D-CaseとSysMLとの連携によるディペンダブルなシステムの開発

デモ資料

目次

1. 概要
2. D-CaseとSysMLとの連携による
   ディペンダブルなシステムの開発手法
3. 本手法の適用例
   ～ ISO26262に準拠した車載システムの開発 ～

概要

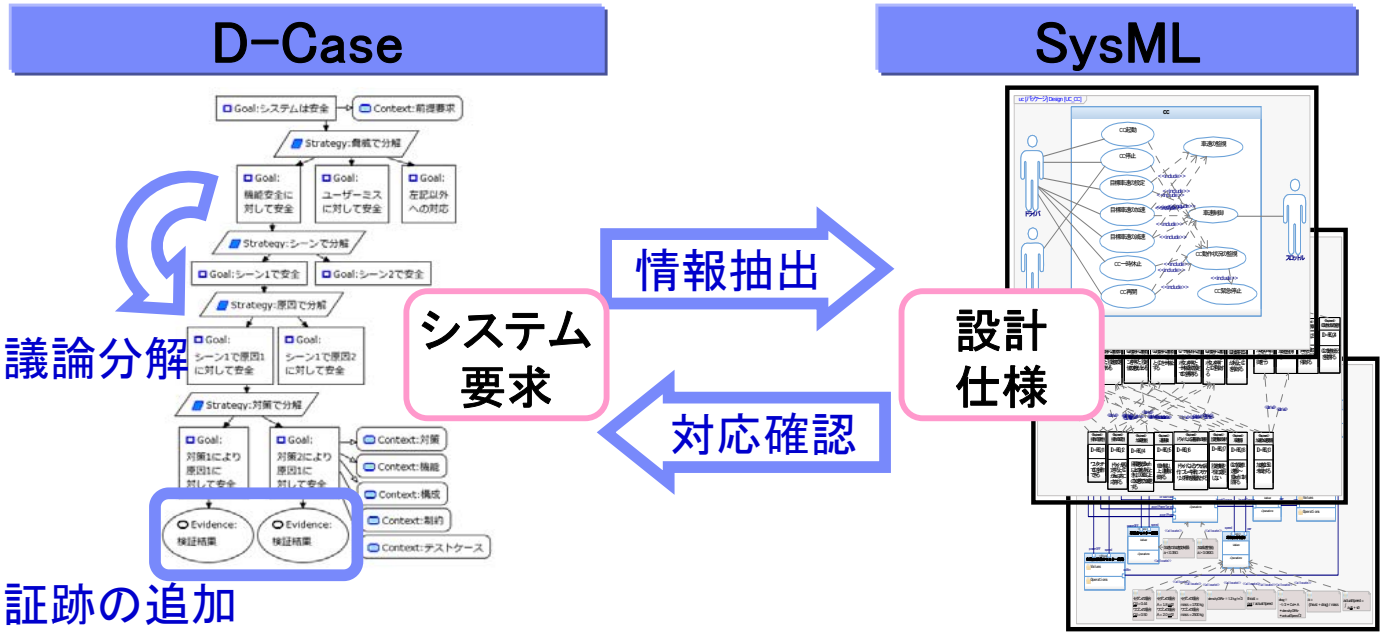
目的とアプローチ

目的

ディペンダビリティを備えるシステムの開発を実現

アプローチ

開発の上流工程から下流工程に至るまでの、
D-CaseとSysMLモデルとの一貫した連携



成果

D-CaseとSysMLモデルの連携方法のガイドとサンプルの提供

ガイド内容

- D-CaseとSysMLのモデリングフロー
- D-Caseノードの記述スタイル
- D-CaseとSysMLモデルの情報連携

対象のシステム

- 自動車の機能安全に関する国際規格ISO26262に準拠する車載システムへの適用

対象の読者

- ディペンダブルなシステムの開発を検討中の開発者

D-CaseとSysMLとの連携による
ディペンダブルなシステムの開発手法

ディペンダブルなシステムの開発に必要な特性

開発フェーズ	必要な特性
要件定義	システム要求が ディペンダビリティを満足している
システム設計	設計仕様が 要求を正確に反映している
ソフトウェア開発 ハードウェア開発	
システム検証	検証結果が ディペンダビリティを満足すること を説明可能である

ディペンダブルなシステムの開発における課題

1 システム要求を過不足なく導出することができない

ディペンダビリティを阻害するすべての要因を除去することが可能な要求を導出する必要がある

2 設計仕様を正確に導出することができない

- ディペンダビリティから導出した要求に含まれる設計情報を活用して、システム設計を行う必要がある
- 導出した設計仕様を、ディペンダビリティから導出した要求と照合し、過不足なきことを確認する必要がある

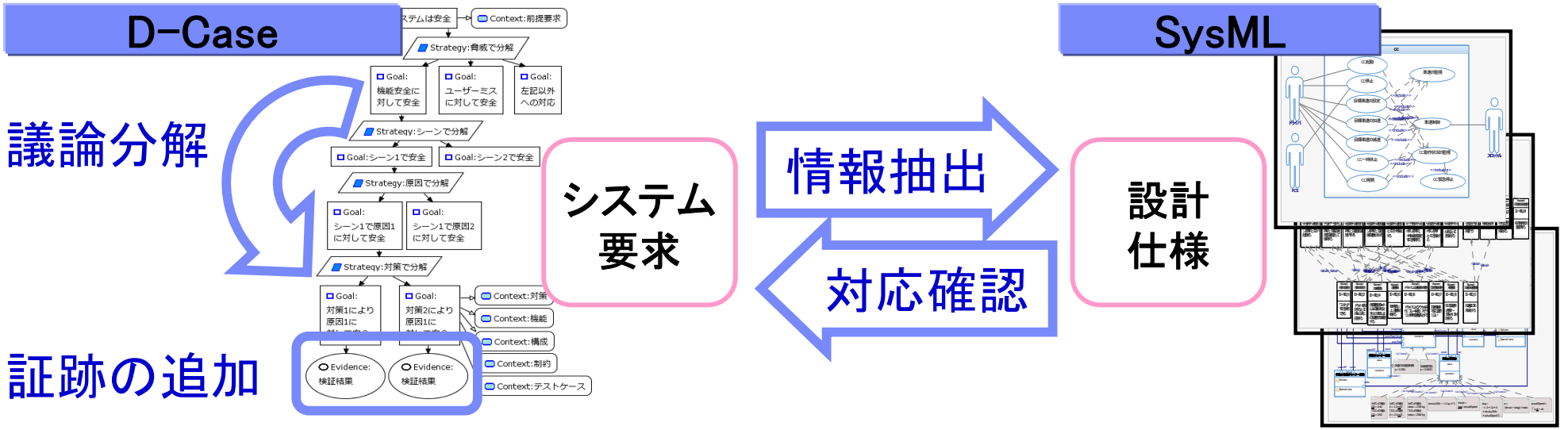
3 検証結果が要求や設計仕様をどのように満足するか説明することができない

要求や設計仕様との関連付けにより、検証結果の位置付けを明確化する必要がある

D-Caseによる課題解決

開発対象のディペンダビリティを開発の上流工程から下流工程に至るまで一貫して実現するためD-Caseを活用

1	ディペンダビリティを阻害するすべての要因を除去することが可能な要求を導出する	D-Caseによる議論分解
2	ディペンダビリティから導出した要求に含まれる設計情報の活用によりシステム設計を行う	D-Caseからの情報抽出
	導出した設計仕様を、ディペンダビリティから導出した要求と照合し、過不足なきことを確認する	D-Caseと仕様との対応確認
3	検証結果が要求や設計仕様を満足することを説明する	D-Case上での証跡の追加



アプローチ ～ D-Caseによる議論分解

1 システム要求を過不足なく導出

ディペンダビリティを阻害するすべての要因を抽出し、
その対策をシステム要求として整理する

D-Caseの議論分解パターン

ディペンダビリティへの脅威の明確化

↓

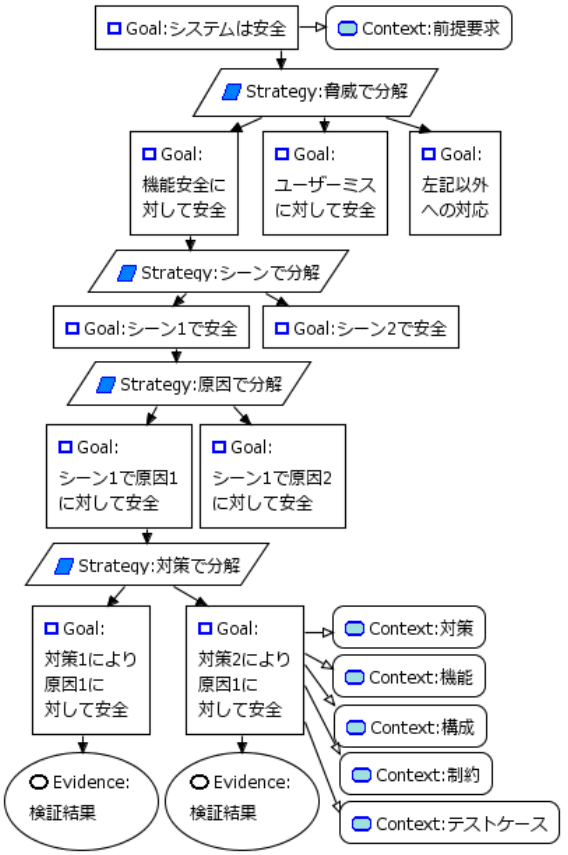
脅威の発生シーンの明確化

↓

原因の明確化

↓

対策の明確化

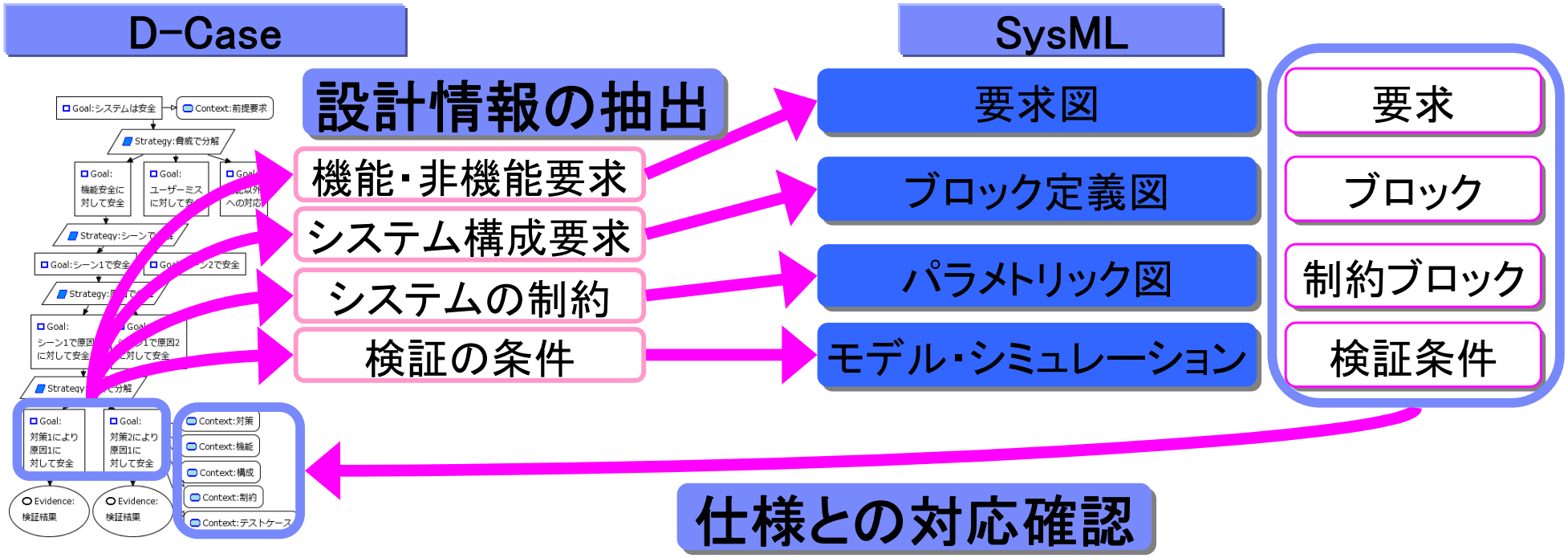


アプローチ ～ D-Caseからの情報抽出と対応確認

2 設計仕様を正確に導出

ディペンダビリティから導出した
要求に含まれる設計情報を抽出し
システム設計を行う

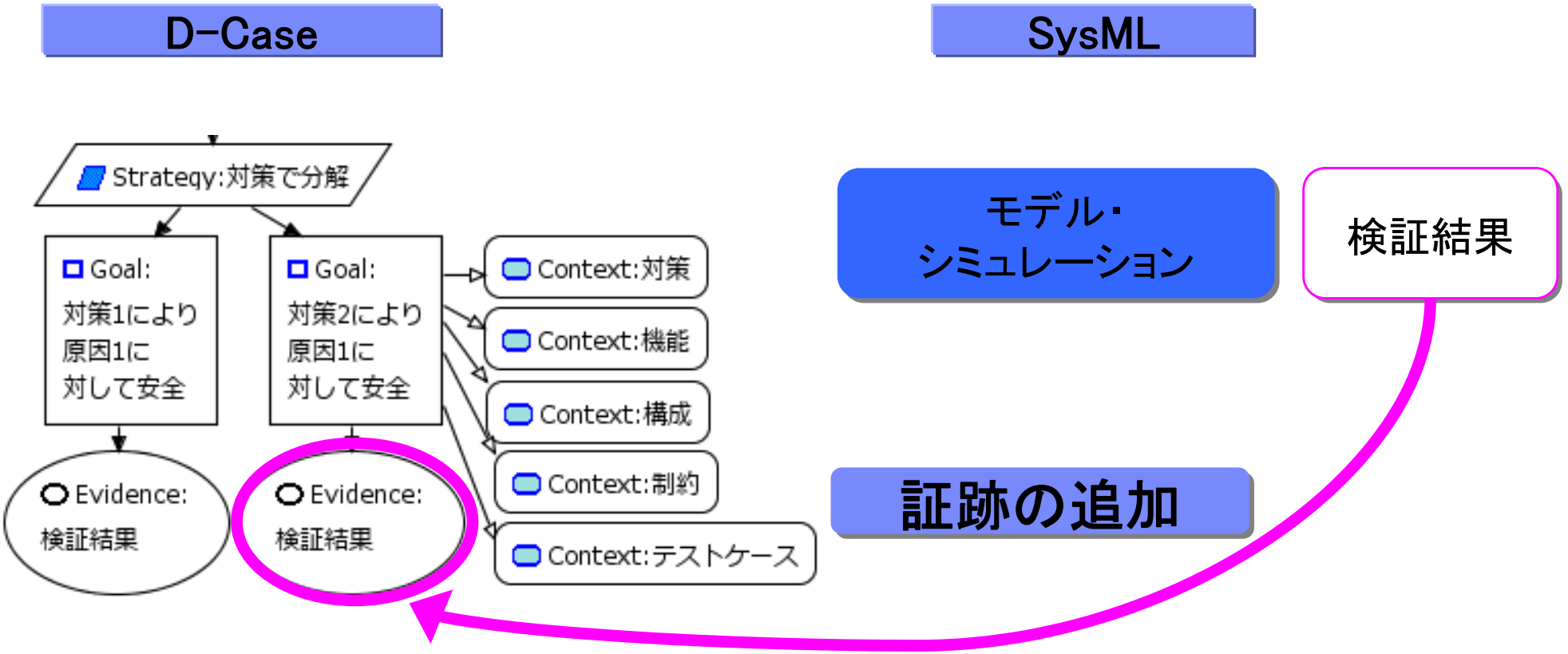
導出した設計仕様を
ディペンダビリティに基づく要求と
照合し、過不足なきことを確認する



アプローチ ～ D-Caseへの証跡の追加

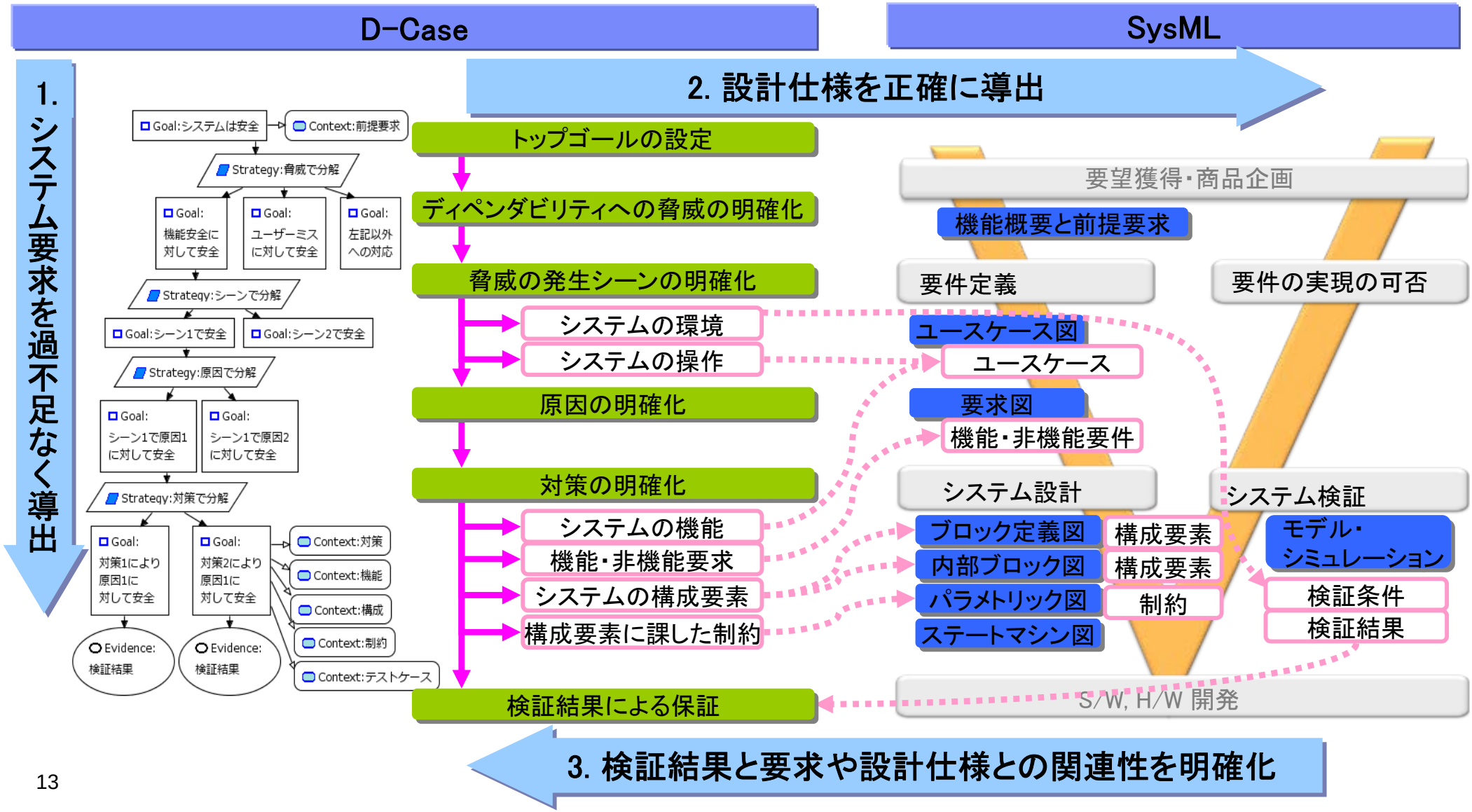
3 検証結果と要求や設計仕様との関連性を明確化

検証結果をエビデンスとしてゴールに関連付け、
関連する要求や設計仕様との整合性を確認する



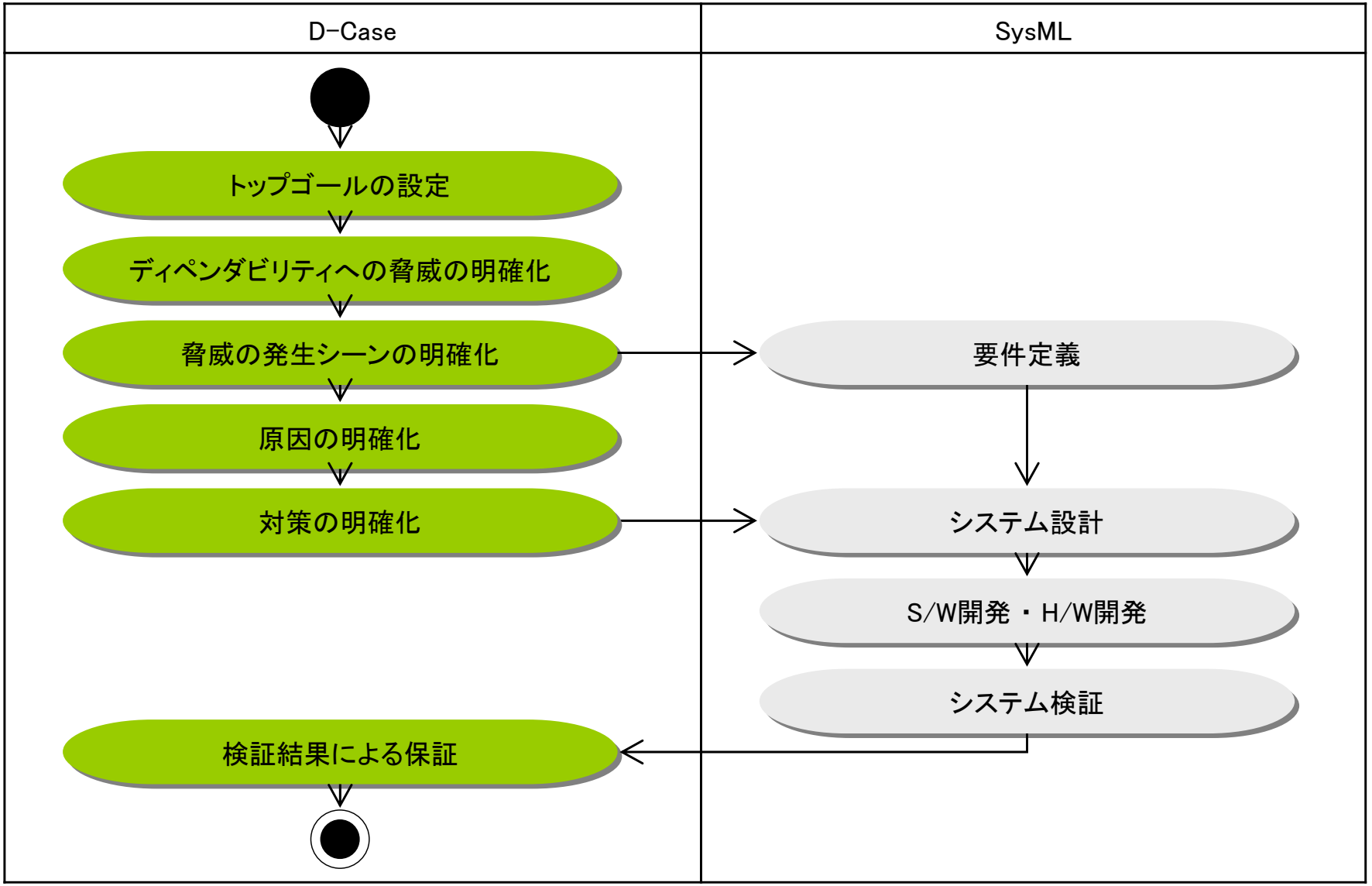
ディペンダブルなシステムの開発手法 ～ 全体像

D-CaseとSysMLモデリングとの連携によるシステム開発



ディペンダブルなシステムの開発フロー

D-CaseとSysMLとの連携によりディペンダブルなシステムを開発



フロー詳細 ～ トップゴールの設定 と ディペンダビリティへの脅威の明確化

ディペンダビリティの観点からトップゴールを設定し、脅威を大別

D-Case

トップゴールの設定

ディペンダビリティの観点に基づき
トップゴールと前提を設定

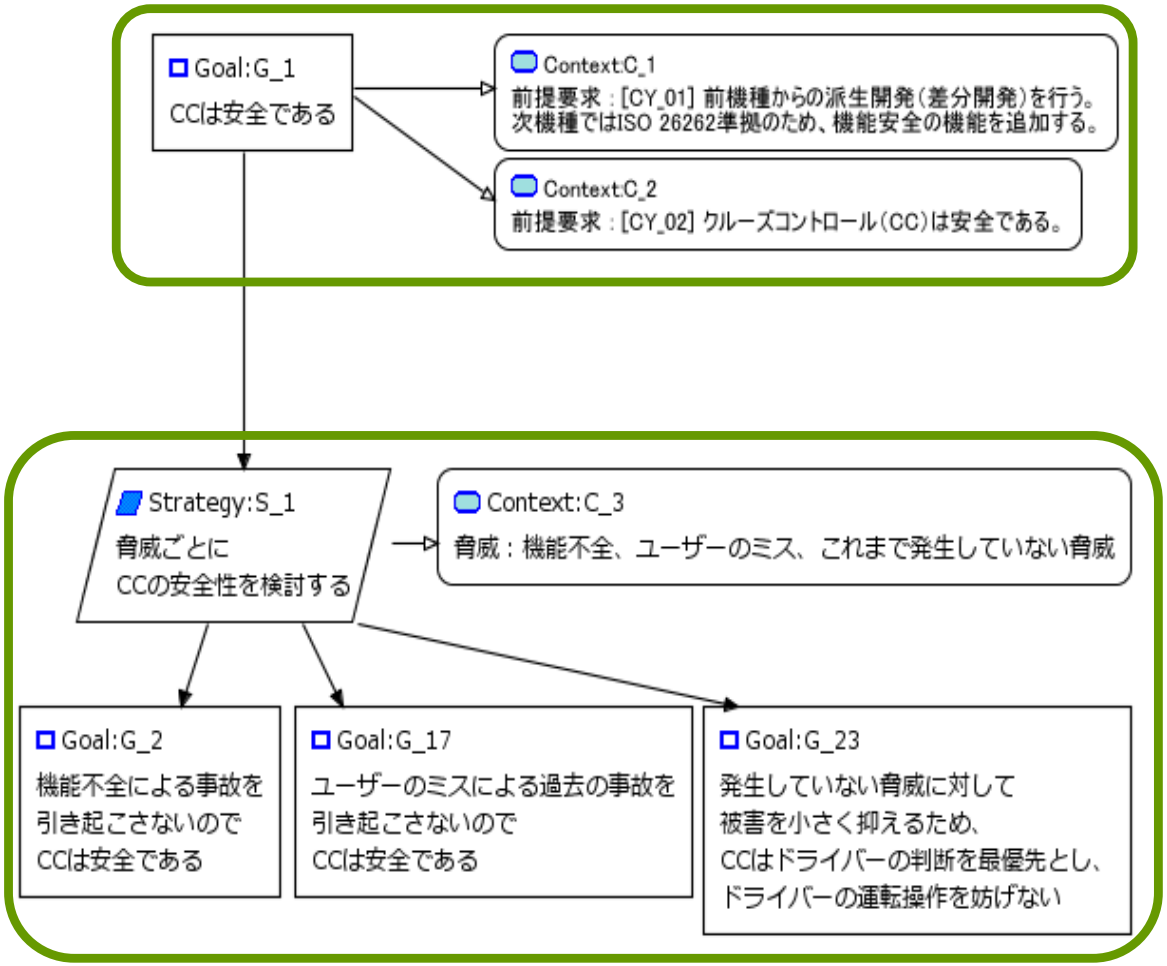
ディペンダビリティに着目し
システムが満たすべき性質と
前提を設定

ディペンダビリティへの脅威の明確化

ディペンダビリティを阻害する
すべての脅威を抽出するため大別

脅威を機能不全、ユーザーのミス、
発生していない脅威に大別し、
トップゴールを分解

例. 車載クルーズコントロール（CC）システムの安全性



フロー詳細 ～ 脅威の発生シーンの明確化

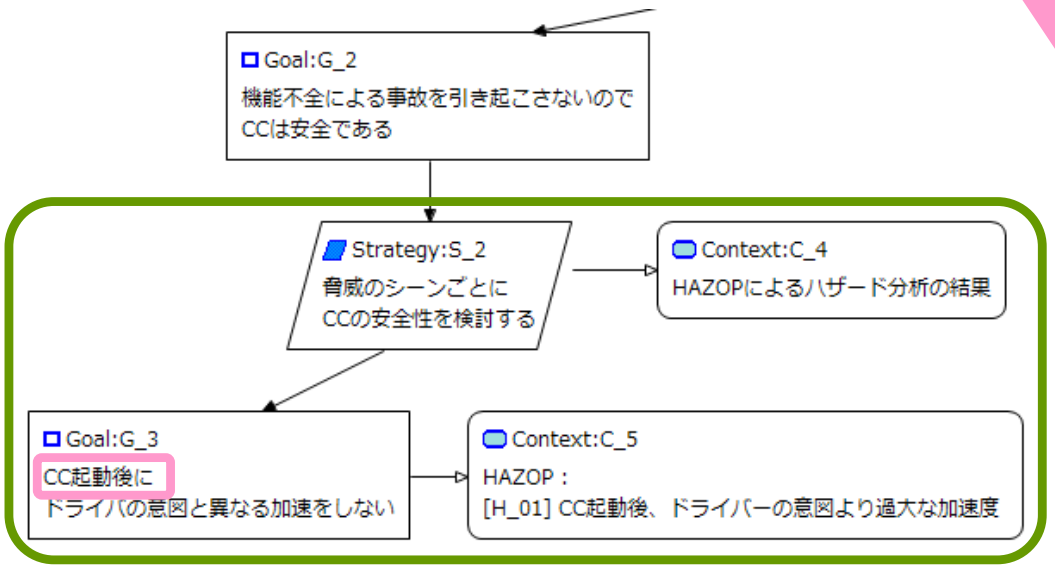
脅威ごとに発生シーンを明確化し、システムの環境と操作を明確化

D-Case

脅威の発生シーンの明確化

ディペンダビリティを阻害する すべてのシーンの抽出

- 脅威を引き起こす操作を明確化
- システムの置かれた環境を明確化



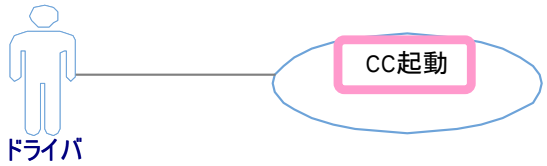
反映

SysML

ユースケース図　ユースケース

システムの利用者と操作の関係の明確化

脅威を引き起こす操作を基に
システムのユースケースを更新



モデル・シミュレーション　検証の条件

検証条件の明確化

システムの置かれた環境を基に
検証シナリオの条件を更新

フロー詳細 ～ 原因の明確化 と 対策の明確化

脅威を引き起こす原因を分析し、対策を立案

D-Case

原因の明確化

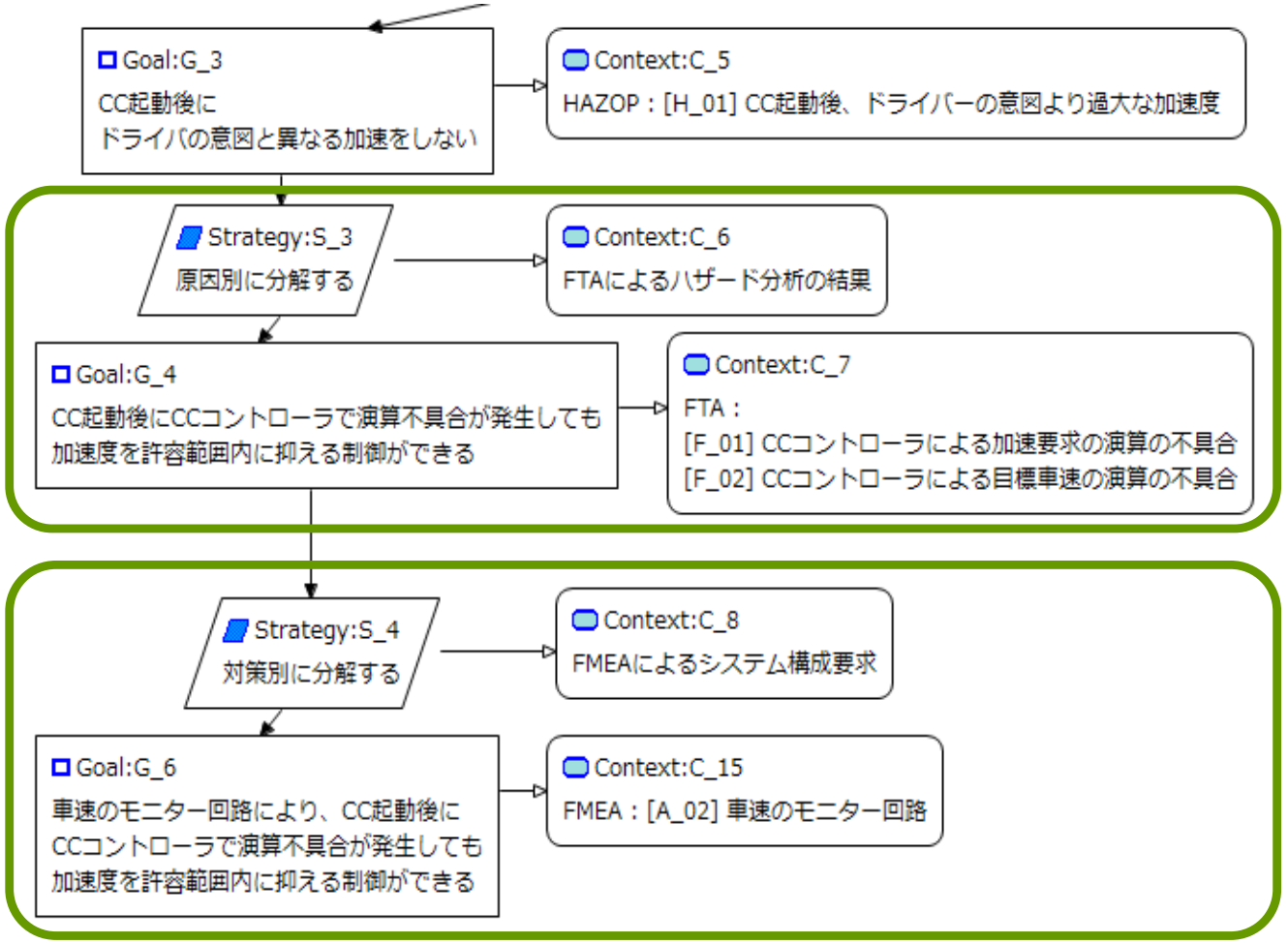
対策を立てられる粒度に至るまで原因を深堀り

各シーンで脅威を引き起こす原因を分析

対策の明確化

原因ごとに対策を立案

脅威の原因ごとに対策をシステム要求として整理



フロー詳細 ～ 要件定義

対策を基にシステム要求を明確化し、要件を導出

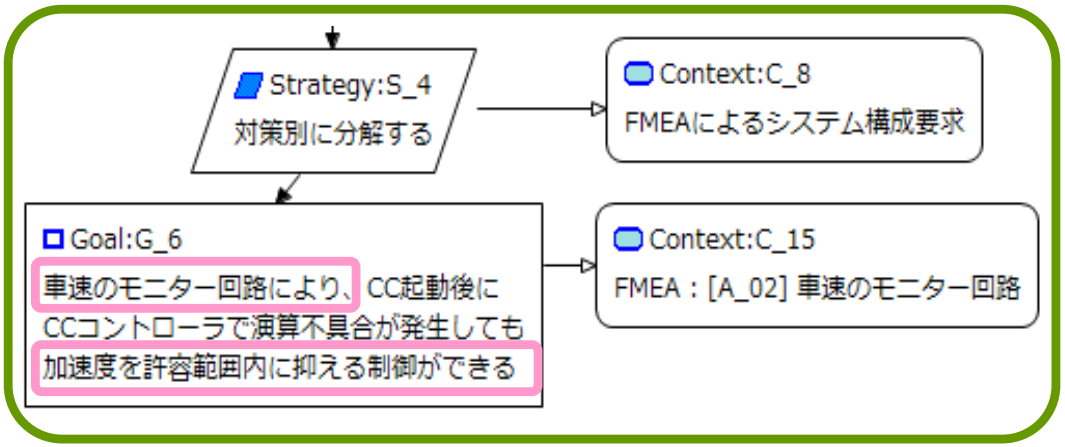
D-Case

対策の明確化

対策ごとにシステム要求を詳細化

- 機能要求の明確化
- 非機能要求の明確化
- システム構成要求の明確化
- 構成要素に課された制約の明確化

反映

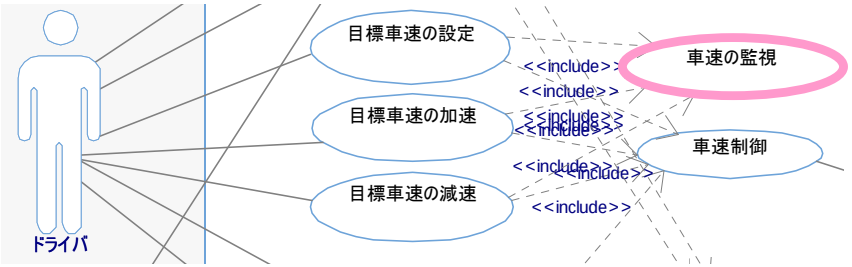


SysML

ユースケース図 ユースケース

システムの利用者と操作の関係

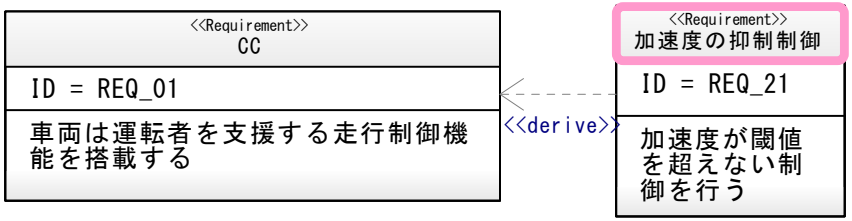
機能要求を基にユースケースを更新



要求図 機能・非機能要件

システムの機能・非機能要件を明確化

機能・非機能要求を基に要件を更新



フロー詳細 ～ システム設計

システム要求や要件に基づき設計仕様を明確化

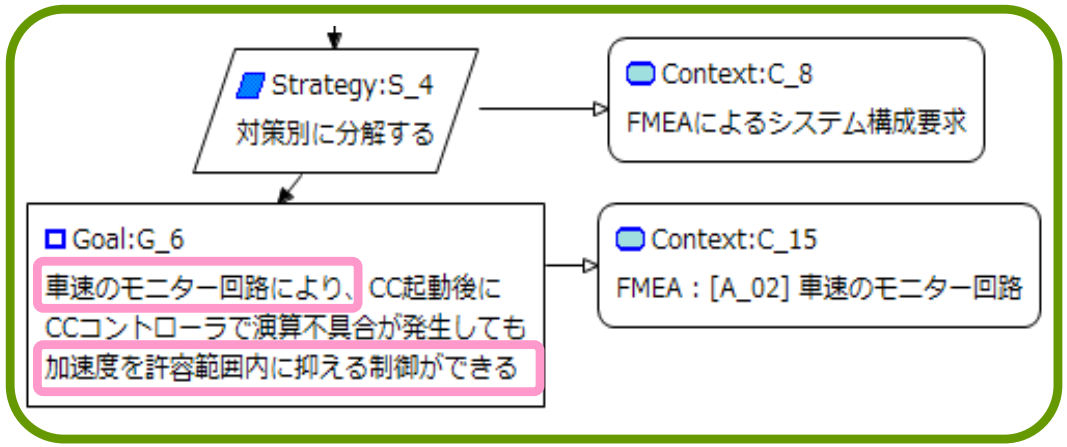
D-Case

対策の明確化

対策ごとにシステム要求を詳細化

- 機能要求の明確化
- 非機能要求の明確化
- システム構成要求の明確化
- 構成要素に課された制約の明確化

反映



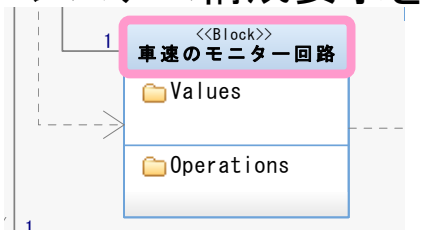
SysML

ブロック定義図
内部ブロック図

構成要素

システムのアーキテクチャを明確化

システム構成要求を基にブロックを更新

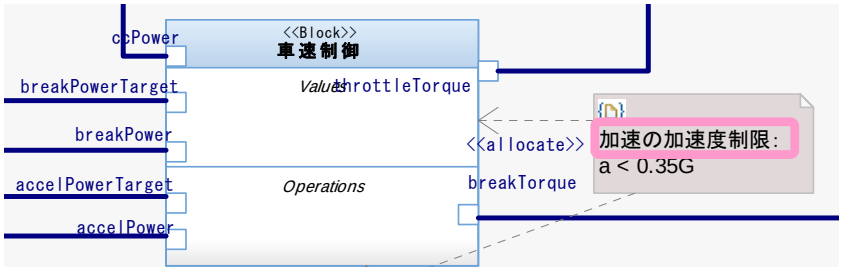


パラメトリック図

制約

システムの前提と制約を明確化

ブロックに課された制約を更新



フロー詳細 ～ 検証結果による保証

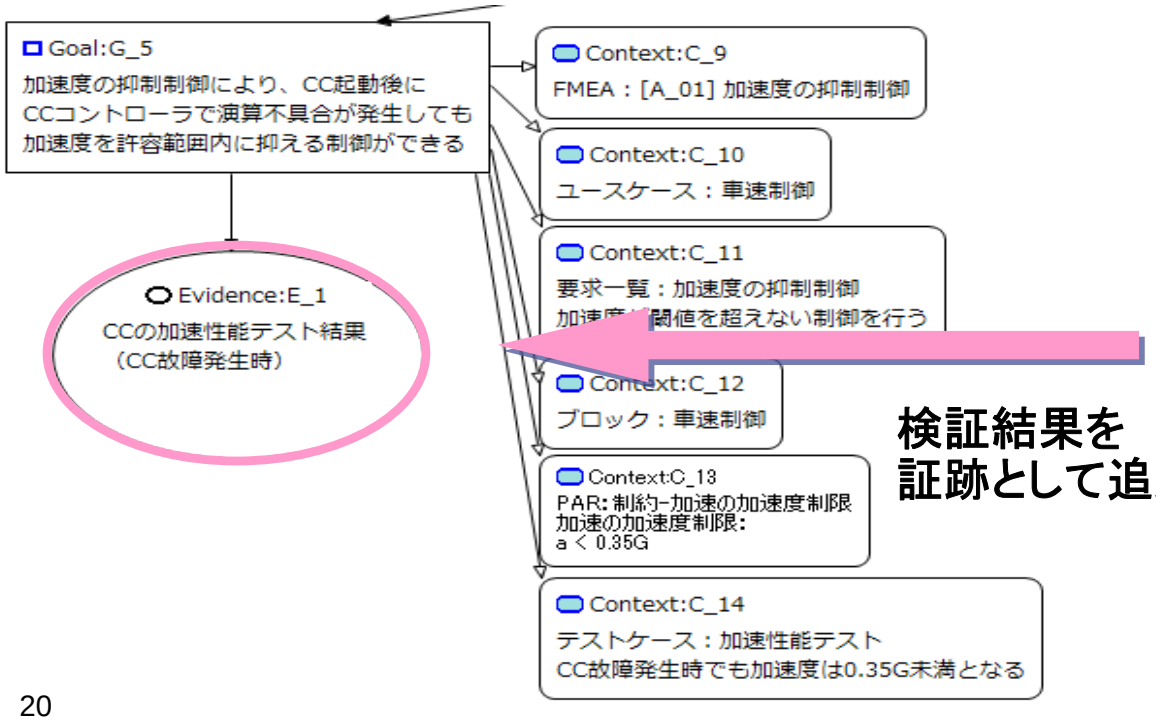
システム要求を満足することを検証結果により保証

D-Case

検証結果による保証

システム要求を満足することを明示

検証結果をエビデンスとしてゴールに関連付け



検証結果を証跡として追加

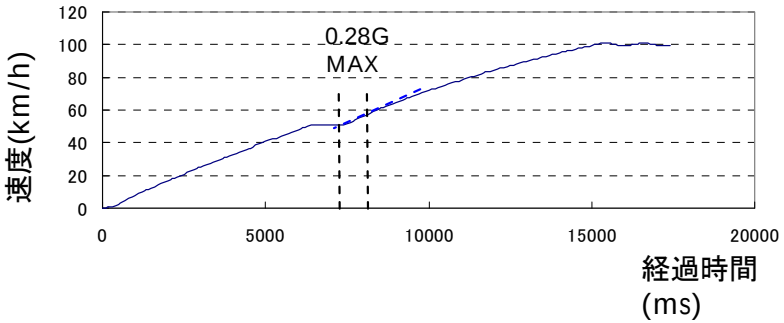
SysML

モデル・シミュレーション

検証結果

実行可能なモデルによるシステム検証

シミュレーション結果の獲得



本手法を適用するメリット

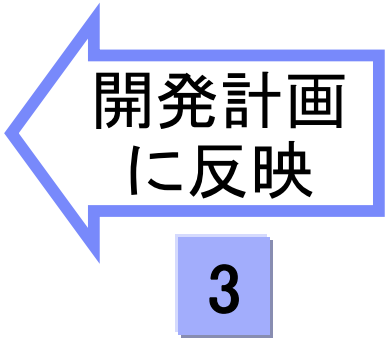
1. 開発の上流工程から下流工程まで開発の意図を反映し、品質を向上
2. D-Caseの情報からモデルを作成可能
3. D-Caseが開発プロセスをコントロールし、開発計画の精度を向上

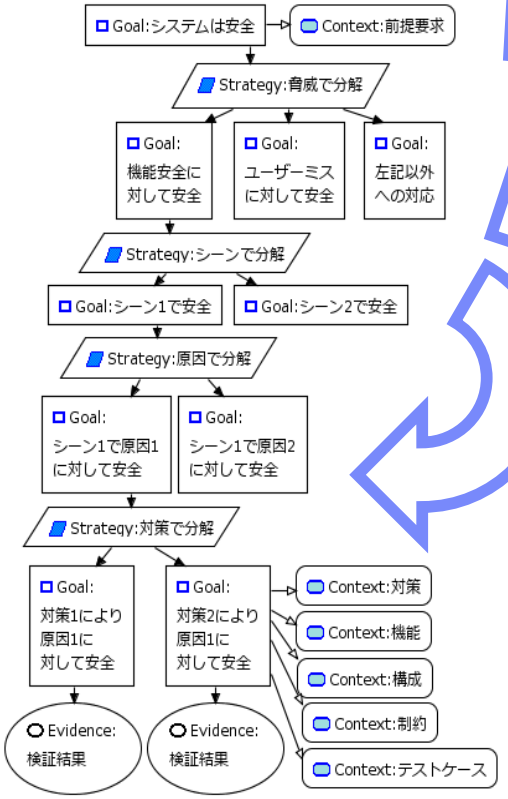
開発プロセス	D-Case	SysMLによる開発成果物

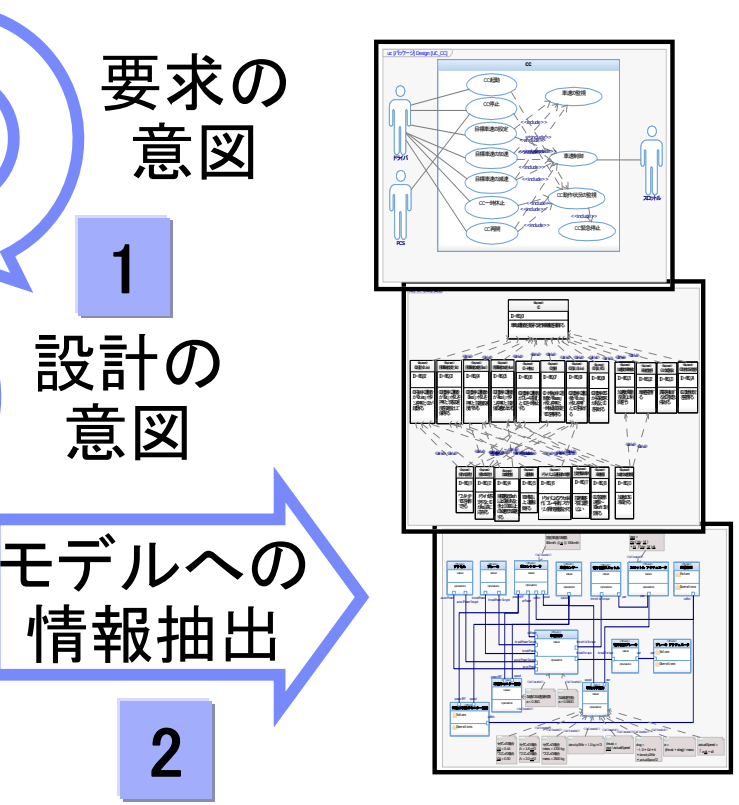
開発の計画

開発の実施

開発のコントロール

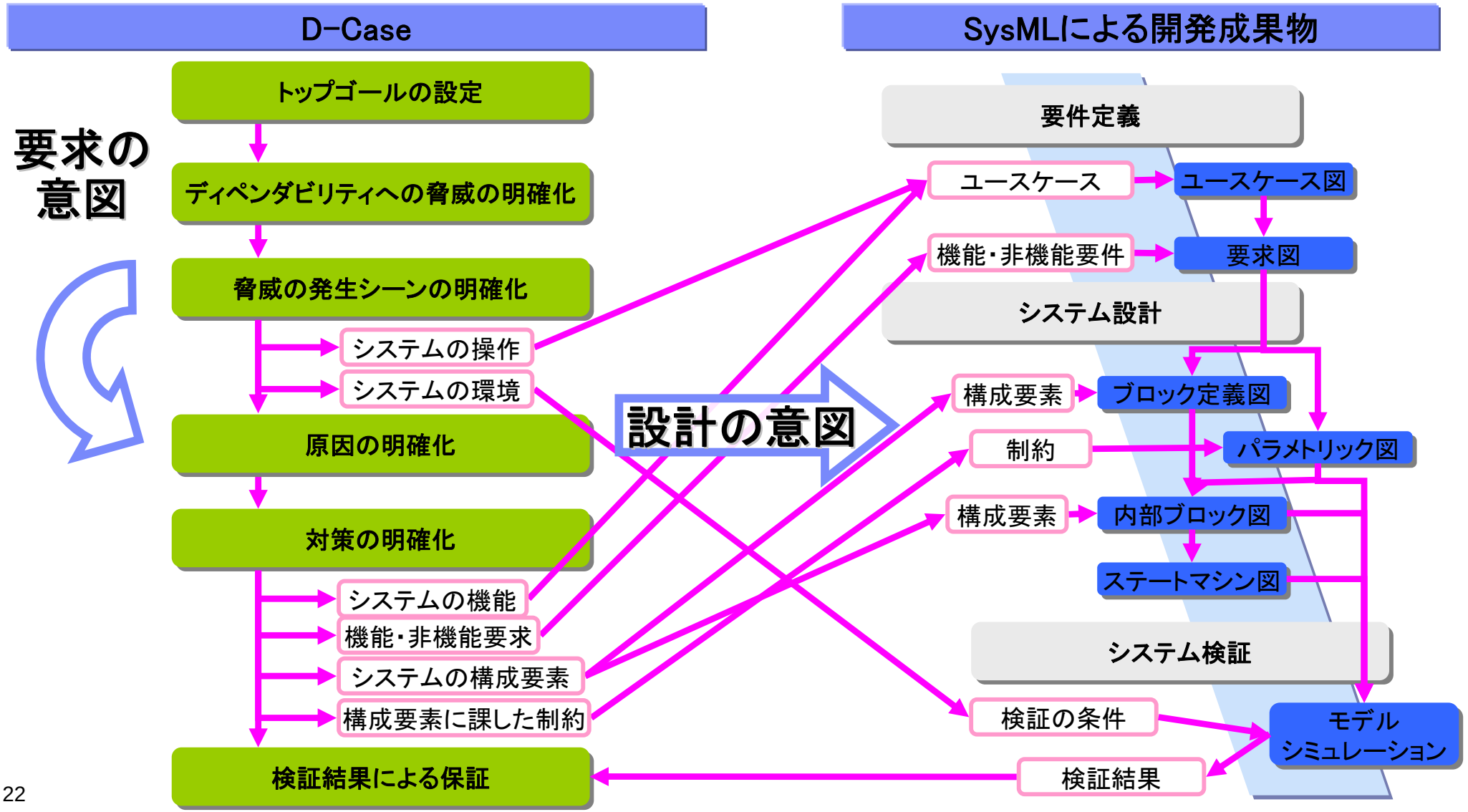






メリット 1. 開発の上流工程から下流工程まで開発の意図を反映し、品質を向上

D-Caseで開発の意図を明確化し、SysMLモデルに反映



メリット 2. D-Caseの情報からモデルを作成可能

D-Caseに記載された記述を基に、SysMLモデル要素を更新

D-Case

脅威の発生シーンの明確化

- システムの操作
- システムの環境

Goal:G_3
CC起動後に
ドライバの意図と異なる加速をしない

対策の明確化

- システムの機能
- 機能・非機能要求
- システムの構成要素
- 構成要素に課した制約

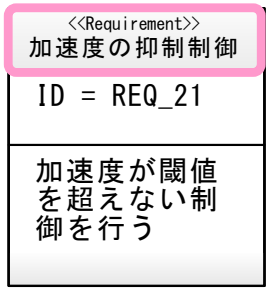
Goal:G_5
加速度の抑制制御により、CC起動後に
CCコントローラで演算不具合が発生しても
加速度を許容範囲内に抑える制御ができる

反映

SysMLによる開発成果物

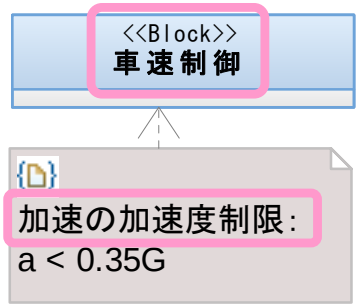
要件定義

ユースケース図
ユースケース



要求図
機能・非機能要件

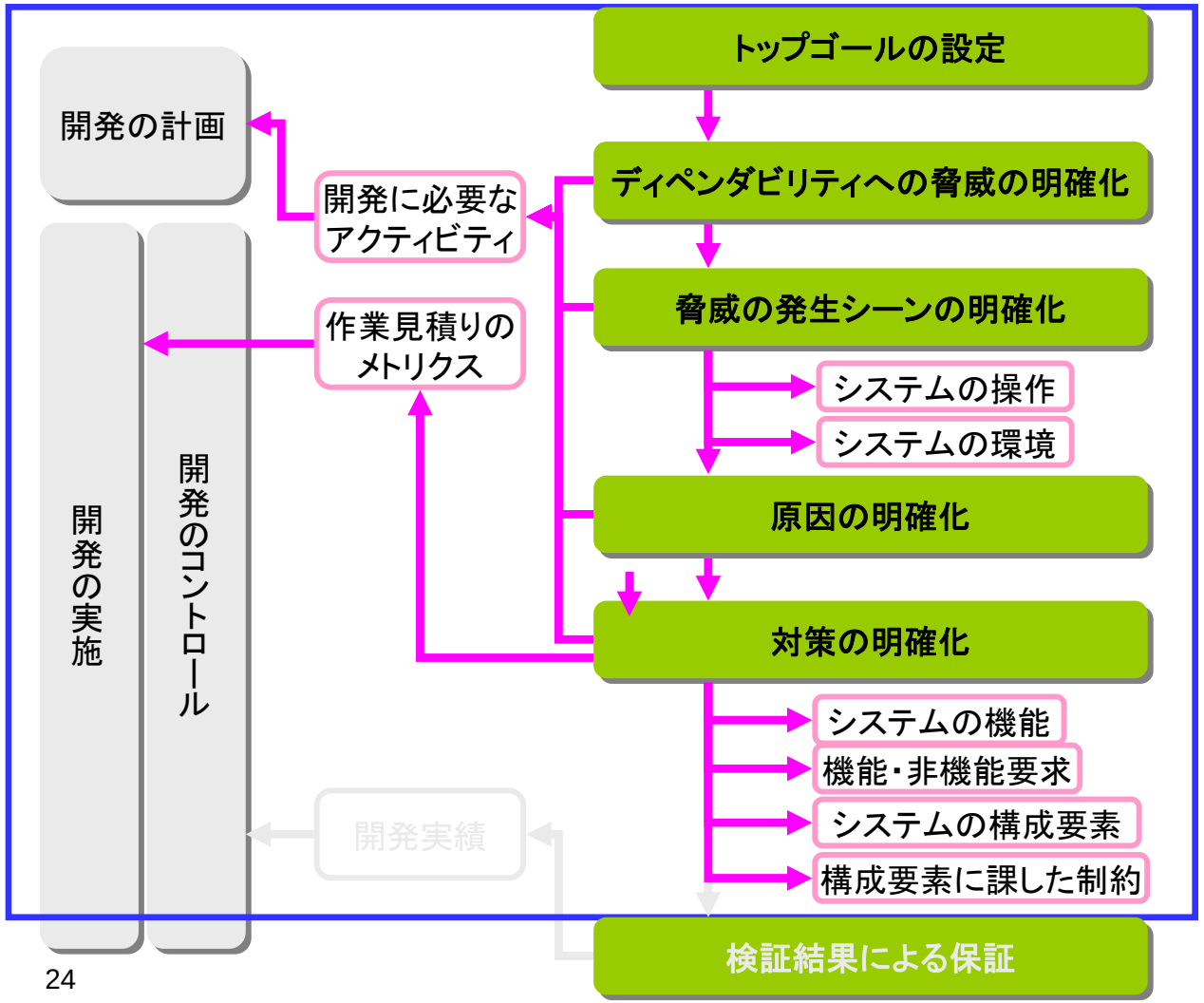
システム設計



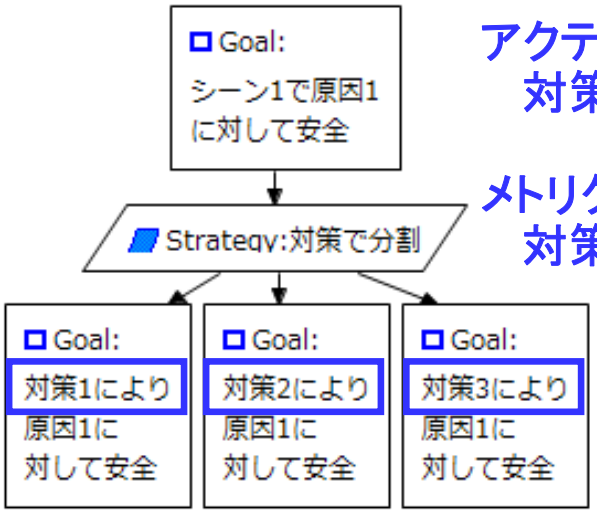
ブロック定義図
構成要素
パラメトリック図
制約

メリット 3. D-Caseが開発プロセスをコントロールし、開発計画の精度を向上

網羅的な対策立案により、開発計画の精度を向上



- 開発に必要なアクティビティの抽出
  - ➡ 開発計画の精度向上
- 作業見積もりの基本となるメトリクスの収集
  - ➡ 工数計画の精度向上



アクティビティ：
対策の内容

メトリクス：
対策の数

本手法の適用を支援するガイド

モデリングガイドとテンプレートを提供

モデリングガイド

- D-CaseとSysMLのモデリングフロー
- D-Caseノードの記述スタイル
- D-CaseとSysMLモデルの情報連携

テンプレート

- D-Caseの議論分解パターンとノードの記述
- SysMLモデルの構造

ガイドのモデリング対象

開発を完了した自動車に搭載される既成システムを
機能安全のISO26262に適合するための差分開発

本手法の適用例

～ ISO26262に準拠した車載システムの開発 ～

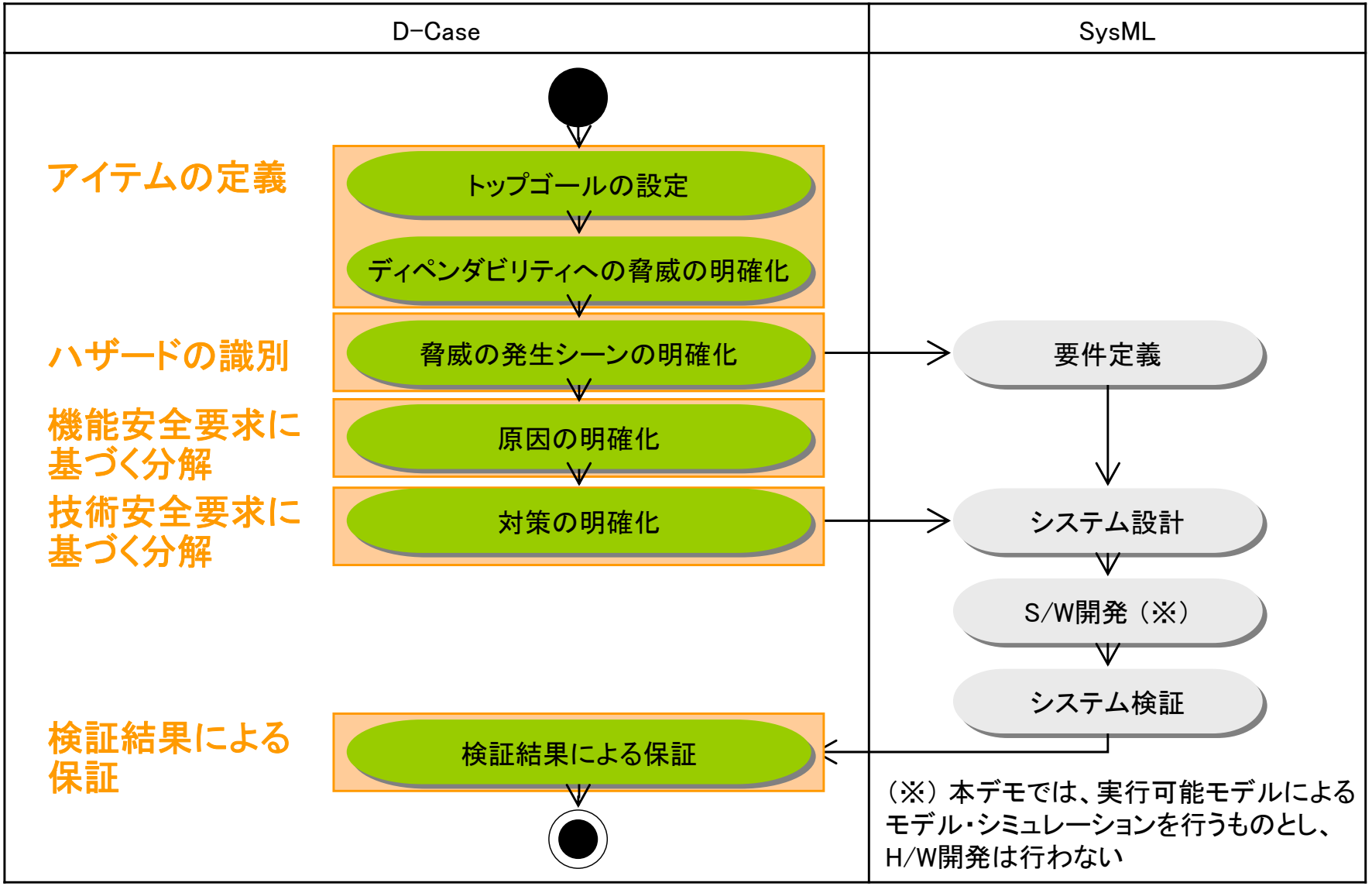
本手法とISO26262の対応関係

本手法はISO26262の安全ライフサイクルと対応

ISO26262	D-Case	SysML
3.5 アイテム定義 アイテムの定義	トップゴールの設定 安全性の観点からトップゴールと前提を設定	
	ディペンダビリティへの脅威の明確化 安全性を阻害する脅威を大別	ユースケース図 システムの利用者と操作の関係の明確化
3.7 ハザード分析とリスク評価 ハザードの識別	脅威の発生シーンの明確化 安全性を阻害するすべてのシーンを明確化	要求図 システムの機能・非機能要件を明確化
3.8 機能安全コンセプト 機能安全要求に基づく分解	原因の明確化 対策を立てられる粒度まで原因を深堀り	ブロック定義図 システムのアーキテクチャを明確化
		パラメトリック図 システムの前提と制約を明確化
4.6 技術安全要求の仕様 技術安全要求に基づく分解	対策の明確化 対策ごとにシステム要求を詳細化	モデル・シミュレーション 検証条件の明確化 実行可能なモデルによるシステム検証
4.7 システム設計 検証結果による保証	検証結果による保証 システム要求を満足することを明示	

本デモの開発フロー

D-CaseとSysMLを連携してISO26262に準拠するシステムを開発



ISO26262に基づくD-CaseとSysMLモデリング

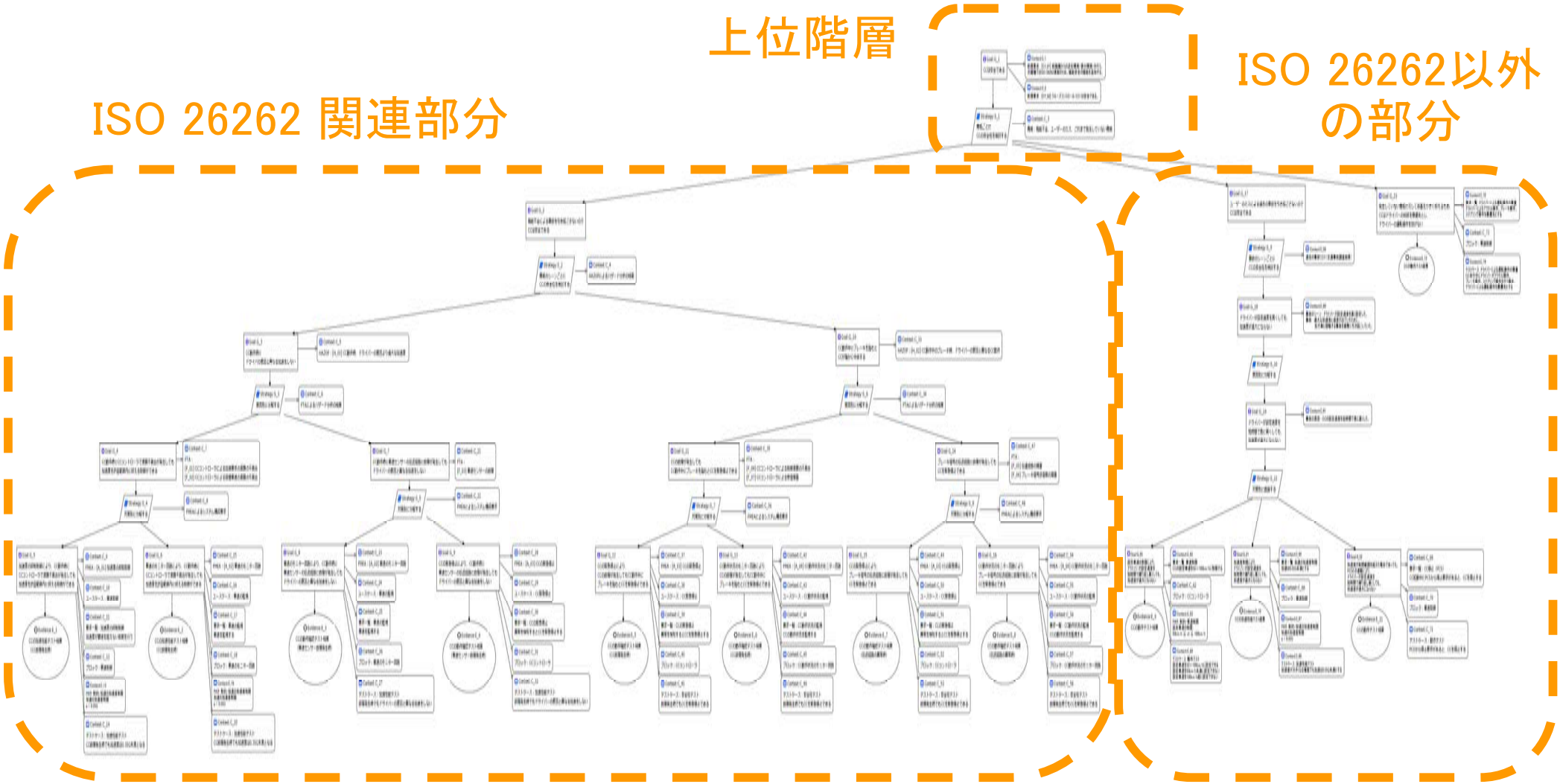
- 安全ライフサイクル基づくD-Caseの構造
- D-Caseとの連携によるSysMLモデリング

モデリングガイドの構成

対象	D-Case		SysML	
カテゴリ	D-Caseの構造	ノードの記述スタイル	D-Caseからの情報抽出	D-Caseへの関連付け
ガイド項目	アイテムの定義	達成すべきゴールと前提の記述	–	–
	ハザードの識別	システムの操作と環境の記述	システムの操作と環境	ユースケース検証の条件
	機能安全要求に基づく分解	対策を立てられるまで原因を明確化	–	–
	技術安全要求に基づく分解	システム要求の記述	機能・非機能要求構成要素と制約	ユースケース、要求構成要素、制約
	検証結果による保証	検証に必要な情報の記述	制御の条件と処理	検証結果

モデリングガイドに基づき作成したD-Caseの概要

ISO26262関連部分とそれ以外にゴールを分解して詳細化



ISO26262の安全性要求と信頼性要求

機能安全に必要である安全性と信頼性に着目

機能安全のISO26262

電気電子(E/E)システムの機能不全の振る舞いにより
引き起こされるハザードが原因となる許容できないリスクが無い状態

安全性要求と信頼性要求に分解

安全性要求

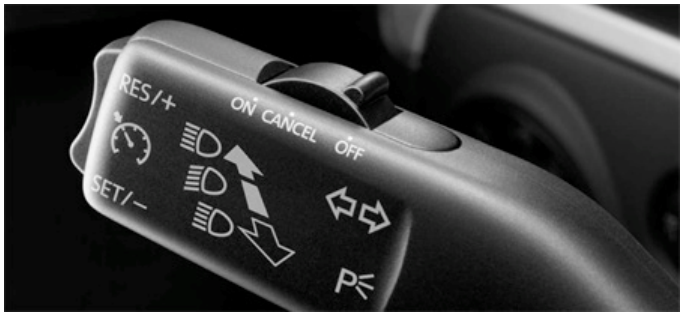
安全機能等によりシステムの故障に対してもハザードを
許容可能な範囲に低減可能である

信頼性要求

システムが正しく機能する状態を保つ

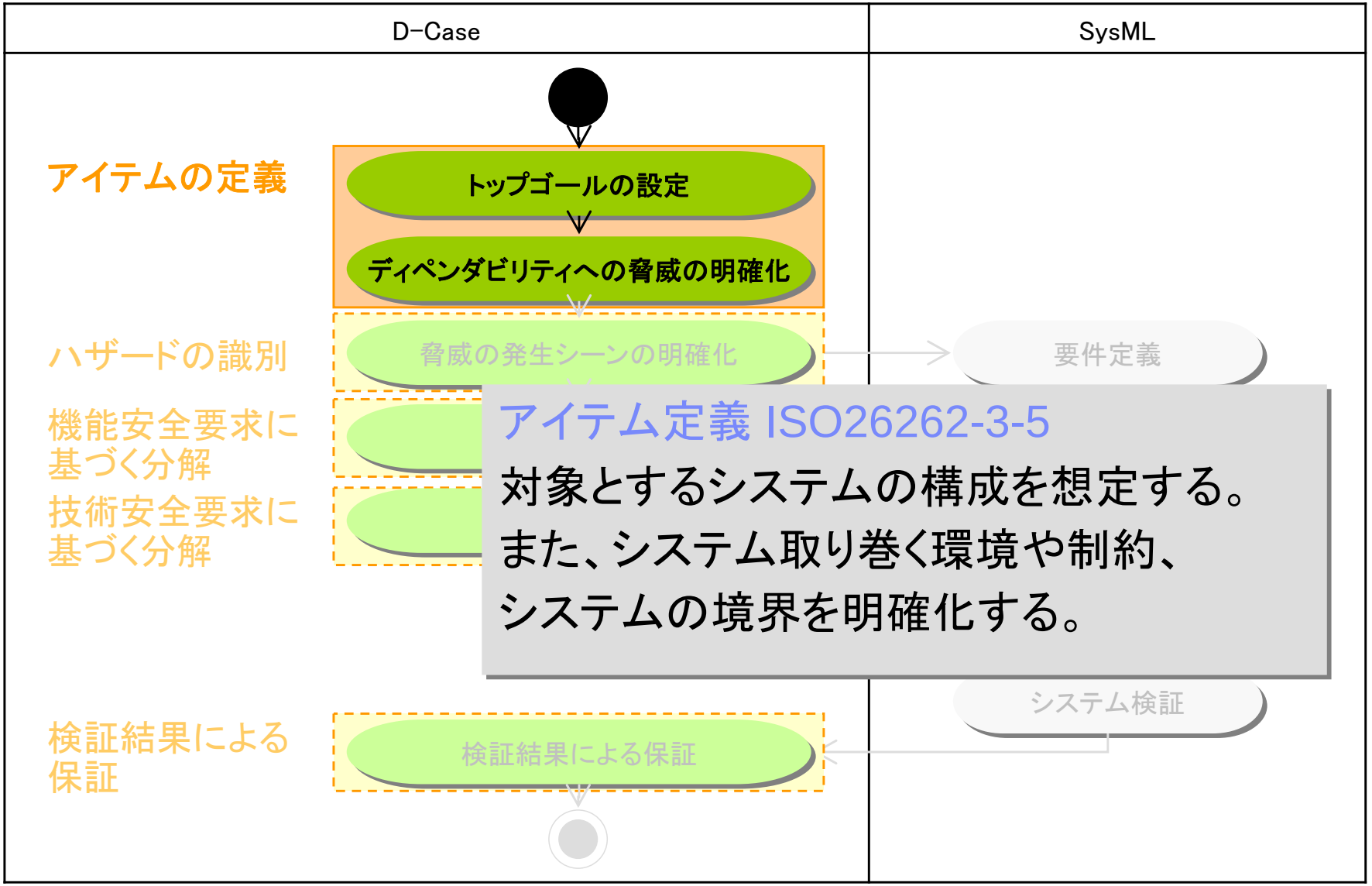
デモ ～ 開発の対象システム

システム	クルーズコントロール（CC）
前提	開発を完了したシステムを機能安全に適合させる差分開発
機能	ドライバの設定による速度制御
機能不全	システムの故障または演算不具合
ハザード	CC起動後、ドライバーの意図より過大な加速度
機能安全要求	CC起動後にドライバーの意図と異なる急加速をしない
安全性要求	CC起動後にCCコントローラで演算不具合が発生しても 加速度を許容範囲内に抑える制御が可能である
信頼性要求	CC起動後に車速センサーの伝送経路に故障が発生しても ドライバーの意図と異なる加速をしない



開発フロー ～ アイテムの定義

安全性の観点からトップゴールと前提を設定し、脅威を抽出



トップゴールの設定

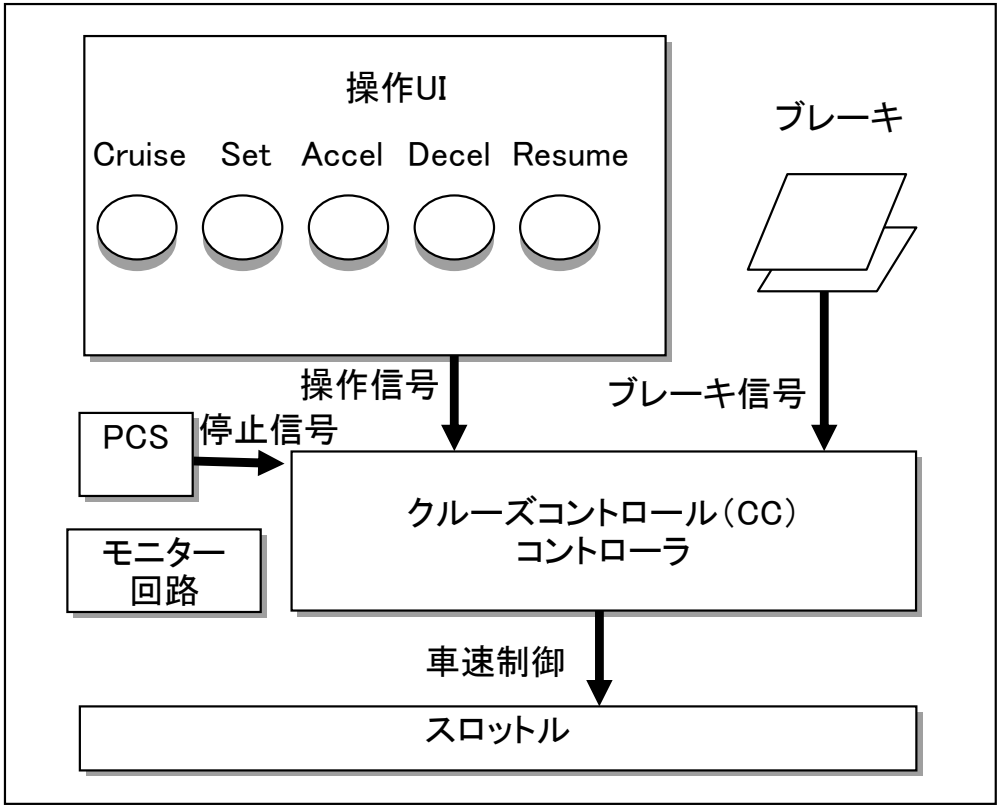
開発対象システムが満たす前提要求を明確化

機能概要

クルーズコントロール（CC）は、ドライバの指示を受け、走行制御を行うことで、走行の支援を実現する。

前提要求

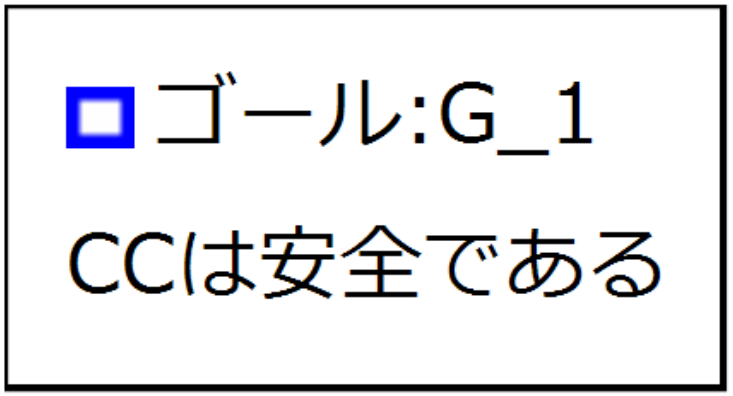
- [CY_01] 前機種からの派生開発（差分開発）を行う。
  次機種ではISO 26262準拠のため、
  機能安全の機能を追加する。
- [CY_02] クルーズコントロール（CC）は安全である。
- [CY_11] クルーズコントロール（CC）は、操作UIに
  クルーズの起動または停止を行うボタン、
  クルーズ復帰と加速開始と加速停止を
  指示するボタンで構成される。
- [CY_12] ドライバは、この操作UIとブレーキから
  CCに指示を送る。
- [CY_13] ドライバは運転中、クルーズコントロールを
  いつでも指示できる。
- [CY_21] 利用するOSはxx OSとする。



トップゴールの設定

ISO26262に適合するシステムの性質として安全性に着目し
D-Caseのトップゴールとして設定

D-Caseのトップゴール



トップゴール:CCは安全である

ゴールの記述

ゴールは命題の形式で記述する
例「<システム>は<安全>である」

（凡例）
D-Case　SysML　ISO 26262

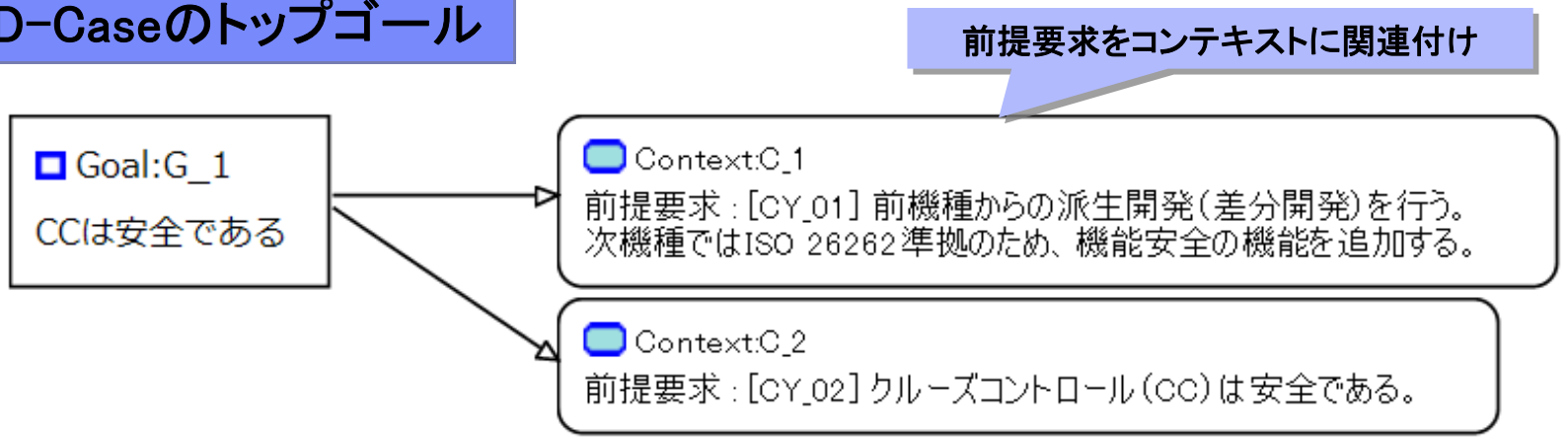
トップゴールへの前提の関連付け

トップゴールの達成に必要な前提要求をコンテキストに抽出

前提要求

・[CY_01] 前機種からの派生開発（差分開発）を行う。
　　次機種ではISO 26262準拠のため、機能安全の機能を追加する。
・[CY_02]クルーズコントロール（CC）は安全である。

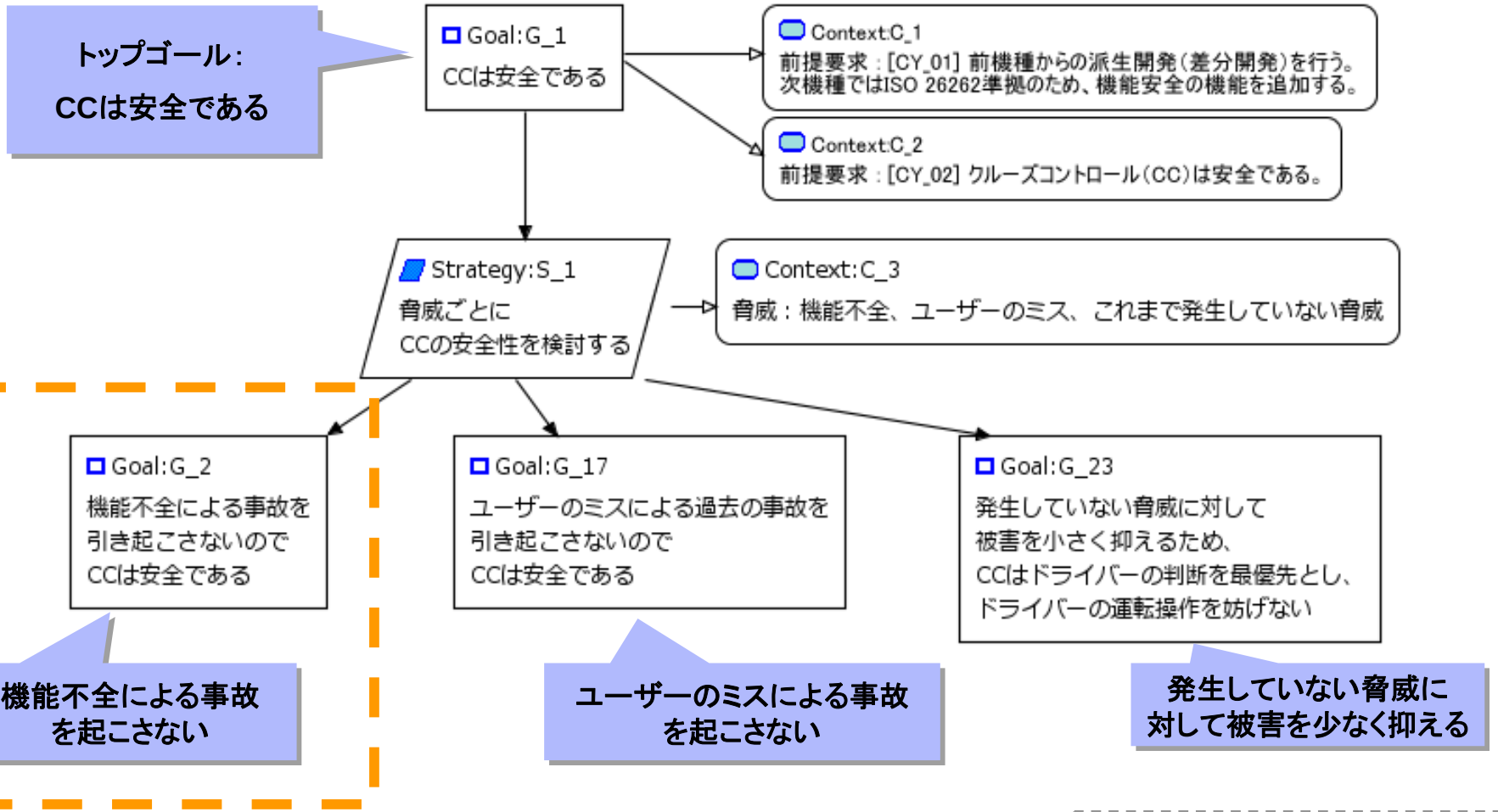
D-Caseのトップゴール



（凡例）
D-Case　SysML　ISO 26262

安全性を阻害する脅威の大別

差分開発の対象となる機能安全の観点に基づきトップゴールを分解

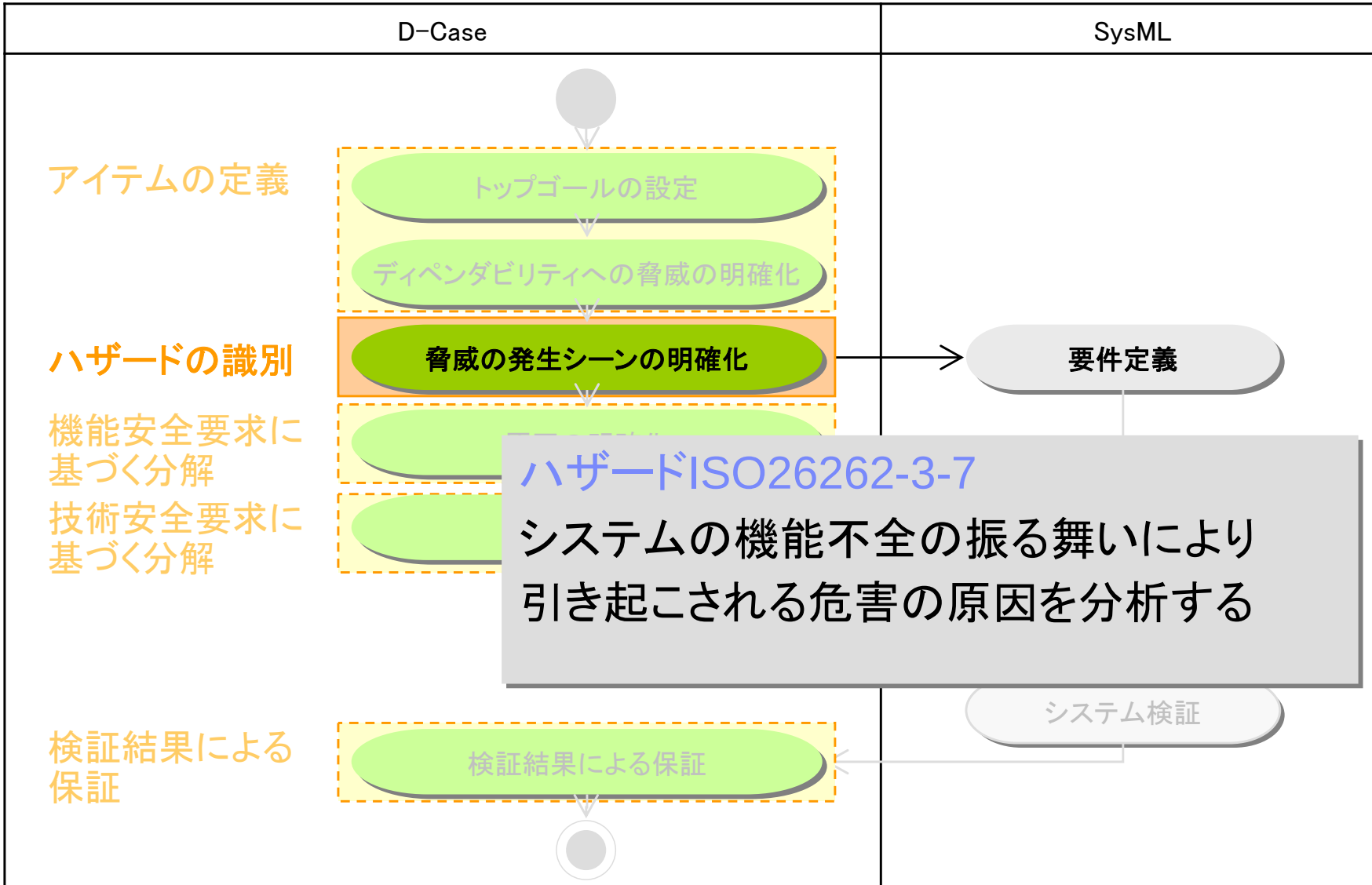


ISO 26262 関連部分

（凡例）
D-Case　SysML　ISO 26262

開発フロー ～ ハザードの識別

安全性を阻害するすべてのシーンを抽出



HAZOP分析に基づく脅威の発生シーンの明確化

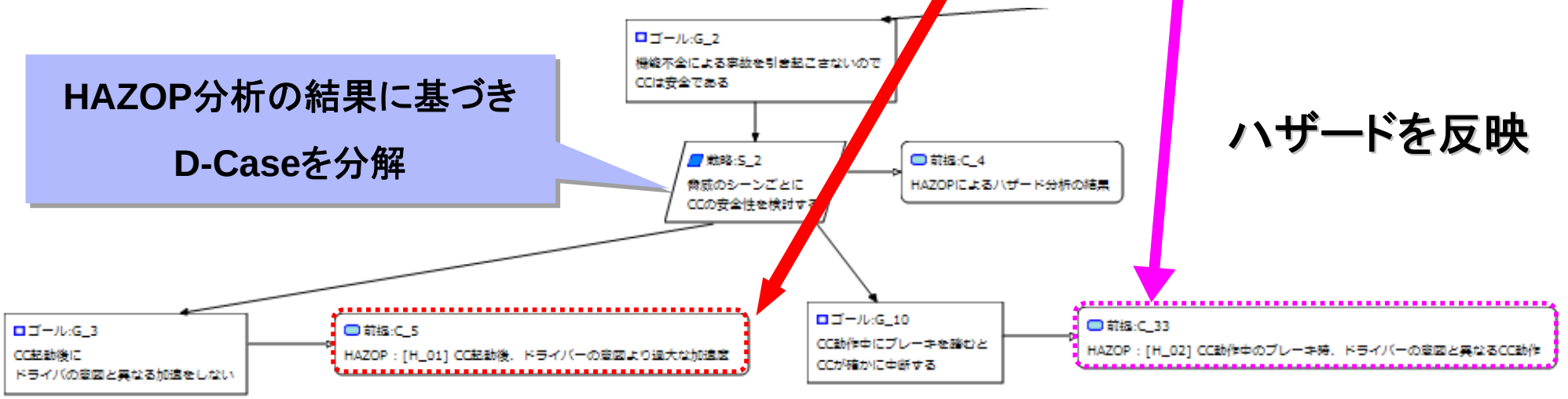
HAZOP分析により導かれるハザードの発生シーンによりゴールを分解

HAZOP分析でハザードを把握

ID	制御出力	期待値からの逸脱状態（ガイドワード）	シーン	ハザード
H_01	CCコントローラ	過大	CC起動後	ドライバーの意図より過大な加速度
H_02	CCコントローラ	不作動	CC起動後のブレーキ時	ドライバーの意図と異なるCC動作

HAZOP分析の結果に基づき D-Caseを分解

ハザードを反映



（凡例）

D-Case　SysML　ISO 26262

要件定義 ～ ユースケース図と検証条件の更新

システムのユーザーと操作の関連をユースケース図に反映し、
検証条件を抽出しテストケースに反映

D-Case

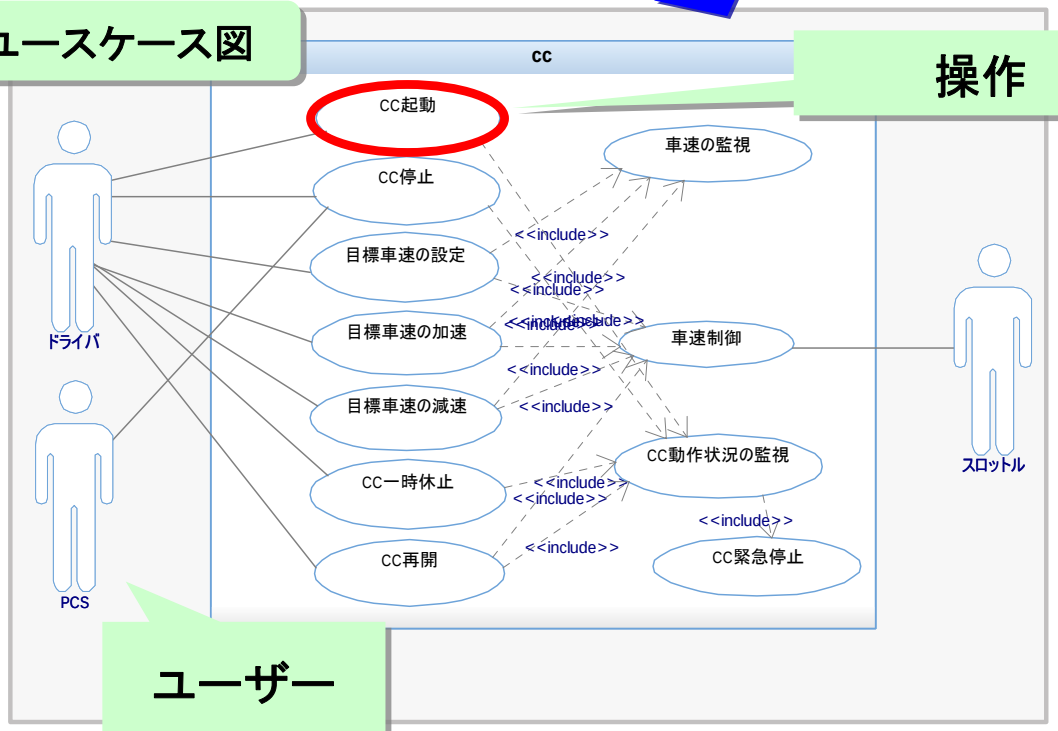
Goal:G_3
CC起動後に
ドライバの意図と異なる加速をしない

Context:C_5
HAZOP：[H_01] CC起動後、ドライバーの意図より過大な加速度

機能に関する記述を参照して
ユースケースを更新

検証条件として
加速度の制約を抽出

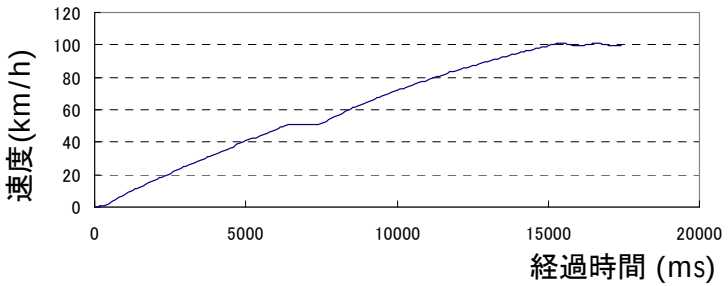
ユースケース図



操作

ユーザー

検証条件

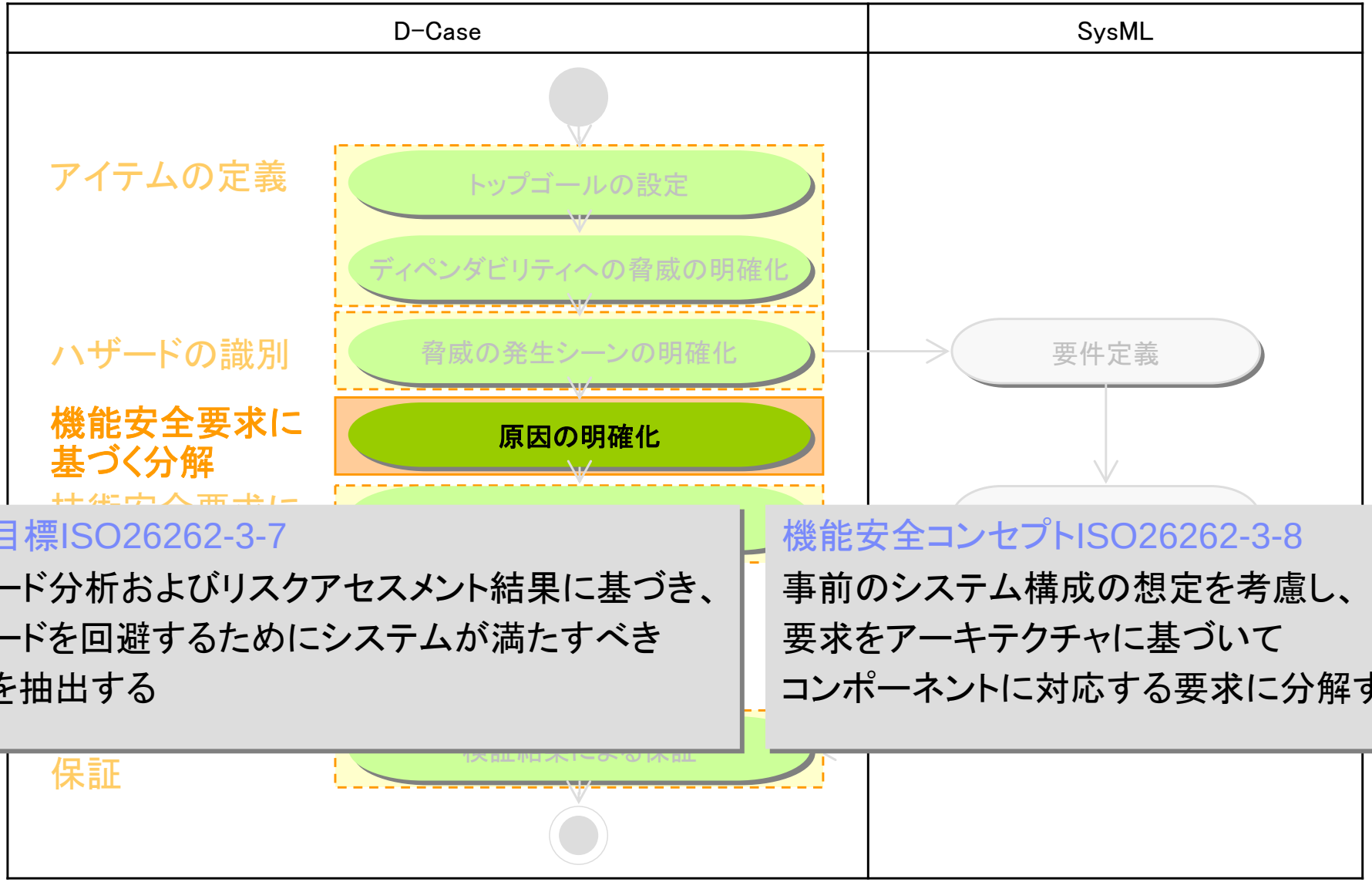


ドライバーの意図通りの加速度

（凡例）
D-Case　SysML　ISO 26262

開発フロー ～ 機能安全要求に基づく分解

対策を立案可能な粒度に至るまで原因分析を実施



安全目標ISO26262-3-7

ハザード分析およびリスクアセスメント結果に基づき、
ハザードを回避するためにシステムが満たすべき
要求を抽出する

機能安全コンセプトISO26262-3-8

事前のシステム構成の想定を考慮し、
要求をアーキテクチャに基づいて
コンポーネントに対応する要求に分解する

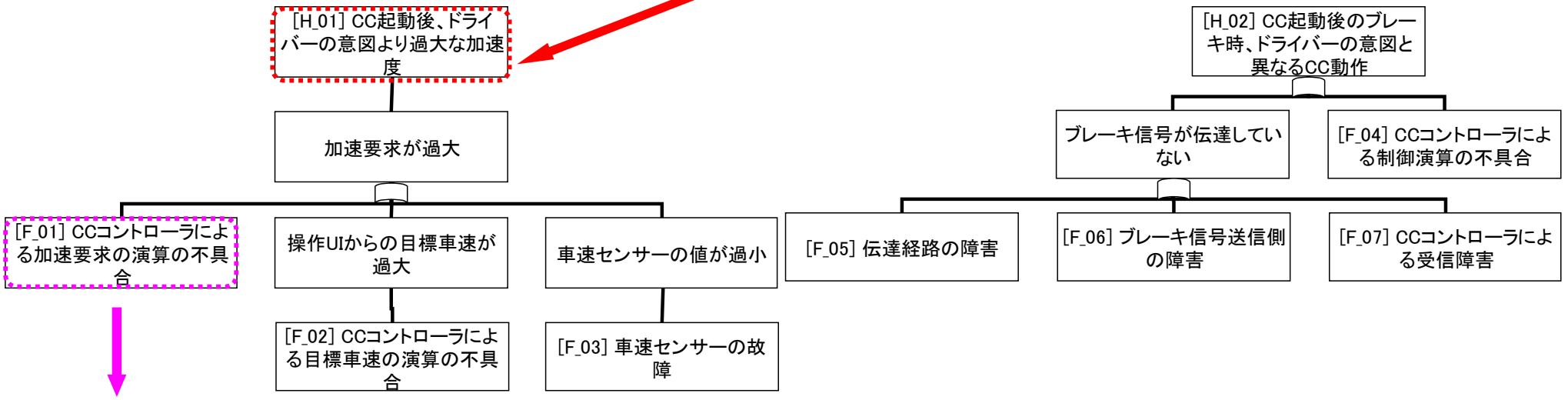
FTA分析に基づく脅威の原因の明確化

FTA分析により、ハザードの原因を分析

HAZOP分析で得られたハザード

ID	制御出力	期待値からの逸脱状態（ガイドワード）	シーン	ハザード
H_01	CCコントローラ	過大	CC起動後	ドライバーの意図より過大な加速度
H_02	CCコントローラ	不作動	CC起動後のブレーキ時	ドライバーの意図と異なるCC動作

FTAでハザードに至る経路、原因を把握



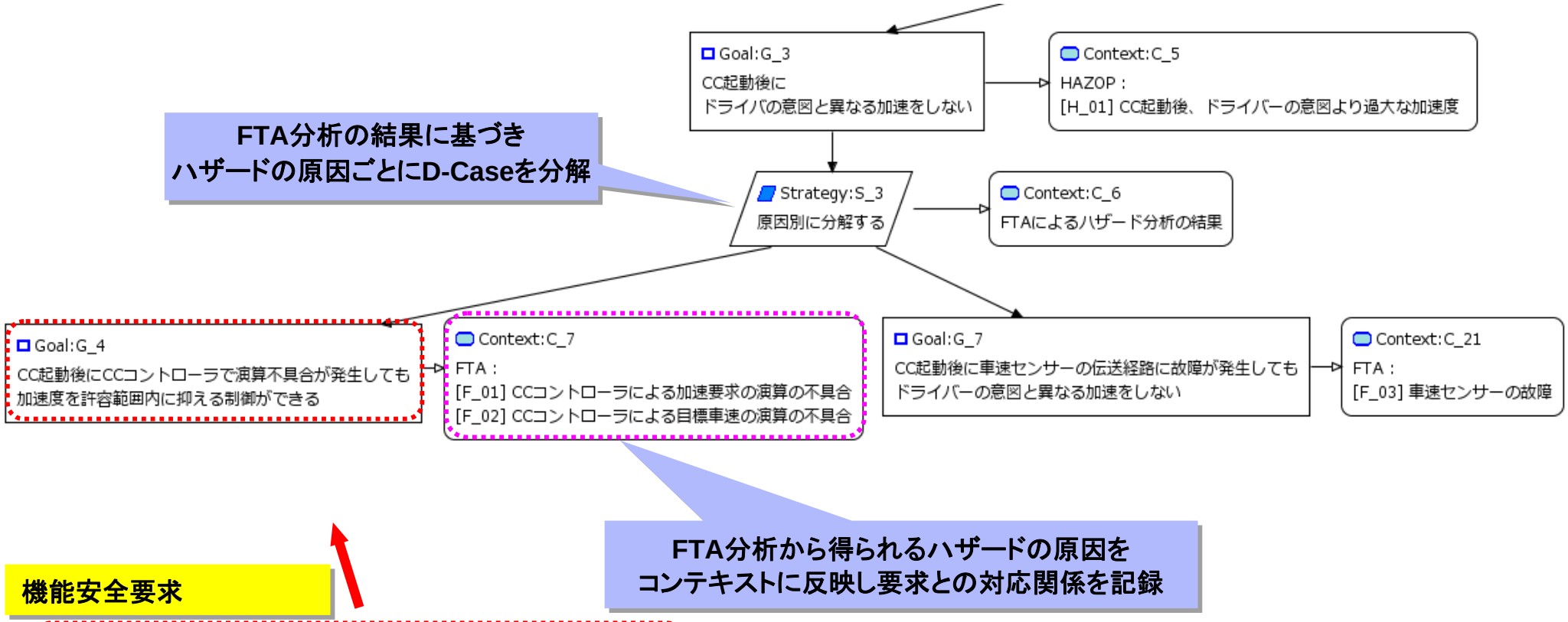
機能安全要求の抽出

CCコントローラで演算不具合が発生しても
加速度許容範囲内に抑える制御が可能である

（凡例）
D-Case　SysML　ISO 26262

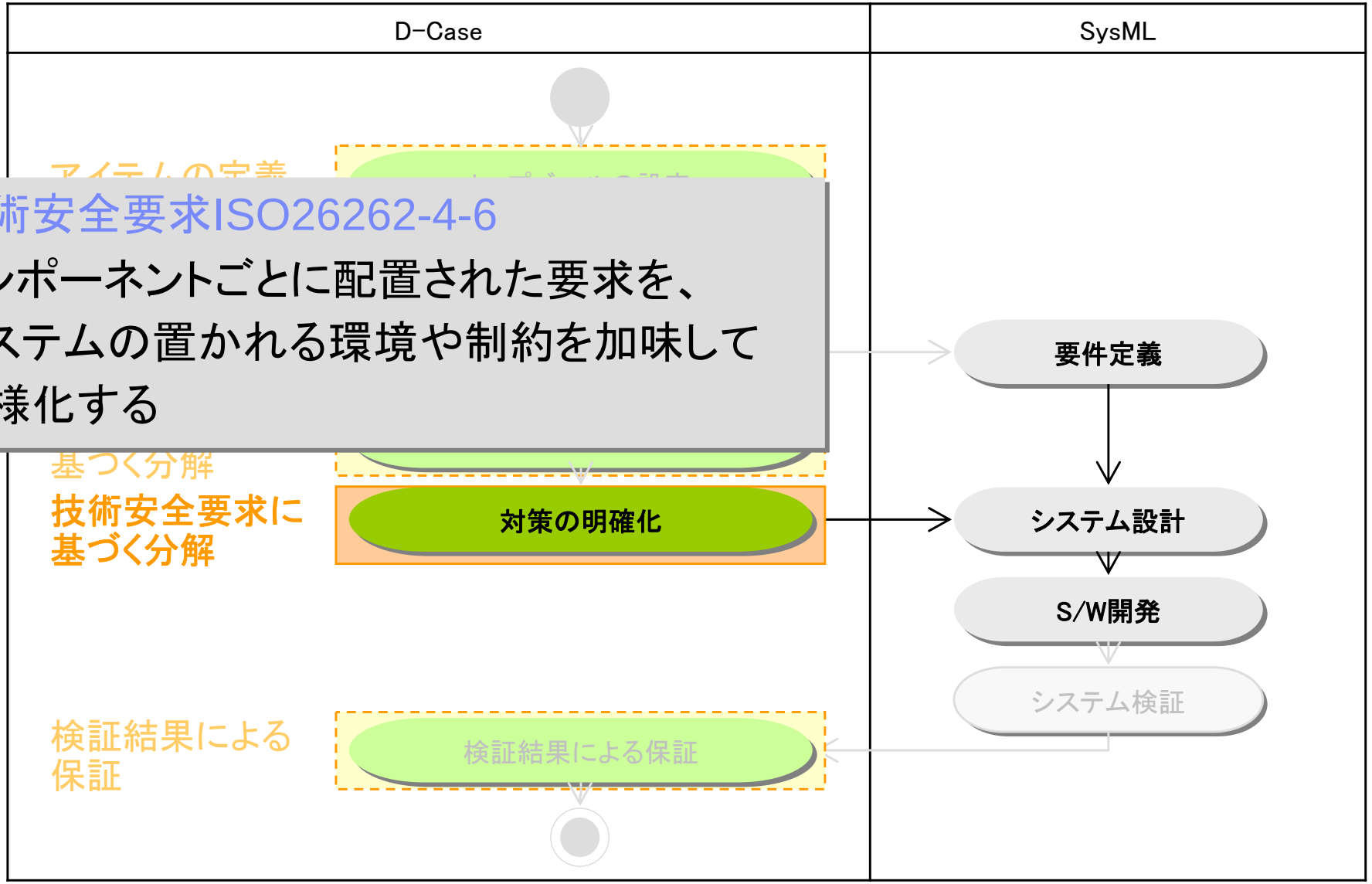
FTA分析に基づく脅威の原因の明確化

FTA分析結果から導出されるハザードの原因によりゴールを分解



開発フロー ～ 技術安全要求に基づく分解

対策を基にシステム要求、要件、設計仕様を詳細化

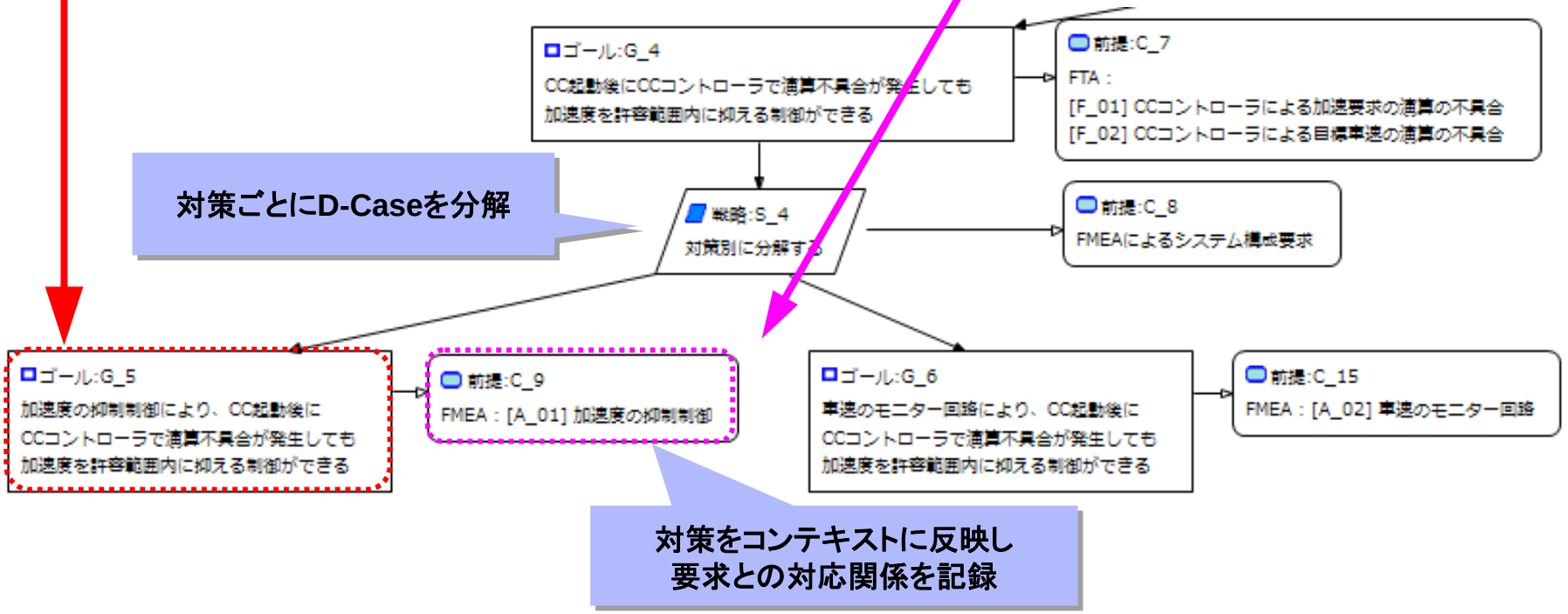


FMEA分析に基づく対策の明確化

FMEA分析により対策を検討し、システム要求を過不足なく抽出

FMEA分析により対策を検討

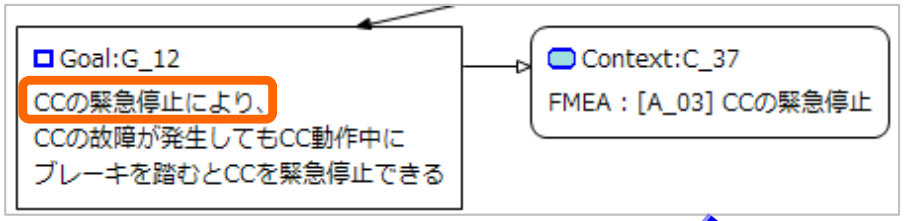
ID	コンポーネント	故障モード	要因	影響の深刻度	発生の頻度	検知の難易度	リスク優先度	対策	
								S/W	H/W
F_01	CCコントローラ	演算(加速要求)不具合	プログラム不具合	6 (中)	1 (小)	5 (中)	30	[A_01] 加速度の抑制制御	[A_02] 車速のモニター回路
F_02	CCコントローラ	演算(目標車速)不具合	プログラム不具合	3 (小)	1 (小)	1 (小)	3		
F_03	車速センサー	異常値	故障	9 (大)	5 (中)	1 (小)	45	[A_03] CCの緊急停止	[A_04] CC動作状況のモニター回路
F_04	CCコントローラ	演算(制御)不具合	プログラム不具合	9 (大)	1 (小)	1 (小)	9		
F_05	伝送経路	異常値	故障	9 (大)	5 (中)	1 (小)	45		
F_06	ブレーキ	演算(送信)不具合	断線	9 (大)	1 (小)	1 (小)	9		
F_07	CCコントローラ	演算(受信)不具合	断線	9 (大)	1 (小)	5 (中)	45		



要件定義 ～ ユースケース図の更新

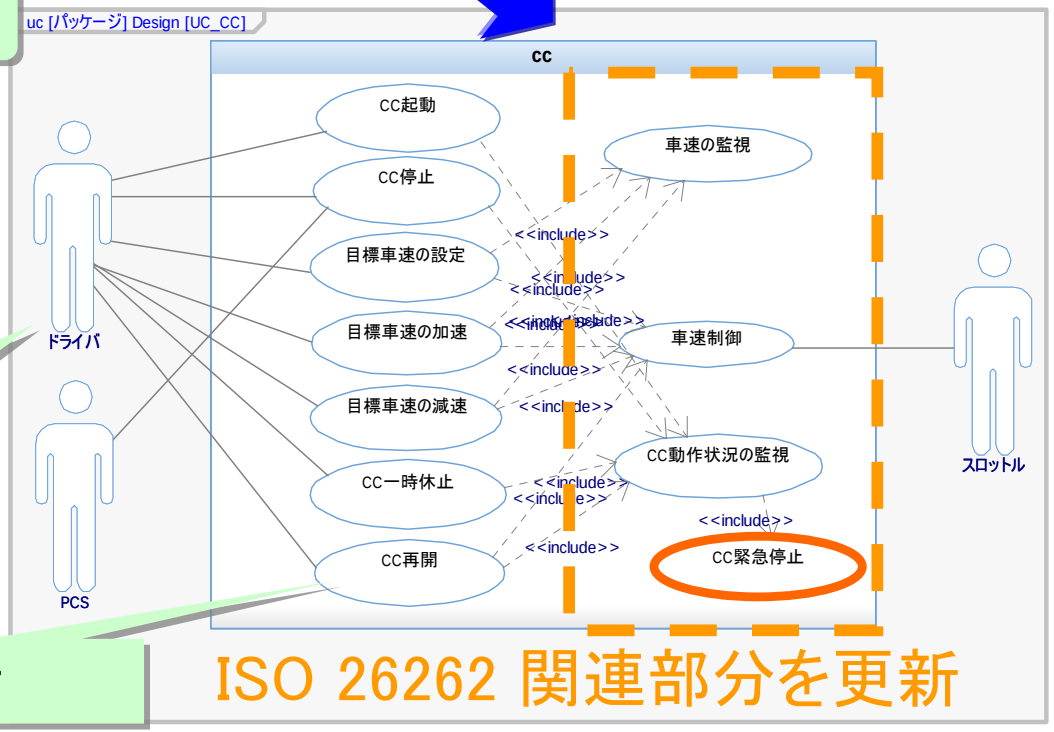
システム要求を基に、ユーザーと操作の関連をユースケース図に反映

D-Case



機能安全に関する機能を参照し
ユースケースを更新

ユースケース図



ユーザー

操作

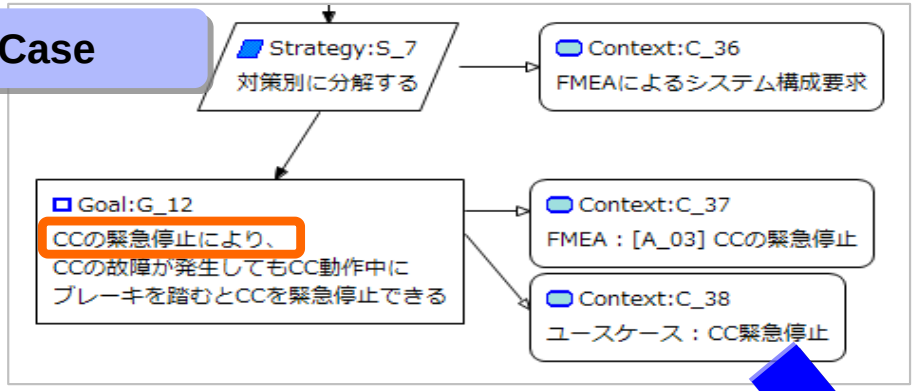
ISO 26262 関連部分を更新

（凡例）
D-Case **SysML** **ISO 26262**

要件定義 ～ 要求図の更新

システム要求を基に、詳細な機能要件と非機能要件を要求図に反映

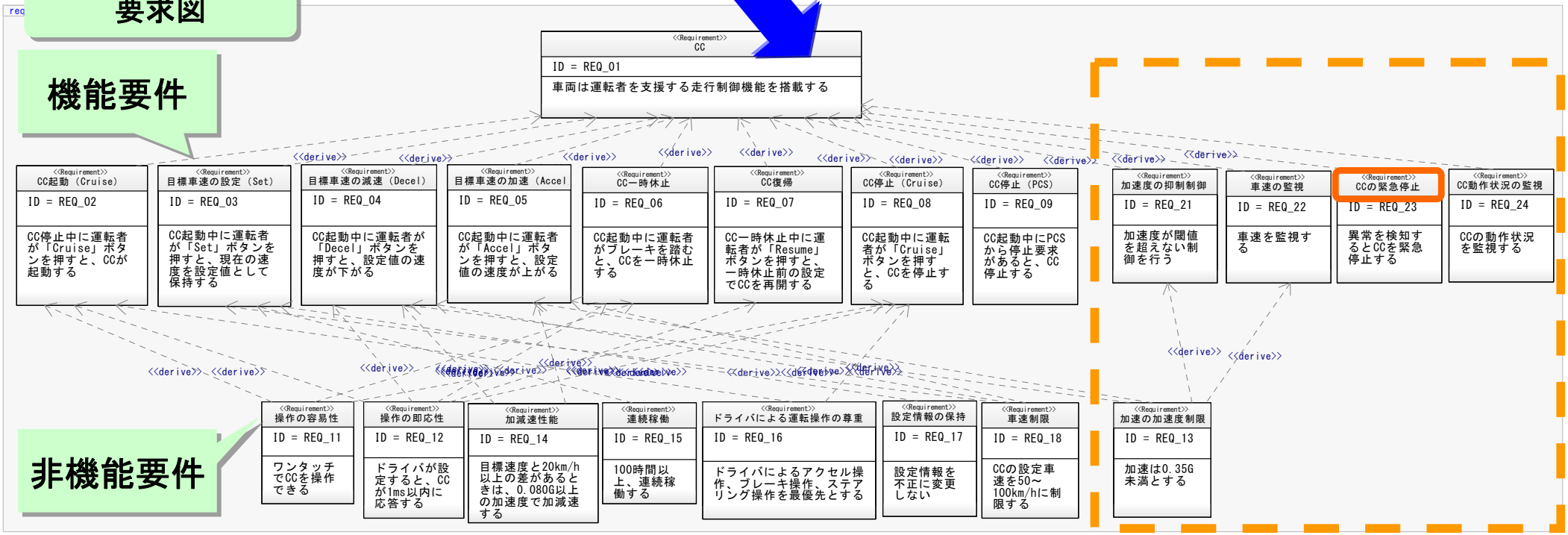
D-Case



機能安全に関する機能を参照し要求図を更新

要求図

機能要件



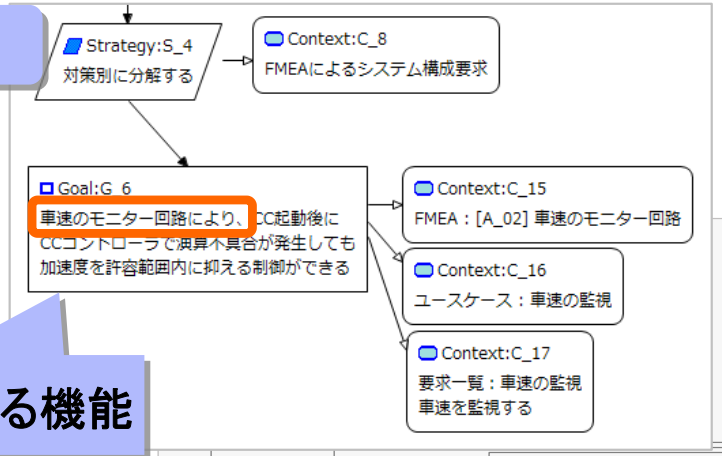
非機能要件

ISO 26262 関連部分を更新

システム設計 ～ ブロック定義図の更新

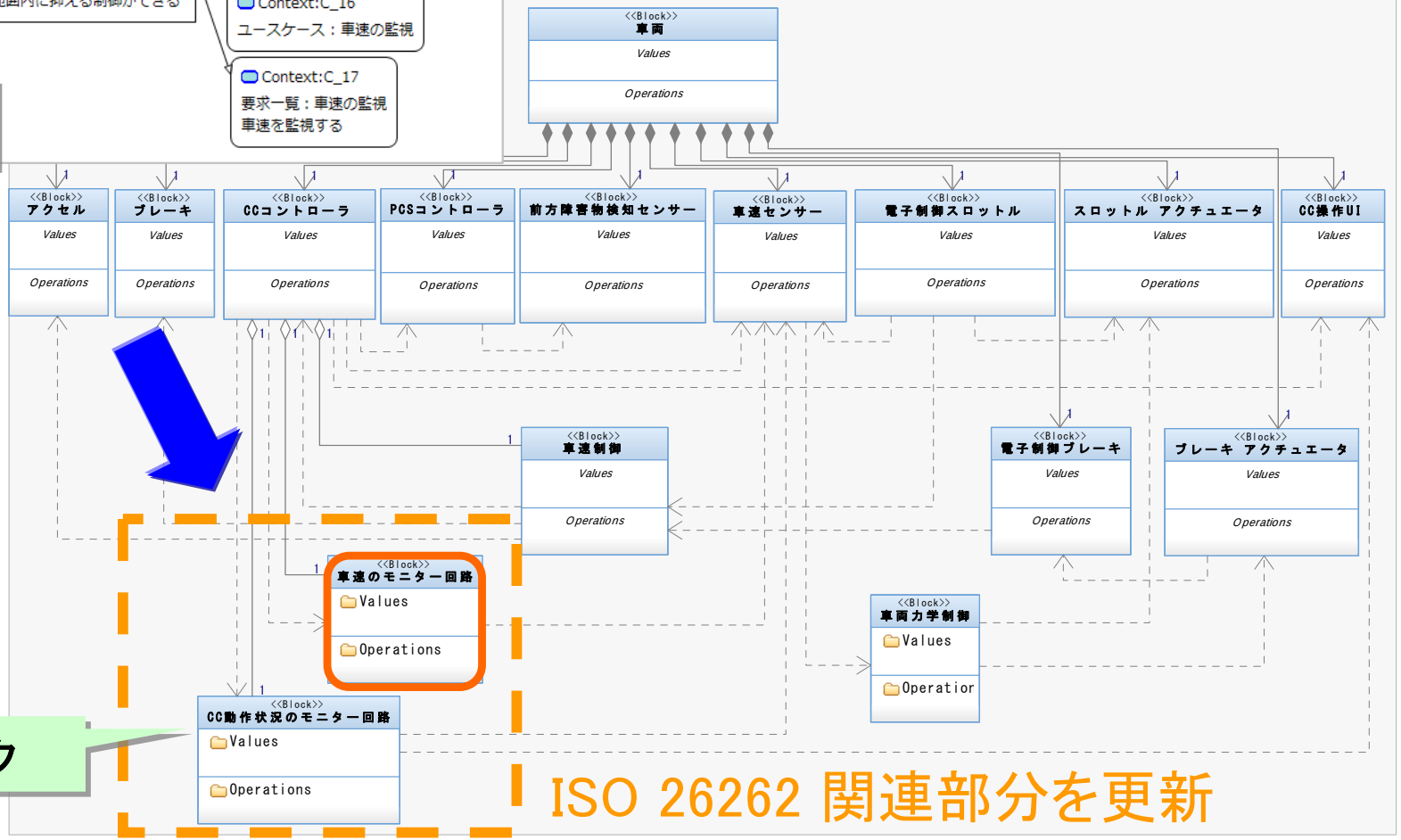
システム構成要求を基に、システム・アーキテクチャをブロック定義図に反映

D-Case



機能安全に関する機能

ブロック定義図



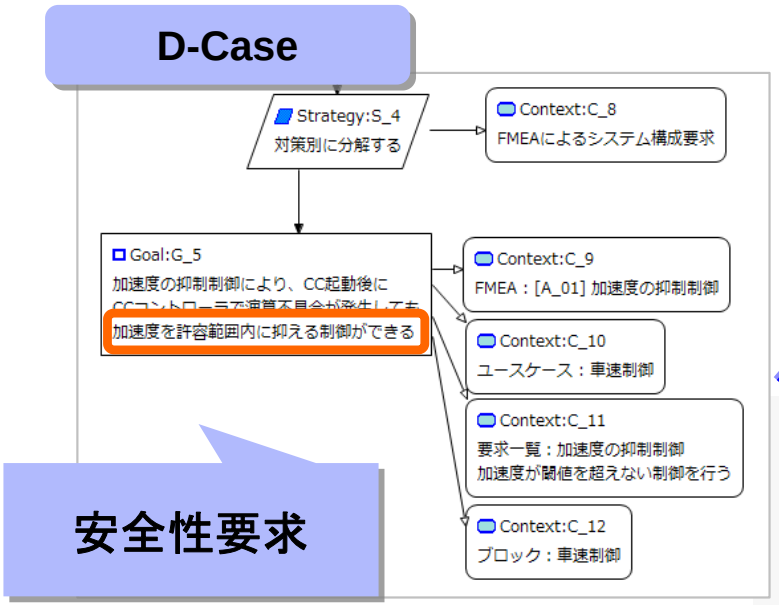
前機種のブロック定義図に
技術安全要求の検討により
導出されたブロックを追加

ブロック

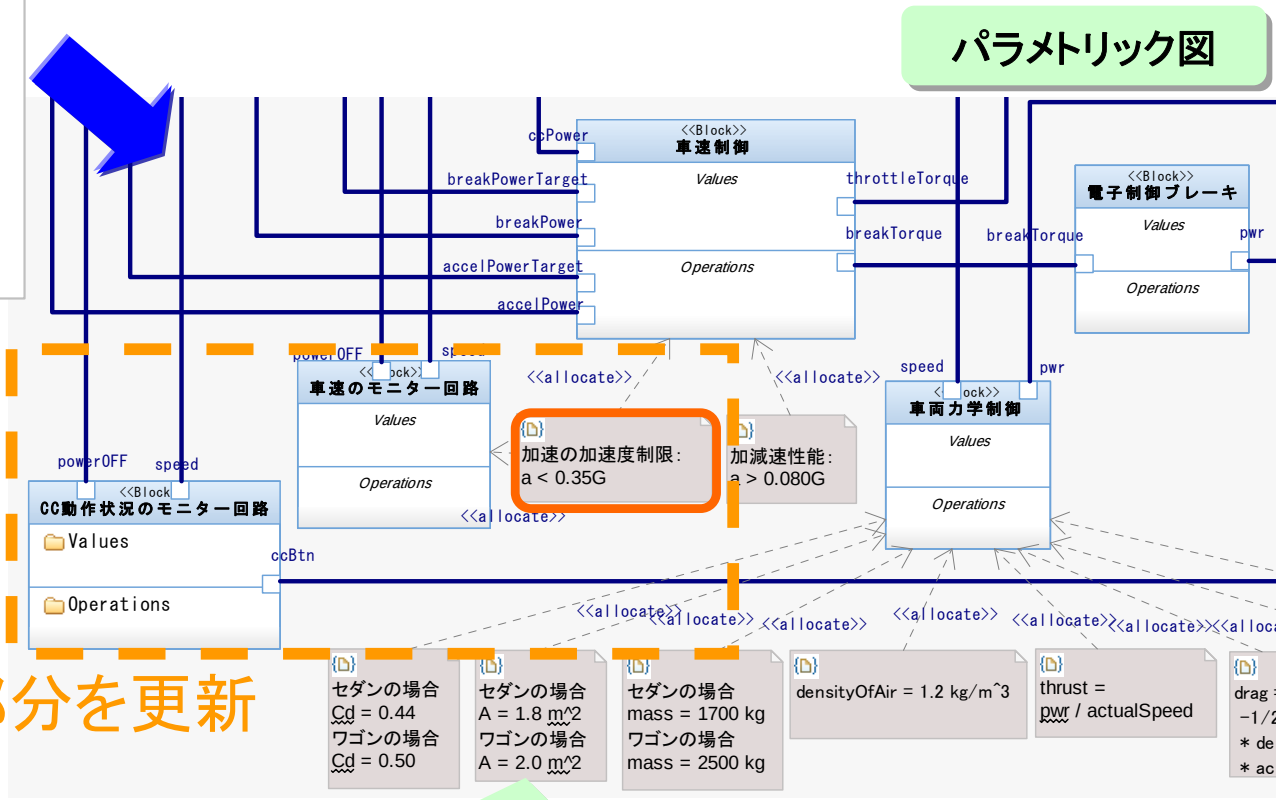
ISO 26262 関連部分を更新

システム設計 ～ パラメトリック図の更新

安全性要求を基にシステムの制約、関連する値と数式をパラメトリック図に反映



安全性要求から制約の必要性が判明する
パラメトリック図に制約を追加



ISO 26262 関連部分を更新

システム設計 ～ 内部ブロック図の更新

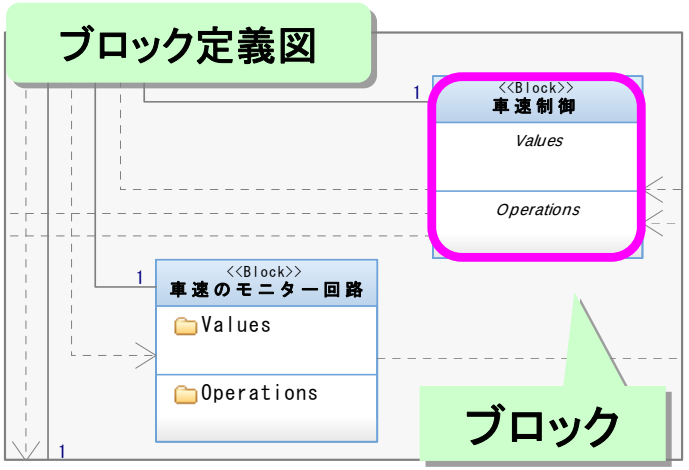
ブロック内部の構造を内部ブロック図で詳細化

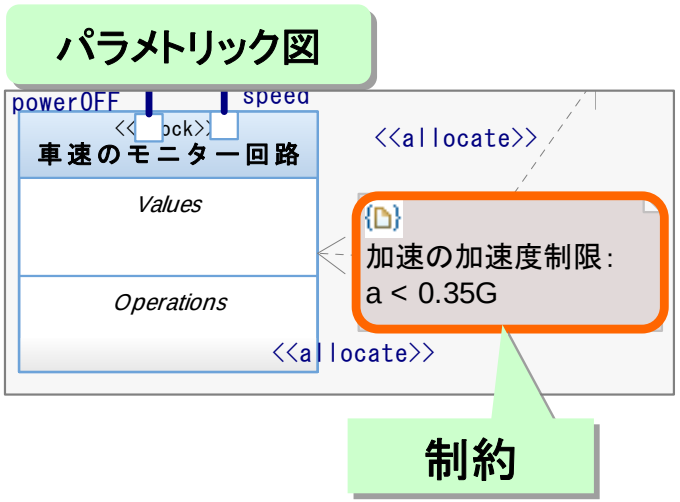
内部ブロック図

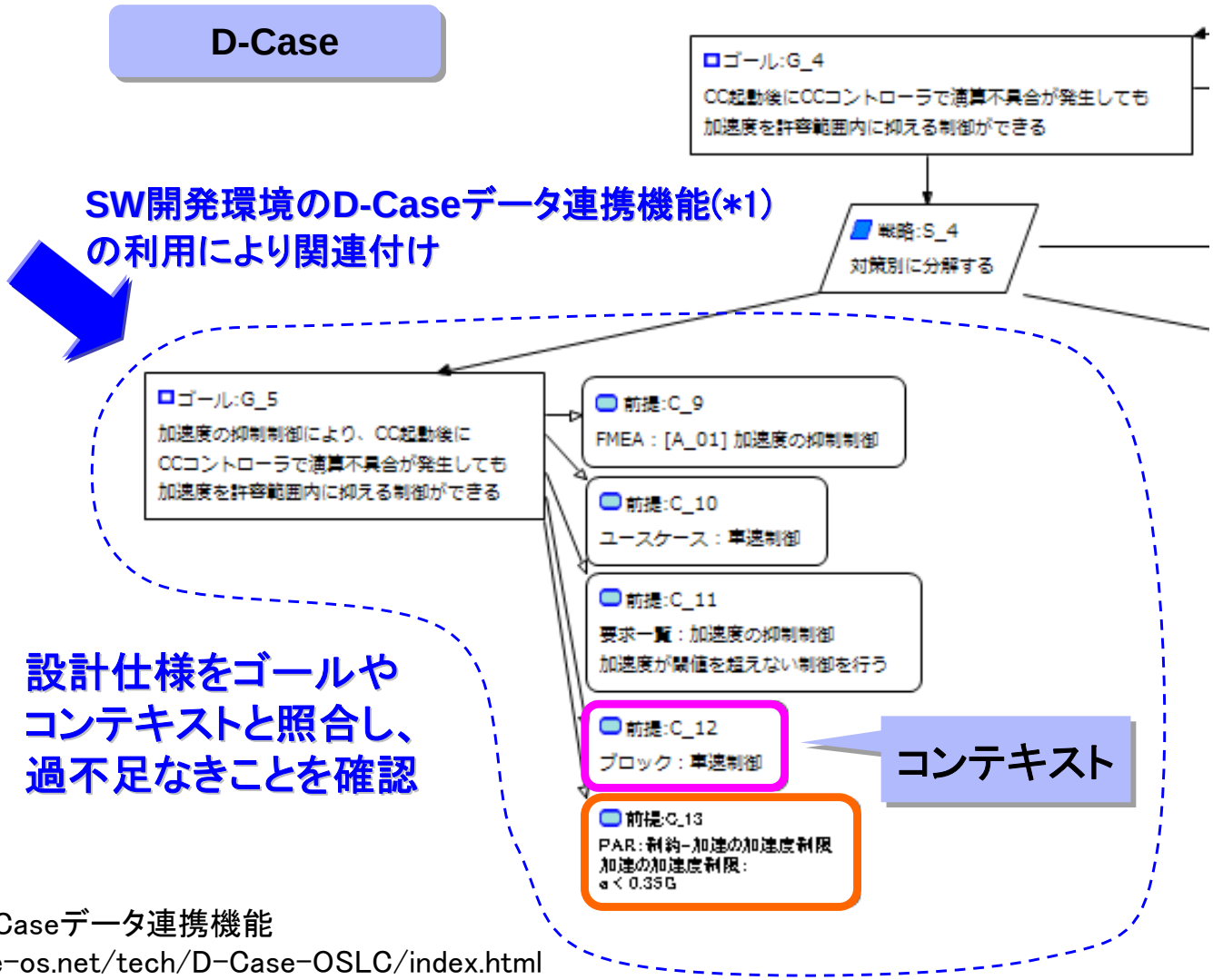
ブロック定義図と同様に、
D-Caseの中からシステムの
構成要素を抽出し、
ブロック内部の構造を
詳細化

システム要求と設計仕様との対応確認

ブロック定義図のブロックやパラメトリック図の制約をコンテキストに関連付け、設計仕様に過不足なきことを確認





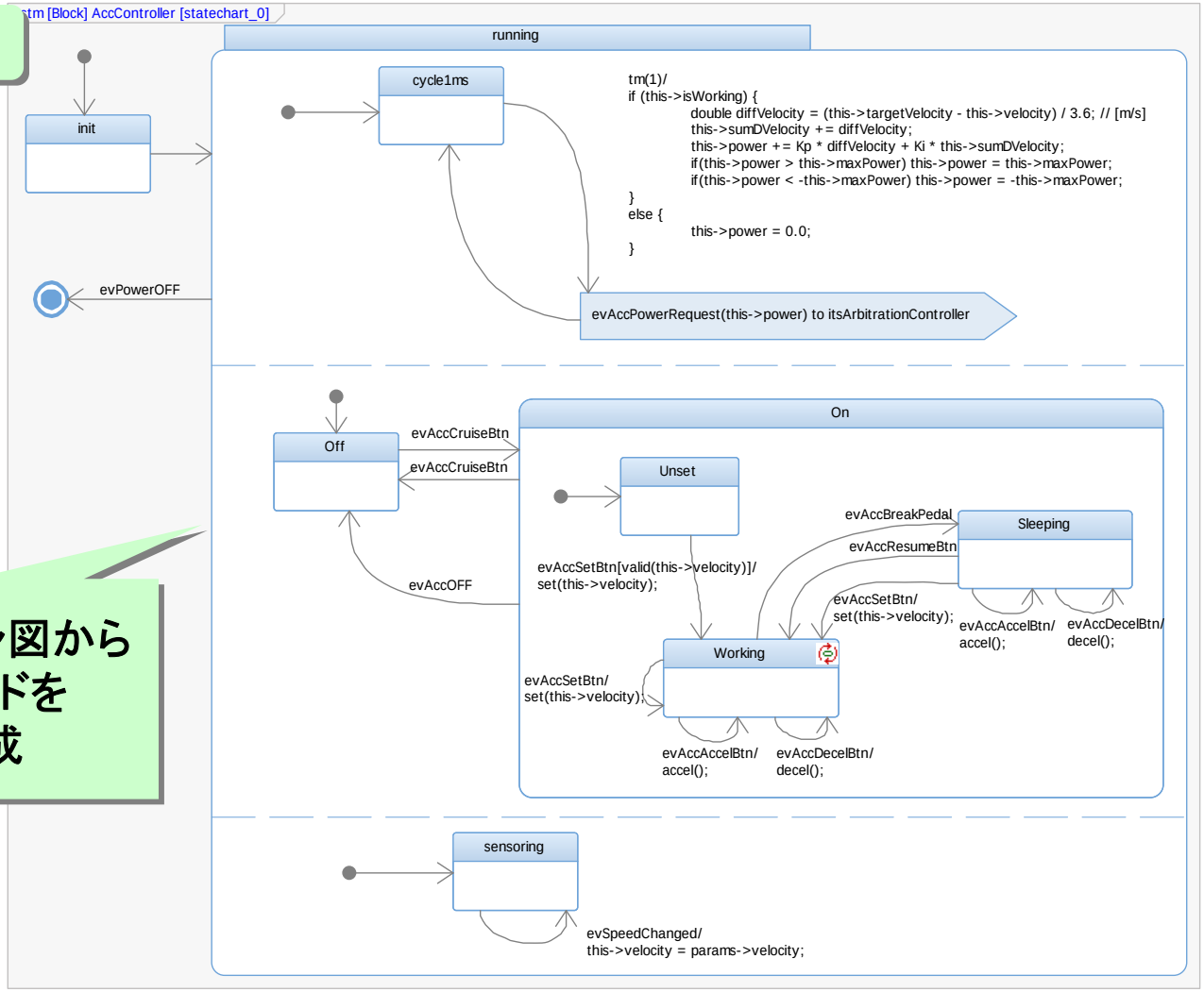


(*1) SW開発環境のD-Caseデータ連携機能
http://www.dependable-os.net/tech/D-Case-OSLC/index.html

S/W開発

ステートマシン図でシステムの動的な振舞を定義し、ソースコードを生成

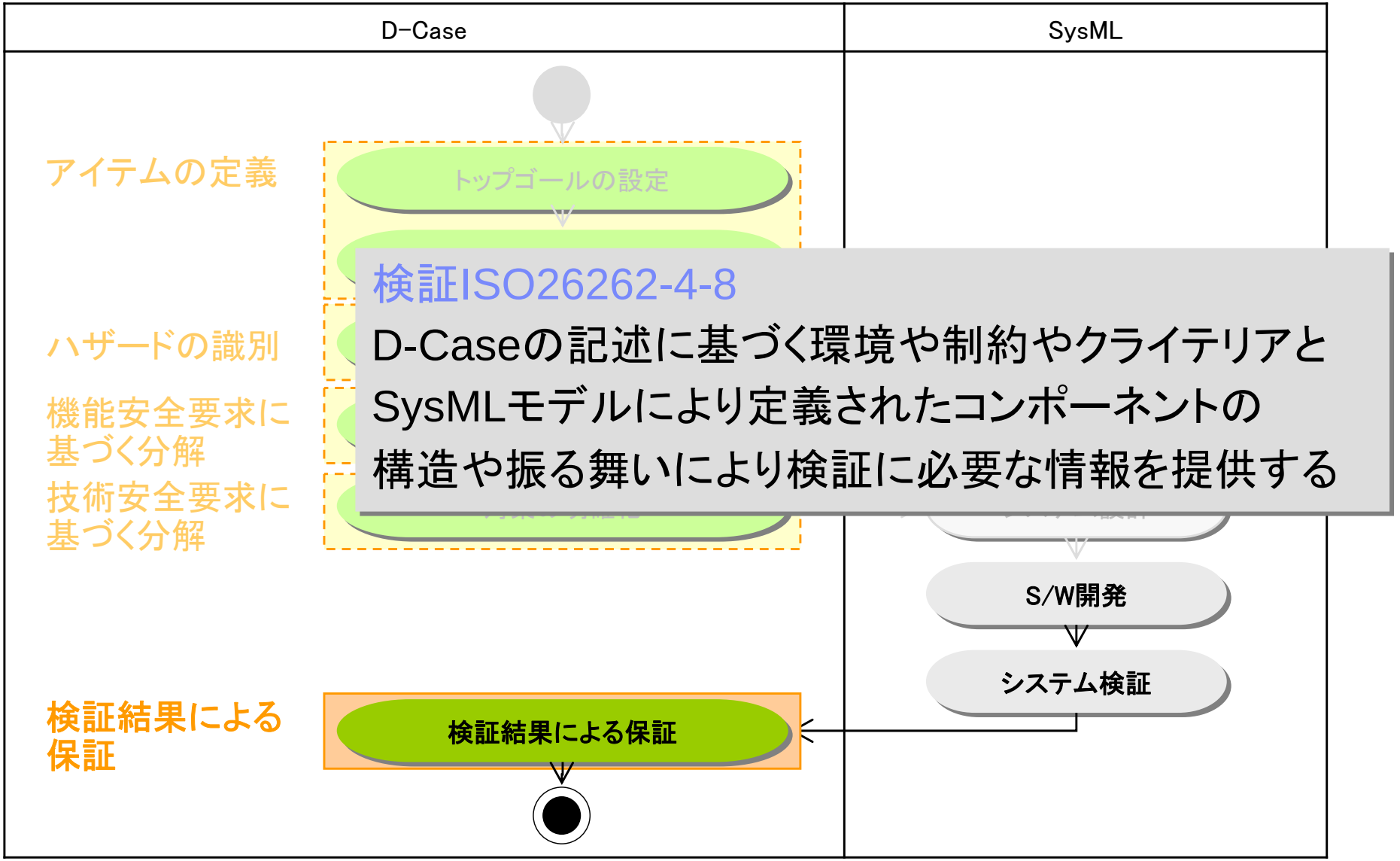
ステートマシン図



ステートマシン図から
ソースコードを
自動生成

開発フロー ～ 検証結果による保証

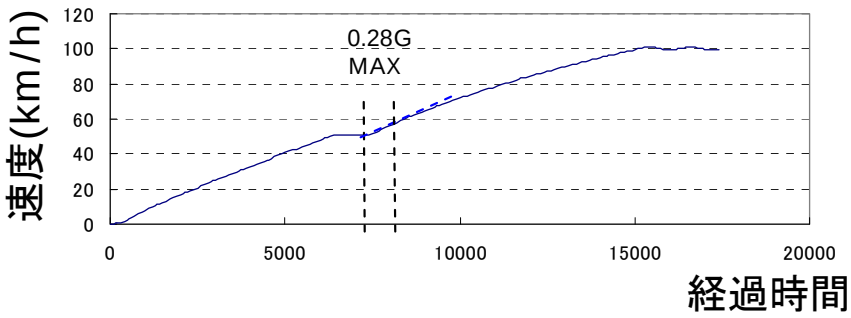
システム要求を満足することを明示



システム検証

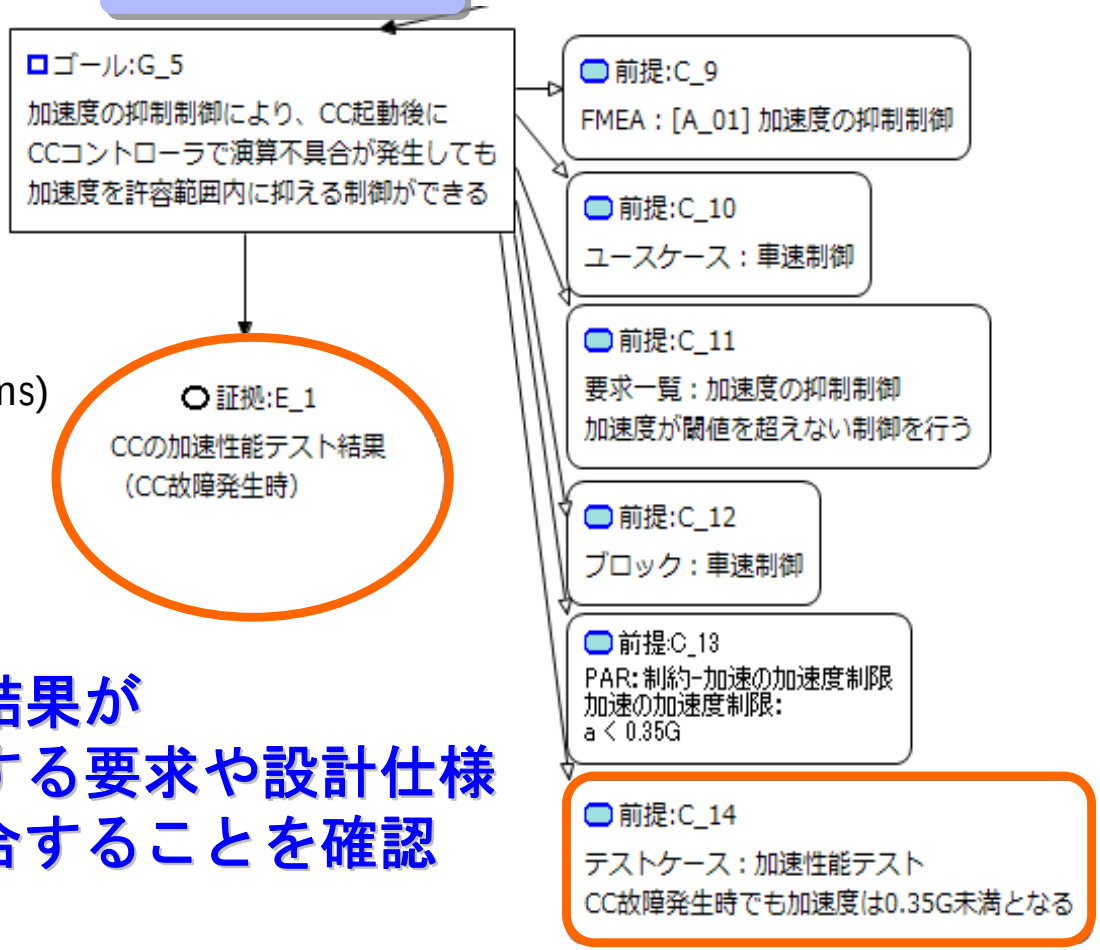
モデル・シミュレーションの結果をエビデンスとしてゴールに関連付け、検証結果と要求や設計仕様との関連性を明確化

シミュレーション結果



検証結果：クリア！

D-Case



検証結果が関連する要求や設計仕様と整合することを確認

まとめ

**開発の上流工程から下流工程に至るまで一貫した
ディペンダビリティを実現する開発手法とガイドの提供**

ディペンダブルなシステムの開発

D-CaseとSysMLモデルとの連携により実現

- システム要求を過不足なく導出
- SW開発環境のD-Caseデータ連携機能の利用により設計仕様を正確に導出
- 検証結果と要求や設計仕様との関連性を明確化

ガイド

- D-CaseとSysMLのモデリングフロー
- D-Caseノードの記述スタイル
- D-CaseとSysMLモデルの情報連携

END OF PACKAGE