MBDセミナー(基礎)
-モデルベース開発による 高品質な車載ソフトウェア開発-

2021/10/22

富士通株式会社
エンベデッドビジネス統括部
小野寺　純一

目次

- MBDとは
- MBDプロセス
- MBDにおける検証技法
- MBDデモ
- MBD実践編

FUJITSU

MBDとは

MBDとは

FUJITSU

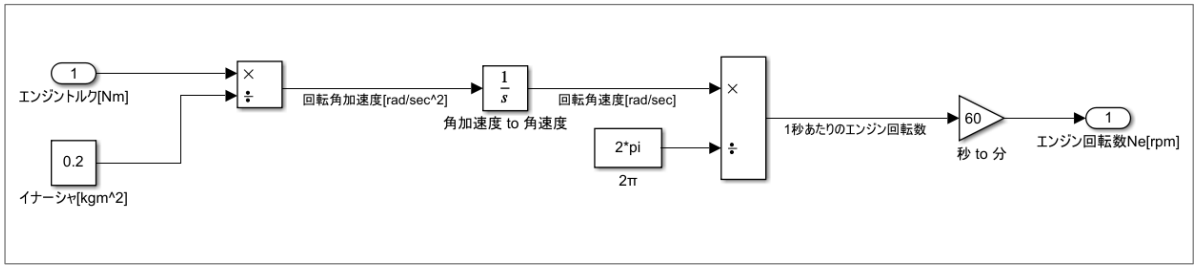
○Model Based Development：モデルベース開発

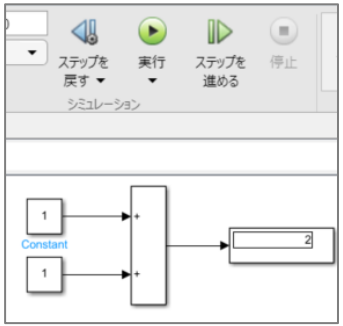
○モデルを起点として設計・実装・検証を行う開発手法

モデルとは？

動く仕様書　**設計中に検証可能**　**ソースコード自動生成が可能**

Simulink®モデル

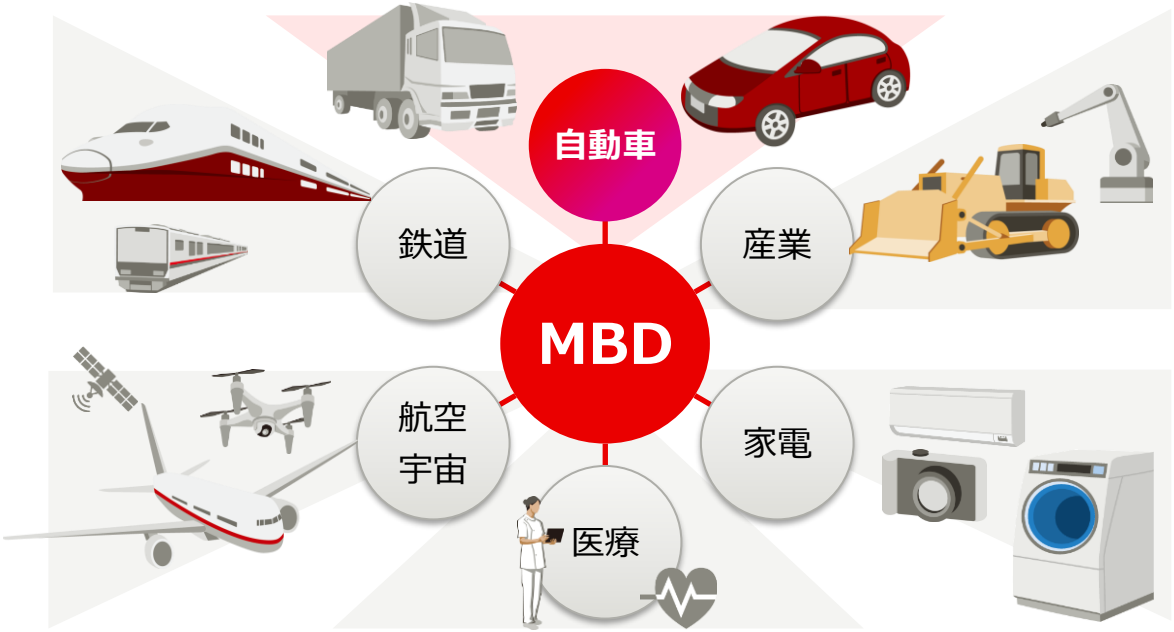




※本研修ではMathworks®社のMATLAB®/Simulink®を使ったMBDを説明します

MBDの導入分野

- ○自動車分野のソフトウェア開発ではデファクトスタンダード
- ○その他の分野にも適応が進んでいる



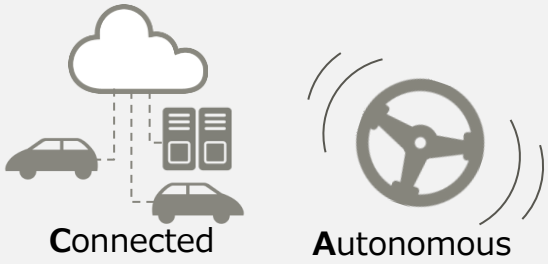
MBD導入背景

FUJITSU

業界の流れ

車載システムの複雑化

CASE化



課題

検証の効率化

開発の効率化

機能安全への対応
(ISO26262)

MBDによって解決

MBDのメリット

FUJITSU

1 工数削減＆品質向上

2 コミュニケーション円滑化

3 フロントローディングの実現

MBDのメリット

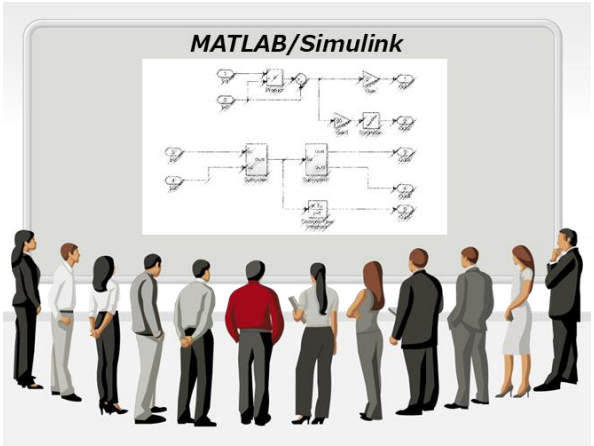
1 工数削減＆品質向上

- 上流工程におけるシミュレーションにより手戻り工数の削減が可能
- MBDツールには多数の部品があらかじめ用意されており、GUIで直感的に操作できるため、設計コンセプトの具体化が容易である
- モデルからコードを自動生成することができるため、開発工数の削減およびソフトウェア品質の向上が可能

MBDのメリット

2 コミュニケーション円滑化

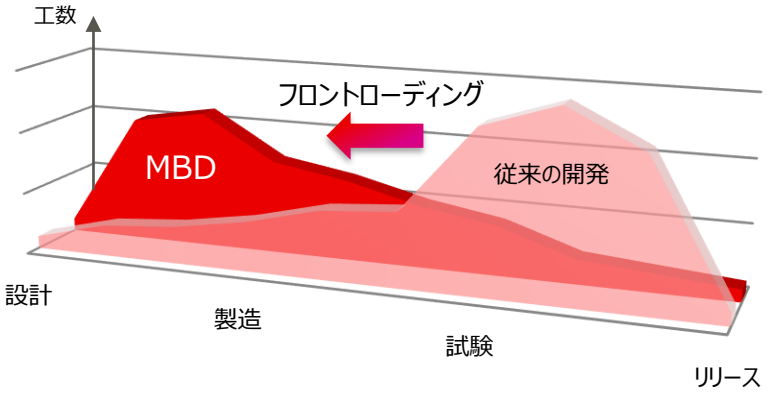
- 文章による曖昧さを回避できるため、仕様の書き手と読み手とのコミュニケーションが円滑になる
- MBDの適用はソフトウェアに限らず、ハードウェアなどにも適用可能なため、システム全体を共通言語で全員で共有することができる



MBDのメリット

3 フロントローディングの実現

- MBDでは、設計（モデル作成）をしながらシミュレーションを実行することができ、問題の早期発見が可能
- 制御対象もモデル化すれば、実機の完成を待たずにシステムの検証が可能



MBDのデメリット

FUJITSU

1 開発環境整備コストが高い

- 開発環境として、モデルベース開発ツールを揃える必要がある（以下の例は、Mathworks®社製品）
  - MATLAB®、Simulink® 、Stateflow®
  - コード生成ツール、検証ツール、制御対象のモデリングツール、など

2 モデルベース開発の知識が必要

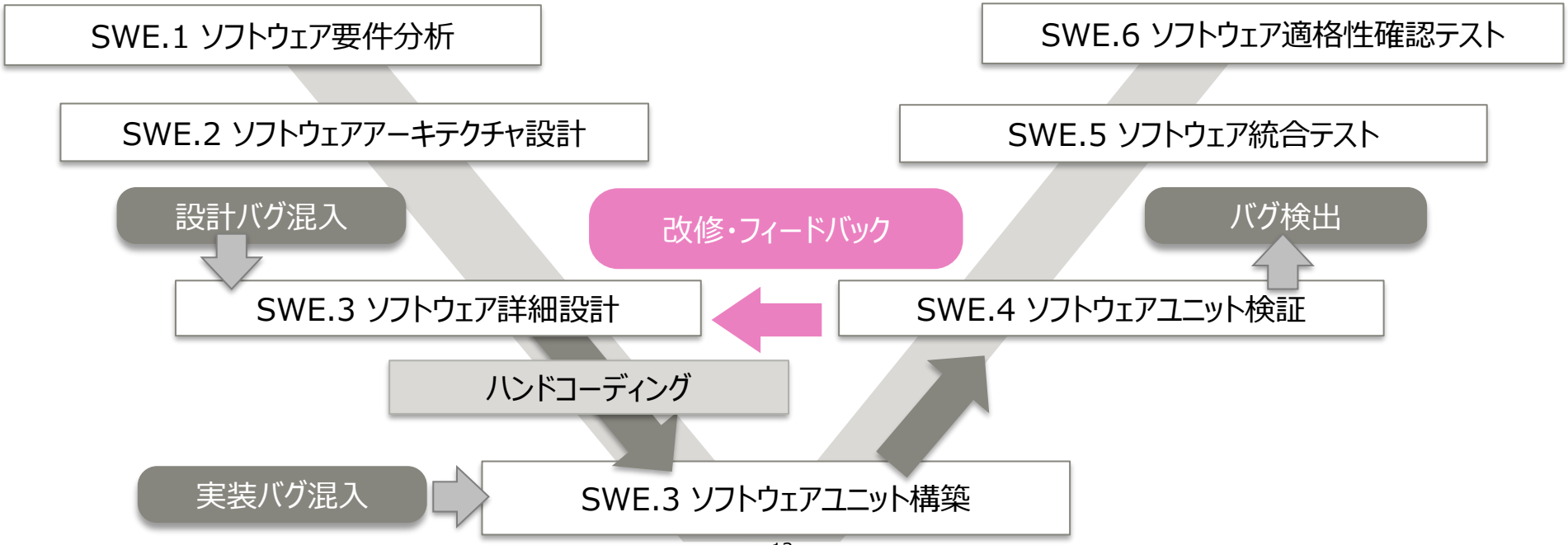
- 一般的な車載ソフトウェア開発を経験したエンジニアでも、慣れるまでが難しい
  - モデルを読んだり作成するには慣れが必要
  - ブロック毎に、詳細なチューニングパラメータが存在する

MBDプロセス

従来の開発プロセスV字モデル

FUJITSU

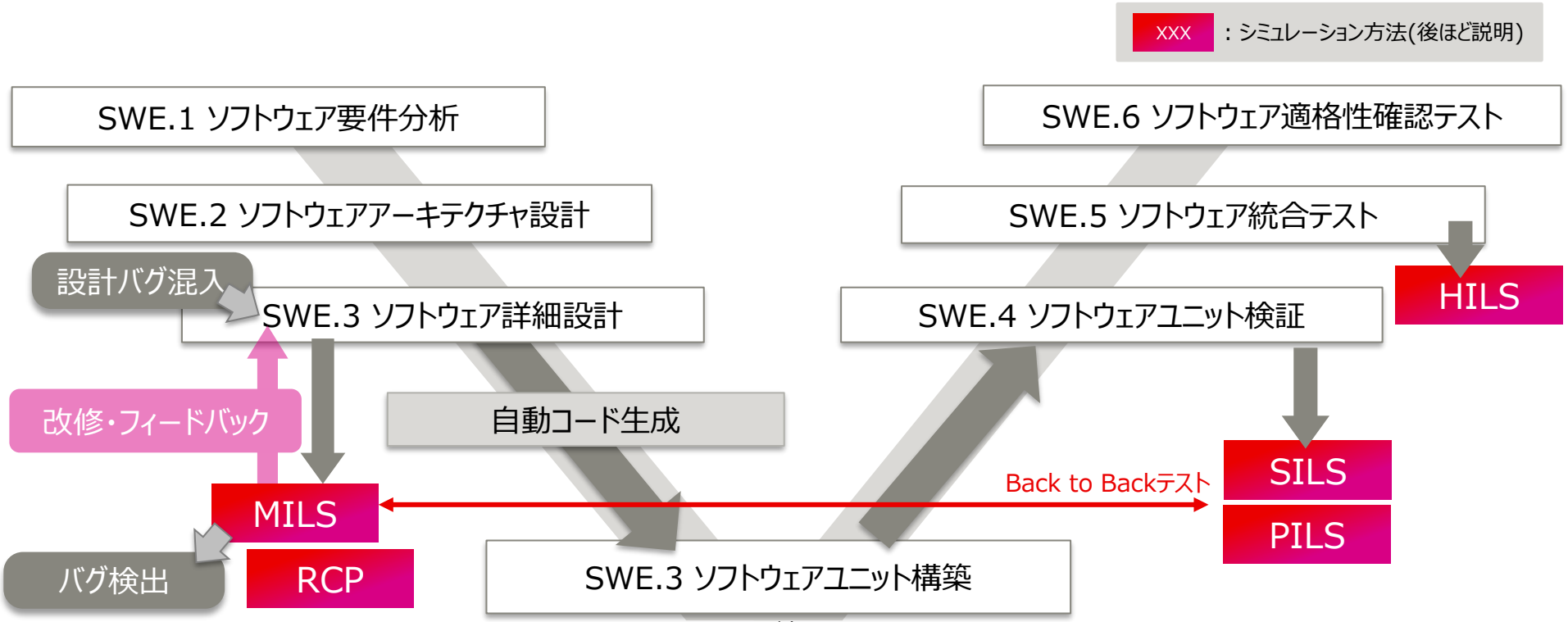
○設計内容がテストされずにV字の右側(検証フェーズ)に進むため、バグの検出が遅れて戻り工数が多くなってしまう



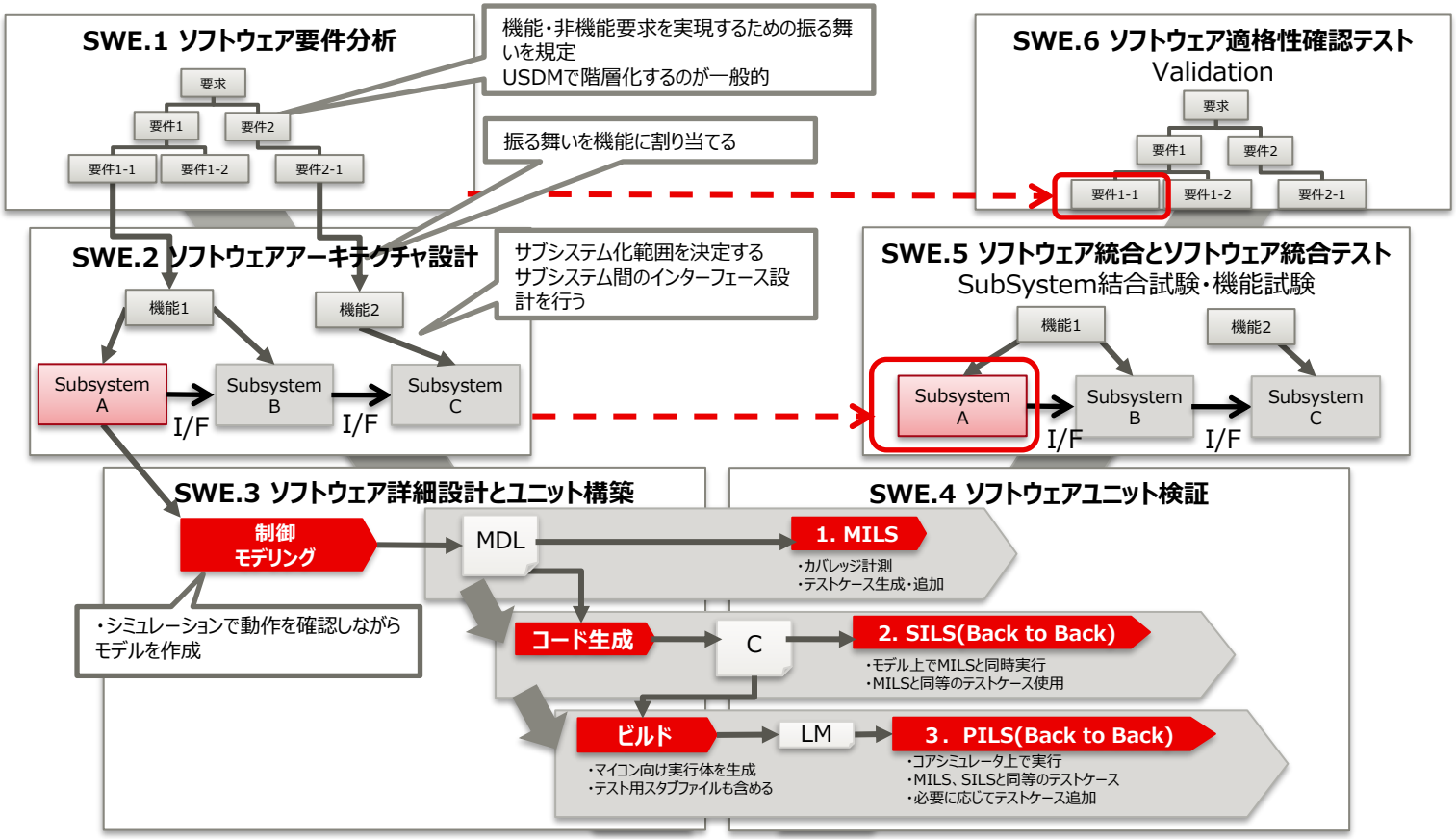
MBDの開発プロセスV字モデル

FUJITSU

○設計内容をシミュレーションすることで、設計工程でバグを検出・改修可能



MBDの開発フローの概略



各プロセスにおけるアクティビティ

FUJITSU

プロセス		アクティビティ
SWE.1	ソフトウェア 要件分析	■各制御要素で、機能・非機能要求を実現するための振る舞いを規定 ・・・USDMで階層化するのが一般的
SWE.2	ソフトウェア アーキテクチャ設計	■ソフトウェアの機能・構造を定義する ・・・機能:動的な振る舞いを表現する ・・・構造:コンポーネントレベルのサブシステム ■各サブシステムの実装方式を決定する ・・・Simulink®／Stateflow®／ハンドコード ■サブシステム間のインターフェース設計を行う
SWE.3	ソフトウェア 詳細設計	■モデルの構造を設計する ・・・ユニットレベルのサブシステム ■アーキ設計従いモデリングを行う ・・・シミュレーションで動作を確認しながら設計する ・・・動的な振る舞いを表現する ・・・JMAAB等のガイドラインに従うのが一般的

各プロセスにおけるアクティビティ

FUJITSU

プロセス		アクティビティ
SWE.3	ソフトウェア ユニット構築	■モデルからソースコードを生成する ■ソースコードをビルドしロードモジュールを生成する
SWE.4	ソフトウェア ユニット検証	■モデル設計どおりにロードモジュールが動作することを評価する ・・・SILS、PILS環境で検証するのが一般的
SWE.5	ソフトウェア 統合テスト	■アーキ設計にて規定されたとおりに振る舞うことを評価する ・・・実機環境、またはHILS環境で検証するのが一般的
SWE.6	ソフトウェア 適格性確認テスト	■ソフトウェア要件分析にて規定されたとおりに振る舞うことを評価する

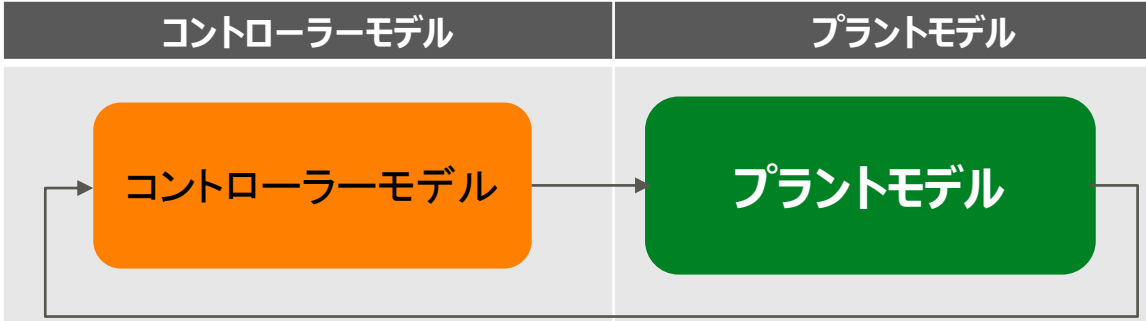
FUJITSU

MBDにおける検証技法

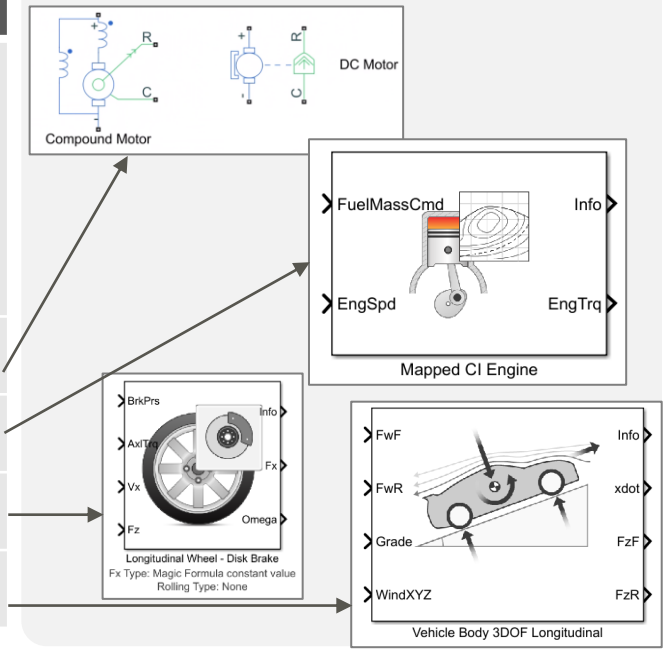
検証技法の前に　プラントモデルとは

FUJITSU

- コントローラーが制御する対象をモデル化したものをプラントモデル(＝制御対象モデル)という
- プラントモデルを使うと実機レス検証が可能になる

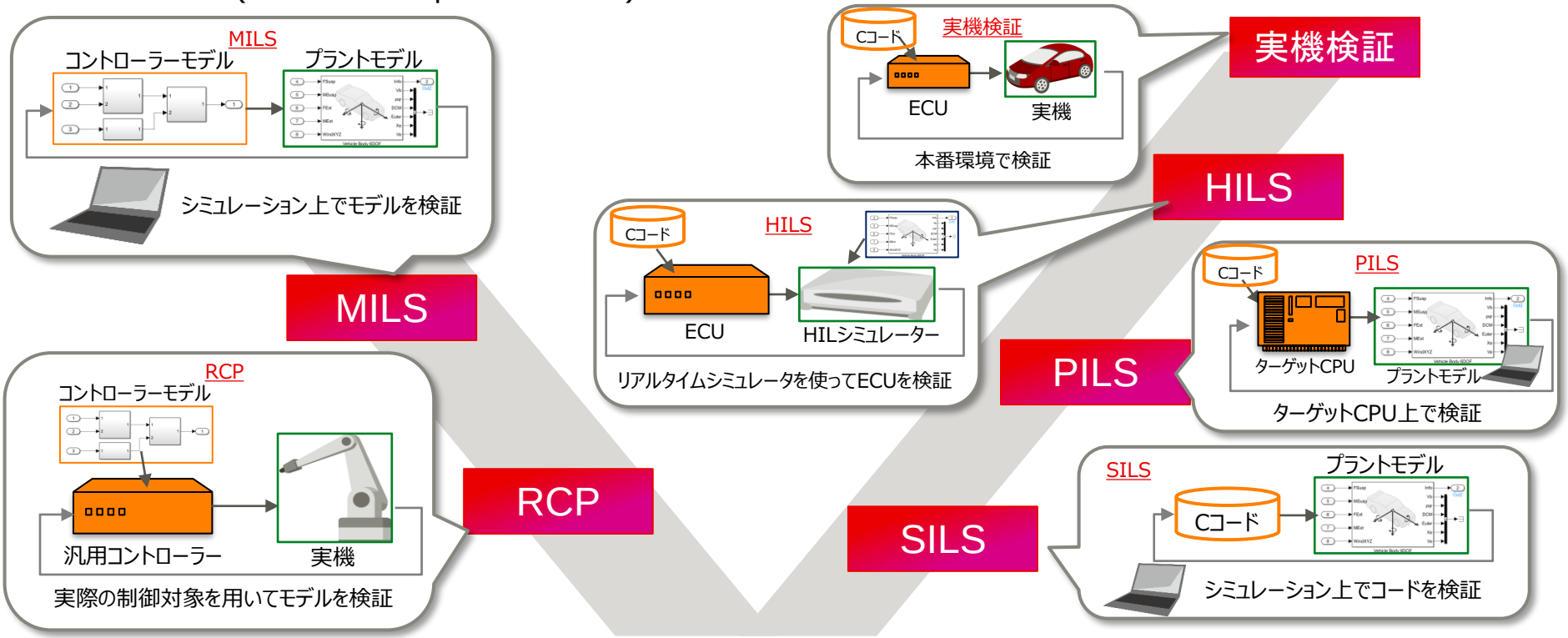
コントローラーモデル	プラントモデル
	
モーター制御	モーター自体
エンジン制御	エンジン自体
衝突回避制御	ブレーキ自体(または車自体)
自動運転制御	車自体

プラントモデルイメージ



MBDにおける検証技法

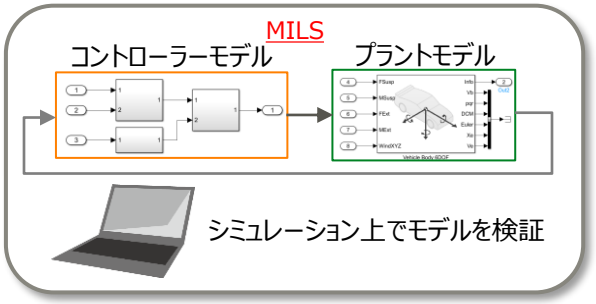
- MBDにおける検証技法は、MILS,RCP,PILS,HILSなどがある(すべて実施するわけではなく、PJによって取捨選択する)
- 基本的にXILS(X In the Loop Simulation)と名称がつけられており、X部はシミュレーションのレベルによって変わる



MILS

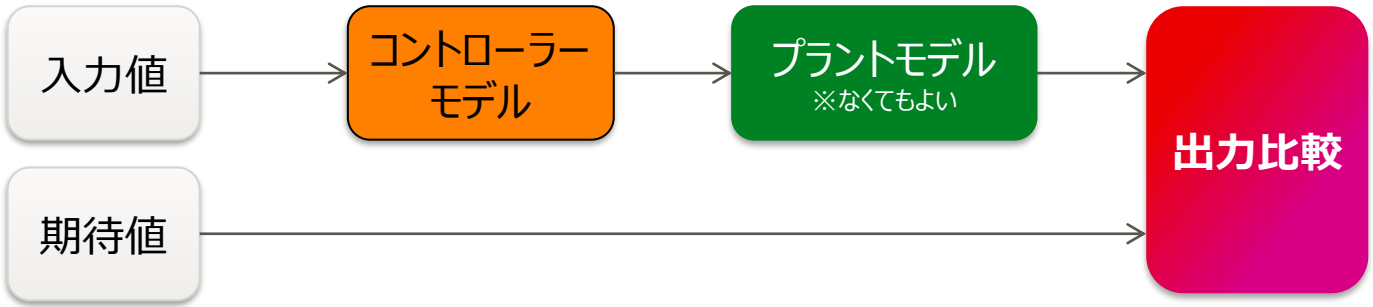
FUJITSU

Model In the Loop Simulation ：モデル上で実行検証



- 実行環境
  - MATLAB®/Simulink®モデル
- 目的
  - ロジックの妥当性を検証する
- 検証観点
  - モデル設計により期待される出力と実際のモデルからの出力を比較し、一致しているか？
  - 一致していない場合は設計意図に反したロジックとなっていると言える。

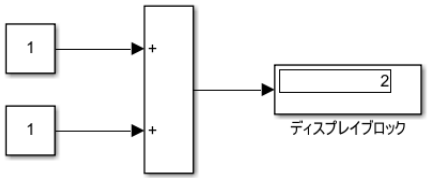
- 主にやること
  - 入力値とそれに対する出力期待値をセットにしてテストデータを作成する
  - テストデータをシミュレーション環境に取り込む
  - 入力値をコントローラモデルのインプットポートにつなぐ
  - 期待値と出力値を比較できるようにする
    - Ex)出力期待値と出力値の差分をとる
    - Ex)出力期待値のグラフと出力値のグラフを重ねあわせる
    - Ex)期待値と出力値が異なっていた場合にシミュレーションを停止する
  - シミュレーションを行い、出力比較の結果を確認する



MILS

- 以下、すべてMILSです（1/2）
  - 下記は、モデルを**作りながら**どんな動きか確認するときや、意図しない動きが起きた時の**原因解析**時に行うことが多い

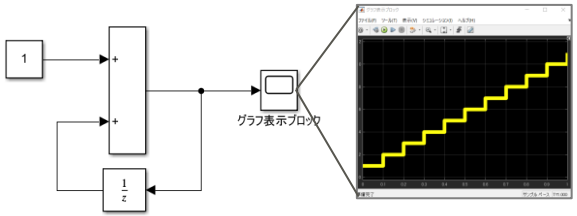
displayブロックで流れてるデータを確認



一番簡易的な方法
信号線にどんなデータが流れているか確認

上記モデルは1+1の結果を見てるため2が表示される

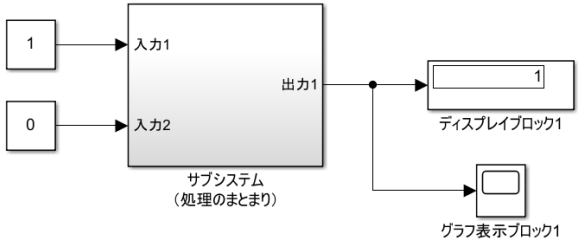
scopeブロックで流れてるデータの時間変化を確認



信号線に流れてるデータの時間変化を確認

上記モデルは1+前回値の結果を見てるため時間が
経過するにつれて1づつ大きくなる

処理のまとまりの動きを確認

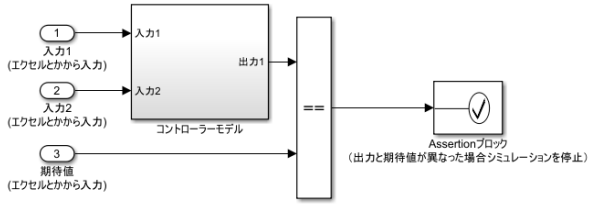


処理のまとまり(関数のようなイメージ)に
入力値を入れ、どのような動きになるか確認

MILS

- 以下、すべてMILSです（2/2）
  - 下記は、ある程度モデルが完成してから行うことが多い

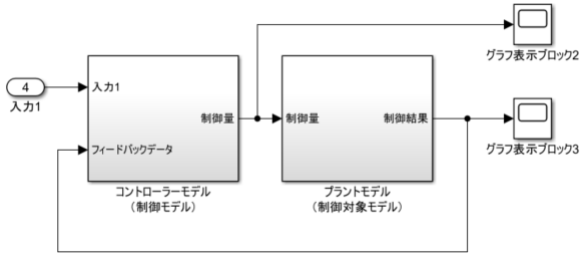
Assertionブロックで期待値との一致性を確認



モデルからの出力と期待値を比較し、
不一致であった場合シミュレーションを停止

入力や期待値は検証項目書からインポートしたり、
Signal Builderで作成したりする

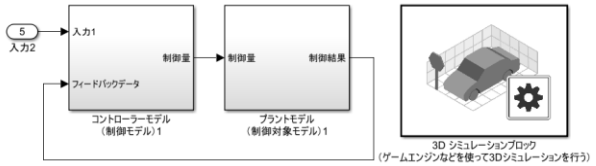
プラントモデルを用いたシミュレーション



制御対象をモデル化したプラントモデルを用い、
実装した場合にどのように動くかPC上で検証

プラントを使うことにより、実機を使わずにフィードバック
制御などの検証が可能

3Dエンジンを用いたシミュレーション



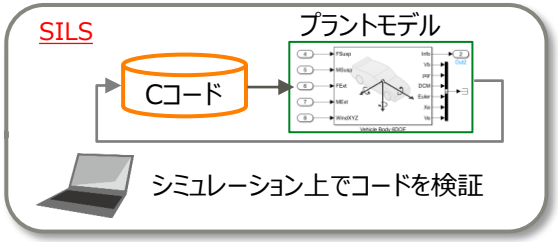
3Dエンジンを使用したシミュレーション

ゲームエンジンなどを使って、3Dワールド上にプラントを
配置しシミュレーションを行う。カメラデータなどをコントロ
ーラーにフィードバックし処理を行ったりすることが可能

SILS

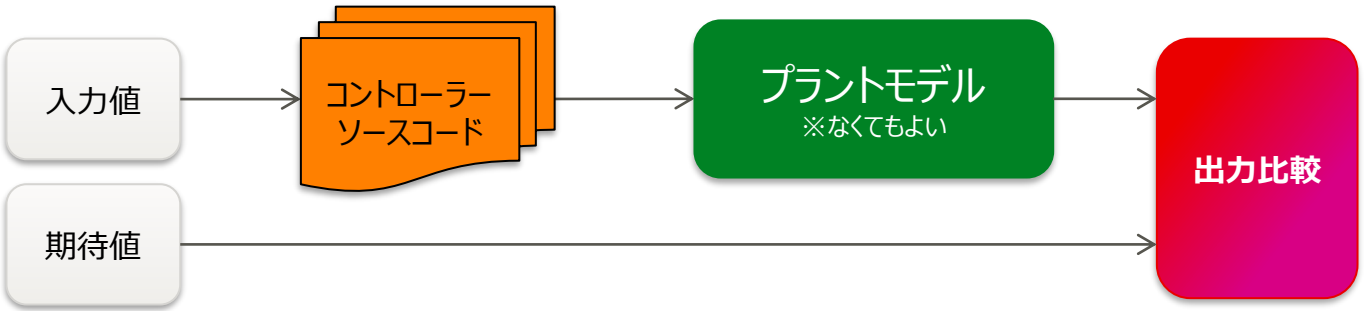
FUJITSU

Software In the Loop Simulation　：コントローラをモデルではなくC言語などのソフトウェアで実行



- 実行環境
  - MATLAB®/Simulink®モデル
- 目的
  - 生成コードとモデルの一致性を確認する
  - 固定小数点設計の誤りを検出/コード生成設定の誤りを検出、など
- 検証観点
  - モデル設計により期待される出力と実際のCコードからの出力を比較し、一致しているか？
  - 一致していない場合は設計意図に反した生成コードとなっていると言える。

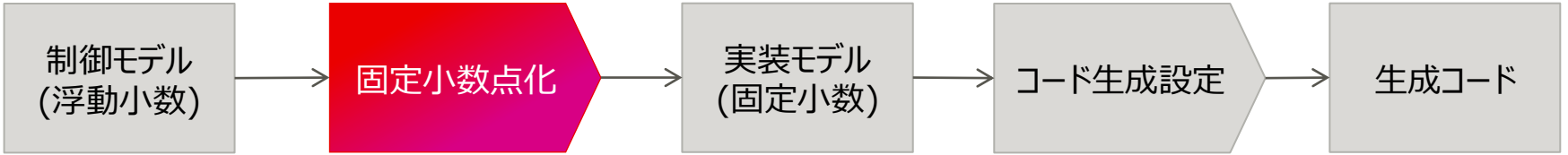
- 主にやること
  - ソースコード生成のためのコンフィグレーションパラメータを設定する
  - modelブロックを配置し、SILモードに設定する（ソースコードをS-Function化し、モデル内に取り込んでもよい）
  - ソースコードからの出力とMILSの結果を比較できるようにする
    - Ex)出力期待値と出力値の差分をとる
    - Ex)出力期待値のグラフと出力値のグラフを重ねあわせる
    - Ex)期待値と出力値が異なっていた場合にシミュレーションを停止する
  - シミュレーションを行い、出力比較の結果を確認する(ISO26262では、モデルとソースコードの一致確認が要求されている)

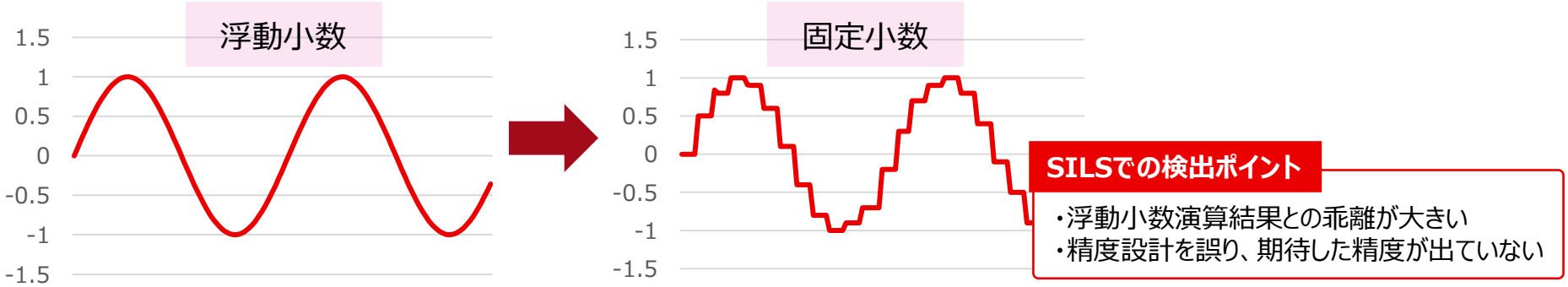


SILS

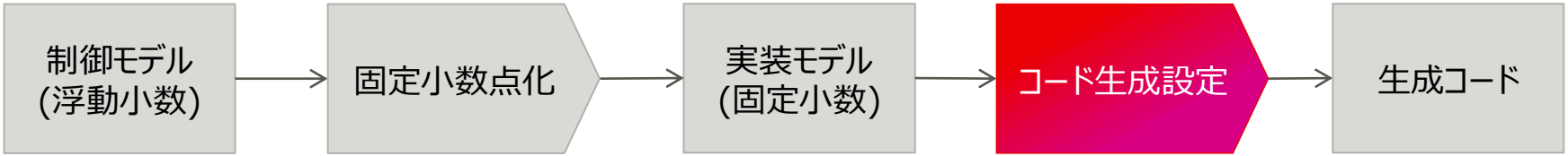
FUJITSU

- 固定小数点設計の誤りを検出
  - モデルのシミュレーション(MILS)を浮動小数点演算で実行するのに対して、コード生成にあたっては固定小数点演算に置き換えることが多い。限られたマイコン資源の中で処理効率化やメモリ量を削減するため。
  - 浮動小数点と固定小数点なのでMILSとSILSの結果の間には少なからず演算誤差は発生するが、仕様上許容できる誤差か否かを判断する必要がある。





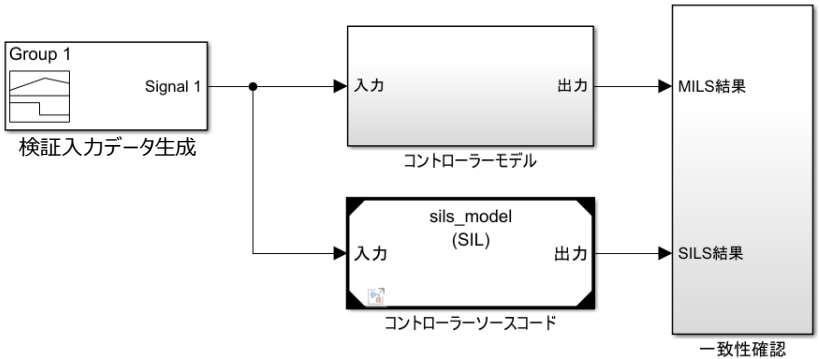
- コード生成設定の誤りを検出
  - コード生成にあたって、生成コードの最適化設定を決定する。
  - この設定次第ではMILS実行結果とSILS実行結果の間に差異が生じる可能性がある。
  - その差異を考察し許容するか設定を見直すか検討する。

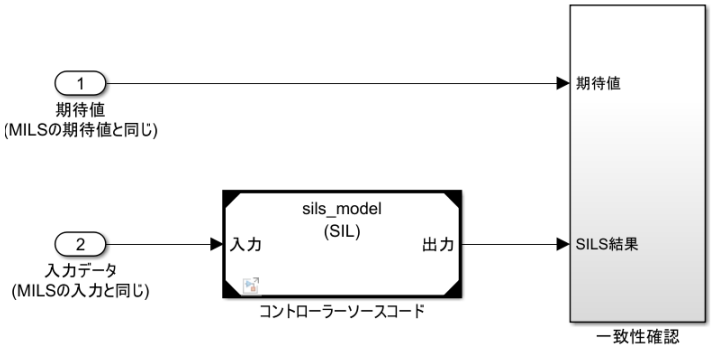


- 生成コードの挙動に影響しうる設定項目例

#	項目	概要
1	余分なローカル変数の削除 (式のたたみ込み表現)	コードの効率化のため、たたみ込み表現(単一式に統合)を有効にするか否か。デバッグには無効のほうが向いている。
2	ブロック削減	実行時間短縮のためブロックのグループを縮約・削除するか否か。
3	条件付き入力分岐実行	Switchブロック、MultiPort Switchブロックがモデルに含まれている場合のモデルの実行を高速化するか否か。
4	ベクトルの割り当てに対してmemcpyを使用	for ループを memcpy 関数と置き換えることで最適化を図るオプション。

- SILSの具体的なイメージは以下です
  - 基本的には、MILSと同じデータを使い、一致性を確認する



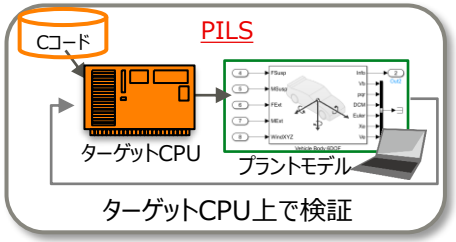


わざわざ、モデルのシミュレーション実行はせず、
MILS実行時のデータを流してもよい

PILS

FUJITSU

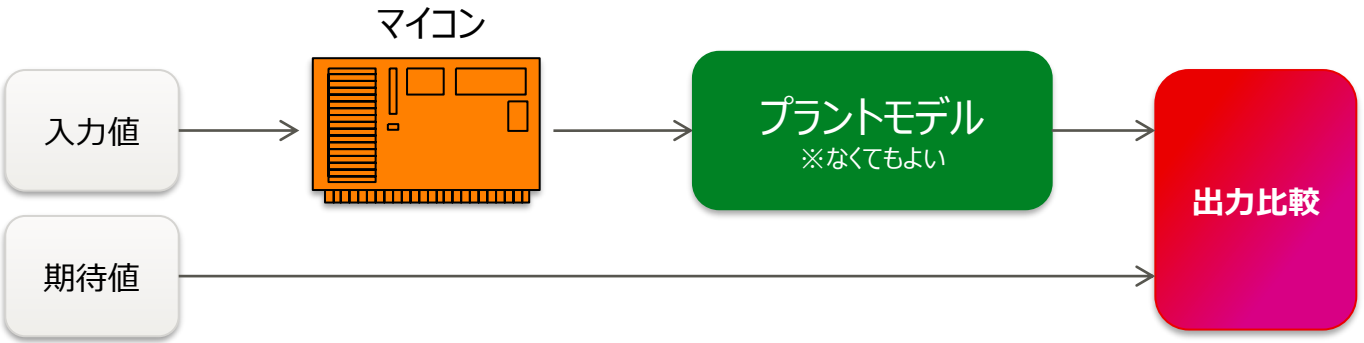
Processor In the Loop Simulation　：マイコン上で実行



- 実行環境
  - マイコン/マイコンシミュレータ
- 目的
  - マイコン上での実行コードがモデルと一致することを検証する
  - コンパイラ,ビルド環境の誤りを検出/処理系違いにより不一致の検出、など
- 検証観点
  - モデル設計により期待される出力と実際のマイコンの演算結果を比較し、一致しているか？
  - 一致していない場合は設計意図に反した実行コードとなっていると言える。

PILS

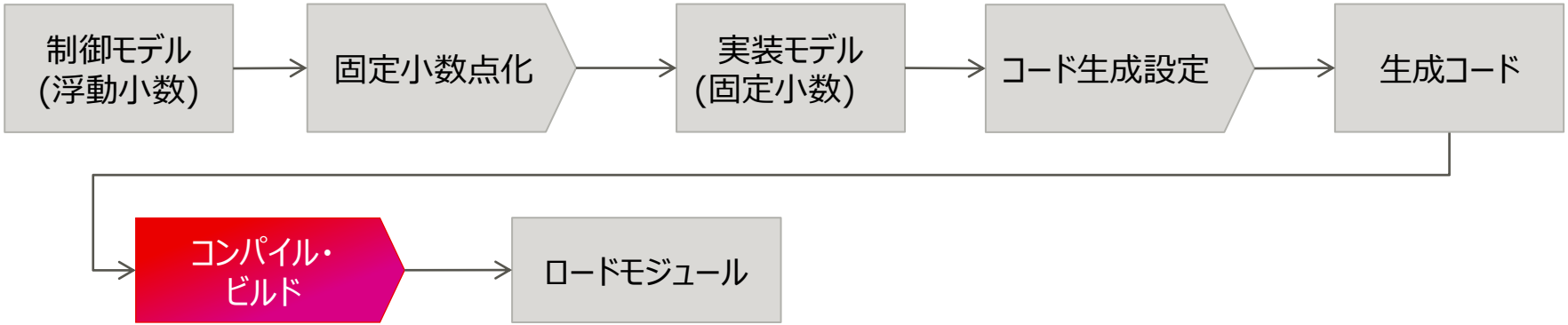
- 主にやること
  - コード生成されたコントローラを、マイコン上で実行可能な形式にビルドする
  - マイコンとシミュレーション環境を接続する
  - テストデータを入力し、出力比較結果を確認する



PILS

FUJITSU

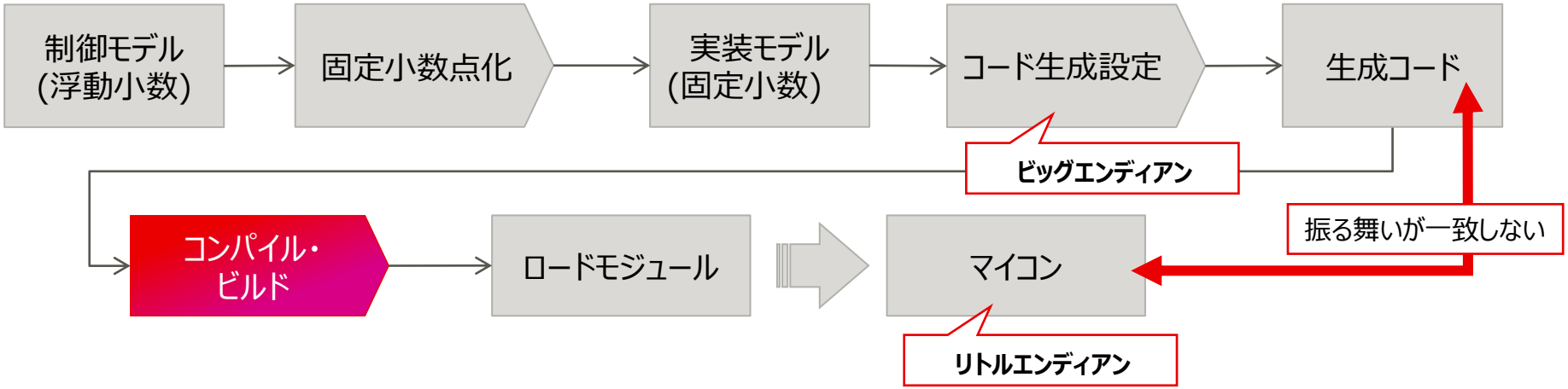
- コンパイラ、ビルド環境の誤りを検出
  - 生成コード(C言語)をコンパイル・ビルドしていく過程で意図しない差異が生じる可能性がある。
    - コンパイラの最適化設定による動作の差異



PILSでの検出ポイント

- 最適化設定により、モデルで意図した機械語に変換されていない

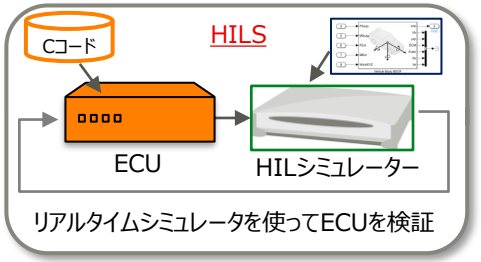
- 処理系違いによる不一致の検出
  - ANSI-Cで未規定の動作をC言語で記述された場合、ふるまいは処理系に依存する。
  - コード生成時の設定と、実際のマイコン仕様のエンディアンが異なっている場合、振る舞いが一致しない(バウンダリ等も同様)



HILS

FUJITSU

Hardware In the Loop Simulation　：コントローラーのみ実機上で実行

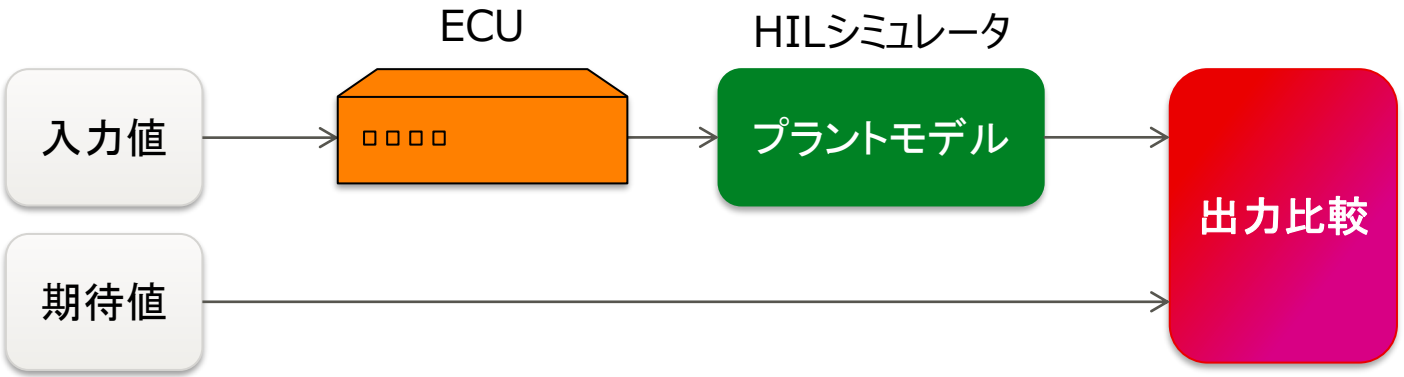


- 実行環境
  - ECU
  - プラントも実物に近い環境で動作させるため、専用のエミュレータにプラントモデルをロードして実行
- 目的
  - ECUの出力がモデルと一致することを検証する
  - 実機上での応答性の検証、など
- 検証観点
  - モデル設計により期待される出力とECUの出力を比較し、一致しているか？
  - 一致していない場合は設計意図に反したECUとなっていると言える。

HILS

FUJITSU

- 主にやること
  - コントローラを実機にロードする
  - プラントモデルをHILシミュレータにロードする
  - 実機とHILシミュレータを接続する
  - テストデータを入力し、出力比較結果を確認する



HILS

FUJITSU

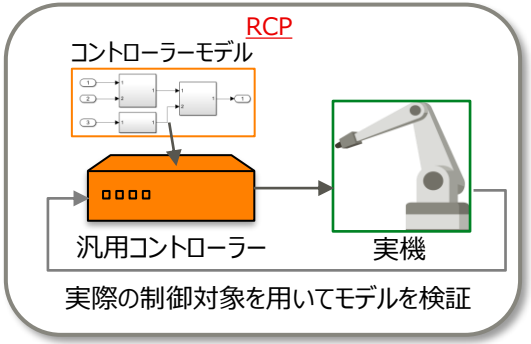
- リアルタイムなECUの応答性を検証
  - 期待した応答性を発揮できてるかどうかの検証はECUに実装されないと検証できない

- 非MBDの開発でもHILSが行われる場合がある。
  （実機の完成前や、実機を使用しないでECUを評価したい場合など）

FUJITSU

Rapid Control Prototyping：プラントに実機を用い、コントローラーはモデルをのせて実行

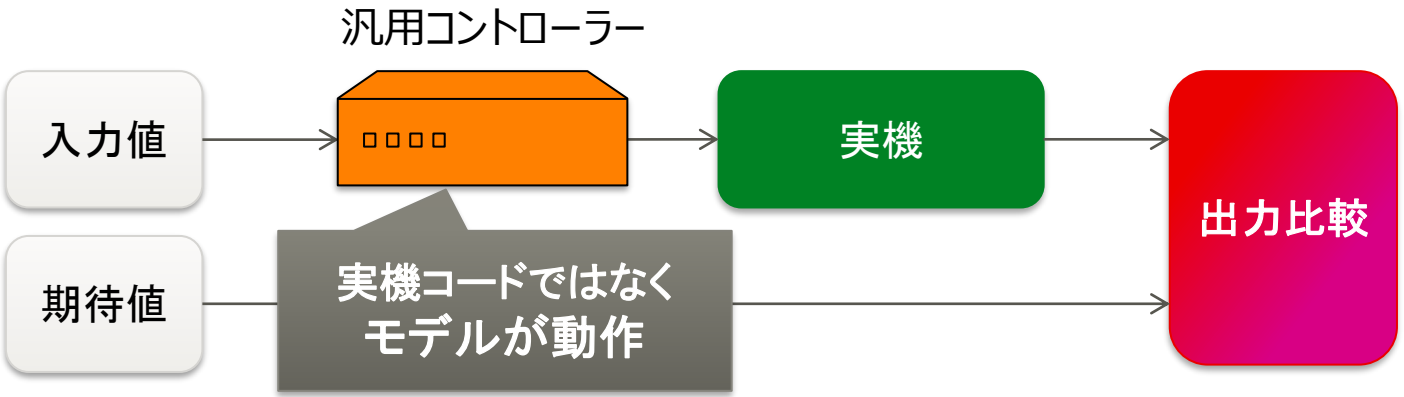
コントローラーは一部もしくは全部（一部：バイパス手法　全部：フルパス手法）



- 実行環境
  - 汎用コントローラー
  - プラントは実機
- 目的
  - 実機上でのロジックの動作を早期に評価する(コード生成～実機組み込みを行う前)
- 検証観点
  - 設計意図どおりに実機が動作しているか？
  - 意図通りでない場合はモデルの見直しやハードウェア見直しの必要があると言える。

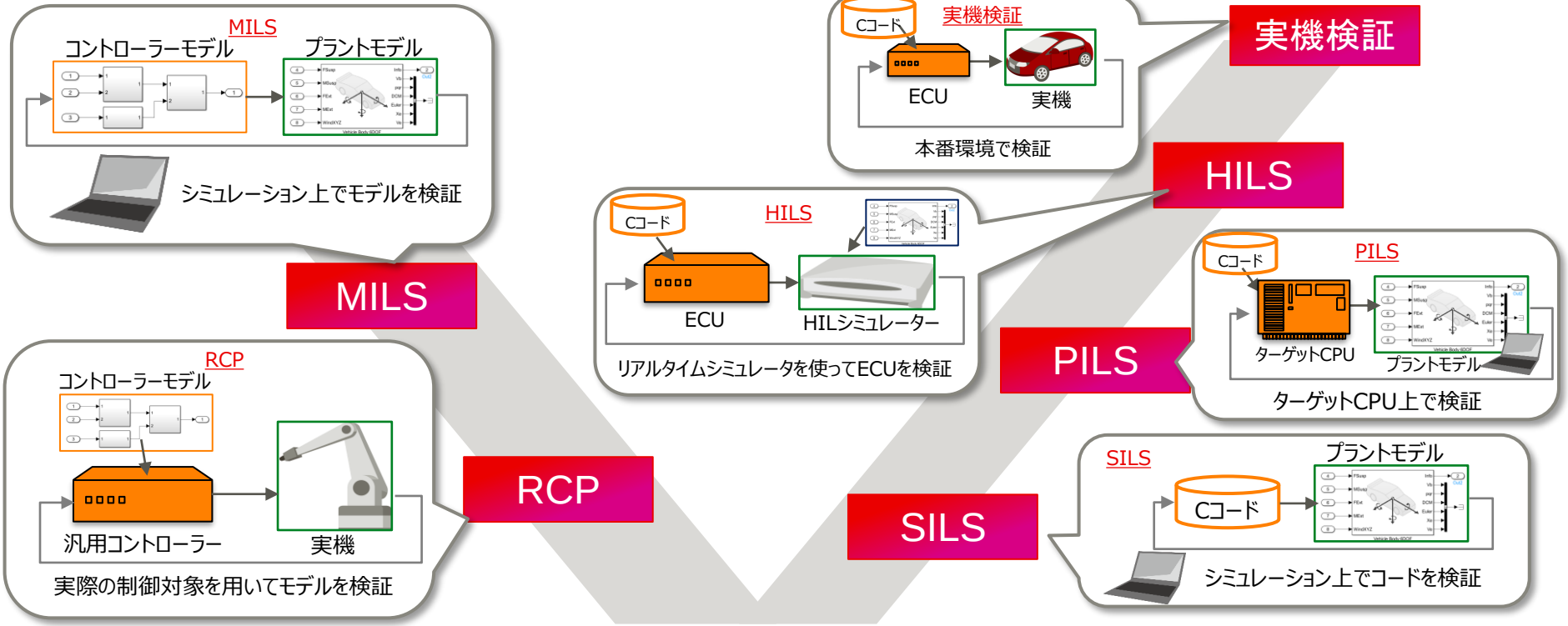
RCP

- 主にやること
  - コントローラとプラント（実機）を接続する
  - テストデータを入力し、出力を測定し、期待値との一致を確認する



MBDにおける検証技法(再掲)

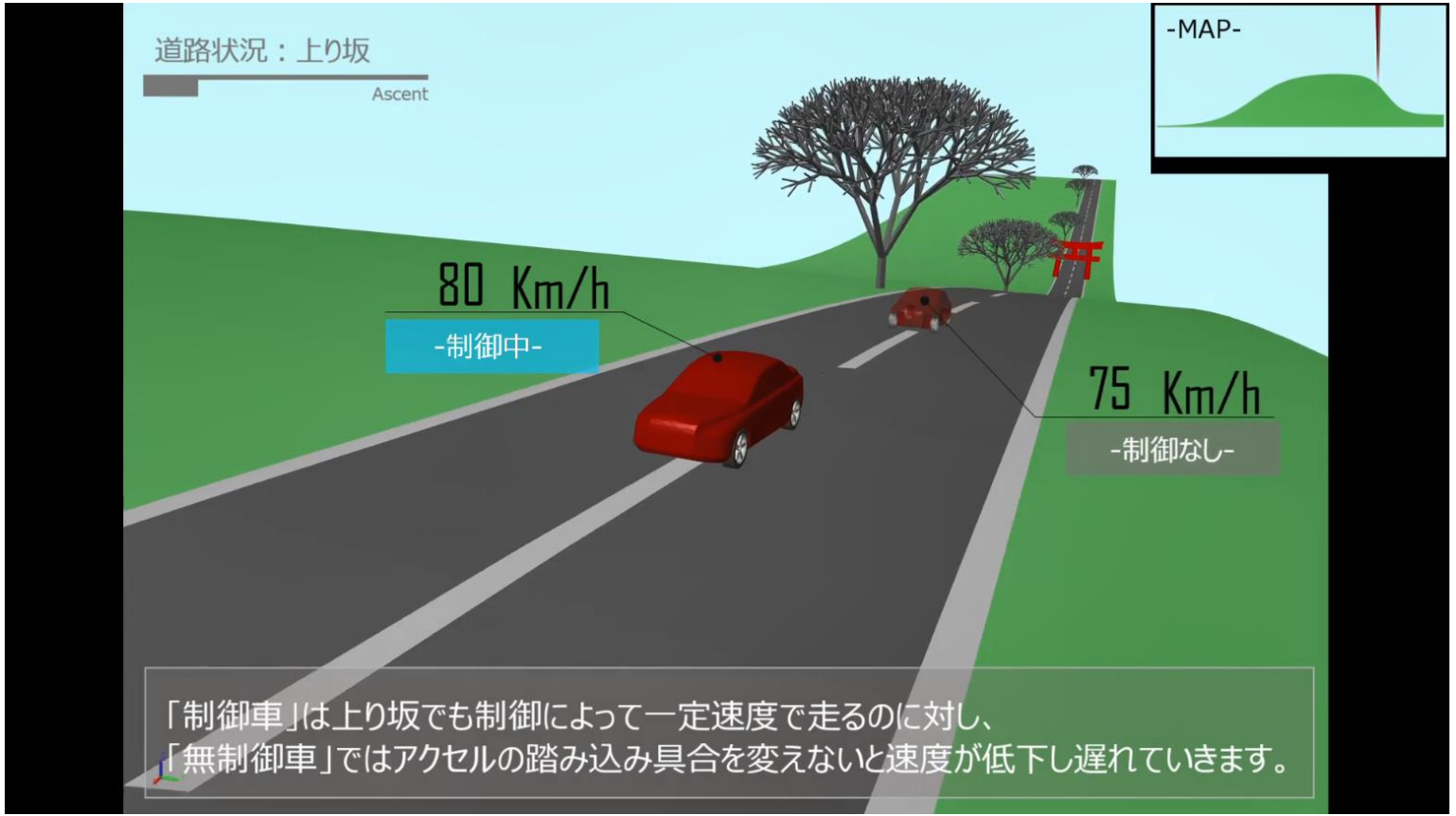
- MBDにおける検証技法は、MILS,RCP,PILS,HILSなどがある(すべて実施するわけではなく、PJによって取捨選択する)
- 基本的にXILS(X In the Loop Simulation)と名称がつけられており、X部はシミュレーションのレベルによって変わる



FUJITSU

MBDデモ

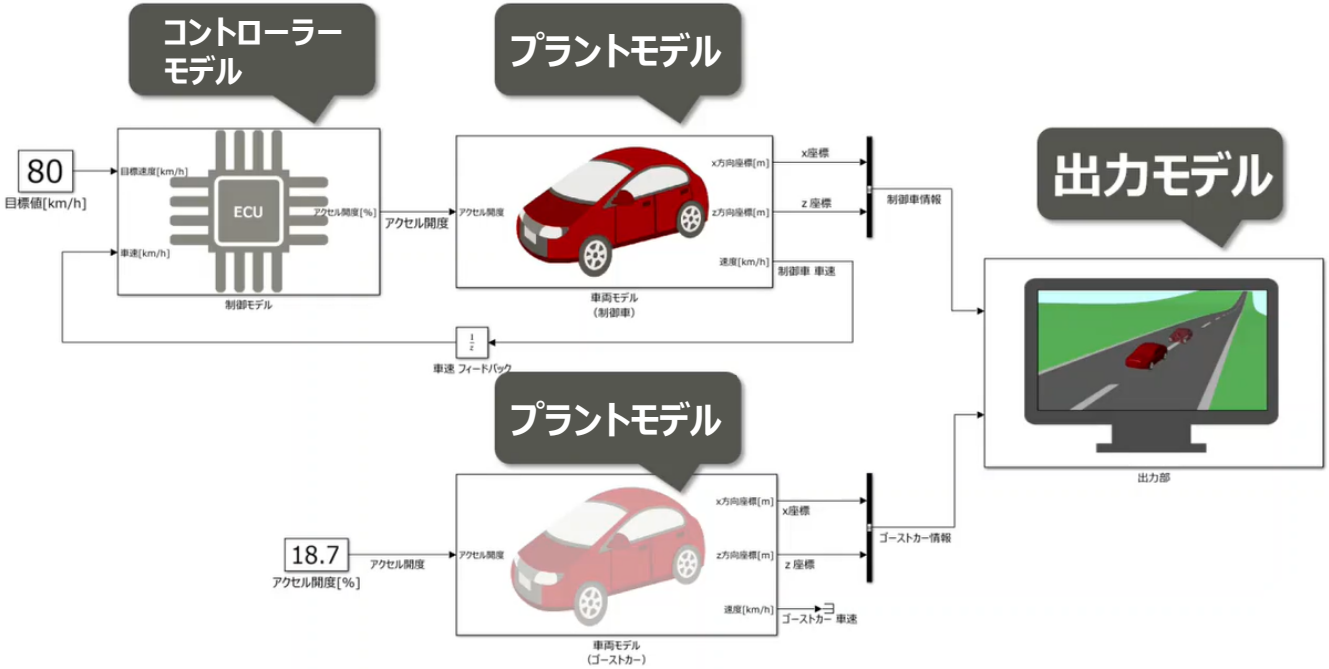
MBDデモ①（クルーズコントロールのMILS）



MBDデモ①（クルーズコントロールのMILS）

FUJITSU

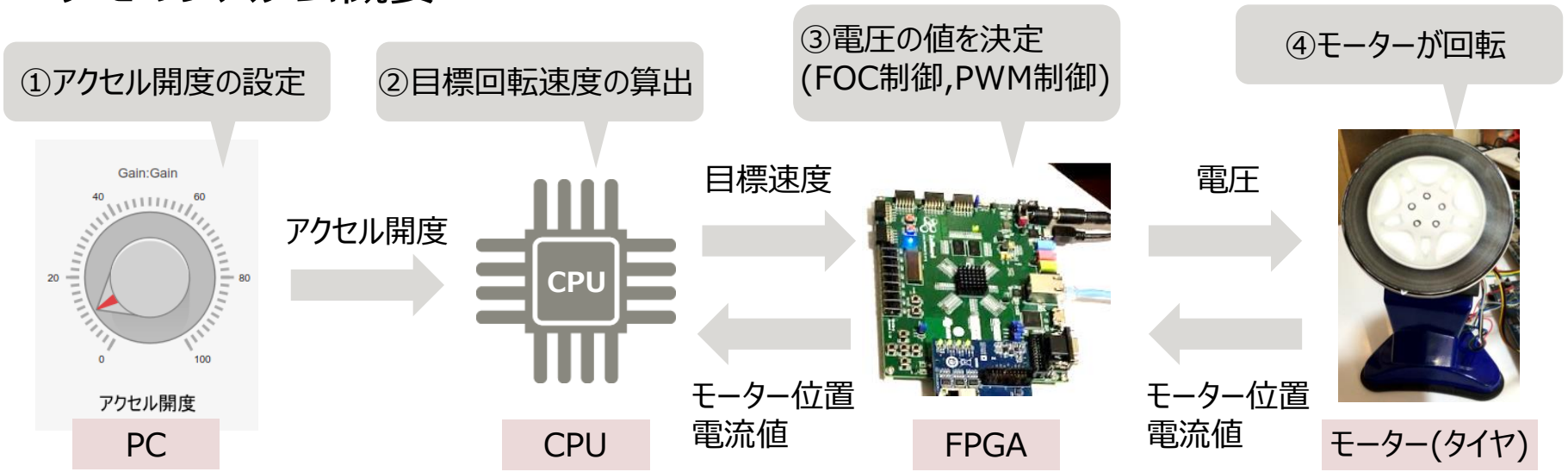
○プラントモデルを使ったMILSを行うと、デモのようにコントローラーが実機上でどのような動きをするか検証できる



MBDデモ② (FPGA実装デモ)

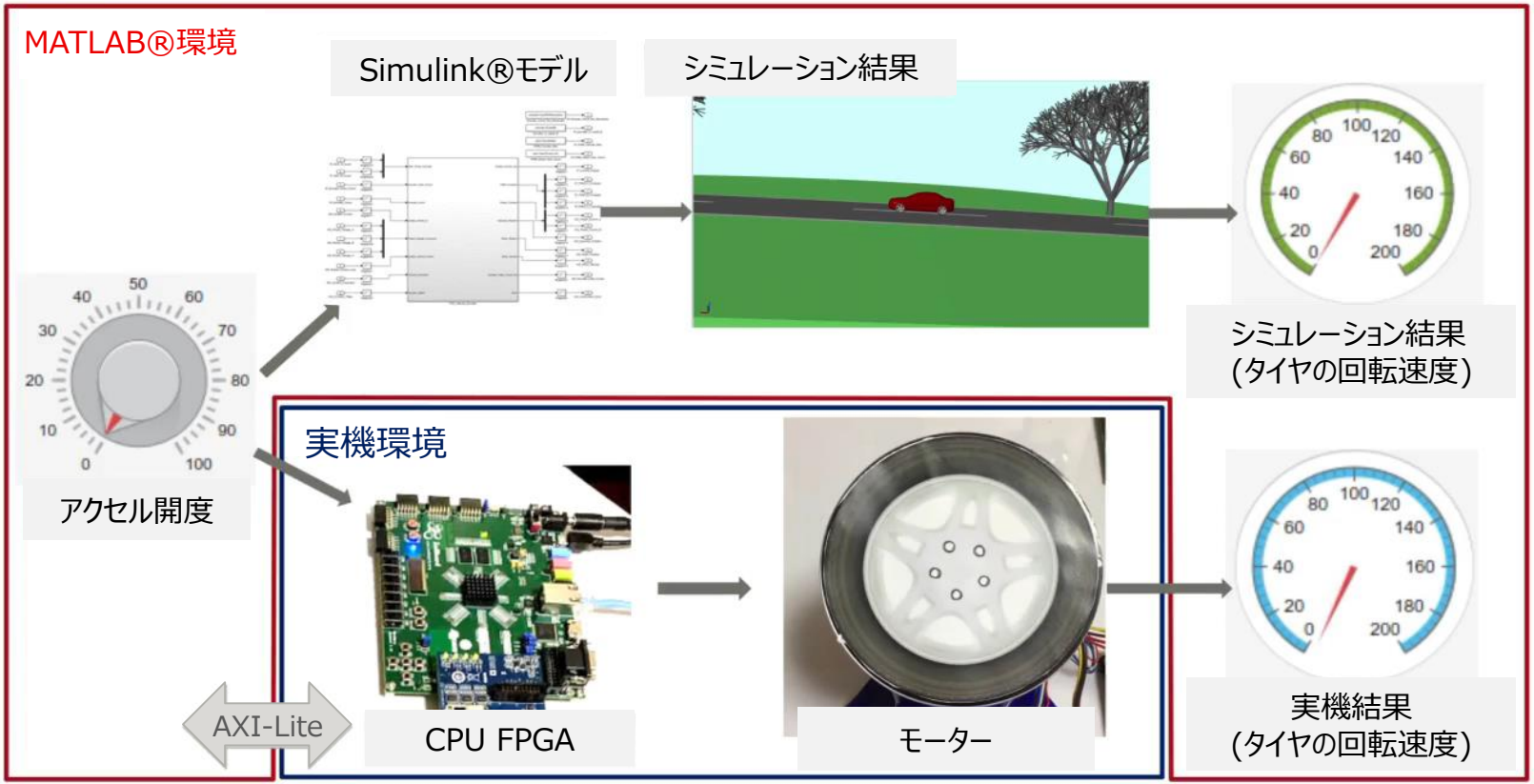
FUJITSU

- デモ内容
  - アクセル開度から、タイヤの回転速度を制御するモデル(モーター制御)をFPGAに実装し検証を行う
- デモのシステム概要



MBDデモ② (FPGA実装デモ)

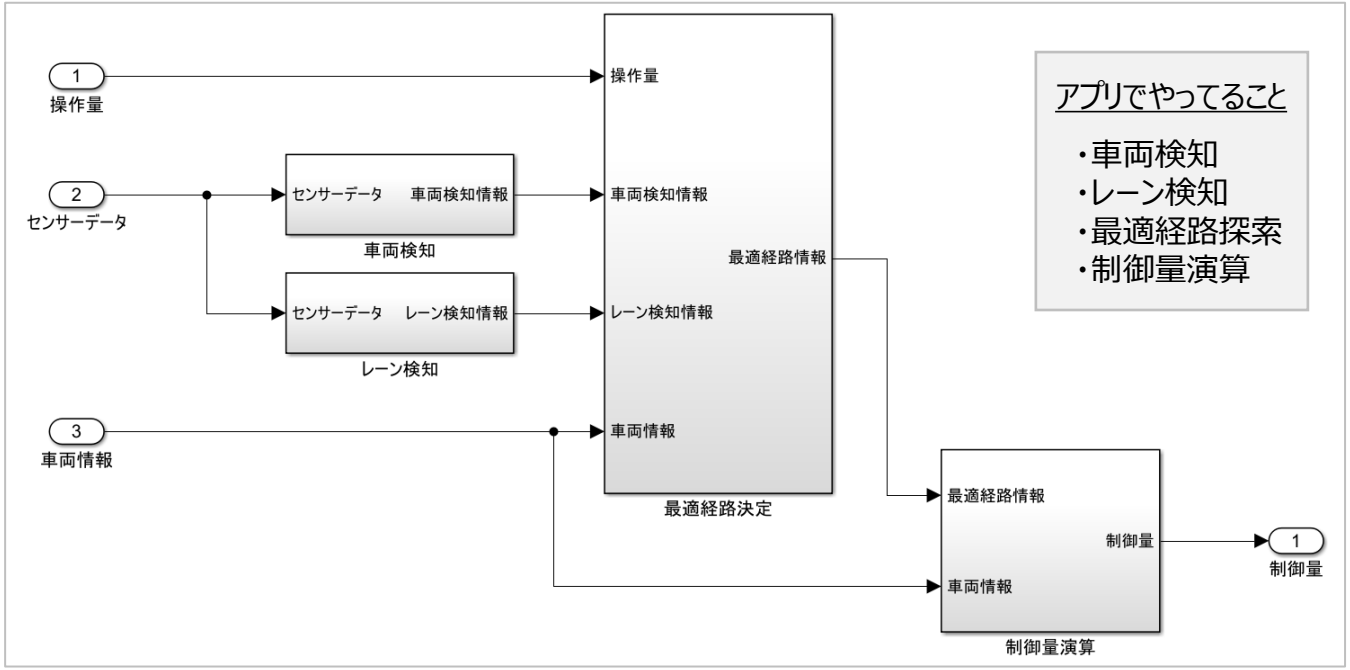
FUJITSU



MBDデモ③（3Dシミュレーションを使ったMILS）

FUJITSU

○以下のような自動運転アプリケーションのモデルを作成　→　検証したい



MBDデモ③（3Dシミュレーションを使ったMILS）

FUJITSU

○車両シミュレーション環境を構築　→　実車レス検証

MBDデモ③（3Dシミュレーションを使ったMILS）

FUJITSU

○実際のシミュレーション動画

MBD教育サービスについて

MBD教育サービスについて

FUJITSU

- 以下のようなMBDの教育サービスを提供しています。
  - MBD実践編
    - ブロックを知る　　：よく使われるブロックを紹介します
    - モデルを読めるようになる　　：モデルの読み方のコツなどを紹介します
    - モデルを作成できるようになる：MBDツールの操作方法や、きれいなモデルを作るためのコツや、練習問題を用意しています。
  - Stateflow®研修
  - MBDワークショップ

- 下記サイト下部のフォームからお気軽に問い合わせください。
  https://www.fujitsu.com/jp/group/bsc/services/embedded/automotive/model-based-development/

Thank you