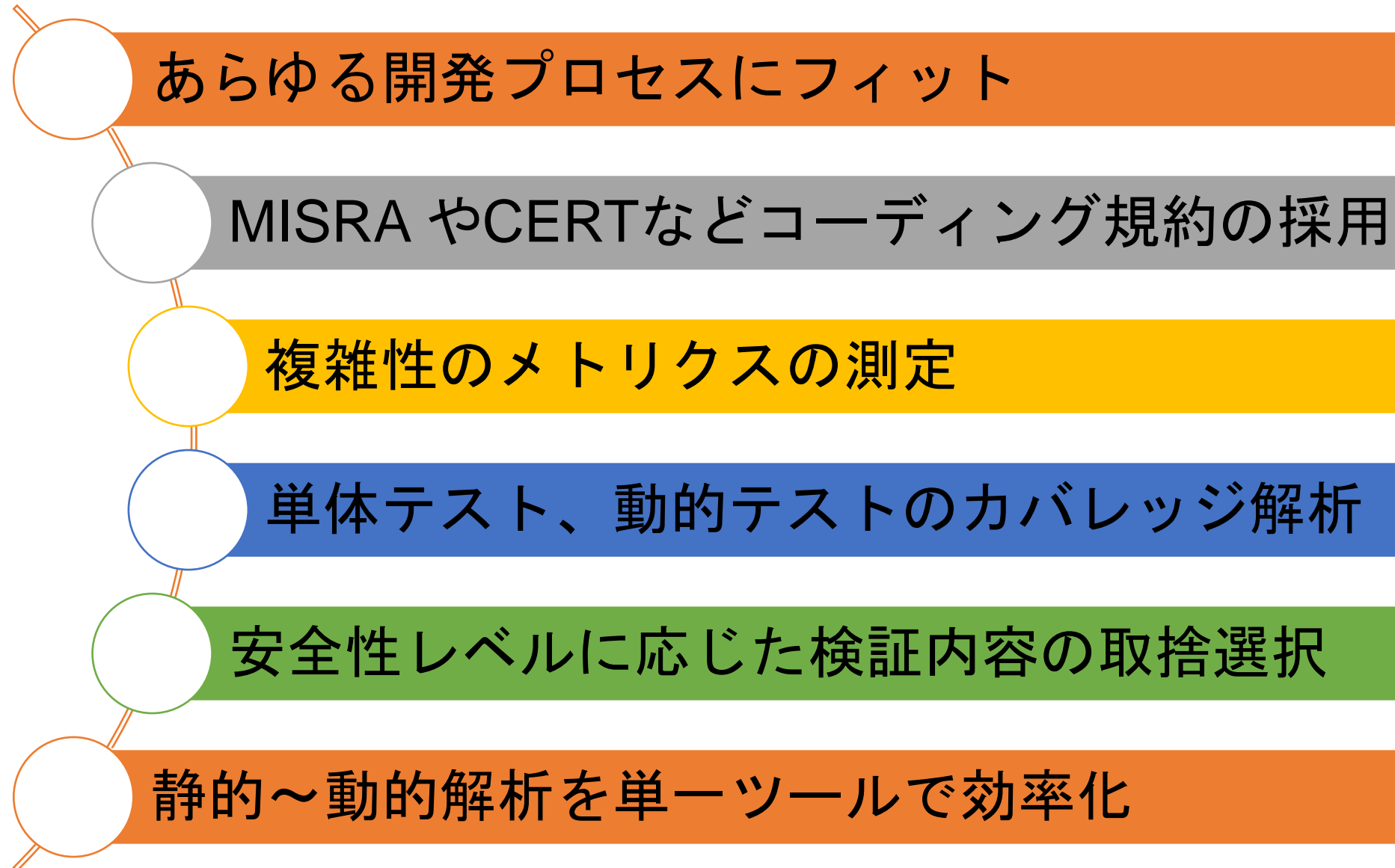
- Left Pane:** System-level requirements (SYS\_0010 to SYS\_0130).
- Middle Pane:** High-level requirements (HLR\_0090 to HLR\_0190).
- Right Pane:** Low-level requirements (LLR\_0284 to LLR\_0370).
- Bottom Pane:** Test cases (TCI\_5100 to TCI\_5230).

The bottom pane shows a list of test cases, including:

- TCI\_5100: Verify that soiling factor is calculated correctly per formula described in LLR\_...
- TCI\_5110: Verify that the Cell::Cell is correctly instantiated
- TCI\_5120: Verify that a given is instantiated and initialised correctly
- TCI\_5125: Verify that Cell::SetEmergencyOutputLevel is set to its defined emergency de...
- TCI\_5130: Verify that powered output settings is assigned consistently for any given si...
- TCI\_5140: Low Level Requirements based tests
- TCI\_5180: Verify that for a given lamp and spacing for exit signs and sirens the correct l...
- TCI\_5190: Verify that TunnelData::DataIn::ReadContent loads .ini file and stores the dat...
- TCI\_5200: Verify that tokens are copied into ZoneData are the are parsed from the .ini f...
- TCI\_5210: Verify that Lamp::Lamp is called for construction of the Lamp object. Additi...
- TCI\_5220: Verify that Lamp::SetLumensOutput outputs the number of lumens per lamp
- TCI\_5230: Verify that Lamp::GetMaximumLumens() returns the Maximum lumens allow...

各スタンダードのオブジェクトの達成管理機能も備え  
認証プロセスのリスクとコストを軽減

- 
- 1 航空業界のベストプラクティス
  - 2 静的解析、コーディング規約チェックなど
  - 3 動的テストとカバレッジ解析
  - 4 要件トレーサビリティ
  - 5 まとめ



Need more information?

 **FUJI SETSUBI**  
[www.fuji-setsu.co.jp](http://www.fuji-setsu.co.jp)





LDRA社

<https://ldra.com/>

LDRA スタンドアード認証支援テストツール

<https://www.fuji-setsu.co.jp/products/LDRA/>



富士設備工業(株)電子機器事業部  
<https://www.fuji-setsu.co.jp>