DynaSpec

モデルベース形式検証ツールDynaSpecによる発見困難な不具合の自動検出

2021/11/26

製造企画マーケティング部
太田洋二郎

構造計画研究所 KOZO KEIKAKU ENGINEERING Inc.
株式会社　構造計画研究所
〒164-0001 東京都中野区中央4-5-3

❒ AGENDA

■本セッションでお伝えしたいこと

- □KKEが提案する設計の姿
- □KKE提案の設計・検証手法のメリット
- □モデルベース形式検証ツールDynaSpecのご紹介

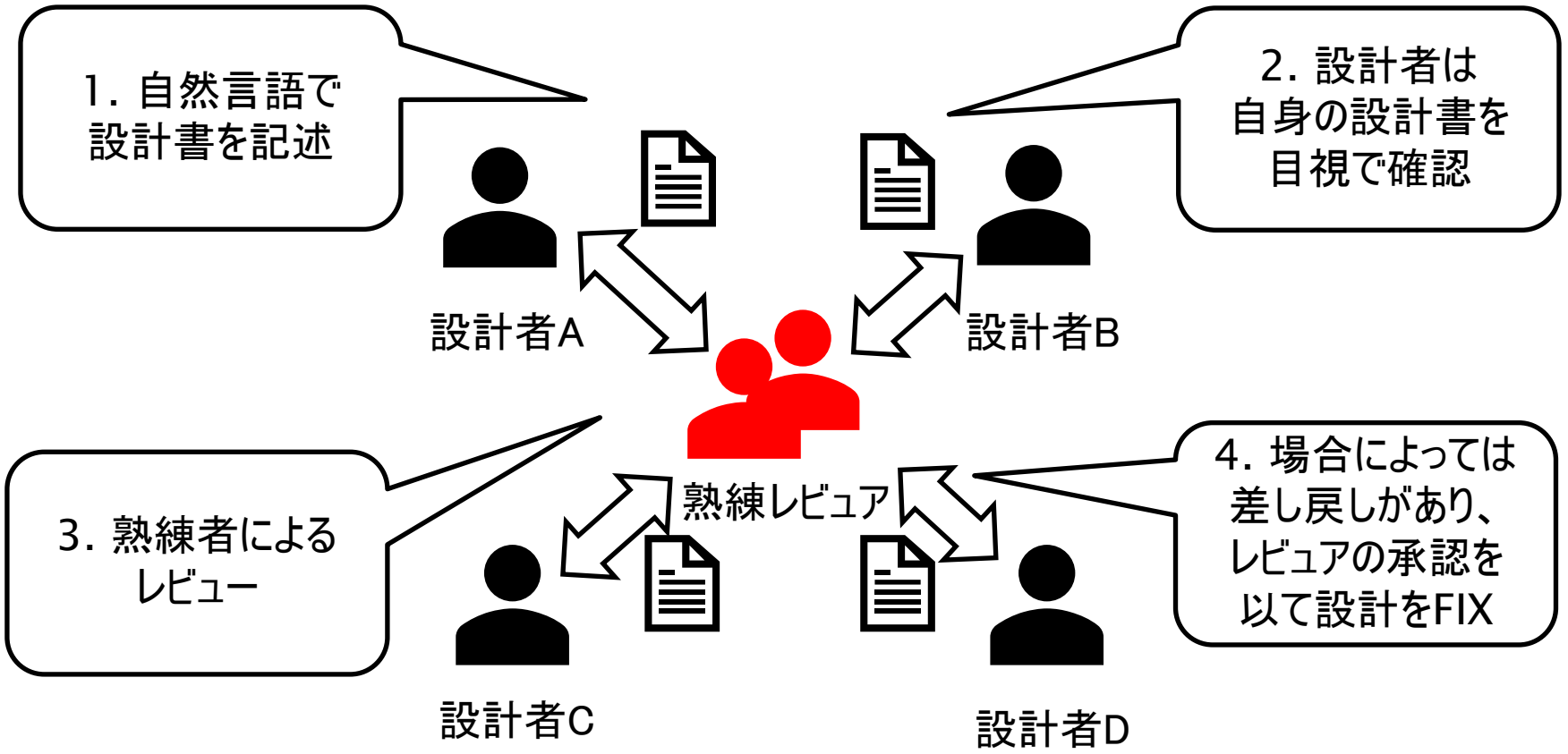
■SysMLサンプルモデルによる形式検証適用例のご紹介

- □適用対象システム
- □DynaSpecによりSysMLモデルを形式検証するための実行ステップ
- □システム設定モデル
- □検証条件モデル
- □DynaSpec検証デモ

■まとめ

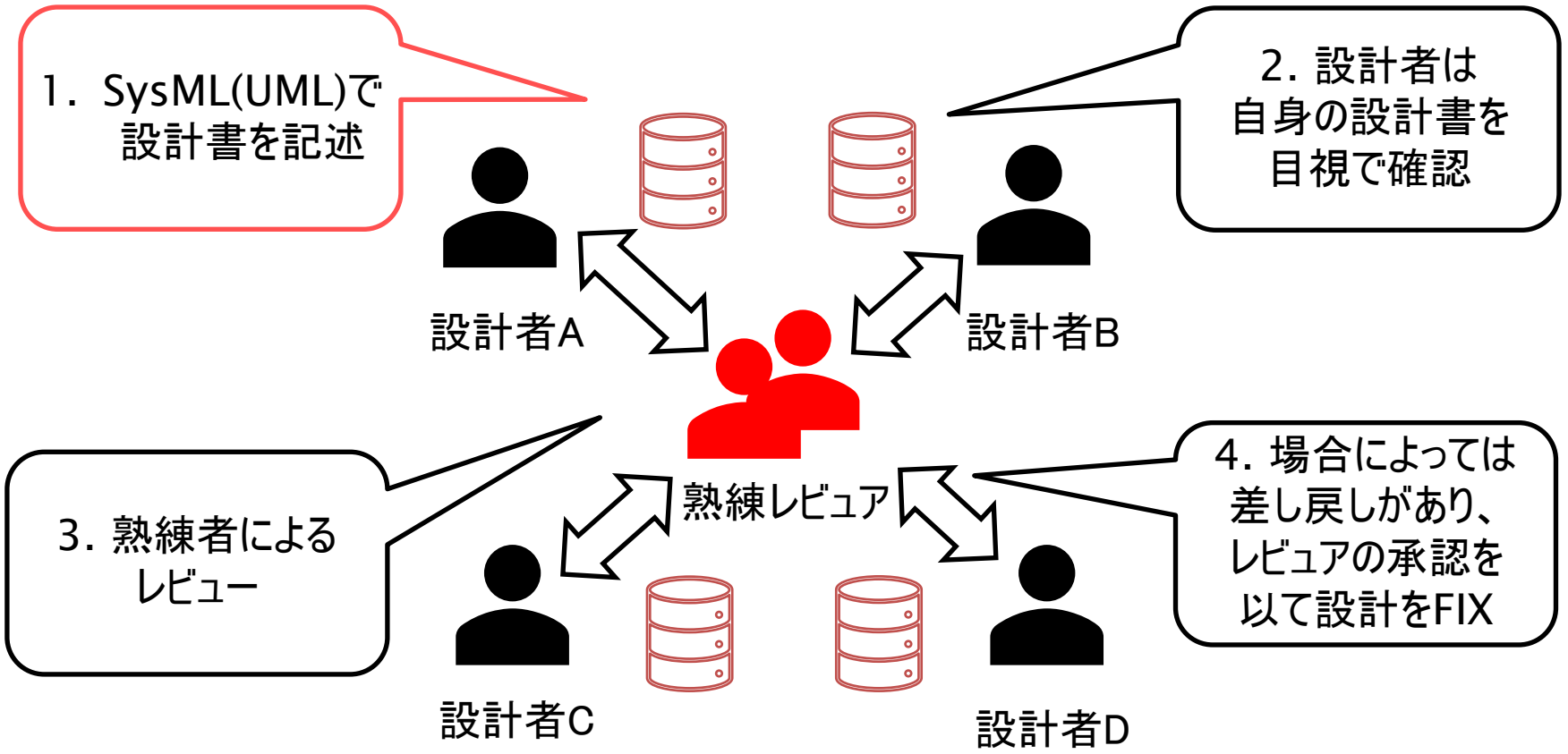
❐本セミナーでお伝えしたいこと
～「モデリング＋ツールによる検証」で設計者自身が検証できる環境を！～

従来のシステム設計と検証方法



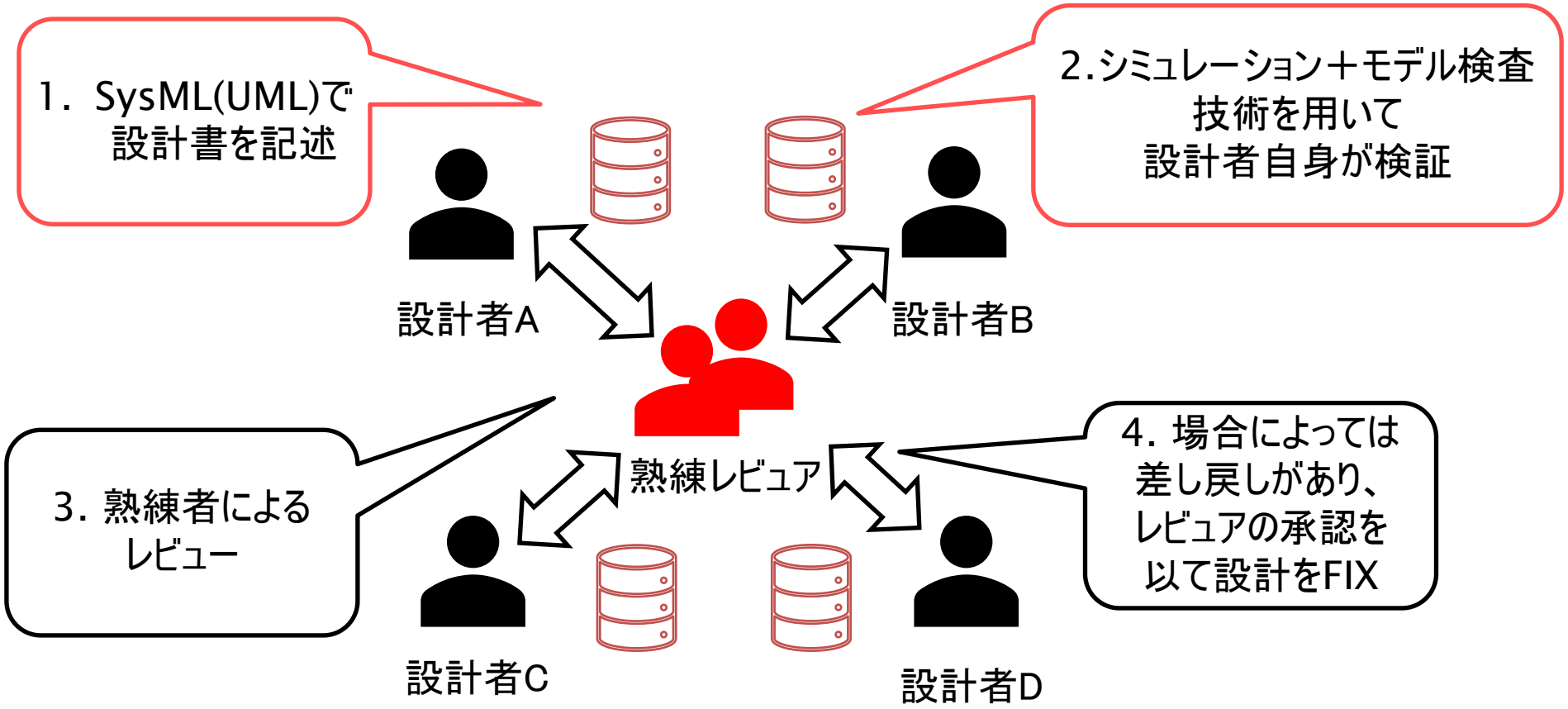
❐本セミナーでお伝えしたいこと
～「モデリング＋ツールによる検証」で設計者自身が検証できる環境を！～

KKEが提案する設計の姿



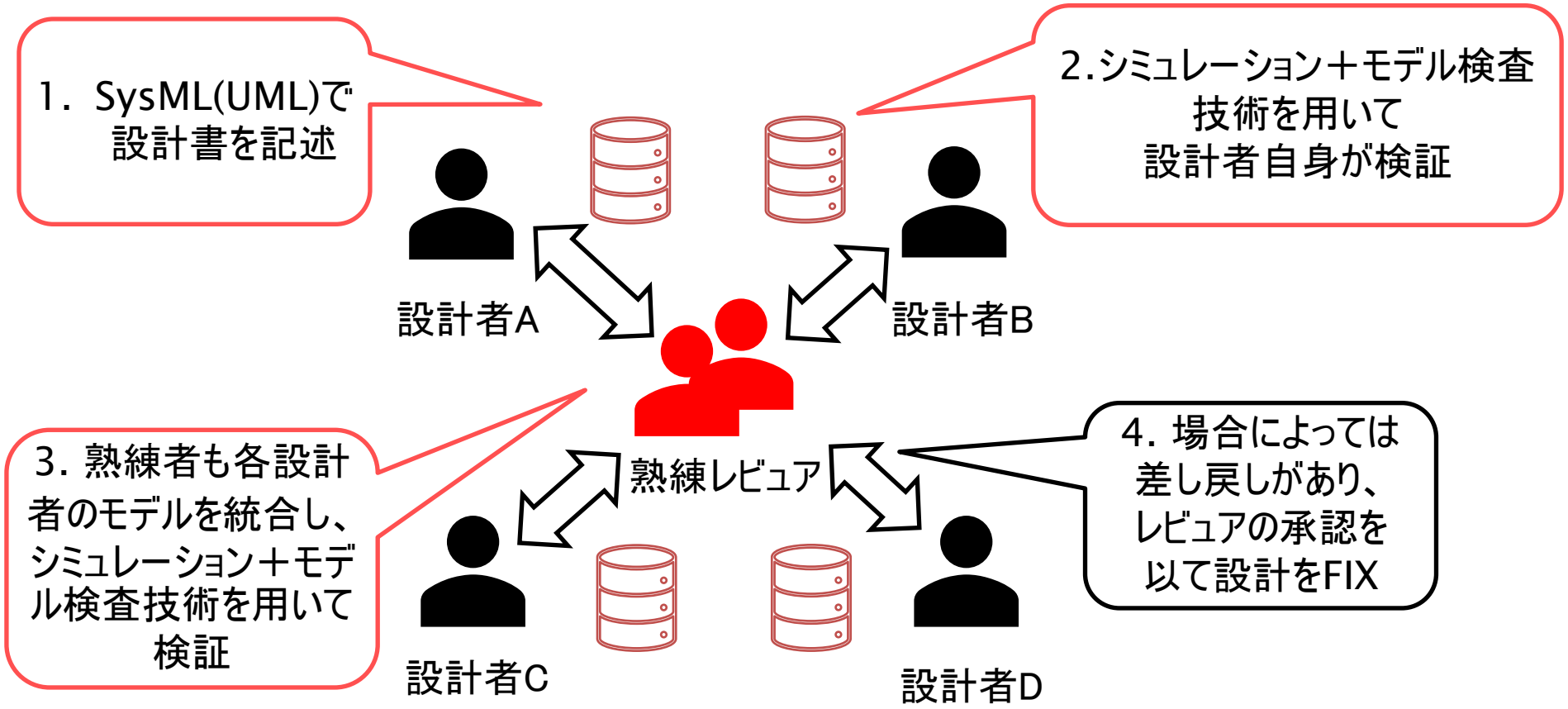
❐本セミナーでお伝えしたいこと
～「モデリング＋ツールによる検証」で設計者自身が検証できる環境を！～

KKEが提案する設計の姿



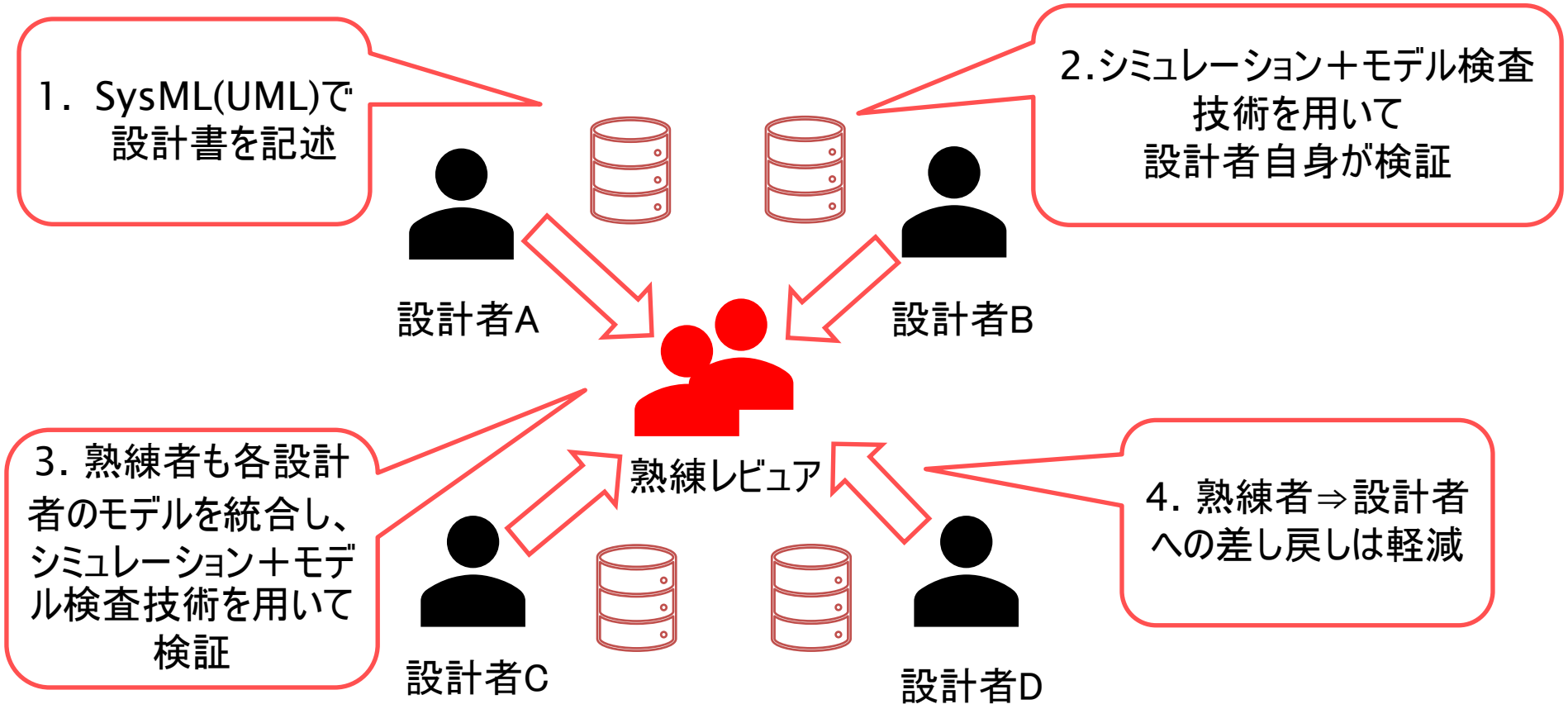
❐本セミナーでお伝えしたいこと
～「モデリング＋ツールによる検証」で設計者自身が検証できる環境を！～

KKEが提案する設計の姿



❐本セミナーでお伝えしたいこと
～「モデリング＋ツールによる検証」で設計者自身が検証できる環境を！～

KKEが提案する設計の姿



❐本セミナーでお伝えしたいこと
～「モデリング＋ツールによる検証」で設計者自身が検証できる環境を！～

KKE提案の設計・検証手法のメリット

✓ **仕様に関する理解齟齬軽減**
- UML/SysMLモデルを用いることで、仕様に関して曖昧な表現がなくなる。

✓ **熟練レビュアへの依存軽減**
- シミュレーション＋モデル検査を導入することにより、属人性が軽減する。

✓ **設計⇔レビューの手戻り軽減**
- 設計者自身が検証できるので、レビュアの負荷軽減＋手戻りも軽減する。

✓ **発見困難な不具合を早期発見**
- モデル検査技術を用いると、人間系では発見困難な不具合も検出可能になる。

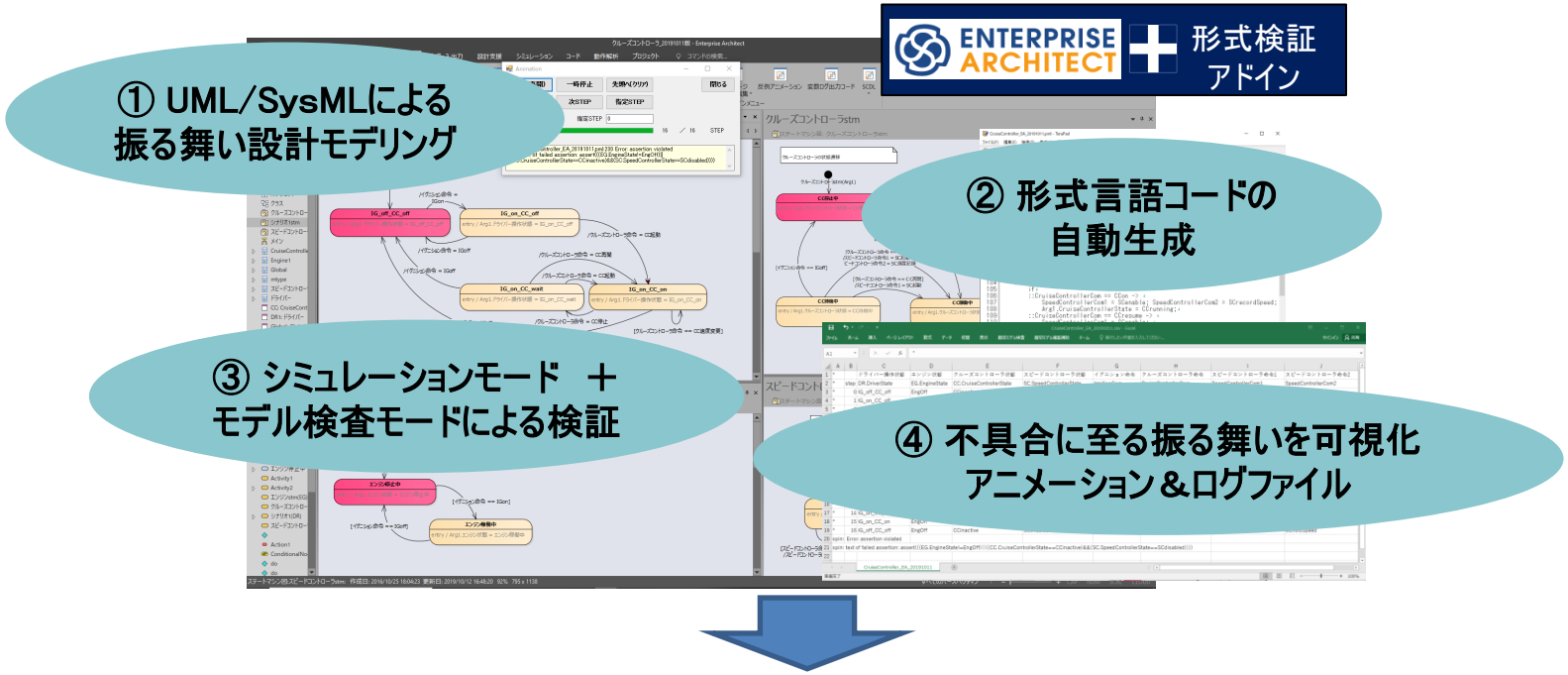
ENTERPRISE ARCHITECT ＋ DynaSpec によって実現

❑本セミナーでお伝えしたいこと
～「モデリング＋ツールによる検証」で設計者自身が検証できる環境を！～

モデルベース形式検証ツール「DynaSpec」はSparx Systems社のモデリングツールEnterprise Architectのアドインツールであり、以下の特徴を持ちます。

- EA上の操作でモデリング～シミュレーション＋モデル検査～結果可視化まで可能
- 専用言語習得不要で、形式検証(モデル検査)を実施可能

DynaSpec（Dynamic Specification：動く仕様）



導入ハードルが高かった形式検証を開発現場へ！

❒DynaSpecを用いないモデル検査

- DynaSpecを使用せずモデル検査を行うと、以下のような問題に直面するかと思います。

モデル検査専用言語
(SPINであればPromela)
を習得する必要がある。

```
// 構造体／変数の定義
mtype IgnitionCom;  // attribute Name:イグニション命令 //
int step;   // attribute Name:step //
mtype CruiseControllerCom;  // attribute Name:クルーズコントローラ命令 //
mtype SpeedControllerCom1;  // attribute Name:スピードコントローラ命令1 //
mtype SpeedControllerCom2;  // attribute Name:スピードコントローラ命令2 //

typedef _Engine // ElementID:247 Name:Engine //
{
    mtype EngineState;  // attribute Name:エンジン状態 //
}

typedef _Driver // ElementID:221 Name:ドライバー //
{
    mtype DriverState;  // attribute Name:ドライバー操作状態 //
}

typedef _SpeddController    // ElementID:217 Name:スピードコントローラ //
{
    mtype SpeedControllerState; // attribute Name:スピードコントローラ状態 //
}

typedef _CruiseController   // ElementID:235 Name:CruiseController //
{
    mtype CruiseControllerState;    // attribute Name:クルーズコントローラ状態 //
}
bool initializingAlreadyCompletedFlag = false;  // attribute Name:initializingAlreadyCompletedFlag //
bool initializingAlreadyCompletedFlagObject = false;    // attribute Name:initializingAlreadyCompletedFlagObject //
mtype {EngOn,EngOff,CCrunning,CCwait,CCinactive,SCenabled,SCdisabled,IG_off_CC_off,IG_on_CC_off,IG_on_CC_on,IG_on_CC_wait,IGoff,IG

// インスタンス／変数初期値の定義(クラス図/ブロック定義図)
```

Promelaコード

・検査結果の可読性が著しく低い。
・不具合の存在を検出できても、
反例ログをトレースが難しい。

```
■SPIN実行（pan.cファイルの生成）：
spin -a UML_1.pml

■gcc実行（pan.exeファイルの生成）：
gcc -DMEMLIM=80000 -O2  -DCOLLAPSE -DVECTORSZ=5000 -DXUSAFE -DSAFETY -DNOCLAIM -w -o pan pan.c

■モデル検査実行（pan.exeの実行）：
pan -m1000

pan:1: assertion violated ((EG.EngineState!=21)||((CC.CruiseControllerState==18)&&(SC.SpeedControllerState==16))) (at depth 65)
pan: wrote UML_1.pml.trail

(Spin Version 6.5.1 -- 6 June 2020)
Warning: Search not completed
  + Partial Order Reduction
  + Compression

Full statespace search for:
  never claim           - (not selected)
  assertion violations  +
  cycle checks          - (disabled by -DSAFETY)
  invalid end states  +

State-vector 28 byte, depth reached 65, errors: 1
       66 states, stored
        0 states, matched
       66 transitions (= stored+matched)
        0 atomic steps
hash conflicts:         0 (resolved)

Stats on memory usage (in Megabytes):
    0.003 equivalent memory usage for states (stored*(State-vector + overhead))
    1.303 actual memory usage for states
```

モデル検査器SPINの結果ファイル

DynaSpecを使えば…

**専用言語を習得することなく、
UML/SysMLモデルからコードを自動生成！**

**反例ログをCSVベース、もしくは
アニメーションで可視化することが可能！**

◻ここまでのまとめ

ENTERPRISE ARCHITECT によるモデル化　＋

モデルベース形式検証ツール DynaSpec によって

設計者自身が検証できる環境を整備することで

⇩

仕様に関する理解齟齬軽減

熟練レビュアへの依存軽減

設計⇔レビューの手戻り軽減

発見困難な不具合を早期発見

を実現することが可能です

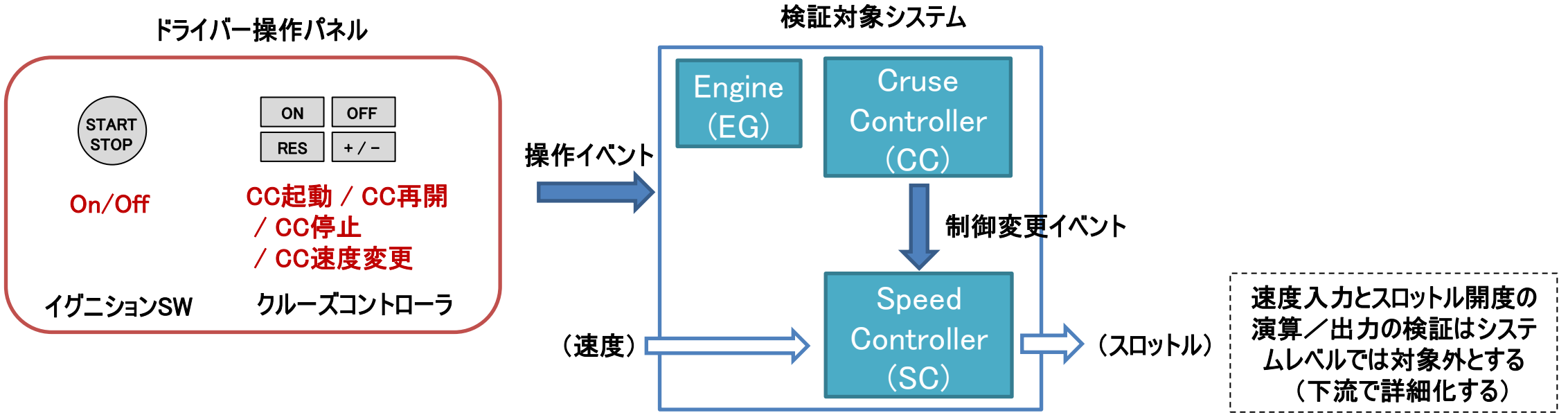
SysMLサンプルモデルによる形式検証適用例のご紹介

- クルーズコントローラシステムのモデル化と検証例

❒ DynaSpec適用対象システム ： クルーズコントローラ

- システム設計
  - サブシステム構成と制御イベントを設計
  - 各コントローラの制御仕様を設計（状態遷移図）
- 検証方針
  - ドライバーが想定外の操作を行っても制御が破綻しないかをモデル検査で検証
    - ➡ サブシステム間の**制御モードの不整合**、**デッドロック**が起きないか等

サンプルモデルは、中島震著
「 SPINモデル検査」のクルーズコントローラ
システムを参考にしています。



❑SysMLの一般的なモデル化プロセス

要求定義・分析

- 要求図
- ユースケース図

システム設計

論理設計

- ブロック定義図
- 内部ブロック図
- ステートマシン図
- アクティビティ図

物理設計

- ブロック定義図
- 内部ブロック図
- パラメトリック図

システム設計の論理設計、
すなわち、システムの振る舞いを検証する手法
「モデル検査」について本日はご紹介します

❑DynaSpecによりSysMLモデルを形式検証するための実行ステップ

要求定義・分析

要求図

ユースケース図

✓ 要求の定義に加え、要求を満たすかどうかを検証するためテストケースと関連づける。

システム設計

【システム設計モデル】

ブロック定義図

ステートマシン図

アクティビティ図

✓ 要求を実現するためのアーキテクチャを設計する（構造、振る舞い）。

形式検証モデルへ拡張

＋ 【検証条件モデル】

ブロック定義図

ステートマシン図

アクティビティ図

検証対象ダイアグラム

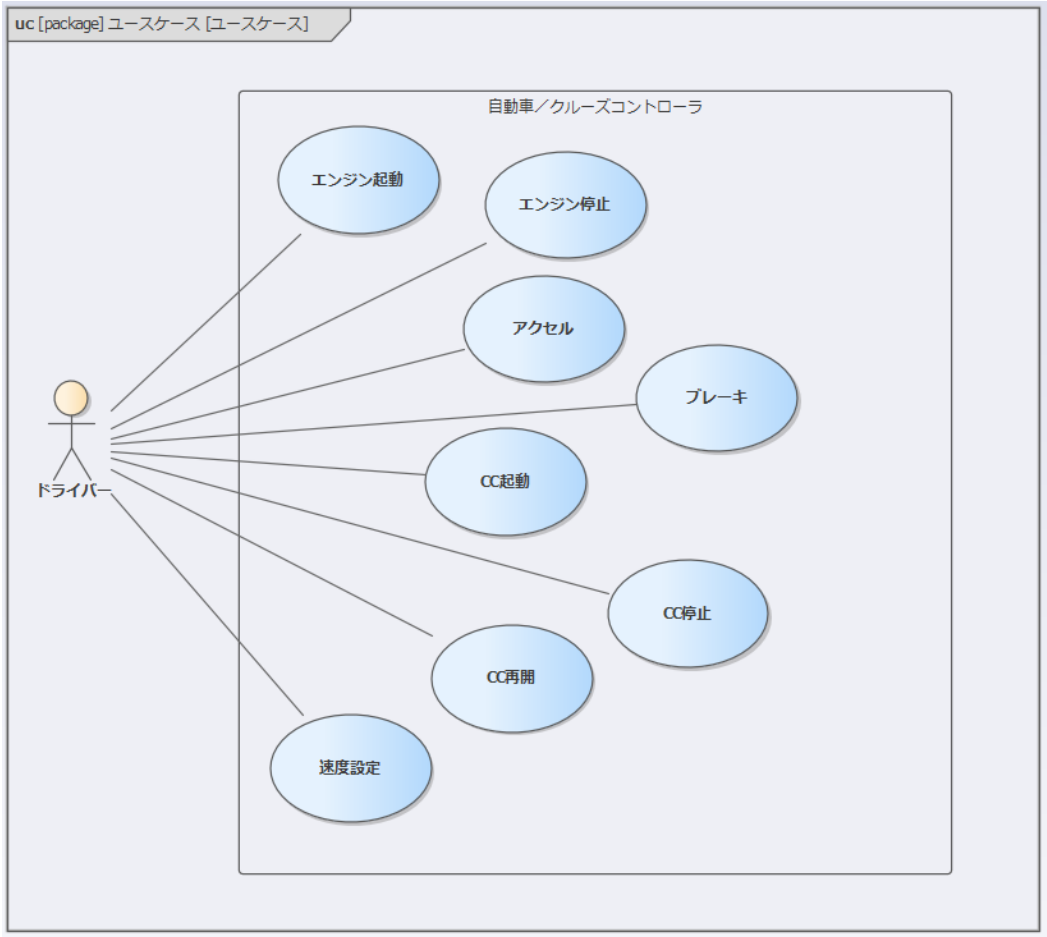
✓ システムが取りうる振る舞いを網羅的に再現させるため、外部環境モデルを新たに作成する。（**検証シナリオに相当するもの**）

✓ 要求定義段階で導出した**テストケースを、検査式として再定義する**。

✓ システム全体をどのようなロジック、プロセスで動かすかをメインプロセスとして定義する。

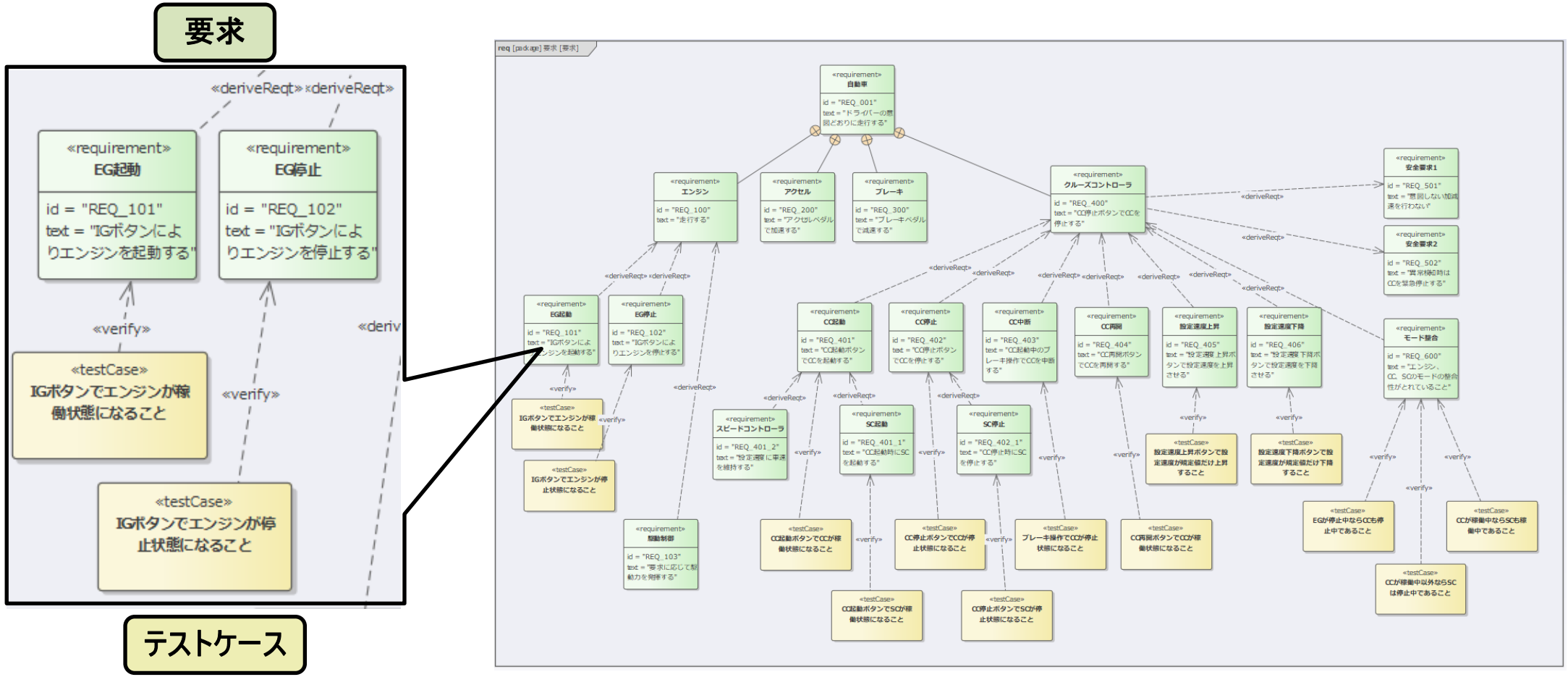
❒ SysMLモデリング：ユースケース図

- システム機能をユースケース図で定義する。
  - システム利用者からみた機能を定義
  - 目的を達成するためにシステムがどのように使われるか？



❒ 要求図

- ユースケースで定義した機能に関する要求に加え、非機能要求、安全要求、制約等も含めて要求図を定義する。
- 要求を満たすかどうかを検証するため、テストケースと関連づけて定義する。
  - テストケースは別の検査用ダイアグラムにおいて検査式として再定義する。



❒ 要求図（検査項目の導出）

■ 要求から容易に導出できるテストケースに加え、システム全体を俯瞰し、常に満たしておくべき条件、起きて欲しくない事象等も考慮して要求／テストケースを定義する。

□ 例）

➡ 以下の状態の組み合わせ以外はモード不整合として許容しない

	EG状態	CC状態	SC状態	備考
モード①	停止中	停止中	停止中	
モード②	稼働中	停止中	停止中	
モード③	稼働中	待機中	停止中	
モード④	稼働中	稼働中	稼働中	

□ 要求

➡ EG状態、CC状態、SC状態のモードの整合性がとれていること

□ 導出したテストケース

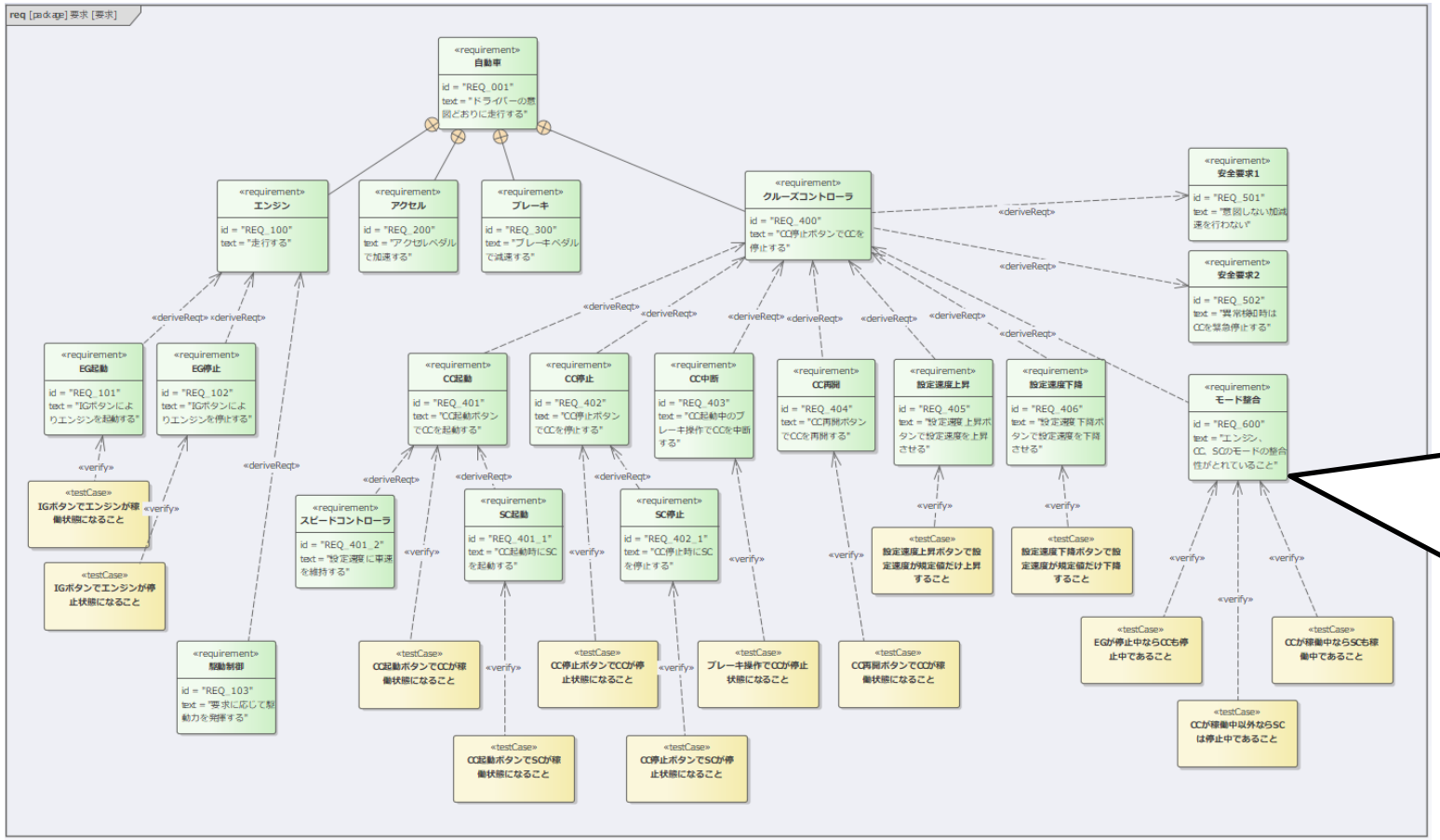
➡ EG状態が停止中ならCC状態も停止中であることを確認

➡ CC状態が稼働中ならSC状態も稼働中であることを確認

➡ CC状態が稼働中以外ならSC状態は停止中であることを確認

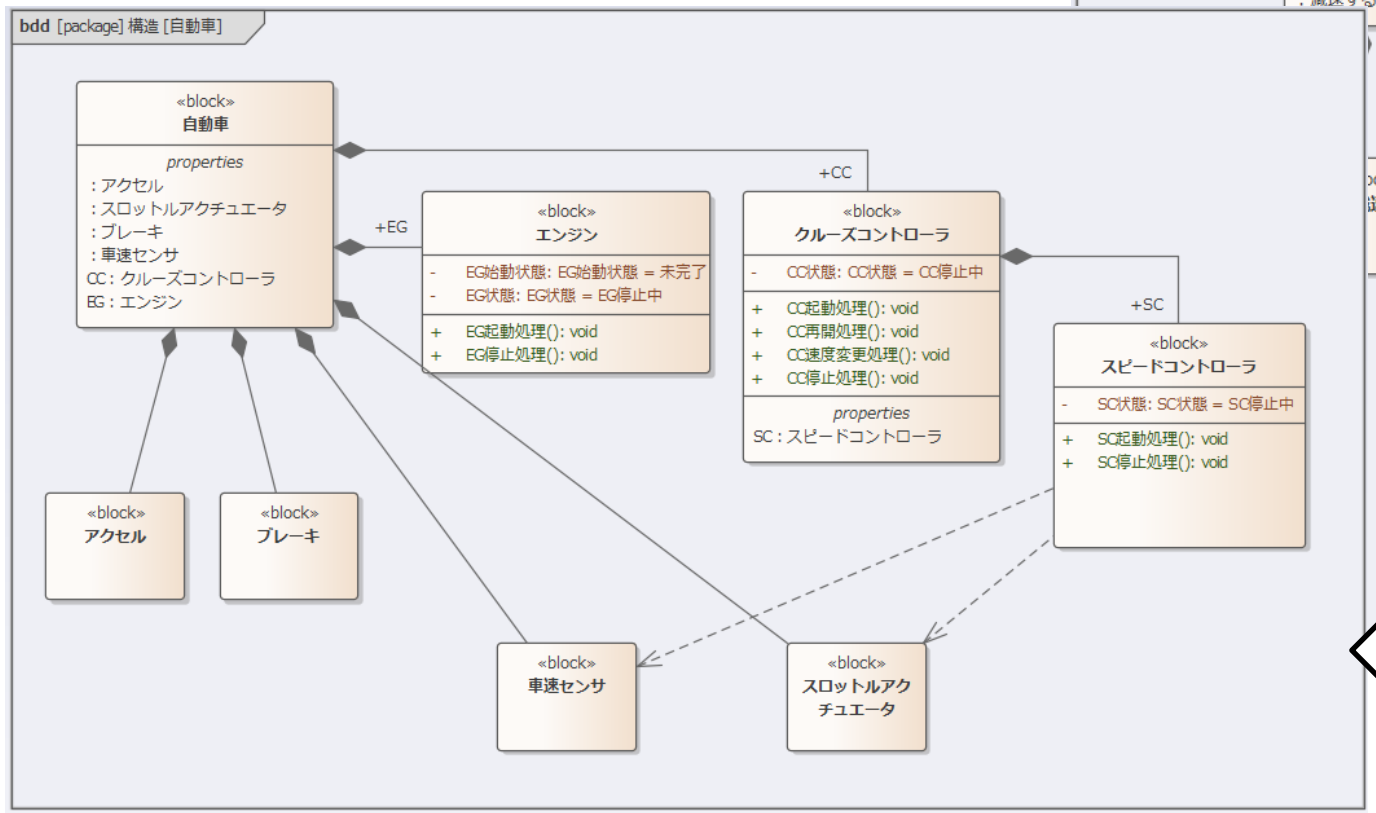
❒ 要求図

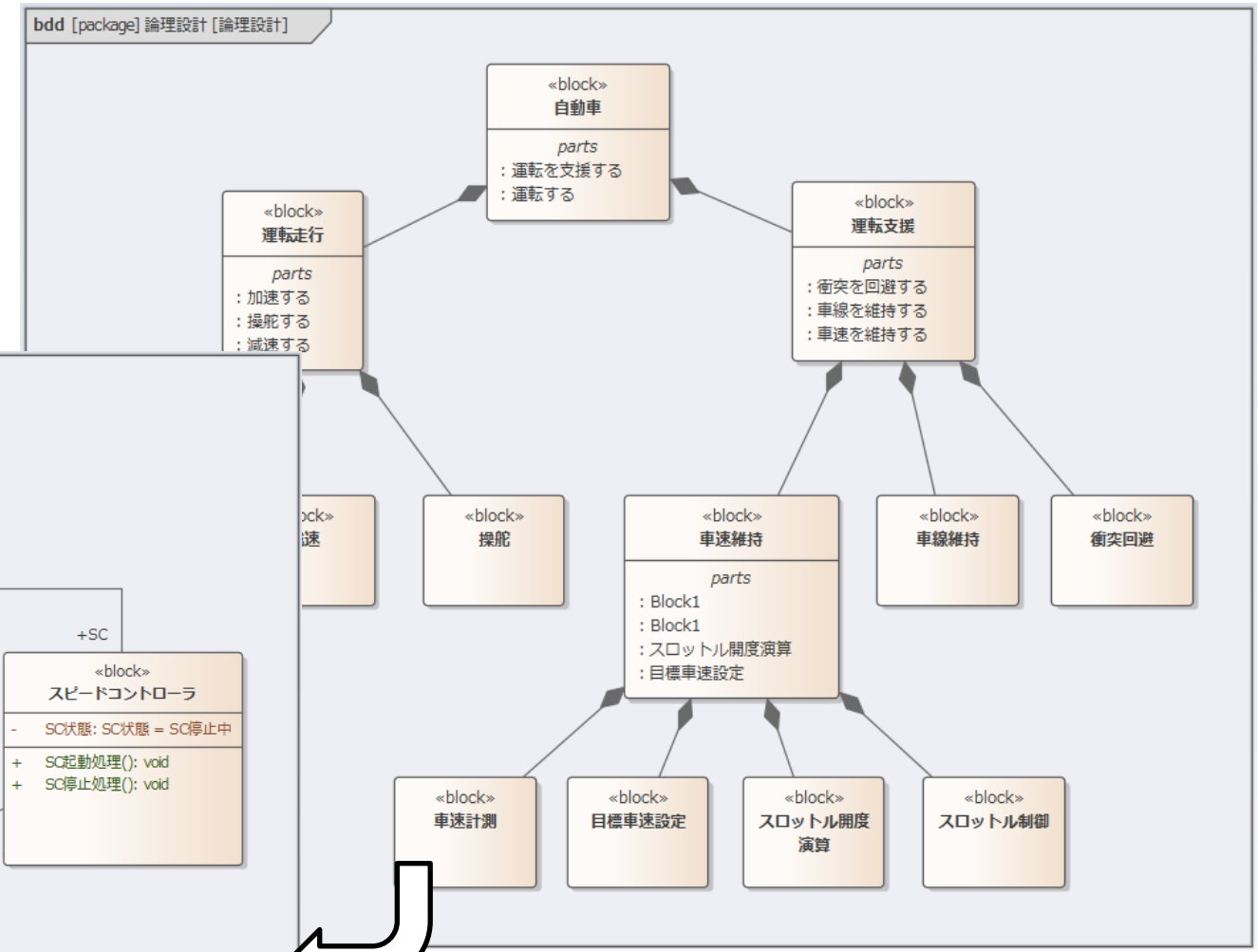
■ 要求／テストケースの定義（追加）



❒ ブロック定義図

- システムの構成要素をブロックで定義する。
  - 論理設計の段階では機能視点でシステムの構成を考える
  - 物理設計では物理的な部品の視点でシステムの構成を考える
    - ➡ 部品への機能の割り当て

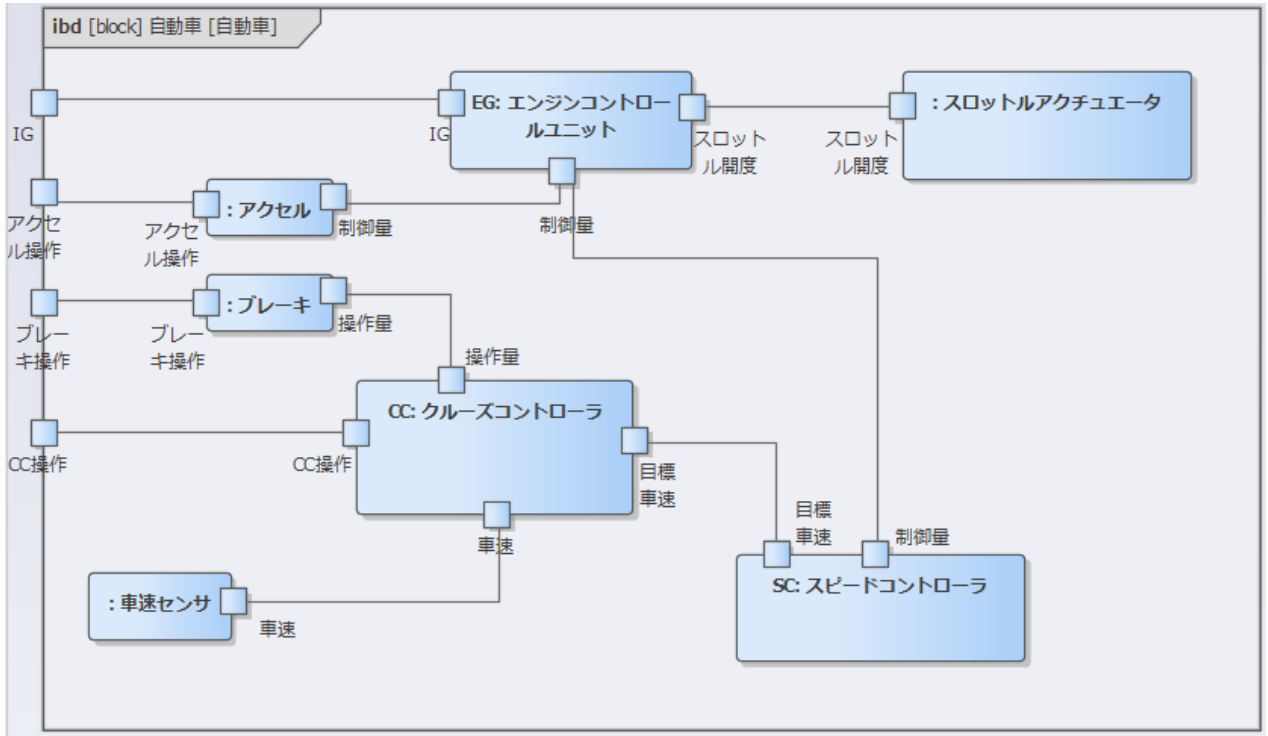




論理設計から物理設計へ

❒ 内部ブロック図

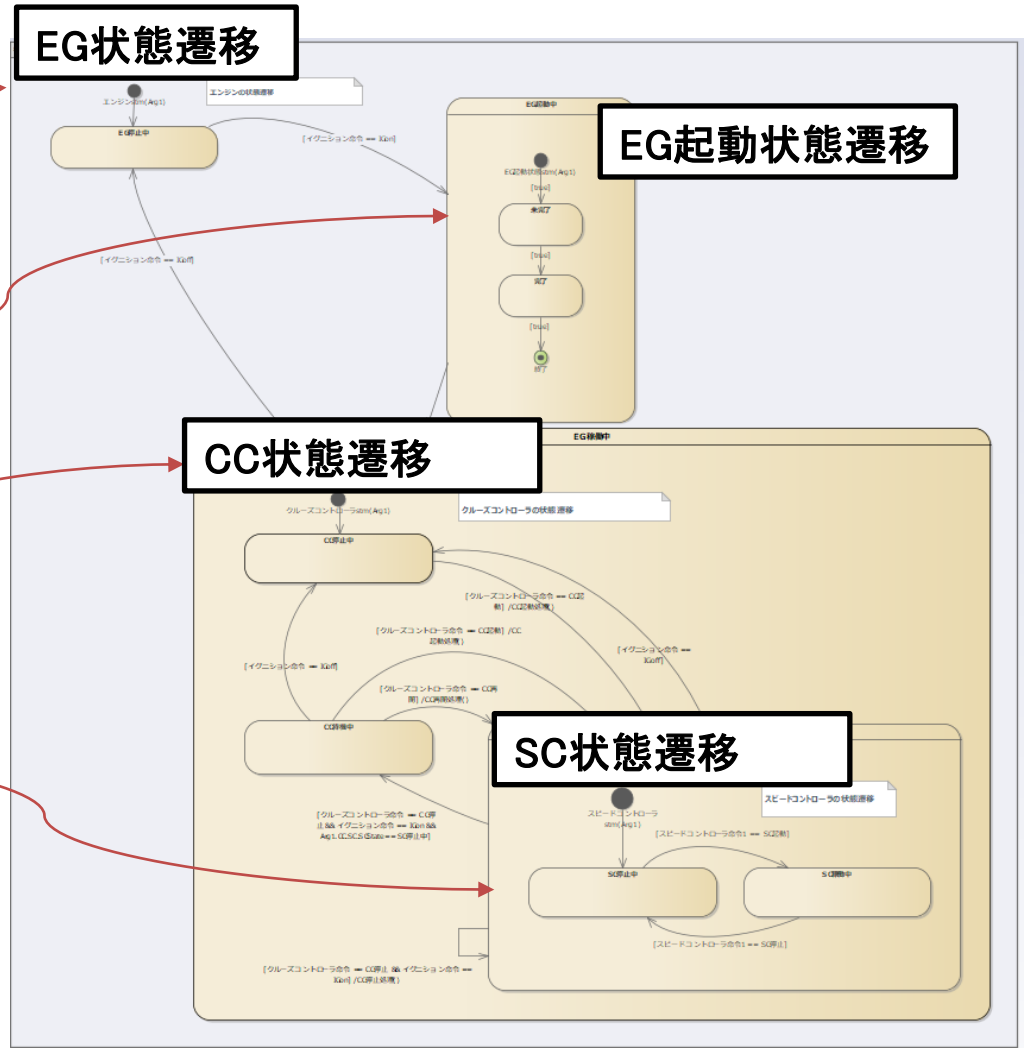
- ブロックの内部構造を内部ブロック図で定義する。
  - パーツ間の接続関係（情報、資源等の入出力関係）を定義する。
    - ➡ 入出力パラメータ間の関係・制約はパラメトリック図で表現するが、本サンプルモデルでの検証目的外であるため割愛する。



ステートマシン図

- システム構成要素の振る舞い（状態遷移）をステートマシンで定義する。
  - とり得る状態、遷移条件
- 必要に応じて階層構造で表現する。

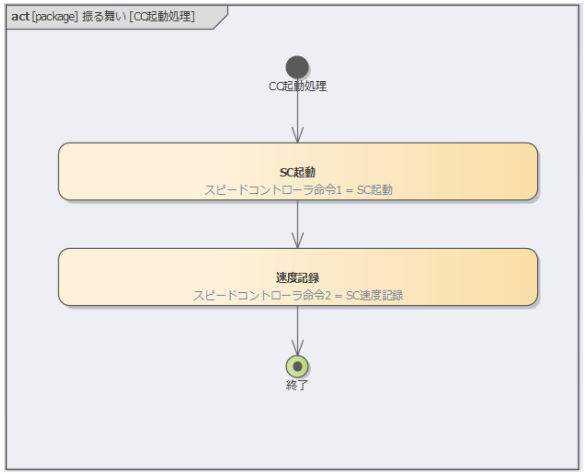
  - エンジンコントロールユニット（↓EG状態）
    - EG停止中
    - EG起動中（↓EG起動状態）
      - 未完了
      - 完了
    - EG稼働中（↓CC状態）
      - CC停止中
      - CC稼働中（↓SC状態）
        - SC停止中
        - SC稼働中
      - CC待機中

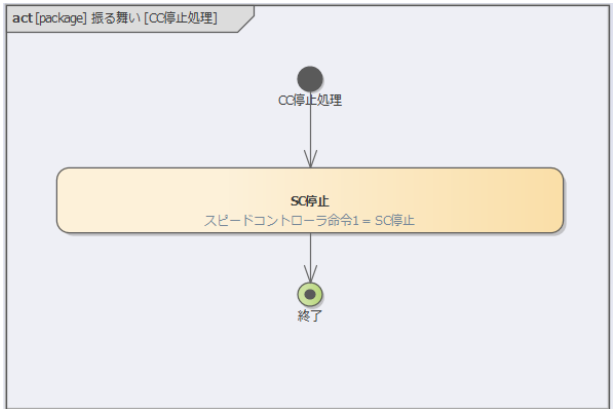


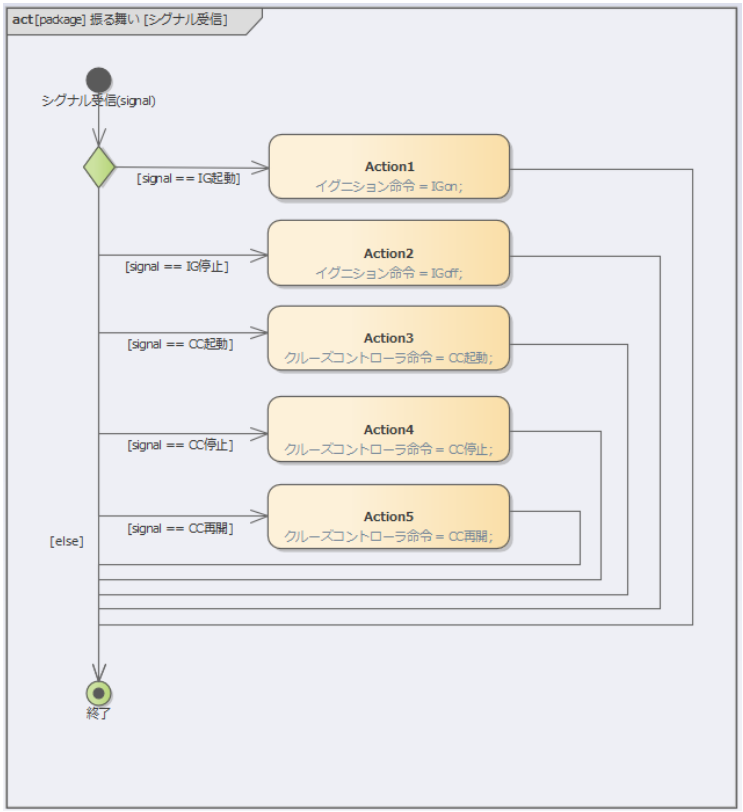
□ アクティビティ図

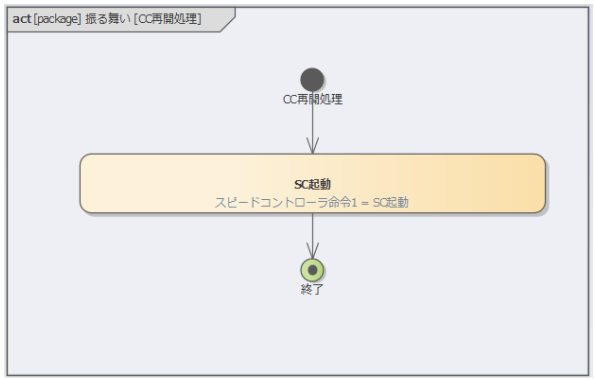
■ イベント発生時に呼び出す振る舞いをアクティビティ図で定義する。

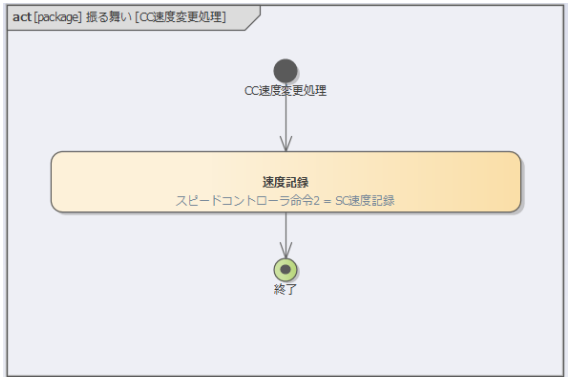
□ 処理の順序、分岐・合流、同期・非同期等、並行プロセス、その他の複雑な表現









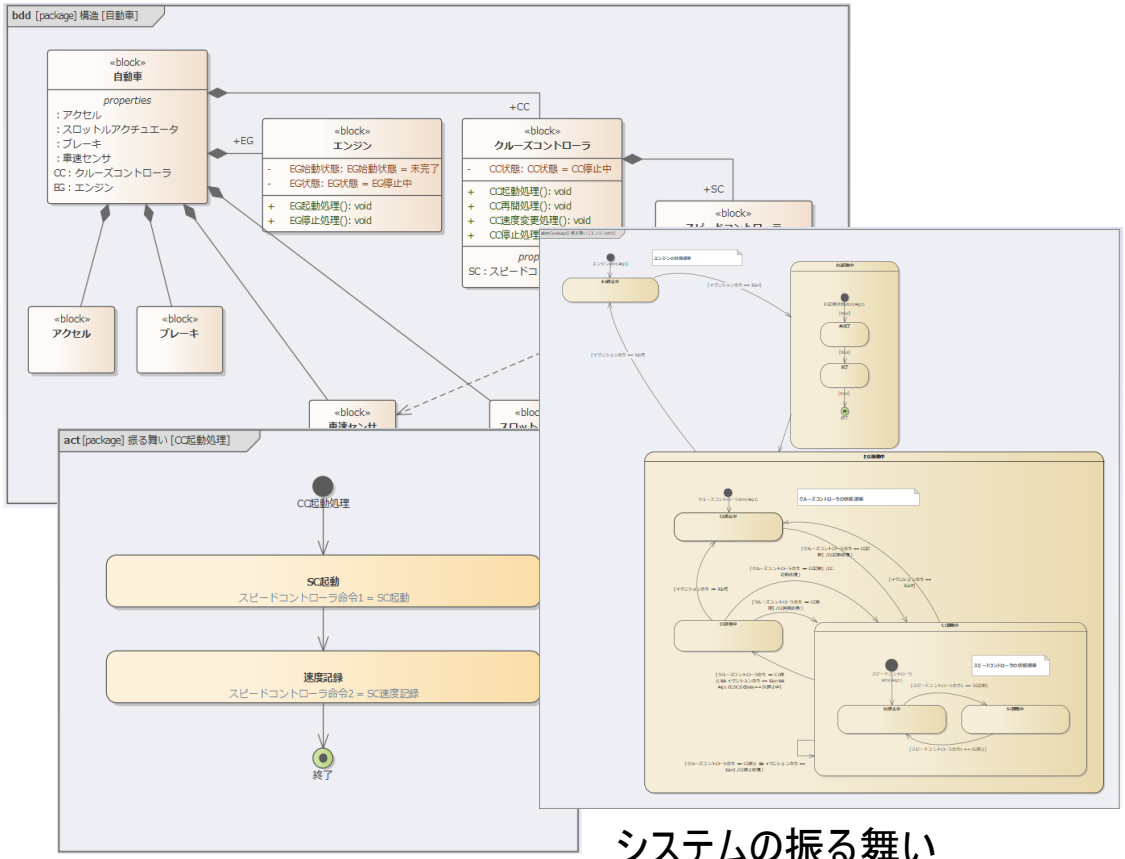


❑SysMLモデルを検証のためのモデルに拡張する

■ システム設計モデル（実装対象）に加えて検証のための条件モデルを作成する

【システム設計モデル】

システムの構成要素

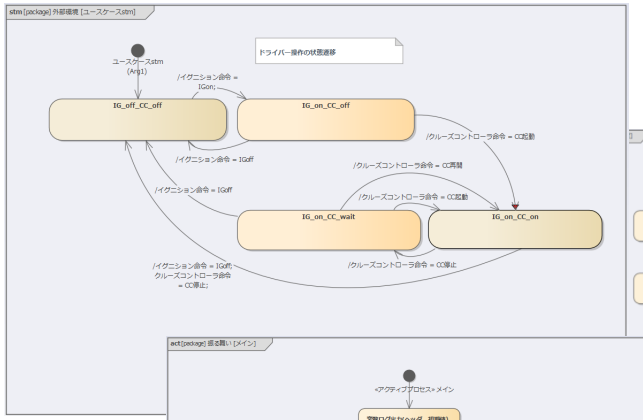


システムの振る舞い

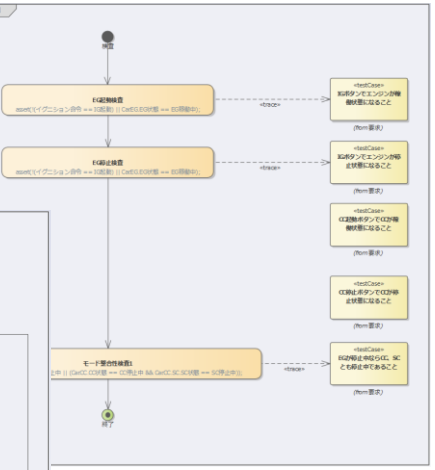
＋

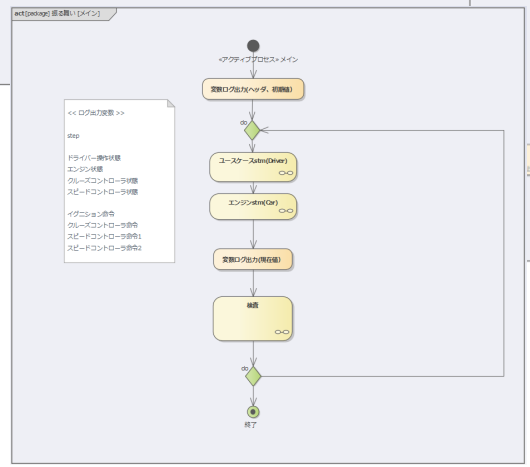
【検証条件モデル】

外部環境モデル（ドライバーの振る舞いモデル）



制約違反の検査式

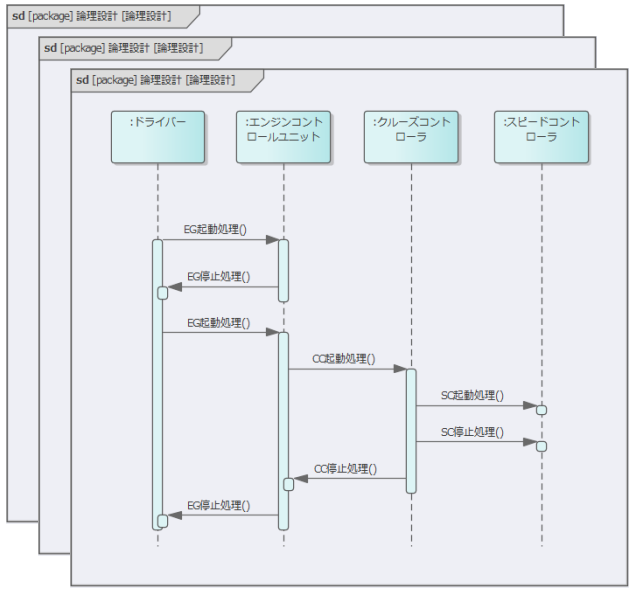




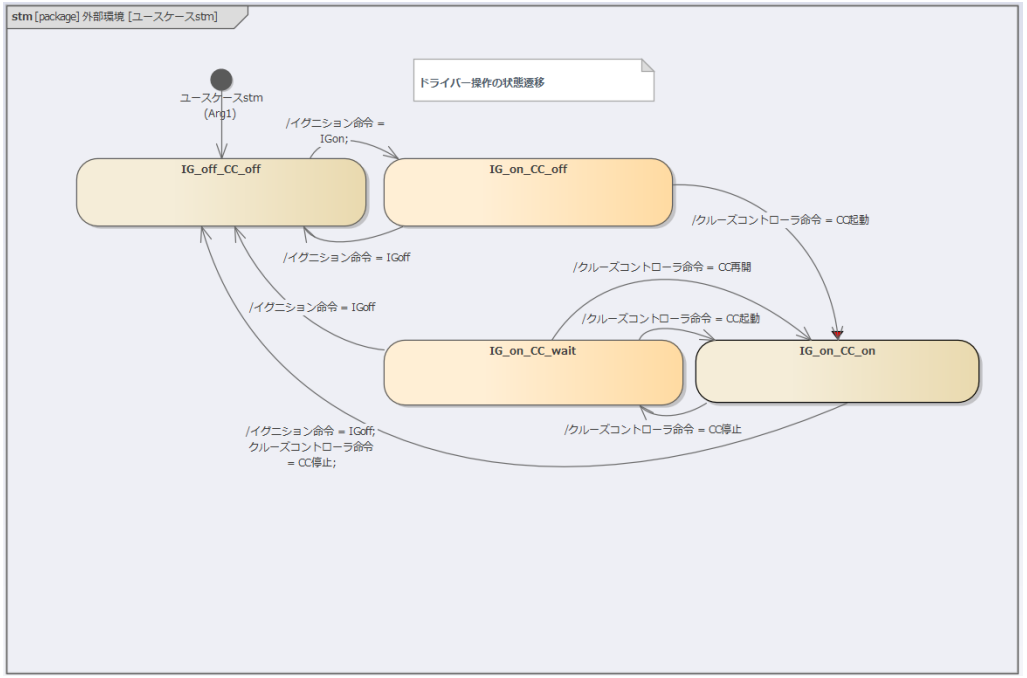
全体制御メインプロセス

❒検証条件モデル：外部環境モデル

- システムの振る舞い検証を行うため、システムの外部環境（ドライバー）の振る舞いモデルを表現する。
  - 一般的な検証手法では複数の振る舞いパターンを定義する
    - ユースケース図、シーケンス図による検証
    - シナリオベースのシミュレーションによる動的な検証
  - モデル検査ではシナリオ定義の代わりにドライバーの振る舞いをステートマシンで表現する。
    - 遷移条件を非決定とすることで、モデル検査時にあらゆる遷移やタイミングを網羅的に再現できる。

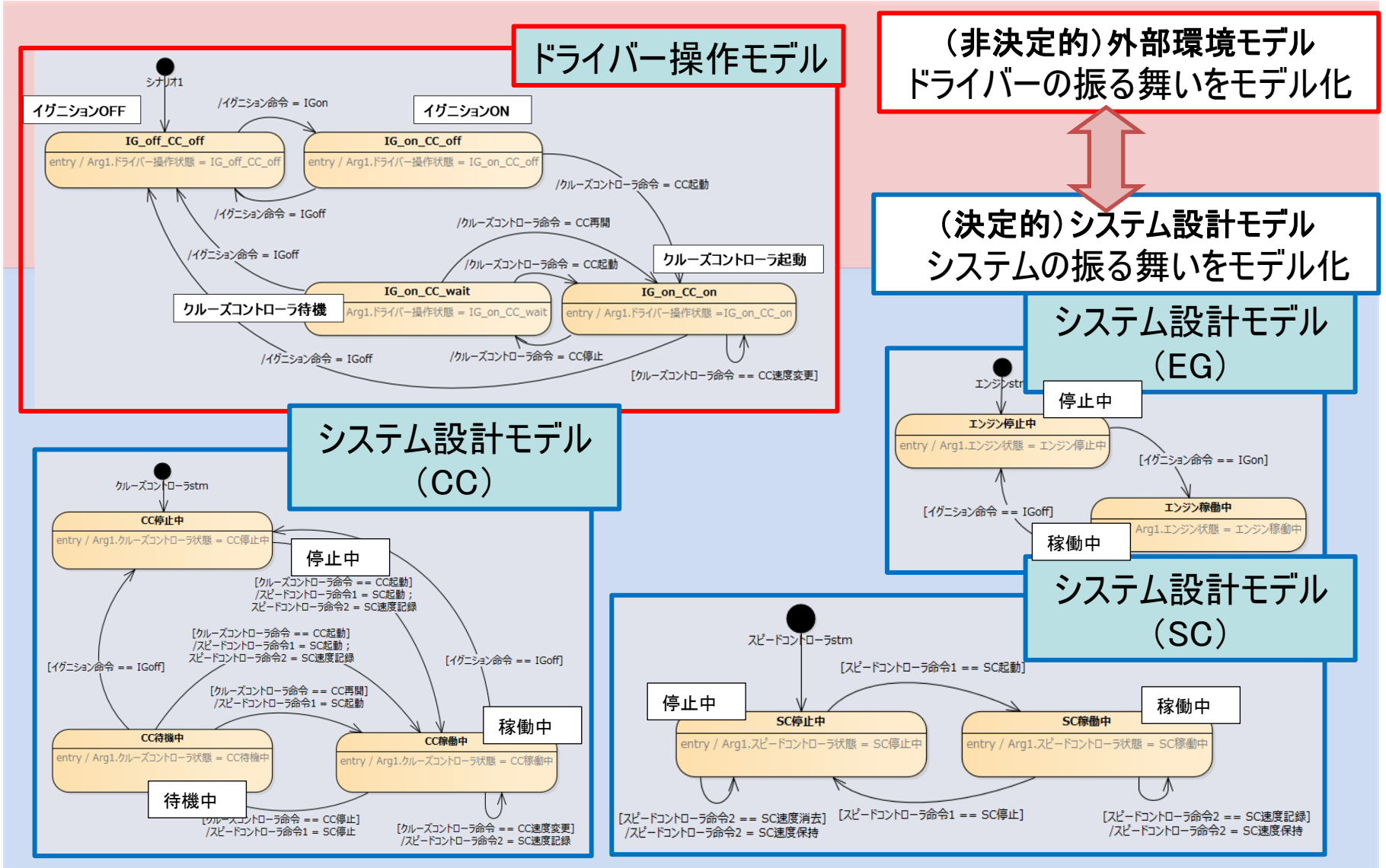


複数のドライバー操作シナリオ



1つのドライバーの操作モデル

❑ （決定的）システム設計モデルと（非決定的）外部環境モデル

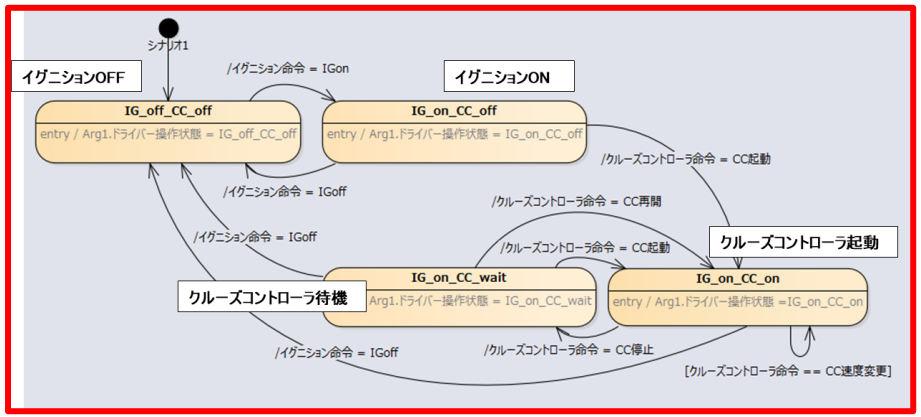


遷移条件（トリガー/ガード）を未定義にすることで、非決定的遷移するモデルとなり、モデル検査により振る舞いパターンが網羅的に再現される。

❒ 外部環境（操作シナリオ）のモデル化パターン

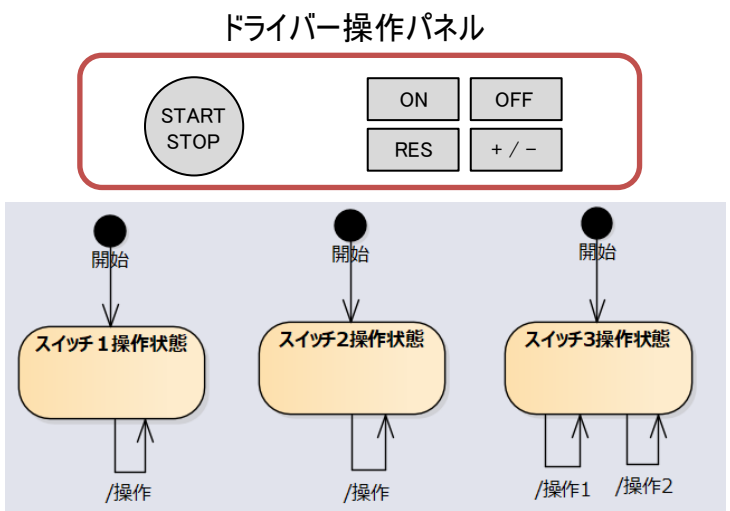
ドライバー操作モデル例 ①

- 現実に起こり得る操作を考慮したモデル
  - 意味のある操作のみを表現
  - ・・・
  - ※あまり作り込むと網羅性が損なわれる



ドライバー操作モデル例 ②

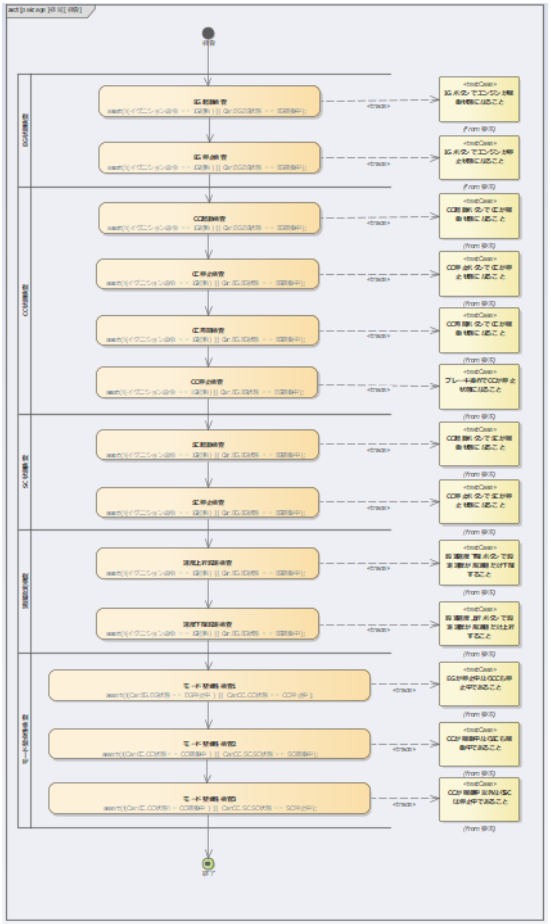
- スッチ毎に独立して非決定的に操作するモデル
  - 既にONの状態でONを押す（連打も含む）
  - 複数のスッチを同時に押す
  - ・・・
  - ※想定外の振る舞いの網羅性は高い
  - ※現実に起こりえない振る舞いによる不具合が大量に検出され、検証結果の分析が大変になることも

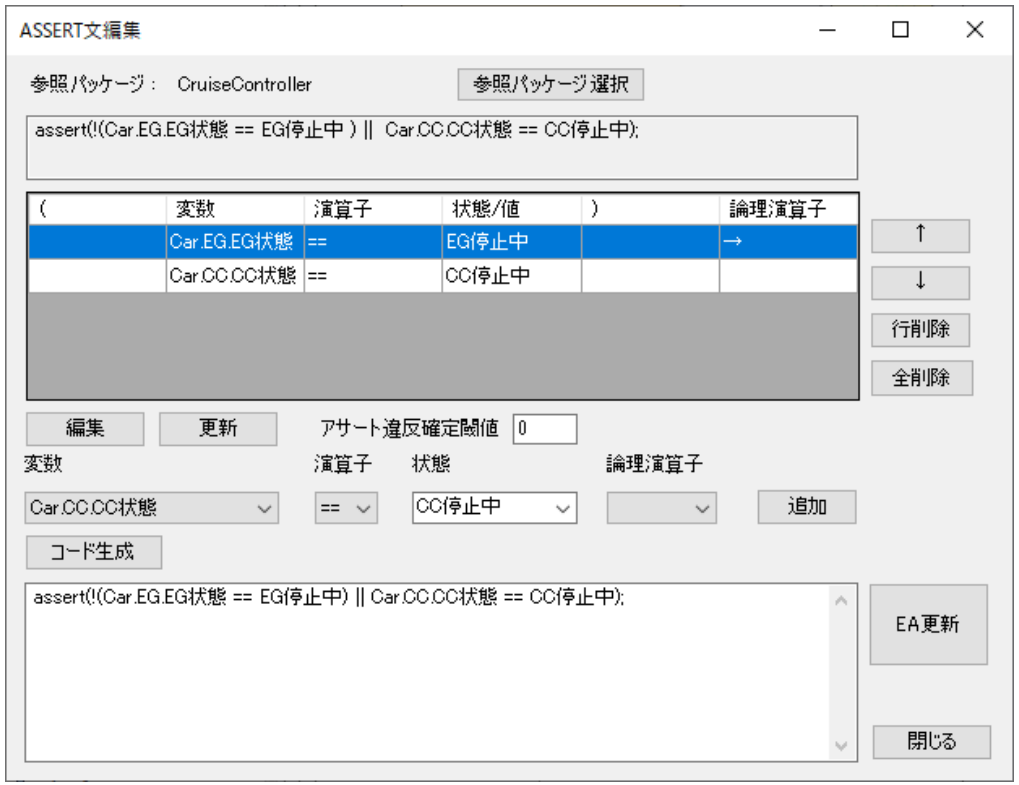


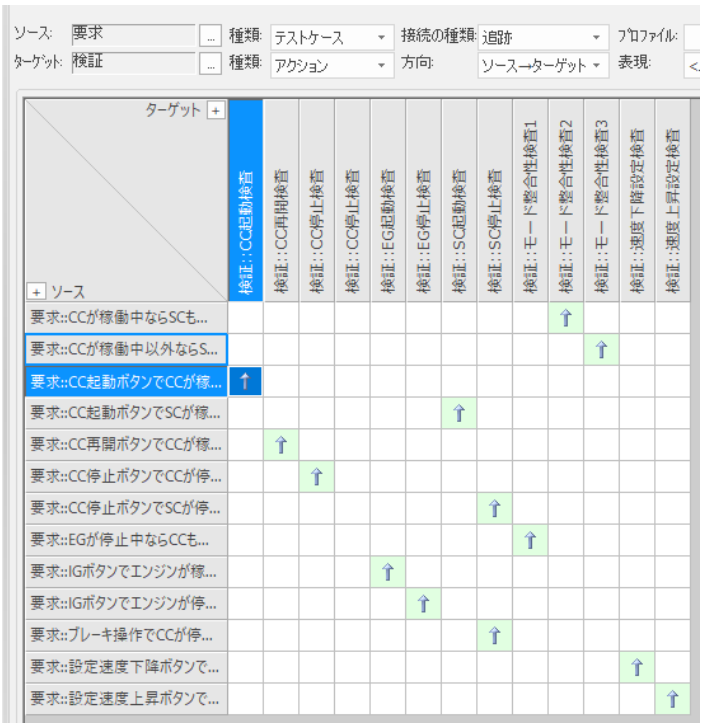
目的に応じてどんな振る舞いを網羅したいのかを考える

□アクティビティ図（検証用モデル：検査）

- 要求・テストケースの条件を満たすかの検査式をASSERTで表現する。
  - □ 例）EGが停止中ならCCも停止中であること　⇒　Car.EG.EG状態 == EG停止中 → Car.CC.CC状態 == CC停止中
- 全てのテストケースに対して検査式が定義されているかのトレーサビリティを取る。

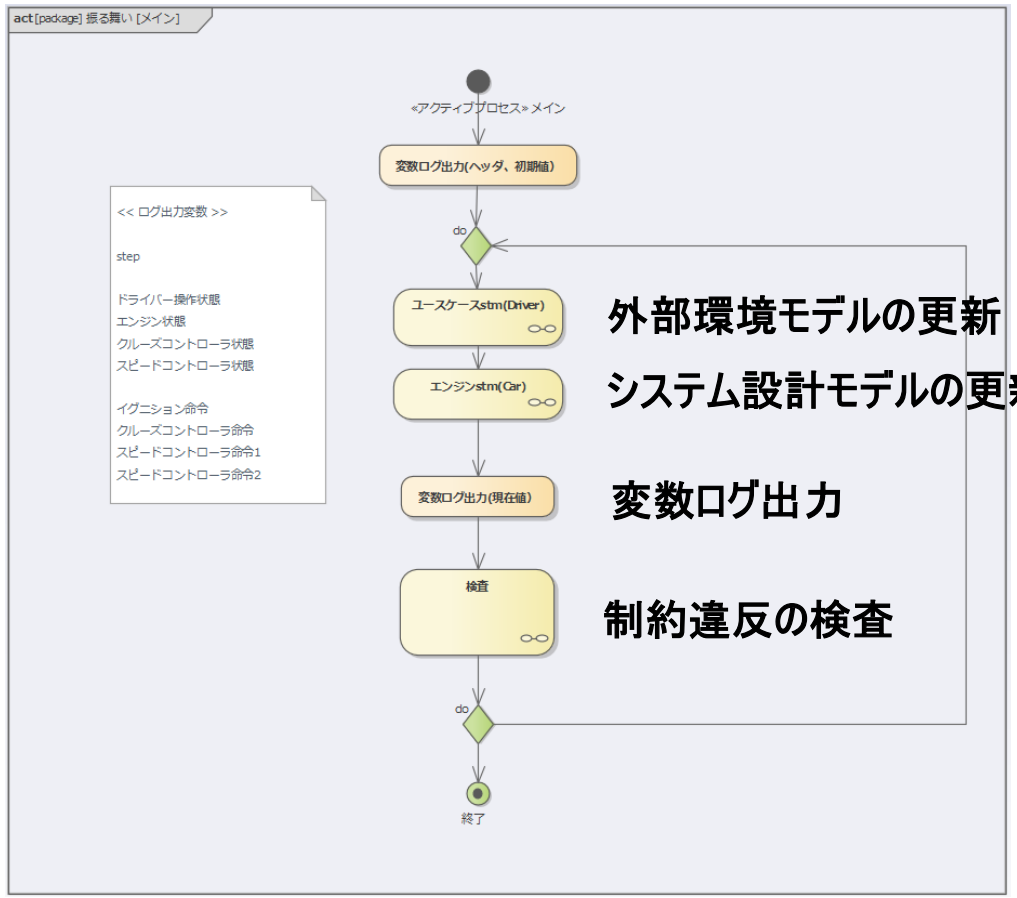




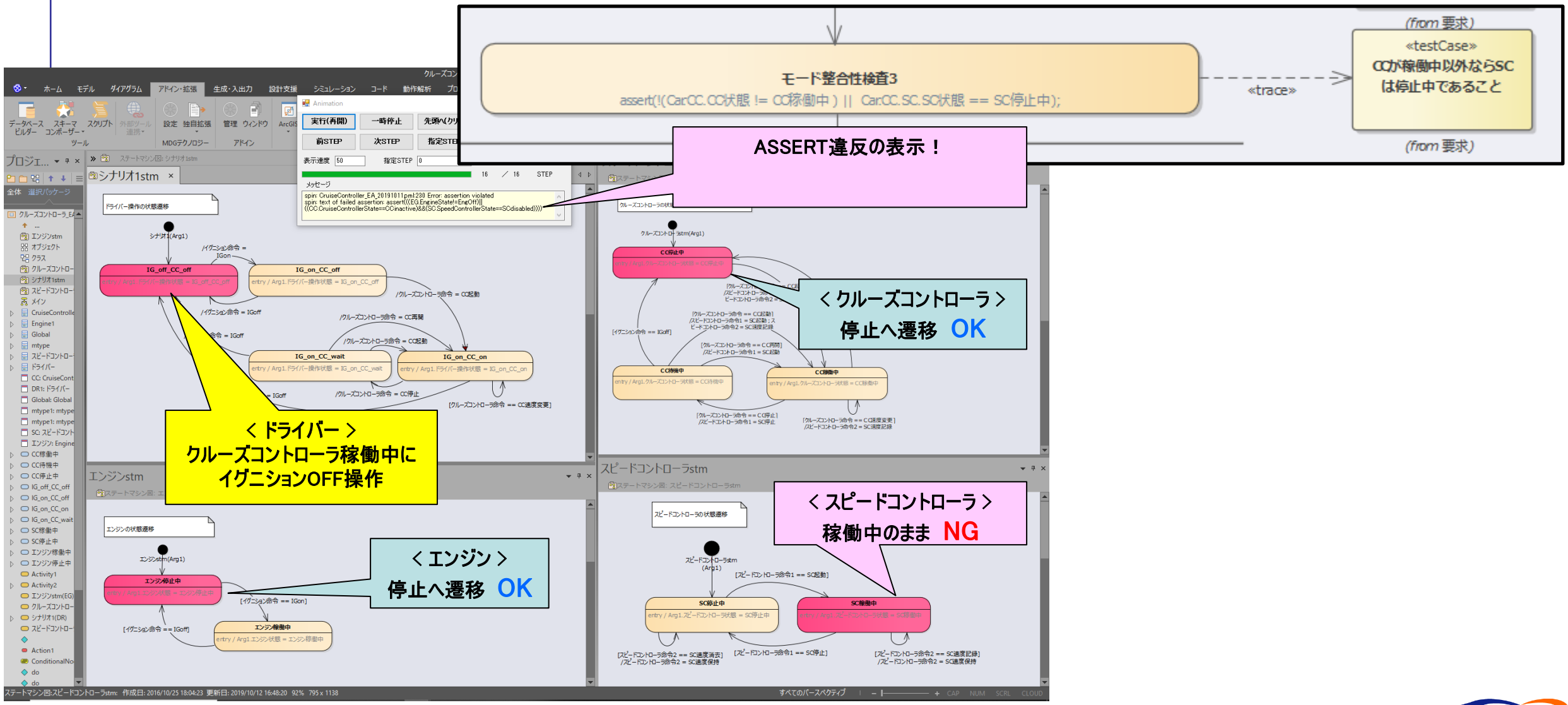


□アクティビティ図（検証用モデル：メインプロセス）

- 外部環境モデル、制御モデルの状態更新、変数のログ出力、システム状態が制約を満たしているかの検査を行う検証メインプロセスをアクティビティ図で定義する。

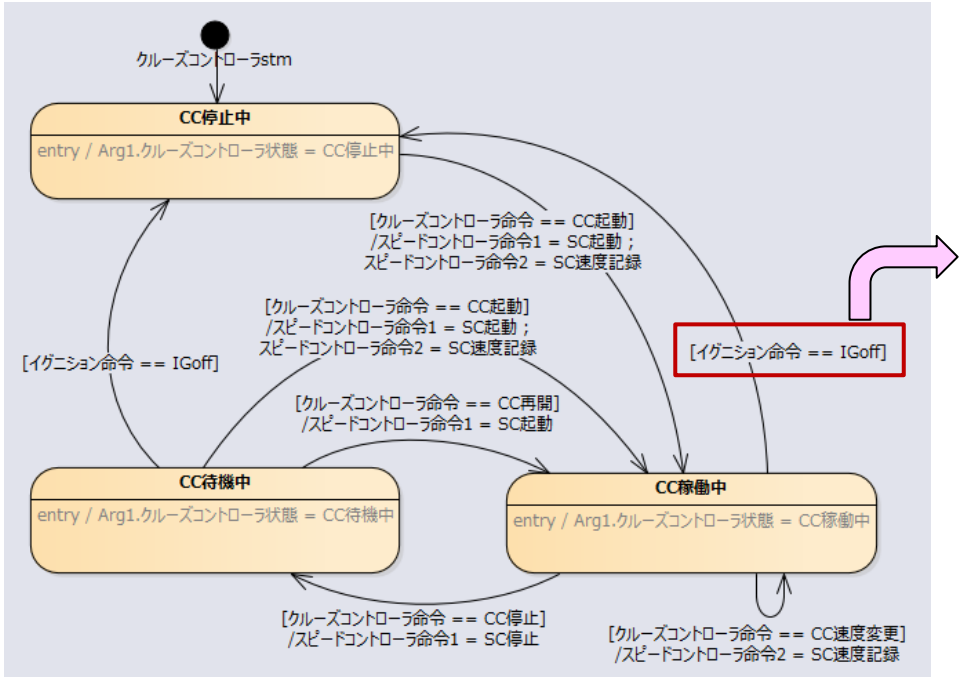


モデル検査結果（アニメーションデモ）



モデル検査結果と対策

- 不具合の原因
  - クルーズコントローラが「IGoff」命令を受け取って「停止中」に遷移する際、スピードコントローラに対して「SC停止」命令を発行する処理が抜けていた。



対策
命令の追加

[イグニション命令 == IGoff]
/ スピードコントローラ命令1 = SC停止

テスト段階で顕在化する（あるいはテスト段階でも発覚しにくい）不具合が、モデル検査（状態遷移の網羅検査）により上流設計段階で検出できる。

❒まとめ

- Enterprise Architect ＋DynaSpecの導入で
  - 設計者がシステム全体を俯瞰するSysMLモデリングを実施し
  - 設計者が自らモデルを動かしながら検証することで

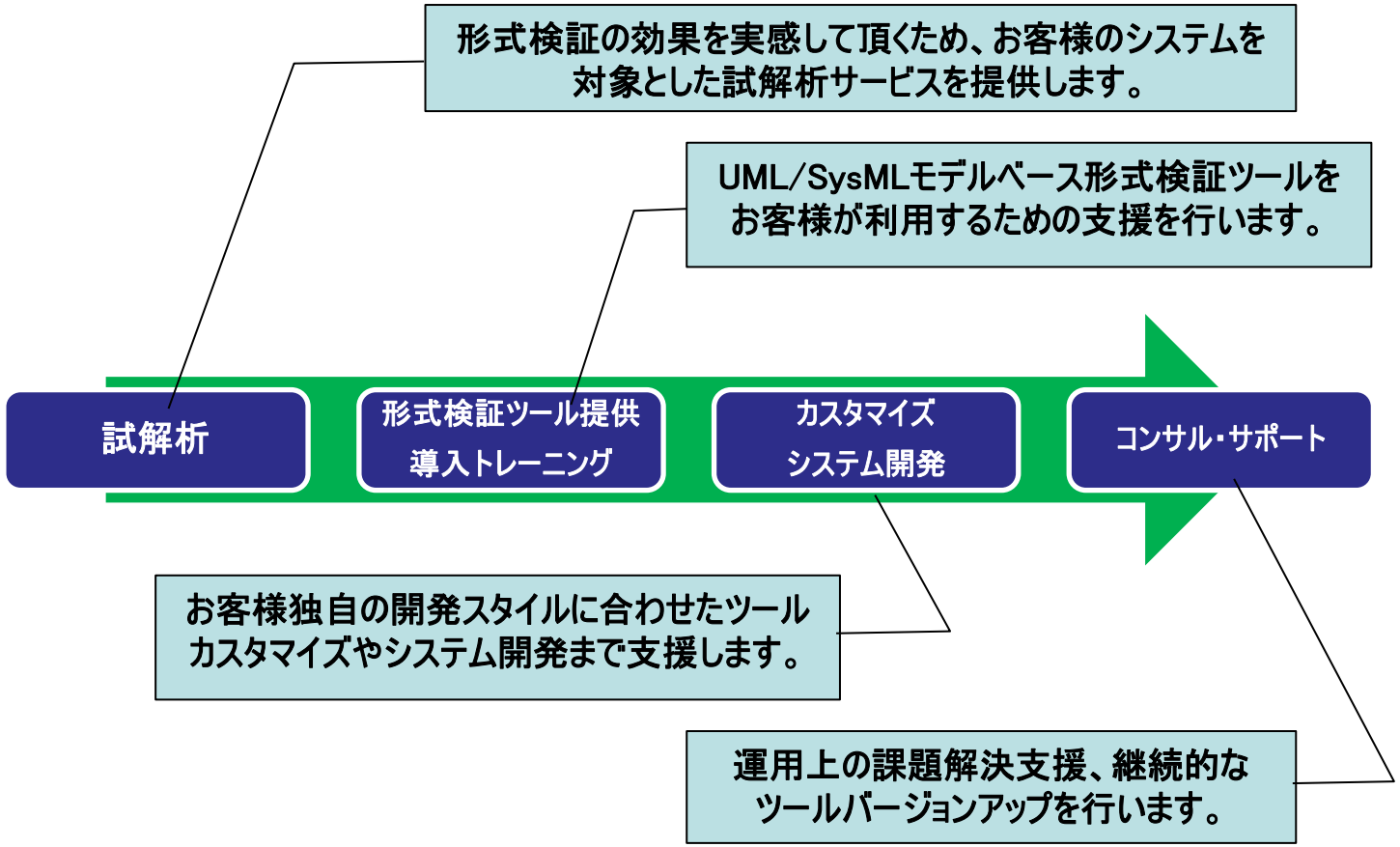
  設計品質、開発効率を向上できることを紹介しました。

- サンプルモデルとデモを通し
  - 対象システムをどのようにモデル化するか
  - 起こり得る振る舞いを網羅的に再現させるために対象システムの外部環境をどのようモデル化するか
  - 対象システムが守るべき制約をどのように導出し、制約違反をどのように検出するか

  を示し、モデル検査で発見困難な不具合をいかにして自動的に検出できるのかを紹介しました。

❒ KKEの形式検証サービス

■ 形式検証の導入フェーズに応じたサービスを提供します。



KKEは形式検証の導入から運用まで一貫したサポートを行います。

□～モデルベース形式検証ツールDynaSpec「個別」無料体験セミナーについて

■12月6日から、**期間限定で**弊社モデルベース形式検証ツールDynaSpecのオンライン無料個別体験セミナーを実施致します。

- □お申込みは**12月1日**から開始致します。
- □https://kke.lmsg.jp/v2/seminar/11672/jWW39c25

【オンライン開催】
モデルベース形式検証ツール『DynaSpec』
「個別」無料体験セミナー

～ご参加の方皆様に「DynaSpec」
無償体験版をご提供します！～

12月1日（水）8:00 に受付開始