



Dealing with SOUP using the LDRA Tool Suite

レガシーコード に代表されるSOUPに対する実践的なテストについて紹介

http://www.fuji-setsu.co.jp/files/LDRA_Dr.Dobb%27s_SOUP.pdf

- **開発経路が未知なソフトウェア**

Software of Unknown (or Uncertain) Pedigree (or Provenance)

再利用 派生、差分開発されるレガシーコード、グラフィックライブラリ、制御系など自動生成されるコード、JavaやAdaのランタイム環境、OS など

- **国際スタンダードを意図して開発されていない**

- **課題：**

SOUP を管理可能な開発のベースとして
ベストプラクティスでシステムを開発すること

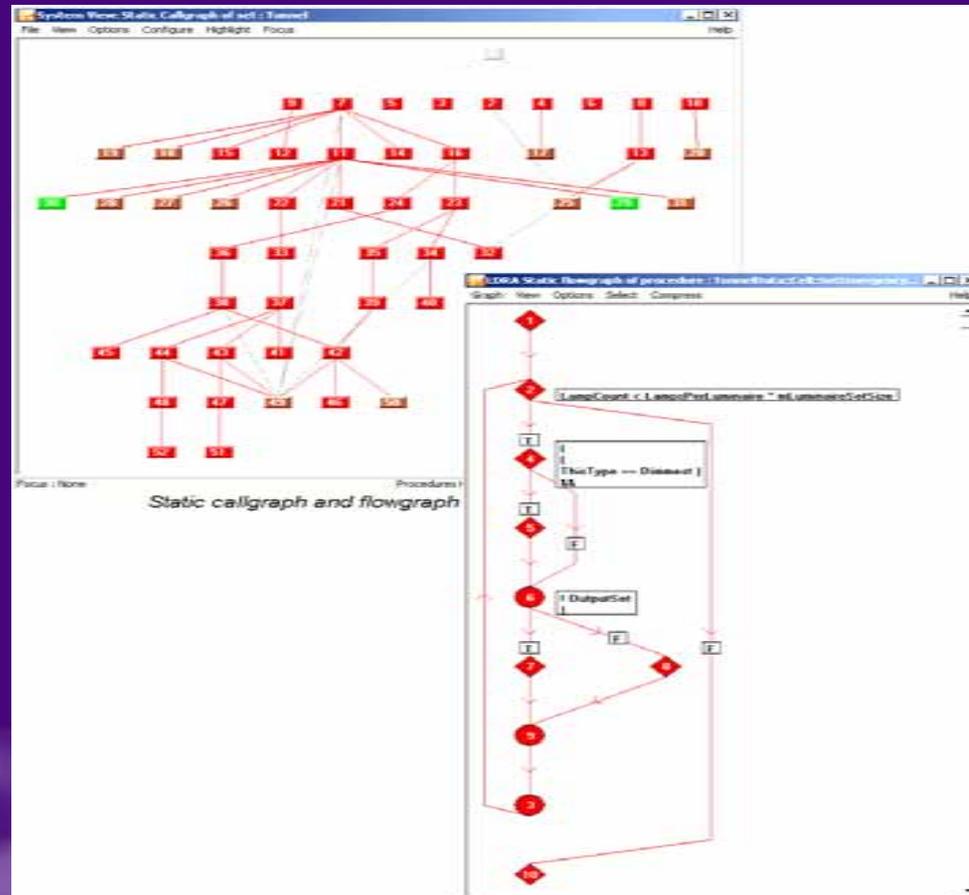
- **さて、どうしましょう、？**



SOUP をベースにするような開発にも活用できるのでしょうか？

- 既存コードを理解すること
- プログラミングスタンダードへの対応
- 機能拡張への対処
- 変更の影響を受ける箇所をテストすること
- 適切なコードカバレッジレベルの確保
- 改訂コードとSOUPが同等であることの証明
- SOUP は要件を満たしているか

コールグラフやフローグラフで 既存コードを視覚的に理解すること



プログラミングスタンダードへの対応には 徐々に社内ルールを進化させること

LDRA Testbed ® Code Review Report

System : Tunnel

Overall Result: FAIL

Report Production	Report Configuration	Analysis phases
<ul style="list-style-type: none"> • C/C++ LDRA Testbed Version: 7.7.1 • Config. File: C:\Testbed\cpp\cppreport.dat • Produced On: Wed Aug 20 2008 at 11:48:39 • Penalty File: C:\Testbed\cpp\cpppen.dat 	<ul style="list-style-type: none"> • Report Level: Summary Report • Procedures Reported: Fails Only • Programming Standards Model: Customer Sample • Line Numbers refer to: Original Source File • Violation Details: Violations and Associated Source • Reporting Scope: Full analysis scope 	<ul style="list-style-type: none"> • Static: Yes • Complexity: Yes • Static Data Flow: Yes • Information Flow: No • Cross Reference: Yes

**TunnelData::Zone::Zone
(16 to 30 Zone.cpp) - FAIL**

Standards Violation Summary

Code	Line	Violation	Standard
M	23	No brackets to loop body.	MISRA-C++:2008 06-03-01
<i>Source: mTheLampTypeID {i}=Brightest;</i>			

Code	Violation	Parameter
C	Defined parameter has possible clear path	this.mTheLampTypeID

Internal standards model can cross reference to international standards

- 機能拡張への対処
- 変更の影響を受ける箇所をテストすること

System View: Combined Dynamic Callgraph of set : Tunnel

File View Options Configure Highlight Focus Help

Commands Help

Procedures

- 1) 'Unknown'
- 2) main
- 3) TunnelData::Tunnel::AdjustLighting
- 4) TunnelData::Tunnel::AdjustPoweredLighting
- 5) TunnelData::Tunnel::AdjustEmergencyLighting
- 6) TunnelData::Zone::CalculateOutputFormula
- 7) TunnelData::Zone::AssignPoweredCellsOutput
- 8) TunnelData::Zone::AssignEmergencyCellsOutput
- 9) TunnelData::Cell::SetEmergencyOutputLevel
- 10) TunnelData::Cell::SetPoweredOutputLevel
- 11) TunnelData::SystemData::GetSoilingFactor
- 12) TunnelData::Cell::CalculateCellOutput
- 13) TunnelData::Lamp::SetLumensOutput
- 14) TunnelData::SystemData::GetEmergencyLampLumens
- 15) TunnelData::Lamp::SendPowerToLamp
- 16) SystemData::Instance
- 17) GetLampPowerRequired

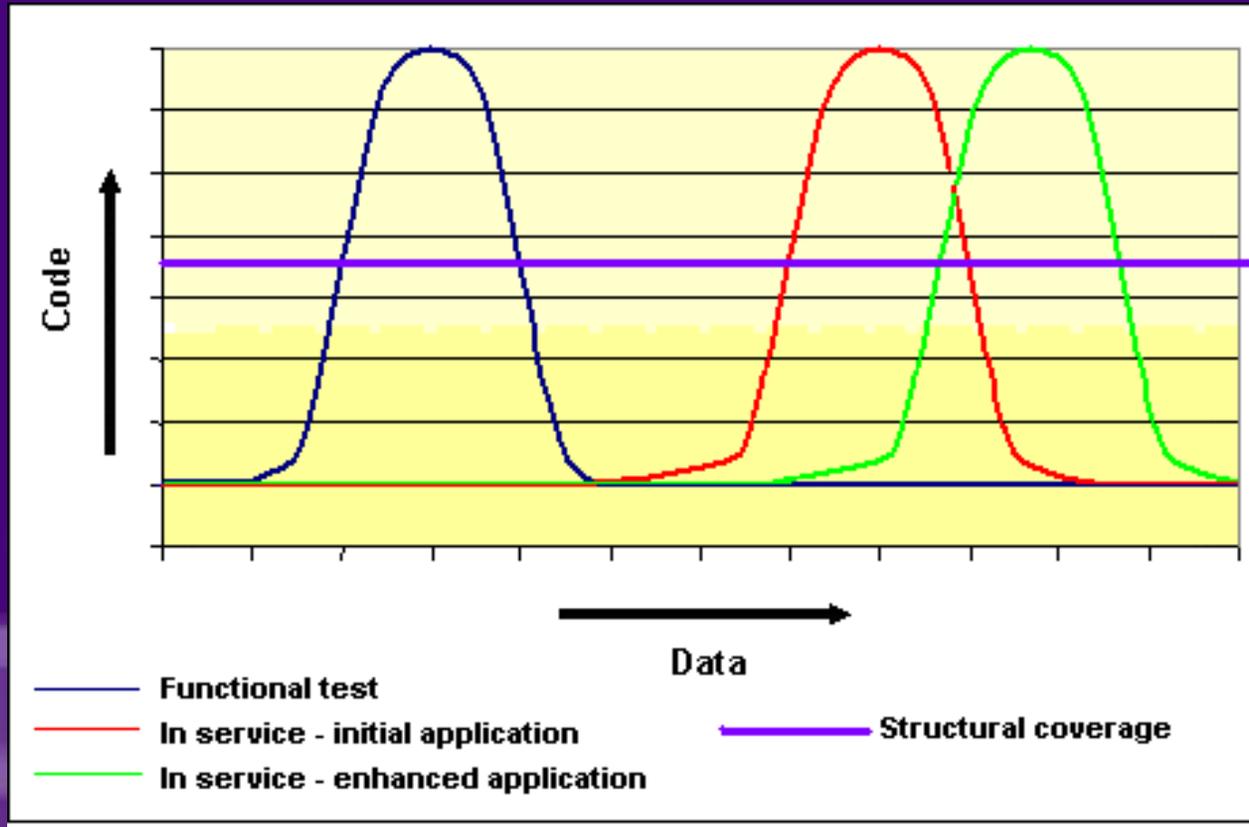
Value	Name	Type	Use	Regression Analysis
I 1000	PhotometerDemand	Float_64	Input parameter applied through local	Assigned
I (Bool)(0)	PowerFailure	Bool	Input parameter applied through local	Assigned

Focus : TunnelData::Tunnel::AdjustLighting; Highlighted : All Descendents All Ancestors

Procedures Highlighted : None

適切なコードカバレッジレベルの確保

構造化カバレッジを活用すれば
入力データ値で偏ることの無いテストを行える



改訂されたコードとSOUPが同等の機能であることの証明

Test Case	Regression P / F	Procedure	Module : Procedure	Value	Name	Type
1	PASS	char_to_fruit	1 : 6	I 127	first_char	char
2	PASS	char_to_fruit	1 : 6	I &ldra_qq_lv_4	name	fruit*
3	PASS	char_to_fruit	1 : 6	◇ *** Suspended ...	name	fruit*
4	PASS	char_to_fruit	1 : 6	◇ *** Suspended ...	%	int
5	----	-----	-----	-----	-----	-----
Test Case	Regression P / F	Procedure	Module : Procedure	Value	Name	Type
1	PASS	char_to_fruit	1 : 6	I -128	first_char	char
2	PASS	char_to_fruit	1 : 6	I &ldra_qq_lv_4	name	fruit*
3	PASS	char_to_fruit	1 : 6	◇ *** Suspended ...	name	fruit*
4	PASS	char_to_fruit	1 : 6	◇ *** Suspended ...	%	int
5	----	-----	-----	-----	-----	-----
Test Case	Regression P / F	Procedure	Module : Procedure	Value	Name	Type
1	PASS	char_to_fruit	1 : 6	I 0	first_char	char
2	PASS	char_to_fruit	1 : 6	I &ldra_qq_v_4	name	fruit*
3	PASS	char_to_fruit	1 : 6	◇ *** Suspended ...	name	fruit*
4	PASS	char_to_fruit	1 : 6	◇ *** Suspended ...	%	int
5	PASS	char_to_fruit	1 : 6			
6	PASS	char_to_fruit	1 : 6			
7	PASS	char_to_fruit	1 : 6			
8	PASS	char_to_fruit	1 : 6			
9	PASS	char_to_fruit	1 : 6			
10	PASS	char_to_fruit	1 : 6			
11	PASS	char_to_fruit	1 : 6			
12	PASS	char_to_fruit	1 : 6			
13	PASS	char_to_fruit	1 : 6			

SOUP はどの要件を満たすべきなのか

要件からコード・検証結果に至るトレーサビリティ

The screenshot displays the LDRA Verification Plan tool interface, illustrating the traceability from requirements to code and test results. The interface is divided into several panes:

- Lighting Control:** A tree view showing the project structure, including requirements like REQ_0001 through REQ_0015 and REQ_0016.
- Software Requirements:** A list of requirements, with REQ_0016 (Thread) highlighted in blue.
- SVP - LDRA Verification Plan:** A tree view showing the verification plan, including sub-projects like Tunnel ID - Unit Test and Tunnel ID - System Test. The System Test for REQ_0016 is highlighted in blue.
- SVR - LDRA Verification Report:** A tree view showing the verification report, including test cases like TC1, TC2, and TC3, and source files like Systemdata.cpp, Lamptype.cpp, and Tunnel.cpp. The System Test for REQ_0016 (Tunnel ID - System Test) is highlighted in blue.

Blue lines connect the highlighted requirement (REQ_0016) in the Software Requirements pane to the corresponding System Test in the SVP pane, and then to the System Test in the SVR pane, demonstrating the traceability from requirements to code and test results.

At the bottom, the Messages pane shows a message: "Requirement REQ_0016 - Installation and Configuration' is covered".

SOUP で要件は満たされるか？

This document has been generated by TBreq

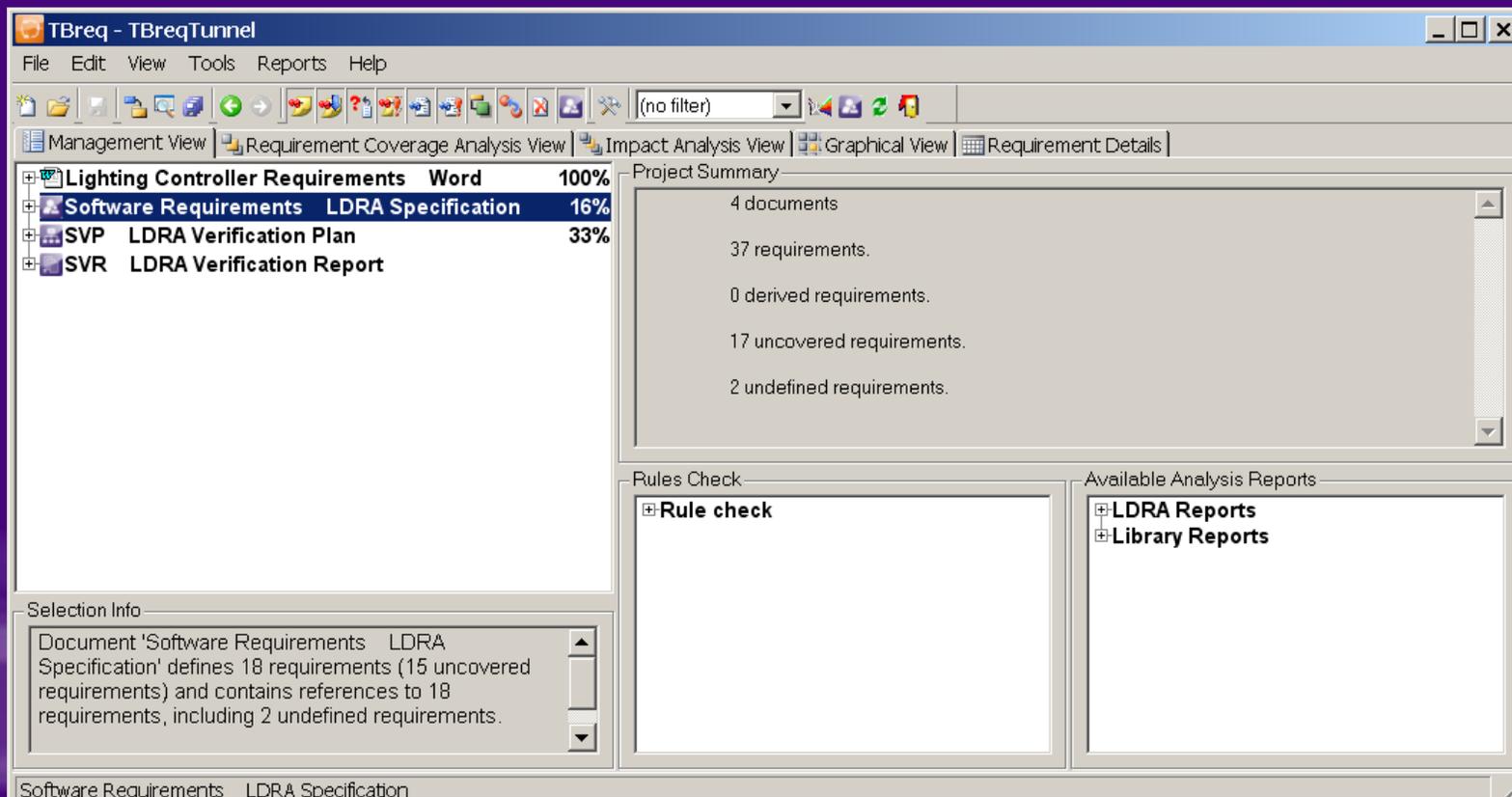
Software Verification Report

1. Table of verification requirements

This table is based on the "SVR" document of your TBreq project (type: "LDRA Verification Report").

Verification Task	Verification Group	Project Manager Status	Defect Report ID	Tester	Test Case Identifier	"Source File Name" (Procedure)
System Test for REQ_0016 (Tunnel I/O - System Test)	Tunnel I/O - System Test	Passed		BillCody (Developer)	REQ_001 6_TCI	- "Datain.cpp" (GetData)- "Systemdata.cpp" (InitialiseParams)- "Tunnel.cpp" (InitialiseTunnel)
Unit Test for LL_1_u001 (Tunnel I/O - Unit Test)	Tunnel I/O - Unit Test	Failed	UT_187311371_1	BillCody (Developer)	LL_1_TC1 LL_1_TC2 LL_1_TC3	- "Systemdata.cpp" (GetSoilingFactor)
Code Review for LL_2_u001 (Tunnel Power Failure - Code Review)	Tunnel Power Failure - Code Review	Failed	CR_1698920814_1	BillCody (Developer)	TCI	- "Lamptype.cpp" (InitialiseLampType)

- ハイレベル、ローレベル、派生要件を含めて
- システムの全ての機能が検証されることを裏付ける



The screenshot displays the TBreq - TBreqTunnel application window. The interface includes a menu bar (File, Edit, View, Tools, Reports, Help), a toolbar, and a main workspace divided into several panes. On the left, a tree view shows a project structure with the following items and their coverage percentages:

Item	Coverage
Lighting Controller Requirements Word	100%
Software Requirements LDRA Specification	16%
SVP LDRA Verification Plan	33%
SVR LDRA Verification Report	

The central pane, titled 'Project Summary', provides a detailed overview of the requirements analysis:

- 4 documents
- 37 requirements.
- 0 derived requirements.
- 17 uncovered requirements.
- 2 undefined requirements.

At the bottom left, the 'Selection Info' pane displays details for the selected document:

Document 'Software Requirements LDRA Specification' defines 18 requirements (15 uncovered requirements) and contains references to 18 requirements, including 2 undefined requirements.

At the bottom right, the 'Available Analysis Reports' pane lists:

- LDRA Reports
- Library Reports

The status bar at the bottom of the window indicates the current document: 'Software Requirements LDRA Specification'.