

# MathWorksの要件ベースのソフトウェア開発ソリューション

MathWorks Japan  
アプリケーションエンジニアリング部（制御）  
シニアアプリケーションエンジニア  
大越 亮二

# はじめに

## 本Webセミナーの対象者

- **MBDによる製品開発に従事されている方**  
(対象エリア: 主に制御ソフトウェア開発領域)
- **機能安全等各規格要求に対応すべく要件と各種成果物のトレーサビリティに関するMathWorksツールの使いどころを知りたい方**

## 本Webセミナーでお伝えしたいこと

**MBDによる要件ベースのソフトウェア開発を進める上で  
MBD各開発工程において有用と思われるMathWorksツールの機能・使い方**

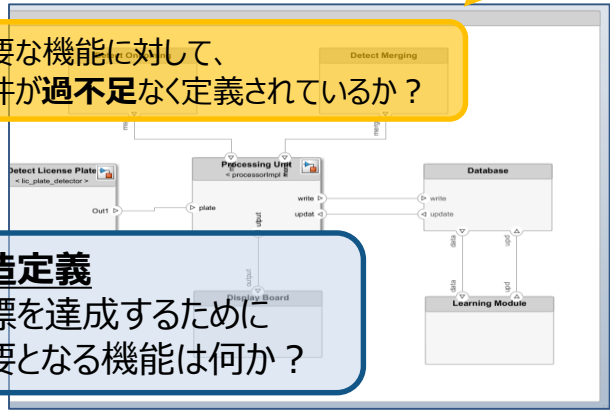
# 要件ベースによるソフトウェア開発のポイント

## 1. Simulink Requirementsによる要件定義/管理

**要件定義**  
各機能を実現するための要件は？

ID	Summary	Implemented	Verified
#1	Driver Switch Request Handling	<div style="width: 100%; background-color: blue;"></div>	<div style="width: 100%; background-color: green;"></div>
#2	Switch precedence	<div style="width: 100%; background-color: blue;"></div>	<div style="width: 100%; background-color: green;"></div>
#3	Avoid repeating commands	<div style="width: 100%; background-color: blue;"></div>	<div style="width: 100%; background-color: green;"></div>
#4	Long Switch recognition	<div style="width: 100%; background-color: blue;"></div>	<div style="width: 100%; background-color: green;"></div>
#7	Cancel Switch Detection	<div style="width: 100%; background-color: blue;"></div>	<div style="width: 100%; background-color: red;"></div>
#8	Set Switch Detection	<div style="width: 100%; background-color: blue;"></div>	<div style="width: 100%; background-color: green;"></div>
#9	Enable Switch Detection	<div style="width: 100%; background-color: blue;"></div>	<div style="width: 100%; background-color: green;"></div>
#10	Resume Switch Detection	<div style="width: 100%; background-color: blue;"></div>	<div style="width: 100%; background-color: green;"></div>
#11	Increment Switch Detection	<div style="width: 100%; background-color: blue;"></div>	<div style="width: 100%; background-color: green;"></div>
#15	Decrement Switch Detection	<div style="width: 100%; background-color: blue;"></div>	<div style="width: 100%; background-color: green;"></div>
#19	Cruise Control Mode	<div style="width: 100%; background-color: blue;"></div>	<div style="width: 100%; background-color: green;"></div>

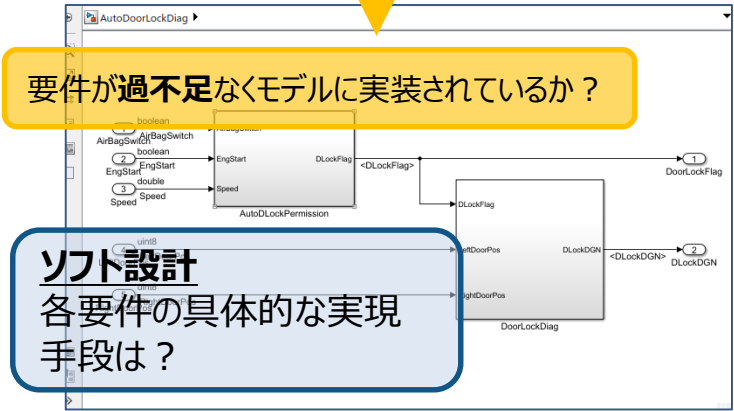
必要な機能に対して、要件が**過不足**なく定義されているか？



**構造定義**  
目標を達成するために必要となる機能は何か？

## 2. System Composerによる機能定義

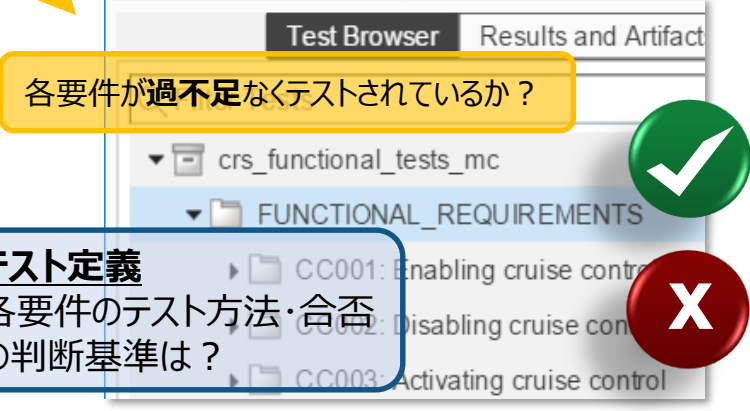
要件が**過不足**なくモデルに実装されているか？



**ソフト設計**  
各要件の具体的な実現手段は？

## 3. Simulink/Stateflowによるモデリング・シミュレーション

各要件が**過不足**なくテストされているか？



**テスト定義**  
各要件のテスト方法・合否の判断基準は？

## 4. Simulink Testによるテストの定義・管理

# アジェンダ

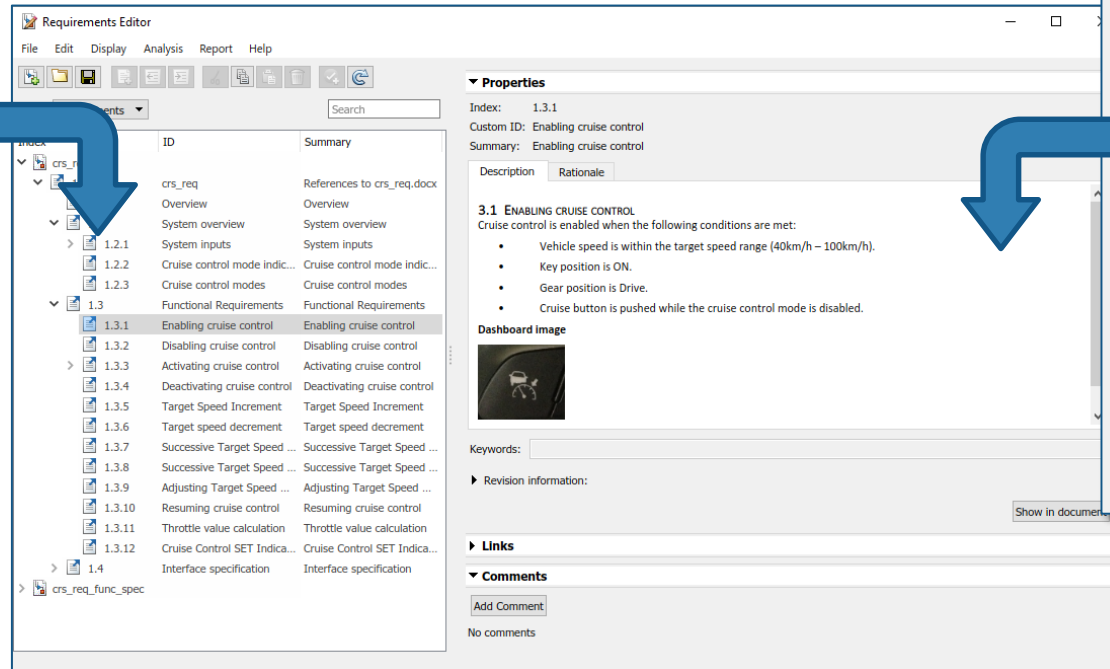
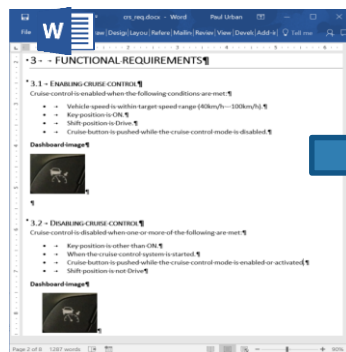
1. **要件定義/管理 (Simulink Requirements)**
2. 構造・機能定義 (System Composer)
3. モデリング・シミュレーション ( Simulink/SLDV )
4. テストの定義・管理 ( Simulink Test )

## 要件エディタによる要件の作成・外部仕様書の取り込み

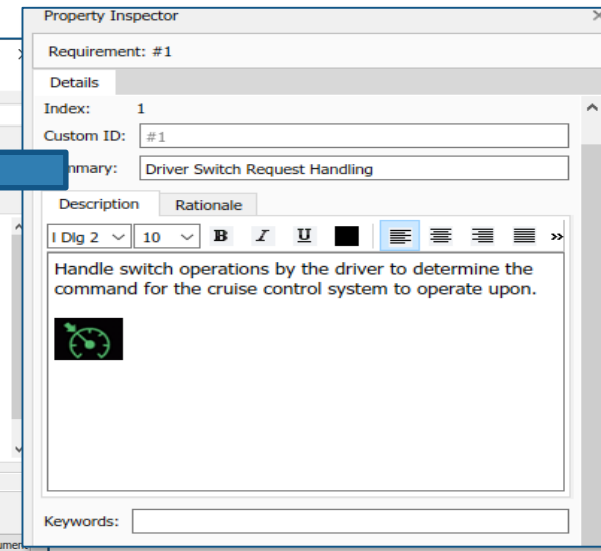
- Simulink上で要件を作成(定義)できる
- Word/Excelで作成された外部仕様書は、要件エディタに取り込むことも可能
  - 要件エディタ上の各要件を「読み取り専用」として取り扱うことで、外部仕様書をリファレンスにすることもできる
  - 外部要件を更新した場合、影響を受けそうなモデルやテストを検出可能(要トレーサビリティ設定)
- ReqIF経由で外部要件管理ツールとの連携も可能



外部仕様書の取り込み



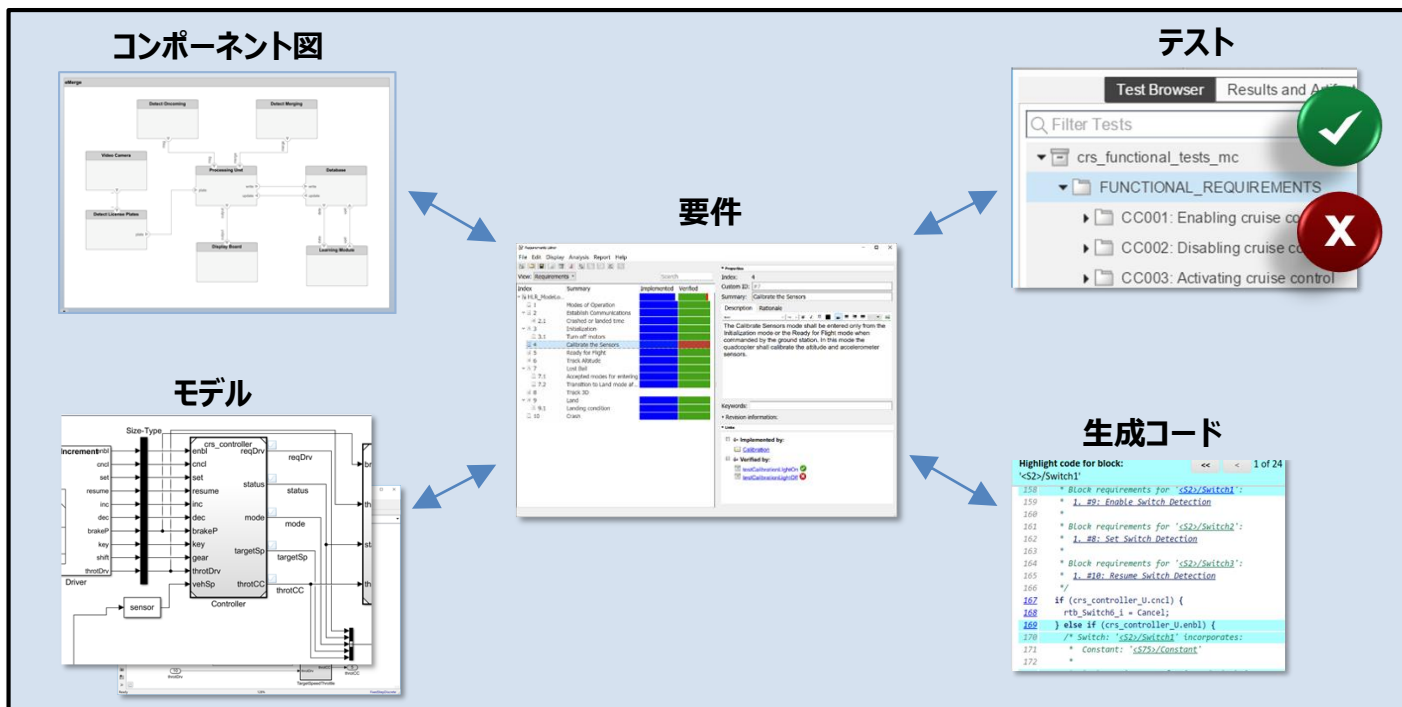
新規要件の作成



外部要件管理ツールとの連携

## 要件トレーサビリティ および 実装・検証状態の可視化

- 以下に対して、双方向のトレーサビリティを担保可能できる
  - 要件 ~ System Composerで作成したアーキ図内のコンポーネントダイアグラム
  - 要件 ~ モデル(Simulink/Stateflow)
  - 要件 ~ Test Manager内のテスト
- トレーサビリティ機能により、実装・検証状態を可視化できる



実装ステータス      検証ステータス

Summary	Implemented	Verified
Driver Switch Request Handling	Blue bar	Green bar
Switch precedence	Blue bar	Green bar
Avoid repeating commands	Blue bar	Green bar
Long Switch recognition	Blue bar	Green bar
Cancel Switch Detection	Blue bar	Red bar
Set Switch Detection	Blue bar	Green bar
Enable Switch Detection	Blue bar	Green bar
Resume Switch Detection	Blue bar	Green bar
Increment Switch Detection	Blue bar	Green bar

**実装ステータス**

- Blue square: 実装済み
- Light blue square: 正当化
- White square: リンクなし

**検証ステータス**

- Green square: 合格
- Red square: 失敗
- Orange square: テスト未実施
- Light blue square: 正当化
- White square: テストなし

## 要件変更時の影響分析

### 要件を変更した場合、影響範囲を特定可能

- 外部仕様書を「読み取り専用の参照」としてインポートしている場合は、要件エディタ上で「更新」を実行することによって変更点を検出可能
- 要件に紐づけられているオブジェクト(アーキテクチャモデル、モデル、テスト)へとトレース可能



Index	ID	
1	crs_req	crs_req.docx
1.1	1 Overview	Overview
1.2	2 System overview	System overview
1.3	3 Functional Requirements	Functional Requirements
1.3.1	3.1 Enabling cruise control	Enabling cruise control
1.3.2	3.2 Disabling cruise control	Disabling cruise control
1.3.3	3.3 Activating cruise control	Activating cruise control
1.3.4	3.4 Deactivating cruise control	Deactivating cruise control
1.3.5	3.5 Target Speed Increment	Target Speed Increment
1.3.6	3.6 Target speed decrement	Target speed decrement
1.3.7	3.7 Successive Target Speed Increment	Successive Target Speed Increment
1.3.8	3.8 Successive Target Speed Decrement	Successive Target Speed Decrement
1.3.9	3.9 Adjusting Target Speed while driving	Adjusting Target Speed while driving
1.3.10	3.10 Resuming cruise control	Resuming cruise control

▼ 要件の交換

更新    エクスポート    ロックをすべて解除

インデックス: 1.3.1

カスタム ID: 3.1 Enabling cruise control

概要: Enabling cruise control

説明 根拠

Arial 10 B I U

### 3.1 Enabling cruise control

Cruise control is enabled when the following conditions are met:

- Vehicle speed is within the target speed range (40km/h - 100km/h).
- Key position is ON.**
- Gear position is Drive.
- Cruise button is pushed while the cruise control mode is disabled.

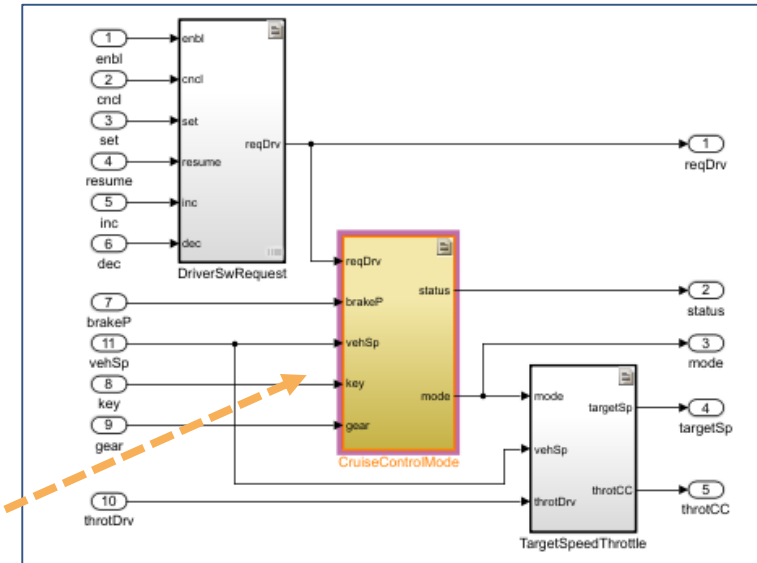
Dashboard image

情報:

10 kshoji 26-2月-2020 16:08:08

1 kshoji 26-2月-2020 16:15:46

実装元: DriverSwRequest



**3.1 Enabling cruise control** 変更

Cruise control is enabled when the following conditions are met:

- Vehicle speed is within the target speed range (40km/h - 100km/h).
- Key position is OFF.**
- Gear position is Drive.
- Cruise button is pushed while the cruise control mode is disabled.

トレース

## 要件変更の管理

- 変更検出時、リンクビューに切り替えることで、変更点を管理できる
  - コメントを追加して、「問題点をクリア」を実行することで、要件の履歴管理が可能
  - 誰がどの日時に変更したのかを履歴管理可能(変更内容は手動で入力)



Index	ID	
demo*		
1	crs_req	crs_req.docx
1.1	1 Overview	Overview
1.2	2 System overview	System overview
1.3	3 Functional Requirements	Functional Requirements
1.3.1	3.1 Enabling cruise control	Enabling cruise control
1.3.2	3.2 Disabling cruise control	Disabling cruise control
1.3.3	3.3 Activating cruise control	Activating cruise control
1.3.4	3.4 Deactivating cruise control	Deactivating cruise control
1.3.5	3.5 Target Speed Increment	Target Speed Increment
1.3.6	3.6 Target speed decrement	Target speed decrement

インデックス: 1.3.1  
カスタム ID: 3.1 Enabling cruise control  
概要: Enabling cruise control

説明 根拠

Arial 10 B I U

### 3.1 Enabling cruise control

Cruise control is enabled when the following conditions are met:

- Vehicle speed is within the target speed range (40km/h - 100km/h).
- Key position is ON.**
- Gear position is Drive.
- Cruise button is pushed while the cruise control mode is disabled.

変更

変更日: 26-2月-2020 16:15:46

▼ リンク

実装元: DriverSwRequest

リンクビューで表示



表示: リンク 検索

Label	Source	Type
crs_controller.slmx	変更されたソース: 0/7	
DriverSwRequest_Tests.slmx	変更されたソース: 0/11	
crs_controller.slmx*	変更されたソース: 0/1	
3.1 Enabling cruise control: Ena...	DriverSwRequest	実装

ソース: DriverSwRequest

タイプ: 実装

宛先: 3.1 Enabling cruise control

説明 根拠

3.1 Enabling cruise control

コメントの追加

kshoji が 26-2月-2020 16:23:33 にコメント (改訂:1)

更新された宛先:  
タイムスタンプ (UTC09:00): 26-2月-2020 16:15:46 (旧 26-2月-2020 16:08:08)

kshoji が 26-2月-2020 16:22:42 にコメント (改訂:1)

仕様間違いのため、OFF→ONに変更

▼ 変更情報

変更の問題は検出されませんでした

ソース: リビジョン: N/A (タイムスタンプ: N/A)

宛先: リビジョン: 1 (タイムスタンプ: 26-2月-2020 16:15:46)

▼ コメント

コメントの追加

kshoji が 26-2月-2020 16:22:42 にコメント

仕様間違いのため、OFF→ONに変更

▼ 変更情報

問題: 宛先が変更されました。

保存済み: リビジョン: 1 (タイムスタンプ: 26-2月-2020 16:08:08)

実際: リビジョン: 1 (タイムスタンプ: 26-2月-2020 16:15:46)

問題をクリア

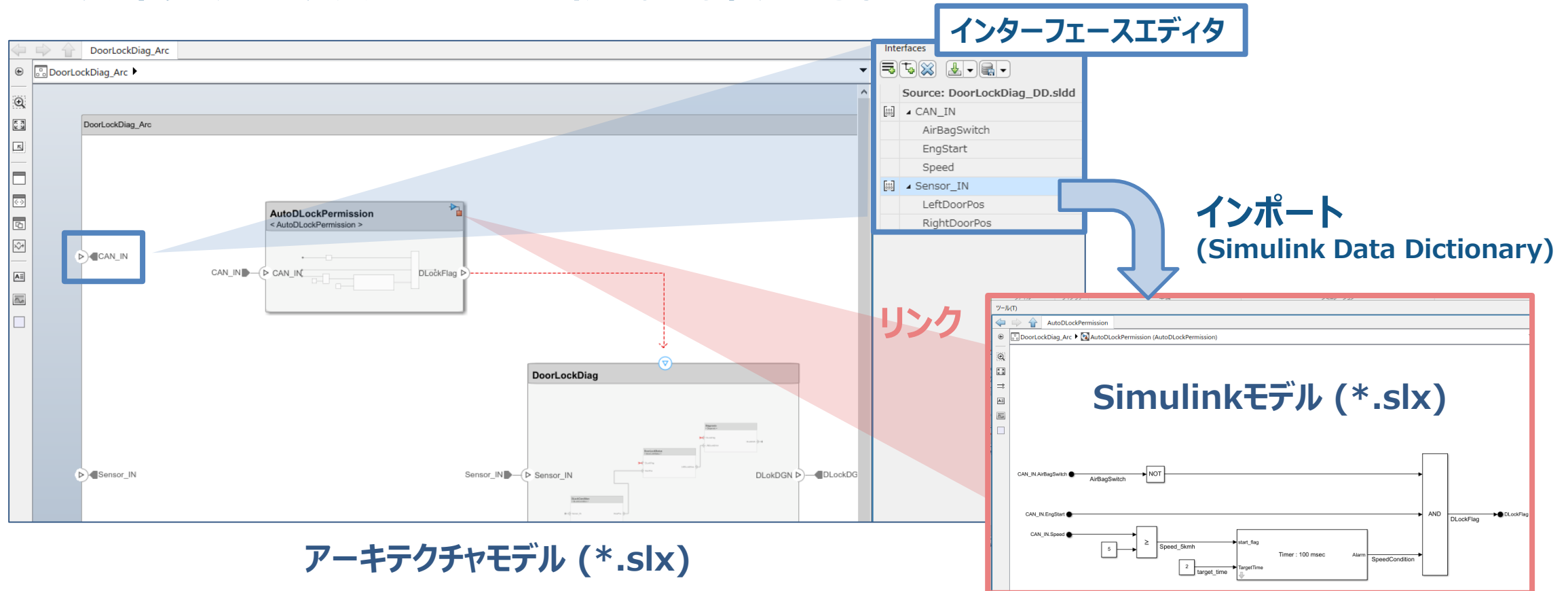
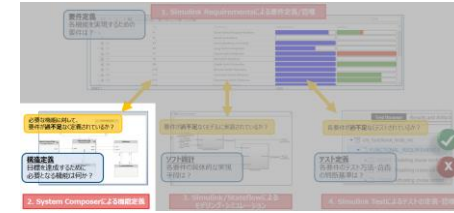


# アジェンダ

1. 要件定義/管理 (Simulink Requirements)
2. **構造・機能定義 (System Composer)**
3. モデリング・シミュレーション ( Simulink/SLDV )
4. テストの定義・管理 ( Simulink Test )

# アーキテクチャモデルの作成

- アーキテクチャモデルを作成することで、SWの機能構造を明確化できる
  - Simulinkに似た専用のエディタにより、簡単にアーキテクチャモデルが作成可能
- アーキテクチャモデル内で定義したコンポーネントとモデルファイルはリンクできるので、ソフトウェアインテグレーションの手段としても利用できる



# アーキテクチャビュー機能による依存関係分析

- 条件を満たすコンポーネントのみにフォーカスを当てたビューを生成できるので、アーキテクチャモデルを変更することなく必要な情報のみアクセス可能



Architecture Views  
Open architecture views gallery

**General**

Name: MyView Color: [Blue]

Description: Description of view

Include referenced models

Filter View

**Filter**

Select: Components with a port

For Which: Name == "ZA2"

AND  OR  NOT

Add to Filter

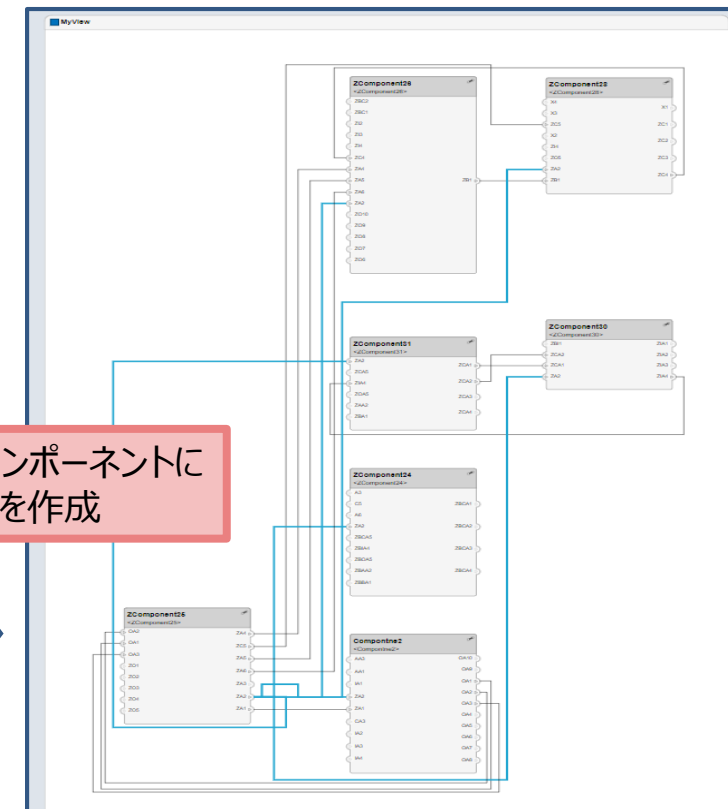
Filter: HasPort(Property("Name") == "ZA2")

**Grouping**

Group selected components into View Components by: None

Cancel Apply

“ZA2”をポートに持つコンポーネントにフォーカスを当てたビューを作成



※: ラインのハイライトは手動でおこなっており、現Verでは機能としては実装されていません



# アジェンダ

1. 要件定義/管理 (Simulink Requirements)
2. 構造・機能定義 (System Composer)
3. **モデリング・シミュレーション ( Simulink/SLDV )**
4. テストの定義・管理 ( Simulink Test )

# Simulink



## Signal Editorによるテストケース作成

- Signal Builderの下記課題を解消し、テスト入力をより設計・管理しやすく
  - 表形式によるデータ点編集、データ型指定、バス信号・Function-Call信号の定義・出力
- Signal Builder → Signal Editor 変換コマンド (R2018a)
  - signalBuilderToSignalEditor

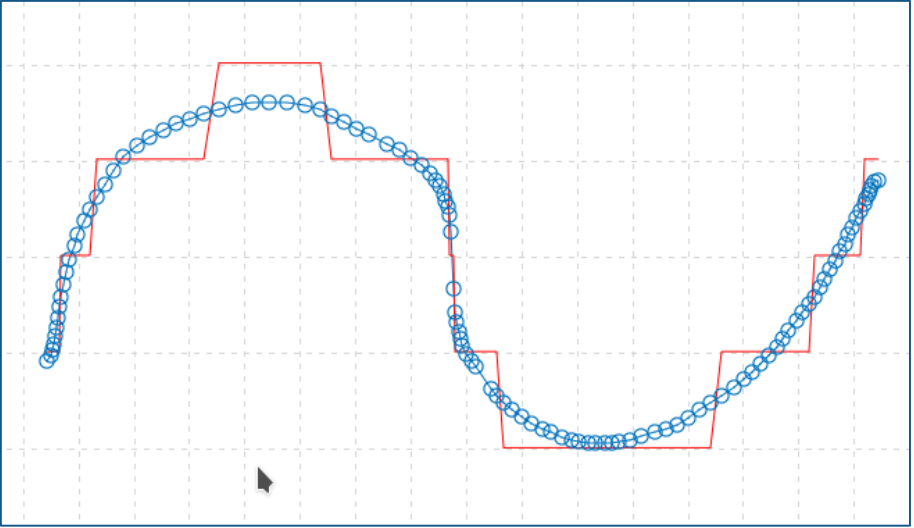
バス信号やFunction-Call信号を含め、各種のテスト信号定義可能

The screenshot shows the Signal Editor window with a plot of a sine wave and a data table below it. The data table has two columns: '時刻' (Time) and 'データ' (Data). The data points are as follows:

時刻	データ
0	0
0.01	0.012566039883352607
0.02	0.02513009544333748
0.03	0.03769018266993454
0.04	0.050244318179769556
0.05	0.06279051952931337
0.06	0.07532680552793272
0.07	0.08785119655074318
0.08	0.1003617148512149

データ型を指定可能

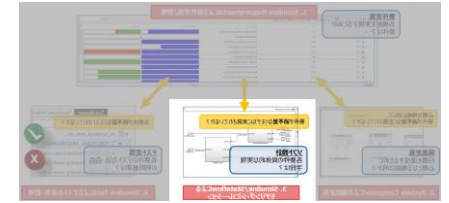
表形式でデータを編集可能



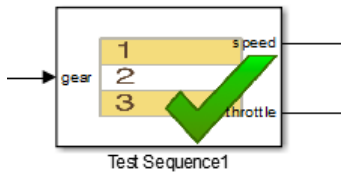
フリーハンドによるテスト信号作成 (R2019a)

# Test Sequenceによるテストケース作成

## Simulink Test



- シーケンシャルなテストパターンを作成可能
  - 状態遷移表を用いてシナリオベースでのテストを設計
  - 出力や内部状態の変化によって、テストパターンを切り替え可能
- 診断の挿入による信号チェック・不具合検証も可能

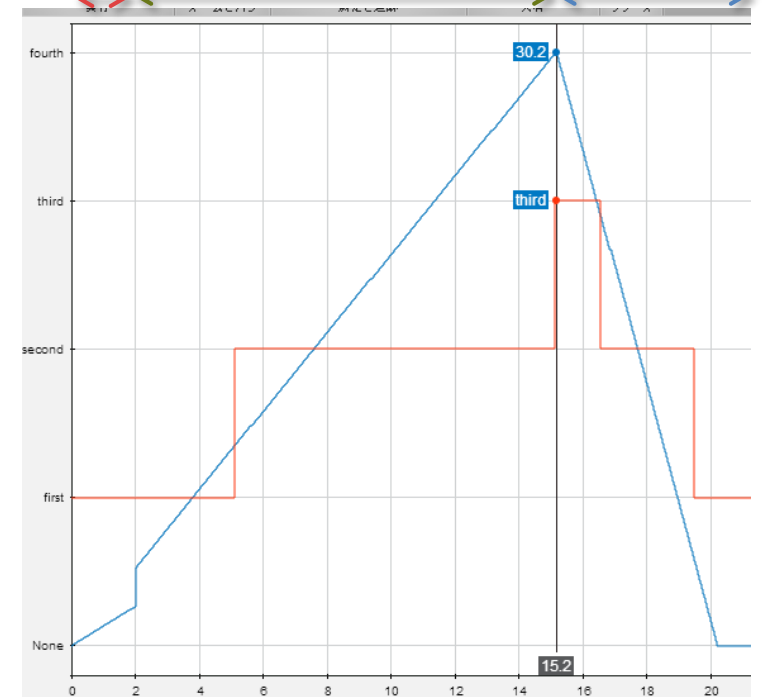


DoubleSTDriven/Test Sequence1 - Test Sequence Editor

Step	Transition	Next Step
init_step speed = ramp (t); throttle = ramp (t);	1. after (2, sec)	step_2
step_2 speed = 2* ramp (t); throttle = 2* ramp (t); peak_speed = speed; peak_throttle = throttle;	1. gear == 3	step_3
step_3 if speed > 0 speed = peak_speed - 6; throttle = peak_throttle - 6; else speed = 0; throttle = 0;		

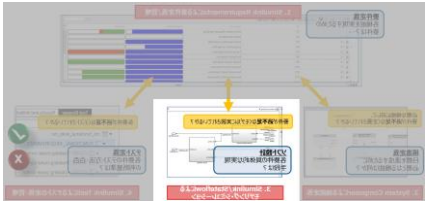
条件を満たすと次の行に遷移

2秒まで 傾き1の単調増加      2秒以降gearが3速になるまで 傾き2の単調増加      3速になれば 傾き6の単調減少



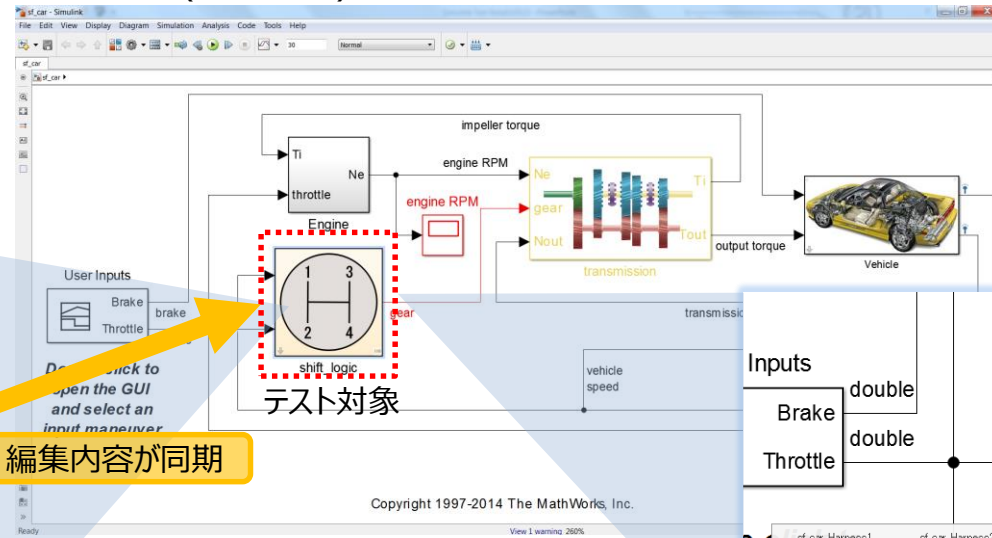
# テストハーネス機能によるテスト用モデルの作成

## Simulink Test

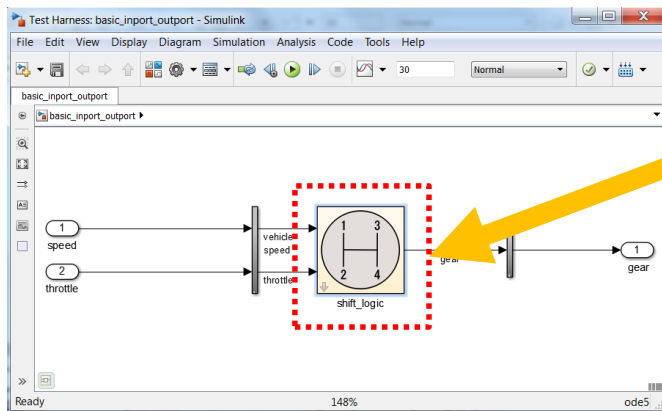


- **サブシステムの単体テストをより実現・管理しやすくなります**
  - サブシステムごと・目的ごとにテストハーネスを作成可能
  - 必要に応じてテストハーネスの保存場所をモデルの内部⇔外部に切り替える可能
  - テストハーネスとメインモデルとは常に同期

メインモデル (モデル本体)

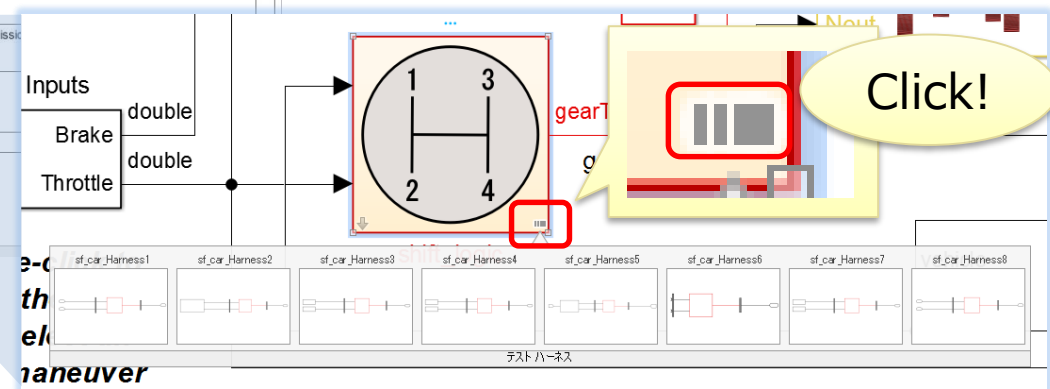


テストハーネス



編集内容が同期

テストハーネスリスト



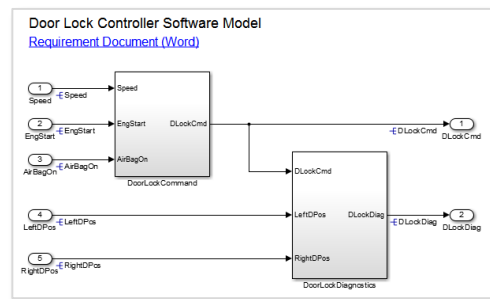
テスト対象に入力と出力を繋げた  
ハーネスモデルを自動作成

一つのサブシステムに複数のテストハーネスを  
作成することが可能



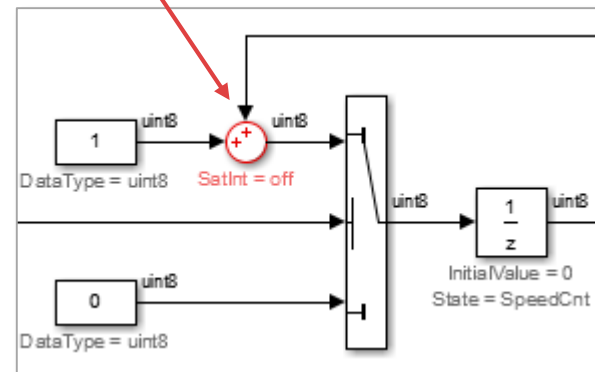
# 設計エラー検出によるランタイムエラーチェック

- I/Fの見直し/回避処理実装の有無のチェックに利用できる
  - 静的解析をおこなうため、Simulinkの診断機能と異なり、テスト入力の用意は不要
  - デッドロジック、整数オーバーフロー、ゼロ除算、配列の範囲外アクセスがモデルに含まれていないかをチェック
- エラーが検出された場合、エラー再現用のテストデータを生成可能



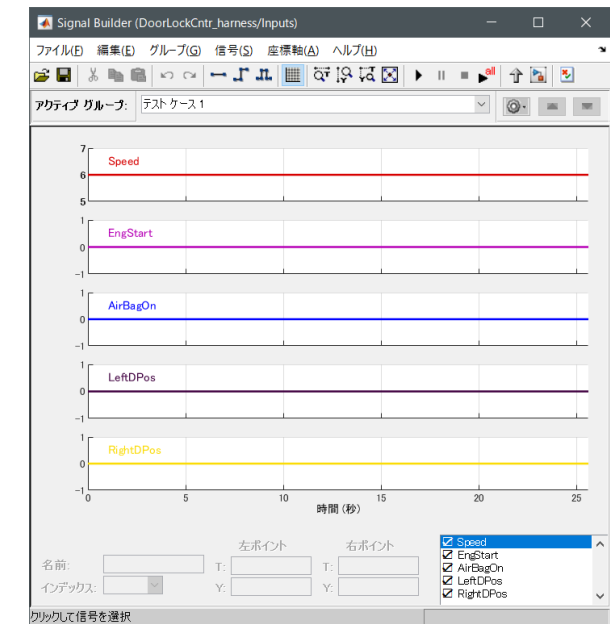
モデル

整数オーバーフローリスク有り



解析

テスト生成



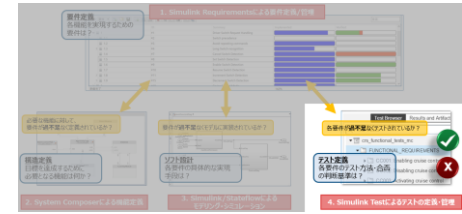


# アジェンダ

1. 要件定義/管理 (Simulink Requirements)
2. 構造・機能定義 (System Composer)
3. モデリング・シミュレーション ( Simulink/SLDV )
4. **テストの定義・管理 ( Simulink Test )**

# Test Managerによるテストの管理や自動化

## Simulink Test



- **テスト定義を資産として管理や再利用が可能**
  - 増大する検証項目の管理が可能
  - テストファイルの再利用で回帰テストを簡単に実現可能
  - テスト定義の共有によるミスコミュニケーション防止や工数削減が可能
- **テスト結果レポートを生成可能**

Test Manager

実行 ステップで実行 停止 並列 レポート 可視化

複数テストの一括管理

各テスト内容の定義

テスト対象や  
テスト入力信号、  
モニター信号  
などの指定

テストの一括実行や  
テストレポートの生成も可能



### テスト結果レポート

**FAIL2**

**Test Result Information**

Result Type: Test Case Result  
 Parent: [機能テスト: 入力・出力期待値比較](#)  
 Start Time: 2015-Aug-12 16:35:18  
 End Time: 2015-Aug-12 16:35:19  
 Outcome: **Failed**  
 Cause Of Failure: **Criteria evaluation resulted in failure.**

**Test Case Information**

Name: FAIL2  
 Type: Baseline Test

**Baseline Comparison**

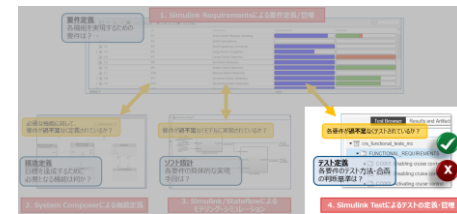
Name	Abs Tol	Rel Tol	Link to Plot
✗ DLockDiag	0	0	<a href="#">Link</a>
✓ DLockCmd	0	0	<a href="#">Link</a>
✓ Speed	0	0	<a href="#">Link</a>
✓ RightDoorPos	0	0	<a href="#">Link</a>
✓ LeftDoorPos	0	0	<a href="#">Link</a>
✓ AirBagOn	0	0	<a href="#">Link</a>
✓ EngStart	0	0	<a href="#">Link</a>

# Test Managerによる単体検証

## Simulink Test

### Subsystemごとの単体検証を自動化可能

- テストの目的に合わせて作られたSubsystem用のテストハーネスをテスト対象として設定可能
- 出力の期待値をベースラインとして指定すれば、シミュレーション結果との一致性を自動判定可能
- 許容誤差範囲を設けることが可能。(時間軸やY軸、絶対許容誤差や相対許容誤差を両方設定可能)



### テスト結果

### Test Manager

▼ テスト対象システム\*

モデル: DoorLockCtr

▼ テスト ハーネス\*

ハーネス: DoorLockCtr\_Harness1

▼ ベースライン基準\*

ベースラインデータをテスト結果に含める

信号名	シート	絶対許容誤差	相対許容誤差	リード許容誤差	ラグ許容誤差
<input checked="" type="checkbox"/> NORMAL1	NORMAL1	0	0.00%	0	0
<input checked="" type="checkbox"/> Speed		0	0.00%	0	0
<input checked="" type="checkbox"/> EngStart		0	0.00%	0	0
<input checked="" type="checkbox"/> AirBagOn		0	0.00%	0	0
<input checked="" type="checkbox"/> DLockCmd		0	0.00%	0	0

許容誤差を含めたベースラインを設定

Subsystem単体テスト用のテストハーネスを指定

テスト実行

テスト結果の自動判定

名前またはタグで結果をフィルター (たとえ)

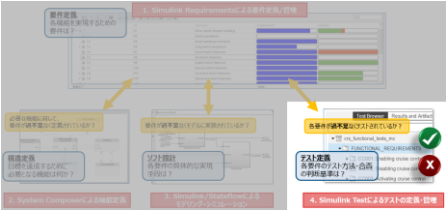
結果: 2020-Feb-21 14:23:37

▼ DoorLockCommand

- 反復 1
  - ベースライン基準の結果
    - Speed:  (4)
    - EngStart:  (0)
    - AirBagOn:  (0)
    - DLockCmd:  (0)
  - Sim 出力 (DoorLockCtrHarness)
    - 反復 2:  (0)
    - 反復 3:  (0)
    - 反復 4:  (0)

許容誤差 (緑) 差分 (赤)

# Simulink Test



## Test ManagerによるExcelテストデータの読み込み

- Excelでのテストデータを管理し、直接、Simulinkモデルとマッピング可能
  - 入出力の信号データタイプや補完方法をExcel上で定義可能
  - シートごとでのインポート可能
  - 入出力データをまとめて定義可能
- モデルからExcelフォーマットのテスト入力カテンプレートも生成可能

入出力データを纏めて定義

A	B	C	D	E	F	G	H
1	Time	Speed	EngStart	AirBagOn	LeftDPos	RightDPos	DLockCmd
2		Type: single	Type: boolean	Type: boolean	Type: single	Type: single	Type: boolean
3		Interp: zoh	Interp: zoh	Interp: zoh	Interp: zoh	Interp: zoh	Enum: DiagStatus
4							Interp: zoh
5	0.1	0	0	0	0	0	NORMAL
6	0.2	0	0	0	0	0	NORMAL
7	0.3	0	1	0	0	0	NORMAL
8	0.4	0	1	0	0	0	NORMAL
9	0.5	5	1	0	0	0	NORMAL
10	0.6	5	1	0	0	0	NORMAL
11	0.7	5	1	0	0	0	NORMAL
12	0.8	5	1	0	0	0	NORMAL

データタイプ・補間方法のインポート対応

シートごとでのインポート可能

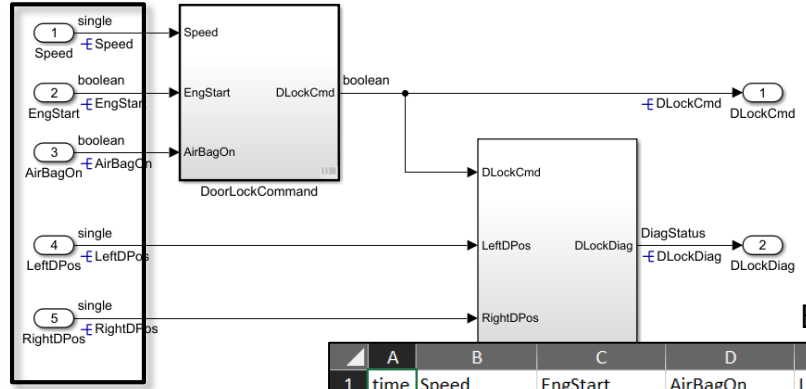
繰り返しシミュレーション実施時のインポートしたシートごとで定義した入力やベースラインを設定可能

▼ 反復\*

▼ テーブルによる反復\*

名前	説明	外部入力	ベースライン
反復 1	なし	NORMAL1	NORMAL1
反復 2	なし	NORMAL2	NORMAL2
反復 3	なし	FAIL1	FAIL1
反復 4	なし	FAIL2	FAIL2

モデル



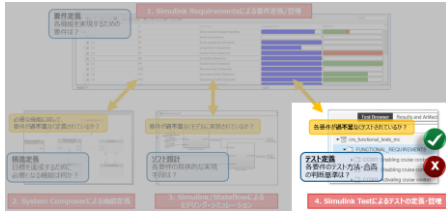
Excelテンプレート

A	B	C	D	E	F
1	time	Speed	EngStart	AirBagOn	LeftDPos
2		Type: single	Type: boolean	Type: boolean	Type: single
3		Source: Input			
4		Interp: zoh	Interp: zoh		
5	0	0	0	0	0
6	10	0	0	0	0
7					

モデルの入力信号の設定からExcelベースのテストテンプレートを生成可能

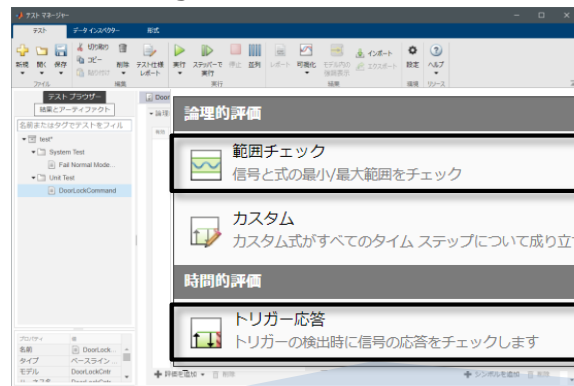
# Test Manager上で要件に基づく評価項目を定義

## Simulink Test

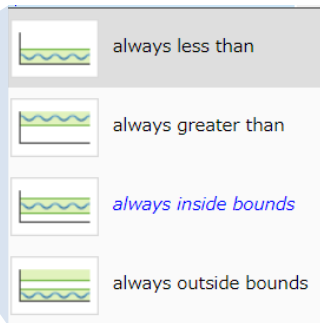


- 要件を自然言語風テンプレートで記述し、シミュレーションとともに成否を評価可能
  - 信号値の上下限・範囲内チェック
  - ～秒経過したら～する
  - 特定条件が～回発生後～になる、など。

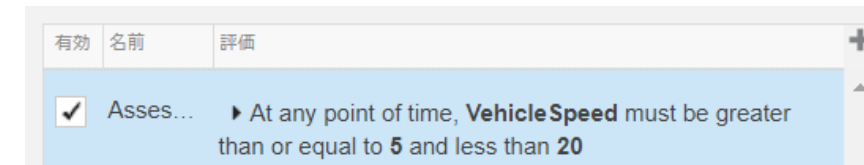
### Test Manager



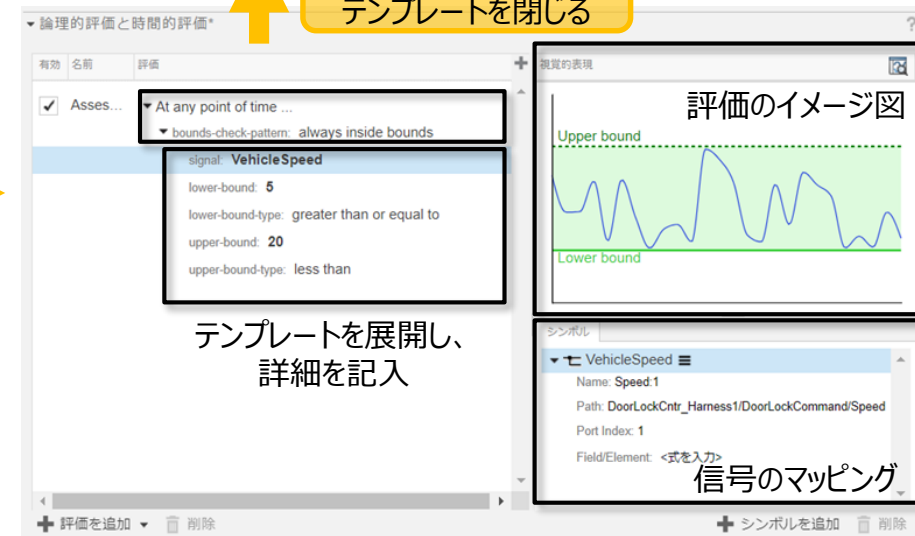
既定テンプレートから評価パターンの選択や  
カスタム評価式の定義を両方可能



評価項目が自然言語で表現される

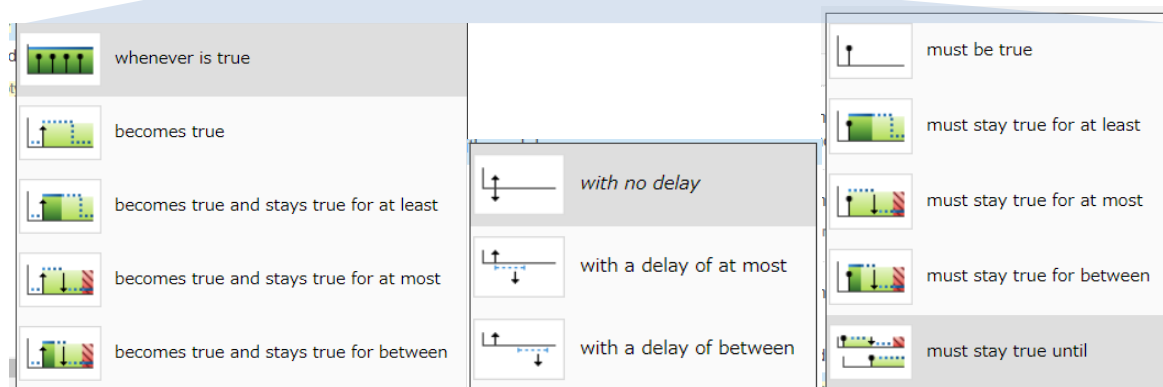


詳細設定後、  
テンプレートを閉じる



評価項目  
の詳細定義

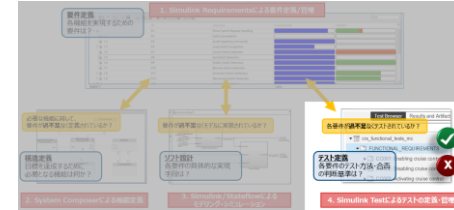
テンプレートを展開し、  
詳細を記入



# Test Manager上でのカバレッジ補完

## Simulink Test Simulink Design Verifier

- **テストの実行と共にカバレッジを計測可能**
  - 一括実行した複数のテスト(\*)で得られたカバレッジの自動累積。
  - 複数回で実行したテストで得られたカバレッジを簡単に加算可能。
  - カバレッジの確認や詳細レポートの生成はTest Managerのテスト結果画面で行うことが可能。
- **カバレッジが100%でない場合、テストの追加生成もTest Manager上から実行可能**



テストマネージャー

テスト ブラウザー 結果とアーティファクト

名前またはタグで結果をフィルタ

名前	ステータス
結果: 2020-Feb-21 16:40:01	4
DoorLockCommand	4
反復 1	✓
反復 2	✓
反復 3	✓
反復 4	✓

確認したい結果 (単体・累積)の選択

名前	ステータス	開始時間
DoorLockComma...	4	02/21/2020 16:40:02

4. 制御指令算出機能 [REQ4]: 制御指令算出機能 [REQ4].(ドアロック...

解析されたモデル	レポート	実数値...	判定	条件	MCDC	実行
DoorLockCtr/DoorLockCommand		2	100%	100%	67%	100%

計測された(単体・累積)カバレッジの表示

未達カバレッジのテストを追加

名前またはタグでテストをフィル

- test\*
  - System Test
    - Fail Normal Mode...
  - Unit Test
    - DoorLockCommand

プロパティ	値
名前	DoorLock...
タイプ	ベースライン...
モデル	DoorLockCtr

▼入力\*

入力データをテスト結果に含める

最後の時間ポイントでシミュレーションを停止

外部入力

名前
<input type="checkbox"/> NORMAL1
<input type="checkbox"/> NORMAL2
<input type="checkbox"/> FAIL1
<input type="checkbox"/> FAIL2
▼ Design Verifier (DoorLockCommand_slidvdata.mat)
<input checked="" type="checkbox"/> テストケース:1

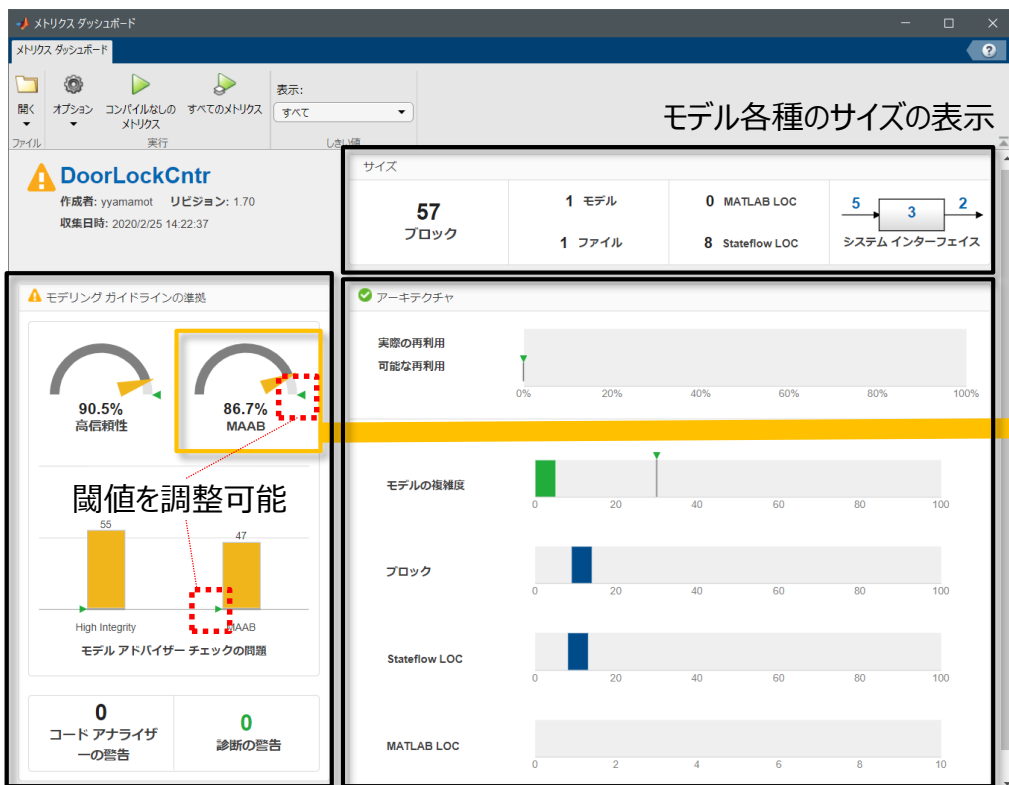
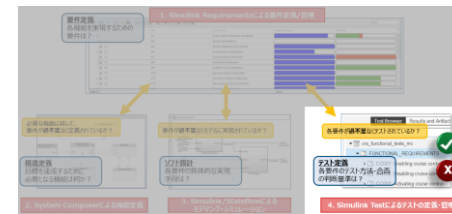
SLDVを用いてテストを追加生成し、Test Managerへ自動登録する



# メトリクスダッシュボードによるモデルの定量化測定

- **モデルのメトリクスを測定可能**
  - モデルのサイズ、複雑度、可読性を定量化可能
  - モデルガイドラインの準拠結果をまとめる
  - ダッシュボードで各測定結果をグラフ表示して可視化可能

## Simulink Check



モデルガイドラインの準拠結果まとめ

モデル各種のサイズの図示化



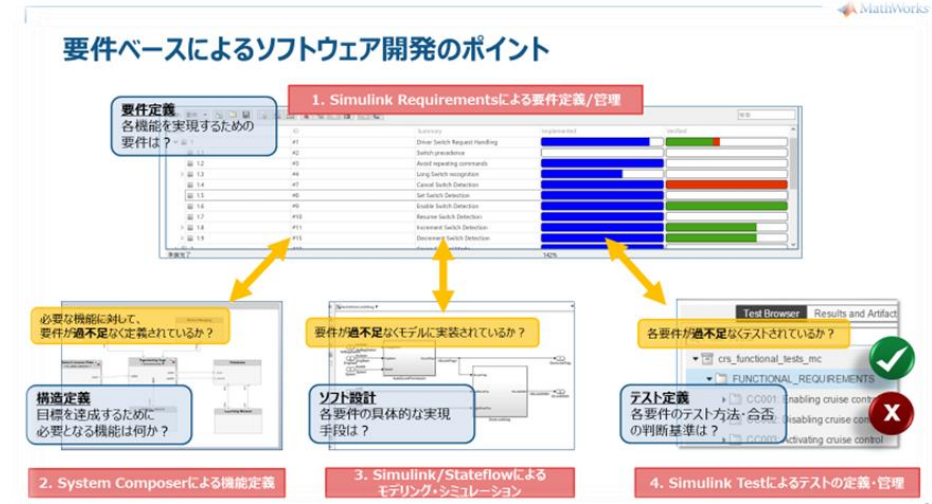
# まとめ

## ■ 本Webセミナーの説明内容の振り返り

- ✓ MBDによる要件をベースとした開発を進める上で有用と思われるMathWorksツールの機能および使い方をご紹介

## ■ MATLABプロダクトの強み

- ✓ SimulinkをはじめとしたMathWorksツールは、要件をベースとした開発が効率的に行える環境を提供。
- ✓ QCDの向上を目的とした製品開発へのMBD取り組みを促進。



規格認証を前提とした製品開発におけるMBD適用に際し、MathWorksツールを是非ご活用下さい。



## 関連情報

- **製品紹介ページ**
  - **Simulink Requirements (要件の作成と管理およびモデルとのリンク)**  
<https://jp.mathworks.com/products/simulink-requirements.html>
  - **System Composer (システムおよびソフトウェアアーキテクチャの設計と解析)**  
<https://jp.mathworks.com/products/system-composer.html>
  - **Stateflow (ロジックベースの制御)**  
<https://jp.mathworks.com/products/stateflow.html>
  - **Simulink Check (モデリング標準への準拠性を検証)**  
<https://jp.mathworks.com/products/simulink-check.html>
  - **Simulink Coverage (モデルおよび生成コードのテストカバレッジを測定)**  
<https://jp.mathworks.com/products/simulink-coverage.html>
  - **Simulink Design Verifier (設計エラーの特定、テストケースの生成)**  
<https://jp.mathworks.com/products/sldesignverifier.html>
  - **Simulink Test (統合・単体テスト管理とテストの自動化)**  
<https://jp.mathworks.com/products/sldesignverifier.html>
  - **その他のプロダクト**  
<https://jp.mathworks.com/products.html>

## 関連情報

- **オンデマンドWebセミナー**

- **今からはじめるSimulink入門**

- <https://jp.mathworks.com/videos/introduction-to-simulink-120092.html>

- **Stateflowで状態遷移設計をより便利に**

- <https://jp.mathworks.com/videos/introduction-of-stateflow-106650.html>

- **制御モデルを活用した量産／組込みソフト開発ソリューション**

- <https://jp.mathworks.com/videos/embedded-production-software-development-by-model-based-design-107673.html>

- **Simulink Testでモデルのテスト実行・管理をより便利に**

- <https://jp.mathworks.com/videos/model-test-and-test-management-made-easy-with-simulink-test-101294.html>

- **30日無料の評価版を試す**

- **製品およびサービスのお見積もり**



Accelerating the pace of engineering and science

© 2020 The MathWorks, Inc. MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See [www.mathworks.com/trademarks](http://www.mathworks.com/trademarks) for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.