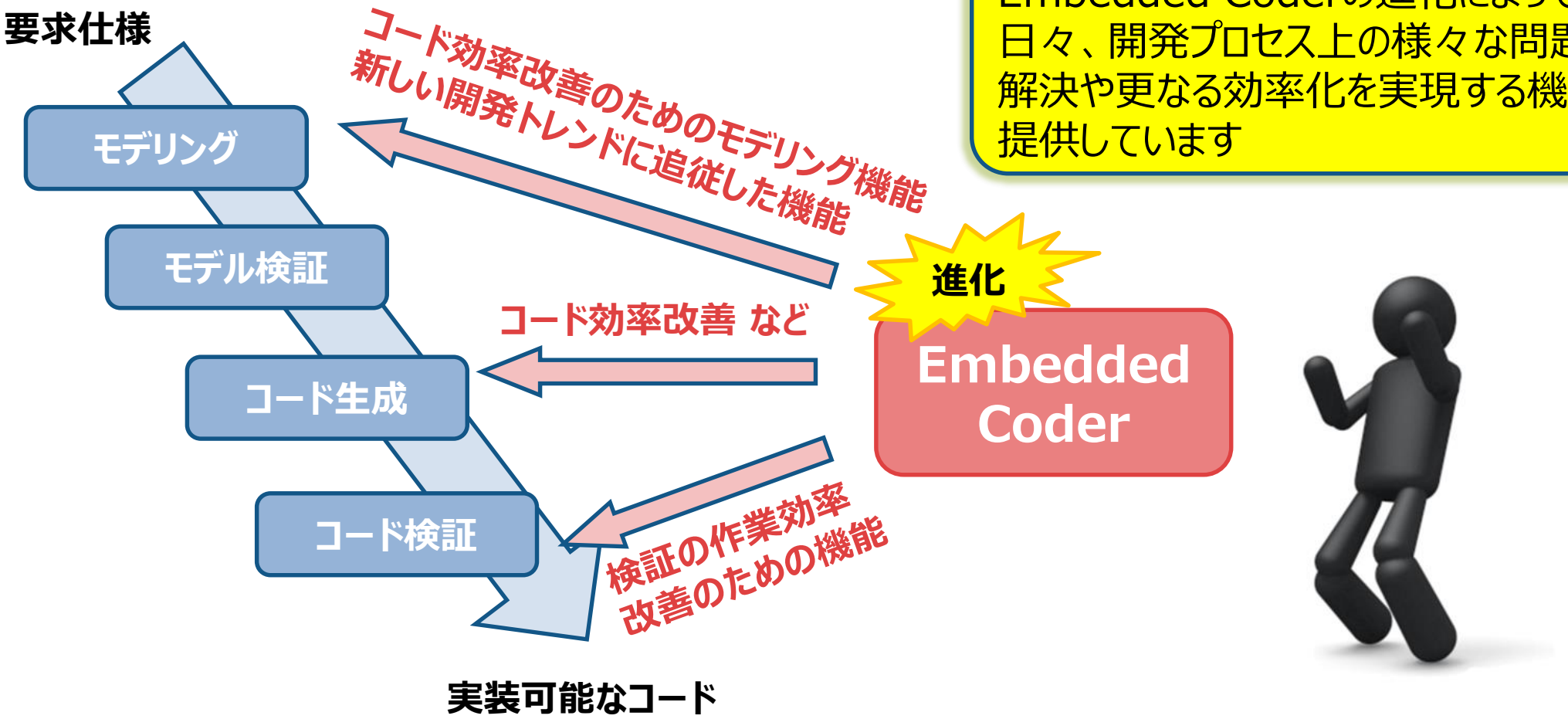
ユースケースにもとづく量産コード生成技術の進化

MathWorks Japan
パイロットエンジニアリング部
シニアパイロットエンジニア
庄子 晃太郎

本発表のスコープ

Embedded Coder™の進化について紹介します



Embedded Coderの進化によって、
日々、開発プロセス上の様々な問題の
解決や更なる効率化を実現する機能を
提供しています

アジェンダ

1. Embedded Coderとは？
2. 自動コード生成ツール適用のメリット
3. Embedded Coderの進化
   - コード生成エンジンの進化
   - 機能の進化
   - 運用フローの進化
4. まとめ

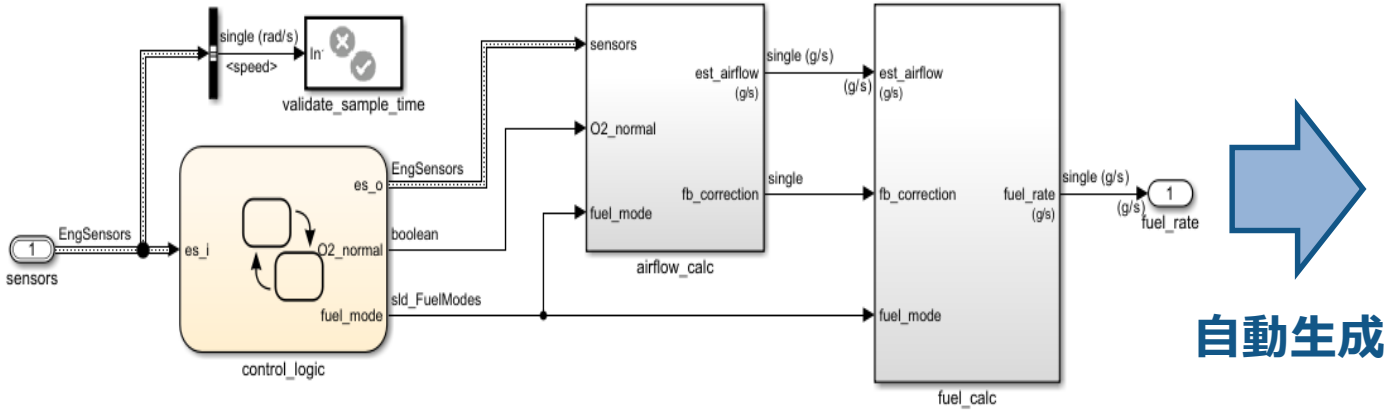
アジェンダ

1. **Embedded Coderとは？**
2. 自動コード生成ツール適用のメリット
3. Embedded Coderの進化
   - コード生成エンジンの進化
   - 機能の進化
   - 運用フローの進化
4. まとめ

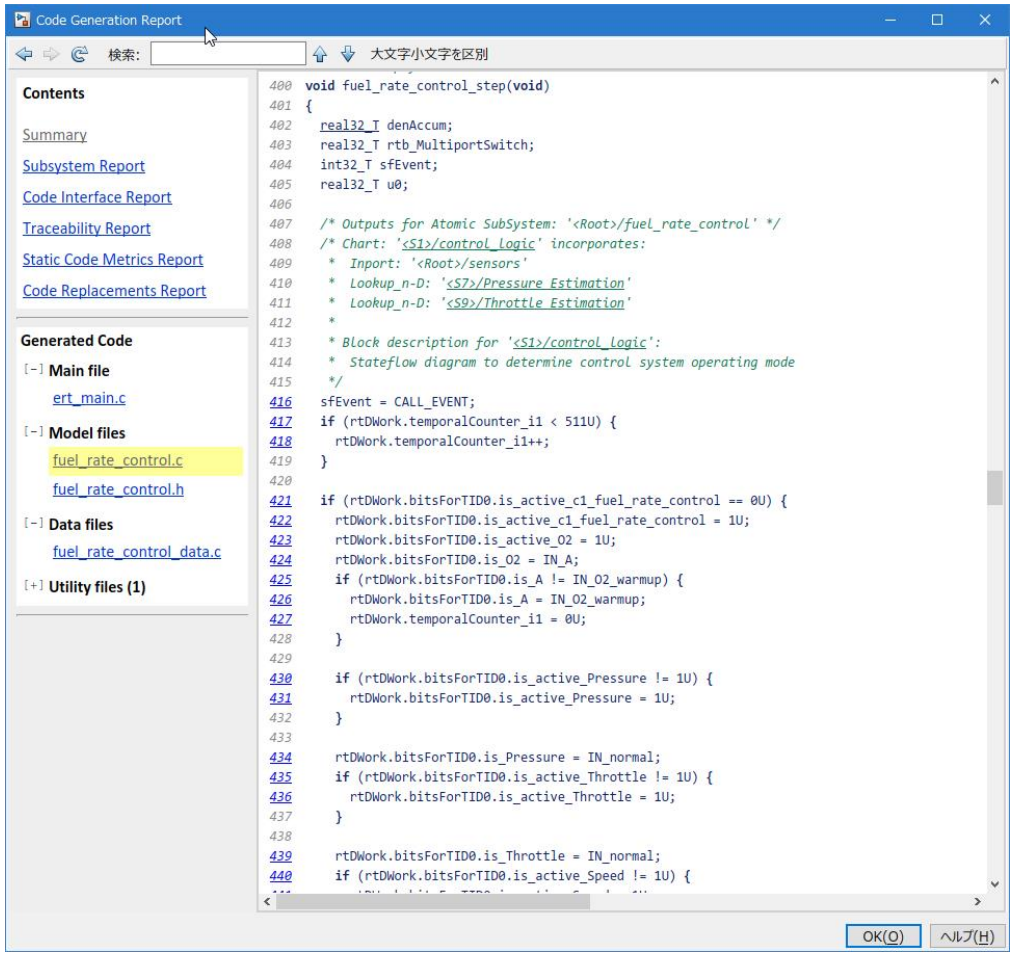
Embedded Coderとは？

Simulink®/Stateflow®で作成されたモデルから、組み込み向けコードを自動生成することができる純正ツールです

- **ボタン1発**でモデルから量産コードが生成できます



自動生成



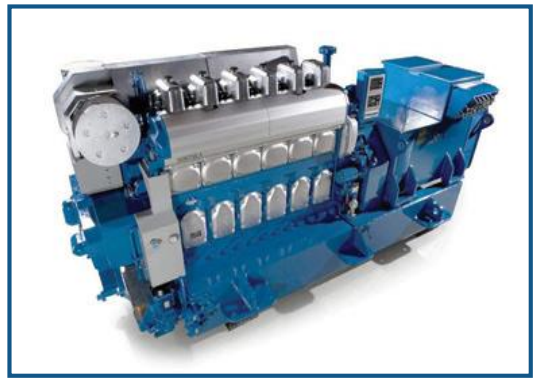
具体的なEmbedded Coderの量産適用事例

DaimlerChrysler
クルーズコントロール制御

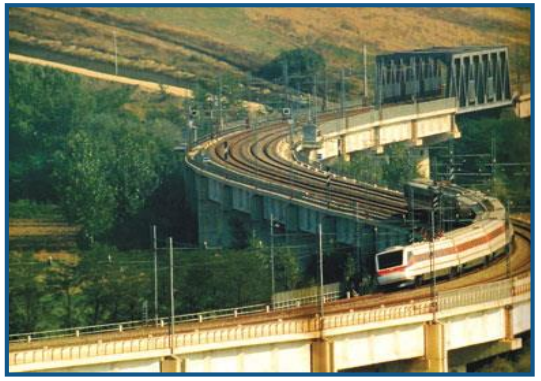
ミツバ
リバーシングワイパー制御

Airbus
空調制御

General Motors
ハイブリッド制御



Wärtsilä
産業用ディーゼルエンジン

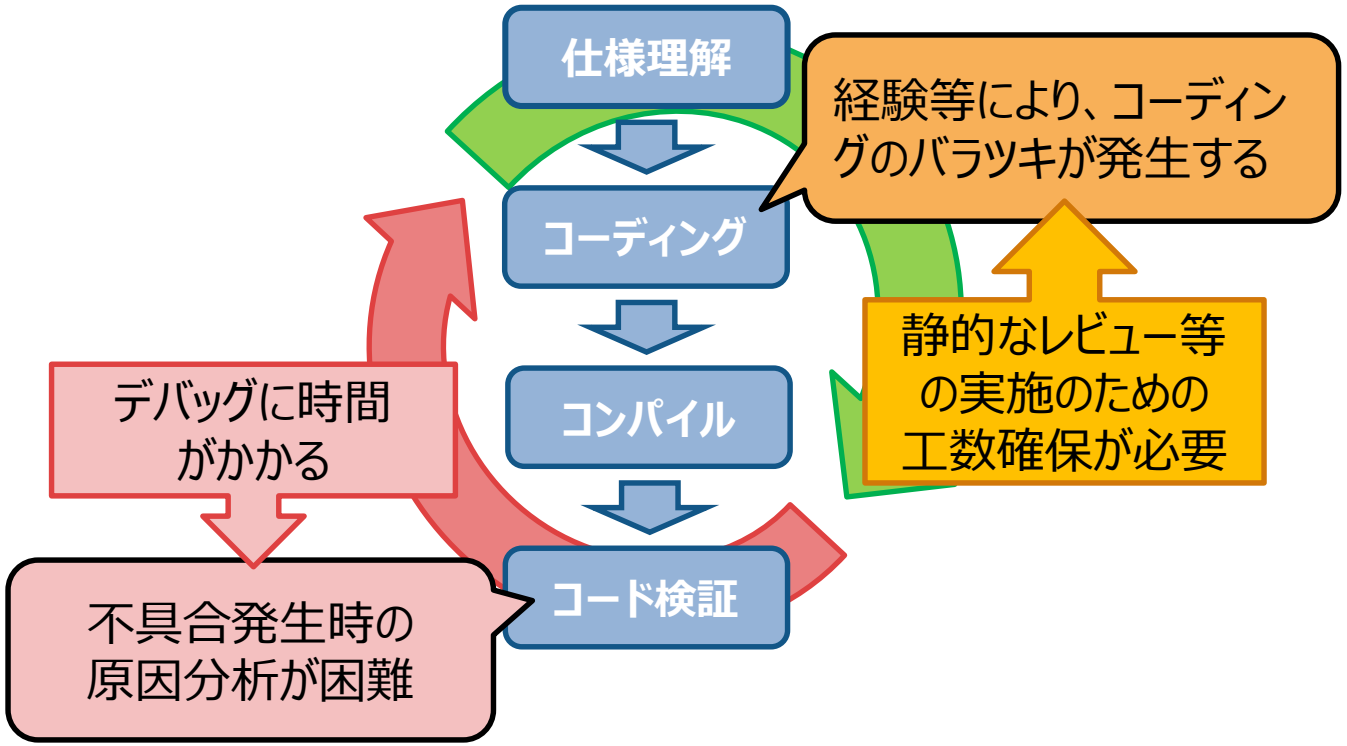


Alstom
鉄道用電力変換システム

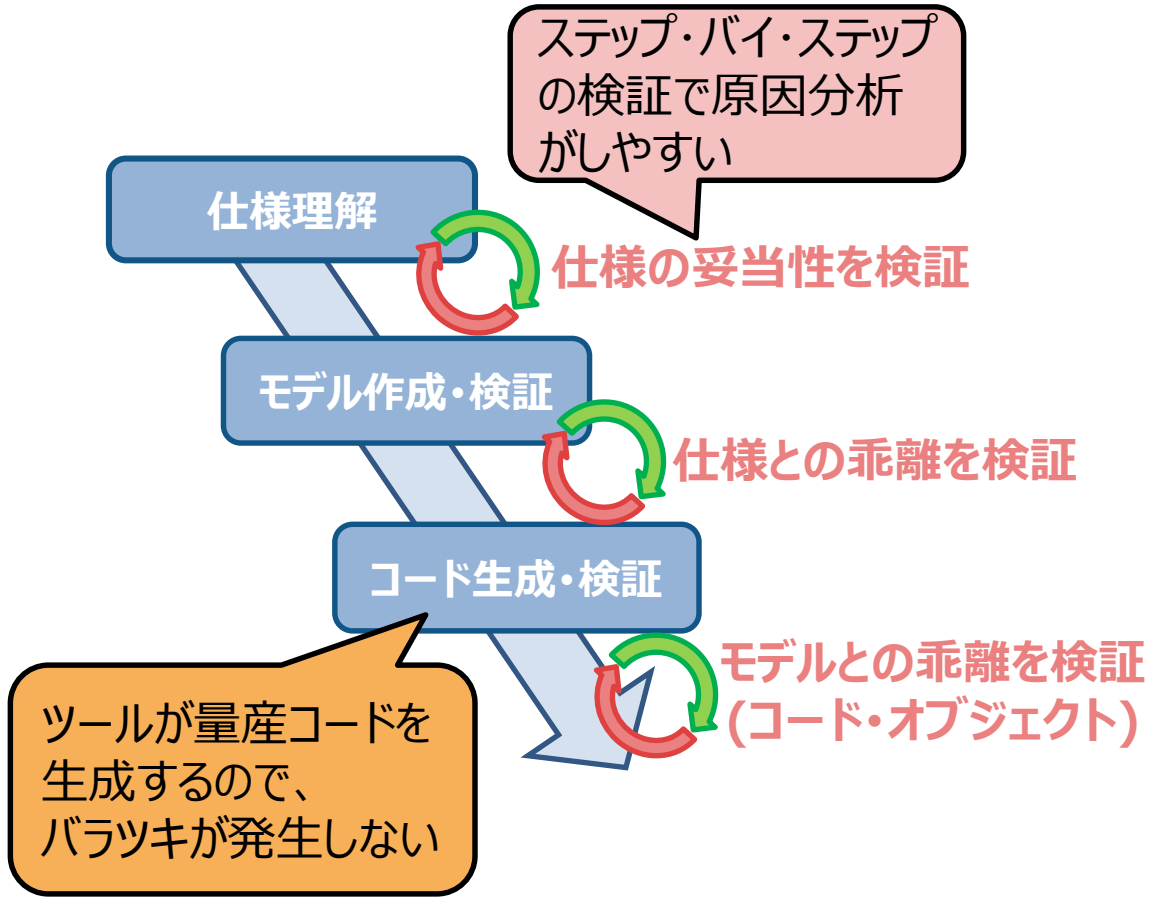
アジェンダ

1. Embedded Coderとは？
2. **自動コード生成ツール適用のメリット**
3. Embedded Coderの進化
   - コード生成エンジンの進化
   - 機能の進化
   - 運用フローの進化
4. まとめ

自動コード生成ツールの適用による工数短縮と高品質の両立



ハンドコードによる従来開発



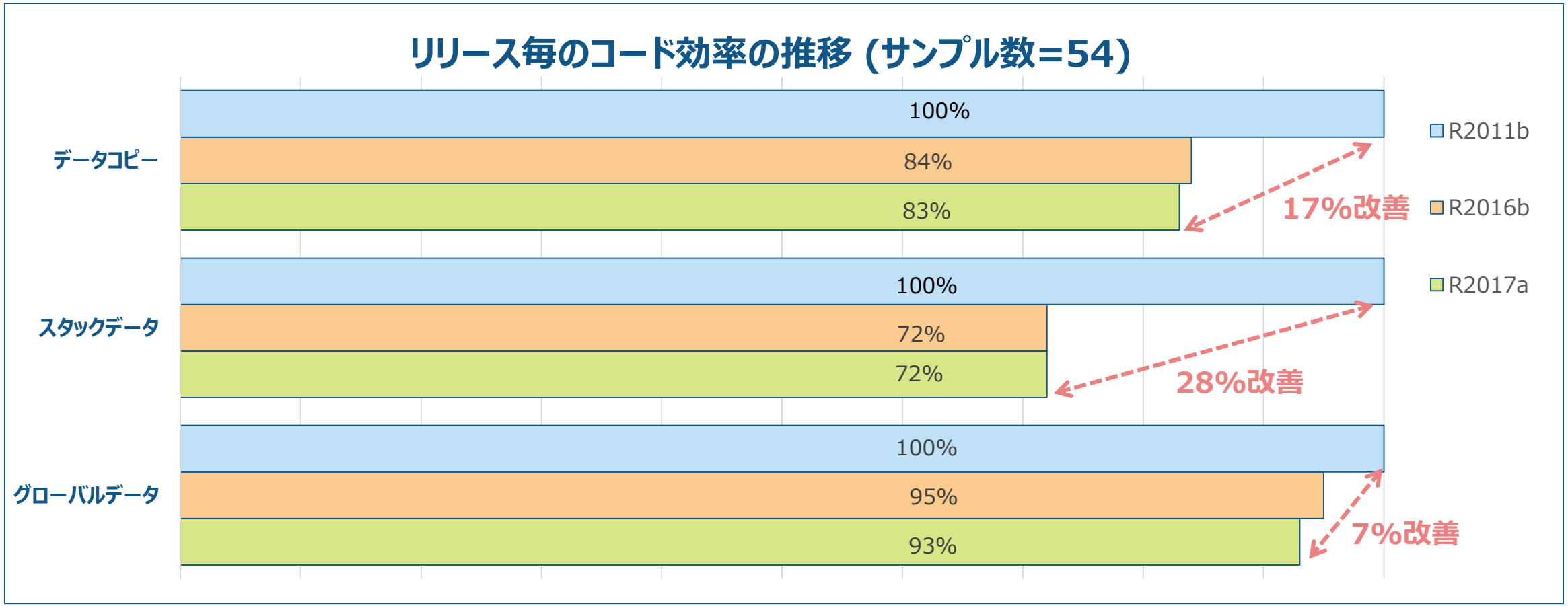
自動コード生成ツールによる開発

アジェンダ

1. Embedded Coderとは？
2. 自動コード生成ツール適用のメリット
3. **Embedded Coderの進化**
   - **コード生成エンジンの進化**
   - 機能の進化
   - 運用フローの進化
4. まとめ

Embedded Coder生成コードの進化

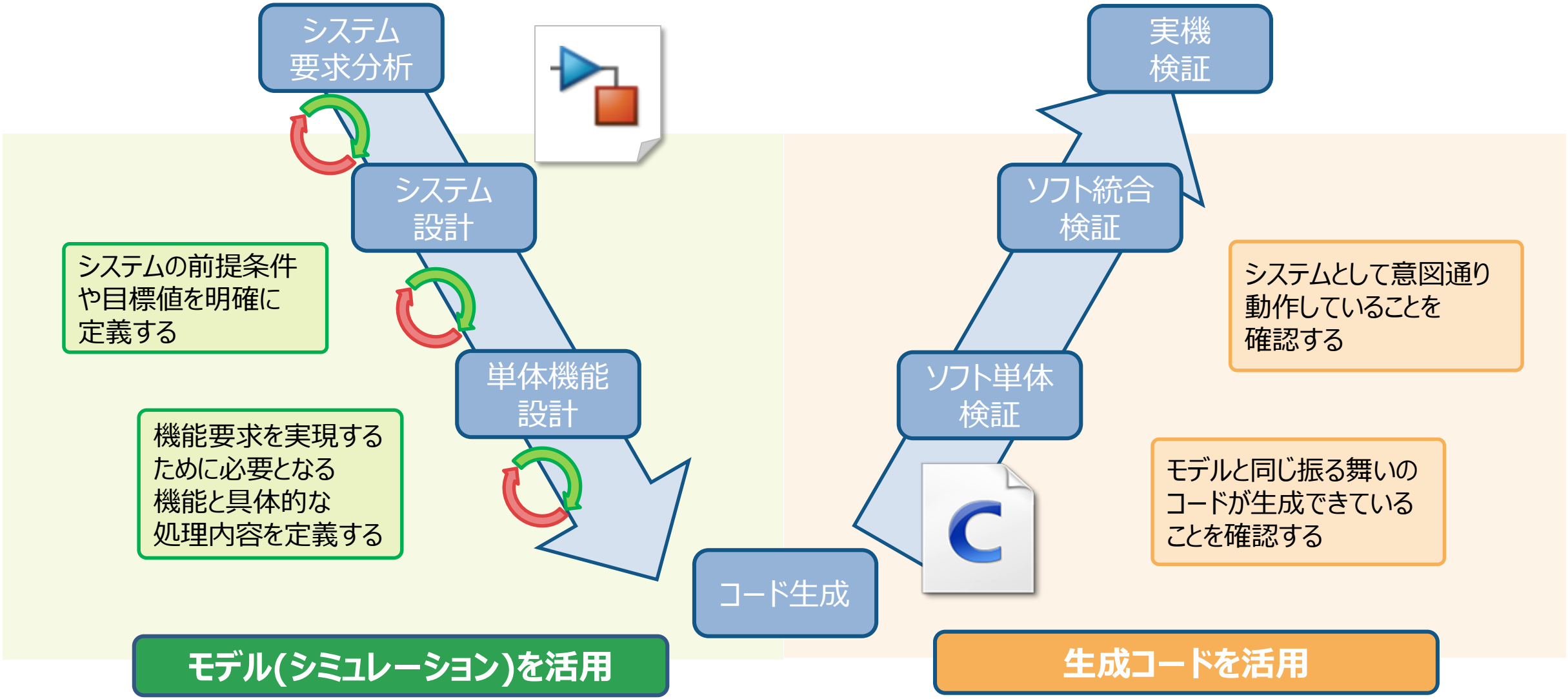
日々改善活動をおこない、バージョンアップの度に更なるコード効率向上を遂げています



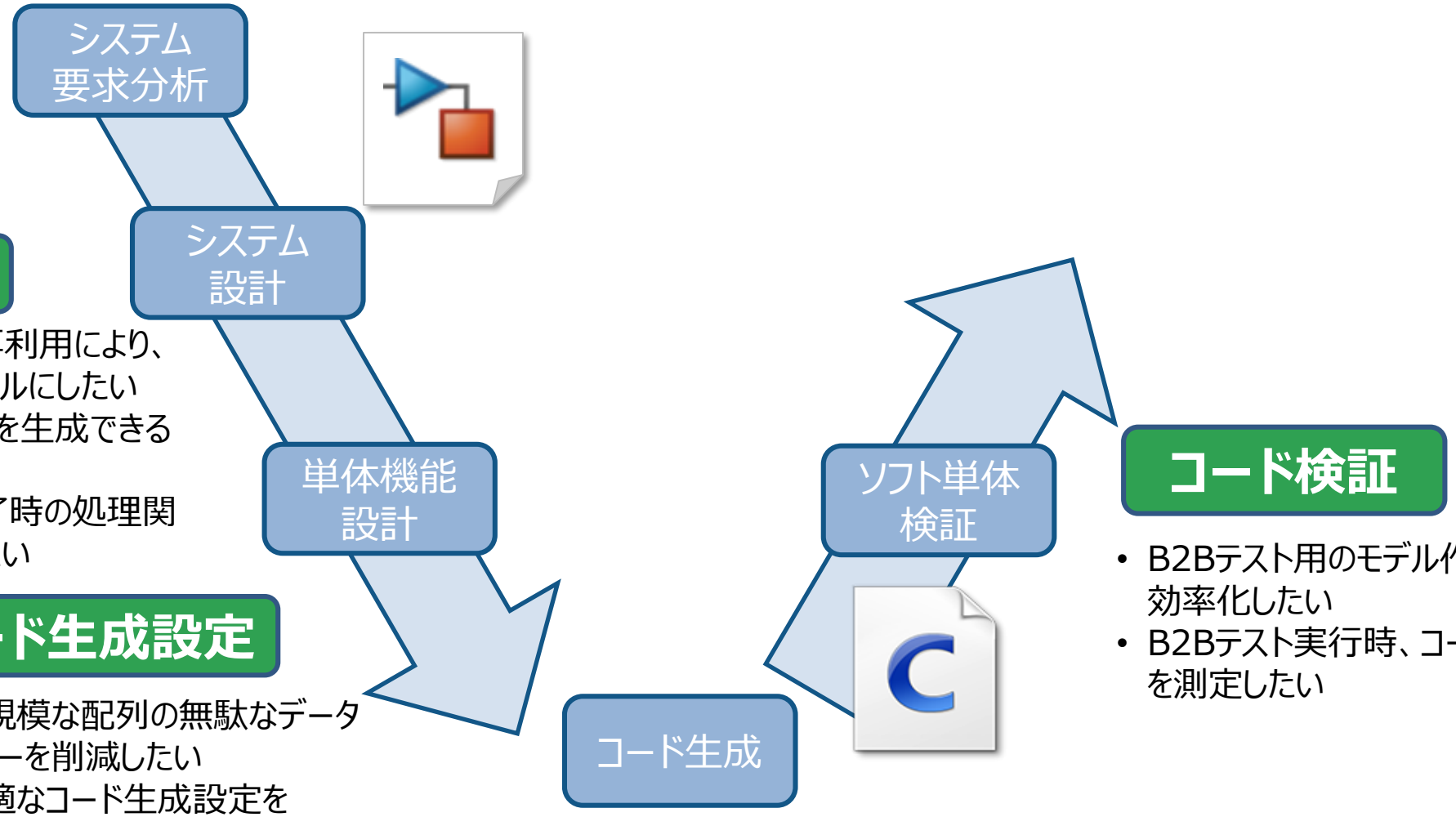
アジェンダ

1. Embedded Coderとは？
2. 自動コード生成ツール適用のメリット
3. **Embedded Coderの進化**
   - コード生成エンジンの進化
   - **機能の進化**
   - 運用フローの進化
4. まとめ

量産 / 組み込み実装向け開発プロセスの流れ



量産コード開発プロセスのよくあるユースケース



モデリング

- 類似サブシステムの再利用により、コード効率の良いモデルにしたい
- MISRA準拠のコードを生成できるモデルを作りたい
- 初期化・リセット・終了時の処理関数をモデルで表現したい

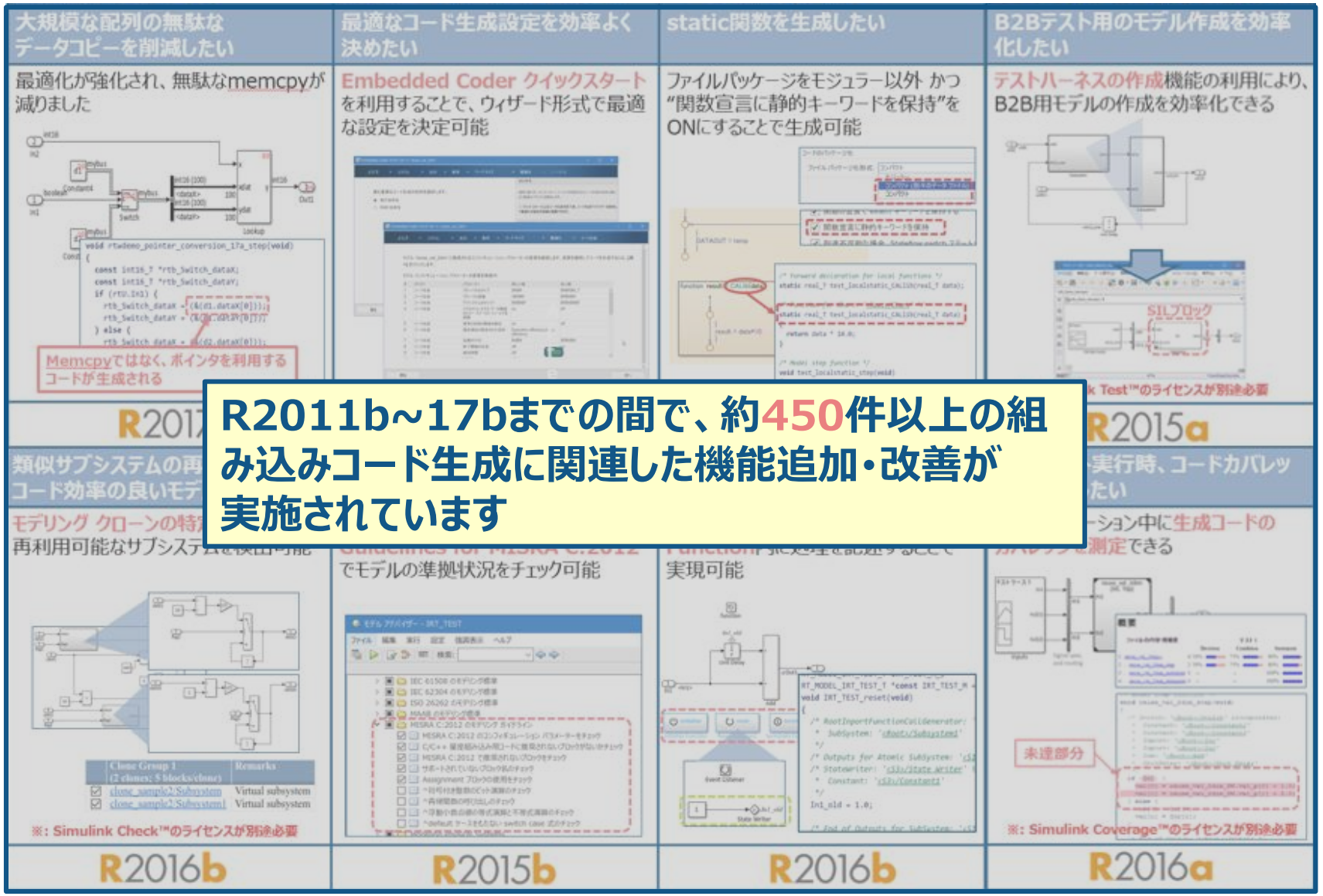
コード生成設定

- 大規模な配列の無駄なデータコピーを削減したい
- 最適なコード生成設定を効率よく決めたい
- static関数を生成したい

コード検証

- B2Bテスト用のモデル作成を効率化したい
- B2Bテスト実行時、コードカバレッジを測定したい

ユースケースに対するEmbedded Coderソリューション



ユースケースに対するEmbedded Coderソリューション

大規模な配列の無駄なデータコピーを削減したい

最適化が強化され、無駄なmemcpyが減りました

R2017b

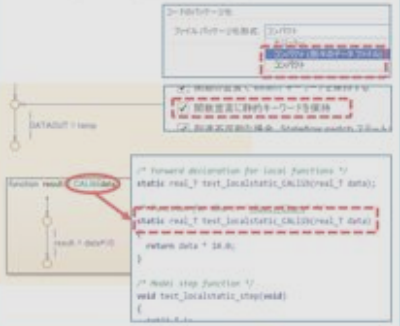
最適なコード生成設定を効率よく決めたい

Embedded Coder クイックスタートを利用することで、ウィザード形式で最適な設定を決定可能

R2015b

static関数を生成したい

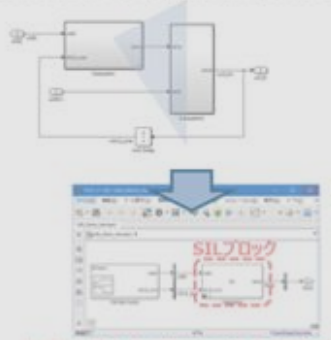
ファイルパッケージをモジュラー以外 かつ“関数宣言に静的キーワードを保持”をONにすることで生成可能



R2017b

B2Bテスト用のモデル作成を効率化したい

テストハーネスの作成機能の利用により、B2B用モデルの作成を効率化できる

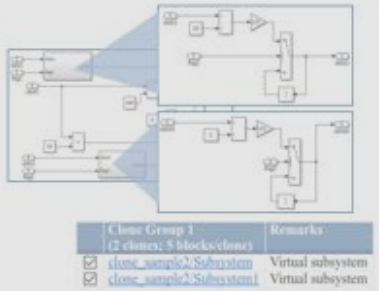


※: Simulink Test™のライセンスが別途必要

R2015a

類似サブシステムの再利用により、コード効率の良いモデルにしたい

モデリング クローンの特定機能により、再利用可能なサブシステムを検出可能

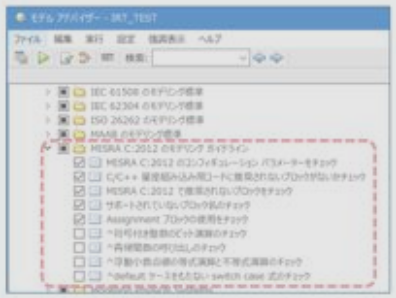


※: Simulink Check™のライセンスが別途必要

R2016b

MISRA C 2012準拠のコードを生成できるモデルを作りたい

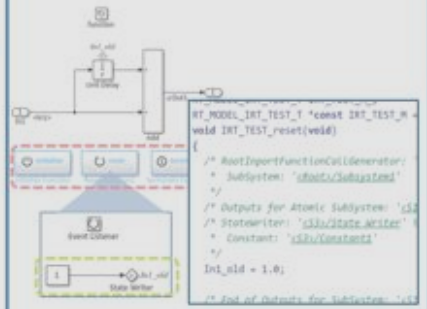
モデルアドバイザーのModeling Guidelines for MISRA C:2012でモデルの準拠状況をチェック可能



R2015b

初期化・リセット・終了時の処理関数をモデルで表現したい

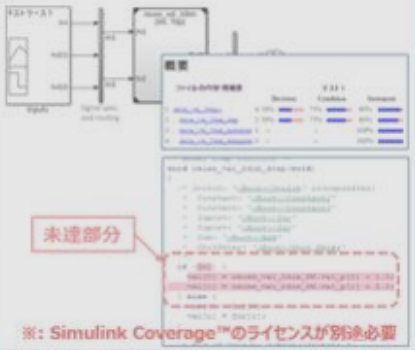
Initialize/Reset/Terminate Function内に処理を記述することで実現可能



R2016b

B2Bテスト実行時、コードカバレッジを測定したい

SILシミュレーション中に生成コードのカバレッジを測定できる



※: Simulink Coverage™のライセンスが別途必要

R2016a

大規模な配列の無駄なデータコピーを削減したい

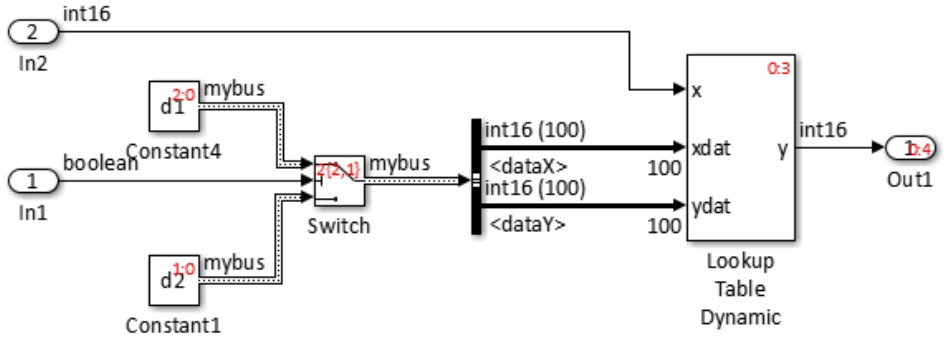
R2017b

最適化が強化され、無駄なmemcpyが減りました

Before

```
void rtwdemo_pointer_conversion_15b_step(void)
{
  int16_T rtb_Switch_dataX[100];
  int16_T rtb_Switch_dataY[100];

  /* Switch: '<Root>/Switch' incorporates:
   *  Inport: '<Root>/In1'
   */
  if (rtU.In1) {
    /* Switch: '<Root>/Switch' incorporates:
     *  Constant: '<Root>/Constant4'
     */
    memcpy(&rtb_Switch_dataX[0], (&(d1.dataX[0])), 100U * sizeof(int16_T));
    memcpy(&rtb_Switch_dataY[0], (&(d1.dataY[0])), 100U * sizeof(int16_T));
  } else {
    /* Switch: '<Root>/Switch' incorporates:
     *  Constant: '<Root>/Constant1'
     */
    memcpy(&rtb_Switch_dataX[0], (&(d2.dataX[0])), 100U * sizeof(int16_T));
    memcpy(&rtb_Switch_dataY[0], (&(d2.dataY[0])), 100U * sizeof(int16_T));
  }
```



After

```
void rtwdemo_pointer_conversion_17a_step(void)
{
  const int16_T *rtb_Switch_dataX;
  const int16_T *rtb_Switch_dataY;
  if (rtU.In1) {
    rtb_Switch_dataX = (&(d1.dataX[0]));
    rtb_Switch_dataY = (&(d1.dataY[0]));
  } else {
    rtb_Switch_dataX = (&(d2.dataX[0]));
    rtb_Switch_dataY = (&(d2.dataY[0]));
  }

  LookUp_S16_S16( &(rtY.Out1), &rtb_Switch_dataY[0],
```

ユースケースに対するEmbedded Coderソリューション

大規模な配列の無駄なデータコピーを削減したい

最適化が強化され、無駄なmemcpyが減りました

R2017b

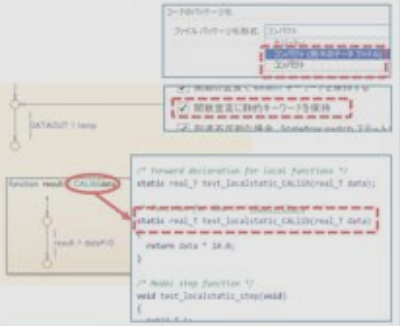
最適なコード生成設定を効率よく決めたい

Embedded Coder クイックスタートを利用することで、ウィザード形式で最適な設定を決定可能

R2015b

static関数を生成したい

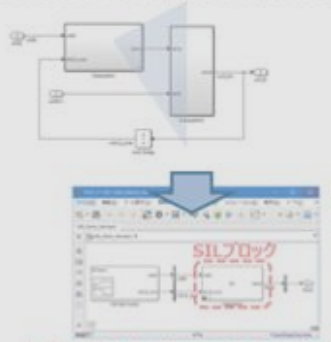
ファイルパッケージをモジュラー以外 かつ“関数宣言に静的キーワードを保持”をONにすることで生成可能



R2017b

B2Bテスト用のモデル作成を効率化したい

テストハーネスの作成機能の利用により、B2B用モデルの作成を効率化できる

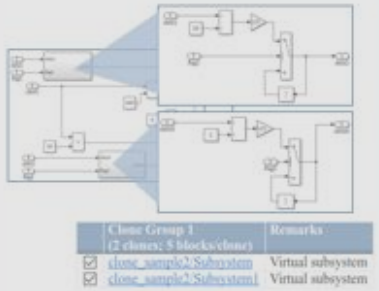


※: Simulink Test™のライセンスが別途必要

R2015a

類似サブシステムの再利用により、コード効率の良いモデルにしたい

モデリング クローンの特定機能により、再利用可能なサブシステムを検出可能

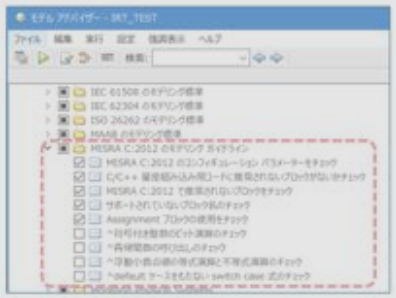


※: Simulink Check™のライセンスが別途必要

R2016b

MISRA C 2012準拠のコードを生成できるモデルを作りたい

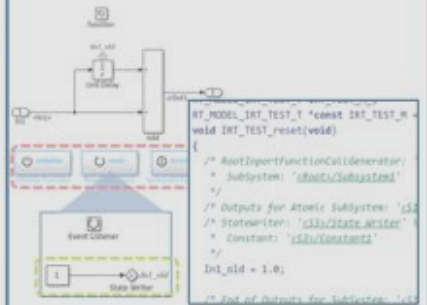
モデルアドバイザーのModeling Guidelines for MISRA C:2012でモデルの準拠状況をチェック可能



R2015b

初期化・リセット・終了時の処理関数をモデルで表現したい

Initialize/Reset/Terminate Function内に処理を記述することで実現可能



R2016b

B2Bテスト実行時、コードカバレッジを測定したい

SILシミュレーション中に生成コードのカバレッジを測定できる

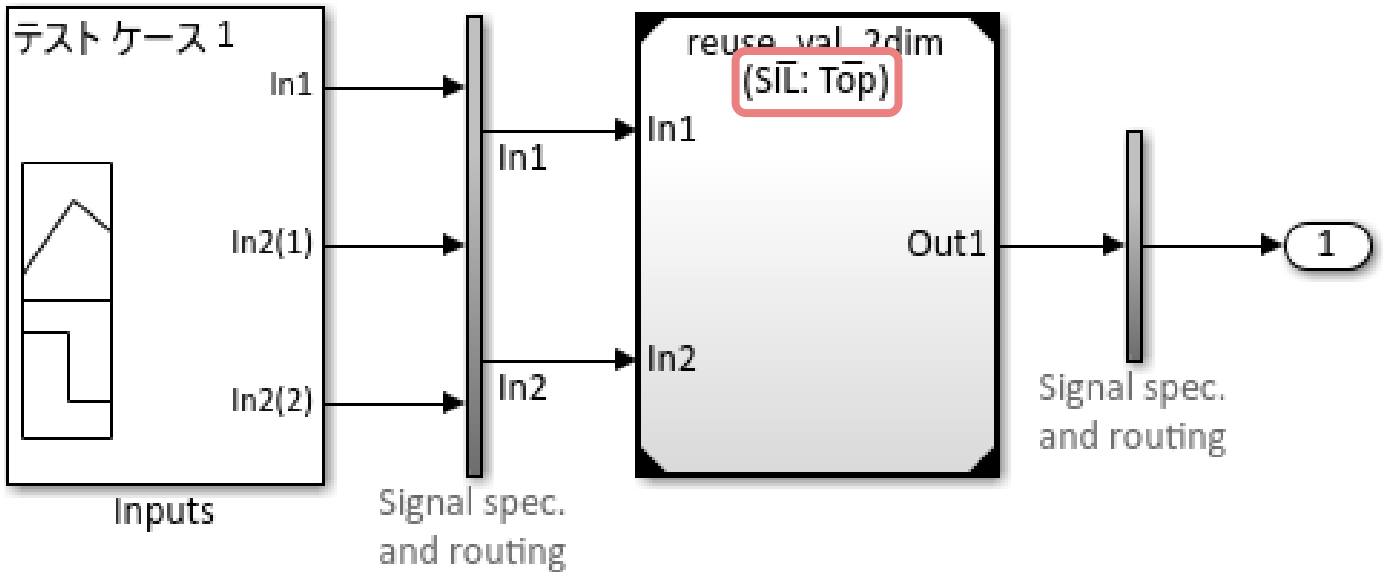
※: Simulink Coverage™のライセンスが別途必要

R2016a

B2Bテスト実行時、コードカバレッジを測定したい

R2016a

自動生成コードのカバレッジが計測できるようになりました



概要

ファイルの内容/複雑度		テスト 1		
		Decision	Condition	Statement
1 . reuse_val_2dim.c	4	50%	75%	90%
2 . . . reuse_val_2dim_step	2	50%	75%	80%
3 . . . reuse_val_2dim_initialize	1	--	--	100%
4 . . . reuse_val_2dim_terminate	1	--	--	100%

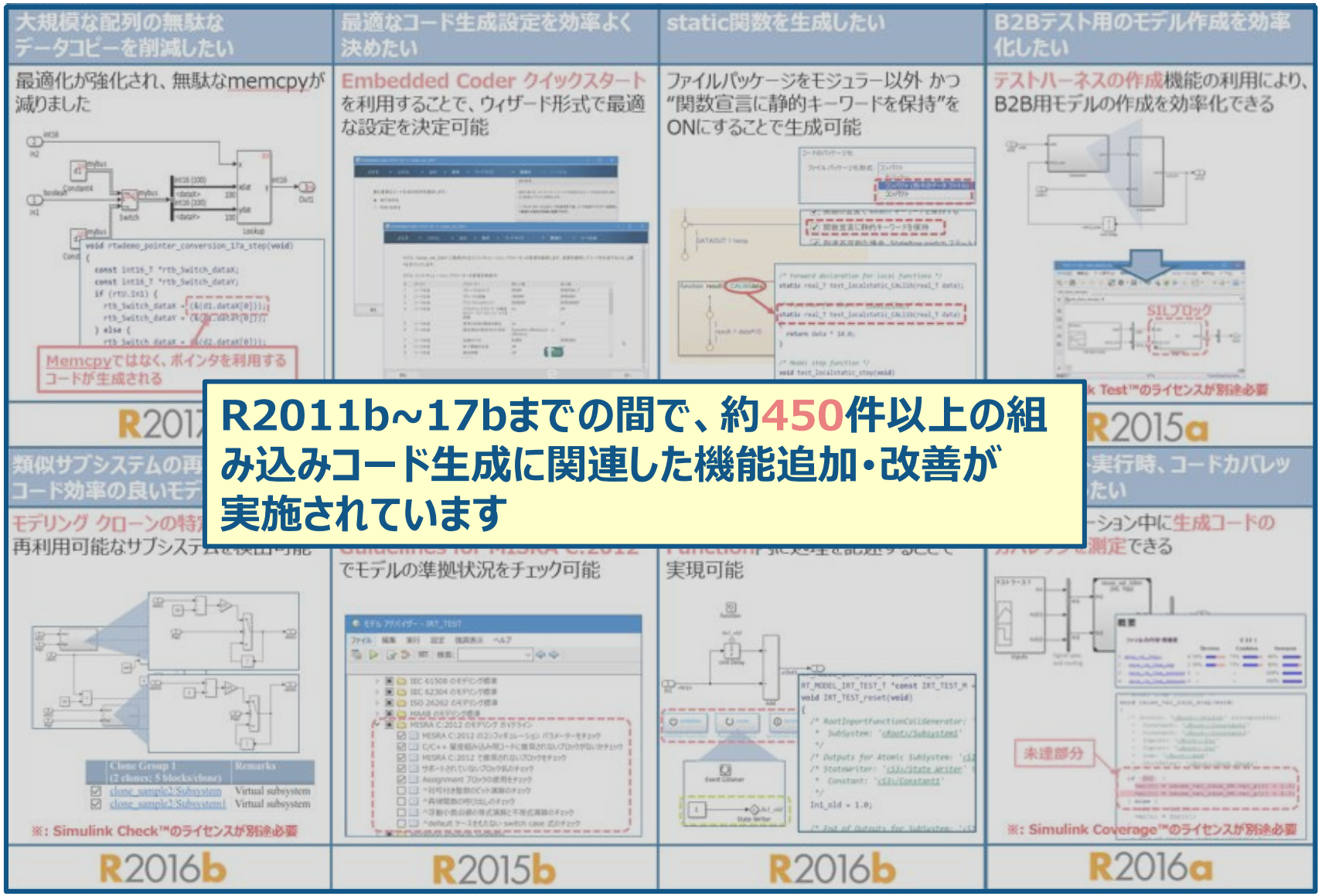
未達部分

```
/* Model step function */
void reuse_val_2dim_step(void)
{
  /* Switch: '<Root>/Switch' incorporates:
   *  Constant: '<Root>/Constant2'
   *  Constant: '<Root>/Constant3'
   *  Inport: '<Root>/In1'
   *  Inport: '<Root>/In2'
   *  Sum: '<Root>/Add'
   *  UnitDelay: '<Root>/Unit Delay'
   */
  if (In1) {
    val[0] = reuse_val_2dim_DW.val_p[0] + 1.0;
    val[1] = reuse_val_2dim_DW.val_p[1] + 2.0;
  } else {
    val[0] = In2[0];
    val[1] = In2[1];
  }

  /* End of Switch: '<Root>/Switch' */
```

※: Simulink Coverage™のライセンスが別途必要

ユースケースに対するEmbedded Coderソリューション



アジェンダ

1. Embedded Coderとは？
2. 自動コード生成ツール適用のメリット
3. **Embedded Coderの進化**
   - コード生成エンジンの進化
   - 機能の進化
   - **運用フローの進化**
4. まとめ

自動コード生成ツールの適用範囲拡大 / 大規模ソフト開発における課題

制御設計
・検証

実装統合
設計・検証

実装単体
設計・検証

コード生成

ソフト単体
検証

MATLABバージョンアップ

- MATLABバージョンアップすると、
  コードを再検証しないといけない

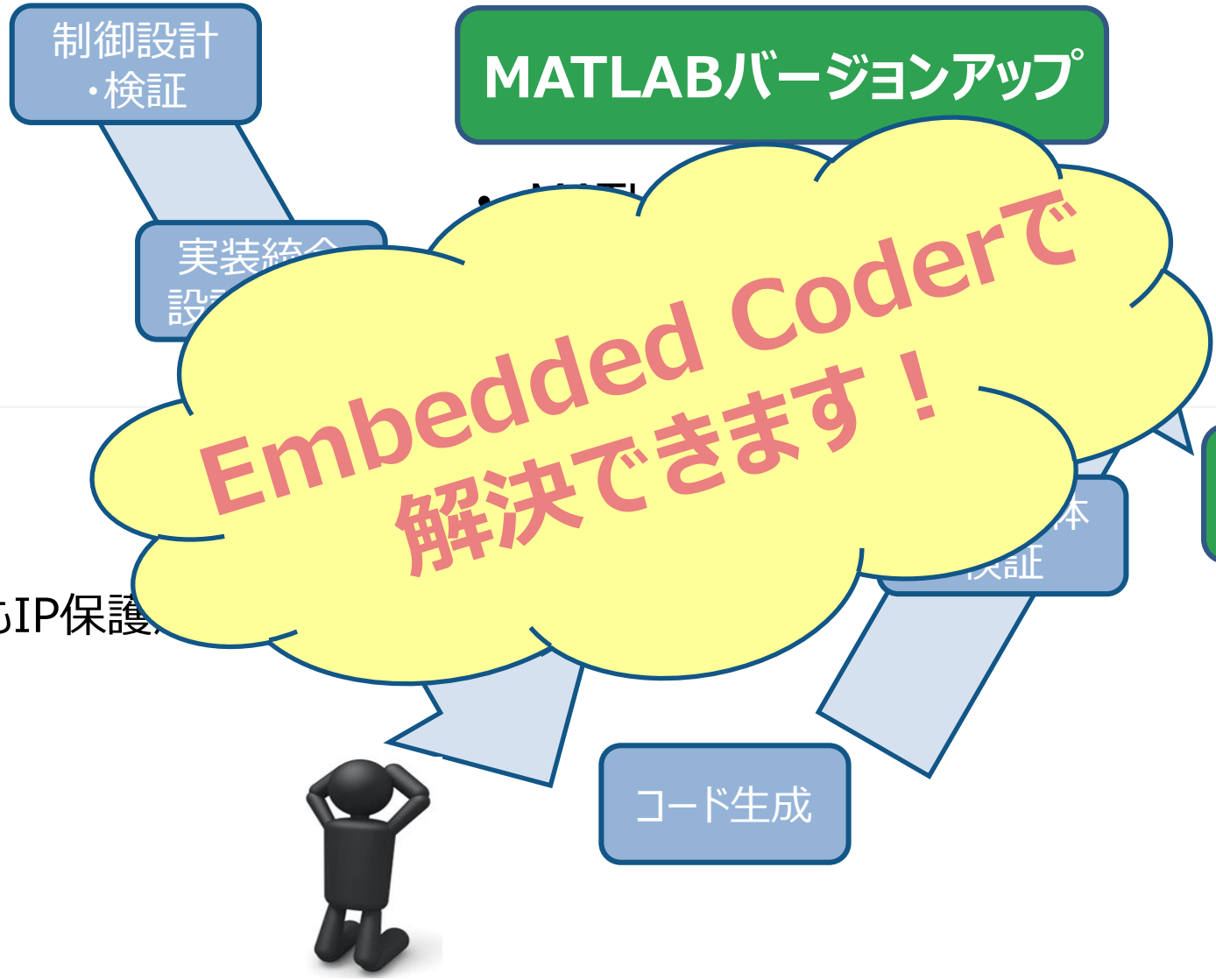
IP保護

- パートナー間でもIP保護が
  必要になる

PILS検証

- PILSは大事だが、
  HWを揃えるのが大変

自動コード生成ツールの適用範囲拡大 / 大規模ソフト開発における課題



IP保護

- パートナー間でもIP保護が必要になる

PILS検証

- PILSは大事だが、HWを揃えるのが大変

自動コード生成ツールの適用範囲拡大 / 大規模ソフト開発における課題

制御設計
・検証

実装統合
設計・検証

実装単体
設計・検証

コード生成

ソフト単体
検証

PILS検証

MATLABバージョンアップ

- MATLABバージョンアップすると、
  コードを再検証しないといけない

Cross Release

IP保護

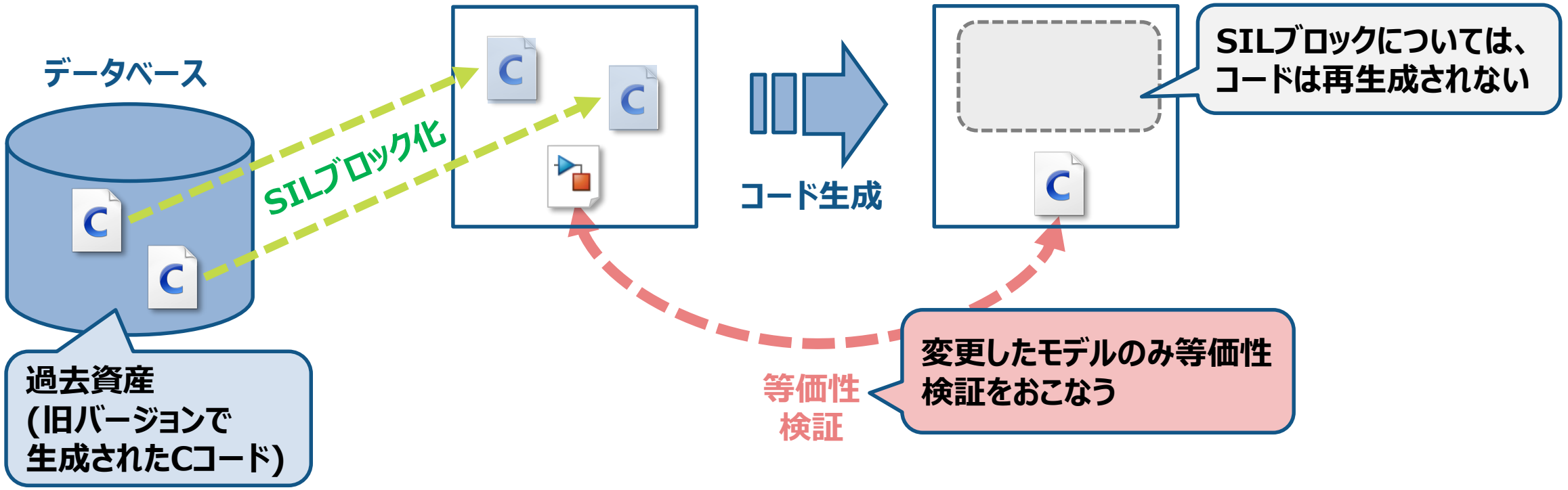
- パートナー間でもIP保護が
  必要になる

- PILSは大事だが、
  HWを揃えるのが大変

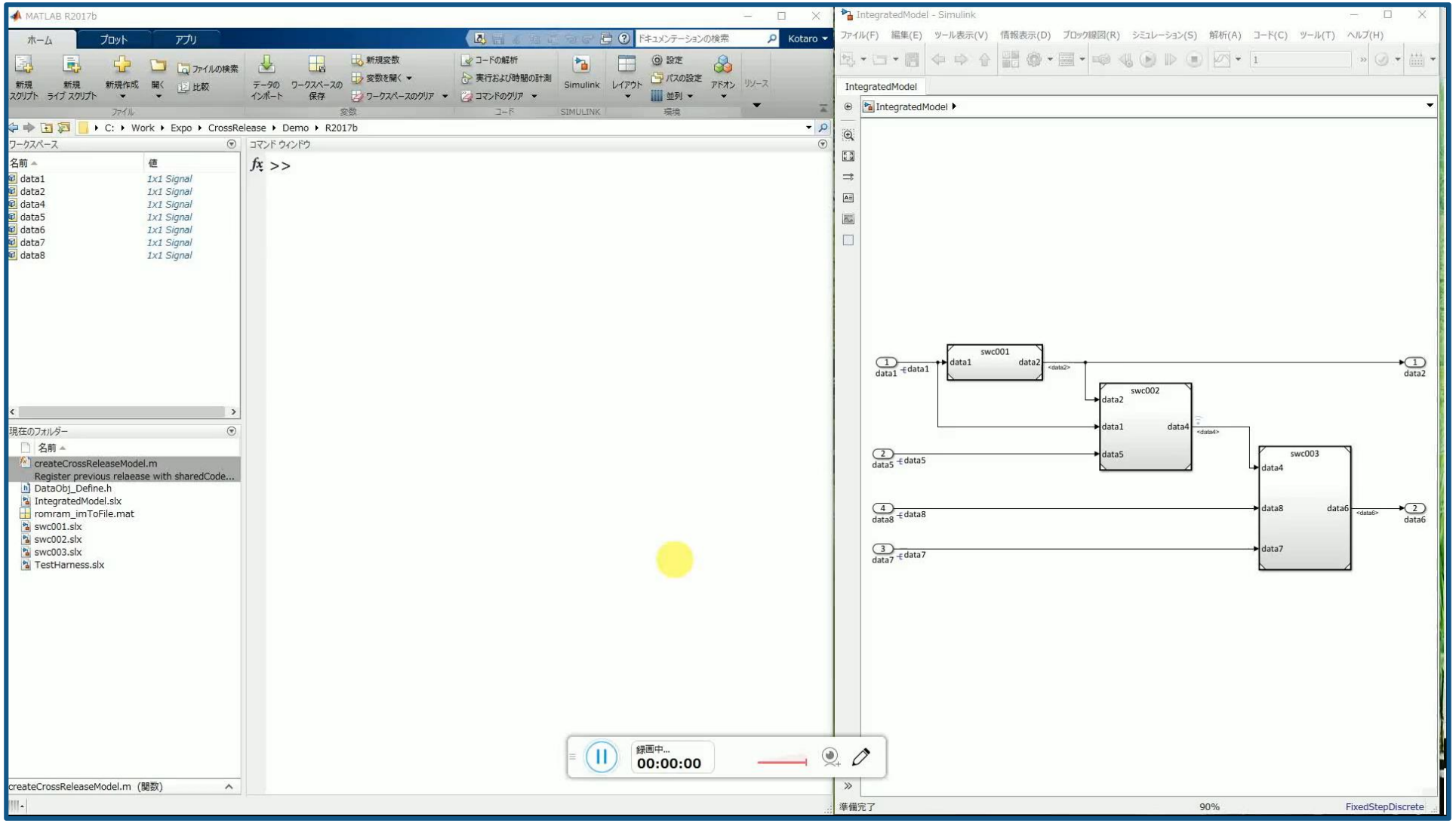
1. Cross Release

R2016b

- 旧バージョンで生成されたコードをSimulinkモデルに実装し、動作させることができるので、仕様変更が発生したモデルのみの検証に集中できます
- 本機能はR2010a以降で生成されたEmbedded Coderコードに対応しています



デモ: Cross Release



ポイント: Cross Release

- 過去生成コードを最新リリースのMATLABでそのまま再利用できます
- Simulink上で、過去生成コードをそのままシミュレーションできます
- Embedded Coderによるコード生成時、過去生成コードが取り込まれたSILブロックからは、コードは生成されません

MATLABバージョンアップ時のコード再検証が不要になります

自動コード生成ツールの適用範囲拡大 / 大規模ソフト開発における課題

制御設計
・検証

実装統合
設計・検証

実装単体
設計・検証

コード生成

ソフト単体
検証

PILS検証

MATLABバージョンアップ

- MATLABバージョンアップすると、
  コードを再検証しないといけない

IP保護

- パートナー間でもIP保護が
  必要になる

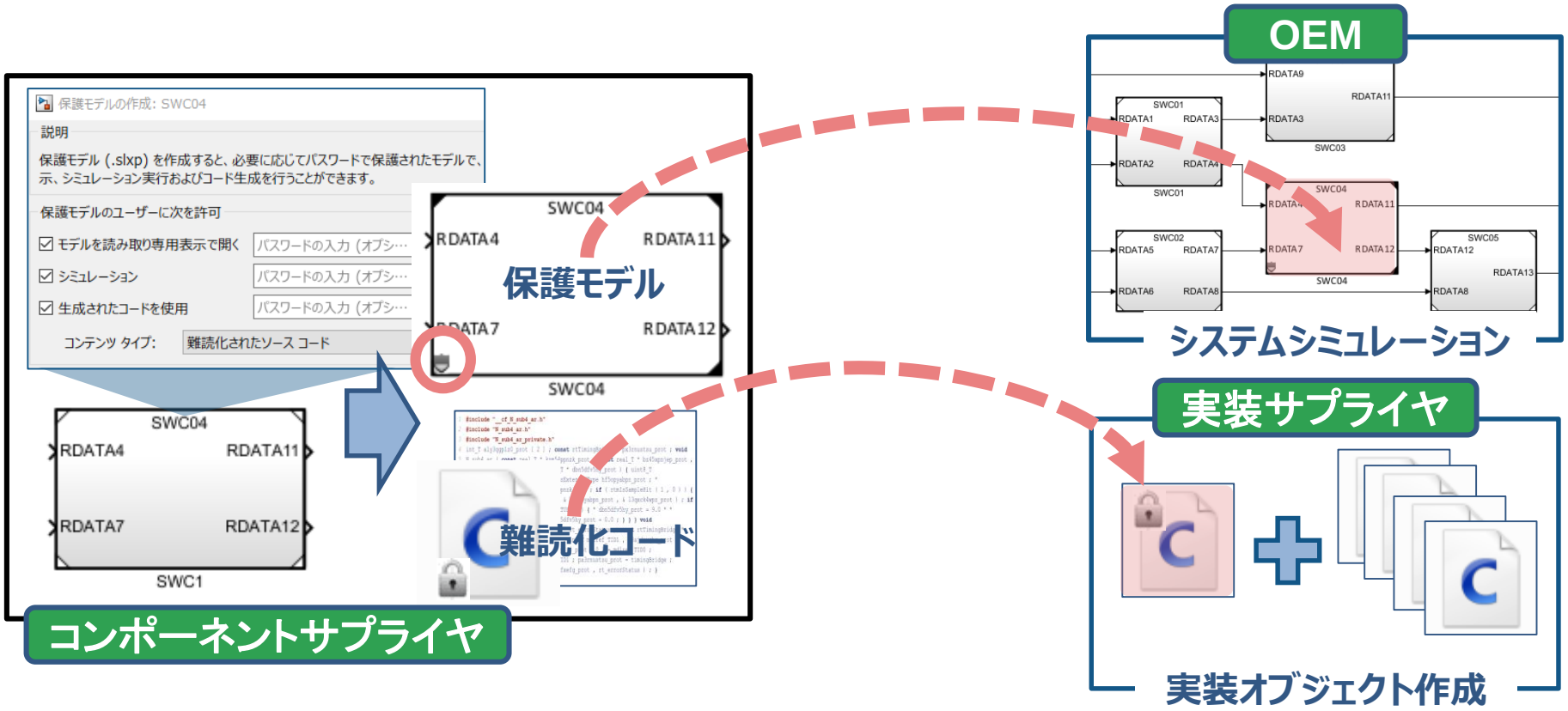
モデル保護

- PILSは大事だが、
  HWを揃えるのが大変

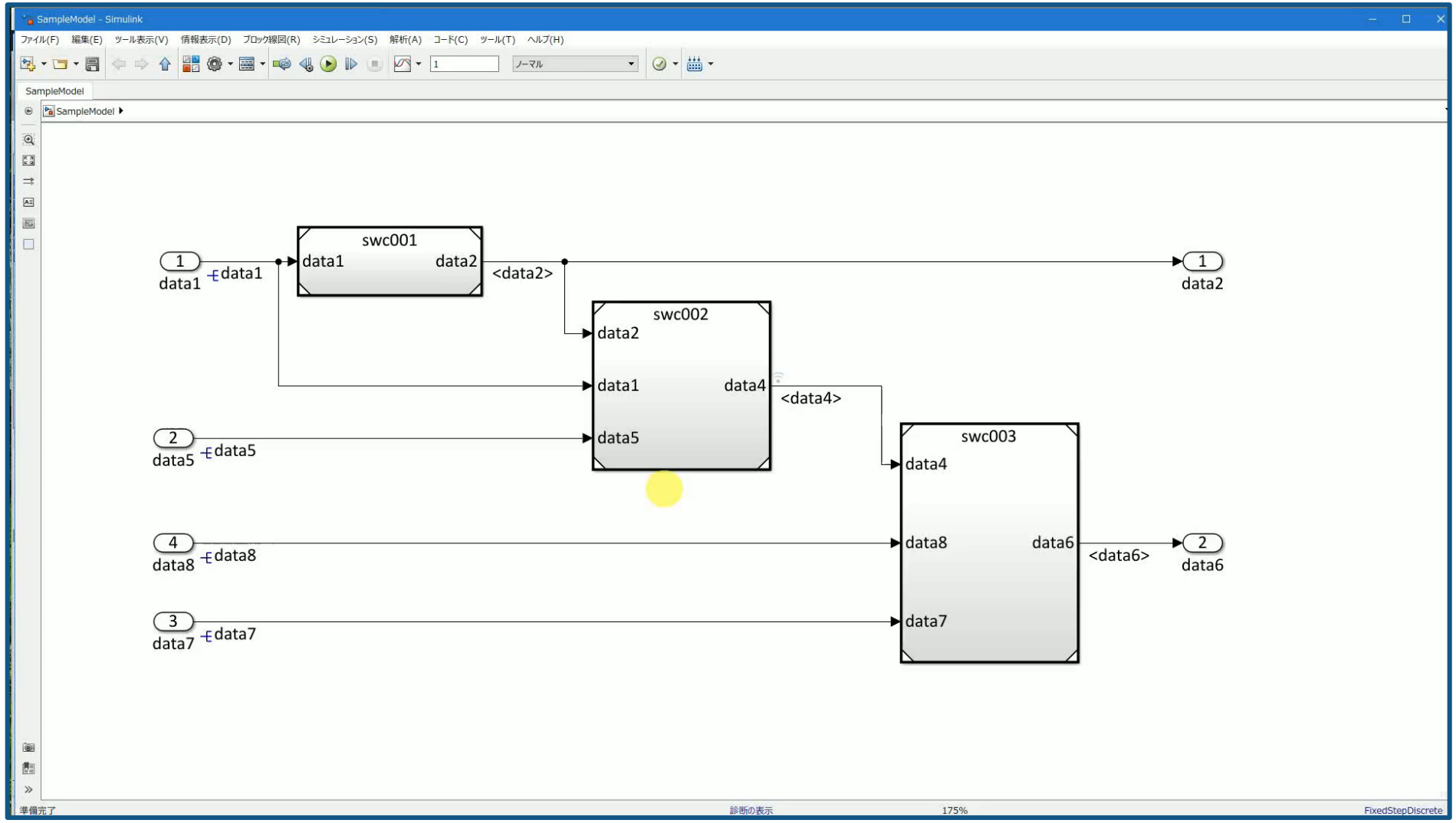
モデル保護

R2012b

- 部品の電制化が進んだことで、コンポーネントサプライヤは部品だけではなく、コントローラモデル・ソフトも開発し、OEMに納品するケースが増えてきています
- コンポーネントサプライヤと実装サプライヤが同一ではない場合は、モデルとコードのIP保護が重要なキーポイントとなります



デモ: モデル保護


SampleModel - Simulink
ファイル(F) 編集(E) ツール表示(V) 情報表示(D) ブロック線図(R) シミュレーション(S) 解析(A) コード(C) ツール(T) ヘルプ(H)
SampleModel
SampleModel
swc001
data1
data2
swc002
data2
data1
data4
data5
swc003
data4
data8
data6
data7
1
data1
data1
<data2>
1
data2
2
data5
data5
<data4>
4
data8
data8
<data6>
2
data6
3
data7
data7
準備完了
診断の表示
175%
FixedStepDiscrete

ポイント: モデル保護

- モデルの読み取り/シミュレーション/コード生成の可否をコントロールできます
- Embedded Coderによるコード生成時、生成コードの難読化有無を選択できます
- 本機能は、モデル参照ブロックのみサポートされます

モデル提供時のIP保護を提供先に応じて細かく設定できます

自動コード生成ツールの適用範囲拡大 / 大規模ソフト開発における課題

制御設計
・検証

実装統合
設計・検証

実装単体
設計・検証

コード生成

ソフト単体
検証

PILS検証

MATLABバージョンアップ

- MATLABバージョンアップすると、
  コードを再検証しないといけない

IP保護

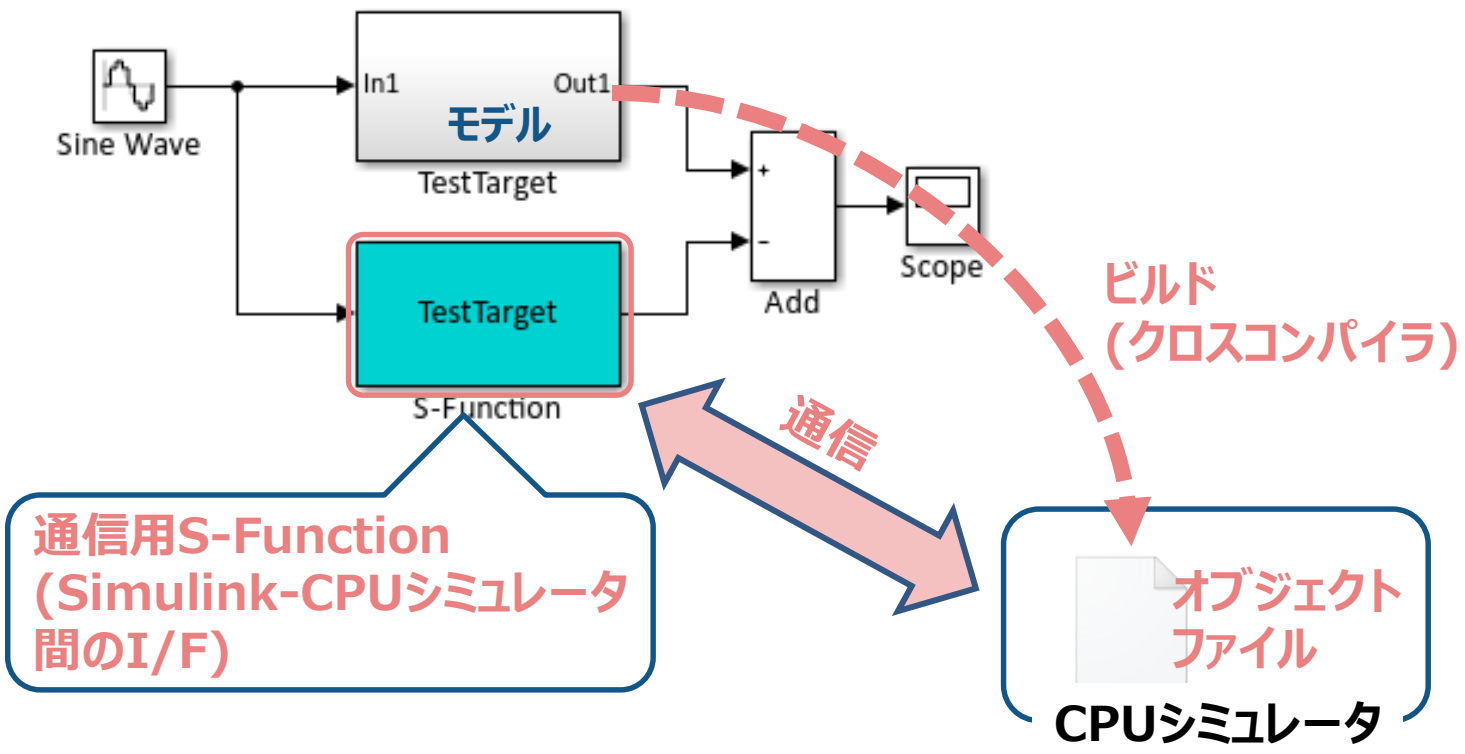
- パートナー間でもIP保護が
  必要になる

- PILSは大事だが、
  HWを揃えるのが大変

CPUシミュレータPILS

CPUシミュレータPILS

各マイコンベンダが提供するIDEのCPUシミュレータを利用することで、
評価ボードを使用せずにPC上でPILSによる検証が可能になります

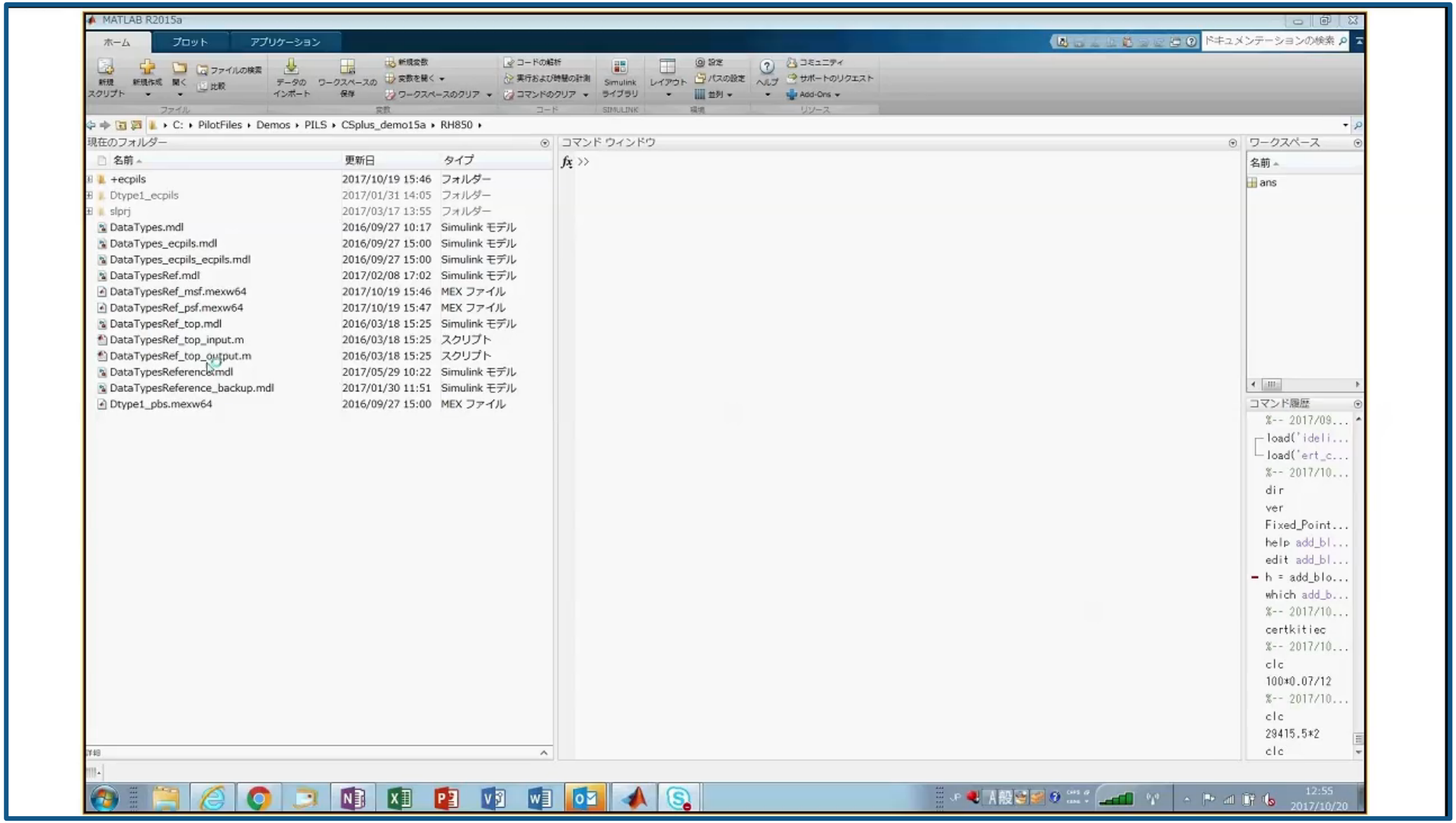


CPUシミュレータPILSによる等価性検証イメージ

対応マイコン	対応IDE, CPUシミュレーター
Renesas RH850, V850 RX/RL78, 78K0R	Renesas CS+ GreenHills MULTI Lauterbach Trace32
Renesas SH2, SH2A	Renesas HEW
ARM Cortex-M	ARM KEIL QEMU　ARM エミュレータ
ARM Cortex-A	GreenHills MULTI Lauterbach Trace32 QEMU　ARM エミュレータ
NXP MPC55xx	Lauterbach Trace32 NXP　Semiconductors　Motor Control Development Toolbox

PILS: Processer-In the Loop Simulation

デモ: CPUシミュレータPILS



ポイント: CPUシミュレータPILS

- Simulink & Embedded Coderは、各種IDEのCPUシミュレータと連携できます
- この仕組みを利用することで、評価ボードを使用せず、PILS検証が実施できます

H/Wの空き状況に依存しないので、
大人数による開発における検証を効率化できます

自動コード生成ツールの適用範囲拡大 / 大規模ソフト開発における課題

制御設計
・検証

実装統合
設計・検証

実装単体
設計・検証

コード生成

ソフト単体
検証

MATLABバージョンアップ

- MATLABバージョンアップすると、
  コードを再検証しないといけない

Cross Release

IP保護

- パートナー間でもIP保護が
  必要になる

モデル保護

PILS検証

- PILSは大事だが、
  HWを揃えるのが大変

CPUシミュレータPILS

アジェンダ

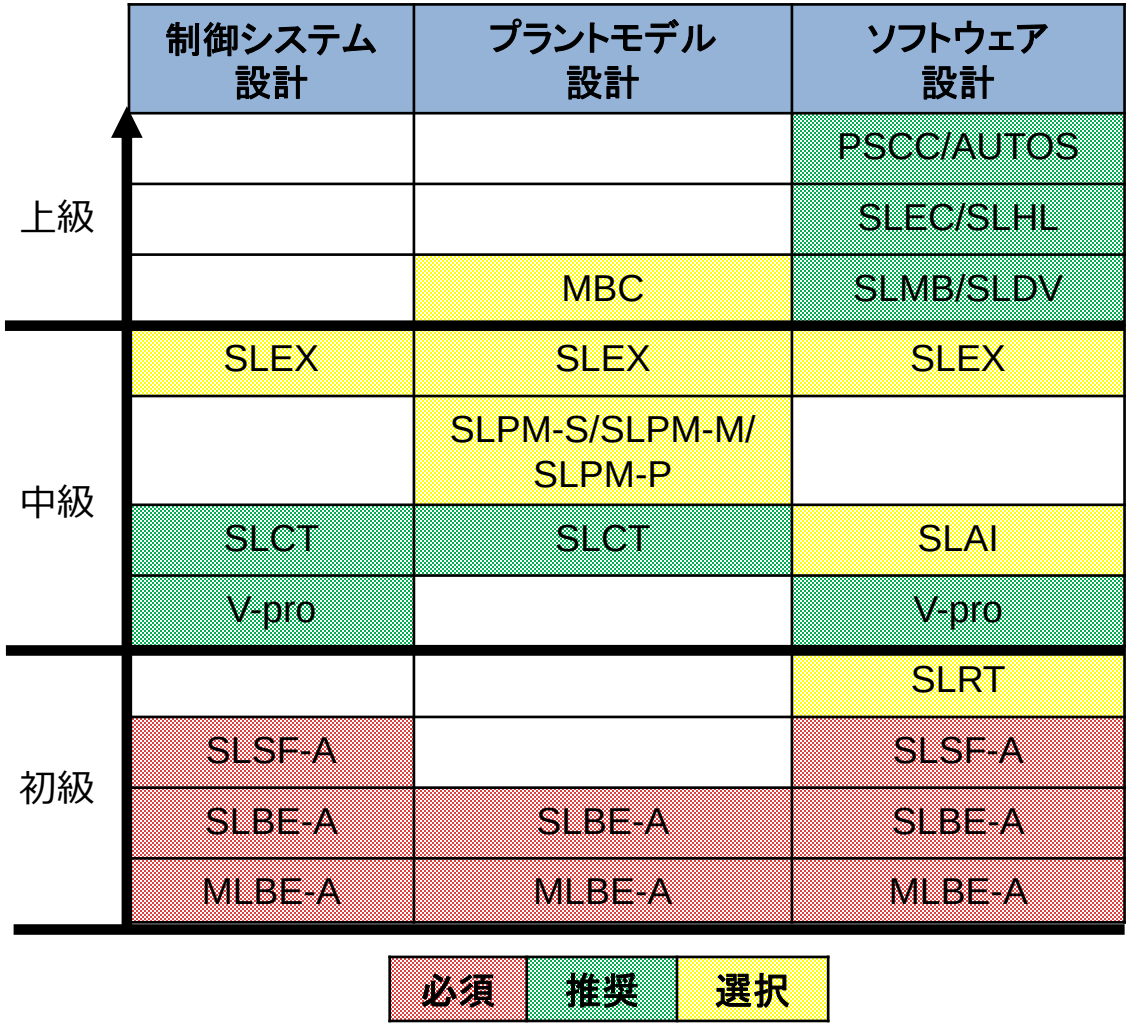
1. Embedded Coderとは？
2. 自動コード生成ツール適用のメリット
3. Embedded Coderの進化
   - コード生成エンジンの進化
   - 機能の進化
   - 運用フローの進化
4. **まとめ**

まとめ

- Embedded Coderは、様々な分野のお客様の製品開発に適用されています
- MathWorksは、Embedded Coderのバージョンアップ および 関連ツールボックスとの連携により、従来の量産コード生成プロセスに関する課題を解決してきました
- お客様の様々なユースケースにおける課題のソリューションを提供するため、MathWorksは日々、Embedded Coderを進化させつづけています

MathWorksが提供するMBDサポート

トレーニング（参考）　制御設計/組み込み系コースマップ

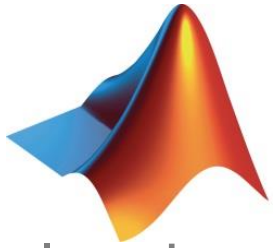


MathWorks トレーニングコース

http://www.mathworks.co.jp/services/training/courses/

コード	コース名
MLBE-A	自動車分野向け MATLAB基礎
SLBE-A	自動車分野向け Simulink 基礎
SLSF-A	自動車分野向け Stateflow 基礎
SLRT	Simulink Coder 基礎
SLCT	MATLAB と Simulink による制御設計
SLPM-S	Simscape によるマルチドメインシステムの物理モデリング
SLPM-M	SimMechanics によるの力学系の物理モデリング
SLPM-P	SimPowerSystems(開発中)
MBC	Model –Based Calibration
SLEX	Simulink とC言語のインタフェース
SLAI	Simulink/Stateflow API
SLMB	Simulink でのモデルの管理と検証
SLDV	Simulink Design Verifier (開発中)
SLEC	Embedded Coderによる量産向けコード生成
AUTOS	AUTOSAR(0.5 day)
SLHL	HDL Coder によるHDLコード生成
PSCC	Polyspace による静的コード検証
V-pro	V-プロセス実習（オンサイトのみ）

ご清聴ありがとうございました



MathWorks®

Accelerating the pace of engineering and science