

MATLAB EXPO

Japan

MW-04_モデルベース検証 x CI/CTソリューション (1/2)

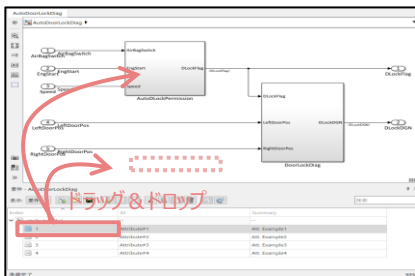
MathWorksのソフトウェア検証の
トータルソリューションのご紹介



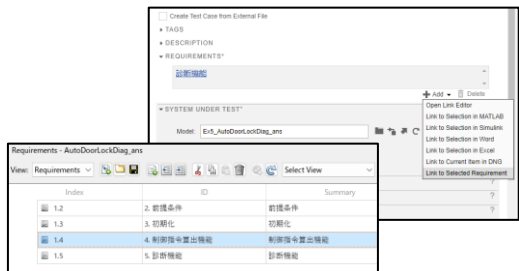
要件トレーサビリティ

モデル・テストとの紐付け

Simulink要素とのリンク設定



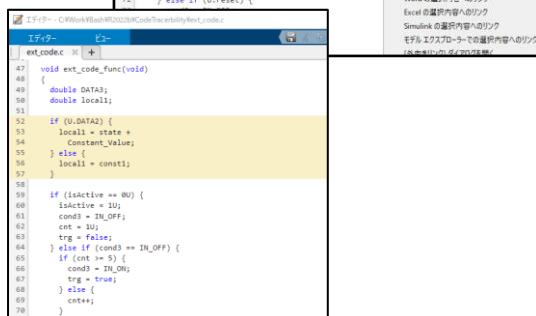
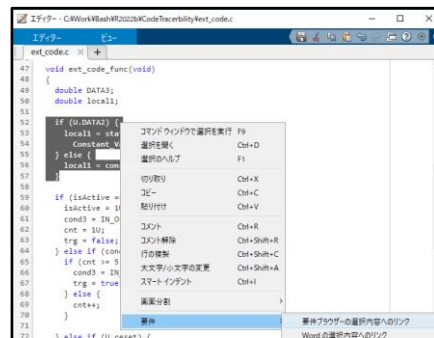
テストとのリンク設定



- ドラッグ & ドロップにより要件を Simulink要素にリンクできる
- 選択した要件をテストマネージャーのテストにリンクできる

Reqs Tbx

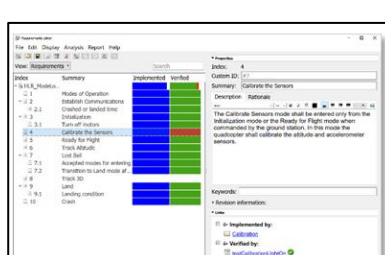
外部テキストとの紐付け



- MATLABエディタで外部テキストファイルを開くことで、選択した要件を指定範囲(行)にリンクできる

Reqs Tbx

実装・検証ステータス



Summary	Implemented	Verified
Driver Switch Request Handling	■	■
Switch precedence	■	■
Avoid repeating commands	■	■
Long Switch recognition	■	■
Cancel Switch Detection	■	■
Set Switch Detection	■	■
Enable Switch Detection	■	■
Resume Switch Detection	■	■
Increment Switch Detection	■	■

実装ステータス

- 実装済み
- 正当化
- リンクなし

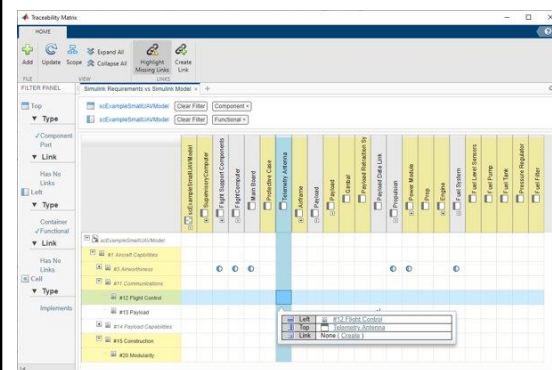
検証ステータス

- 合格
- 失敗
- テスト未実施
- 正当化
- テストなし

- 実装ステータスによって、要件の実装漏れに気付くことができる
- 検証ステータスでは、要件のテスト状況を可視化することができる

Reqs Tbx

トレーサビリティマトリクス



- アーティファクト間のトレーサビリティを確認できる
- マトリクス上で複数リンクの一括作成・削除できる
- マトリクスからレポートを生成できる

Reqs Tbx

形式的な要件の定義

要件テーブル

When a car become the stopped state the engine should transit to the idle stop state.

自然言語仕様書

論理式を用いた形式化

条件	期待値
$u1 \leq 20 \ \&\& \ u2 < 10$	$y1 = \text{true}$
$u1 \leq 20 \ \&\& \ u2 \geq 10$	$y1 = \text{false}$
$(10 \leq u1 < 20) \ \&\& \ u3 == 0$	$y2 = 1$

要件テーブルへの入力

要件	仮定	前提条件	事後条件	アクション		
インデックス	概要	u1	u2	持続時間	y1	y2
*1	Req 1	≤ 20				
1.1	Req 1-1	< 10			true	
		≥ 10			false	
		[10,20)	$u3 == 0$			1

$A \Rightarrow B$

Requirements Table

完全性・一貫性解析

解析

修正

不整合の問題

不整合 1: 次の入力に関して、'y2' には時間 0 で要件 2 と 3 についての手番があります。

時間 0

ステップ 1

u1 15

u2 -

u3 0

不完全性の問題

不完全性 1: 次の入力について、'y1' が時間 0 で指定されていません。

時間 0

ステップ 1

u1 5.9566

u2 1.9184

u3 -5.4067

不完全性 2: 次の入力について、'y2' が時間 0 で指定されていません。

時間 0

ステップ 1

u1 5.9566

u2 1.9184

u3 -5.4067

要件テーブルのデバッグ

Index	Summary
1	Conditions for the first roll
1.1	If the player rolls a 11 on the first roll
1.2	If the player rolls a 12 on the first roll
1.3	Otherwise, establish the

Requirements Table Bre... X

Before Checking Precondition

After Precondition is Valid

[Breakpoints List](#)

Collapse All

Comment Out

Set Breakpoint

Remove Links

Open in Requirements Editor

R2023a

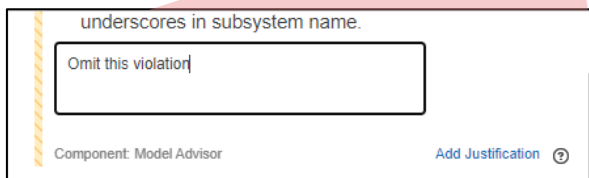
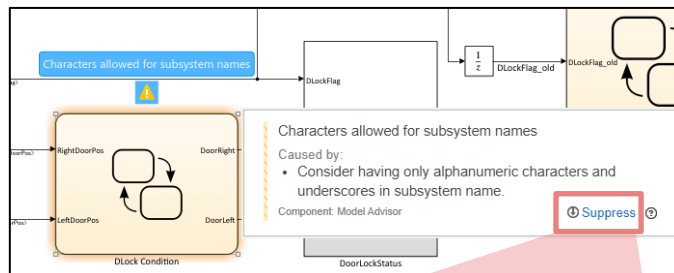
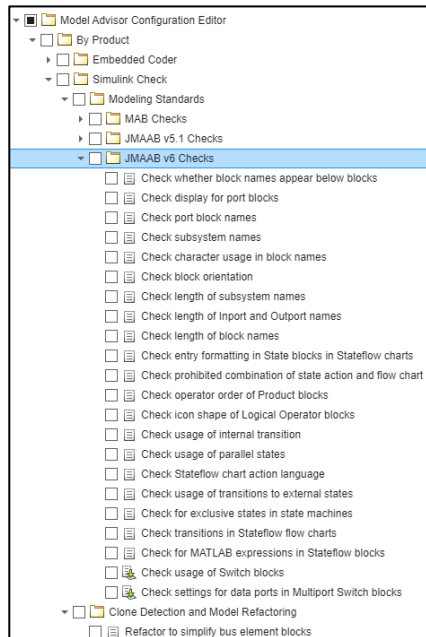
- テーブルの各項目に論理式を当てはめることで、形式的に要件を定義できる
- 要件を階層的に表現できるので、親子関係を明示的に示すことができる

- SLDVの解析機能を用いて、テーブルブロック内に潜む問題を検出できる
 - 完全性: 考慮すべき前提条件の抜け漏れ
 - 一貫性: 複数要件間の矛盾した定義
- 反復的な解析・修正フローによって、仕様書自体の問題に気付くこともできる

- テーブルの任意の行にブレイクポイントを設定し、シミュレーションを一時停止できる
- シミュレーションを一時停止するタイミングは「前提条件の評価前」、「事後条件の成立後」の2種類から選択できる

モデリングルールチェック

編集時チェック



- モデルの編集と同時にモデリングチェックを自動的に実行できる (違反検出時、Simulinkエディター上から詳細結果にアクセスできる)
- モデルアドバイザに実装されている一部のチェックスクリプトのみをサポート
- Simulinkエディター上で違反の正当化を実施できる
- APIを用いてオリジナルチェックプログラムを作成・実装できる(R2022a)

モデルアドバイザ

Check ID: mathworks.maab.jc_0131
jc_0131: Usage of Relational Operator blocks.

Identify Relational Operator blocks that connect to constants with the first (upper) input value.

Summary
Status: ▲ Warning

Report **Result Details**

Identify Relational Operator blocks that connect to constants with the first (upper) input value.

Warning
The following Relational Operator blocks connect to a constant value using the first (upper) input value

- ..._DoorLockDiag_DoorLockStatus_RelationalOperator1
- ..._DoorLockDiag_DoorLockStatus_RelationalOperator2

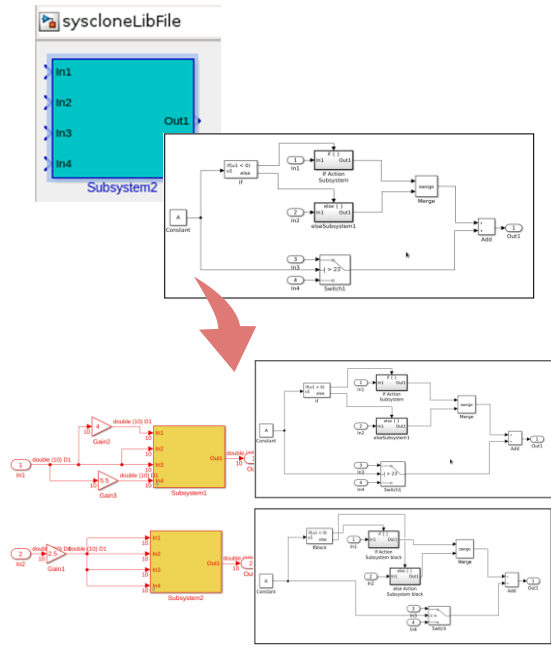
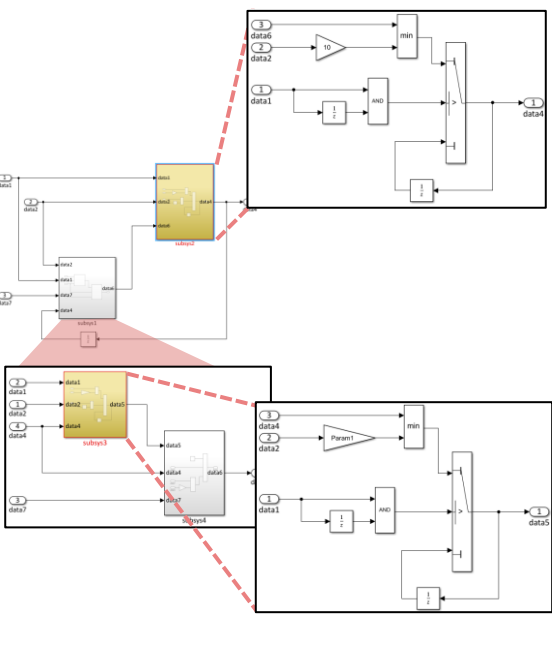
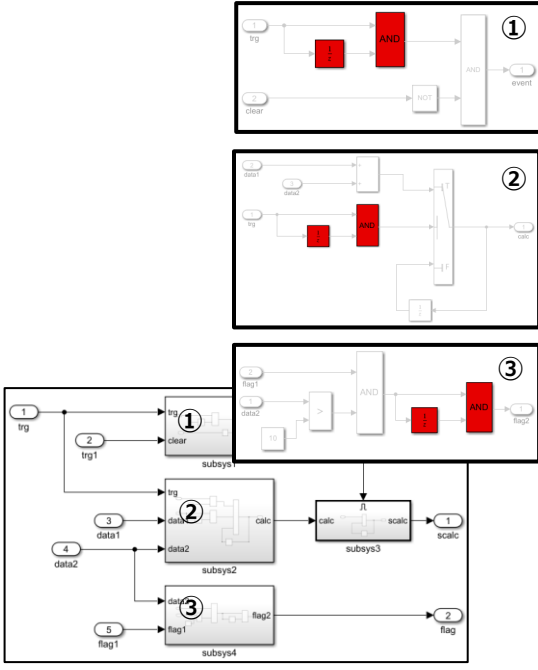
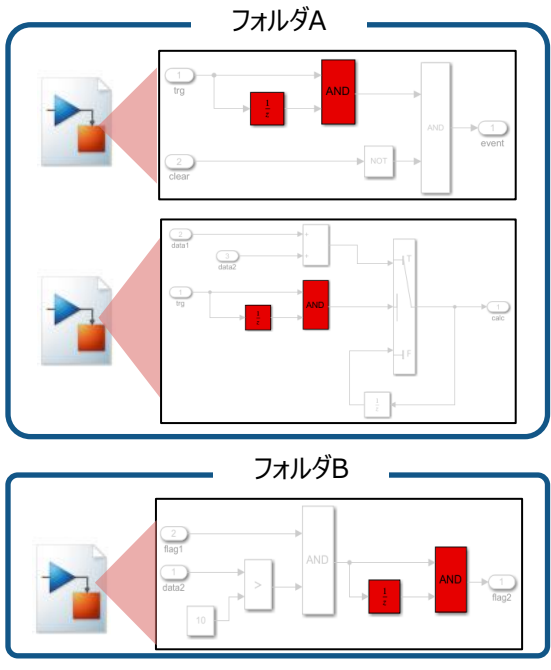
Recommended Action
Make the constant value the second (lower) input.

Report	Result Details															
1	<table border="1"> <thead> <tr> <th>Status</th> <th>Failing Element</th> <th>Description</th> <th>Recommended Action</th> <th>Justifications</th> </tr> </thead> <tbody> <tr> <td>▲</td> <td>..._DoorLockDiag_DoorLockStatus_RelationalOperator1</td> <td>Identify Relational Operator</td> <td>Make the constant value the</td> <td>Omit this violation</td> </tr> <tr> <td>▲</td> <td>..._DoorLockDiag_DoorLockStatus_RelationalOperator2</td> <td>Identify Relational Operator</td> <td>Make the constant value the</td> <td>Omit this violation</td> </tr> </tbody> </table>	Status	Failing Element	Description	Recommended Action	Justifications	▲	..._DoorLockDiag_DoorLockStatus_RelationalOperator1	Identify Relational Operator	Make the constant value the	Omit this violation	▲	..._DoorLockDiag_DoorLockStatus_RelationalOperator2	Identify Relational Operator	Make the constant value the	Omit this violation
Status	Failing Element	Description	Recommended Action	Justifications												
▲	..._DoorLockDiag_DoorLockStatus_RelationalOperator1	Identify Relational Operator	Make the constant value the	Omit this violation												
▲	..._DoorLockDiag_DoorLockStatus_RelationalOperator2	Identify Relational Operator	Make the constant value the	Omit this violation												

Comments: コメントを入力することで正当化できる

- 作成したモデルのモデリングルールの準拠度合いを自動チェックできる
- モデルのレビュー時に必要となる解析結果レポートが生成できる
- 検出された違反の正当化することもできる
- JMAAB Ver.6のモデリングチェックに対応済み(R2023a)
- APIを用いてオリジナルチェックプログラムを作成・実装できる

モデルリファクタリング

ライブラリクローン検出	サブシステムクローン検出	ブロック構造クローン検出	複数モデル間のクローン検出
			
<ul style="list-style-type: none"> 指定したライブラリに対して、厳密・類似するサブシステムを検出できる 	<ul style="list-style-type: none"> モデル内の厳密・類似サブシステムを検出できる 	<ul style="list-style-type: none"> モデル内の厳密・類似するブロック構造を検出できる 	<ul style="list-style-type: none"> 複数フォルダをまたいで複数モデル間のクローンを検出できる (APIのみ対応)

SLCheck

SLCheck

SLCheck

SLCheck

ランタイムエラー・デッドロジック検出

ランタイムエラー検出

Results: sldvdemo_design_error_detection

sldvdemo_design_error_detection/Controller/Sum

Integer overflow Objectives
Overflow Valid
Derived Ranges:
Output 1:[1..255.99609375]

Results: sldvdemo_design_error_detection

sldvdemo_design_error_detection/Controller/Sum2

Integer overflow Objectives
Overflow Error - needs simulation
Derived Ranges:
Output 1:[0..255.99609375]

- 範囲外配列アクセス
- ゼロ除算
- 整数のオーバーフロー
- 非有限で NaN の浮動小数点値
- 非正規浮動小数点値
- 指定された最小値と最大値の違反
- データストアのアクセス違反

- モデルを静的解析し、ランタイムエラーが起きるかどうかを検出できる
- ランタイムエラーを検出した場合、それを再現するための反例テストケースが自動生成される

反例のデバッグ

Results: sldvdemo_design_error_detection

sldvdemo_design_error_detection/Controller/Sum2

Integer overflow Objectives
Overflow Error - needs simulation
Derived Ranges:
Output 1:[0..255.99609375]

Model Slicer setup is now complete. Use the Step Back/Step Forward buttons to debug.

Active Control

Determine if the control is active

Active last step boolean

Compute the target speed

- ランタイムエラーが発生した際のブロックの実行部位 および 関連するブロックの出力を可視化できる
- シミュレーションのステップ実行により、任意のタイミングでのモデルの振る舞いも確認できる

デッドロジック検出

Speed_Sensor_Mode

[speed==0 & press < zero_thresh]/ Sens_Failure_Counter.INC

speed_norm entry: fail_state[SPEED] = 0

speed_fail entry: fail_state[SPEED] = 1

[speed > 0] / Sens_Failure_Counter.DEC

Results: sldvdemo_fuelsys_logic_simple

Transition "[speed==0 & press < zero_th..." from "speed_norm" to "speed_fail"

DEAD LOGIC:
"press < zero_thresh" can only be true

- デッドロジック
- アクティブロジックの特定

- モデル内の到達不能なロジック(デッドロジック)を静的解析により検出できる
- デッドロジックが検出された場合、その原因が表示されるので、モデルの修正を効率的に実施できる

テスト自動生成

モデル・コードカバレッジ テストケース生成

TEST GENERATION TARGET

- Model
Generate tests for top model
- Code Generated as Top Model
Generate tests for code generated as top model
- Code Generated as Model Reference
Generate tests for code generated as model reference

実行状況

16-Sep-2021 15:13:11
このモデルは、EPAの自動ロック診断モデルです。EPAの自動ロック診断モデルのテストケースを生成しています。EPAの自動ロック診断モデルのテストケースを生成しています。

16-Sep-2021 15:13:21
このモデルは、EPAの自動ロック診断モデルです。EPAの自動ロック診断モデルのテストケースを生成しています。EPAの自動ロック診断モデルのテストケースを生成しています。

- モデル または コード内のすべての分岐・論理入力の組み合わせを網羅するテストケースを自動生成できる
- 生成したテストケースをExcelにエクスポートすることもできる

要件網羅テストケース生成

要件網羅テストケース生成

インデックス

インデックス	観測	前提条件	アクション
1	Initialize case	isStartup	cnt = 0, y1 = false
2	Normal case	~isStartup	
2.1	Increment	>=10	prev(cnt)+0.1
2.2	reset	<10	0
2.3	Output calc (Achieved)	true, true, prev(cnt) >= 9	true
2.4	Output calc (Non-Achieved)	prev(cnt) < 9	false
		false	false
		false	false

不整合の問題
不整合の問題は見つかりませんでした。

不完全性の問題
不完全性の問題は見つかりませんでした。

- 要件テーブル内のすべての要件を網羅するテストケースを自動生成できる
- 抜け漏れ・矛盾がない要件テーブルからのみ、テストケースを自動生成できる

補完テストケース生成

補完テストケース生成

既存のテストケース + 補完テストケース = カバレッジ計測

補完テストケース生成

カバレッジ計測結果

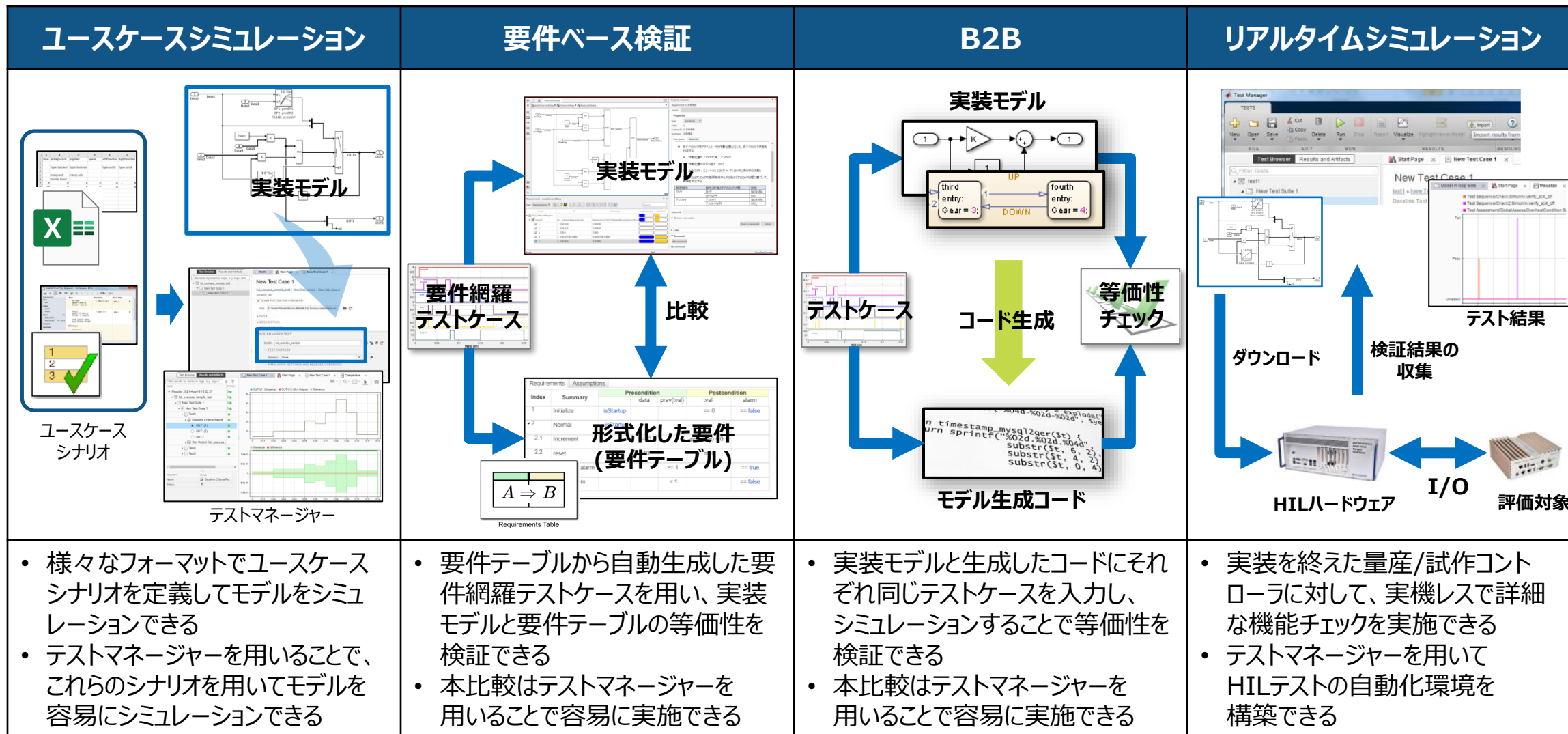
計測された(単体・累積)カバレッジの確認

テストマネージャーのカバレッジ計測結果

累積カバレッジ測定結果

- 既存のテストケースではカバーできなかった未達カバレッジ部位を補完するテストケースを追加生成できる
- テストマネージャーを用いることで、補完テストケース生成を容易に実行できる

シミュレーション・検証



SLTest

SLTest

Reqs Tbx

SLDV

SLTest

SLDV

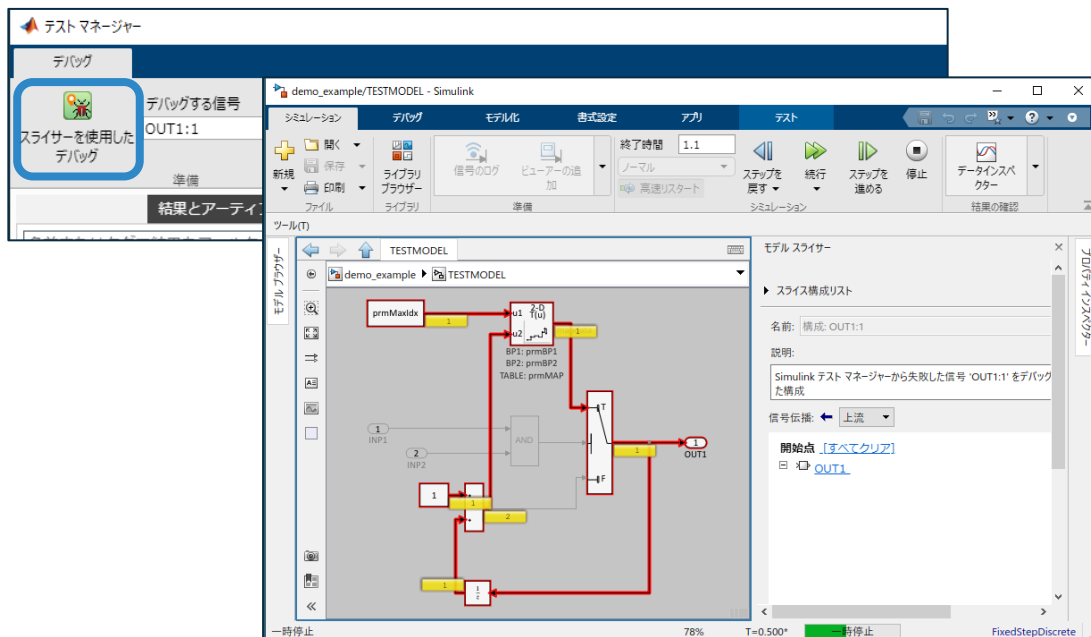
SLTest

モデル/Cカバレッジ測定

モデルカバレッジ計測	コードカバレッジ計測	カバレッジレポート生成	カバレッジフィルタ作成																																																							
	<pre> 32 RT_MODEL_easy_code_cov_19b_T *const easy_code_cov_19b_M = 8 33 34 /* Model step function */ 35 void easy_code_cov_19b_step(void) 36 { 37 /* Switch: '<Root>/Switch' incorporates: 38 * Import: '<Root>/In1' 39 * Import: '<Root>/In2' 40 * Logic: '<Root>/Logical Operator' 41 */ 42 if (In1 In2) { 43 /* Switch: '<Root>/Switch' incorporates: 44 * Constant: '<Root>/Constant' 45 */ 46 Out1 = 1.0; 47 } else { 48 /* Switch: '<Root>/Switch' incorporates: 49 * Gain: '<Root>/Gain' */ } } </pre> <table border="1"> <thead> <tr> <th colspan="3">Simulink Coverage</th> </tr> <tr> <th></th> <th>Condition: 75%</th> <th>MC/DC: 50%</th> </tr> </thead> <tbody> <tr> <td>Decision: 100%</td> <td></td> <td></td> </tr> <tr> <td>Statement: 100%</td> <td>Function: 100%</td> <td></td> </tr> </tbody> </table>	Simulink Coverage				Condition: 75%	MC/DC: 50%	Decision: 100%			Statement: 100%	Function: 100%		<p>Summary</p> <p>Model Hierarchy/Complexity Run 1</p> <table border="1"> <thead> <tr> <th></th> <th>Decision</th> <th>Condition</th> <th>MCDC</th> <th>Execution</th> </tr> </thead> <tbody> <tr> <td>1. easy_cov</td> <td>100%</td> <td>100%</td> <td>0%</td> <td>100%</td> </tr> </tbody> </table> <p>Details</p> <p>1. Model "easy_cov"</p> <p>Metric Cyclomatic Complexity Decision Condition MCDC Execution</p> <p>Logic block "Logical Operator"</p> <p>Parent: /easy_cov</p> <p>Metric Coverage</p> <p>Cyclomatic Complexity 0</p> <p>Condition 100% (4/4) condition outcomes</p> <p>MCDC 0% (0/2) conditions reversed the outcome</p> <p>Execution 100% (1/1) objective outcomes</p> <table border="1"> <thead> <tr> <th colspan="3">Conditions analyzed</th> </tr> <tr> <th>Description</th> <th>True</th> <th>False</th> </tr> </thead> <tbody> <tr> <td>input port 1</td> <td>7</td> <td>4</td> </tr> <tr> <td>input port 2</td> <td>7</td> <td>4</td> </tr> </tbody> </table> <p>MC/DC analysis (combinations in parentheses did not occur)</p> <table border="1"> <thead> <tr> <th>Decision/Condition</th> <th>True Out</th> <th>False Out</th> </tr> </thead> <tbody> <tr> <td>expression for output</td> <td></td> <td></td> </tr> <tr> <td>input port 1</td> <td>(TF)</td> <td>FF</td> </tr> <tr> <td>input port 2</td> <td>(FT)</td> <td>FF</td> </tr> </tbody> </table>		Decision	Condition	MCDC	Execution	1. easy_cov	100%	100%	0%	100%	Conditions analyzed			Description	True	False	input port 1	7	4	input port 2	7	4	Decision/Condition	True Out	False Out	expression for output			input port 1	(TF)	FF	input port 2	(FT)	FF	<p>カバレッジフィルタ</p> <table border="1"> <thead> <tr> <th>判定</th> <th>条件</th> <th>MCDC</th> </tr> </thead> <tbody> <tr> <td>83%</td> <td>100%</td> <td>100%</td> </tr> </tbody> </table> <table border="1"> <tbody> <tr> <td>100%</td> <td>100%</td> <td>100%</td> </tr> </tbody> </table>	判定	条件	MCDC	83%	100%	100%	100%	100%	100%
Simulink Coverage																																																										
	Condition: 75%	MC/DC: 50%																																																								
Decision: 100%																																																										
Statement: 100%	Function: 100%																																																									
	Decision	Condition	MCDC	Execution																																																						
1. easy_cov	100%	100%	0%	100%																																																						
Conditions analyzed																																																										
Description	True	False																																																								
input port 1	7	4																																																								
input port 2	7	4																																																								
Decision/Condition	True Out	False Out																																																								
expression for output																																																										
input port 1	(TF)	FF																																																								
input port 2	(FT)	FF																																																								
判定	条件	MCDC																																																								
83%	100%	100%																																																								
100%	100%	100%																																																								
<ul style="list-style-type: none"> 入力データに対するモデルのカバレッジを測定できる 測定後のモデルの各ブロックの背景色に基づいて測定結果を把握できる 	<ul style="list-style-type: none"> 入力データに対するコードのカバレッジを測定できる 計測対象のブロックをクリックすると、生成コードの該当行を確認できる 	<ul style="list-style-type: none"> カバレッジ計測結果からレポートを生成できる レポートのハイパーリンクをクリックすることで該当するブロックにリンクできる 	<ul style="list-style-type: none"> カバレッジ測定対象から除外するフィルタファイルを生成できる 																																																							

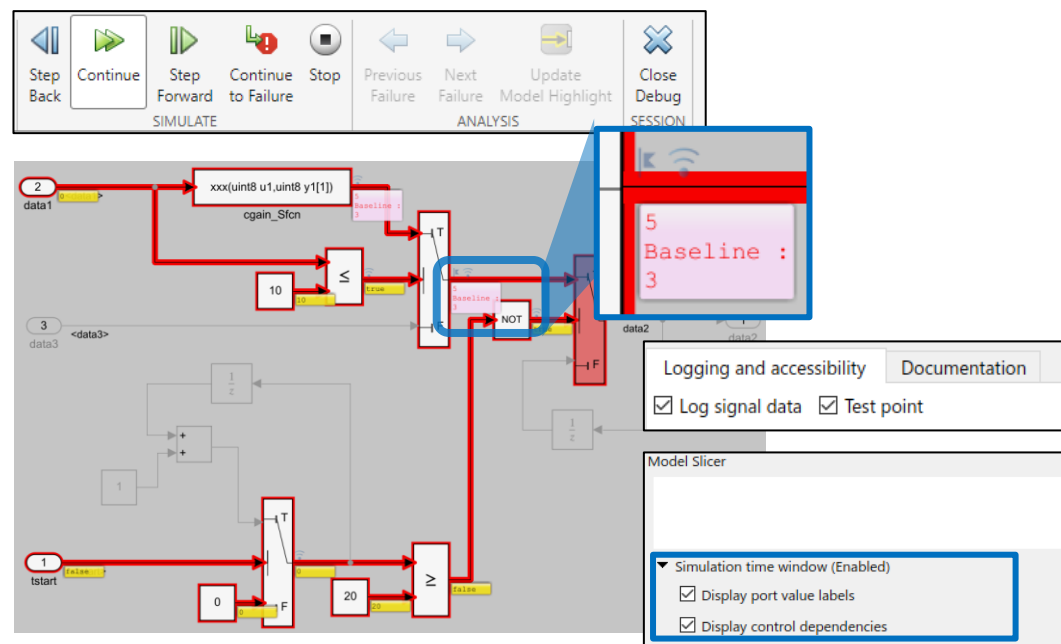
実行パス解析

ユースケースシミュレーション



- テストマネージャーで期待値とモデルの動作が一致しなかった場合、モデル内の実行部位の可視化 および 各ブロック出力を確認しながらモデルをデバッグできる
- シミュレーションのステップ実行もサポートされており、任意のタイミングでのモデルの振る舞いも確認できる

B2B



- B2Bにおいてそれぞれの出力が一致しなかった場合、実行部位の可視化 および 各ブロック毎のモデルとコードの出力を表示できる
- シミュレーションのステップ実行もサポートされており、任意のタイミングでのモデルとコードの振る舞いも確認できる

MATLAB EXPO

JAPAN

Thank you



© 2023 The MathWorks, Inc.