

欲しくなるテストツールの の機能とは



FUJI SETSUBI



***LDRA/テスト自動化ツール**
静的解析、カバレッジ、単体テスト



***T-VEC/モデル検証・テスト自動化**
要求仕様～デザインモデル、実コードまで



***MetaEdit+**
Domain Specific Modeling 開発環境



***Ashling/Jtagデバッガ**
ARC, SmartCards等各種



***ロータバッハ/Jtagデバッガ**
ARM, PowerPC, 68K, MIPS等各種



***XJTAG**
バウンダリスキャン・HWテストツール

- ・ソースコードベース

- ・モデル検証

- ・自動コード生成
(Domain-Specific Modeling)

・ソースコードベース

・モデル検証

・自動コード生成
(Domain-Specific Modeling)



高信頼性向け
30年以上の実績



静的テスト



静的テスト
単体テスト

全
融
て
合



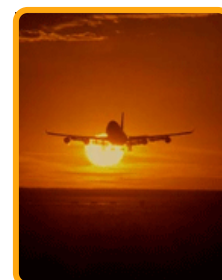
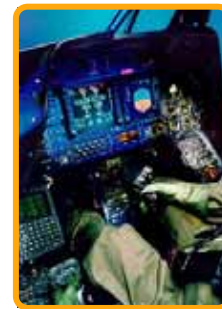
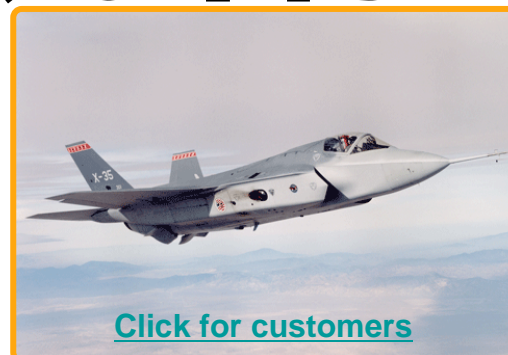
静的テスト

単体テスト

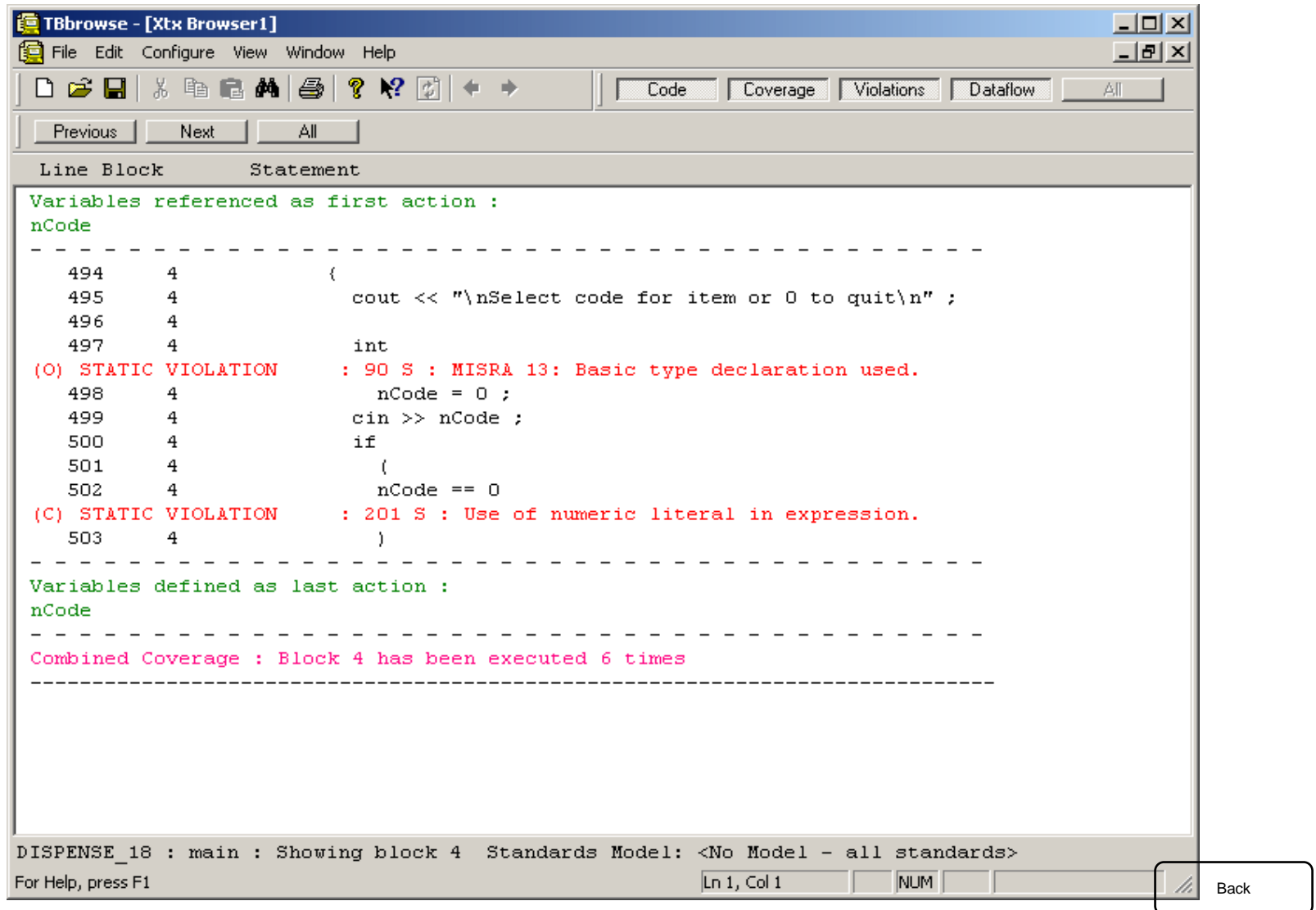
テストカバレッジ

航空、防衛



MISRA 委員会



The screenshot shows a software interface for analyzing code. The main window displays a code block with two MISRA violations highlighted in red. The first violation is at line 497, where a basic type declaration is used. The second violation is at line 502, where a numeric literal is used in an expression. The interface also shows variables referenced and defined, and a combined coverage report indicating the block was executed 6 times.

```
Variables referenced as first action :
nCode
-----
494      4      {
495      4      cout << "\nSelect code for item or 0 to quit\n" ;
496      4
497      4      int
(O) STATIC VIOLATION      : 90 S : MISRA 13: Basic type declaration used.
498      4      nCode = 0 ;
499      4      cin >> nCode ;
500      4      if
501      4      {
502      4      nCode == 0
(C) STATIC VIOLATION      : 201 S : Use of numeric literal in expression.
503      4      }
-----
Variables defined as last action :
nCode
-----
Combined Coverage : Block 4 has been executed 6 times
-----
```

DISPENSE_18 : main : Showing block 4 Standards Model: <No Model - all standards>
For Help, press F1

Ln 1, Col 1 NUM

Back

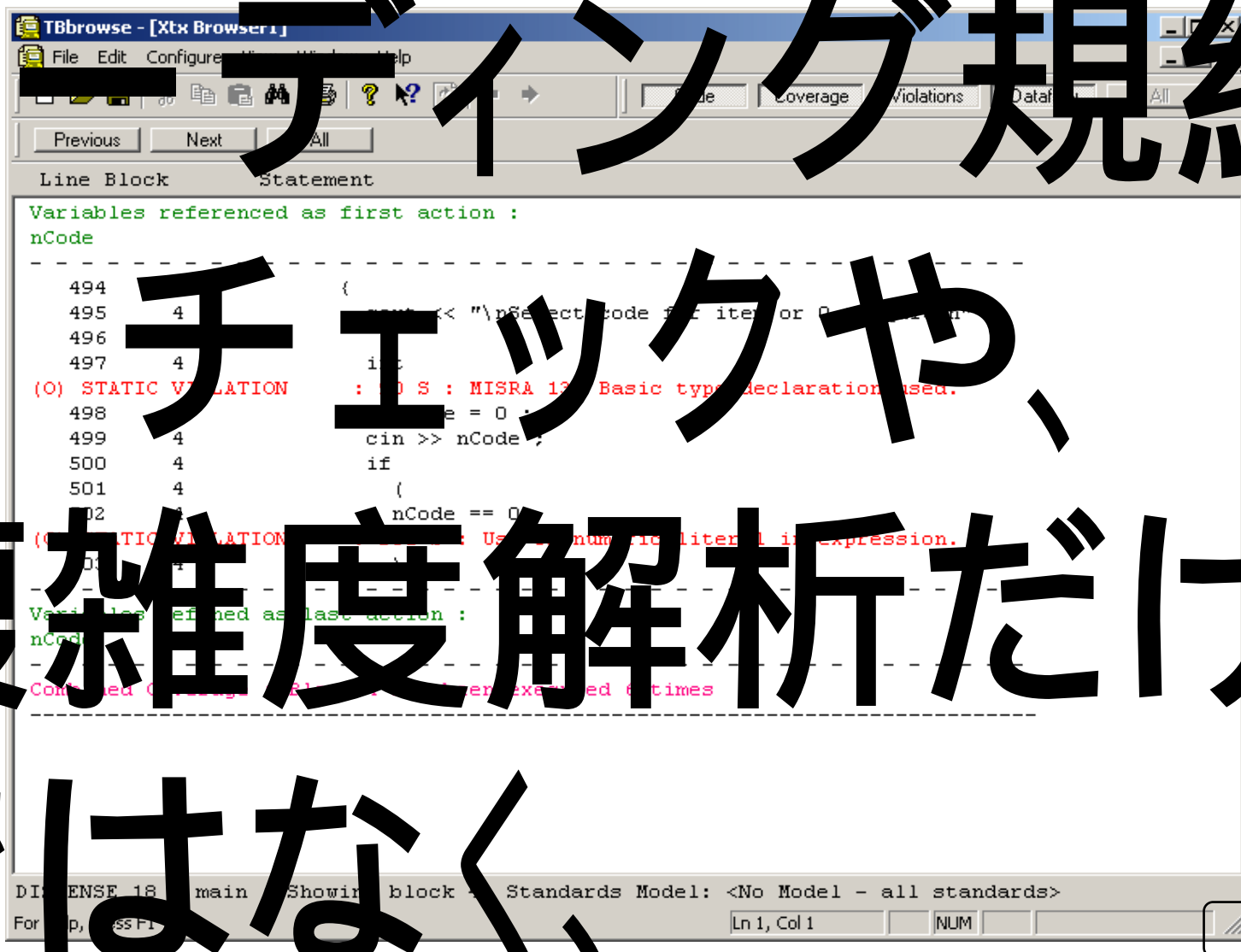


の静的テストは

、

MISRA など

コーディング規約



The screenshot shows a code analysis tool window titled 'TBrowse - [XtX Browser]'. The main content area displays C code with several MISRA violations highlighted in red. The violations are:

- Line 498: (O) STATIC VIOLATION : O S : MISRA 12: Basic type declaration used.
- Line 502: (O) STATIC VIOLATION : Use of numeric literal in expression.

The code snippet is as follows:

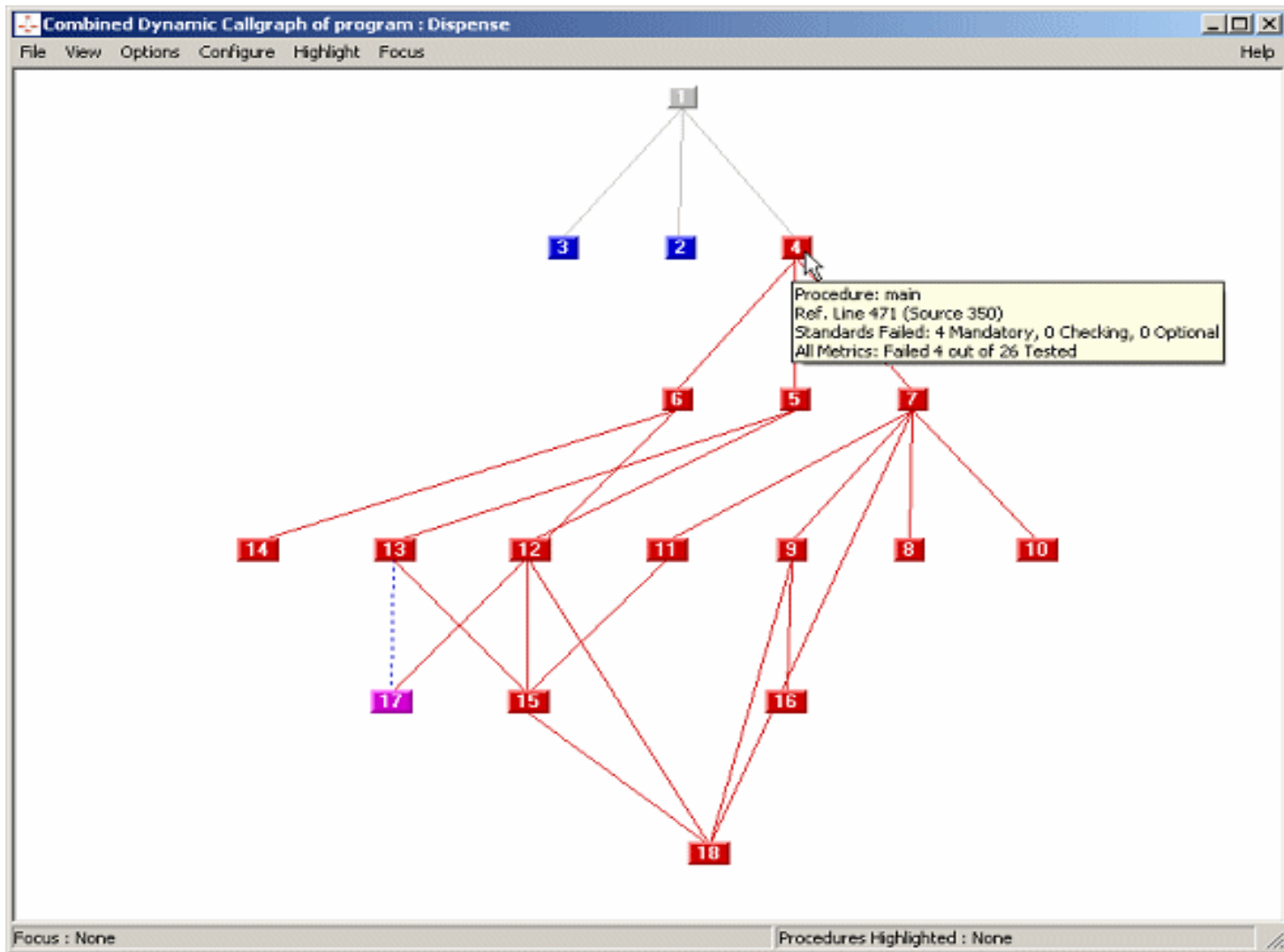
```
494     {  
495     4     printf("< "\nSelect code for item or P...  
496  
497     4     if  
(O) STATIC VIOLATION : O S : MISRA 12: Basic type declaration used.  
498         e = 0 ;  
499     4     cin >> nCode ;  
500     4     if  
501     4     (  
502     4     nCode == 0  
(O) STATIC VIOLATION : Use of numeric literal in expression.  
503     4     }  
-----  
Variables defined as last action :  
nCode  
-----  
Control Coverage : P1 : never exercised 6 times
```

At the bottom of the window, the status bar shows 'DI DIMENSION 18 main Showing block Standards Model: <No Model - all standards>' and 'For op, SSPI'. A 'Back' button is visible in the bottom right corner.

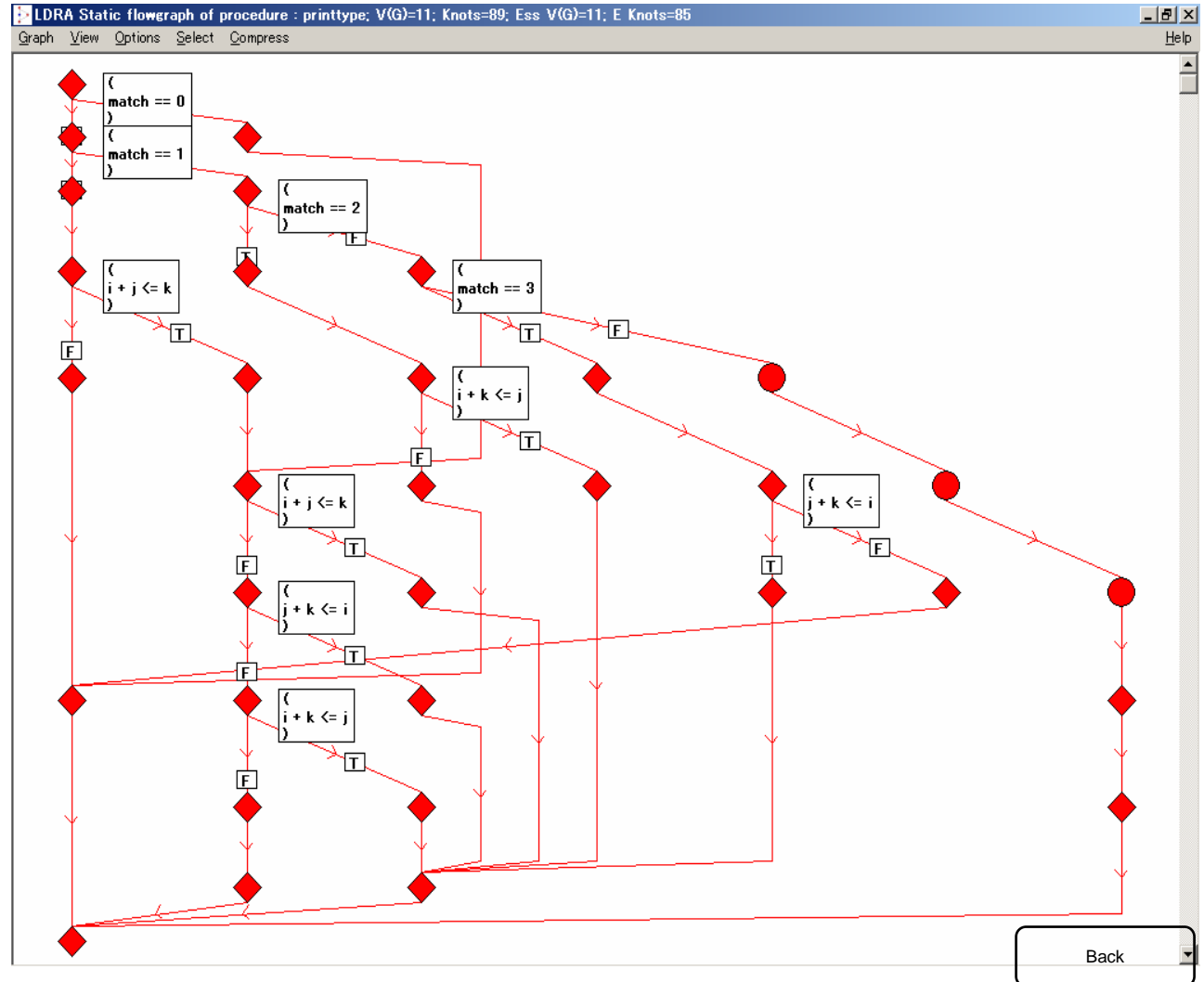
チェックや、 複雑度解析だけ ではなく、

視覚的に

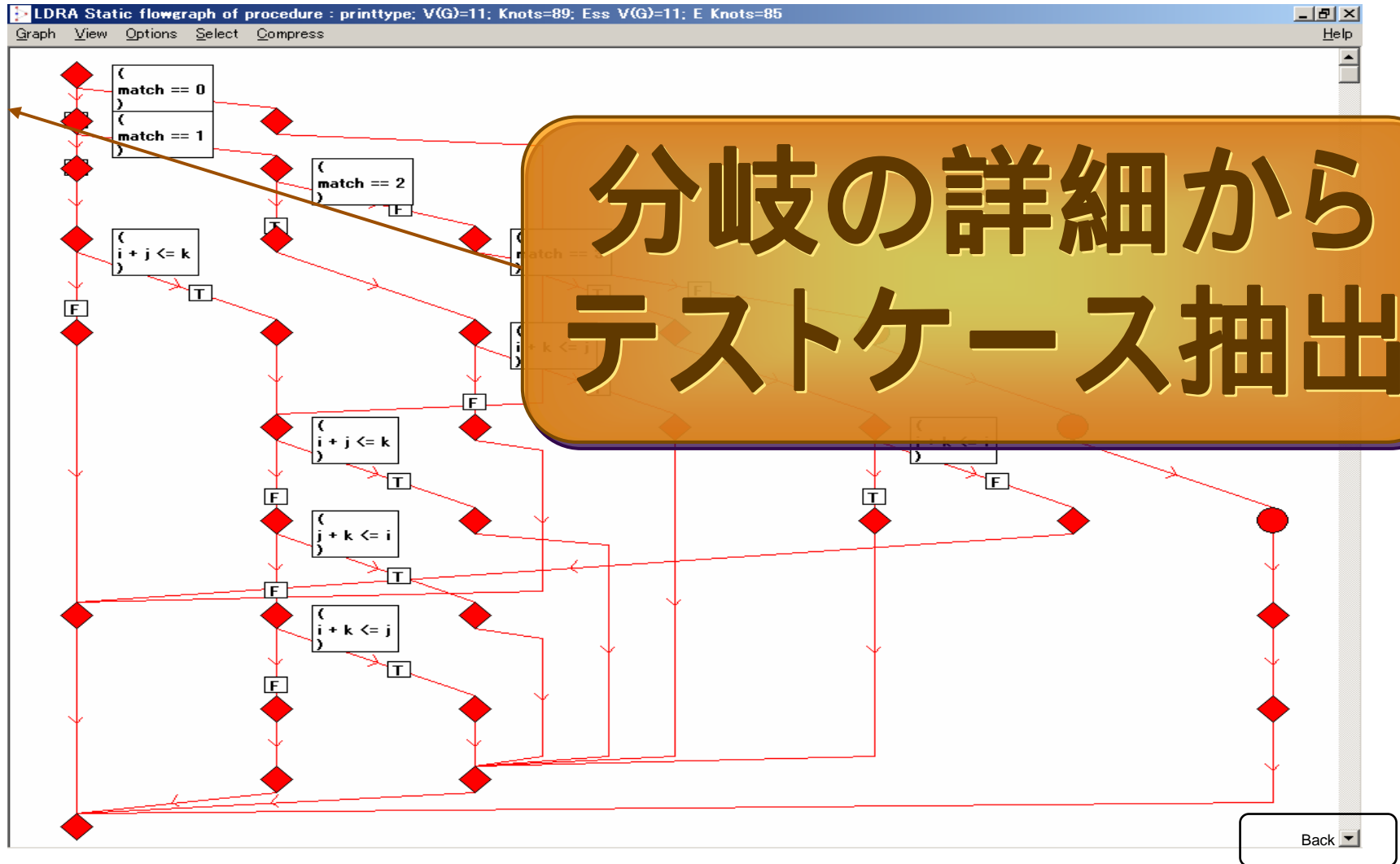
関数コールヤ、



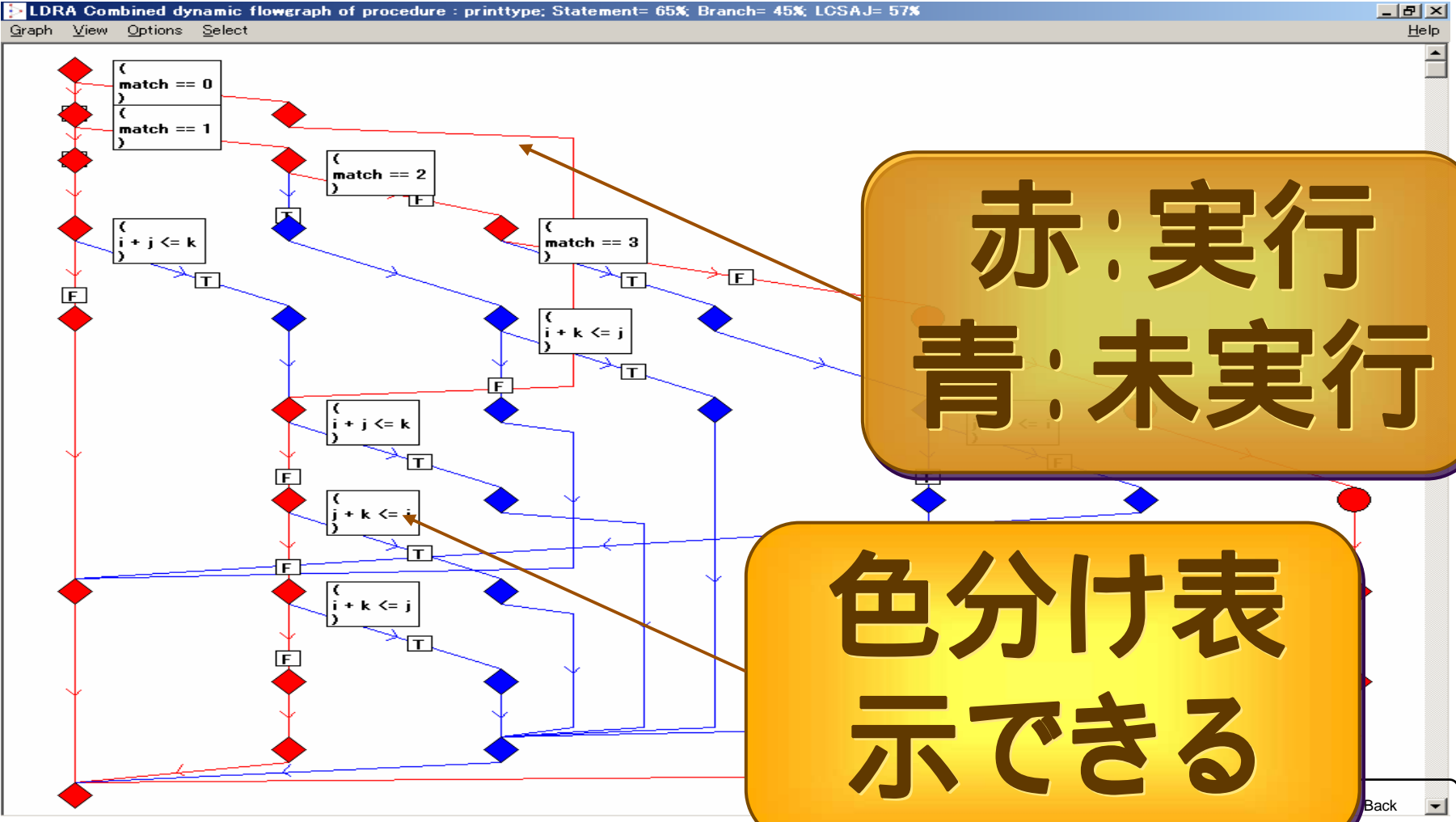
実行フローを 解析



実行フロー内の

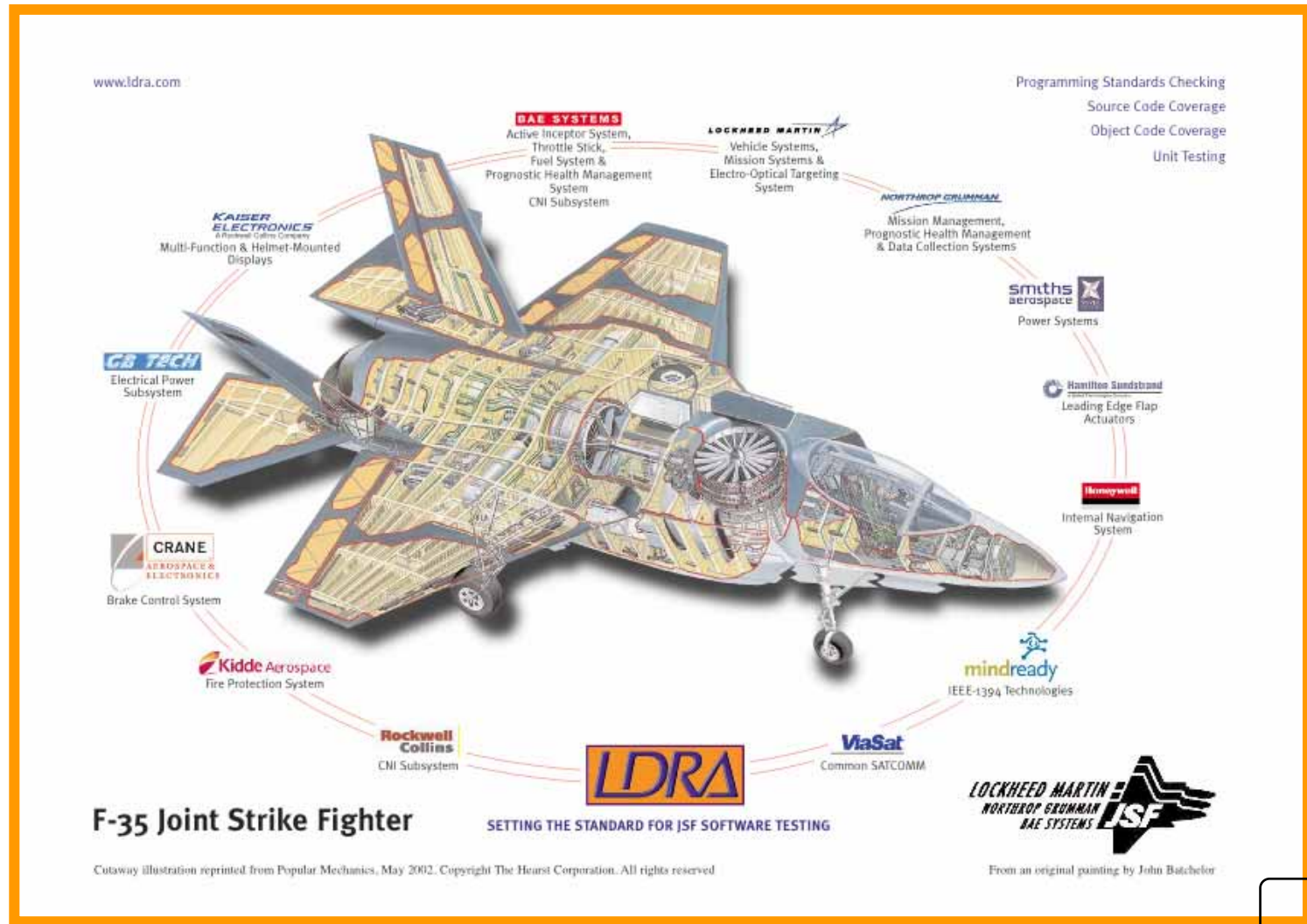


結果のカバレッジ



静的テスト、単
体テスト、カバレ
ツジが有機的に
融合される

JSF開発では、



JSF開発では、
2000～3000人の
SWエンジニアの
標準ツールとして、



JSF開発では、 コーディング規約



単体テスト

ソースコードカバレッジ

オブジェクトコードカバレッジ

JOINT STRIKE FIGHTER

AIR VEHICLE

C++ CODING STANDARDS

FOR THE SYSTEM DEVELOPMENT AND DEMONSTRATION PROGRAM

Document Number 2RDU00001 Rev C

December 2005

JSF C++ コーディング規約



Copyright 2005 by Lockheed Martin Corporation.

DISTRIBUTION STATEMENT A: Approved for public release; distribution is unlimited.

Back

静的テスト、

単体テスト、

カバレッジ

が有機的に融合

されることで、



検証作業、



リグレッション、

要求対テストのトレー

サビリティを効率

DO-178B 全レベル



MISRA C



MISRA-C:2004

IEC 61508 全レベル



高信頼性向けの 機能、性能を持つ ていながら



大規模開発にも 利用できる

コストパフォーマンス でご提供。

F-35 Joint Strike Fighter **LDRA** **SETTING THE STANDARDS FOR JSF SOFTWARE TESTING**

Various partner logos and text are visible, including: Boeing Standards Checking, Source Code Coverage, Object Code Coverage, Unit Testing, Lockheed Martin, Boeing, and others.

Photos: Illustration reprinted from Popular Mechanics, May 2002. Copyright The Boeing Corporation. All rights reserved.

Photo: reprinted permission from Lockheed

[Back](#)

本数割引

2本目8割引

3本目以降9割引

付帯条件なし

ところで、

高信頼性向け
では無い場合

靜的解析

```
File Edit Configure View Window Help
Code Violations All
Annotations Src Lines
(C) STATIC VIOLATION : 201 S : Use of numeric literal in expression. : 31 */
z=7;
(O) STATIC VIOLATION : 187 S : Tab character in source. Col 1 : 32T z = 7 ; */
(C) STATIC VIOLATION : 201 S : Use of numeric literal in expression. : 32 */
z=8;
(O) STATIC VIOLATION : 187 S : Tab character in source. Col 1 : 33T z = 8 ; */
(C) STATIC VIOLATION : 201 S : Use of numeric literal in expression. : 33 */
z=9;
(O) STATIC VIOLATION : 187 S : Tab character in source. Col 1 : 34T z = 9 ; */
(C) STATIC VIOLATION : 201 S : Use of numeric literal in expression. : 34 */
z=10;
(C) STATIC VIOLATION : 190 S : { ... } contents not indented by 2 spaces. Col 1 : 35T16 : */
(O) STATIC VIOLATION : 111 S : MISRA 55: Label is not part of switch statement (MR). : 35 */
(O) STATIC VIOLATION : 187 S : Tab character in source. Col 4 : 36T z = 10 ; */
(C) STATIC VIOLATION : 201 S : Use of numeric literal in expression. : 36 */

(M) DATAFLOW VIOLATION : 71: Procedure definition does not have associated prototype : datanoma */
(M) DATAFLOW VIOLATION : Variables declared but not analysed. */
(M) DATAFLOW VIOLATION : 30: UR flow analysis found a loop. */
(O) DATAFLOW VIOLATION : 7: UR flow analysis found a loop. */
(O) DATAFLOW VIOLATION : 7: UR flow analysis found a loop. */
(M) DATAFLOW VIOLATION : 7: UR flow analysis found a loop. */
(M) DATAFLOW VIOLATION : 7: UR flow analysis found a loop. */
(M) DATAFLOW VIOLATION : 7: UR flow analysis found a loop. */

d knots()
(M) STATIC VIOLATION : 386 S : Less than 2 blank lines separating procs. : 38F */
(M) STATIC VIOLATION : 63 S : MISRA 76: Empty parameter list to procedure/function. : 40T knots() */
(C) STATIC VIOLATION : 175 S : Function starts with lower case character. : 40 */

(C) STATIC VIOLATION : 190 S : { ... } contents not indented by 2 spaces. Col 9 : 41F { */
int u,v;
(O) STATIC VIOLATION : 90 S : MISRA 2: Basic type declaration used. : 42F int */
(O) STATIC VIOLATION : 187 S : Tab character in source. Col 1 : 44F v ; */
(M) STATIC VIOLATION : 274 S : Name found with length less than 2. : 43 */
(M) STATIC VIOLATION : 177 S : Identifier not declared on new line. : 44 */
(M) STATIC VIOLATION : 274 S : Name found with length less than 2. : 44 */
(C) STATIC VIOLATION : 207 S : Use of old style /* comments in C++ : 44 */

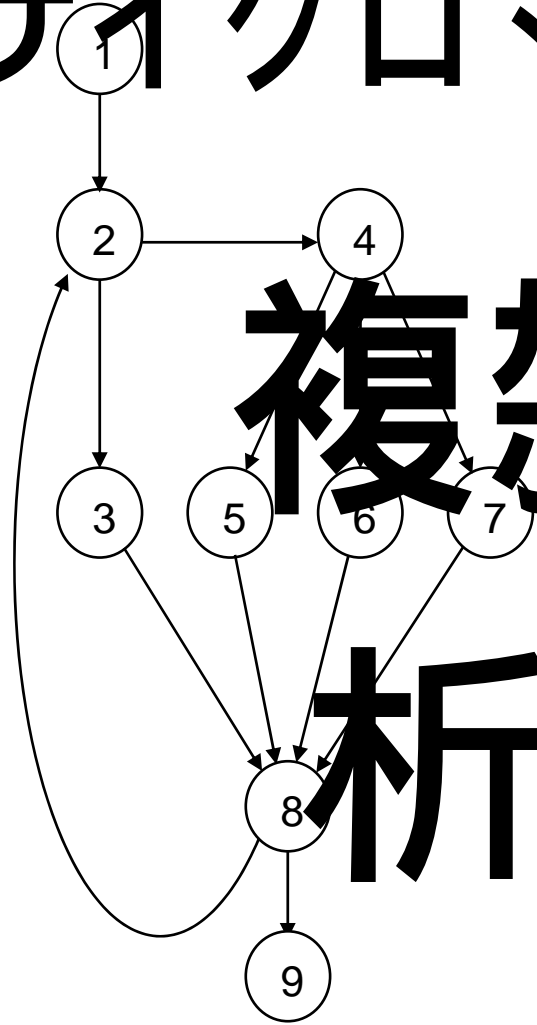
down-down knot */
if (u==1) goto lb;
(O) STATIC VIOLATION : 187 S : Tab character in source. Col 1 : 51T goto lb ; */
(M) STATIC VIOLATION : 116 S : Boolean comparison with 0 preferred. : 48 */
(M) STATIC VIOLATION : 12 S : MISRA 59: No brackets to then/else. : 50 */
(M) STATIC VIOLATION : 13 S : MISRA 56: goto detected. : 51 */
v=0;
(C) STATIC VIOLATION : 190 S : { ... } contents not indented by 2 spaces. Col 1 : 53T1a : */
(O) STATIC VIOLATION : 111 S : MISRA 55: Label is not part of switch statement (MR). : 53 */
(O) STATIC VIOLATION : 187 S : Tab character in source. Col 4 : 54T v = 0 ; */
v=0*

estbed#Examples#Cpp_testbed_examples#Tbsdemo#Tbsdem3.cpp Standards Model: <No Model - all standards>
help, press F1 Ln 1, Col 1
```

靜的解 析?

複雜度解析

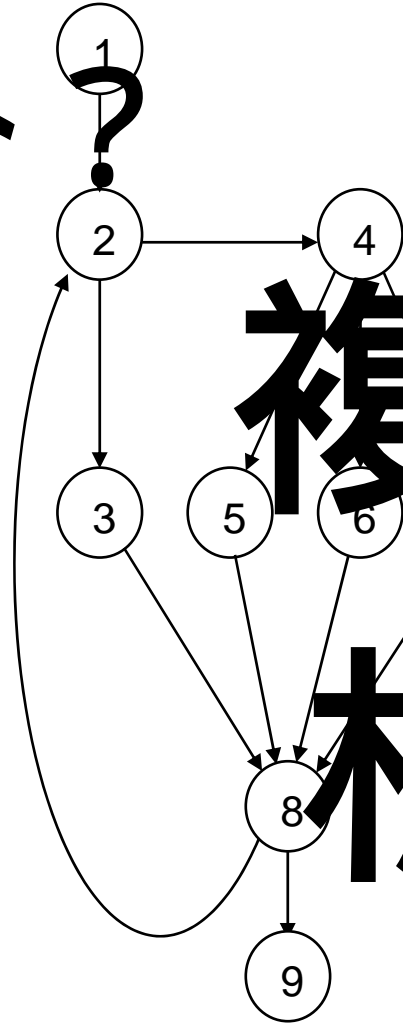
サイクロマティック？



複雑度解 析？

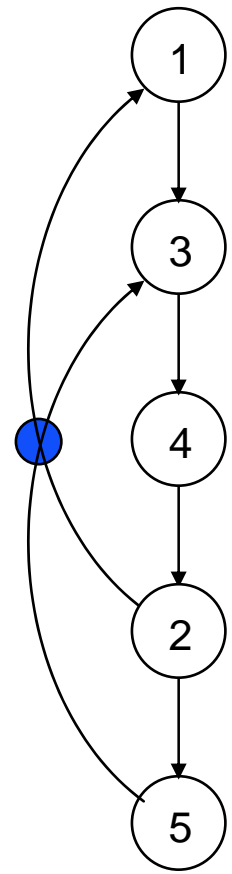
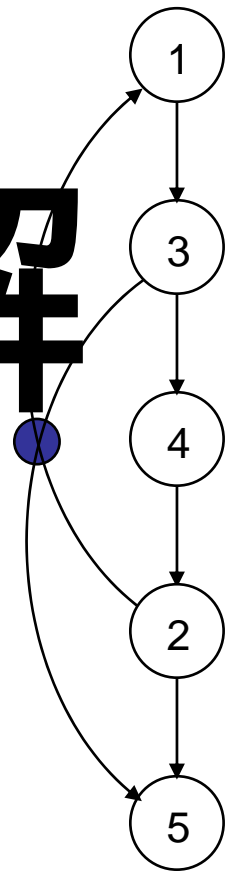
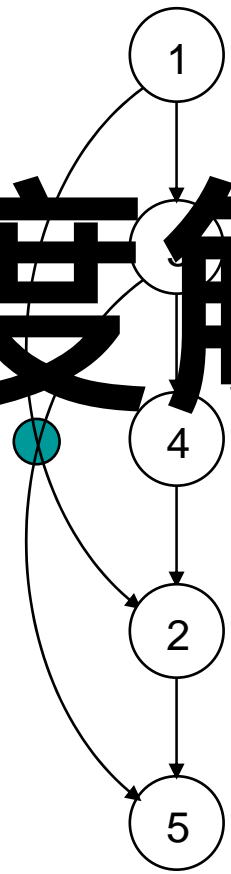
サイクロマティック？ ノット

ト



複雑度解

析？



•Executable ref. Lines

•Executable reformatted Lines

•Essential Knots

サイクロマティック?

デューク?

ノット

•Total Comments

•Number of Basic Blocks

•Total Operands

ト?

•Comments in Headers

•Number of Loops

•Comments in Declarations

•Procedure Exit Points

•Comments in Executable Code

•Number of Face Nodes

•Blank Lines

•Total Comments/Exec. Lines

•Declaration Comments/Exe. Lines

•Code Comments/Exec. Lines

•Average Length of Basic Blocks

•Unique Operands

•Total LCSAJs

•Unreachable LCSAJs

•Maximum LCSAJ Density

•Unreachable Lines

•Unreachable Branches

•File Fan in

•Fan Out

•Knots

•Cyclomatic Complexity

•Vocabulary

•Number of Procedures

•Total LCSAJs

•Unreachable LCSAJs

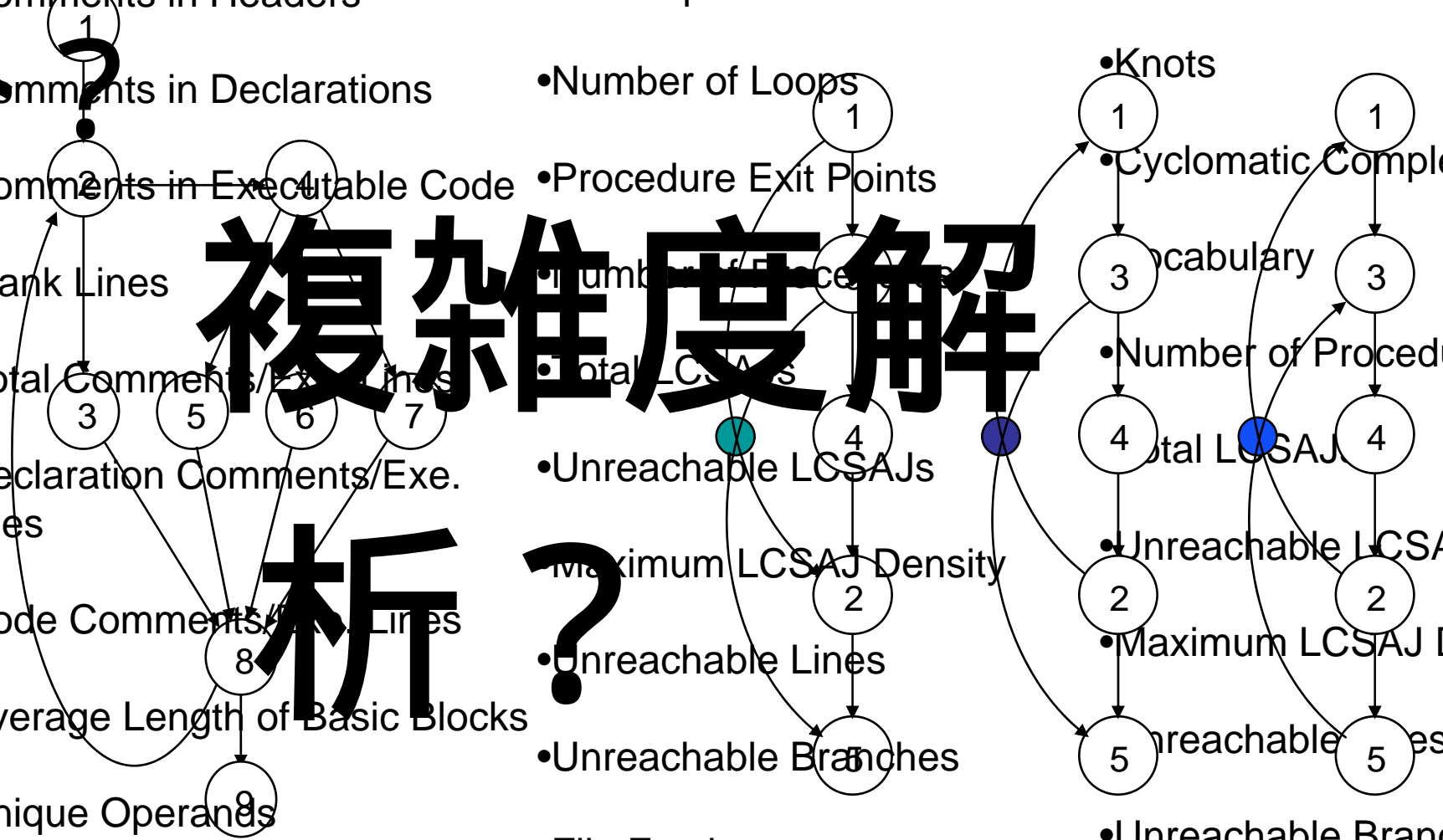
•Maximum LCSAJ Density

•Unreachable Lines

•Unreachable Branches

複雑度解

析



• Executable ref. Lines

• Executable reformatted Lines

• Essential Knots

サイクロマティック？

• Essential Cyclomatic Complexity

• Comments in Headers

• Total Operands

ト？

• Comments in Declarations

• Number of Loops

• Knots

• Comments in Executable Code

• Procedure Exit Points

• Cyclomatic Complexity

• Blank Lines

• Number of Procedures

• Vocabulary

• Total Comments

• Total LCSAJs

• Number of Procedures

複雑度解

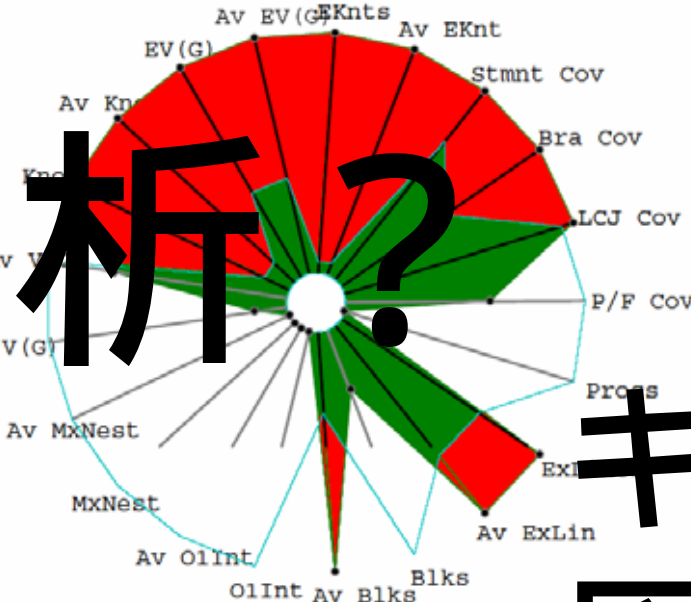
• Declaration Code Lines

• Code Comments

• Average Length

• Unique Operands

• Total LCSAJs



析？

Density

2

3

4

5

• Total LCSAJs

• Unreachable LCSAJs

• Maximum LCSAJ Density

• Reachable LCSAJs

• Unreachable Branches

図？

でも、



の

静的テスト

であれば、

解析済の領域は、

抑止できる。

```
/*LDRA_INSPECTED */
```


複雜度解析も、

可読性、メンテナ ナンス容易性、 テスト容易性に

Procedure	All Metrics	Clarity	Maintainability	Testability
equalsides	77	40	100	100
printtype	54	40	0	63
Total for Thedam9.asp	76	57	73	93

1. Number of Basic Blocks
2. Cyclomatic Complexity

おまとめ表示。

Procedure	All Metrics	Clarity	Maintainability	Testability
equalsides	77	40	100	100
printtype	54	40	0	63
Total for Tbsdem2.cpp	76	57	73	93

1. Number of Basic Blocks
2. Cyclomatic Complexity
3. Knots

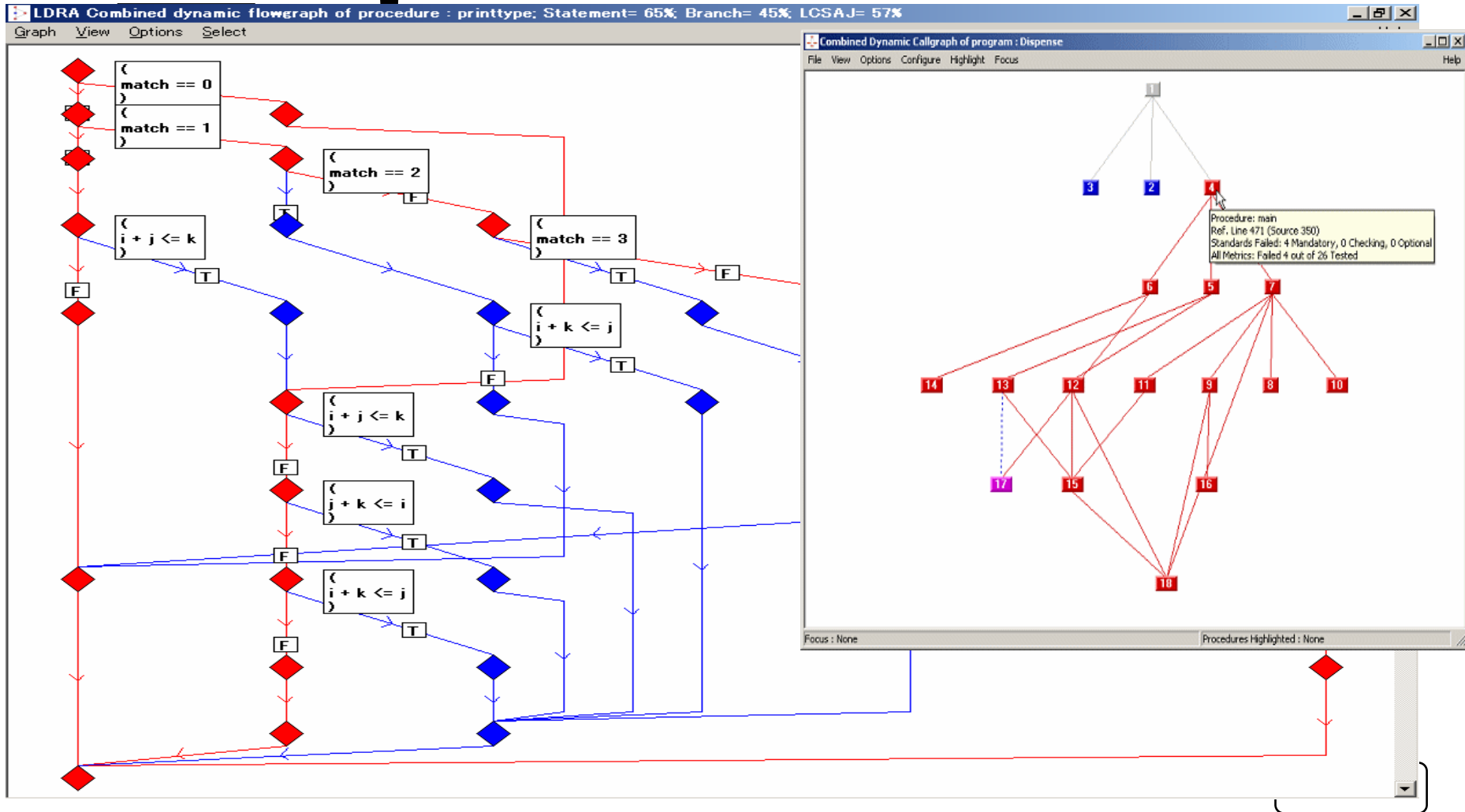
しかも、



単体テスト
テストカバレッジ
が、

視覚的に融

合

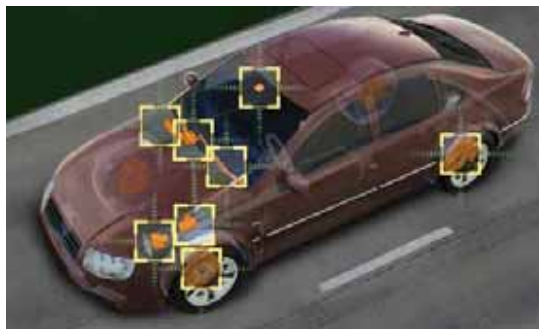


検証作業、
リグレッション、
などを効率化

・ソースコードベース

・モデル検証

・自動コード生成
(Domain-Specific Modeling)



We Cover All Boundaries

モデル検証

モデル検証の

課題は、、、

形式的仕様記述

言語を使って

要求仕様

```
VARIABLE Crew_Interface_Throttles_term_GA_Pressed__VAR ISA Boolean ;
```

```
LOGIC STRUCTURE Autopilot_Modes_Disengaged_Submode_FROM_CLEARED_NoChange__LS(_Autopilot_Modes_Disengaged_Submode:A  
ATTRIBUTE ModelLoc="ttm://FGS?name=Autopilot_Modes_Disengaged_Submode";  
ATTRIBUTE ModelLocID="No-change for mode CLEARED";  
ATTRIBUTE ID="No-change for mode CLEARED";  
CONSTRAINT
```

```
((_Autopilot_Modes_Disengaged_Submode = CLEARED::Autopilot_Modes_Disengaged_Submode__Type)  
AND  
NOT:(_Autopilot_Modes_Engage = DISENGAGED::Autopilot_Modes_Engage__Type))  
AND  
(((Autopilot_Modes_Disengaged_Submode = CLEARED::Autopilot_Modes_Disengaged_Submode__Type)  
AND  
(_Autopilot_Modes_Engage = Autopilot_Modes_Engage__VAR))  
AND  
(_Crew_Interface_Throttles_term_GA_Pressed = Crew_Interface_Throttles_term_GA_Pressed__VAR));
```

```
LOGIC STRUCTURE Autopilot_Modes_Disengaged_Submode_FROM_CLEARED_TO_Normal__D1__LS(_Autopilot_Modes_Disengaged_Subm  
ATTRIBUTE ModelLoc="ttm://FGS?name=Autopilot_Modes_Disengaged_Submode&row=3&col=2";  
CONSTRAINT
```

```
(_Autopilot_Modes_Disengaged_Submode = CLEARED::Autopilot_Modes_Disengaged_Submode__Type)  
AND  
((NOT:(_Autopilot_Modes_Engage = DISENGAGED::Autopilot_Modes_Engage__Type)  
AND  
(Autopilot_Modes_Engage__VAR = DISENGAGED::Autopilot_Modes_Engage__Type))  
AND  
NOT:(_Crew_Interface_Throttles_term_GA_Pressed = FALSE));
```

```
LOGIC STRUCTURE Autopilot_Modes_Disengaged_Submode_FROM_CLEARED_TO_Normal__D2__LS(_Autopilot_Modes_Disengaged_Subm  
ATTRIBUTE ModelLoc="ttm://FGS?name=Autopilot_Modes_Disengaged_Submode&row=3&col=2";  
CONSTRAINT
```

モデルの検証で

終わり???

で、

実装の検証は
どーなるの？

というのが、一般的
なモデル検証、
フォーマルメソッド
の課題



We Cover All Boundaries

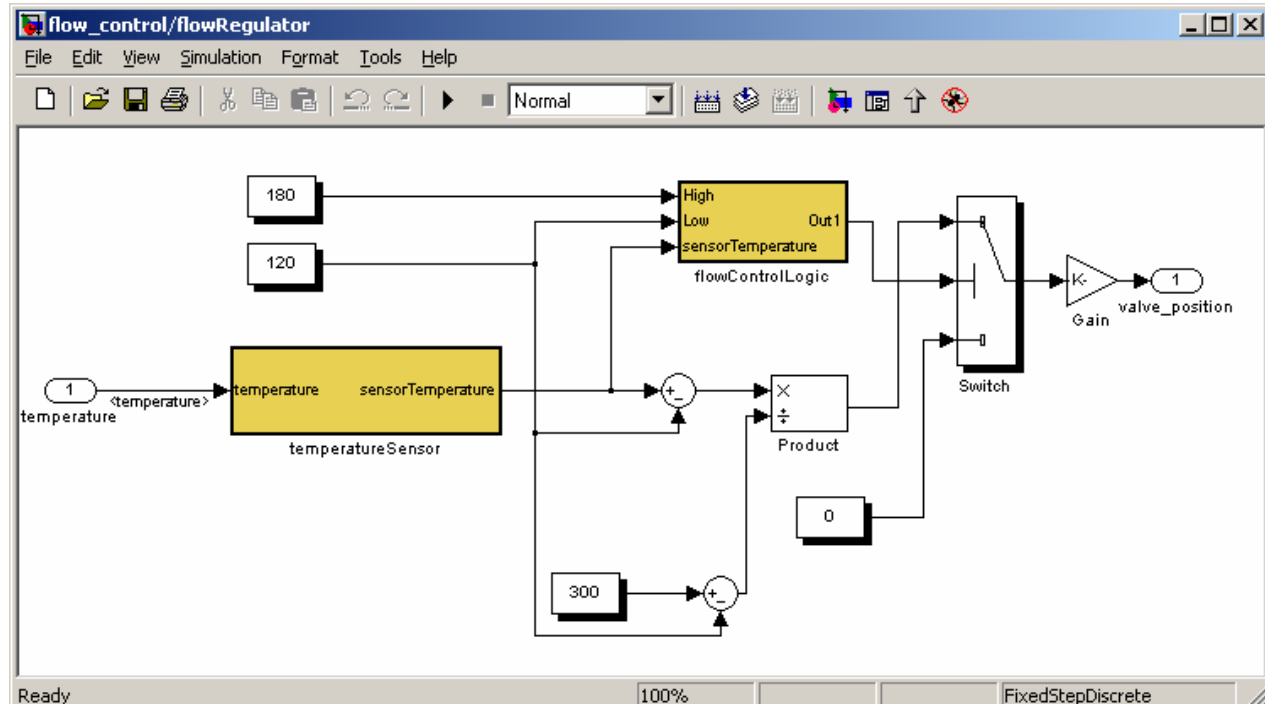
モデル検証は、

形式的仕様

記述言語を

Simulink

モデルや



TTM 要求仕様

の表モデルを

The screenshot displays the T-VEC Tabular Modeler interface. On the left is a tree view of the model structure, including sections for Assertions, Functions, Mode Machines, and Terms. The main workspace shows a table model for 'Flight_Modes_Lateral_Active' with the following data:

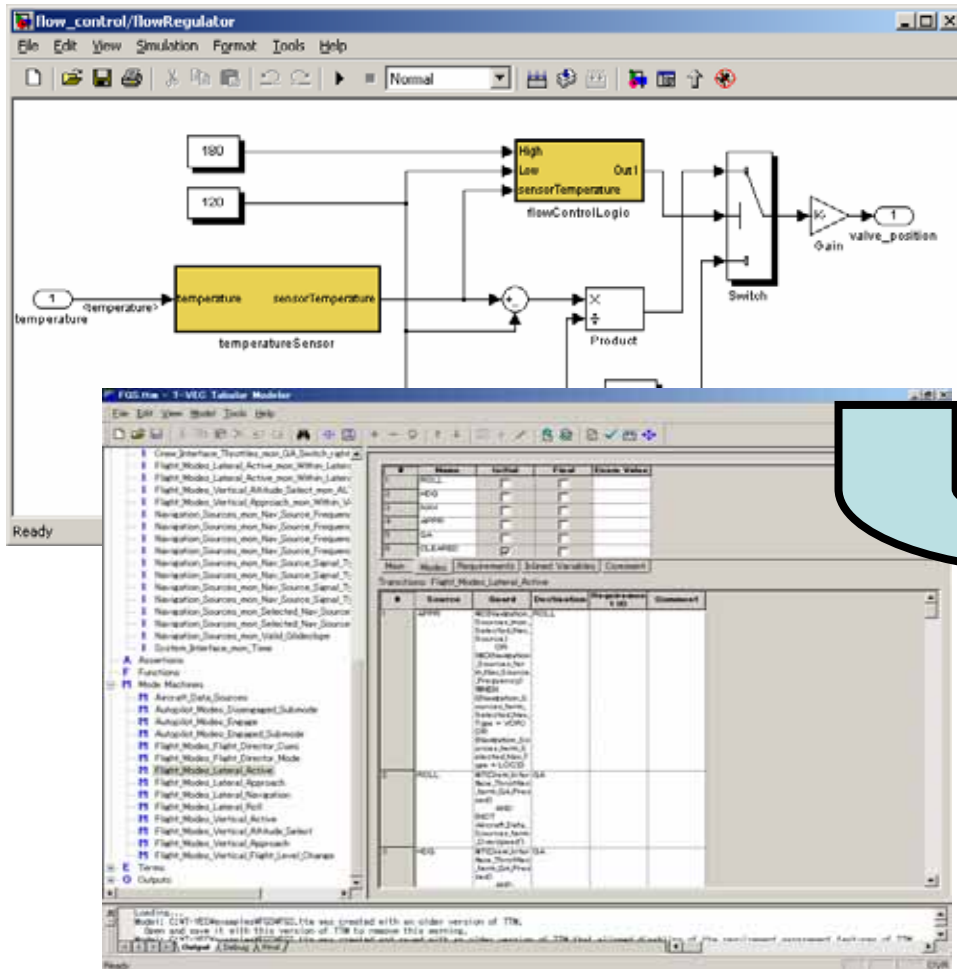
#	Name	Initial	Final	Enum Value
1	ROLL	<input type="checkbox"/>	<input type="checkbox"/>	
2	HOG	<input type="checkbox"/>	<input type="checkbox"/>	
3	NAV	<input type="checkbox"/>	<input type="checkbox"/>	
4	APPR	<input type="checkbox"/>	<input type="checkbox"/>	
5	GA	<input type="checkbox"/>	<input type="checkbox"/>	
6	CLEARED	<input checked="" type="checkbox"/>	<input type="checkbox"/>	

Below the table, the 'Requirements' tab is active, showing a list of transitions for 'Flight_Modes_Lateral_Active' with columns for Source, Guard, Destination, Requirement ID, and Comment.

#	Source	Guard	Destination	Requirement ID	Comment
1	APPR	(NOT Navigation_Sources_mon_Selected_Nav_Source) OR (NOT Navigation_Sources_mon_Selected_Nav_Source) AND (Navigation_Sources_mon_Selected_Nav_Type = VGRD) OR (Navigation_Sources_mon_Selected_Nav_Type = LOC)	ROLL		
2	ROLL	(NOT (Directional_Term_GA_Pressed)) AND (NOT AircraftData_Sources_mon_Diverspeed)	GA		
3	HOG	(NOT (Directional_Term_GA_Pressed)) AND	GA		

At the bottom, a status bar shows a warning message: "Model: C:\MT-VECSamples\FGS\FGS.ttm was created with an older version of TTM. Open and save it with this version of TTM to remove this warning." Below the warning is an "Output" window with "Debug" and "Find" buttons.

変換して生成。



```
VARIABLE Crew_Interface_Throttles_term_GA_Pressed_VAR ISA Boolean ;
```

```
LOGIC STRUCTURE Autopilot_Modes_Disengaged_Submode_FROM_CLEARED_NoChange_LS(Autopilot_Modes_Disengaged_Submode:Autopilot_Modes_Disengaged_Submode)
ATTRIBUTE ModelLoc="tt://FGS?name=Autopilot_Modes_Disengaged_Submode";
ATTRIBUTE ModelLocID="No-change for mode CLEARED";
ATTRIBUTE ReqID="No-change for mode CLEARED";
CONSTRAINT
```

```
((Autopilot_Modes_Disengaged_Submode = CLEARED::Autopilot_Modes_Disengaged_Submode_Type)
AND
NOT:(Autopilot_Modes_Engage = DISENGAGED::Autopilot_Modes_Engage_Type))
AND
(((Autopilot_Modes_Disengaged_Submode = CLEARED::Autopilot_Modes_Disengaged_Submode_Type)
AND
(Autopilot_Modes_Engage = Autopilot_Modes_Engage_VAR))
AND
(Crew_Interface_Throttles_term_GA_Pressed = Crew_Interface_Throttles_term_GA_Pressed_VAR));
```

```
LOGIC STRUCTURE Autopilot_Modes_Disengaged_Submode_FROM_CLEARED_TO_Normal_D1_LS(Autopilot_Modes_Disengaged_Submode:Autopilot_Modes_Disengaged_Submode)
ATTRIBUTE ModelLoc="tt://FGS?name=Autopilot_Modes_Disengaged_SubmodeFrom=3&col=2";
CONSTRAINT
```

```
((Autopilot_Modes_Disengaged_Submode = CLEARED::Autopilot_Modes_Disengaged_Submode_Type)
AND
((NOT:(Autopilot_Modes_Engage = DISENGAGED::Autopilot_Modes_Engage_Type)
AND
(Autopilot_Modes_Engage_VAR = DISENGAGED::Autopilot_Modes_Engage_Type))
AND
NOT:(Crew_Interface_Throttles_term_GA_Pressed = FALSE));
```

```
LOGIC STRUCTURE Autopilot_Modes_Disengaged_Submode_FROM_CLEARED_TO_Normal_D2_LS(Autopilot_Modes_Disengaged_Submode:Autopilot_Modes_Disengaged_Submode)
ATTRIBUTE ModelLoc="tt://FGS?name=Autopilot_Modes_Disengaged_SubmodeFrom=3&col=2";
CONSTRAINT
```

楽です。

直接書くより。

```
VARIABLE Crew_Interface_Throttles_term_GA_Pressed__VAR ISA Boolean ;
```

```
LOGIC STRUCTURE Autopilot_Modes_Disengaged_Submode_FROM_CLEARED_NoChange__LS(_Autopilot_Modes_Disengaged_Submode:A  
  ATTRIBUTE ModelLoc="ttm://FGS?name=Autopilot_Modes_Disengaged_Submode";  
  ATTRIBUTE ModelLocID="No-change for mode CLEARED";  
  ATTRIBUTE ReqID="No-change for mode CLEARED";  
  CONSTRAINT  
    ((_Autopilot_Modes_Disengaged_Submode = CLEARED::Autopilot_Modes_Disengaged_Submode__Type)  
     AND  
    NOT:(_Autopilot_Modes_Engage = DISENGAGED::Autopilot_Modes_Engage__Type))  
     AND  
    ((_Autopilot_Modes_Disengaged_Submode = CLEARED::Autopilot_Modes_Disengaged_Submode__Type)  
     AND  
    (_Autopilot_Modes_Engage = Autopilot_Modes_Engage__VAR))  
     AND  
    (_Crew_Interface_Throttles_term_GA_Pressed = Crew_Interface_Throttles_term_GA_Pressed__VAR));
```

```
LOGIC STRUCTURE Autopilot_Modes_Disengaged_Submode_FROM_CLEARED_TO_Normal__D1__LS(_Autopilot_Modes_Disengaged_Subm  
  ATTRIBUTE ModelLoc="ttm://FGS?name=Autopilot_Modes_Disengaged_Submode&row=3&col=2";  
  CONSTRAINT  
    (_Autopilot_Modes_Disengaged_Submode = CLEARED::Autopilot_Modes_Disengaged_Submode__Type)  
     AND  
    ((NOT:(_Autopilot_Modes_Engage = DISENGAGED::Autopilot_Modes_Engage__Type)  
     AND  
    (Autopilot_Modes_Engage__VAR = DISENGAGED::Autopilot_Modes_Engage__Type))  
     AND  
    NOT:(_Crew_Interface_Throttles_term_GA_Pressed = FALSE));
```

```
LOGIC STRUCTURE Autopilot_Modes_Disengaged_Submode_FROM_CLEARED_TO_Normal__D2__LS(_Autopilot_Modes_Disengaged_Subm  
  ATTRIBUTE ModelLoc="ttm://FGS?name=Autopilot_Modes_Disengaged_Submode&row=3&col=2";  
  CONSTRAINT
```

しかも、

モデルの検証

で終わらな

い！

モデル検証結果

からテストベクタ

を

生成！

Test Vectors (Compressed) For Subsystem : temperatureSensor

Test #	Vector #s	__temperatureSensorData ._output	__temperatureSensorData ._local ._IState_Unit_Delay	temperature_iPort	__local ._IState_Unit_Delay
1	1,9	-1.0e+002	-2.457e+002	-2.73e+002	0.0e+000
2	2,6	3.0e+002	5.4e+002	6.0e+002	0.0e+000
3	3,11	-1.0e+002	-1.00000002457e+010	-2.73e+002	-1.0e+011
4	4,8	3.0e+002	1.000000054e+010	6.0e+002	1.0e+011
5	5	3.0e+002	3.00000000000055e+002	3.33333333333394e+002	0.0e+000
6	7	3.0e+002	3.0000000000002e+002	-2.73e+002	5.4570000000002e+003
7	10	-1.0e+002	-1.0000000000012e+002	-1.11111111111244e+002	0.0e+000
8	12	-1.0e+002	-1.00002e+002	6.0e+002	-6.40002e+003
9	13	-1.0e+002	-1.0e+002	-1.1111111111111e+002	0.0e+000
10	14	3.0e+002	3.0e+002	3.3333333333333e+002	0.0e+000
11	15	-1.0e+002	-1.0e+002	-2.73e+002	1.457e+003
12	16	3.0e+002	3.0e+002	6.0e+002	-2.4e+003

Test #1: Vector(s) : 1,9

```
DCP # 1 : temperatureSensor, temperatureSensor_FR_1, cv_temperatureSensor_RP_1
temperatureSensor_RP_0, RP0, RP1_1
Constrained Input CV Mode      : LOW_BOUND
Unconstrained Input CV Mode    : OPPOSITE
Vector Generation Mode         : SINGLE
Input Value Assignment Order   : SPEC_SEQUENTIAL_INPUT
MCDC Support Level             : DO178B_MCDC
```

期待値と、入力値

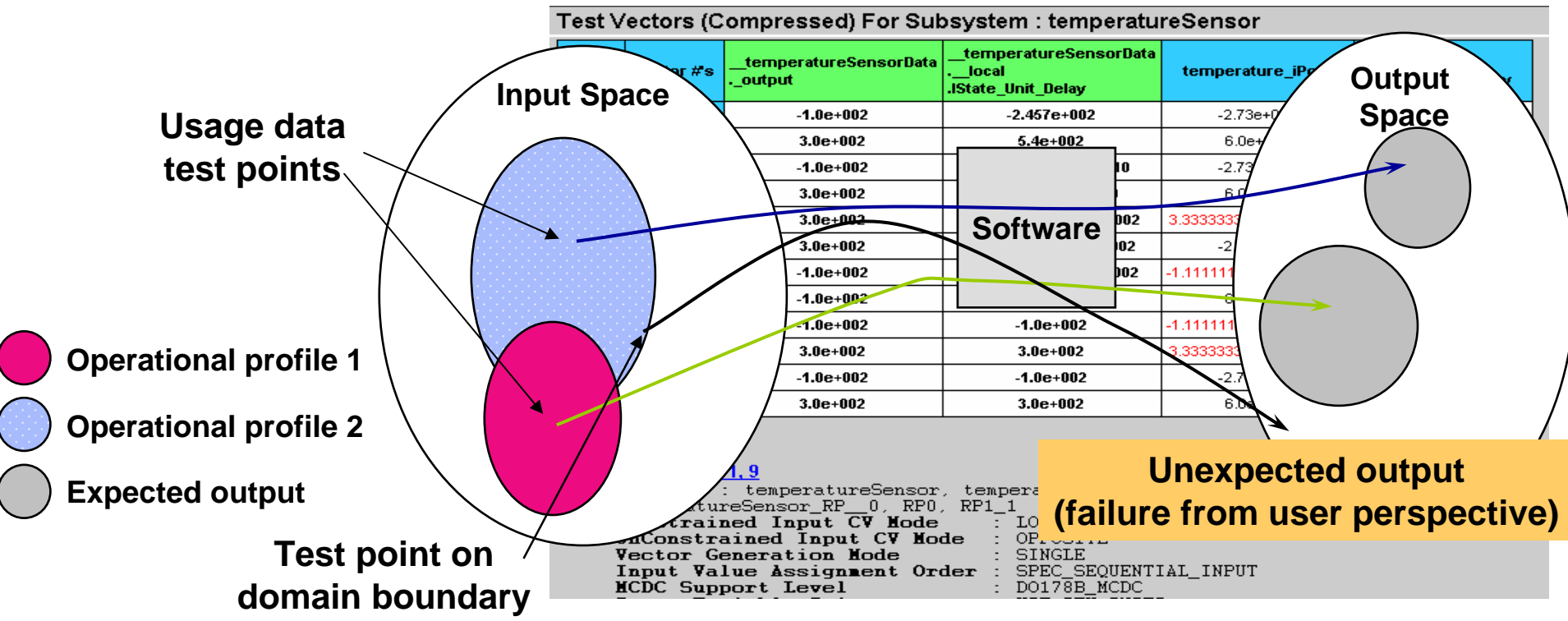
Test Vectors (Compressed) For Subsystem : temperatureSensor

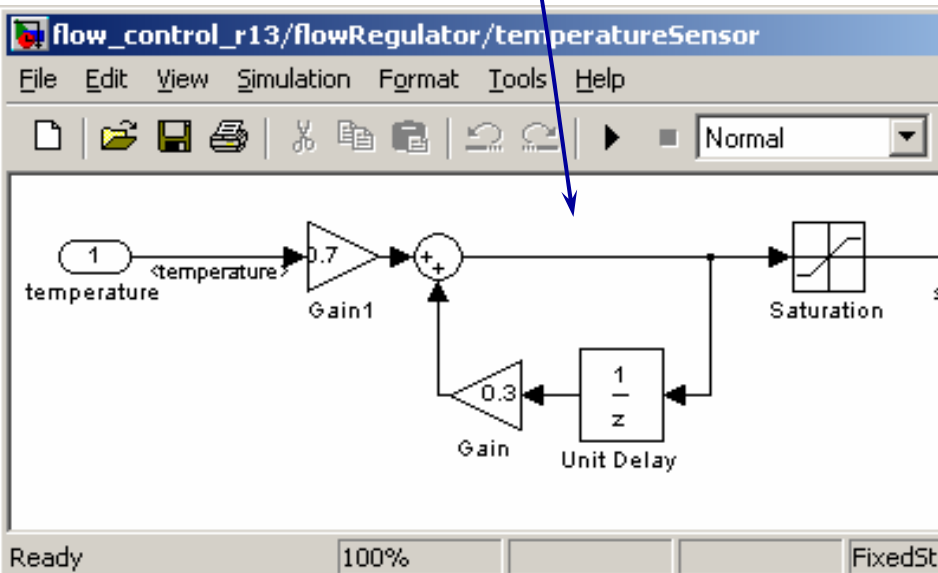
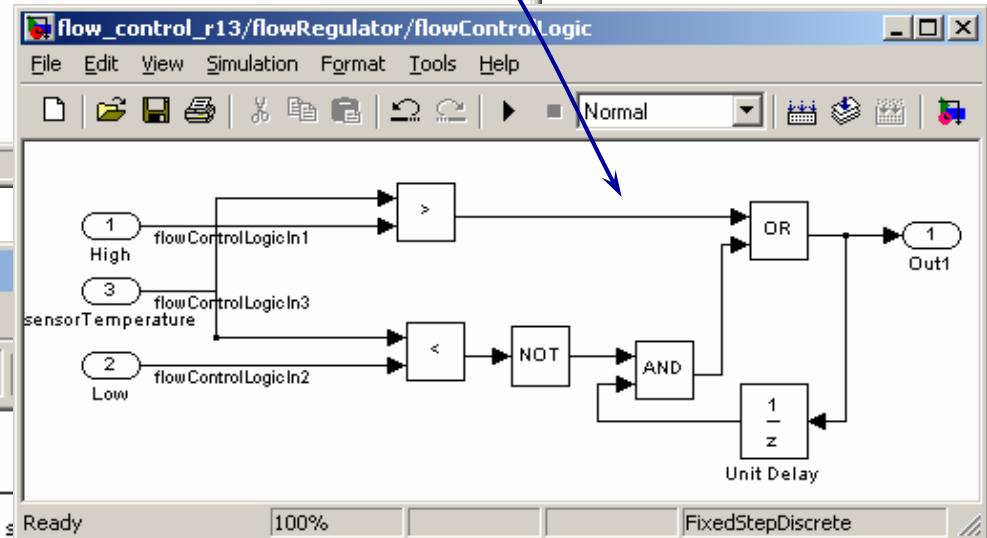
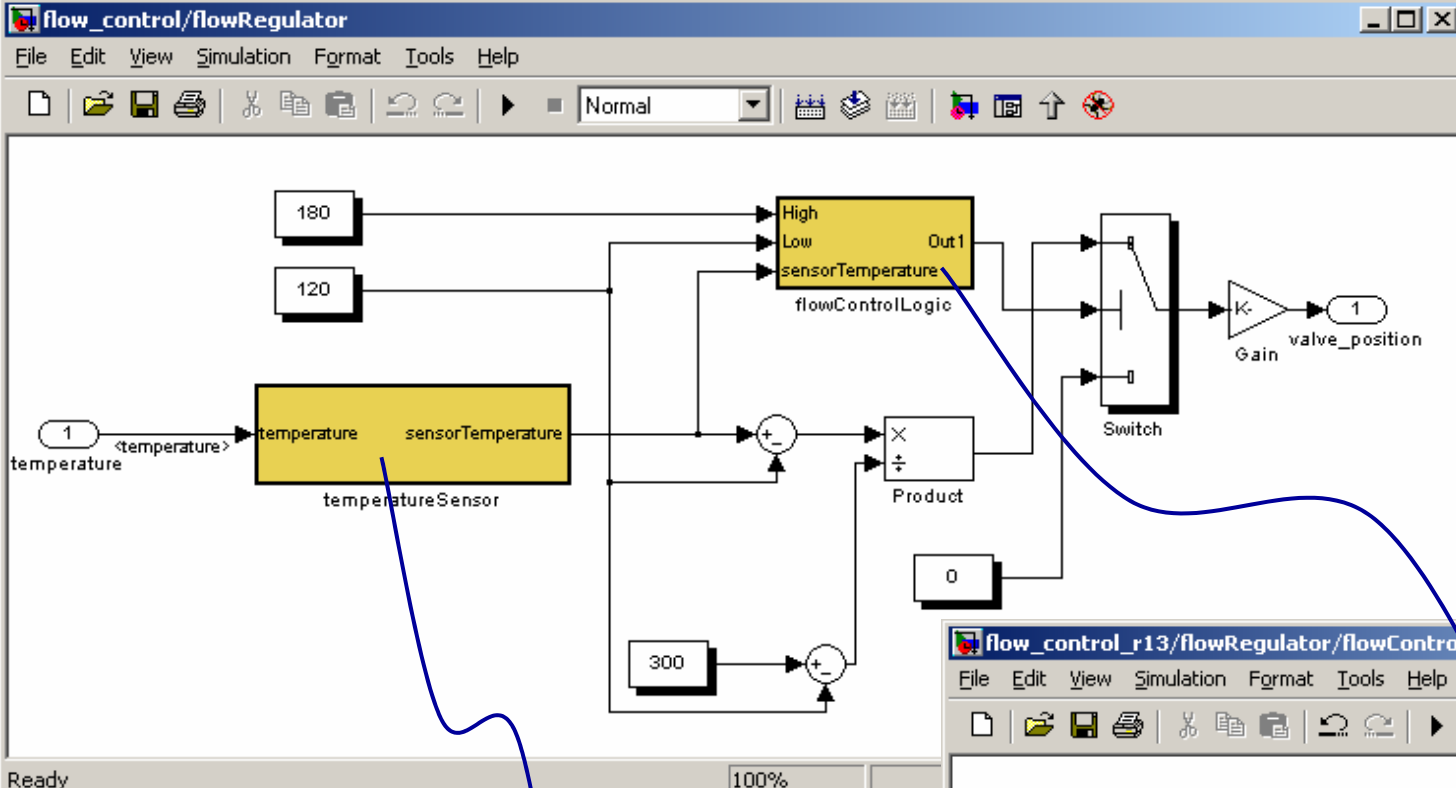
Test #	Vector #'s	__temperatureSensorData ._output	__temperatureSensorData ._local .IState_Unit_Delay	temperature_iPort	__local .IState_Unit_Delay
1	1,9	-1.0e+002	-2.457e+002	-2.73e+002	0.0e+000
2	2,6	3.0e+002	5.4e+002	6.0e+002	0.0e+000
3	3,11	-1.0e+002	-1.00000002457e+010	-2.73e+002	-1.0e+011
4	4,8	3.0e+002	1.000000054e+010	6.0e+002	1.0e+011
5	5	3.0e+002	3.00000000000055e+002	3.33333333333394e+002	0.0e+000
6	7	3.0e+002	3.0000000000002e+002	-2.73e+002	5.4570000000002e+003
7	10	-1.0e+002	-1.0000000000012e+002	-1.11111111111244e+002	0.0e+000
8	12	-1.0e+002	-1.00002e+002	6.0e+002	-6.40002e+003
9	13	-1.0e+002	-1.0e+002	-1.1111111111111e+002	0.0e+000
10	14	3.0e+002	3.0e+002	3.3333333333333e+002	0.0e+000
11	15	-1.0e+002	-1.0e+002	-2.73e+002	1.457e+003
12	16	3.0e+002	3.0e+002	6.0e+002	-2.4e+003

さらに凄いにことに

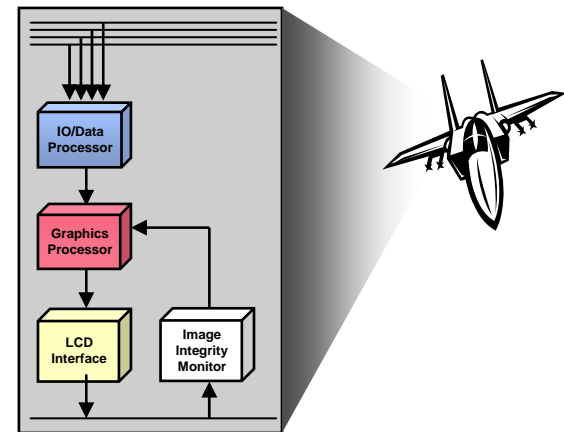
テストベクタは、

サブドメインの境界値を攻める。

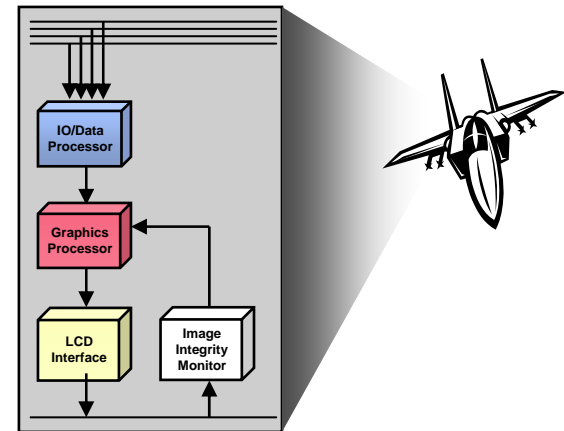




そして、

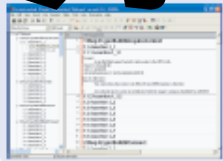


あらゆる環境でも モデルとの一貫性検 証に用いられる

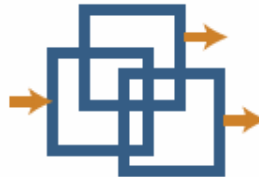


Full Lifecycle Model-Based Verification

要求モデル



DOORS®
Requirements
Management



T-VEC®
Tabular Modeler



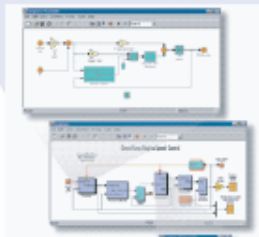
Requirements
Errors



System Test
Vectors



System Test
Scripts/Drivers



Simulink® and
Stateflow® Design



T-VEC® Tester
for Simulink®
and Stateflow®



Design
Errors



Unit Test
Vectors



Unit Test
Drivers
(C and Simulink)

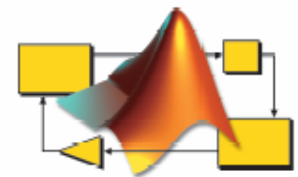
デザインモデル



Implementation through
Real-Time Workshop®
and Stateflow® Coder

Telelogic
Partner

SIMULINK®
Enabled



MathWorks Partner

英国防省にも
採用を推奨さ
れ、



DO178C のモデ

ルベース開発

ガイドライン

にも関わっています。

NIST の SmartCard

モデル検証など

NIST

National Institute of
Standards and Technology
Technology Administration
U.S. Department of Commerce

Interagency Report 6887 - 2003 Edition

Government Smart Card Interoperability Specification

Version 2.1

Teresa Schwarzhoff
Jim Dray
John Wack
Eric Dalci
Alan Goldfine
Michaela Iorga

米国政府機関
スマートカードの
相互運用に
関する仕様書

July 16, 2003

データベースの安全性要求検証 米) 国立標準技術研究所 (NIST)

Oracle8 Security
Target

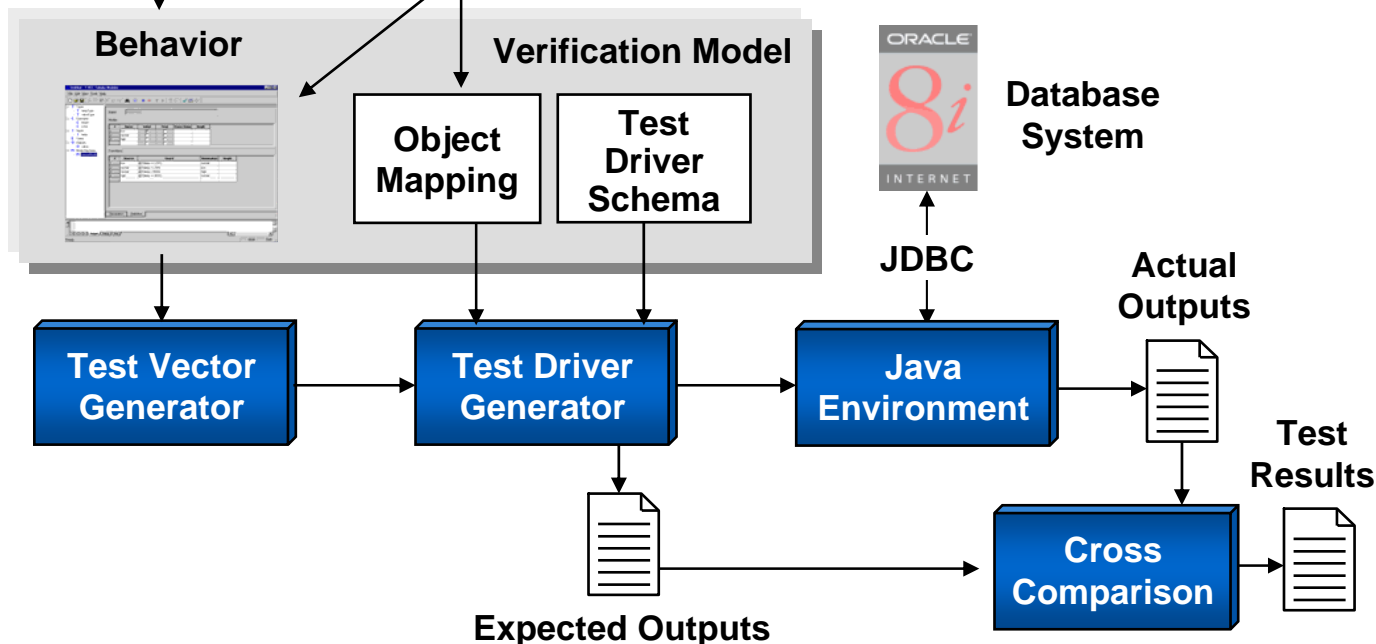


Oracle8 Reference
Oracle8 SQL Reference



SQL

Interfaces
Data dictionary
and
SQL commands



実践的に産業
界で採用され
るモデル検証

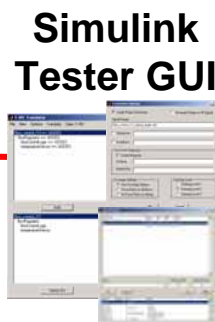
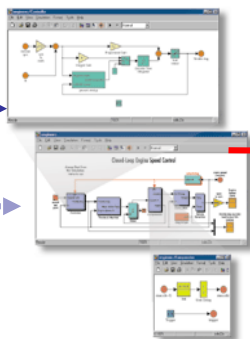
要求仕様 ~ Simulink モデル ~ テスト のトレーサビリティ -

要求仕様書や、
要件管理ツール
(DOORS など)

T-VEC (TTM)
要求仕様を
表でモデル表記

Simulink
Stateflow

T-VEC Test Vector
Generation System



- 要求仕様を表でモデル表記
- 曖昧な要素を形式的モデルに変換

- デザインモデル
- シミュレーション
- 自動コード生成

- モデル検証
- テストベクタ生成
- モデルテストカバレッジ
- テストドライバー生成
- テスト結果解析

- コードカバレッジ (MCDC など)
- 静的解析 (MISRA など)



ロッキードマーチン社によるJSF開発事例

・ソースコードベース

・モデル検証

・自動コード生成
(Domain-Specific
Modeling)

MetaEdit+

Domain-Specific Modeling
for Full Code Generation

自動コード生成

The logo consists of a stylized 'M' shape with a blue top half and a red bottom half, followed by the text 'MetaEdit+' in a bold, black, sans-serif font. The plus sign is blue.

MetaEdit+

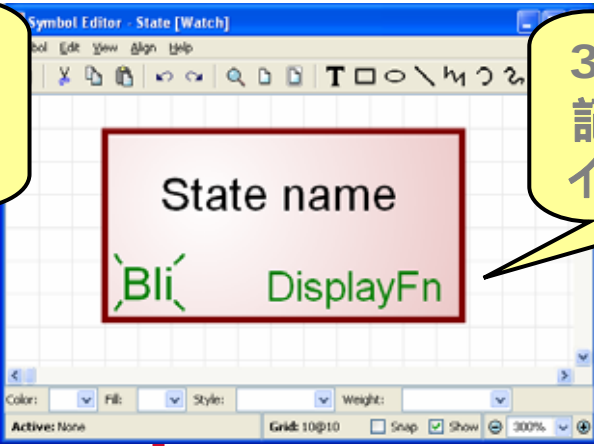
Domain-Specific Modeling
for Full Code Generation

は、

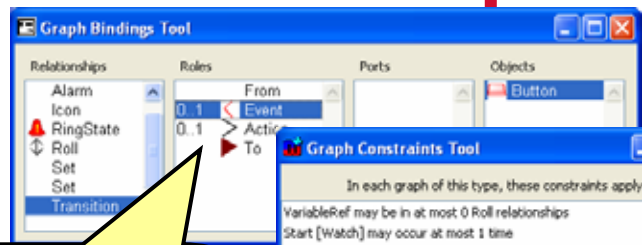
DSMを作るためのツール



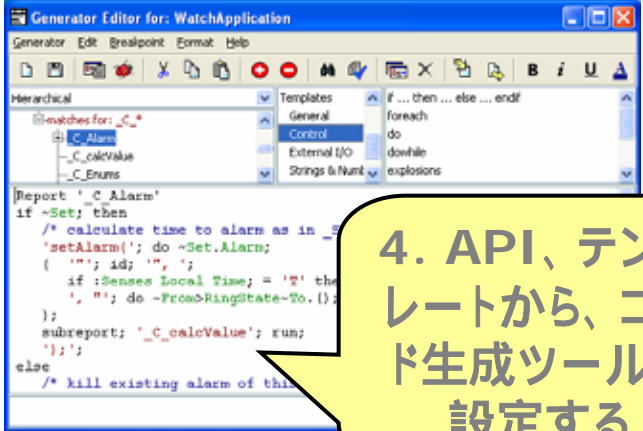
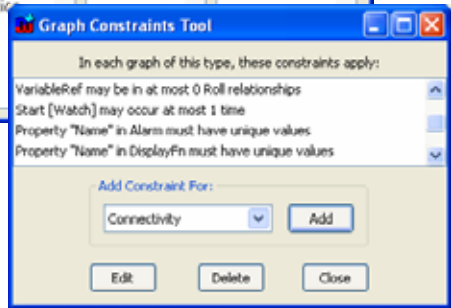
1. 言語のコンセプトとプロパティ(特性)の設定



3. シンボルを記述、もしくはインポートする

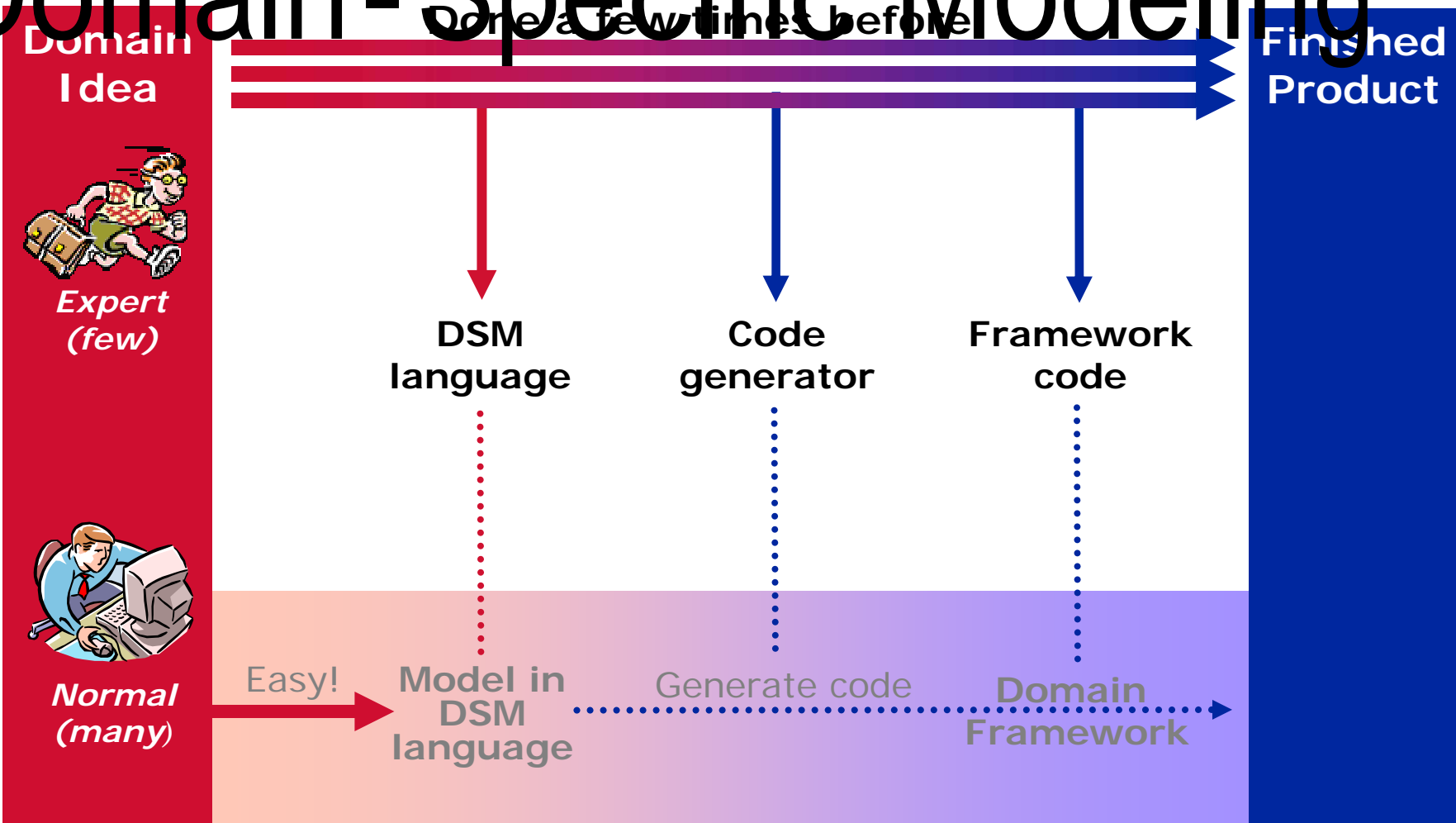


2. テンプレートを使ってルールを選択



4. API、テンプレートから、コード生成ツールを設定する

DSMは、 Domain-Specific Modeling



企業、製品のドメインごとに、

- Domain-Specific にモデリングし、

- Domain-Specific なコードを生成させる

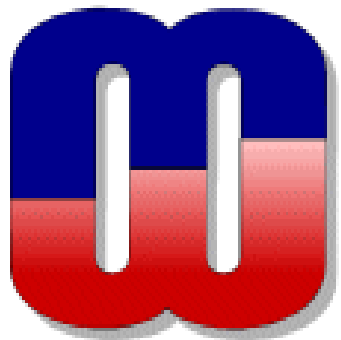
企業、製品のドメインごとに、

- Domain-Specific にモデリングし、
- Domain-Specific なコードを生成させる

企業、製品のドメインごとに、

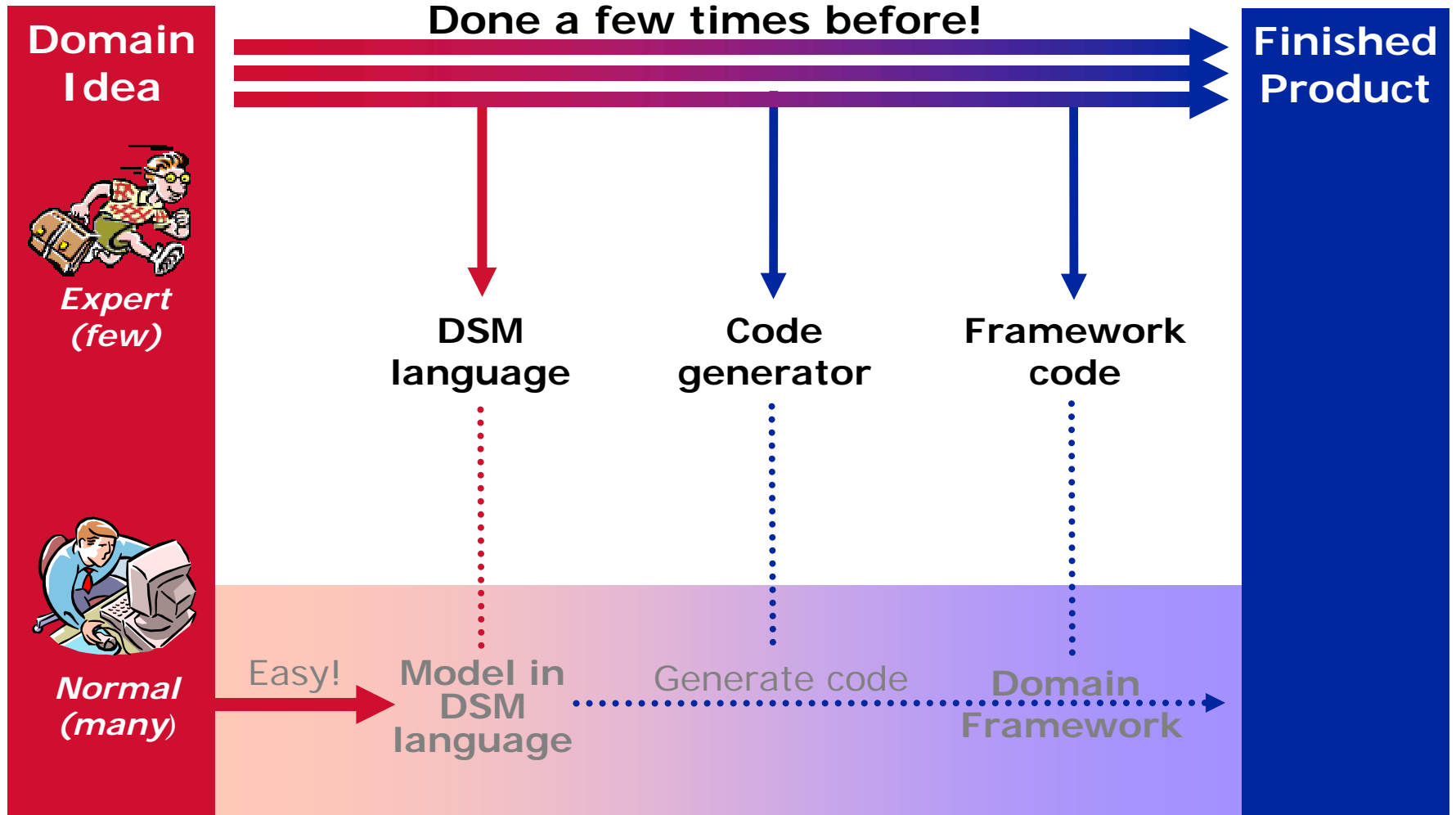
- Domain-Specific にモデリングし、
- Domain-Specific なコードを生成させる

そんなフレームワークを作る
為のツールが、



MetaEdit+ です。

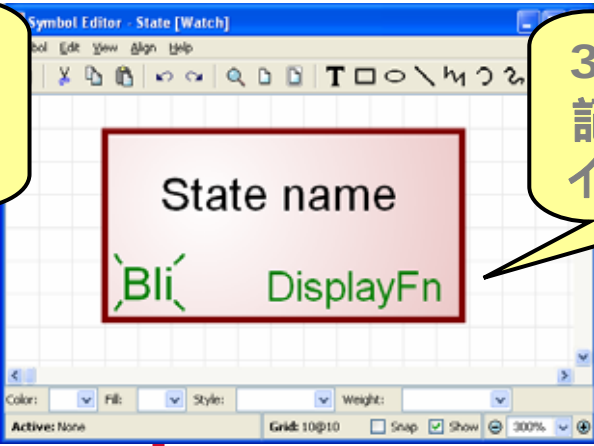
ドメイン精通者がDSM作成 皆は、DSMを使って開発



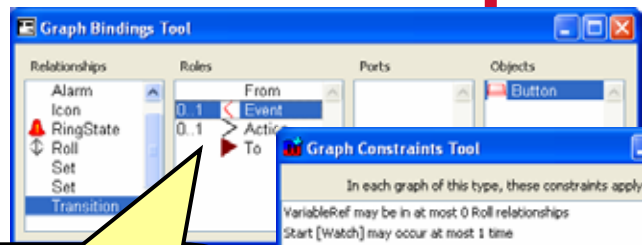
抽象度をモデルに高めて、



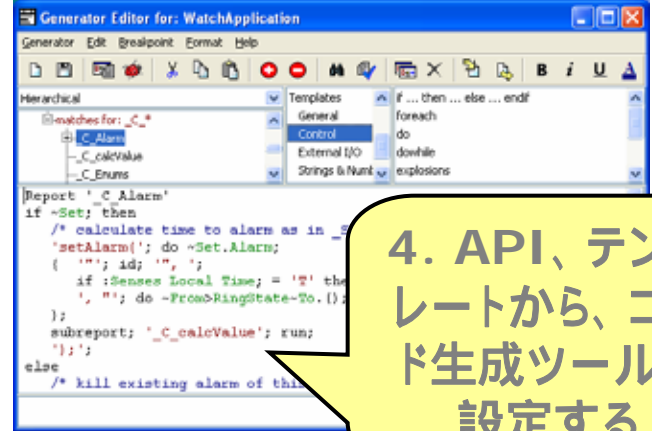
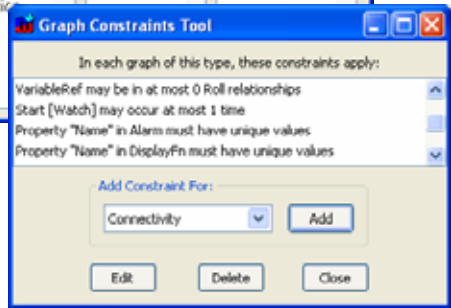
1. 言語のコンセプトとプロパティ(特性)の設定



3. シンボルを記述、もしくはインポートする



2. テンプレートを使ってルールを選択

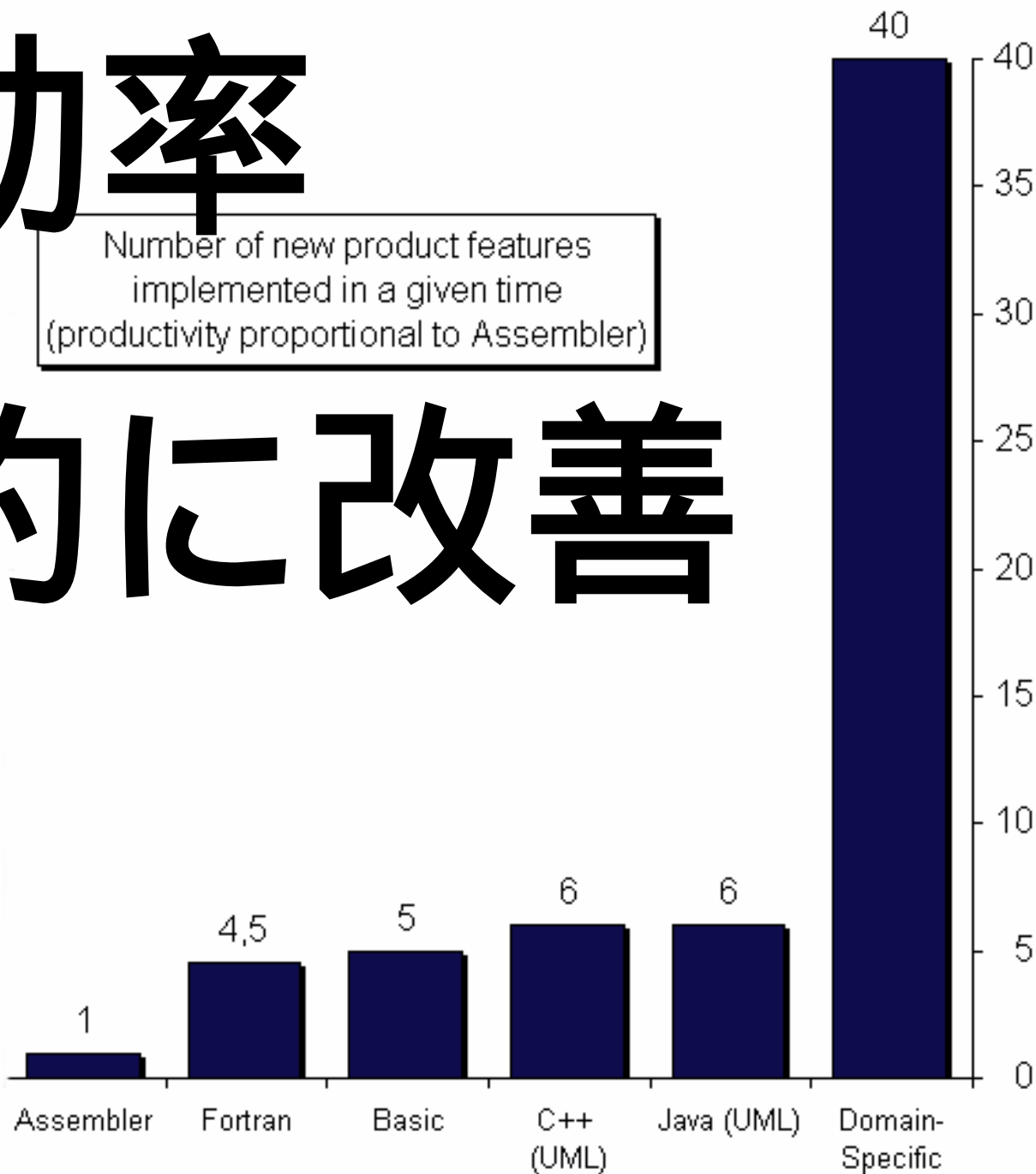


4. API、テンプレートから、コード生成ツールを設定する

開発効率

Number of new product features implemented in a given time
(productivity proportional to Assembler)

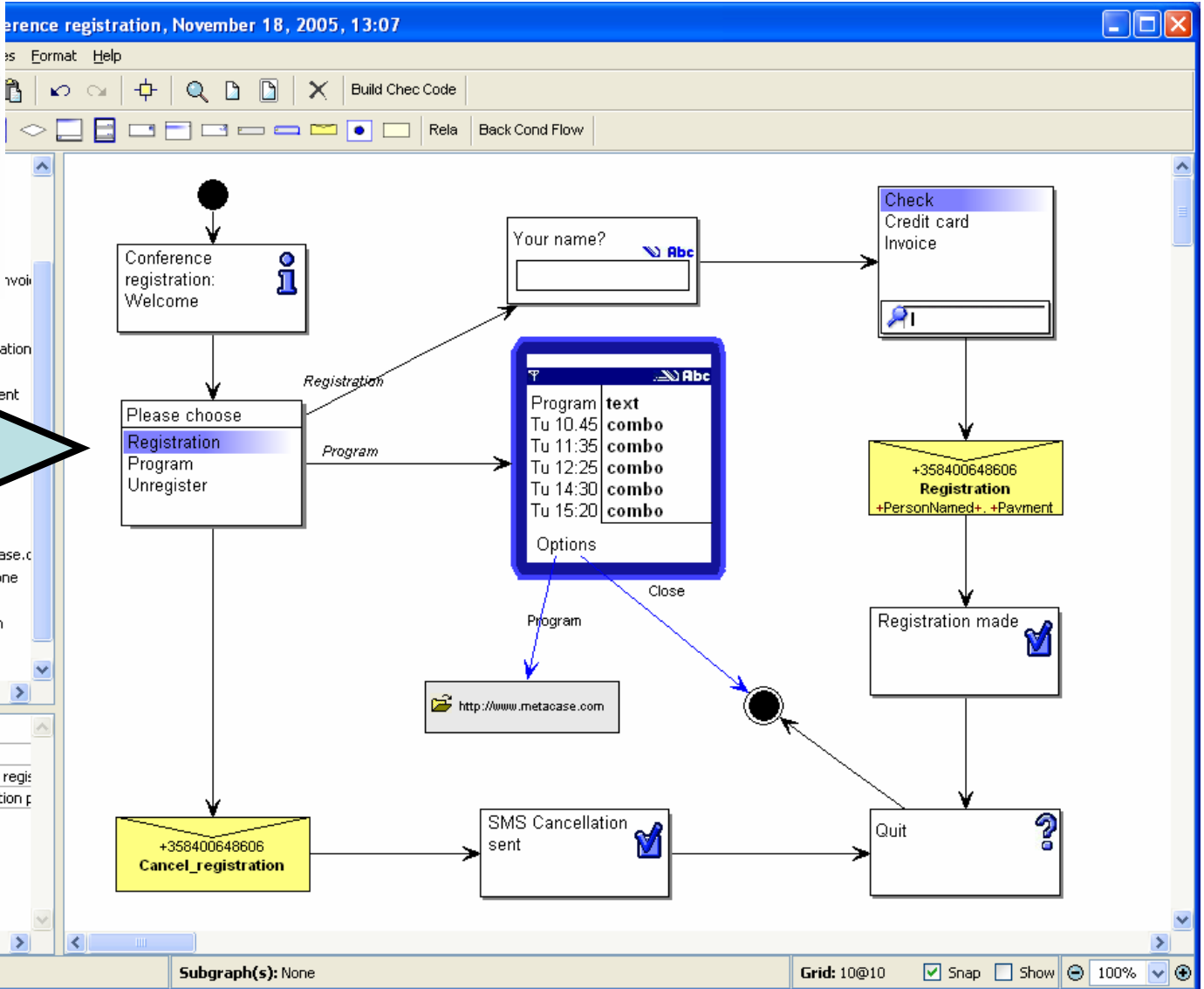
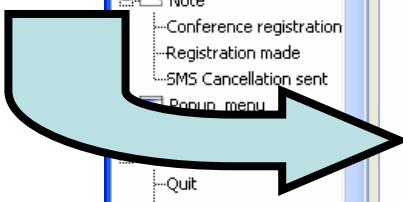
飛躍的に改善



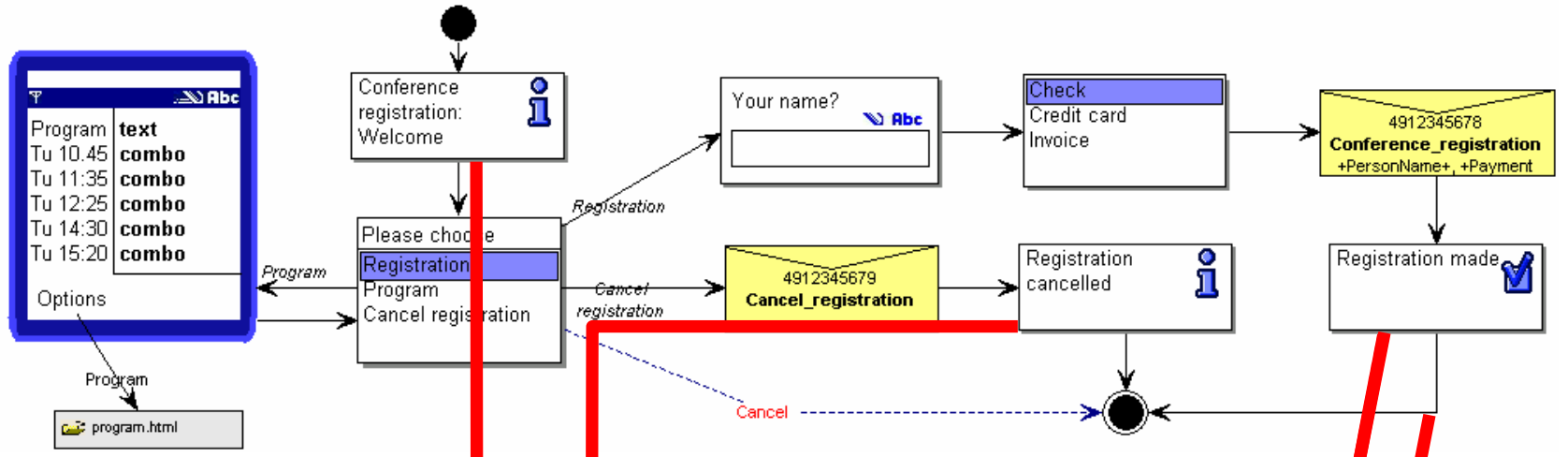
Case1: スマートフォン



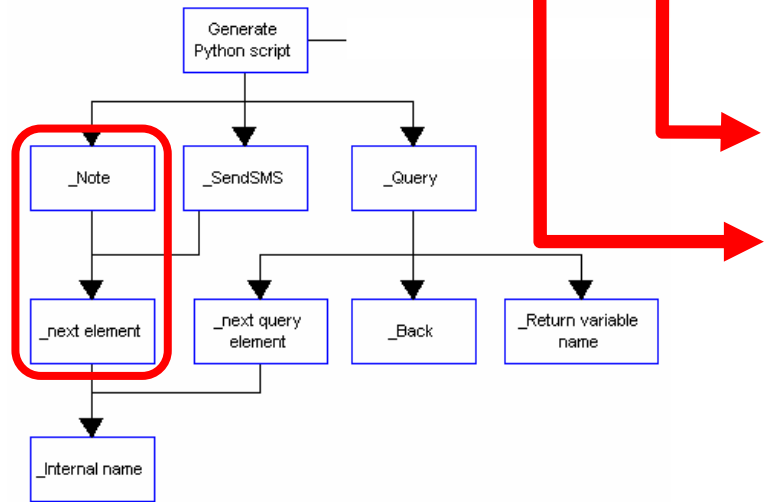
Symbian/Series 60



Function calls – Series 60



Generator definition



Generator output

```
def Note3_2227():
    appuifw.note(u"Registration made", 'conf')
    return Stop3_983

def Note3_6109():
    appuifw.note(u"SMS Cancellation sent", 'conf')
    return Stop3_983

def Note3_2543():
    appuifw.note(u"Conference registration: Welcome", 'info')
    return Popup_menu3_2520

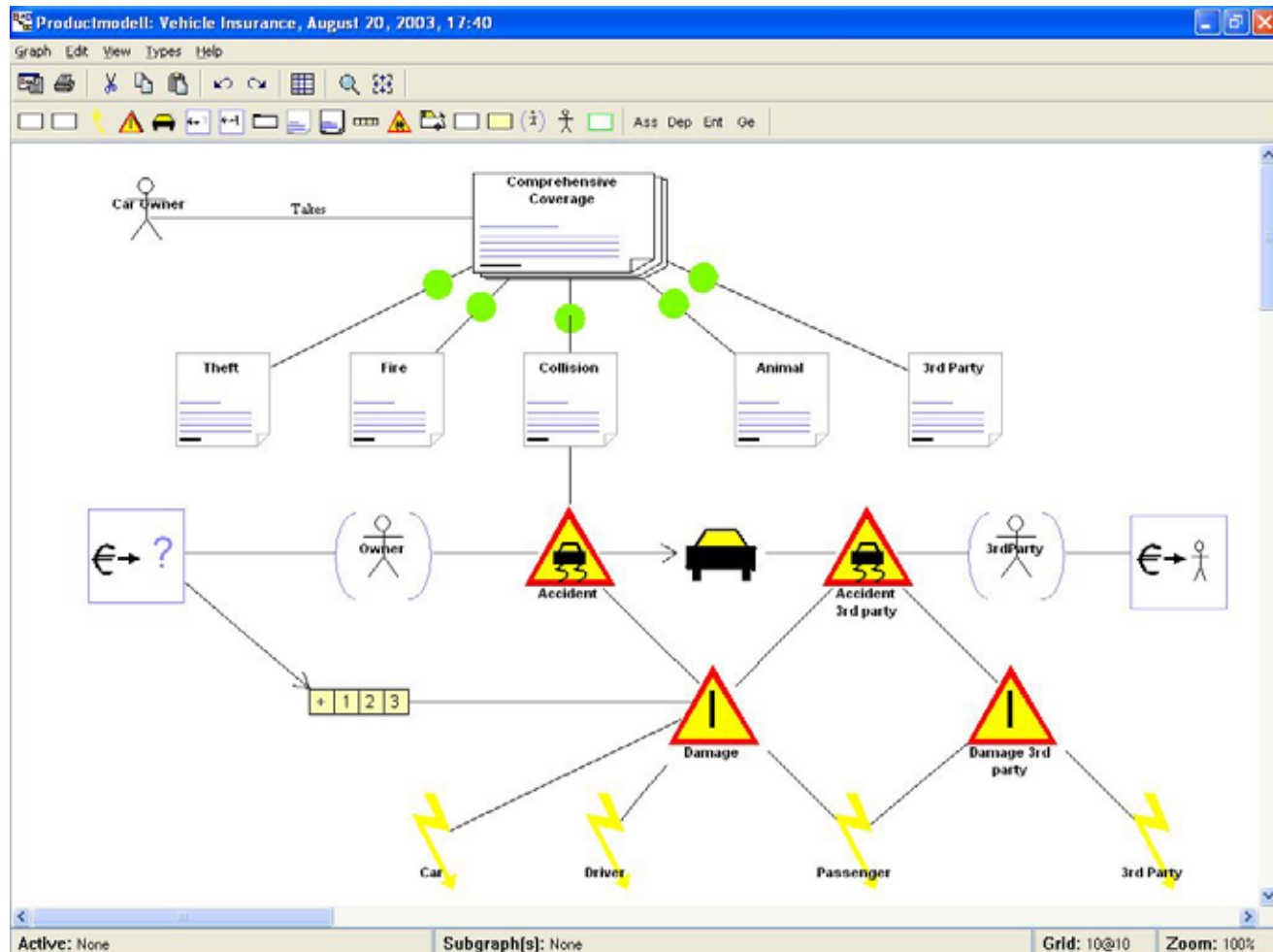
def Stop3_983():
    # This applications stops here
    return appuifw.app.set_exit

...

```



Case2: 保険のeコマースは、



Java で書くより 3倍速くなった

```
Programmer's File Editor - [Basis.java]
File Edit Options Template Execute Macro Window Help
public class Basis extends ProductRepository
{
    public Basis(String name)
    {
        super(name);
        PRODUCT_NAME = Basis;
        HofPackage productpackage = createProduct();
        this.addHofPackage(productpackage);
    }

    public Basis()
    {
        // name of namespace ProductRepository not used
        this(Basis);
    }

    private HofPackage createProduct()
    {
        productpackage_ = new HofPackage(PRODUCT_NAME);

        // Global Instances, will be re-used by each section
        HofAttribute attribute;
        HofAssociation hofAssociation;
        Constant constant;
        AssociationEnd end1;
        AssociationEnd end2;
        Reference reference;

        // =====
        // Tags
        // =====
        beitragszicht_ = new Tag("Tarifizierung", HofModelConstants.TAGID_TARIFIERUNG);
        productpackage_.addContainedTag(beitragszicht_);

        selektionssichttrue_ = new Tag("Selektion_true", HofModelConstants.TAGID_SELEKTION);
        selektionssichttrue_.addValue("True");
        productpackage_.addContainedTag(selektionssichttrue_);

        angebotssicht_ = new Tag("Angebot", HofModelConstants.TAGID_ANGEBOT);
        productpackage_.addContainedTag(angebotssicht_);

        // =====
        // Exceptions
        // =====
        HofException Exception1 = new HofException ("Exception1")
        parameter = new Parameter("ExcepParam1", new DataType("Haftung"));
        .addParameter(parameter)
        parameter = new Parameter("ExceptionParam2", new DataType("string"));
        .addParameter(parameter)

```

Selektionrechner | Kundendaten | Meine Produktpartner | **Meine Stammdaten**

NEU | ÄNDERN | LÖSCHEN | SPEICHERN | ABBRECHEN | HILFE

Daten von: Potter, Harry

- Adressen
 - 1. hohle Gasse 12 (*)
- Kontaktmöglichkeiten
 - 1. Telefon: 00000234 (*)
 - 2. eMail: 1@2 (*)
 - 3. Fax: (*)
- Bankverbindungen
 - 1. 123456a (*)

Hier können Sie Ihre Registrierungsdaten ändern
Bitte senden Sie uns eine e-Mail für Korrekturen in den gesperrten Feldern.

Name:

Vorname:

Pecunet PartnerID:

Anrede: Titel:

Strasse:

PLZ: Land:

Stadt:

Telefon: e-Mail:

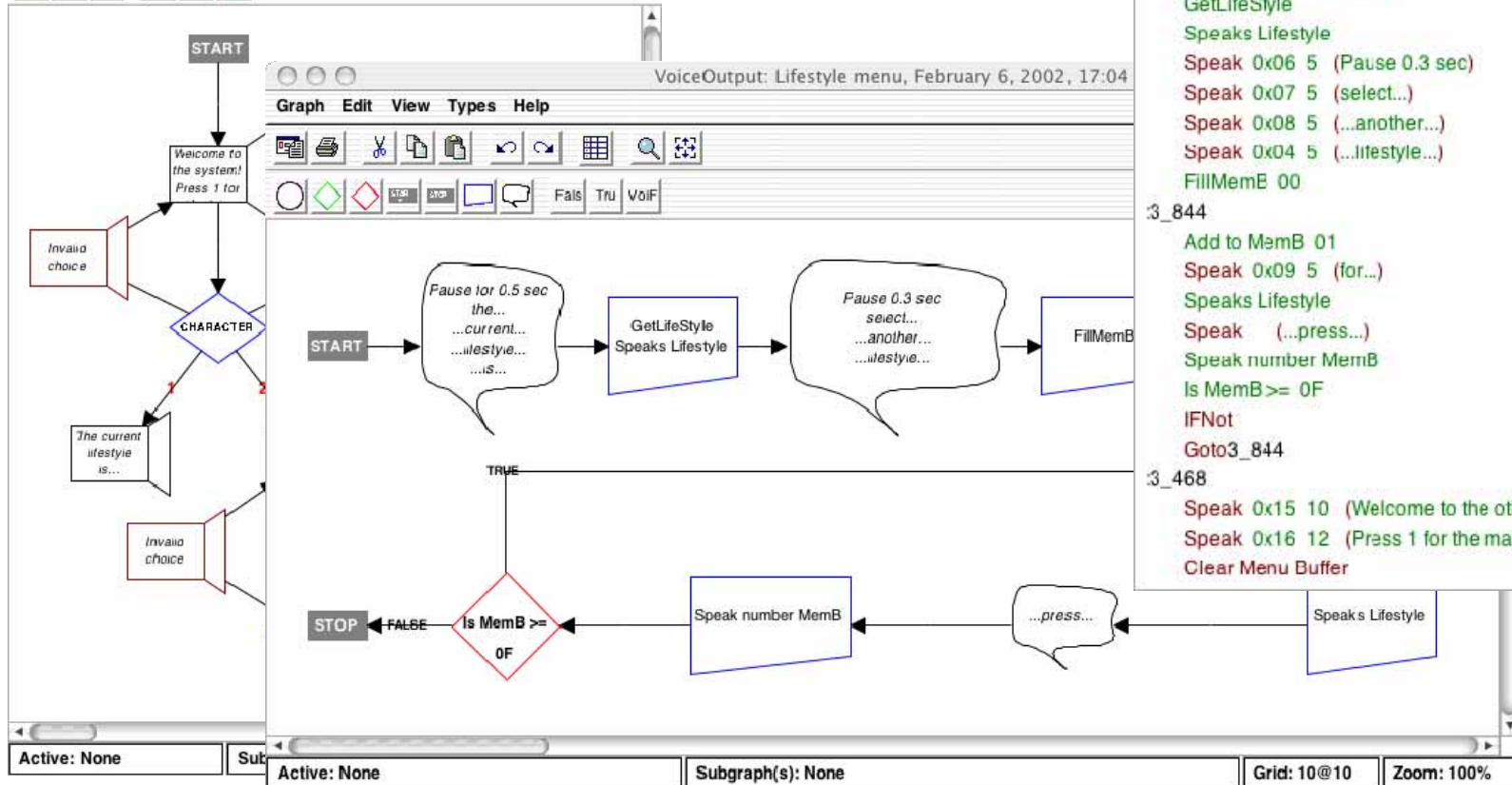
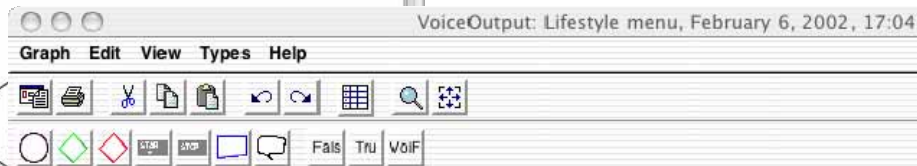
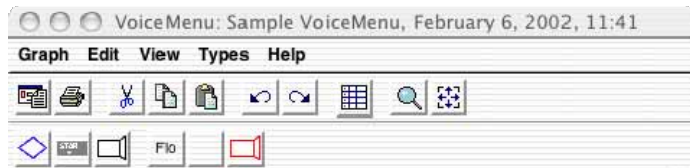
Kontonummer: Bankleitzahl:

Kreditinstitut:

Case3:ホームオートメーションの音声リモコン

- 8ビットマイコンへのアセンブラ
- 開発期間が1週間から1日

Case3: マイクロコントローラの音声メニュー



```
Report Output: Sample VoiceMenu: VoiceMenu
File Edit Help
Goto 3_266
:3_450
Speak 0x01 5 (Pause for 0.5 sec)
Speak 0x02 5 (the...)
Speak 0x03 5 (...current...)
Speak 0x04 5 (...lifestyle...)
Speak 0x05 5 (...is...)
GetLifeStyle
Speaks Lifestyle
Speak 0x06 5 (Pause 0.3 sec)
Speak 0x07 5 (select...)
Speak 0x08 5 (...another...)
Speak 0x04 5 (...lifestyle...)
FillMemB 00
:3_844
Add to MemB 01
Speak 0x09 5 (for...)
Speaks Lifestyle
Speak (...press...)
Speak number MemB
Is MemB >= 0?
IFNot
Goto3_844
:3_468
Speak 0x15 10 (Welcome to the other menu)
Speak 0x16 12 (Press 1 for the main menu,
Clear Menu Buffer
```


**Matsushita
Electric
Works, Ltd.**

“設計、テスト、修正を含めて6時間で
ドメイン固有の設定を行うことができました”
Laurent Safa 氏

SIEMENS

“MetaEdit+ によるモデル化は、
Eclipse に比べて非常に簡単で効果的
であった” Ulf Hesselbarth 氏

“群を抜いて優秀なDSLツールである”
M.Voelter 氏

NOKIA
CONNECTING PEOPLE

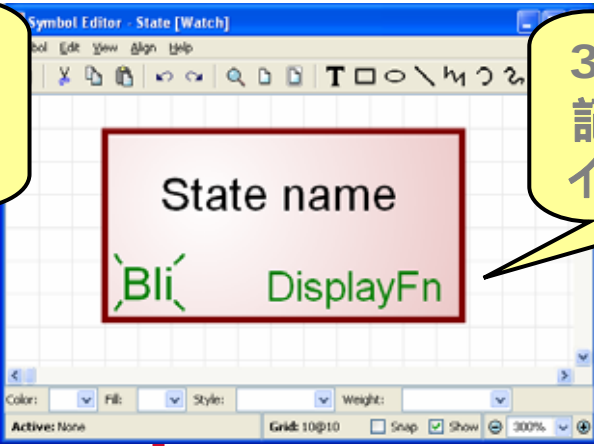
“非常に柔軟性があり、独自のコード
生成ルールを簡単に定義することが
できた” D.Narraway 氏

“最も洗練されたDSMツールです”
Scott Ambler 氏

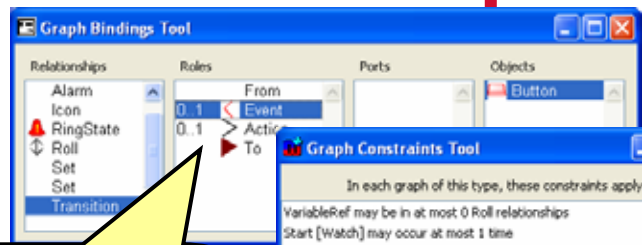
DSMを作成するプロセス



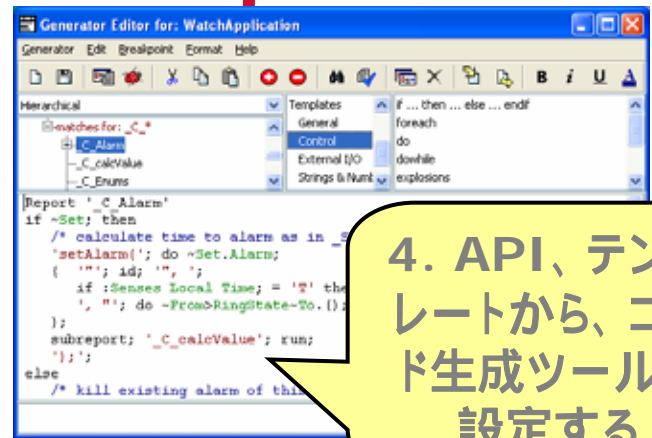
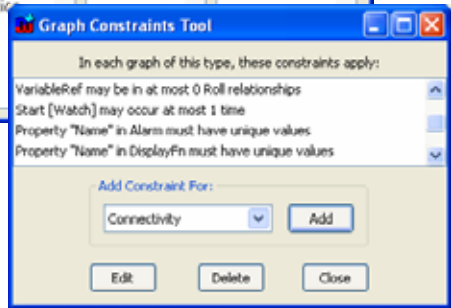
1. 言語のコンセプトとプロパティ(特性)の設定



3. シンボルを記述、もしくはインポートする



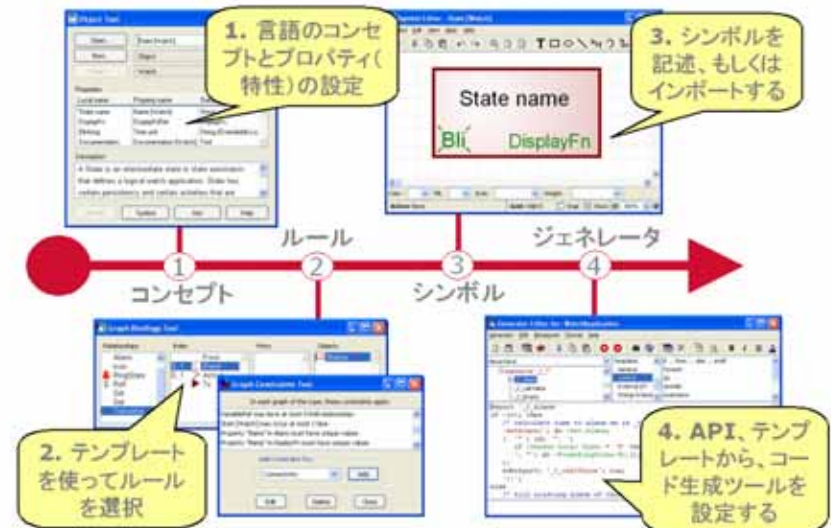
2. テンプレートを使ってルールを選択



4. API、テンプレートから、コード生成ツールを設定する

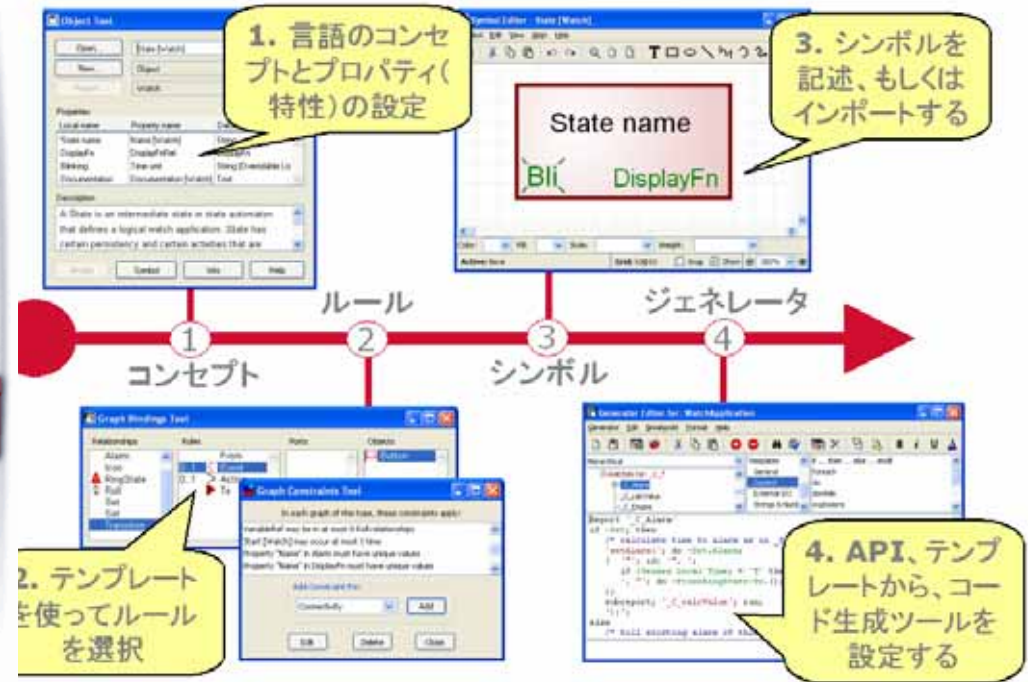
作成されるべきコードを あらかじめ十二分に静的 解析、単体テストすること で、

DSMを作成するプロセス



完成度の高い部品の組合せとしてシステムを構築できる

DSMを作成するプロセス



ありがとうございました。
いきました。



FUJI SETSUBI