LDRA

# LDRA Software Technology

# Agenda

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

# LDRA社テスト自動化支援ツール


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

ソフトウエアテスト、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,

# LDRA Standards Experience & Pedigree

## 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

# Experts in Safety and Security Critical Software







# 航空業界で標準的に採用されるLDRA社テストツール

- 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

# Agenda

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

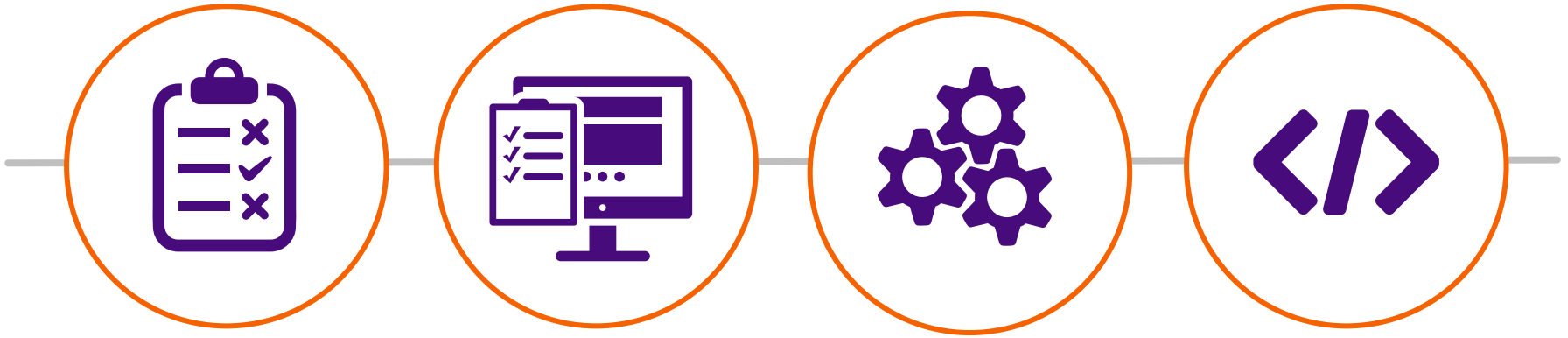
# コーディング規約

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

# 未定義、処理系定義、未規定 の動作を避ける

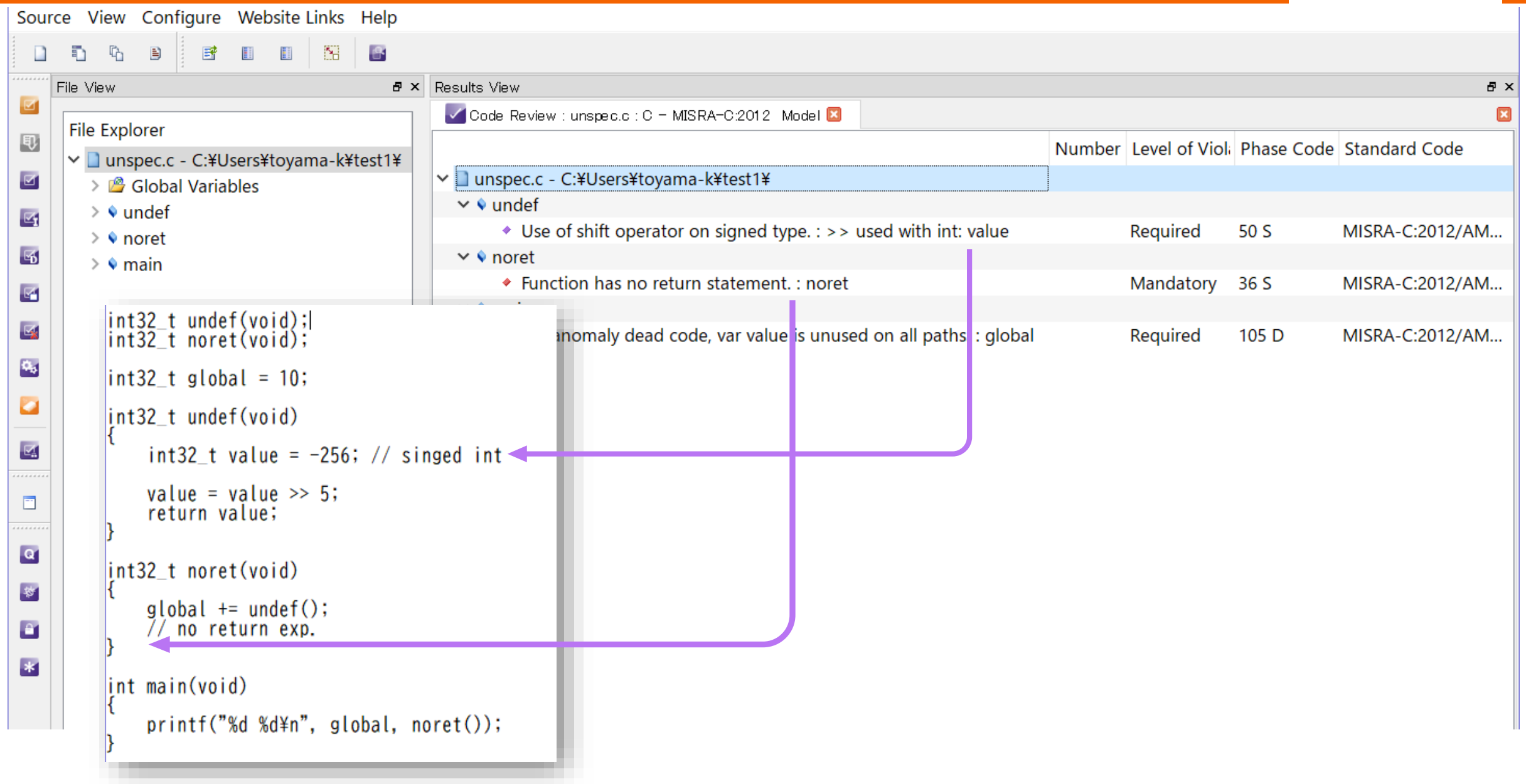
- C/C++によるプログラミングで不具合の原因となる根本的な言語要素:
  - 未定義の動作:言語規格で定義されない動作
  - 未規定の動作:規定されない動作
  - 処理系定義の動作:未規定の動作に対して処理系が独自に動作を実装
- アプリケーションの

  信頼性や移植性が損なわれるので、

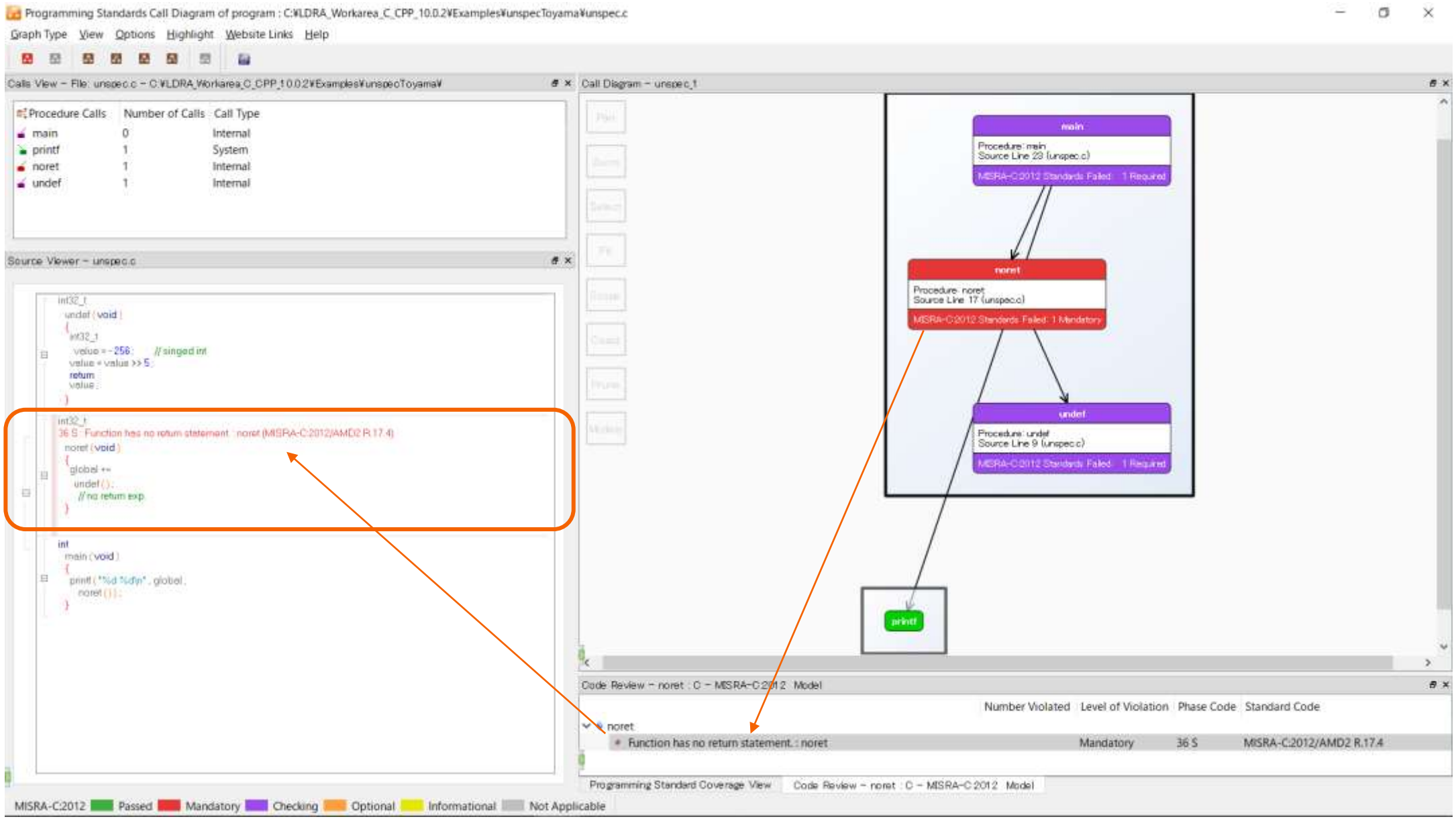
  重要度に関わらずこれらの使用は避けるべき


**MISRAのようなガイドライン を適用すること**

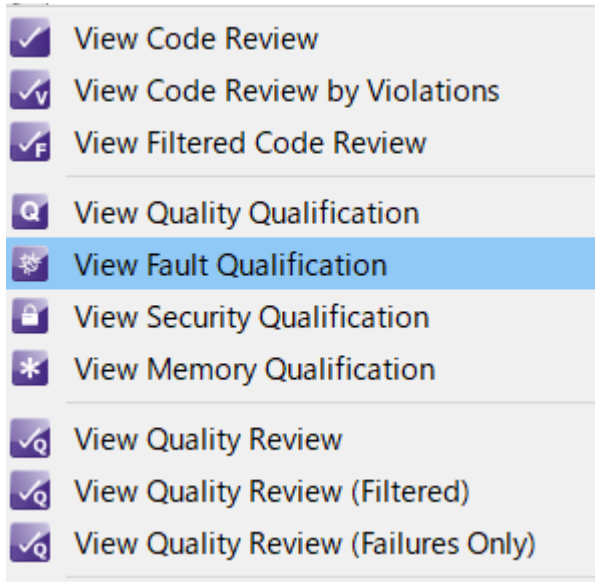
# コーディング規約チェック

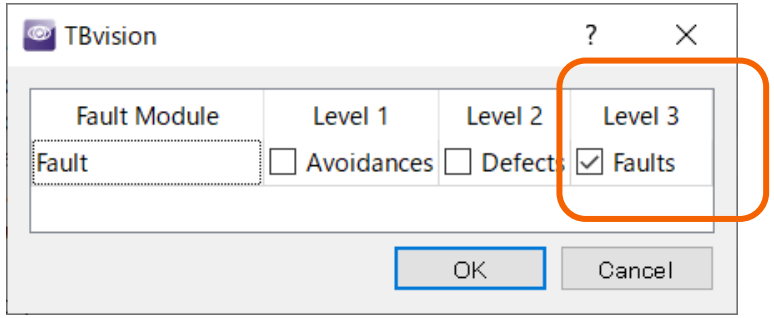


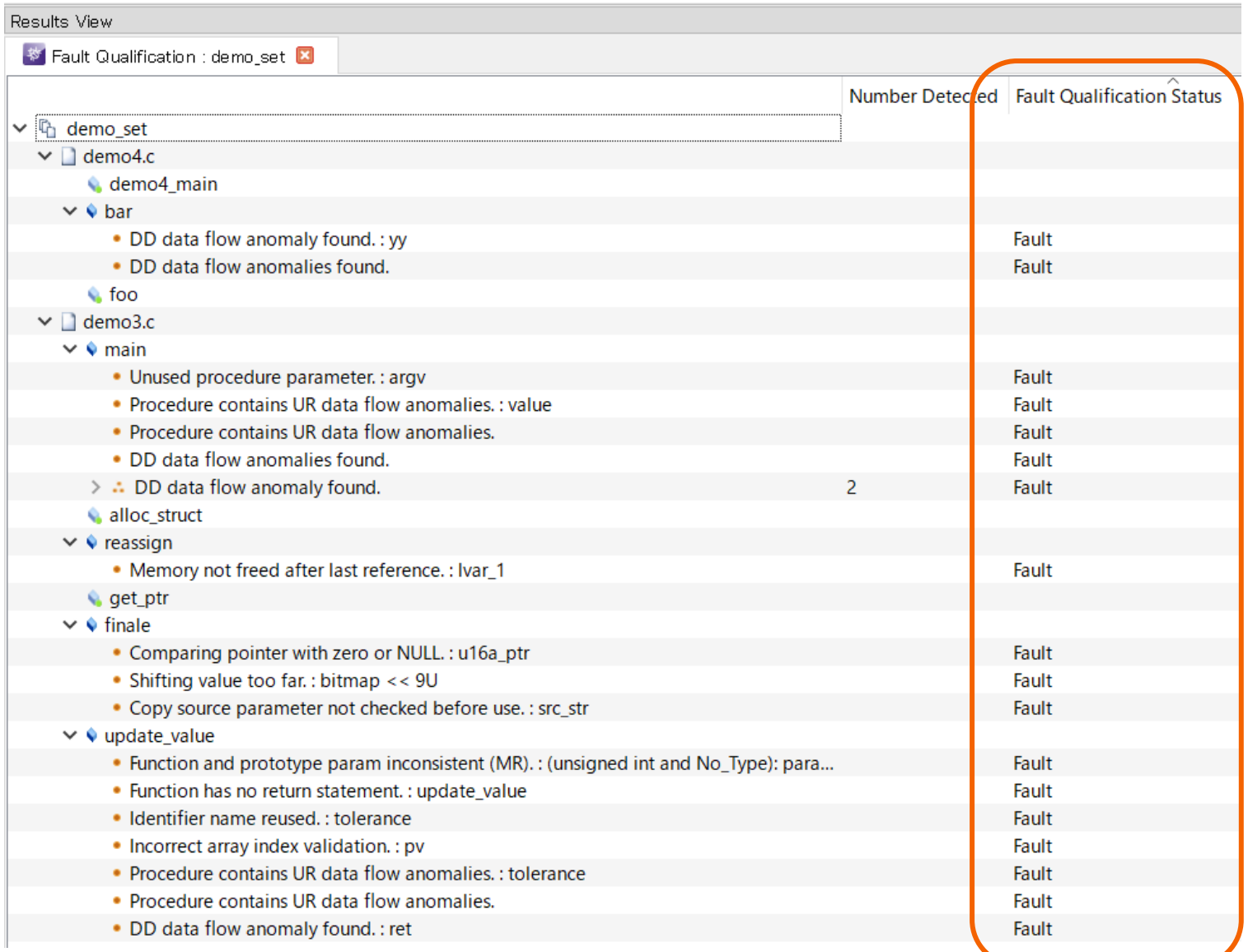
# 関数コールグラフに視覚化



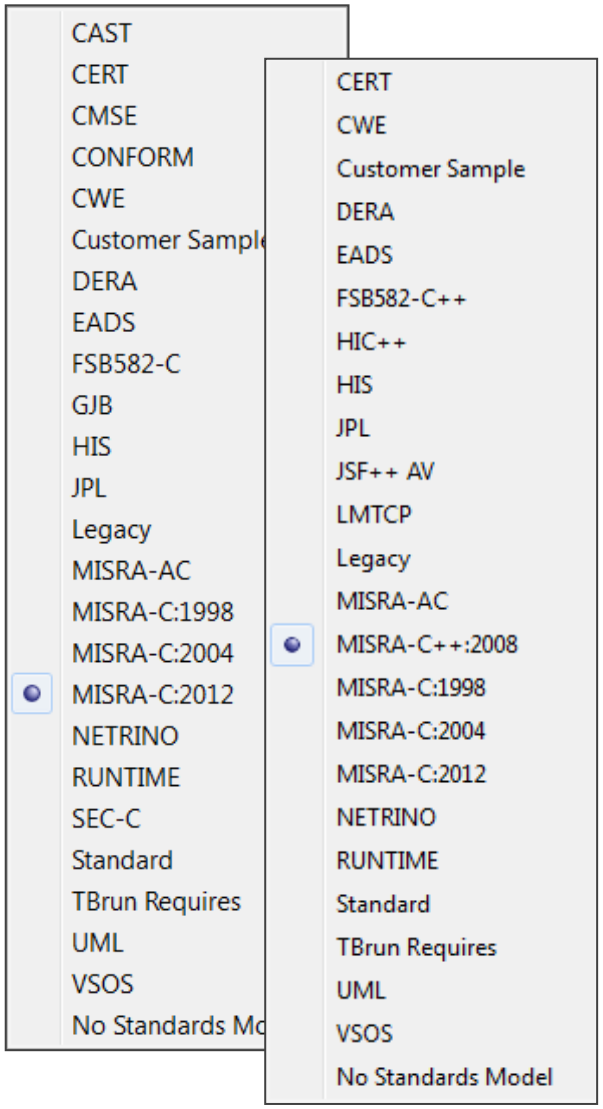
# リスクベースのアプローチ

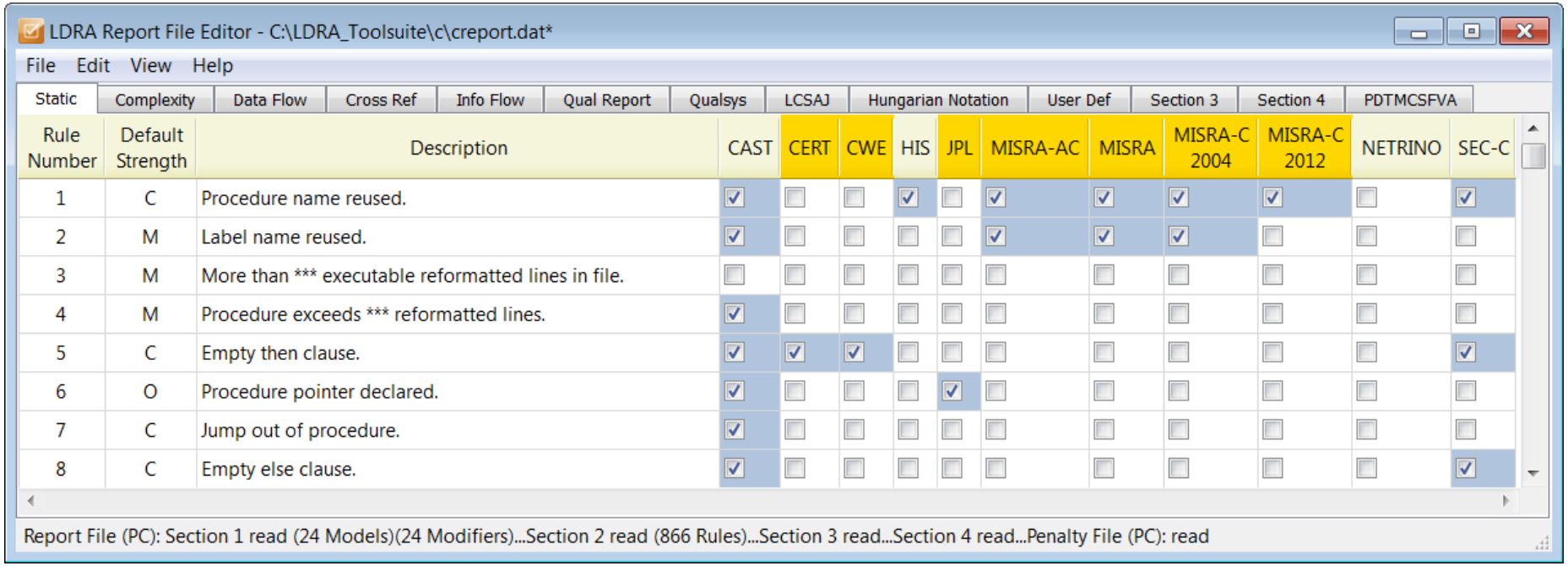


# CERTなど各種コーディング規約をサポート



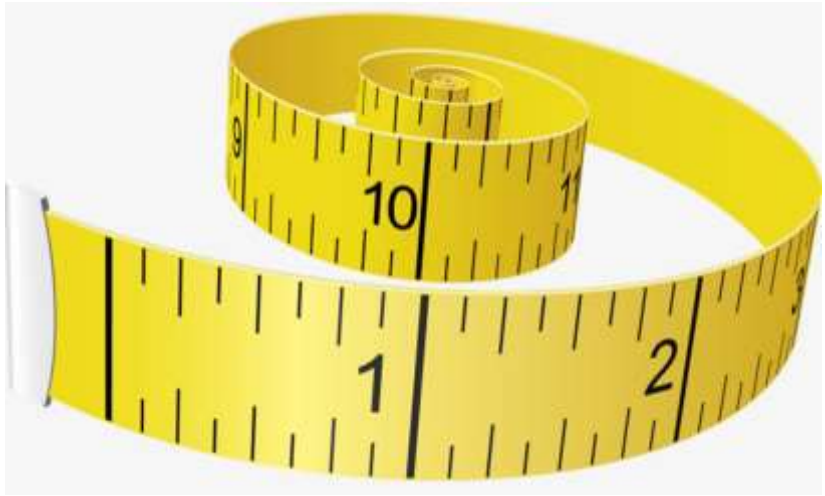

| Rule Number | Default Strength | Description | CAST | CERT | CWE | HIS | JPL | MISRA-AC | MISRA | MISRA-C 2004 | MISRA-C 2012 | NETRINO | SEC-C |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | C | Procedure name reused. | ☑ | ☐ | ☐ | ☑ | ☐ | ☑ | ☑ | ☑ | ☑ | ☐ | ☑ |
| 2 | M | Label name reused. | ☑ | ☐ | ☐ | ☐ | ☐ | ☑ | ☑ | ☑ | ☐ | ☐ | ☐ |
| 3 | M | More than *** executable reformatted lines in file. | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ |
| 4 | M | Procedure exceeds *** reformatted lines. | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ |
| 5 | C | Empty then clause. | ☑ | ☑ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☑ |
| 6 | O | Procedure pointer declared. | ☑ | ☐ | ☐ | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ |
| 7 | C | Jump out of procedure. | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ |
| 8 | C | Empty else clause. | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☑ |

# ソフトウエアの複雑性

- 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

# ソフトウエアの複雑性

- 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”***

# 機能安全規格やセキュリティスタンダードでも推奨

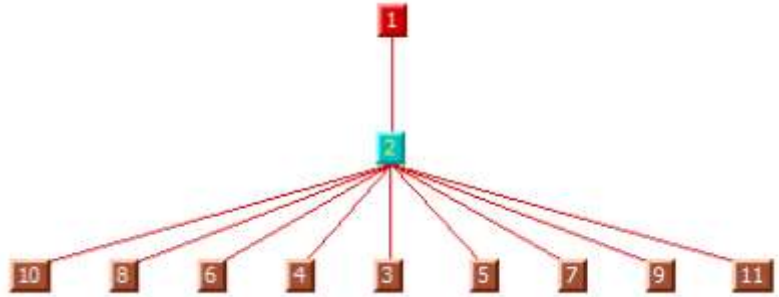
| Certification Standard | Reference | Description |
|---|---|---|
| EN 50128 (Railway) | Table A.12.8 | Limited size and complexity of functions, subroutines, and methods |
| ISO 26262 (Automotive) | Section 5, Table 1A | Enforcement of low complexity |
| IEC 61508 (Industrial) | Annex B. Table B1.7 | 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**

# 複雑性に関するメトリクス

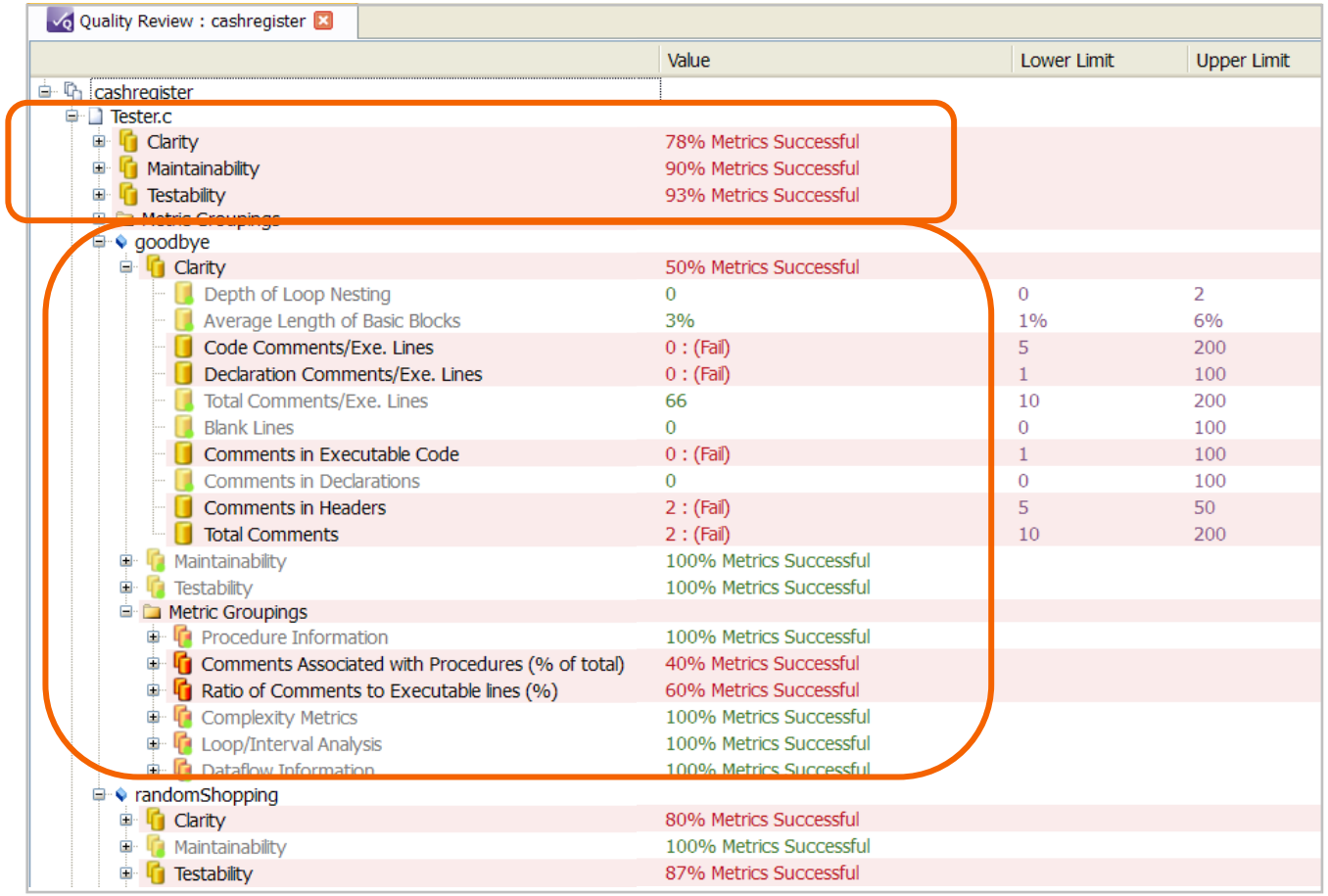
多くのメトリクスがある：

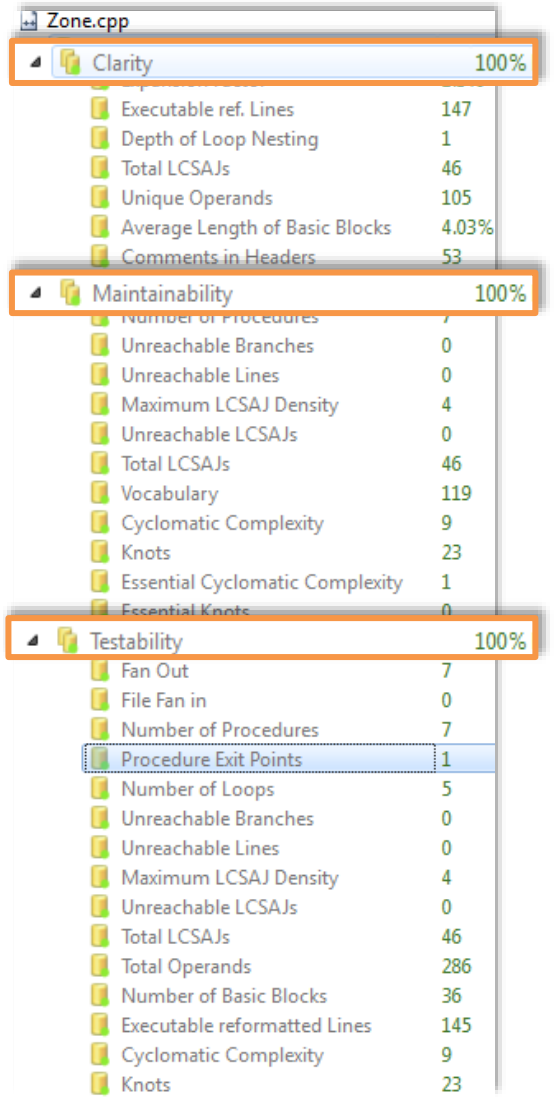
- Number of lines of code
- Number of functions / globals
- Fan in / Fan out
- Depth of loop nesting
- Number of comments / Ratio of comments to code
- Halstead Metrics
- Knots
- Linear Code Sequence And Jumps
- McCabe Cyclomatic Complexity
- Essential Cyclomatic Complexity / Essential Knots



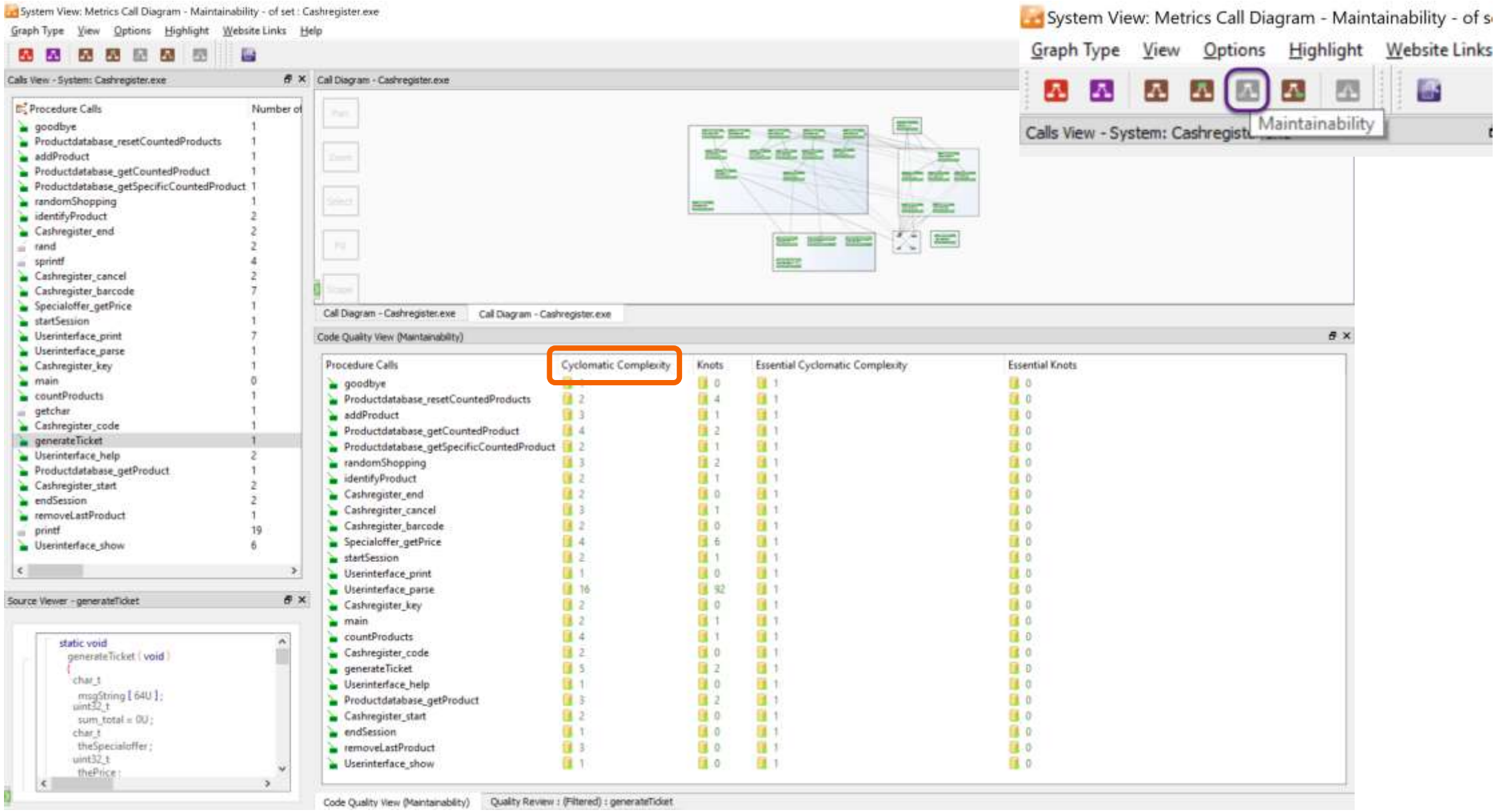
# コードの複雑度と品質のメトリクス

- 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



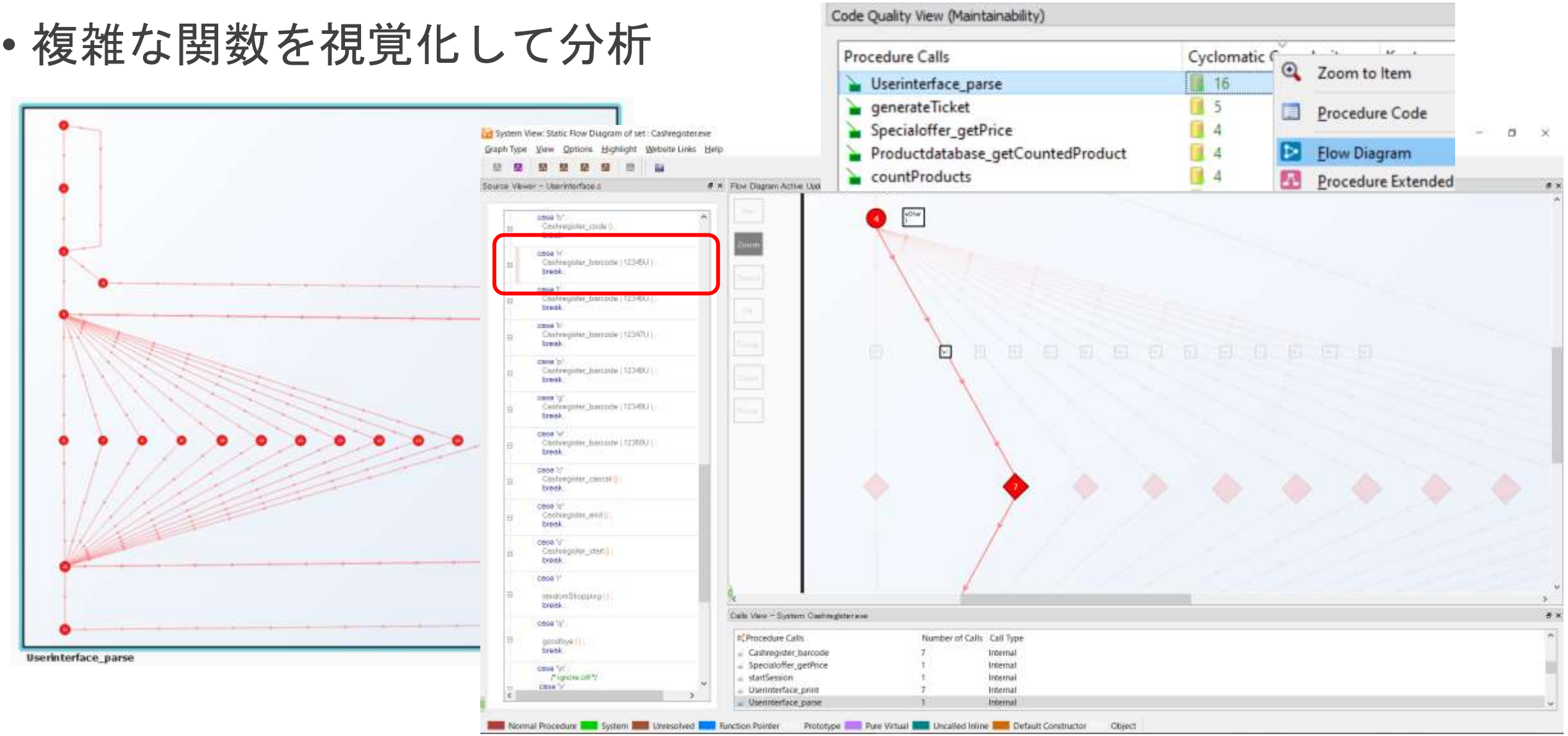



# 例：Maintainability（保守性）で視覚化



# Flow Diagram ブロックや分岐に該当するコードを表示

- 複雑な関数を視覚化して分析

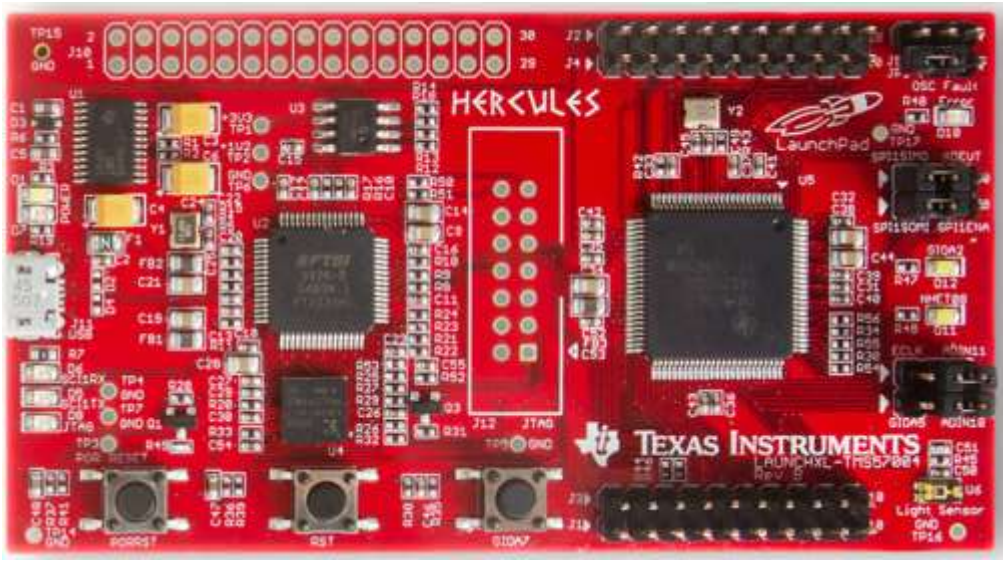


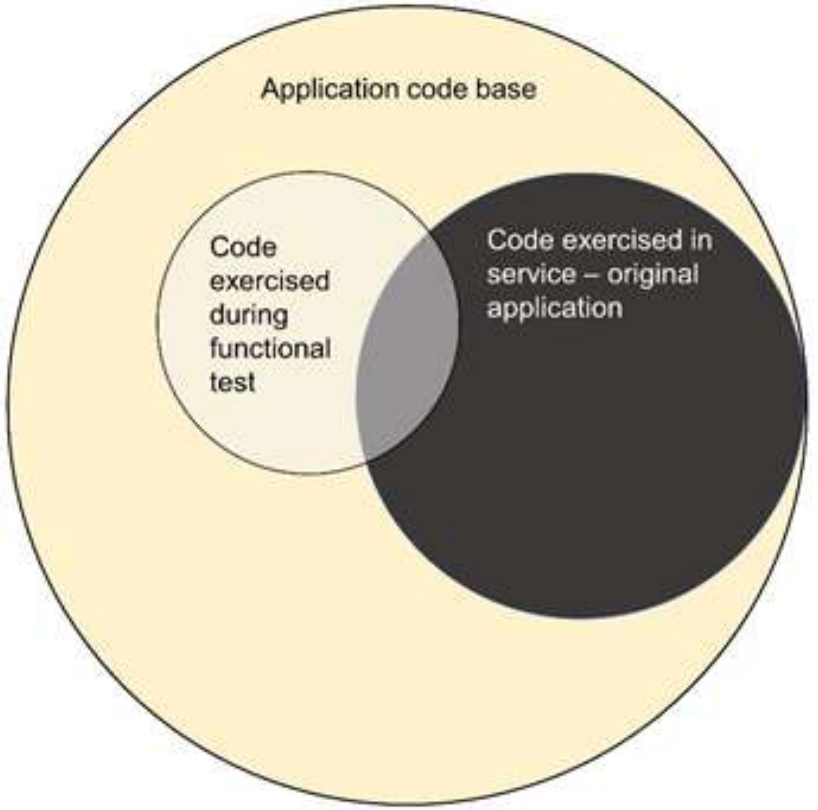
# Agenda

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

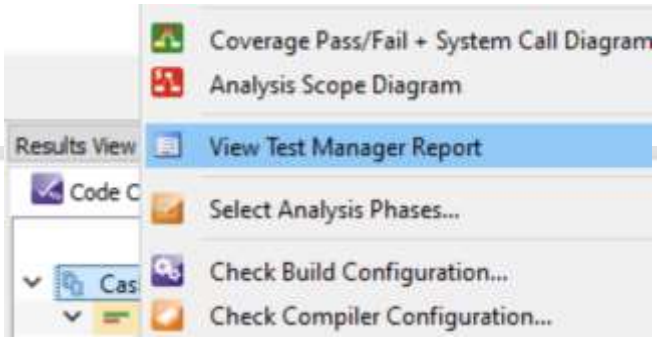
# 動的テスト

- 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イスを評価できる
- 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めない



# カバレッジ解析レポート




ステートメントカバレッジ　分岐カバレッジ　MC/DCカバレッジ

+/-

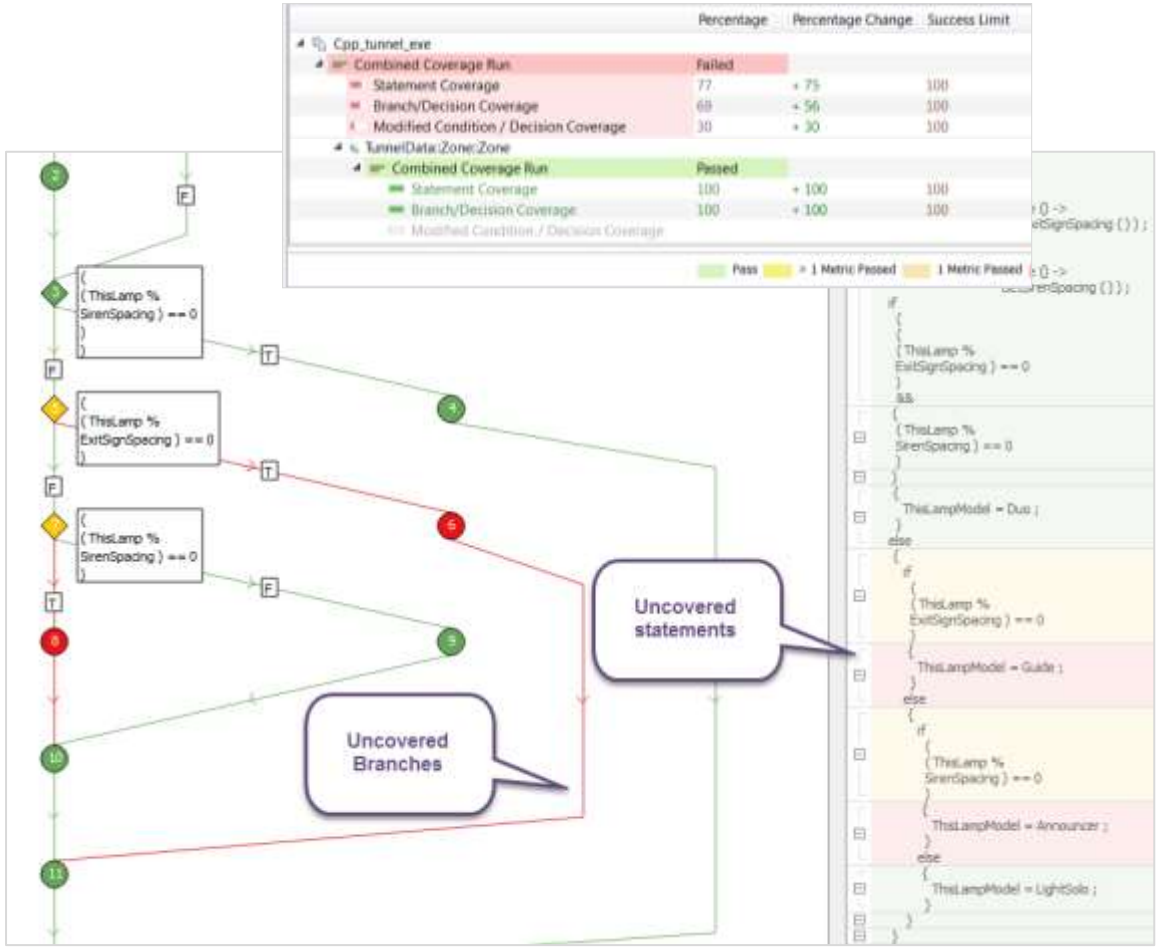
| | Statement (%) | Branch/Decision (%) | MC/DC (%) |
|---|---|---|---|
| Cashregister.exe | 73% | 56% | 100% |
| Cashregister.c | 76% | 53% | na |
| Main.c | 100% | 100% | na |
| Productdatabase.c | 95% | 90% | na |
| Specialoffer.c | 56% | 50% | na |

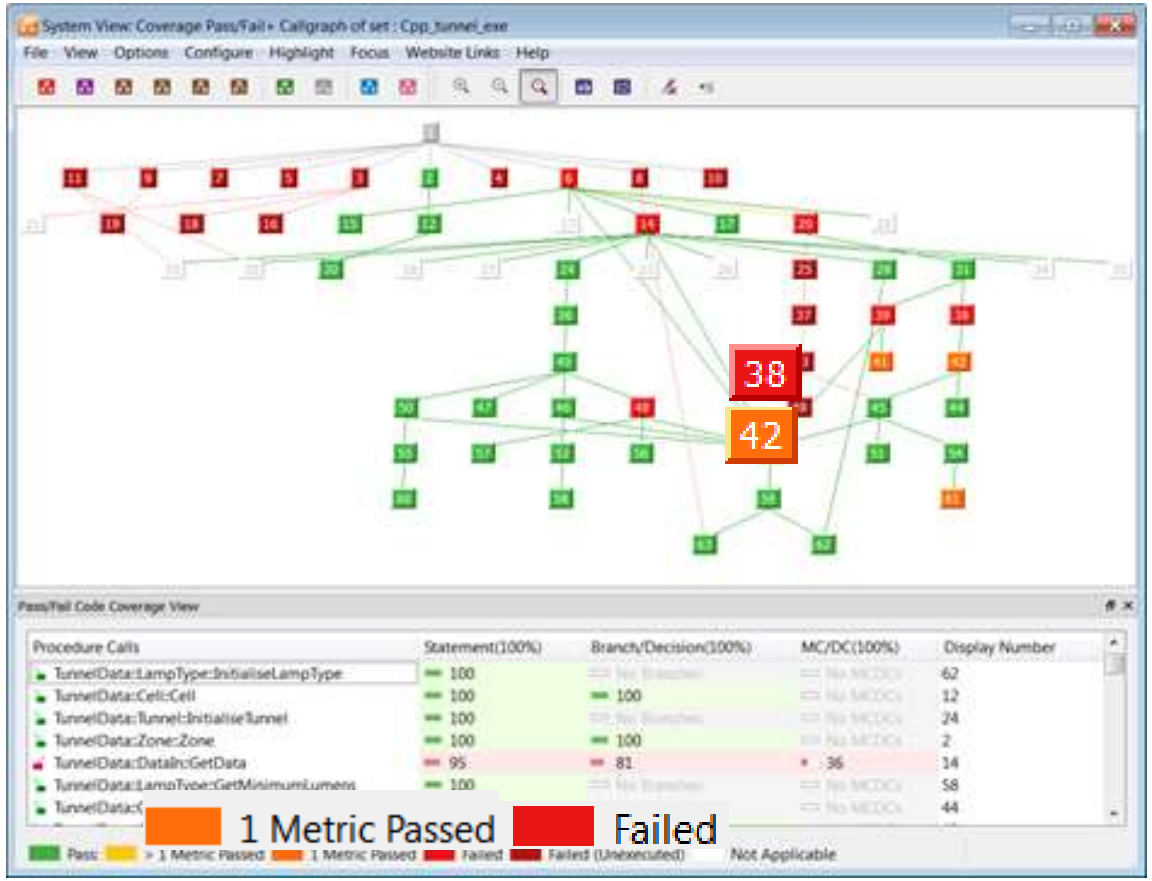
**出荷されて初めて実行されるようなコードを無くし、不要なコードは削除する**

# 構造化カバレッジ：解析民間航空機の場合

| Item | Description | DO-178C Reference | DO-178C Levl 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 |

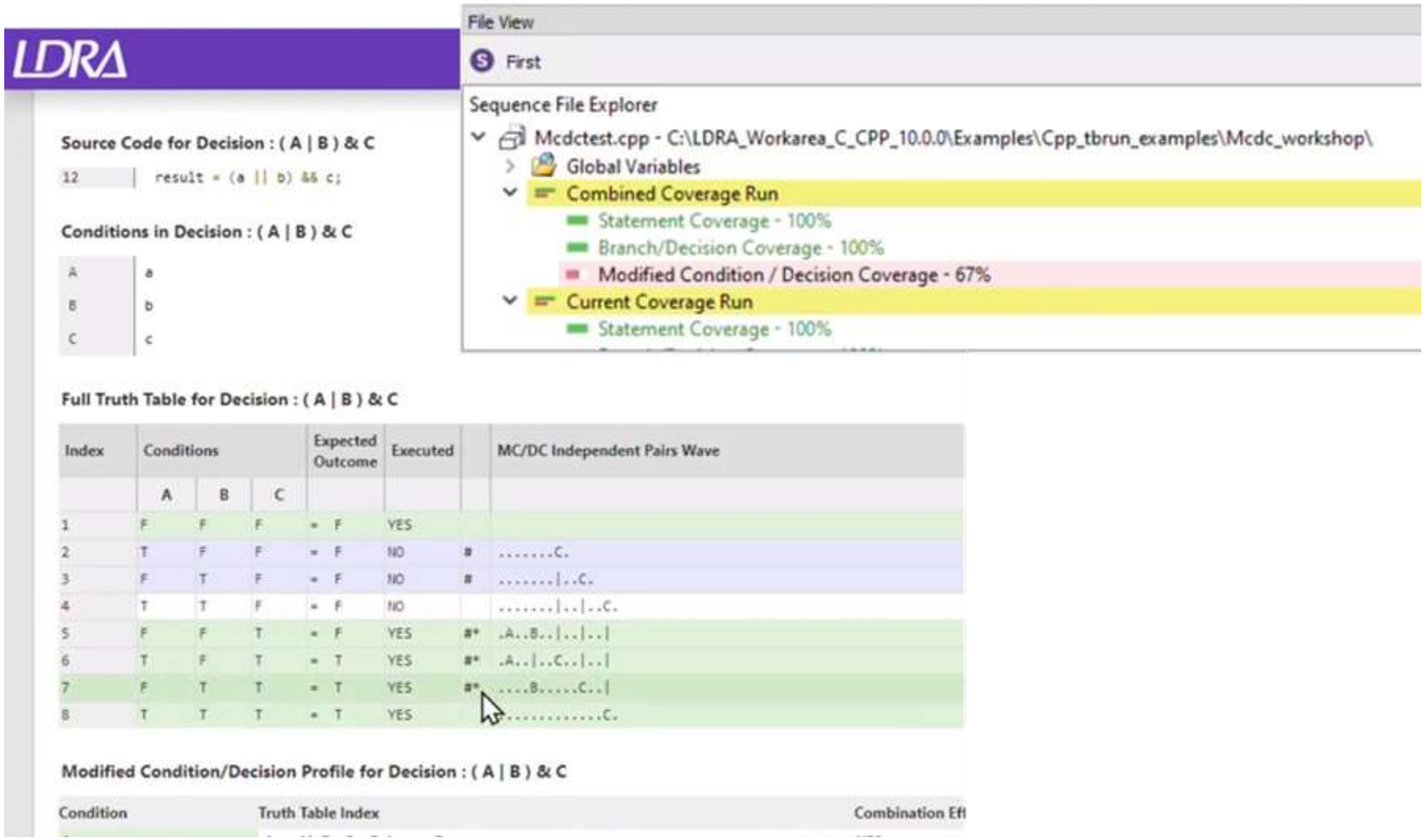
# コールグラフやフローグラフにカバレッジ結果を視覚化




**未実行パスの分岐条件等を分析して、追加のテストケース等の考察に役立てる**

# MCDCカバレッジ


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 A B C Expected Outcome Executed MC/DC Independent Pairs Wave
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%

# コントロールカップリングカバレッジ

## 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 *** |

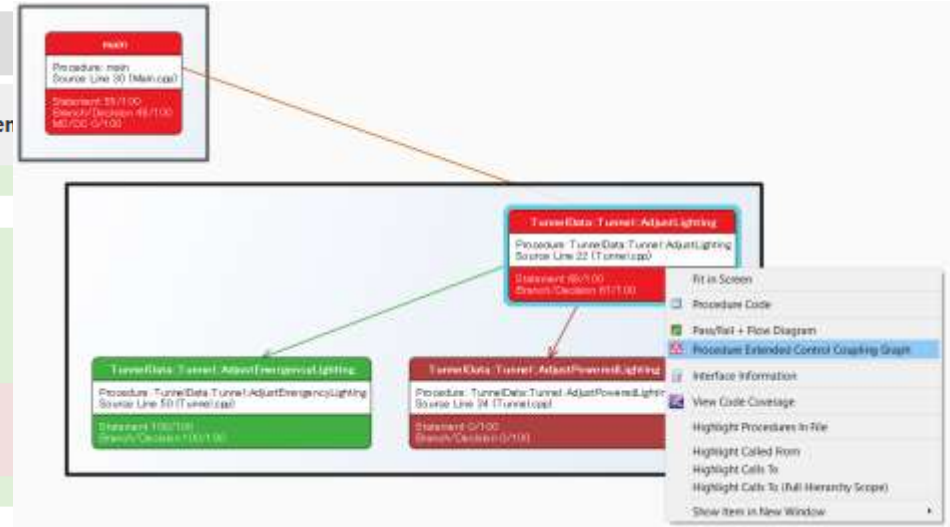
### Control Coupling (Calls In to Procedure)

| 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 | Stater | Coverage |
|---|---|---|
| 22 | `void Tunnel::AdjustLighting (const Float_64 PhotometerDemand, Bool PowerFailure)` | 1 |
| 23 | `{` | - |
| 24 | `if (PowerFailure)` | 1 |
| 25 | `{` | 1 |
| 26 | `AdjustEmergencyLighting();` | 1 |
| 27 | `}` | 1 |
| 28 | `else` | 1 |
| 29 | `{` | 0 *** |
| 30 | `AdjustPoweredLighting(PhotometerDemand);` | 0 *** |
| 31 | `}` | 0 *** |
| 32 | `};` | 1 |



# データカップリングカバレッジ

## 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

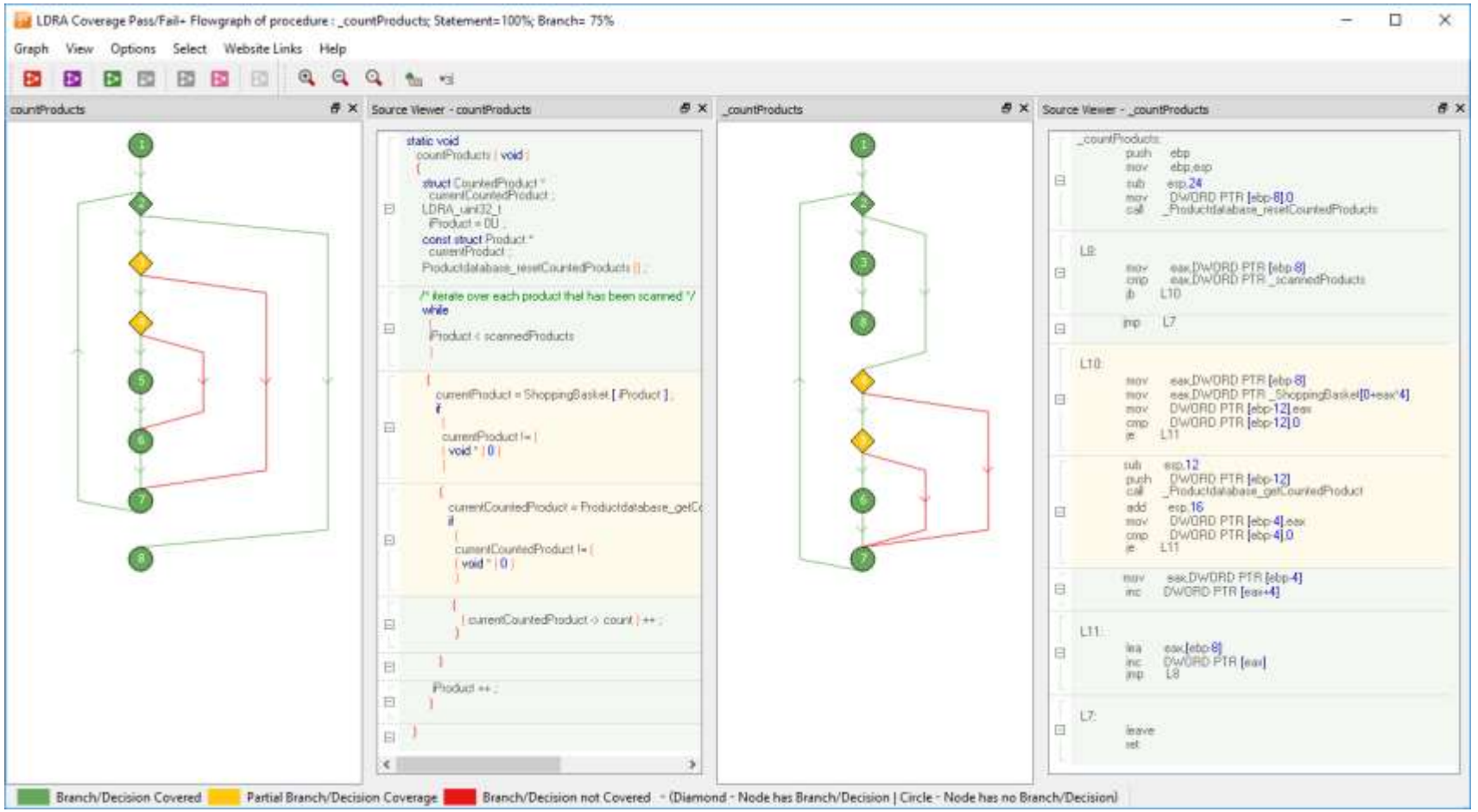
| Variable Name | File | Procedure | Type Code | Attribute Code | Used on lines... |
|---|---|---|---|---|---|
| **PhotometerDemand** | Tunnel.cpp | TunnelData::Tunnel::AdjustLighting | Parameter | Declaration | 22 |
| | | | Parameter | Reference | 30 ***** |
| **PowerFailure** | Tunnel.cpp | TunnelData::Tunnel::AdjustLighting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

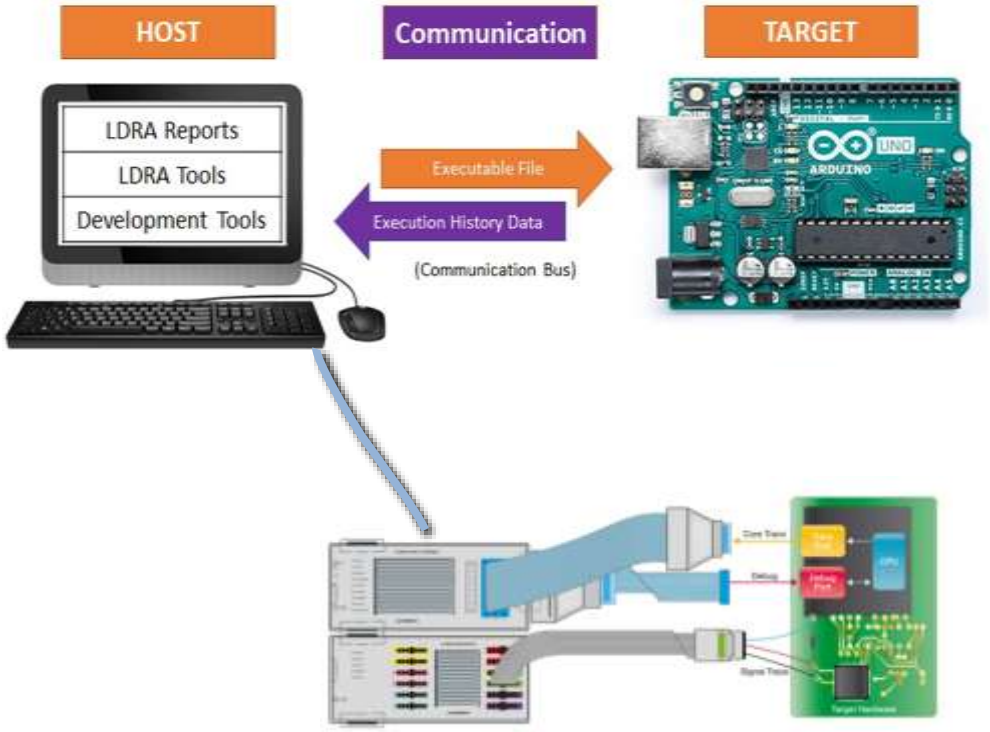
+/-

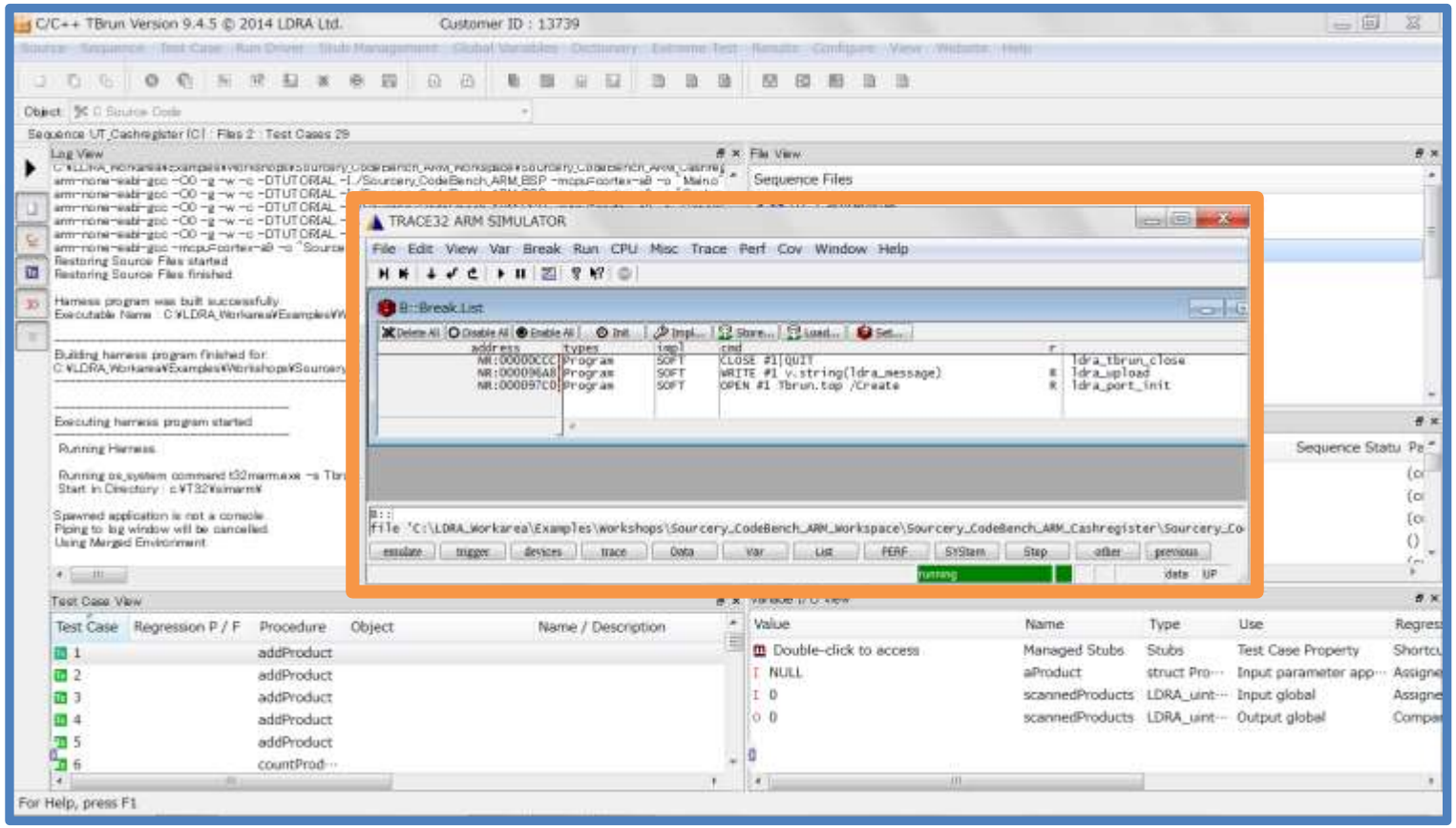
# オブジェクトコードレベルの検証




**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のオブジェクティブを効率的に達成**

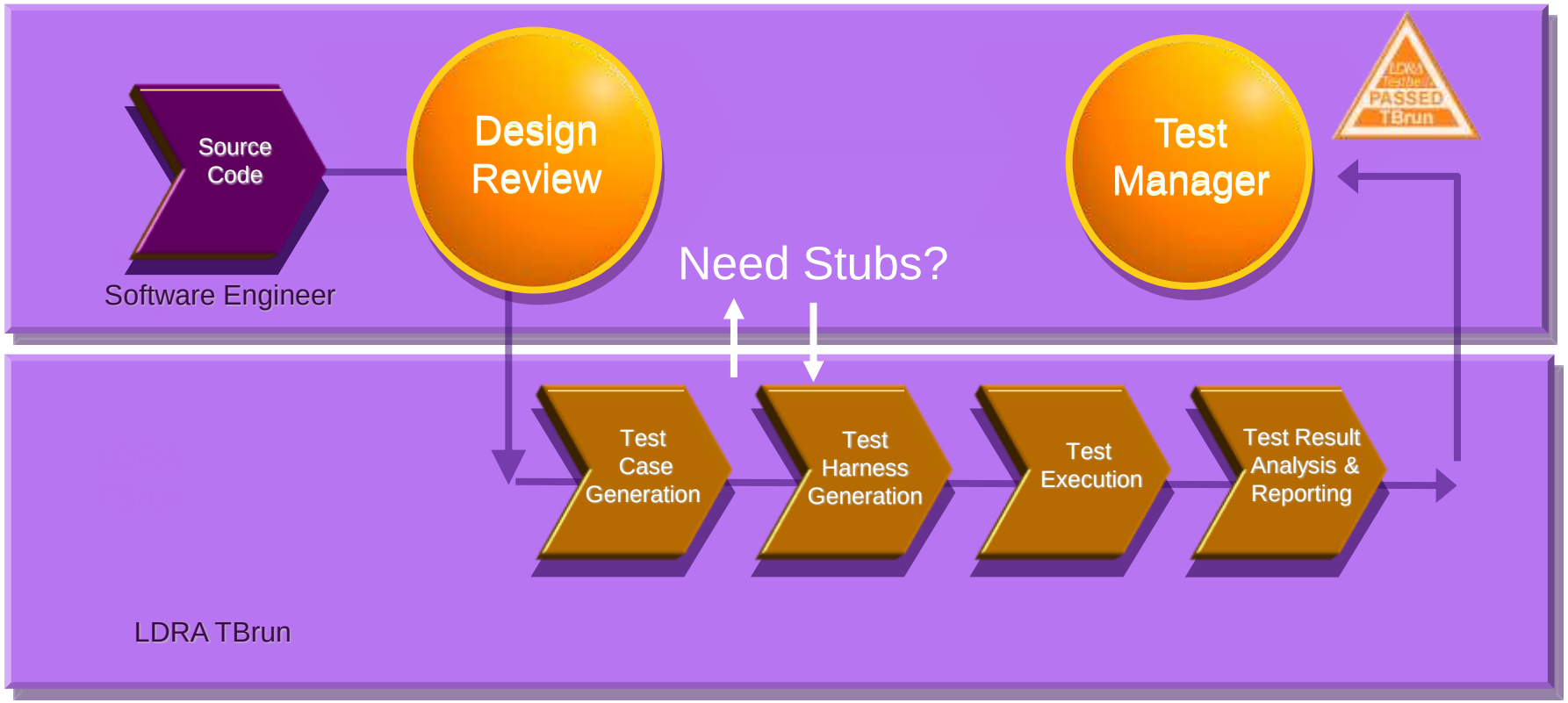
# 単体テスト



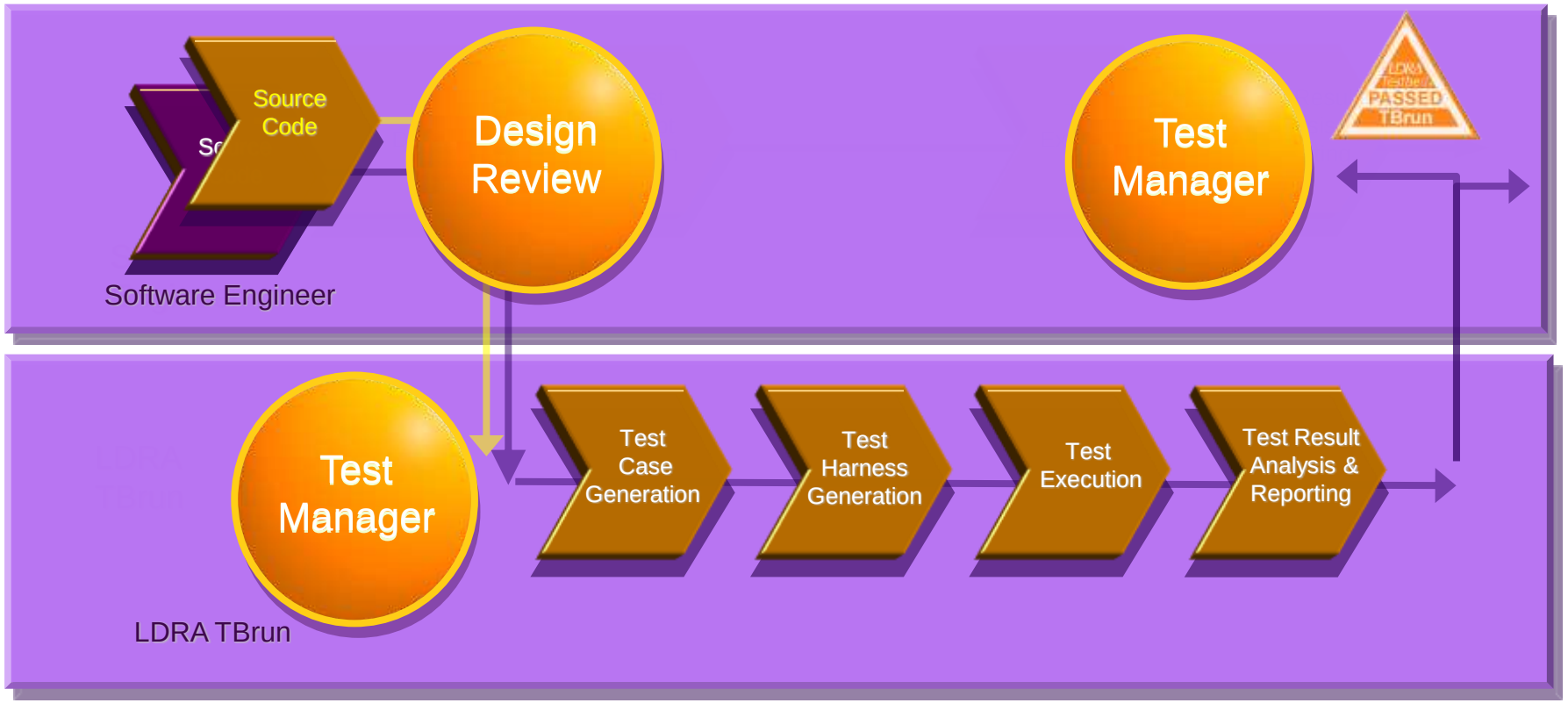

**構造カバレッジオブジェクティブの達成、低レベル要件の検証、**
**検証コストの大幅な削減**

# 単体テスト自動化支援ツール




**静的解析時に得た情報を利用**

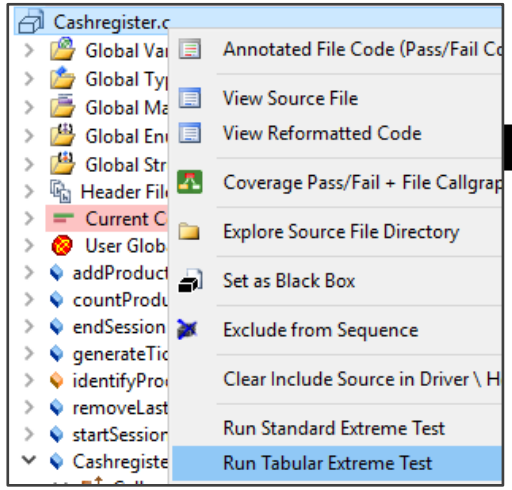
# ソースコードが修正されると

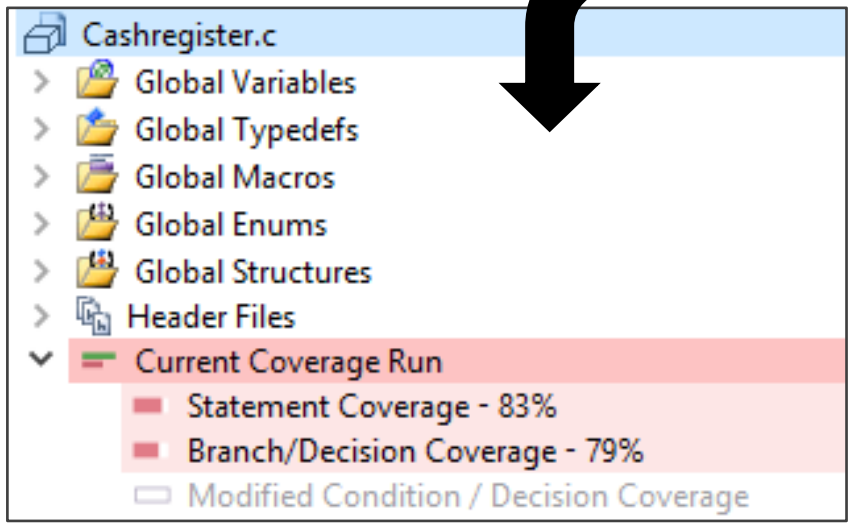



## リグレッションテストを支援

# 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



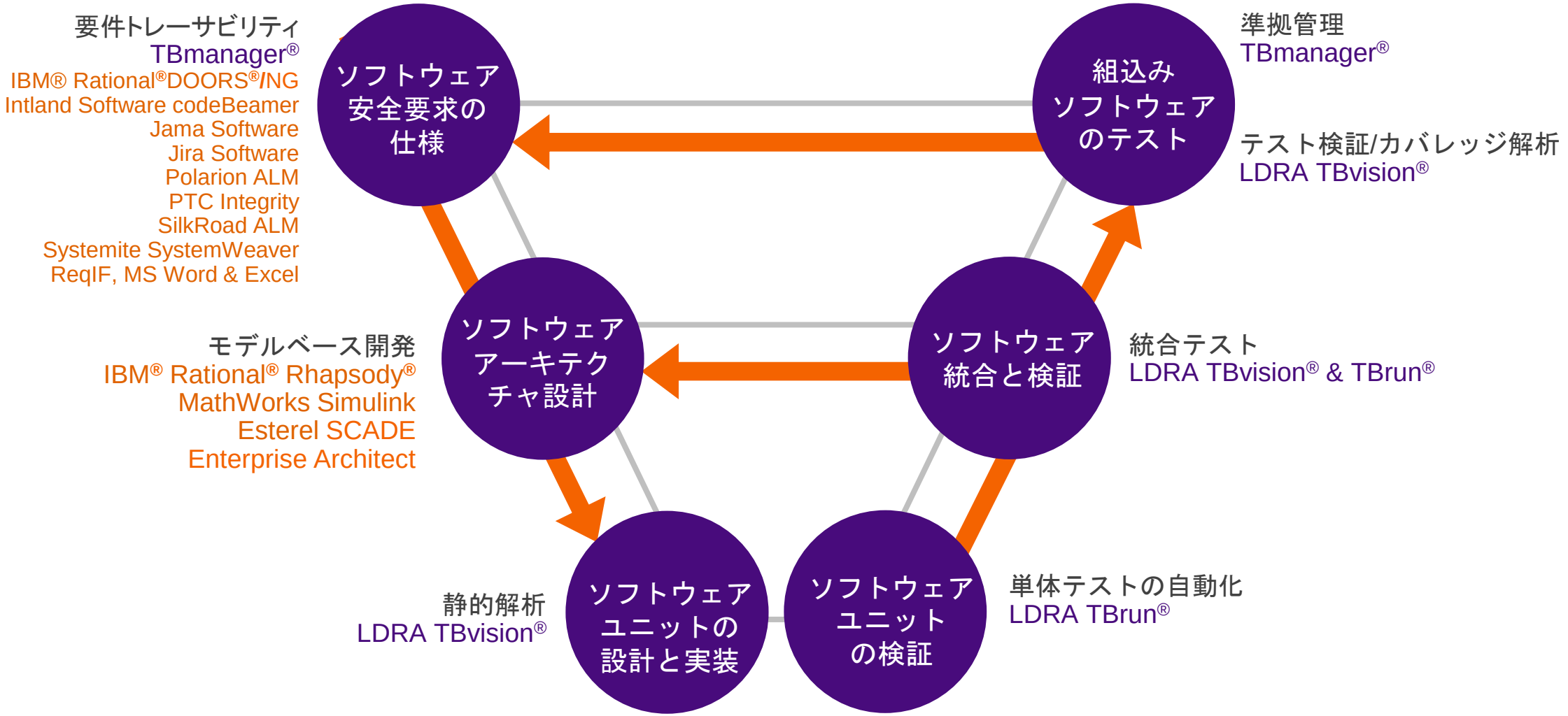

Test Case View

| 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 |

# Agenda

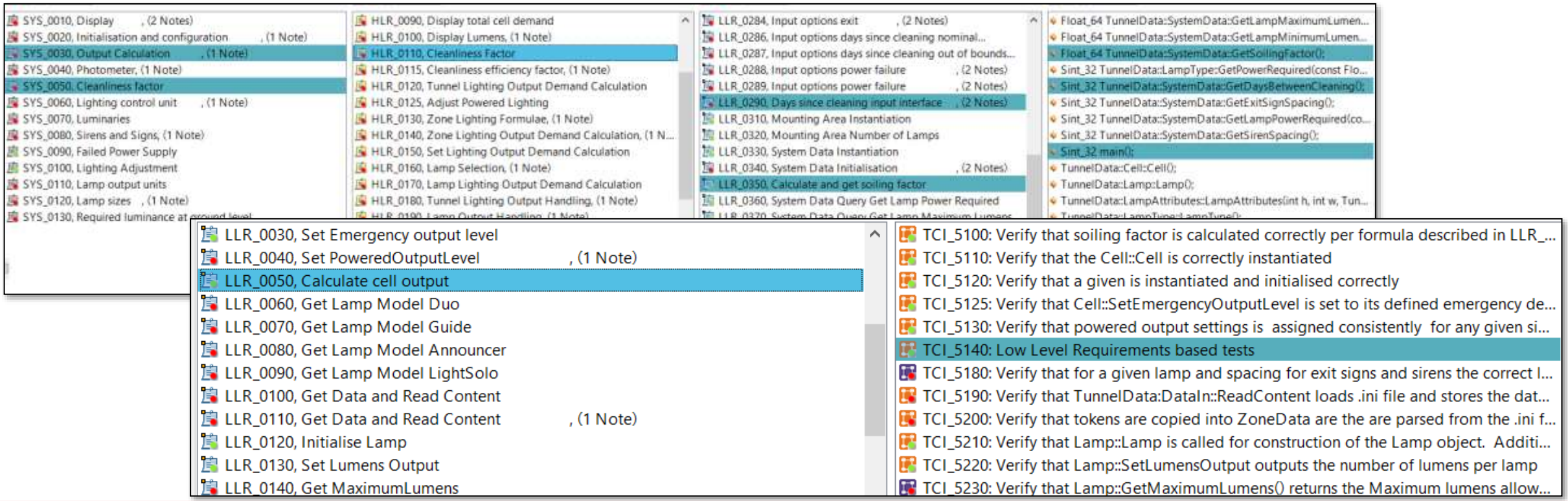
1. 航空業界のベストプラクティス
2. 静的解析、コーディング規約チェックなど
3. 動的テストとカバレッジ解析
4. 要件トレーサビリティ
5. まとめ

# ソフトウェア開発プロセスと自動化ツールチェーン



# TBmanager linking Requirements, Code and Tests

- 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

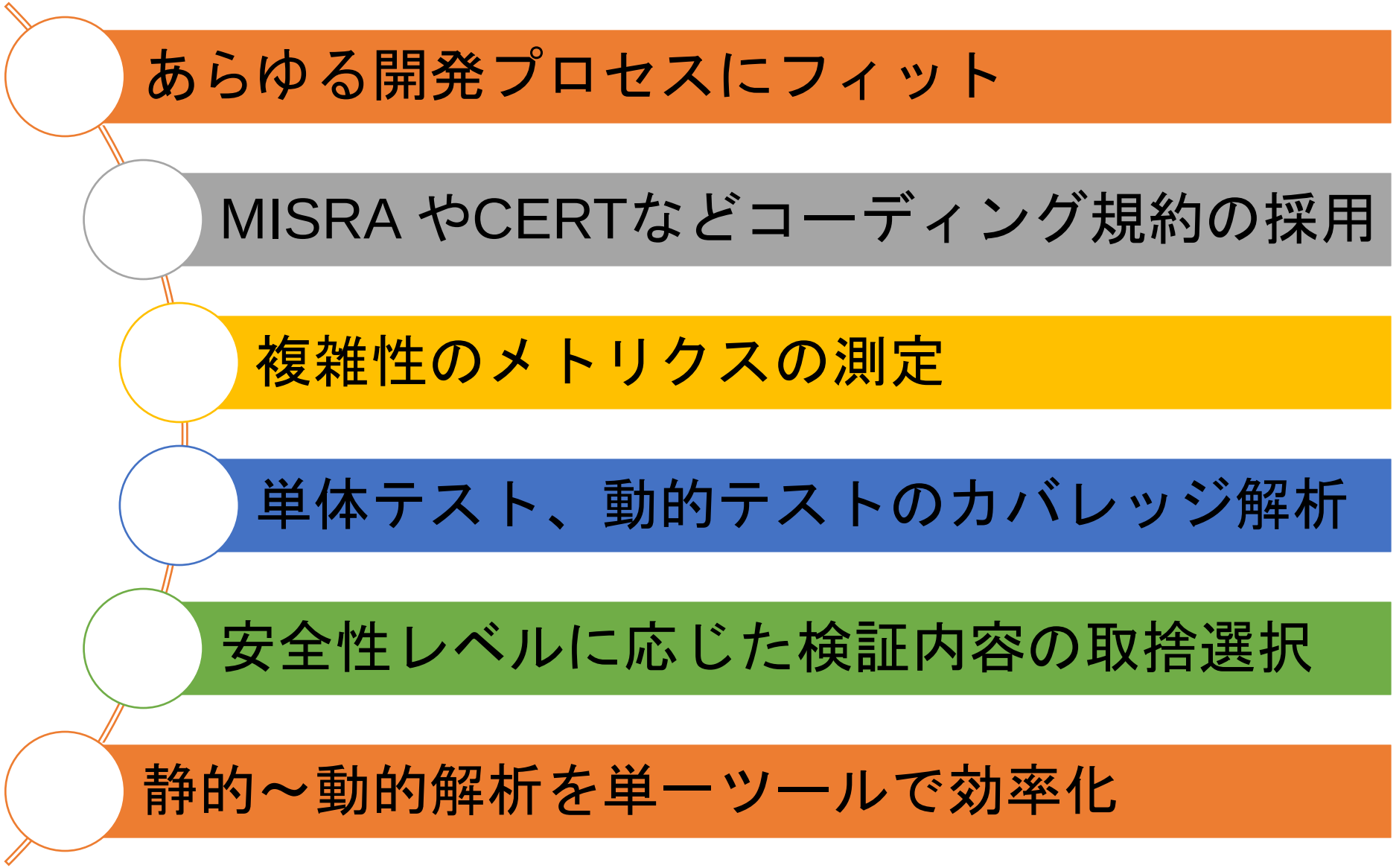



**各スタンダードのオブジェクティブの達成管理機能も備え**
**認証プロセスのリスクとコストを軽減**

# Agenda

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

# 航空業界のベストプラクティスに習う



# Contact Us

Need more information?

FUJI SETSUBI

www.fuji-setsu.co.jp

LDRA.com

LinkedIn

info@ldra.com

FUJI SETSUBI

LDRA社
https://ldra.com/

LDRA スタンダード認証支援テストツール
https://www.fuji-setsu.co.jp/products/LDRA/