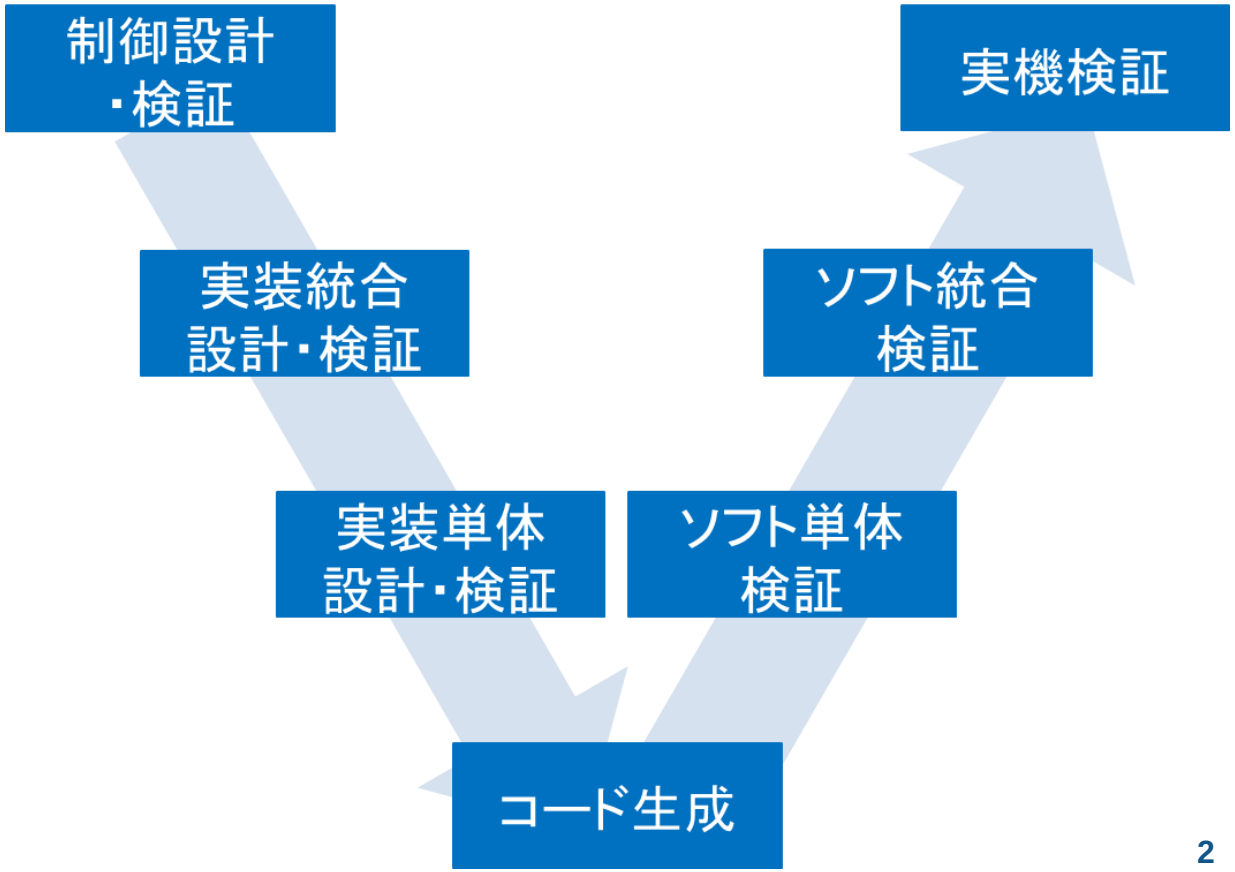
次世代モデルベース検証ソリューションで テスト・デバッグ改善

MathWorks Japan
アプリケーションエンジニアリング部（制御）
リャン　ティファニー

アジェンダ

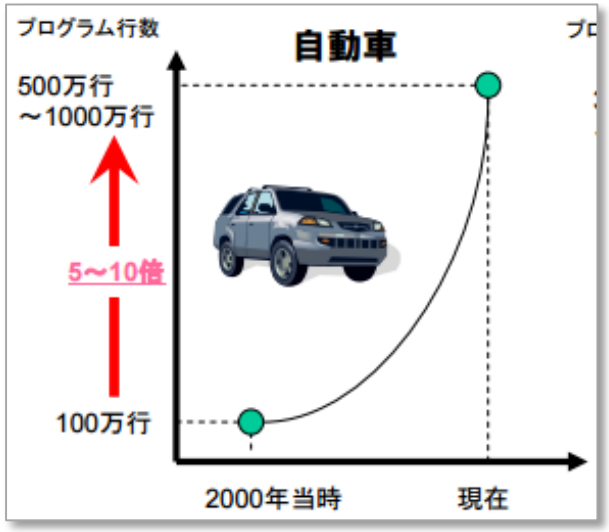
- はじめに
- 検証作業におけるチャレンジ ＆ 新しいソリューション
- まとめ



モデルベースデザイン/開発（MBD）が量産制御ソフト開発に求められる背景

課題

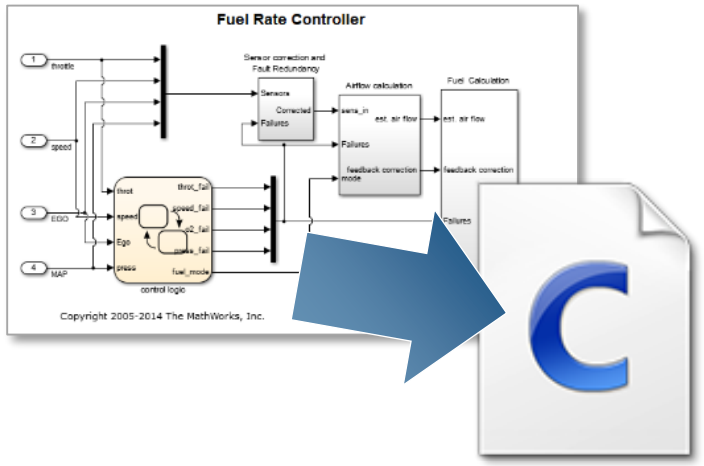
- ソフト規模の巨大化
- 検証項目の増加
- 開発期間の維持・短縮



※ 2011 経済産業省資料より引用

解決策

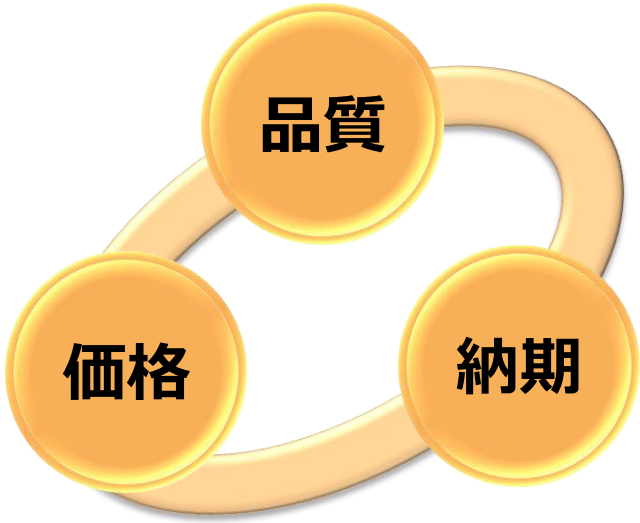
- モデル＆シミュレーションを通じた設計・**検証の前倒し**
- コード自動生成ツールや検証**ツールを用いた省力化**



MATLAB&SIMULINK

効果

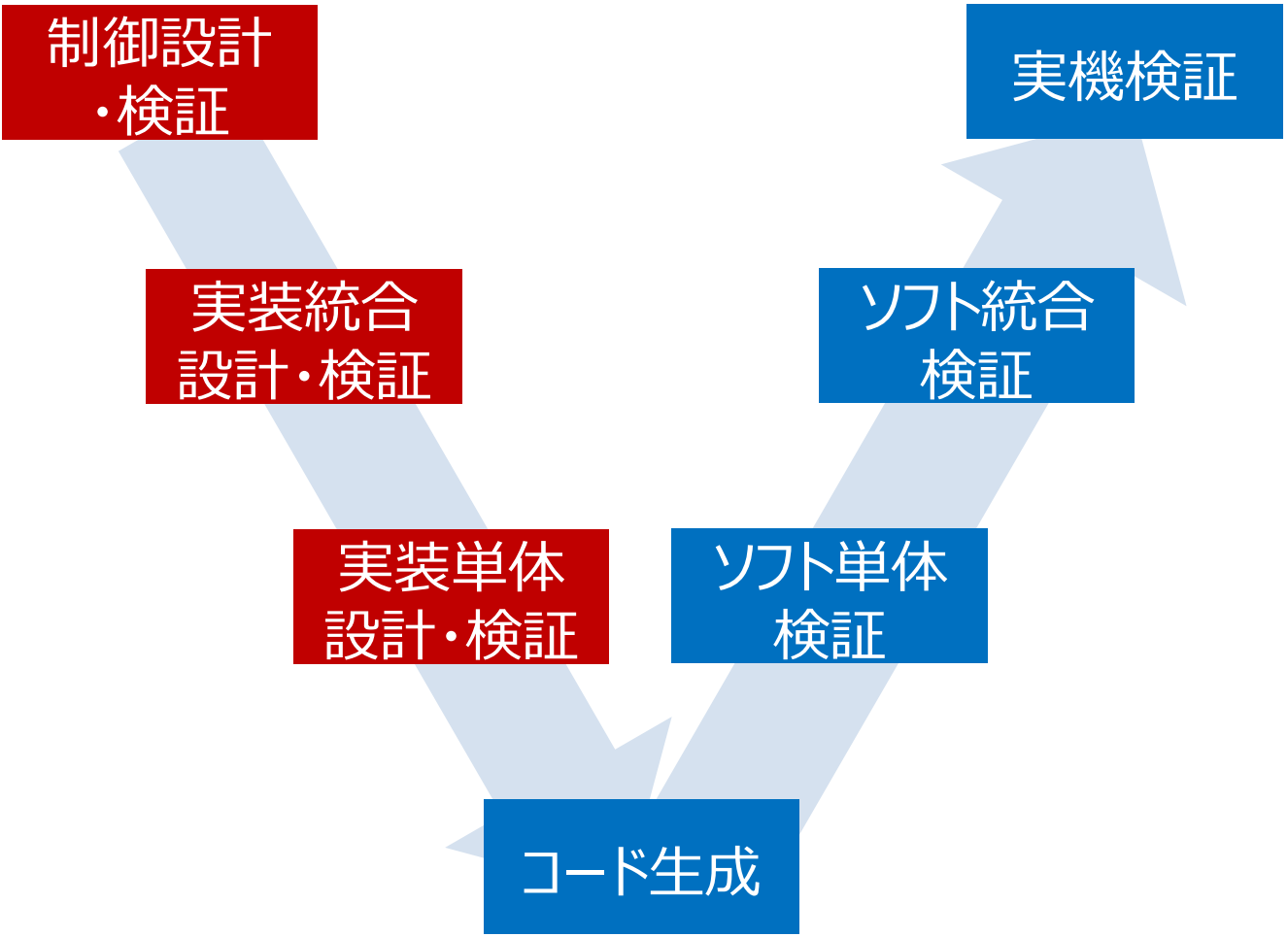
- 早期の制御仕様確定
- 開発効率向上
- ソフト信頼性向上



MBD開発時に実行できる検証作業

仕様通り誤り無く動作するモデルを作成する

(※ 各検証作業の詳細説明については講演資料の付録をご参照ください。)



Simulink Verification & Validation™

要求リンクによる

トレーサビリティ確保

Simulink Design Verifier™

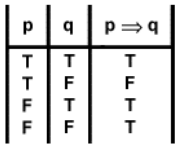
実行時エラー検出による

不具合混入防止

Simulink Verification & Validation

カバレッジ測定による

ヌケモレ発見



Simulink Design Verifier

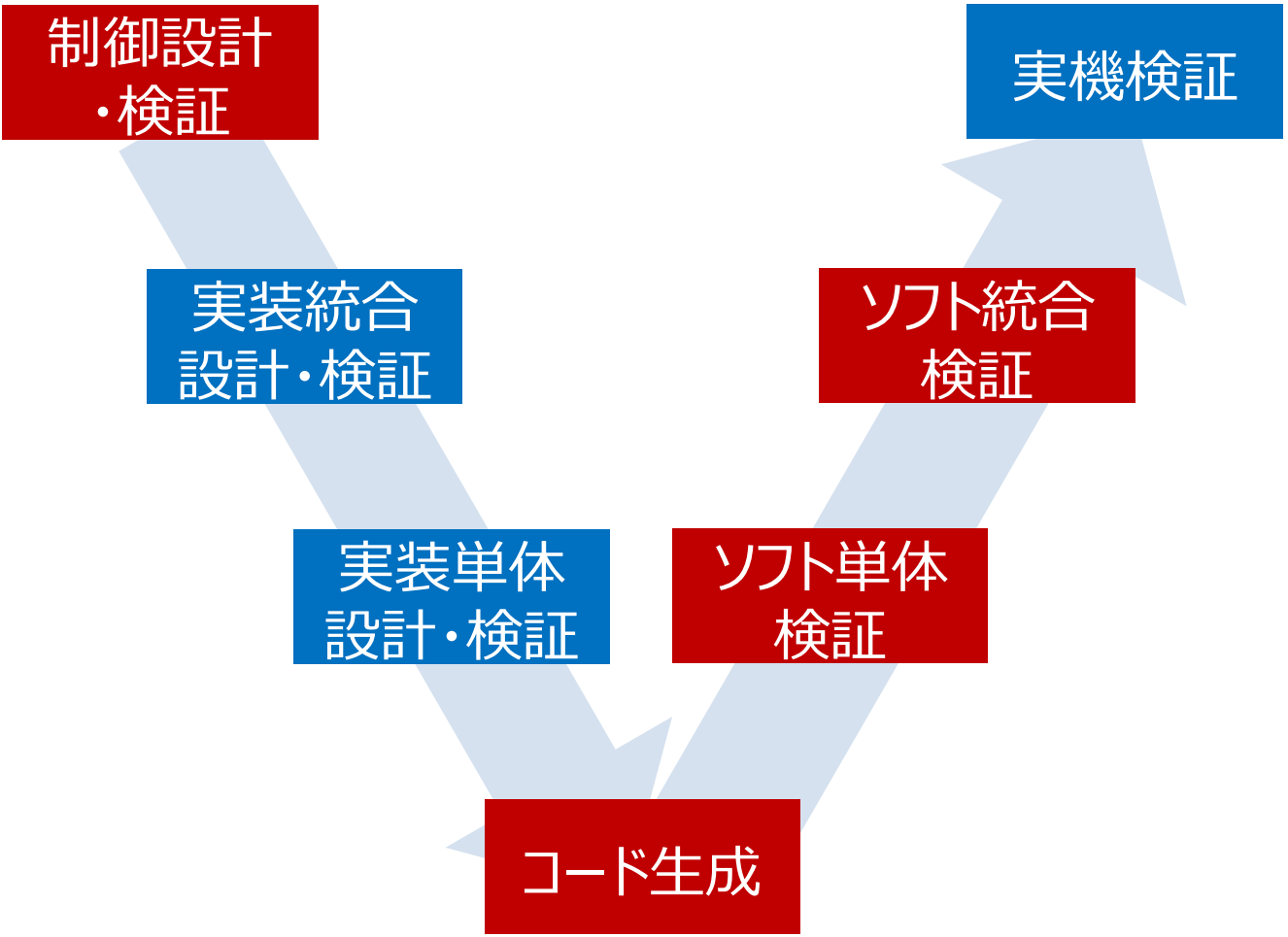
フルカバレッジテスト生成による

テスト品質向上

MBD開発時に実行できる検証作業

仕様通り誤り無く動作するソフトを作成する

(※ 各検証作業の詳細説明については講演資料の付録をご参照ください。)



Embedded Coder® / Simulink Report Generator™ / Simulink Verification & Validation

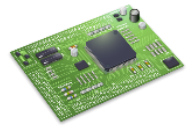
コード生成レポートによる

トレーサビリティ確保

Embedded Coder

モデル・コード等価性検証による

動作保証・性能評価

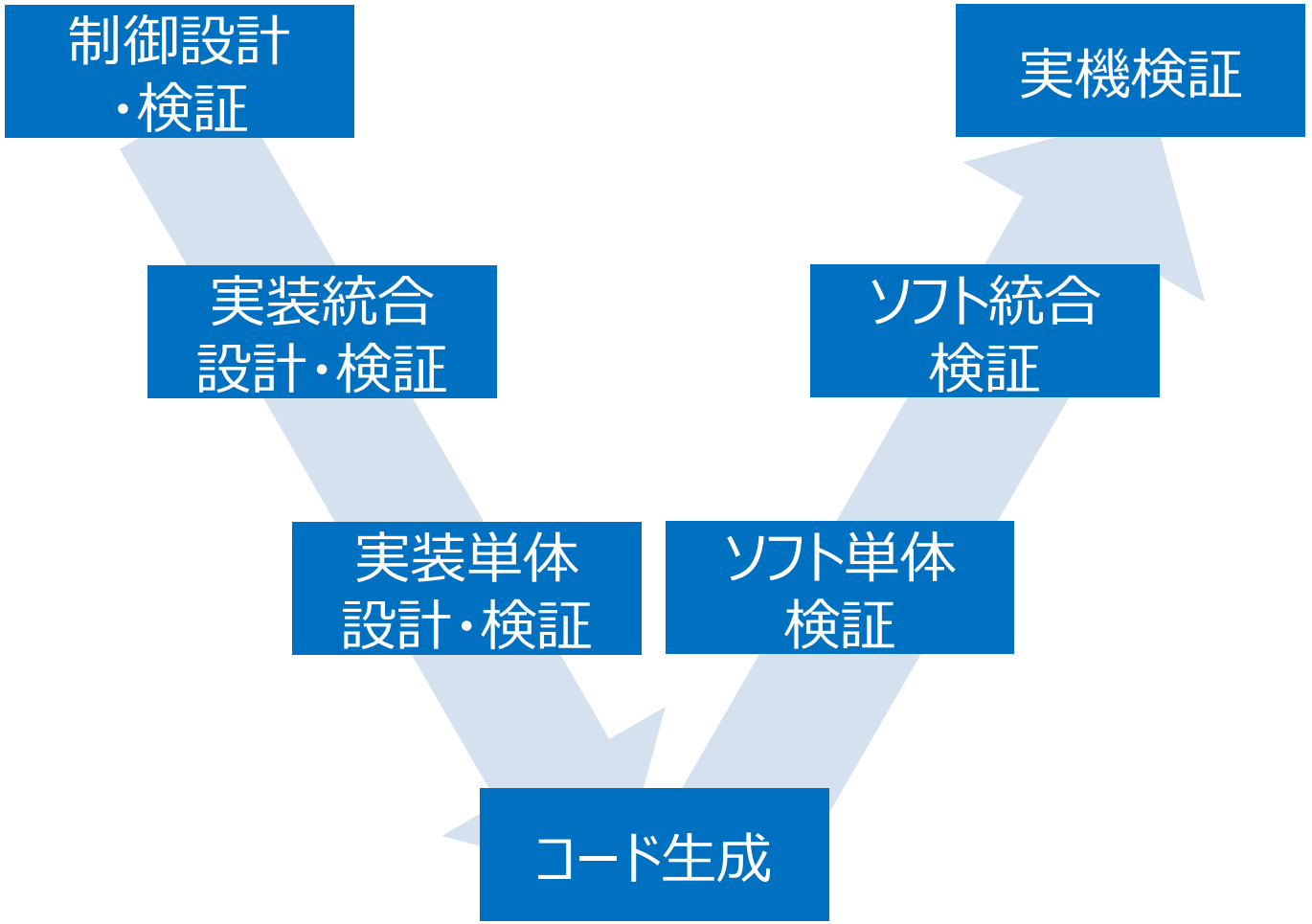


Polyspace Code Prover™

静的解析による

ソフト全体の信頼性確保

新しい検証ソリューション

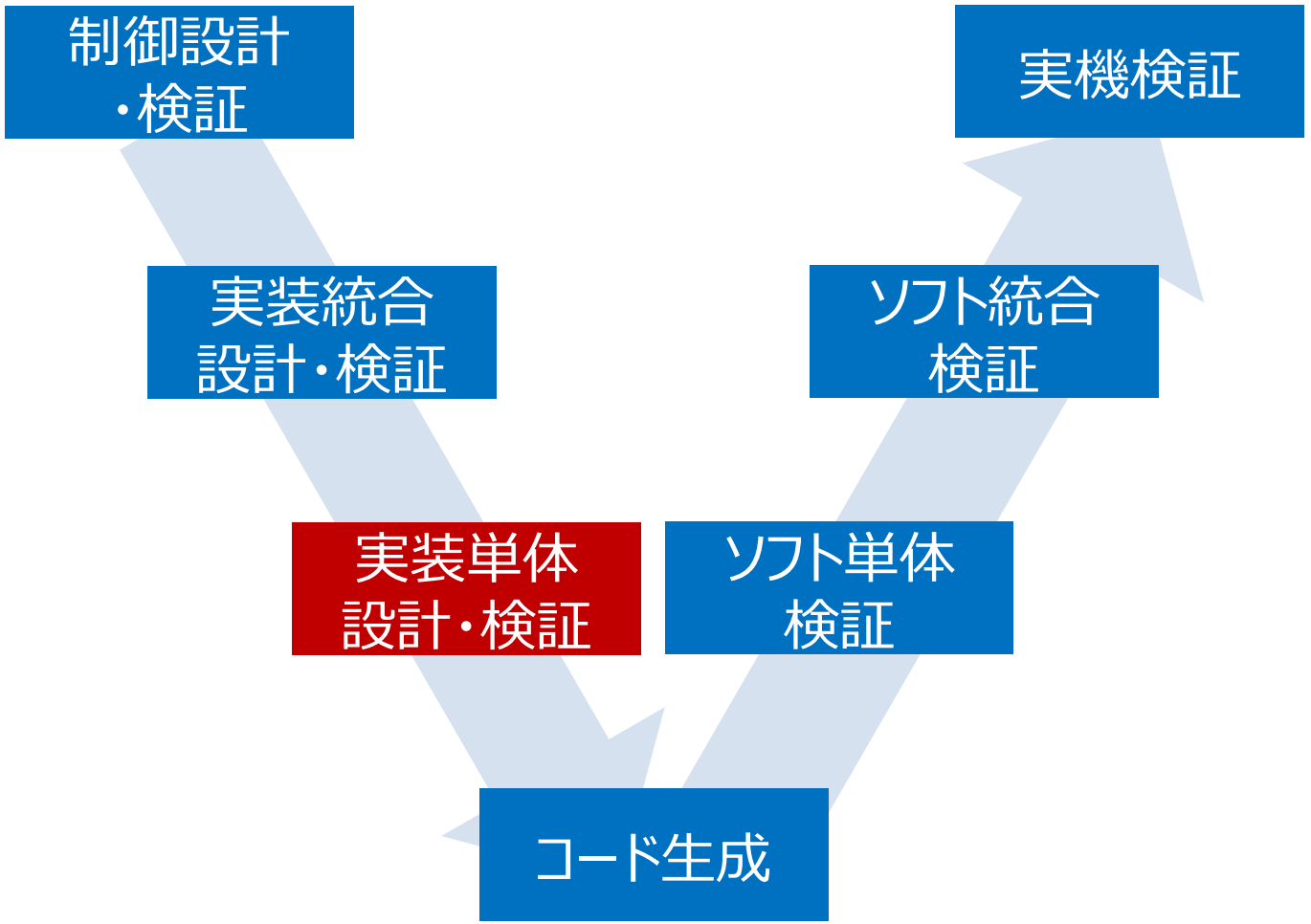


GUI/HMI ブロックを提供する
Dashboard ライブラリ

テスト自動化や一元管理ツール
Simulink Test™

信号依存関係の分析機能
モデルスライサー

新しい検証ソリューション



GUI/HMI ブロックを提供する
Dashboard ライブラリ

テスト自動化や一元管理ツール
Simulink Test™

信号依存関係の分析機能
モデルスライサー

R2015a Simulink® 新機能

モデルテストの操作における課題＆ソリューション

課題

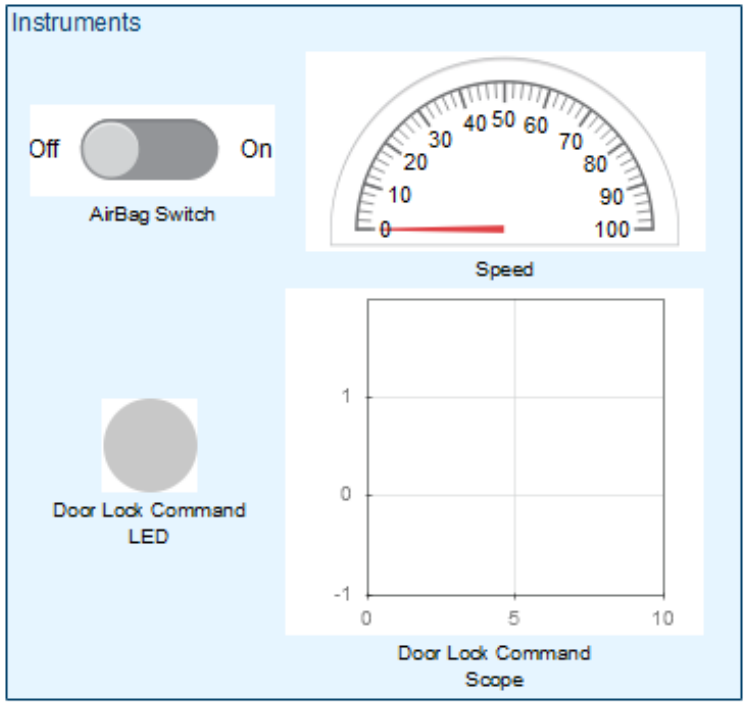
Test Engineer

パラメータ調節や信号確認用のGUIを作りたい。

現場のエンジニアにとってGUIの作成が大変。

解決策

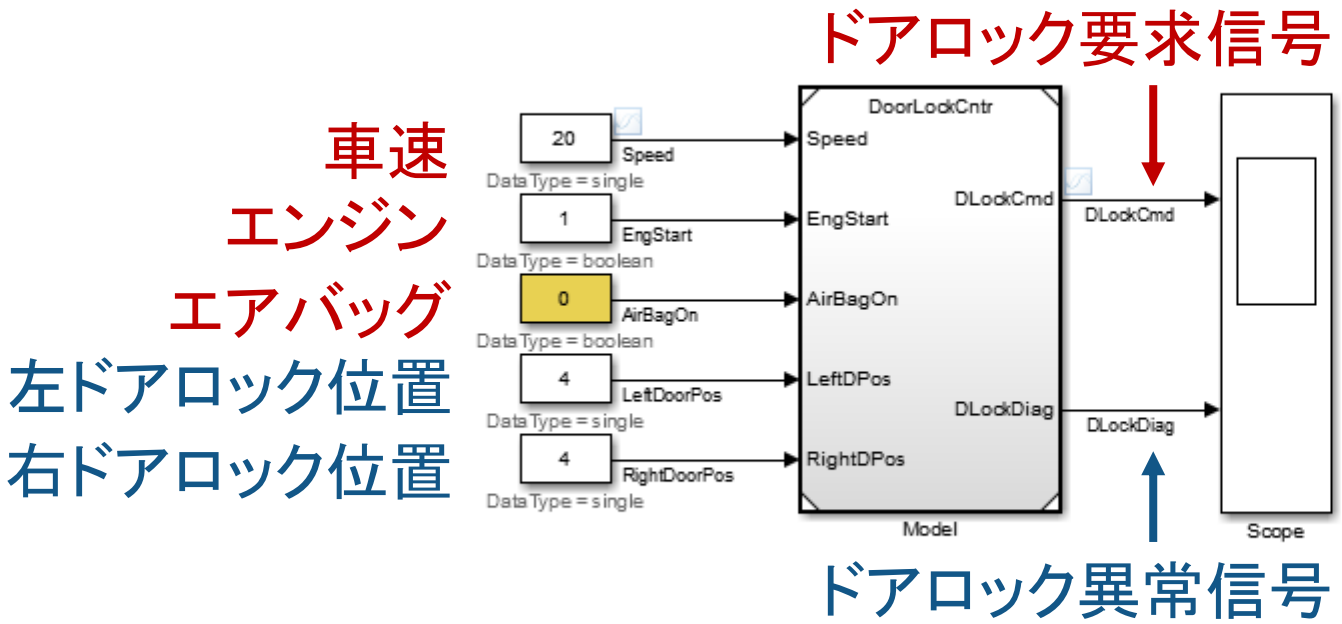
Dashboard ライブラリ の GUI/HMI ブロックを使用し、直感的な表示・パラメータ調節を実現する。

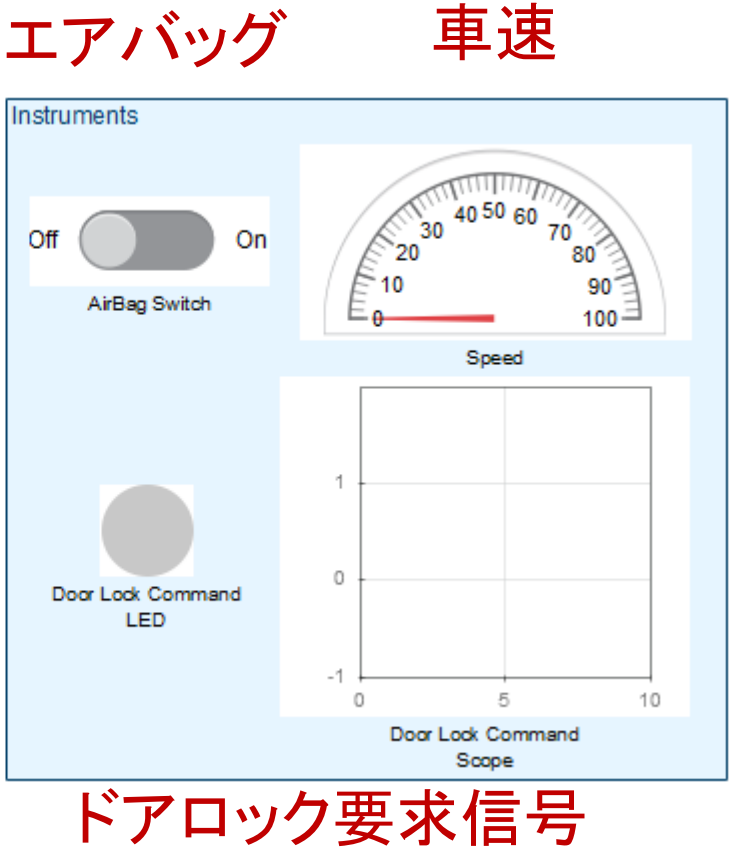


Dashboard ブロックライブラリ利用例
Door Lock 制御ロジックの動作確認モデル

R2015a

- GUI・HMI コンポーネントを用いてパラメータ調節・信号確認を
  より簡単・直感的に

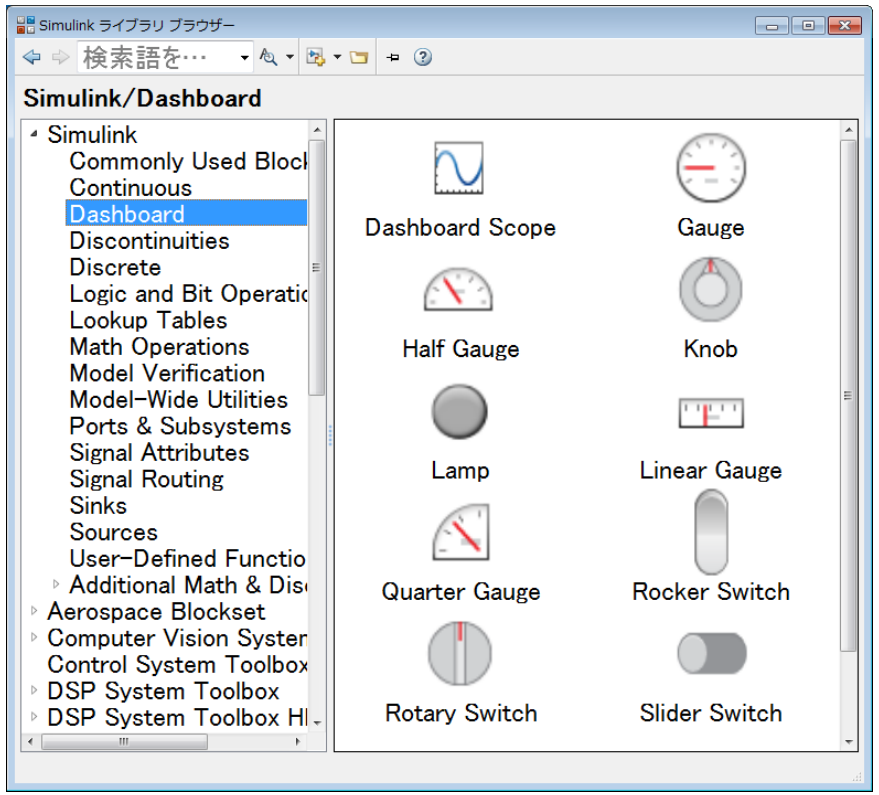




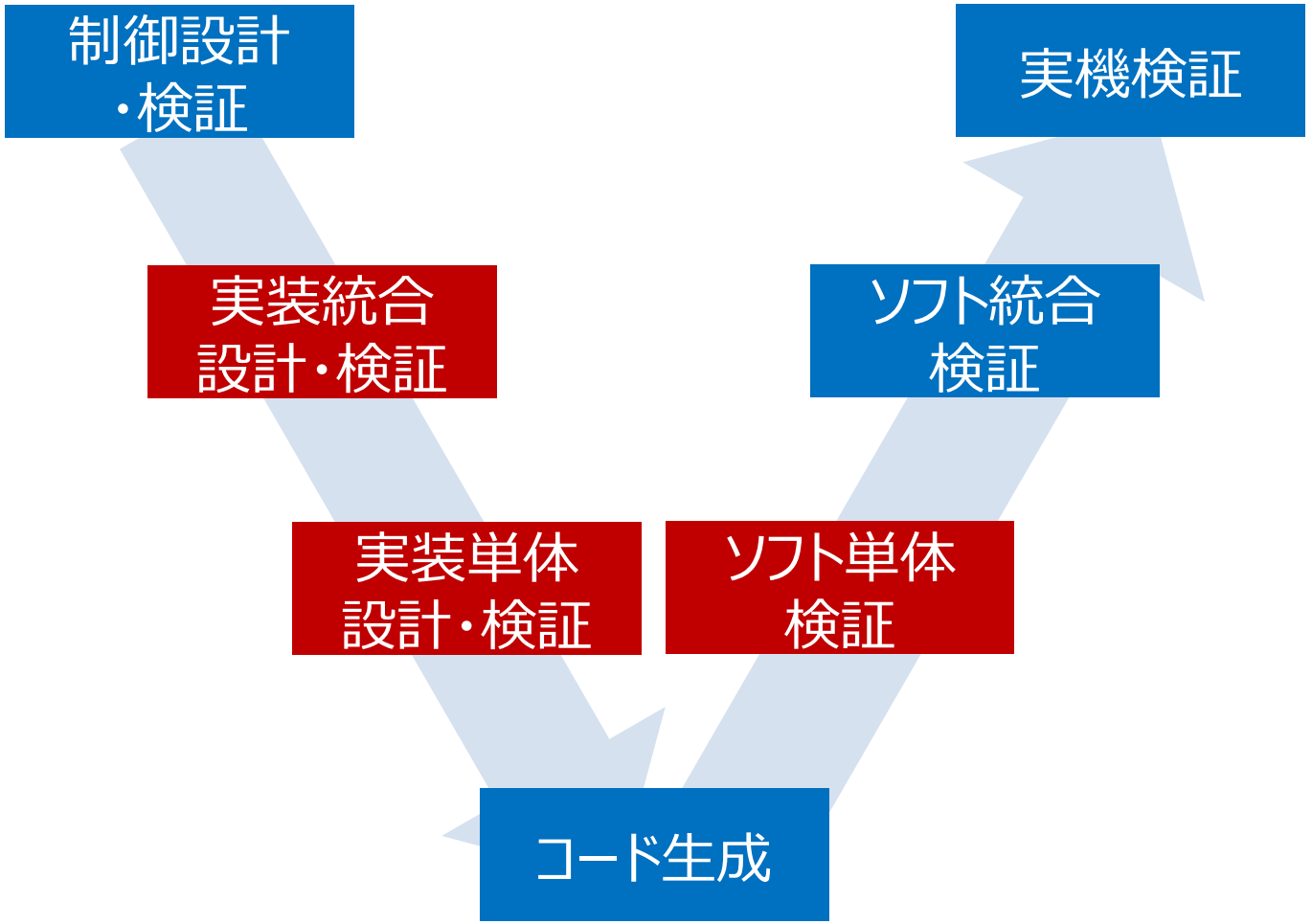
Dashboard ブロックライブラリ

モデル内GUI/HMIで直感的な表示・パラメータ調節が実現可能

- 信号表示・パラメータ調節用UIコンポーネントを提供
  - モデル内Scope
  - ランプ
  - ノブ
  - ゲージ
  - スイッチ



新しい検証ソリューション



GUI/HMI ブロックを提供する
Dashboard ライブラリ

テスト自動化や一元管理ツール
Simulink Test

信号依存関係の分析機能
モデルスライサー

R2015a新製品

テストモデルの作成・管理における課題＆ソリューション

課題

Design Engineer

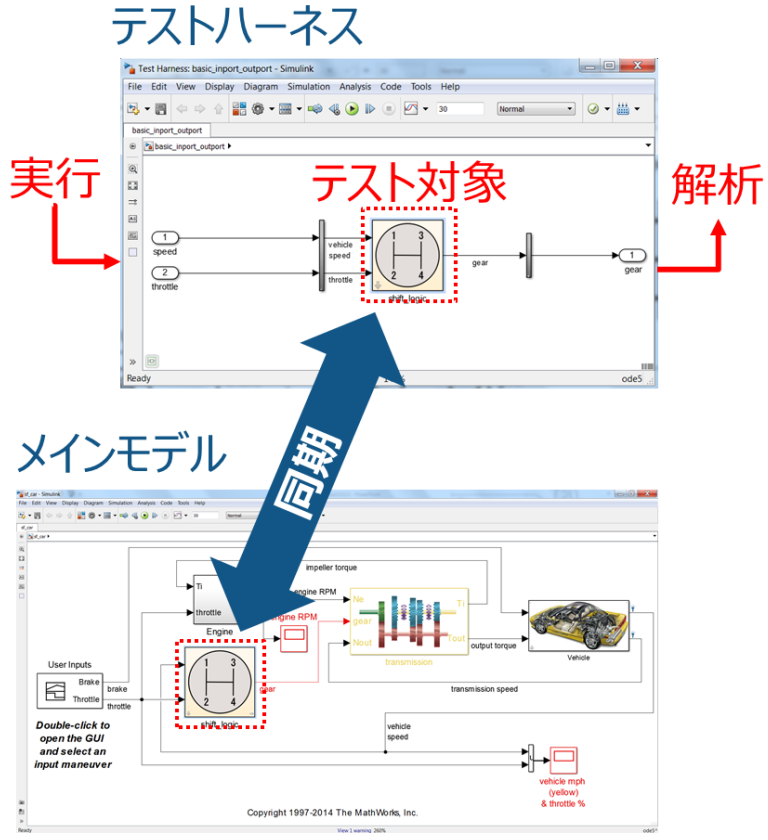
サブシステムだけテストしたい。

ロジックモデルとテストモデルを紐付けて管理したい。

修正の二重作業や手修正による修正の抜け漏れリスクを無くしたい。

解決策

Simulink Test の **テストハーネス** を使用し、
単体・統合テストをシームレスに

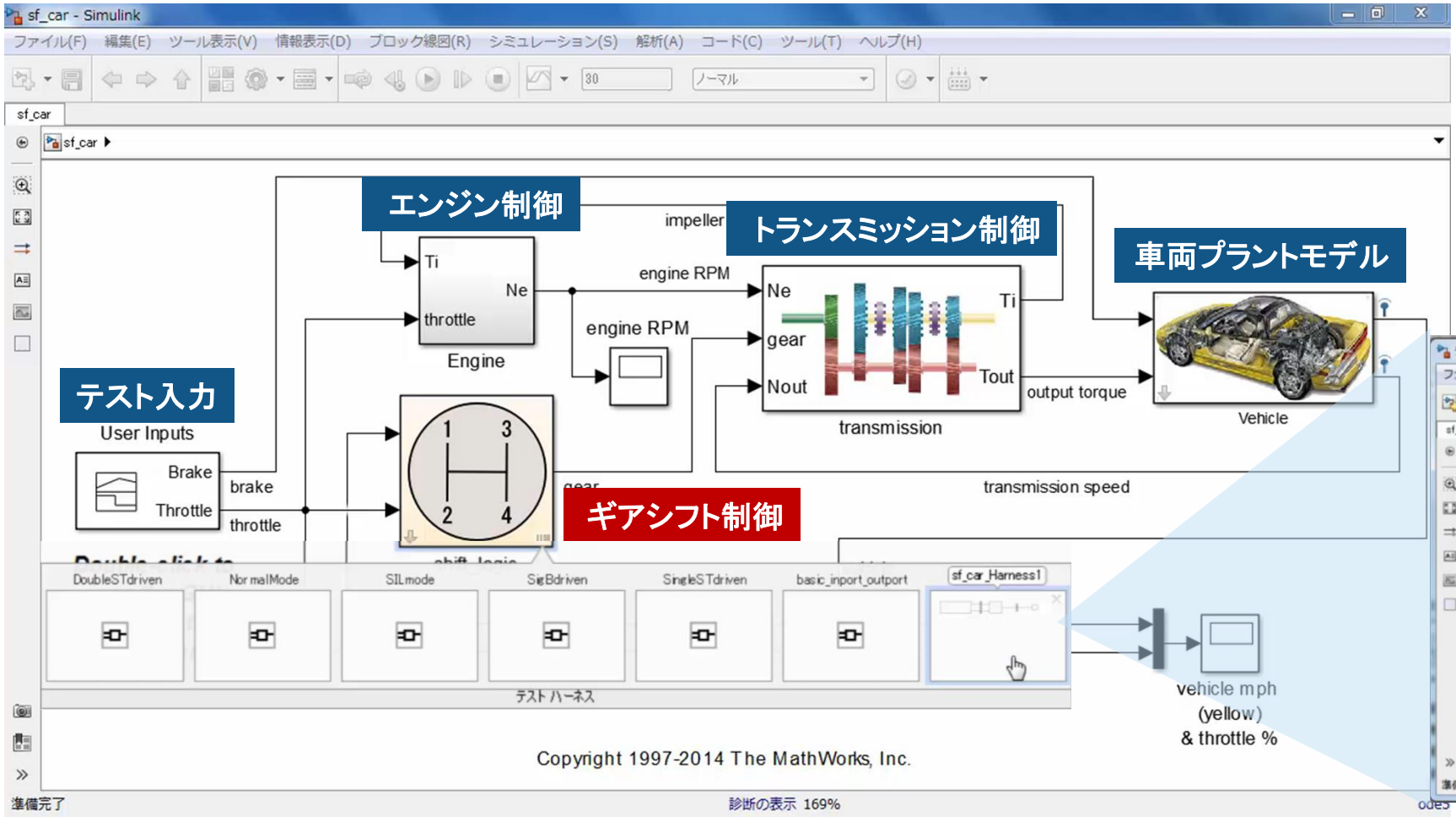


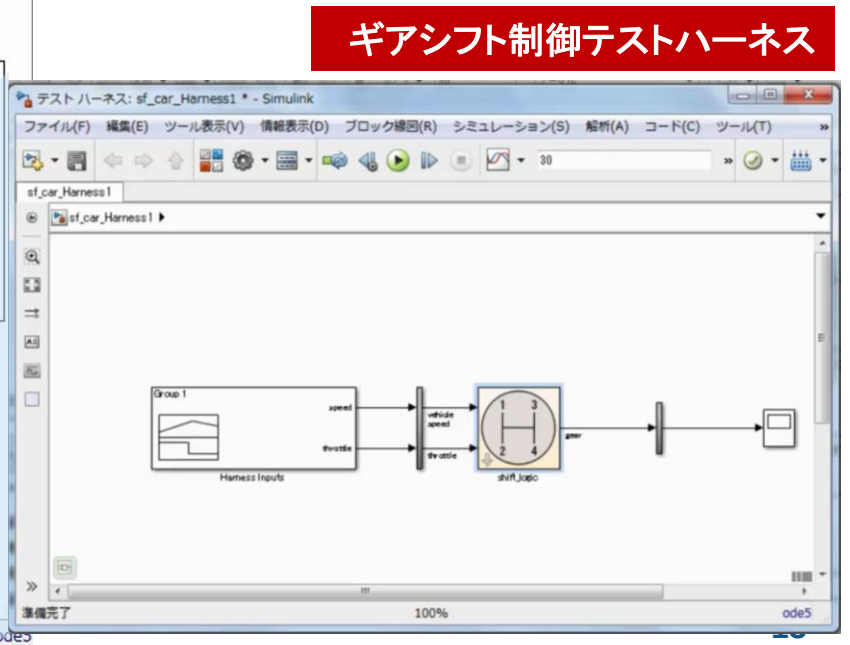
Simulink Test テストハーネス利用例

ギアシフト制御サブシステムのテストハーネスを作成

R2015a

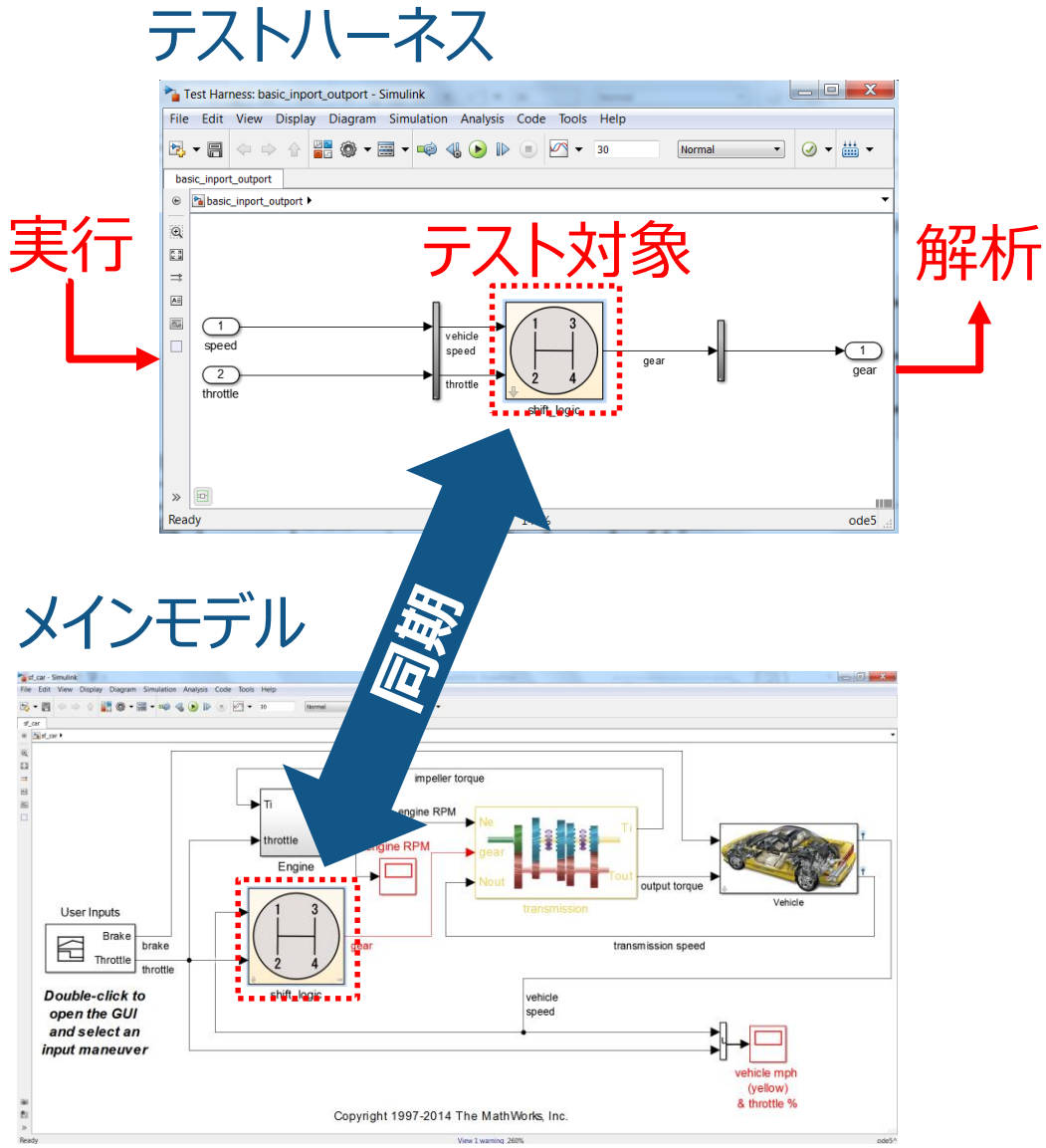
- テストハーネスを用いてハーネスモデルの作成・関連付けをより簡単に





テストハーネス
単体・統合テストをシームレスにします

- モデル全体・サブシステム単体・参照モデル用テストハーネスモデルを作成・関連付け
- 複数テストハーネスを作成・管理可能
- 様々なテスト入出力ブロックを設定可能
- テストハーネスはメインモデルに同期、修正内容を自動で反映
- Data Store Memoryに対応
- SIL/PILに対応



テスト入力信号の作成における課題＆ソリューション

課題

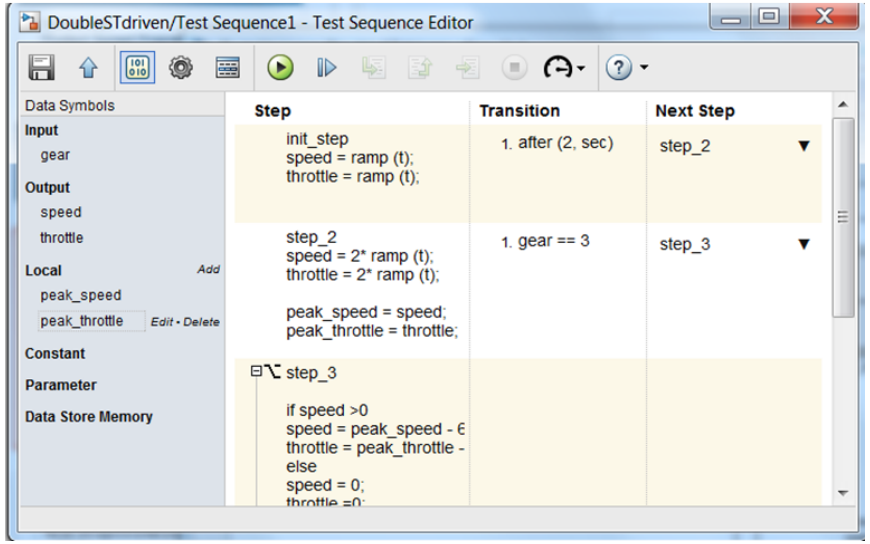
Design Engineer

シナリオベースでテスト入力信号を作成したい

解決策

Simulink Test の **テストシーケンスブロック** を使用し、複雑なテストパターンを簡単に作成できる

Test Sequence1



Simulink Testテストシーケンスブロック利用例

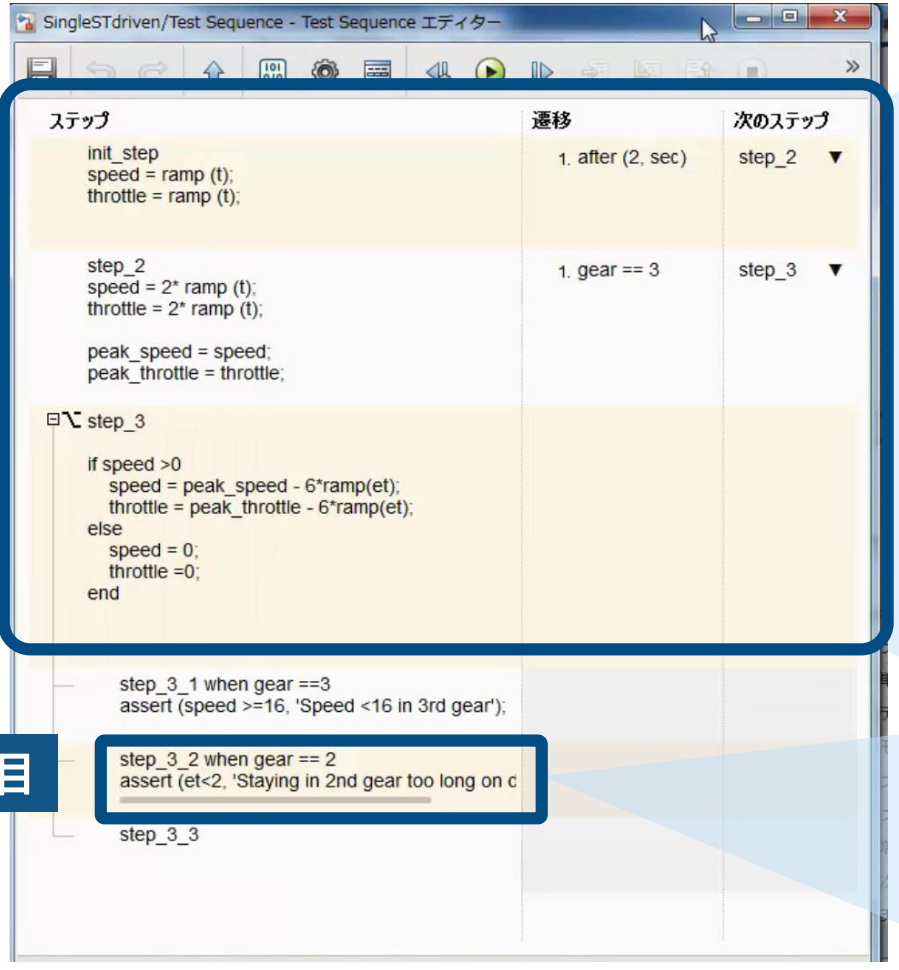
R2015a

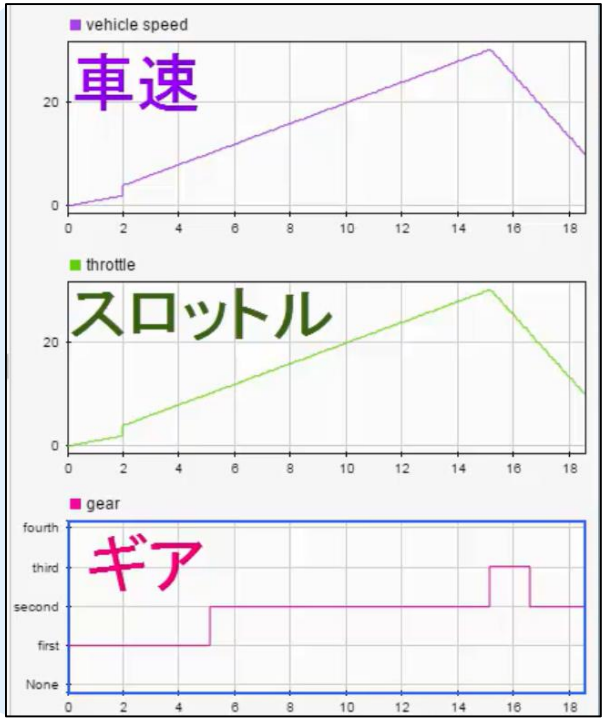
ギアシフト制御のテスト入力や信号検証を定義

- 複雑な入力パターン・検証項目を定義可能

テスト入力

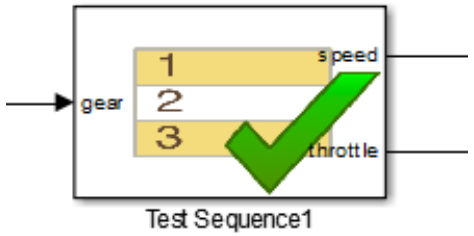
検証項目

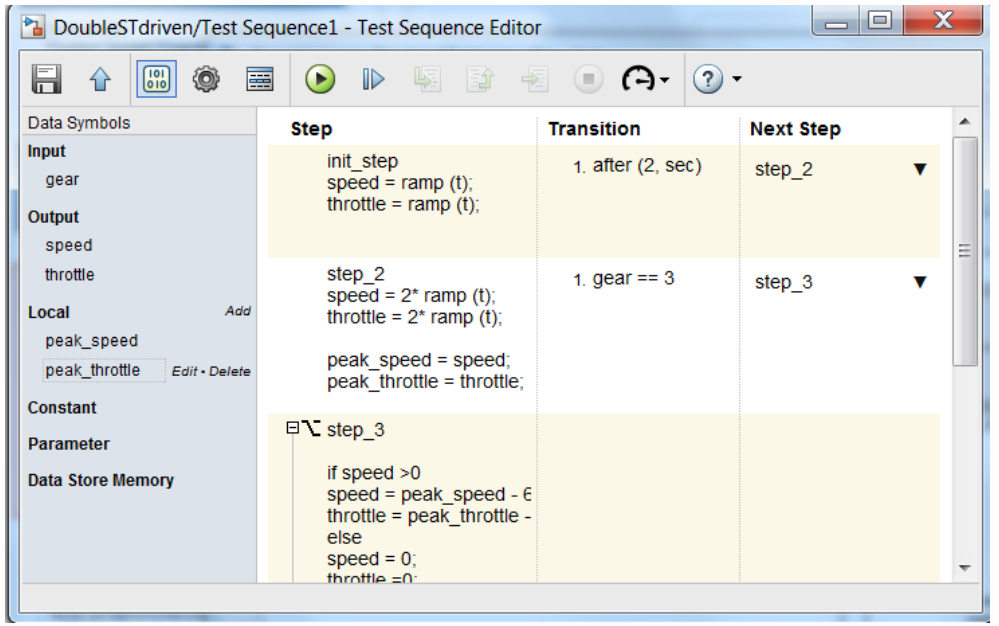




テストシーケンスブロック
複雑なテストパターンを簡単に作成できます

- 状態遷移表を用いて複雑なテストパターンを作成可能
- プラント出力や内部状態を入力として受け取って、テストパターンを切り替え可能 (動的タイミングチャート)
- 診断 (assert) 挿入による信号チェック・不具合検証が可能





テストケースの作成・管理・実行における課題＆ソリューション

課題

Test Engineer

膨大なテストを自動化したい

テスト項目を管理・再利用したい

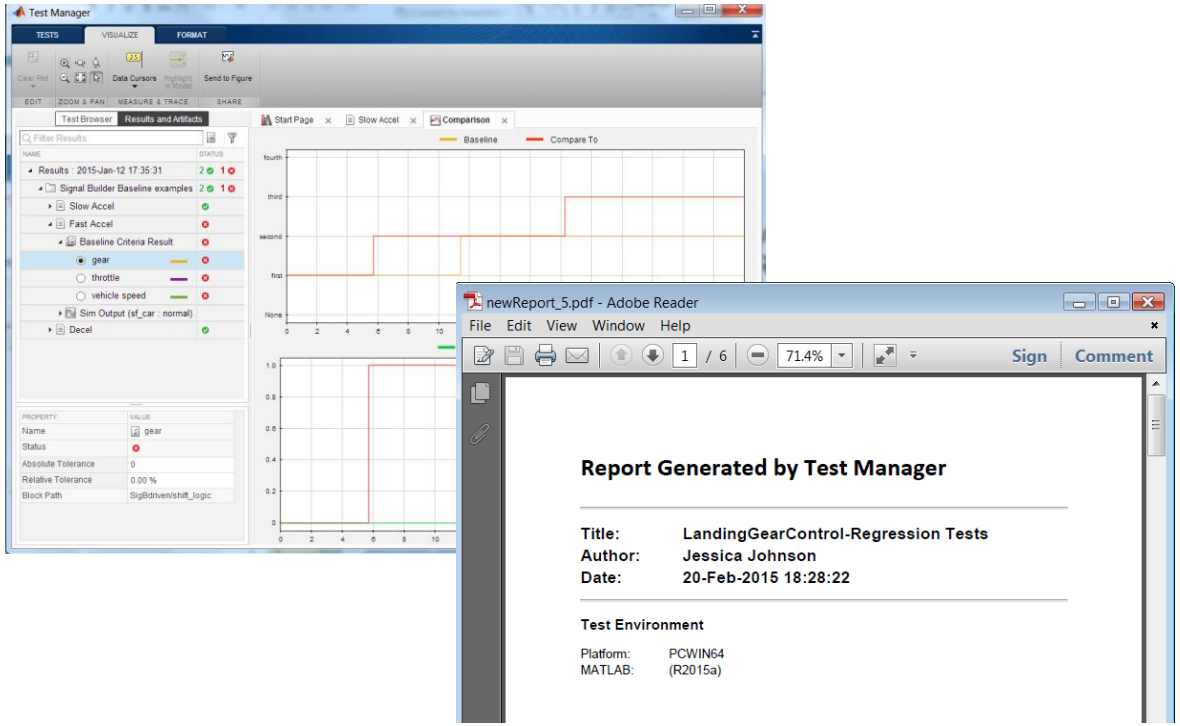
結果をレポートに纏めるのが大変

Manager

テストの結果レポートを見たい

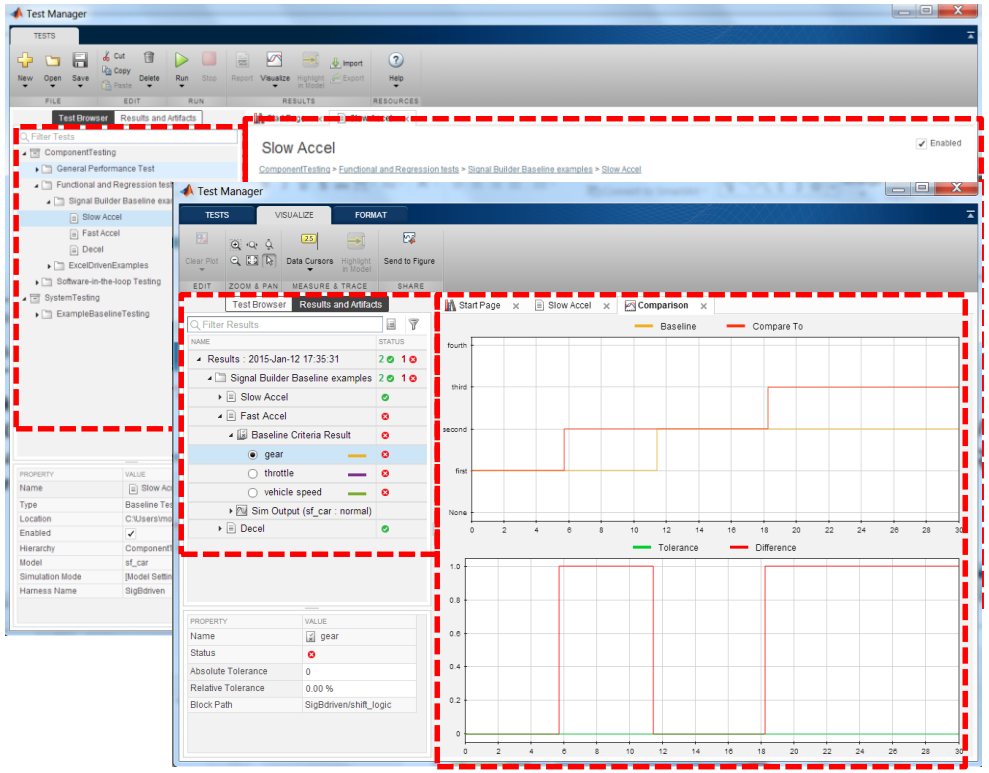
解決策

Simulink Test の **テストマネージャ** を使用し、テスト実行や合否判定とレポート作成の自動化が可能

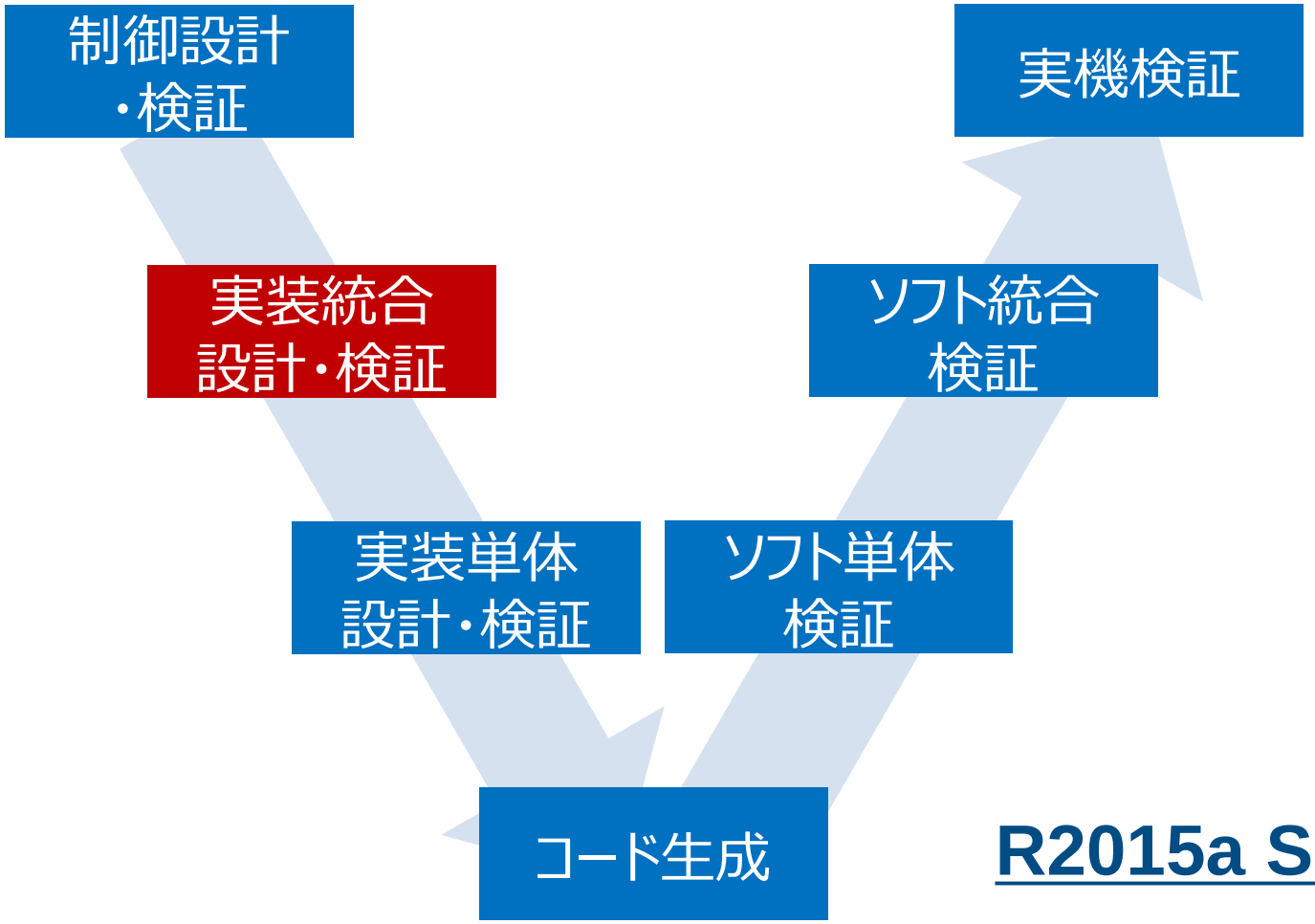


テストマネージャー
複数テストを自動実行して合否レポートを作成可能

- 複数テストを統合管理
- バッチ処理による一括テスト実行
- テスト対象としてモデル全体
  or テストハーネスを指定可能
- テスト結果レポート作成
- Excelからの信号読み込みに対応
- テスト時パラメータ上書き
- テスト用コールバック処理
- MIL/SIL/PILに対応



新しい検証ソリューション



GUI/HMI ブロックを提供する
Dashboard ライブラリ

テスト自動化や一元管理ツール
Simulink Test

信号依存関係の分析機能
モデルスライサー

R2015a Simulink Design Verifier 新機能

大規模なモデルのレビュー・デバッグにおける課題＆ソリューション

課題

Design Reviewer

この信号は上流のどこで計算されている？

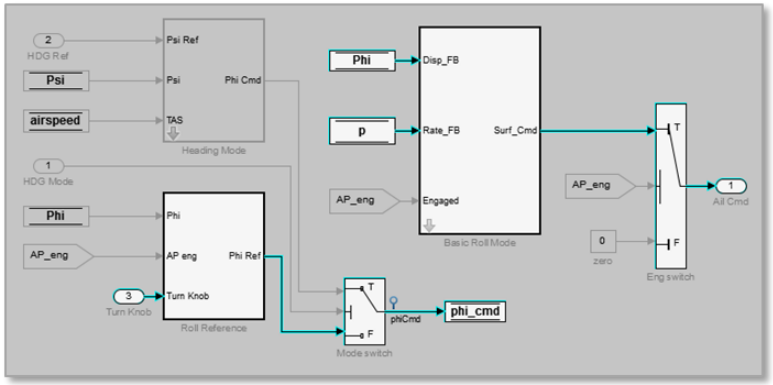
下流のどの信号に影響する？

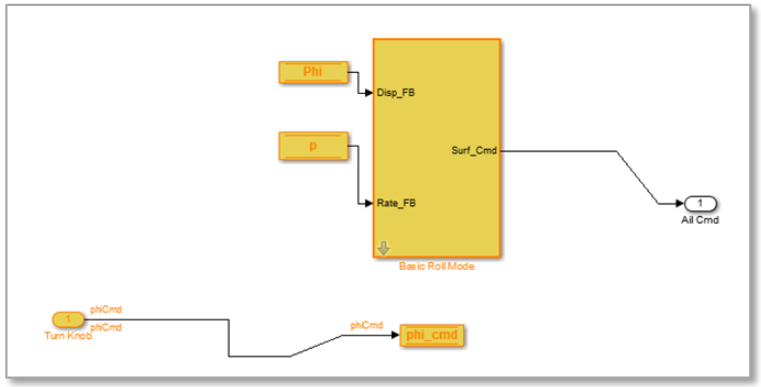
Design Engineer

デバッグする時に
必要な要素だけ取り出したい

解決策

Simulink Design Verifier の **モデルスライサー** を使用し、モデル内の信号依存性解析やモデルの切り出しが可能



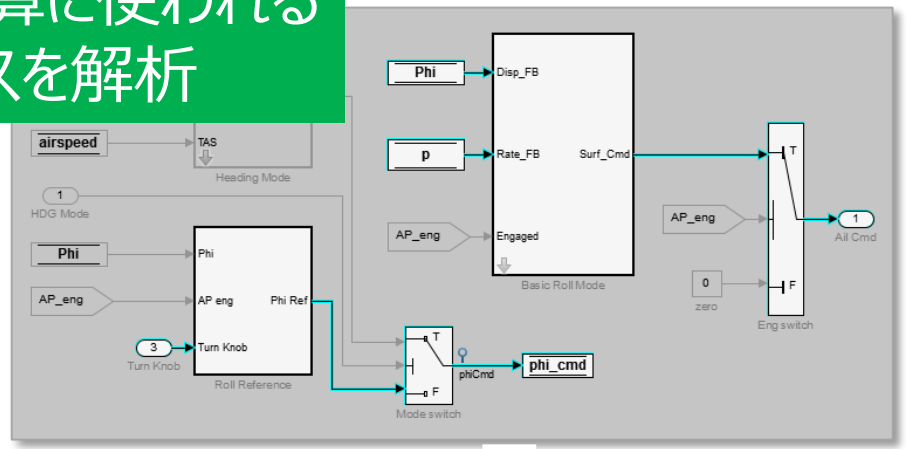


モデルスライサー

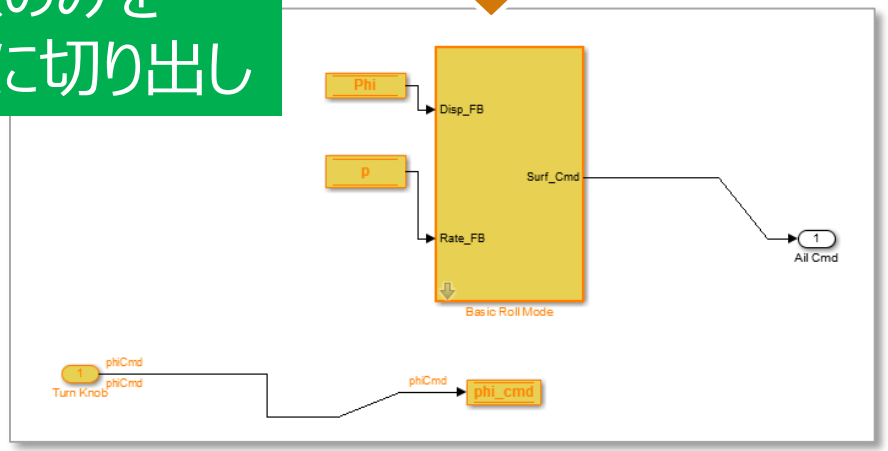
モデルから着目点に基づいて依存関係の解析やモデルの作成を実現

- ブロック線図の静的依存関係の解析
  - 上流方向/下流方向/両方向
- スライスモデルの作成
  - 独立したモデルとして抽出 (開始点から上流方向のみ)
- 実行パスに基づく動的依存関係の解析
  - シミュレーションで利用するブロックの考慮
- 依存関係パス上での設定
  - 解析除外ブロックを除外点として指定可能
  - 条件分岐に対する特定パスを制約点として指定可能

出力計算に使われる
実行パスを解析



実行パスのみを
別モデルに切り出し



※ R2015a 時点では Stateflow® や MATLAB® Function ブロック内パス解析は未対応（全処理がそのまま残ります）

まとめ

検証ツールを用いて、
早期検証作業をより簡単に

- 動作確認時のパラメータ調節をもっと簡単に：
  → **GUI/HMIブロックでテスト環境が充実**　　**Dashboard ライブラリ**

- 膨大なテストケースをより効率的に実行や管理するため：
  → **テスト自動化の実施**　　**Simulink Test**

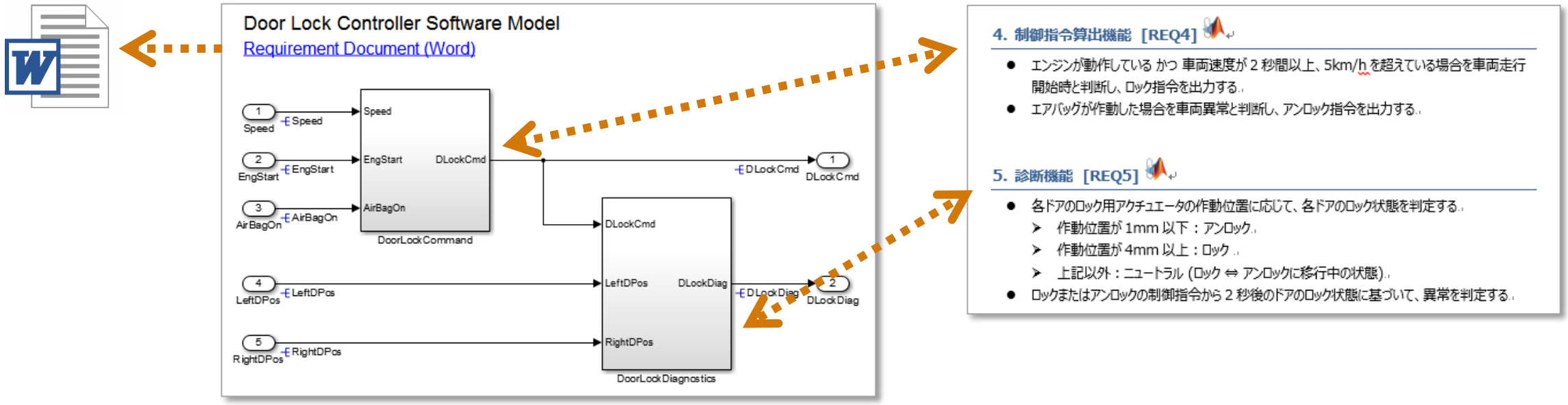
- デザインレビューやロジック検証をもっと効率化するため：
  → **依存関係分析の実施**　　**モデルスライサー**

<付録>

モデル・仕様書間リンクで
トレーサビリティを確保、仕様修正の影響範囲を特定

ハイパーリンク

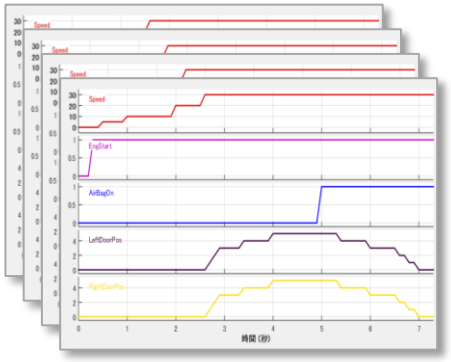
モデル・仕様書間リンク



- **Simulink Verification & Validation™ ：モデル・仕様書間リンク**
  ※ハイパーリンクはSimulink標準機能

カバレッジ測定でテストのヌケモレ発見

- テストをスルーした未実行パスが引き起こす不具合の防止に貢献



テスト入力データ

シミュレーション

モデルカバレッジ

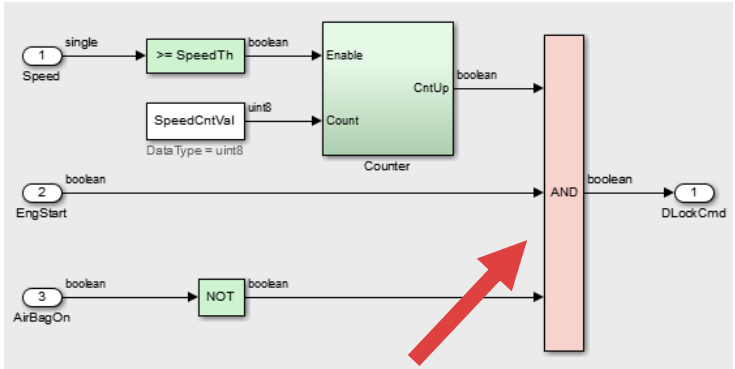
モデルの階層構造/複雑度		テスト 1				
		D1		C1		MCDC
1. DoorLockCntr	13	90%		100%		54%
2. . . . DoorLockCommand	1	100%		100%		67%
3. . . . . Compare To Constant1		NA		100%		NA
4. . . . . Counter	1	100%		100%		NA

S-Functionコードカバレッジ

S-Function ブロック "sldemo_sfun_counterbus"

親: sldemo_lct_bus/TestCounter
カバーされていないリンク:

メトリック	カバレッジ
循環的複雑度	3
条件 (C1)	67% (4/6) 条件結果
判定 (D1)	75% (3/4) 判定結果
MCDC (C1)	50% (1/2) 結果に影響を与えた条件



未達箇所色分け表示

計測可能カバレッジ例

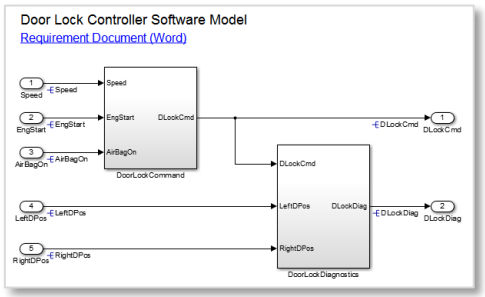
- ステートメント（C0）：コードのみ
- 条件（C1）
- 判定（D1）
- MCDC
- 境界：モデルのみ

※複数テストの累積カバレッジも計測可能

- **Simulink Verification & Validation™ ：カバレッジ測定**

フルカバレッジ入力データ生成でユーザテスト入力を補充

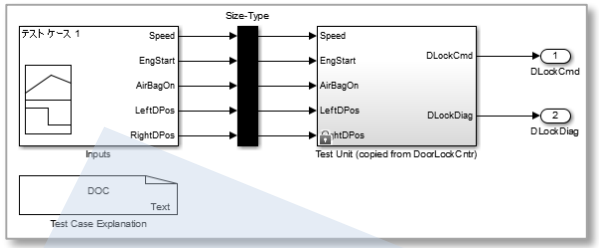
- レアケースを自動生成→テスト品質向上に貢献
- モデル生成コードの網羅的な等価性検証にも利用可能

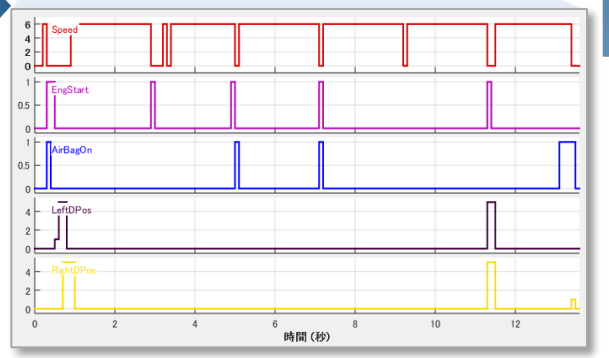


モデル

解析

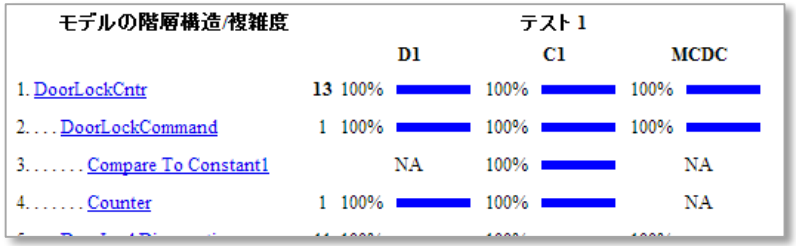
テストモデル





フルカバレッジ入力データ生成

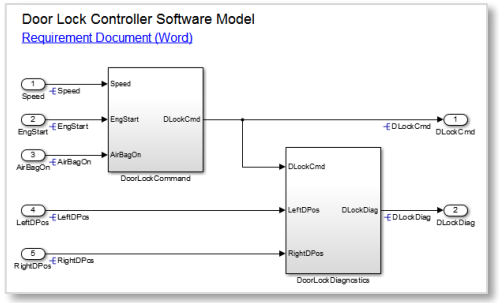
シミュレーション



モデルの階層構造/複雑度	テスト 1					
		D1		C1		MCDC
1. DoorLockCntr	13	100%		100%		100%
2. . . . DoorLockCommand	1	100%		100%		100%
3. . . . . Compare To Constant1		NA		100%		NA
4. . . . . Counter	1	100%		100%		NA

- **Simulink Design Verifier™ ：フルカバレッジ入力データ生成**

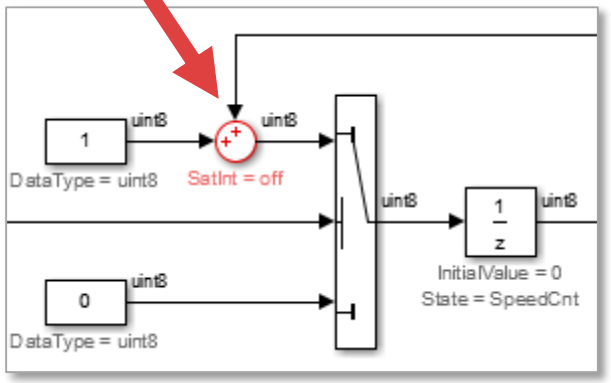
ランタイムエラー検出でテスト困難な不具合混入の防止

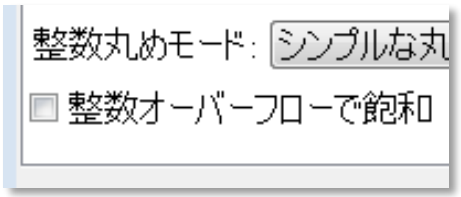


モデル

解析

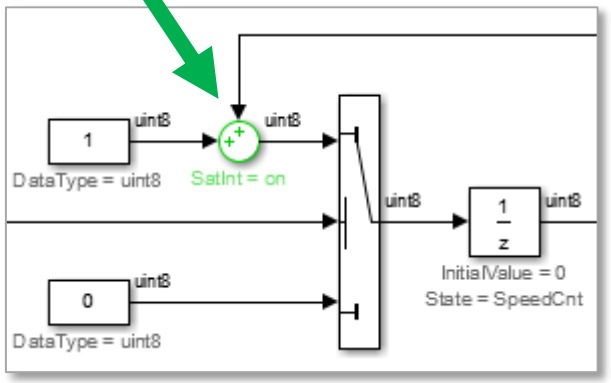
整数オーバーフローリスク有り

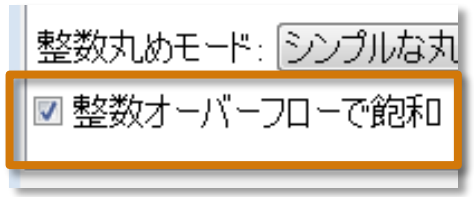




修正

整数オーバーフローリスクを除去



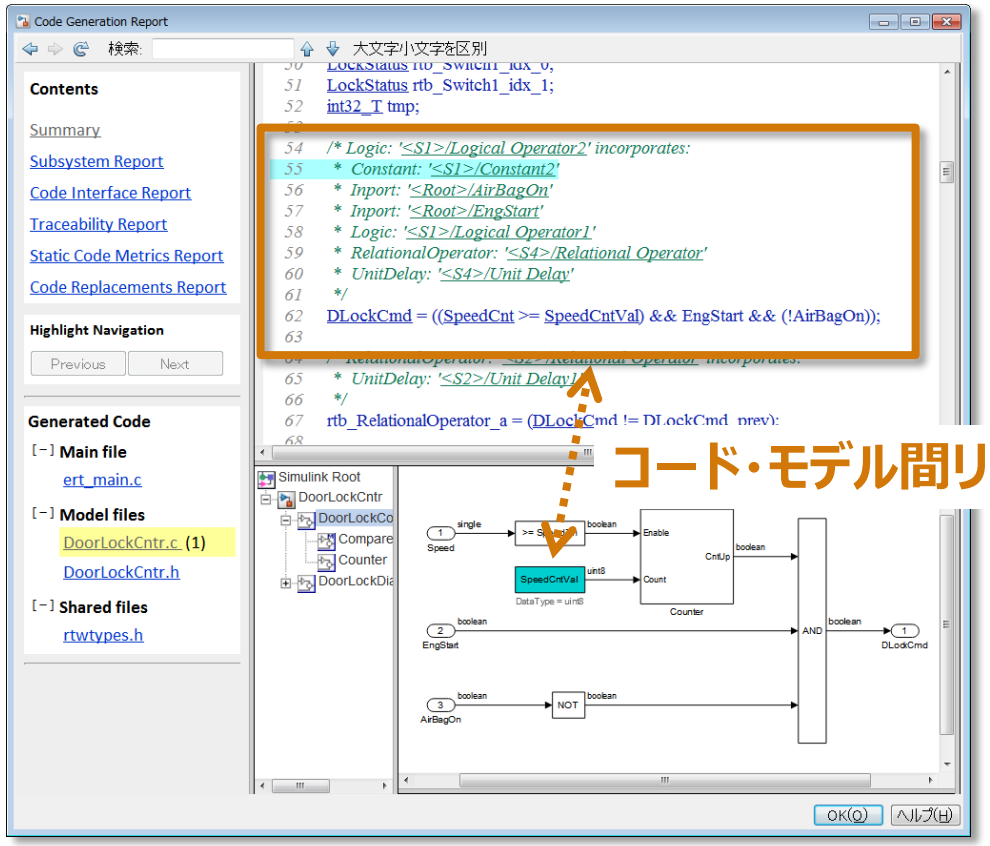


検出可能エラー例

- 整数演算オーバーフロー
- ゼロ除算
- 配列の範囲外アクセス
- デッドロジック

- **Simulink Design Verifier™ ：設計エラー検出**

コード生成レポートで快適なコードレビュー



コード・モデル間リンク

コード生成レポート

※Simulink Report Generatorがない場合はモデルと直接リンク

- モデル⇔コードのトレーサビリティ確保
- コード統計で簡単な性能見積もりが可能

Number of .c files : 1
Number of .h files : 2
Lines of code : 182
Lines : 488
[−] File details

コード統計情報
行数/グローバル変数/ローカル変数スタックサイズ等

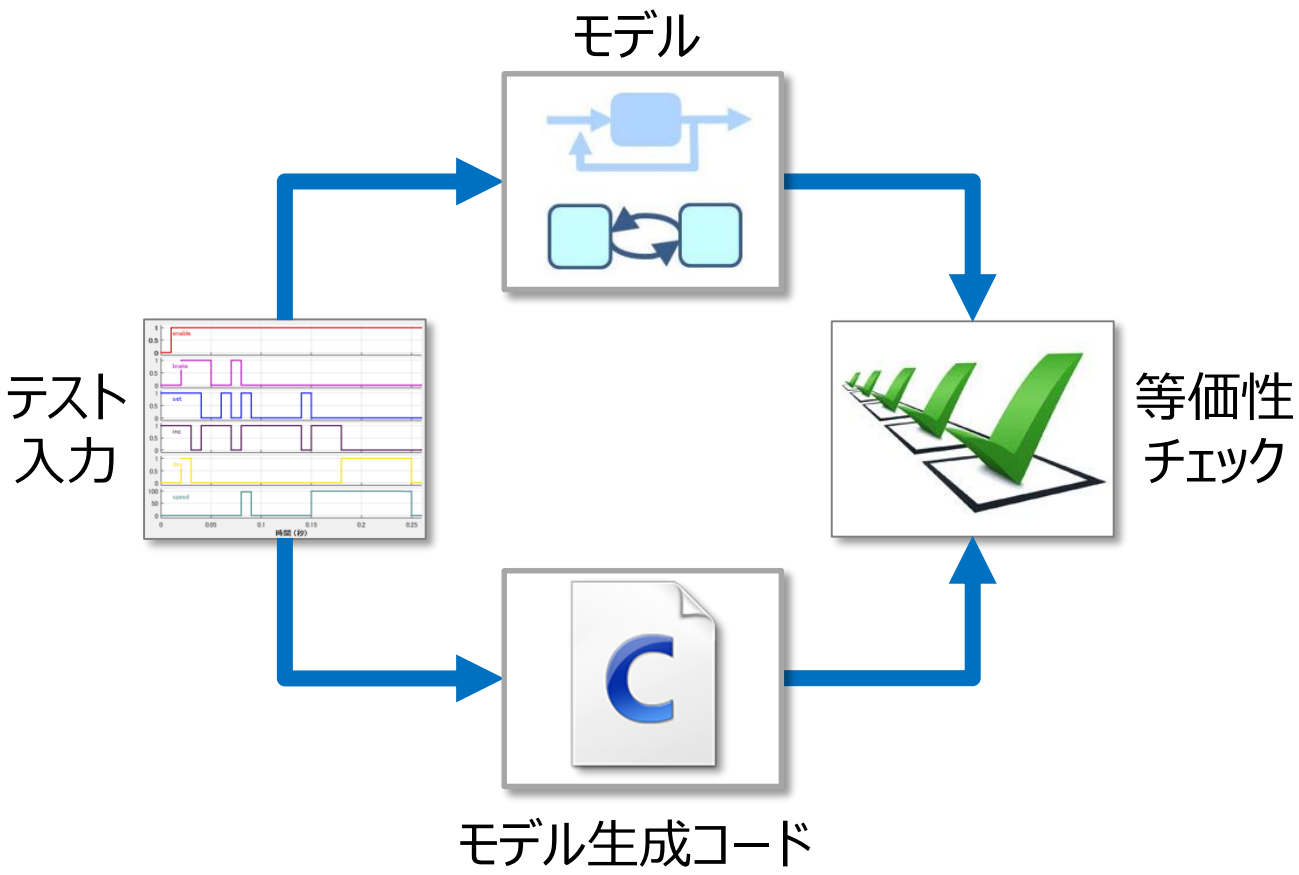
2. Global Variables [hide]

Global variables defined in the generated code.

Global Variable	Size (bytes)	Reads / Writes	Reads / Writes in a Function
DLockDiag	4	4	3
[+] DoorLockCntr_DW	2	8	8
DLockCmd	1	5	5
DLockCmd_prev	1	2	2
DiagCnt	1	5	5
SpeedCnt	1	4	4
Total	**10**	**28**	

- **Embedded Coder®**：コード生成レポート
- **Simulink Report Generator™**：モデルビュー作成

モデル生成コード等価性検証でコードの動作保証・性能評価



SIL (Software In the Loop)

PC CPU上でコード実行

PIL (Processor In the Loop)

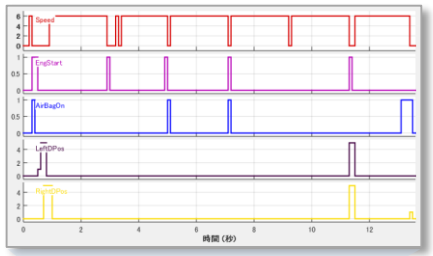
MCU/シミュレータ・エミュレータ上でコード実行

※PIL対応しているかはMCU/IDEによって状況が異なるので要確認

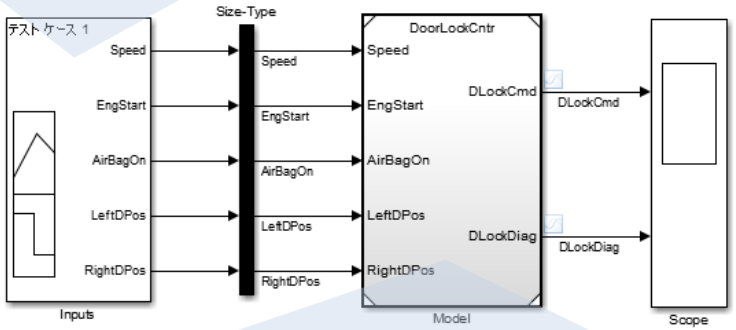
- モデル＆コードの動作等価性
- コード生成ツール設定ミス
- コード生成ツール不具合
- コンパイラ不具合
- 処理系依存動作
- メモリ消費量評価
- 実行速度評価

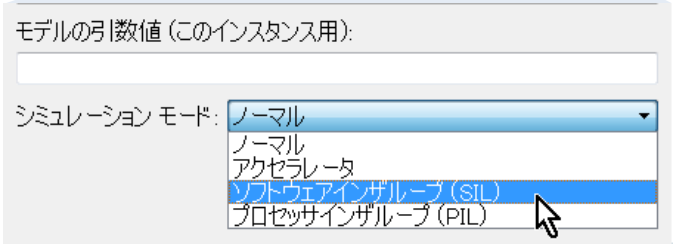
PILのみ

モデル生成コード等価性検証自動化で検証作業の合理化



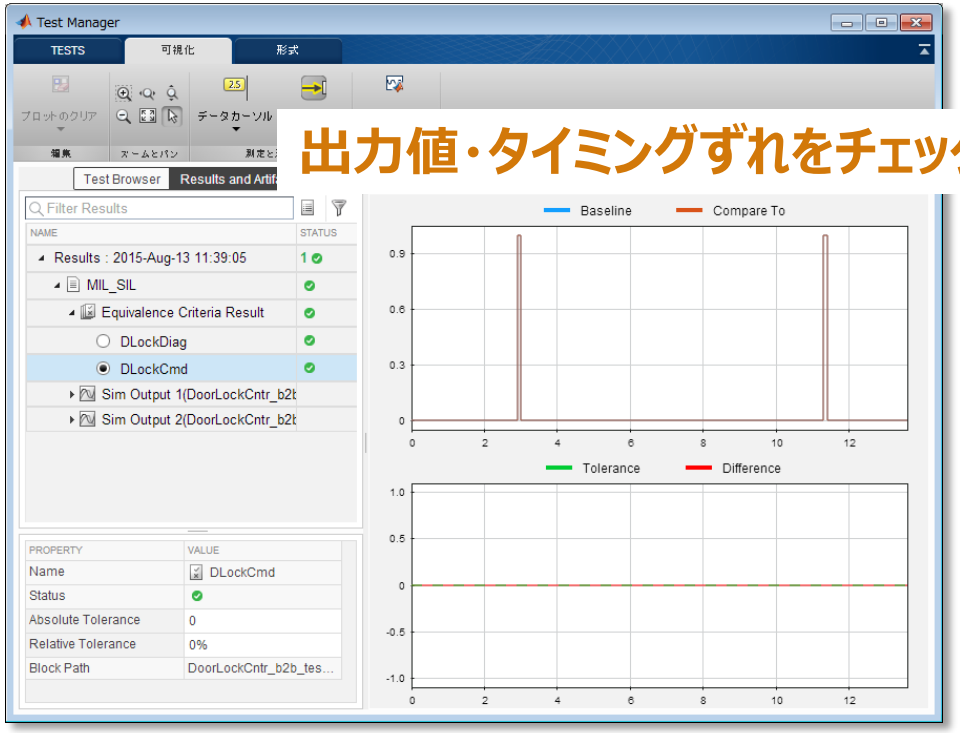
フルカバレッジテスト入力
（Simulink Design Verifier利用）





ノーマル（モデル）とSIL/PILの結果を比較

テストマネージャー

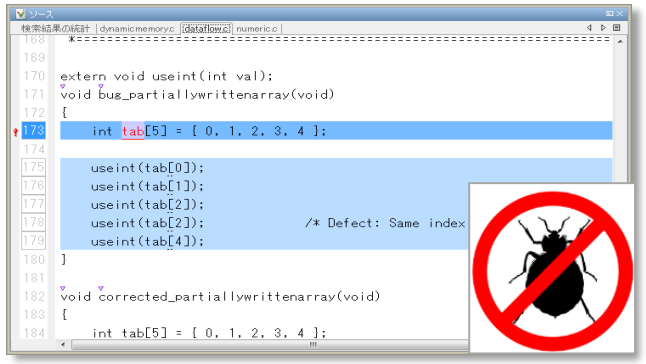


出力値・タイミングずれをチェック

- **Embedded Coder®**：SIL/PILモード
- **Simulink Test™**：テストマネージャー

コード静的解析によるソフト全体の信頼性確保

ハンドコード内エラーやソフト統合時の不具合を検出可能



- バグ検出
  - 高速な解析
- コーディングルールチェック
  - MISRA-C準拠
- コードメトリクス解析
  - コード複雑度

グリーン：正常
ソースコードが安全と証明

レッド：エラー
実行される度にランタイムエラー

グレー：デッドコード
無実行

オレンジ：Unproven
条件によってランタイムエラー

パープル：Violation
MISRA-C/C++, JSF++

```
static void pointer_arithmetic (void) {
    int array[100];
    int *p = array;
    int i;

    for (i = 0; i < 100; i++) {
      *p = 0;
      p++;
    }

  if (get_bus_status() > 0) {
    if (get_oil_pressure() > 0) {
      *p = 5;
    } else {
      i++;
    }
  }

  i = get_bus_status();

  if (i >= 0) {
    *(p - i) = 10;
  }
}
```

variable 'I' (int32): [0 .. 99]
assignment of 'I' (int32): [1 .. 100]

変数値範囲
ツールチップ

- **Polyspace Bug Finder™ ：**
  軽量なバグ検出・コード規約・メトリクス解析
- **Polyspace Code Prover™ ：**
  全分岐パス解析＆エラーの存在/不在を証明