D-Case Driven Development with SysMLの試み

日本アイ・ビー・エム株式会社　グローバル・ビジネス・サービス

豊田　学

目次

はじめに

組込みシステム開発における問題

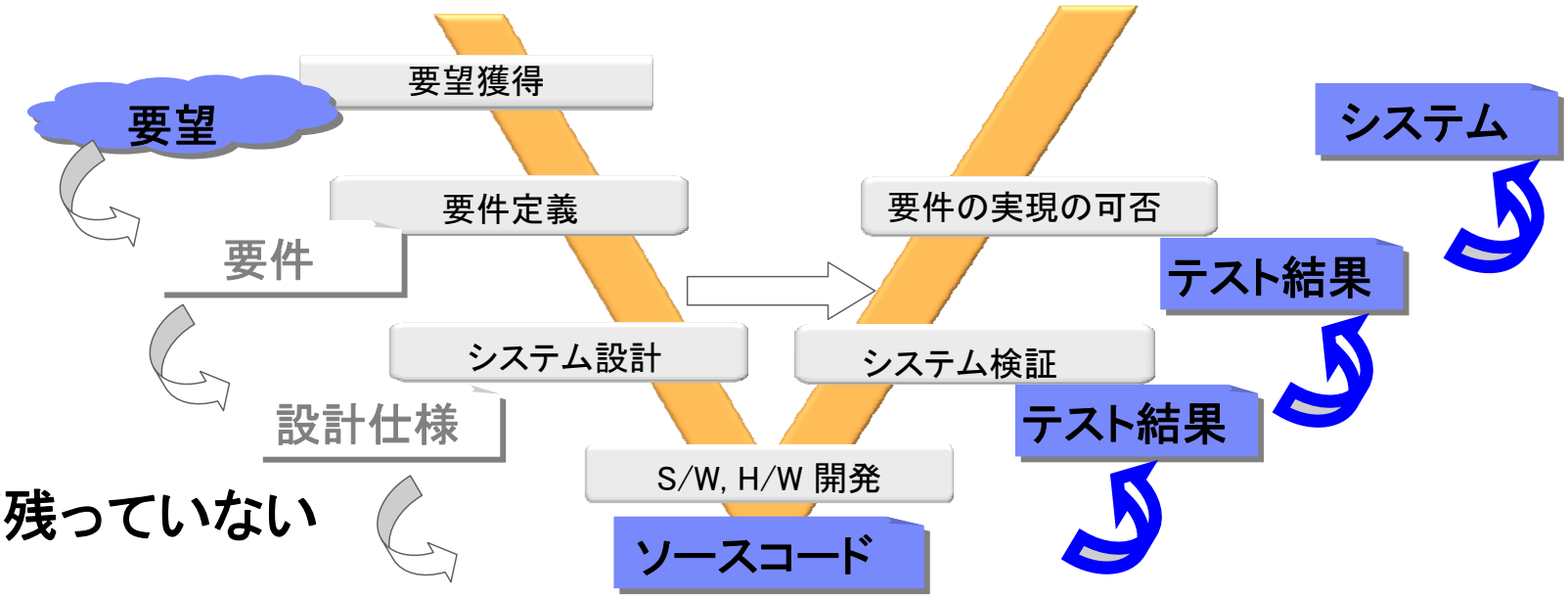
組込みシステム開発における問題

開発の現状 (*1)

- ソースコードをベースとして開発を行っている
- 開発スケジュールに間に合わせ、開発コストを削減するため、上流工程の仕様書が作成されない、または更新されない

課題

上流工程の正確な仕様書がないため、システムがディペンダブルであることの証跡が残っていない



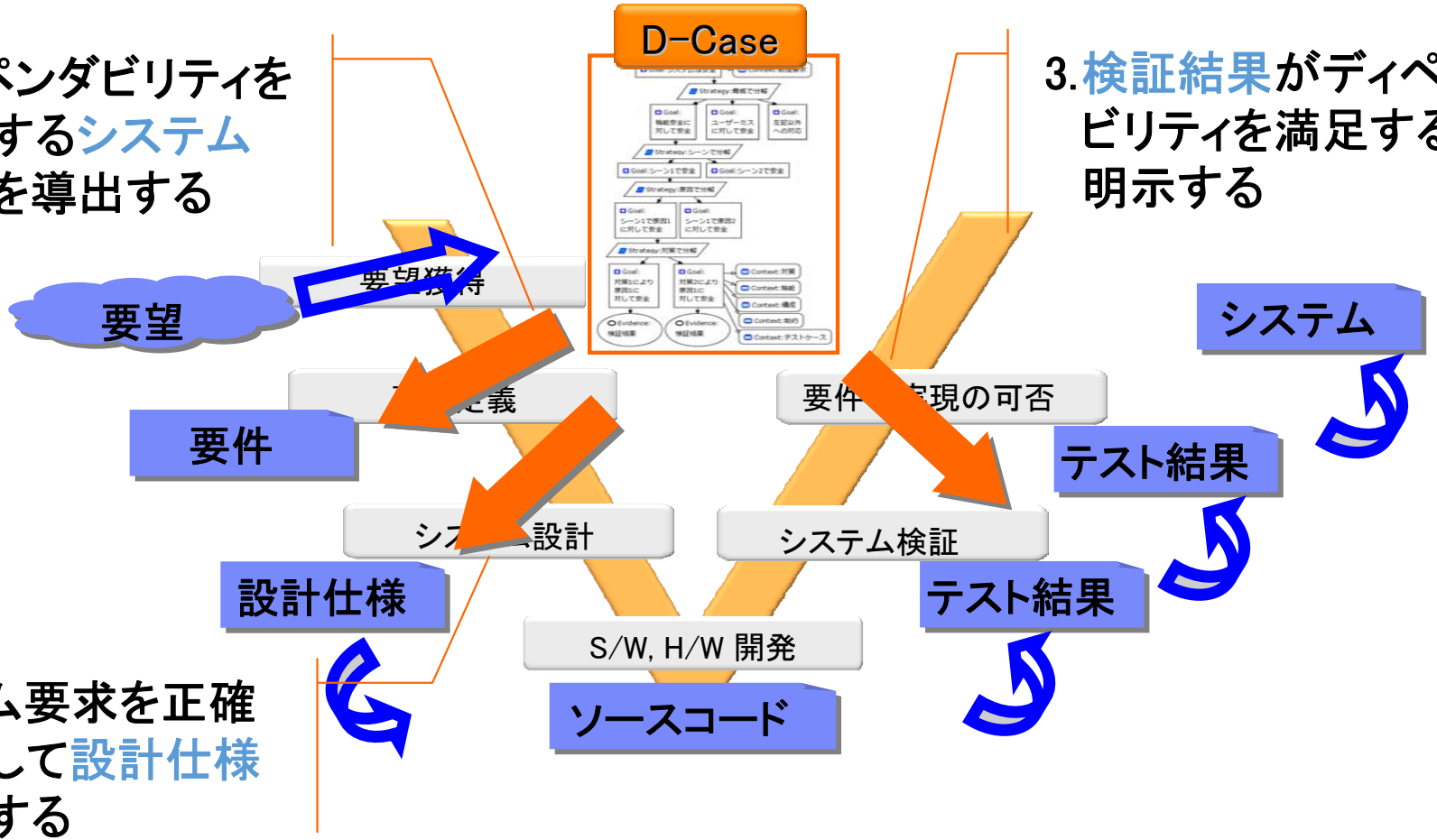
証跡が残っていない

(*1)参考資料: A Development Method Proposal for Embedded Software Based on Model-Based Development, with Result Feedback through Issue Measurements of Derivational Development, ICEUC2012

解決のアプローチ
～ D-Case Driven Development with SysML手法

D-CaseとSysMLモデルとの一貫した連携により、開発の上流工程から下流工程に至るまで、ディペンダブルなシステムを実現する

1.ディペンダビリティを満足するシステム要求を導出する



2.システム要求を正確に反映して設計仕様を導出する

3.検証結果がディペンダビリティを満足することを明示する

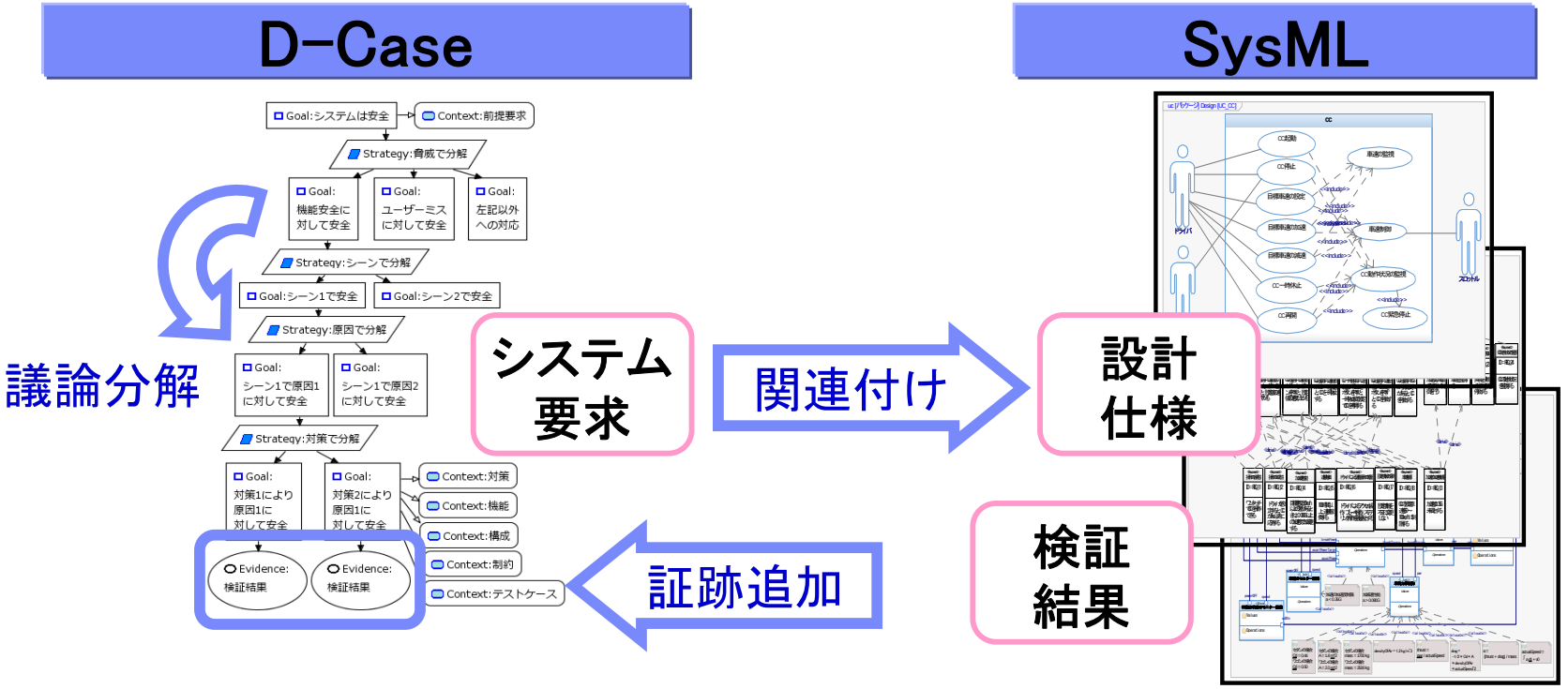
問題を解決する開発手法

D-Case Driven Development with SysML

D-Caseによる課題の解決

D-Case Driven Development with SysML手法

1. D-Caseの議論分解によるディペンダビリティを満足する要求の導出
2. D-Case上での設計仕様の関連付けによる正確な設計仕様の策定
3. D-Case上での証跡追加による検証結果の確認



D-Caseによる要求の導出

D-Caseの議論分解を通して、ディペンダビリティを満足するシステム要求を過不足なく導出する

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

↓

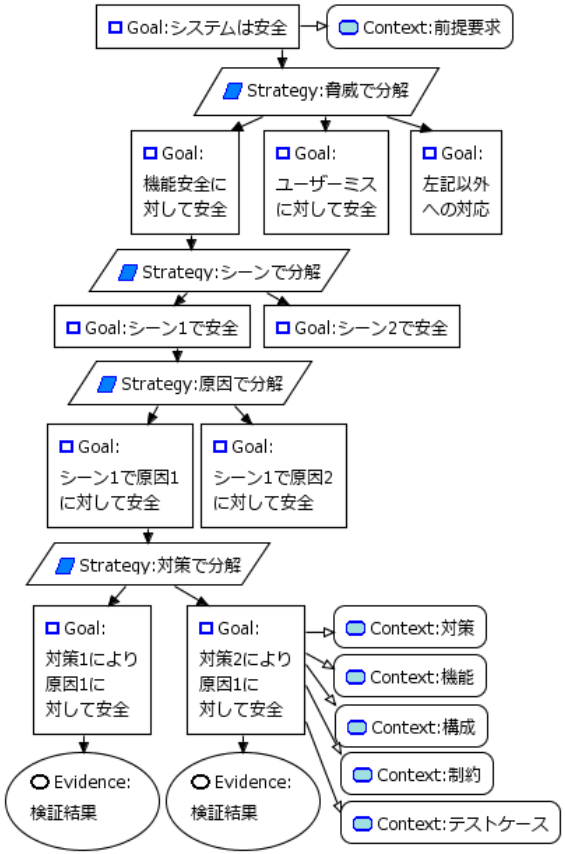
脅威の発生シーンの明確化

↓

原因の明確化

↓

対策の明確化

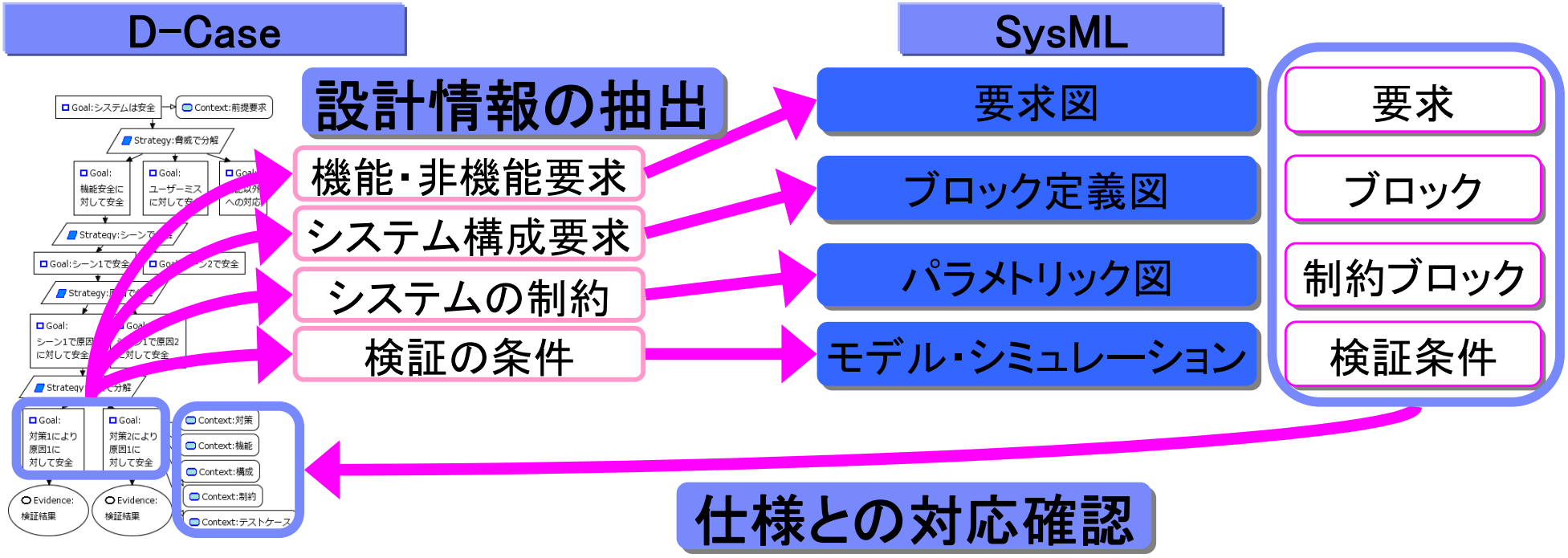


D-Caseに基づく正確な設計仕様の策定

D-Case上にシステム要求や設計仕様を関連付けることにより、
システム要求や他の設計仕様と整合した設計仕様を策定する

・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

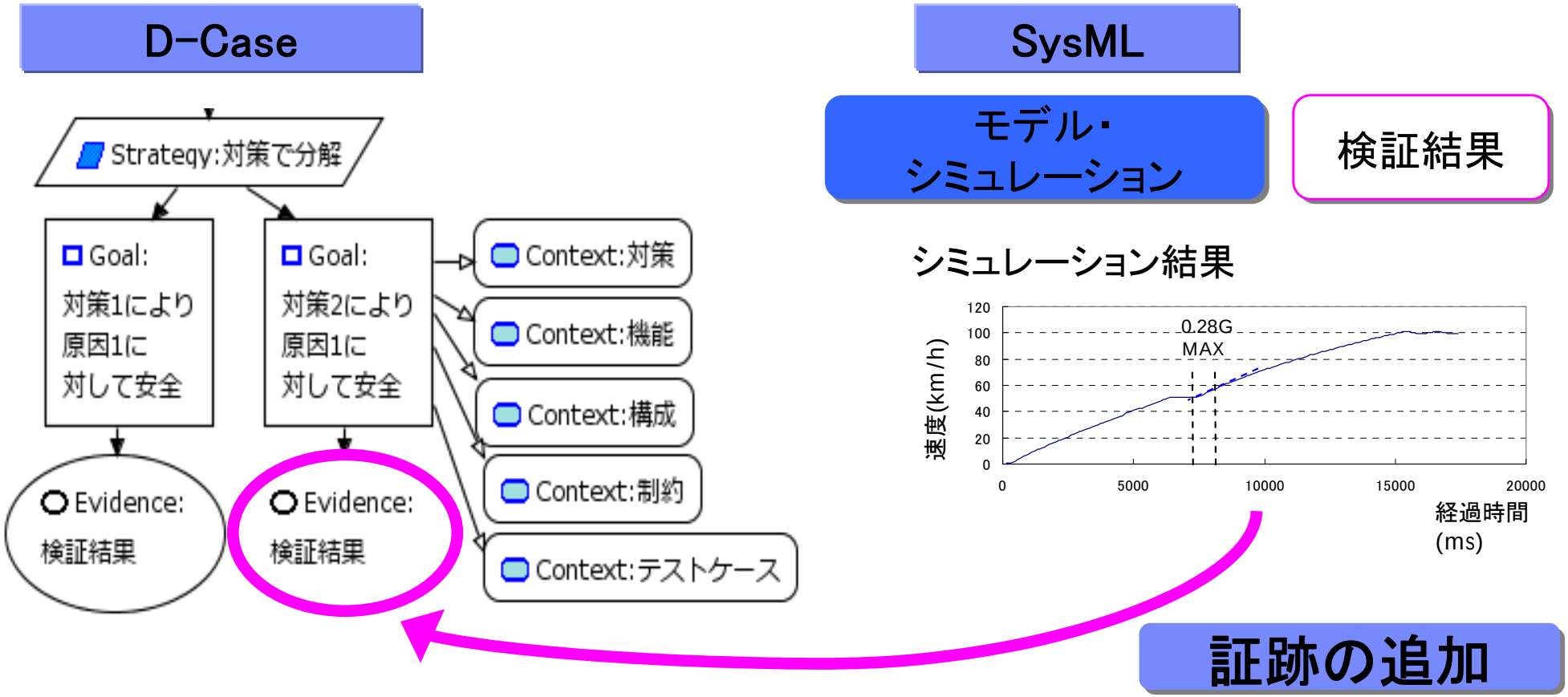
・導出した設計仕様をディペンダビリティに基づく要求と照合し、過不足なきことを確認する



D-Caseでの検証結果の確認

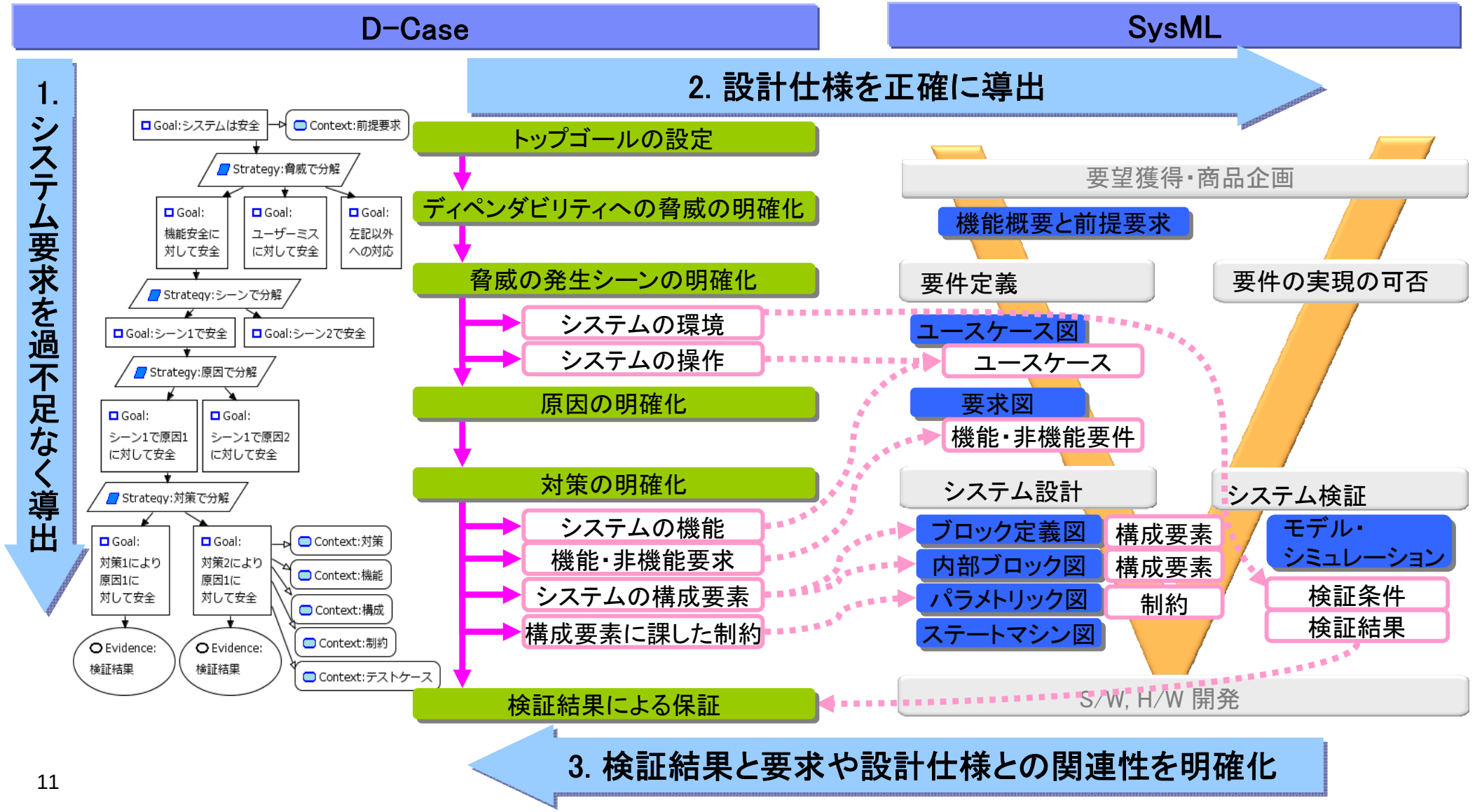
D-Case上での証跡追加により、検証結果と要求や設計仕様との関連性を明確にする

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



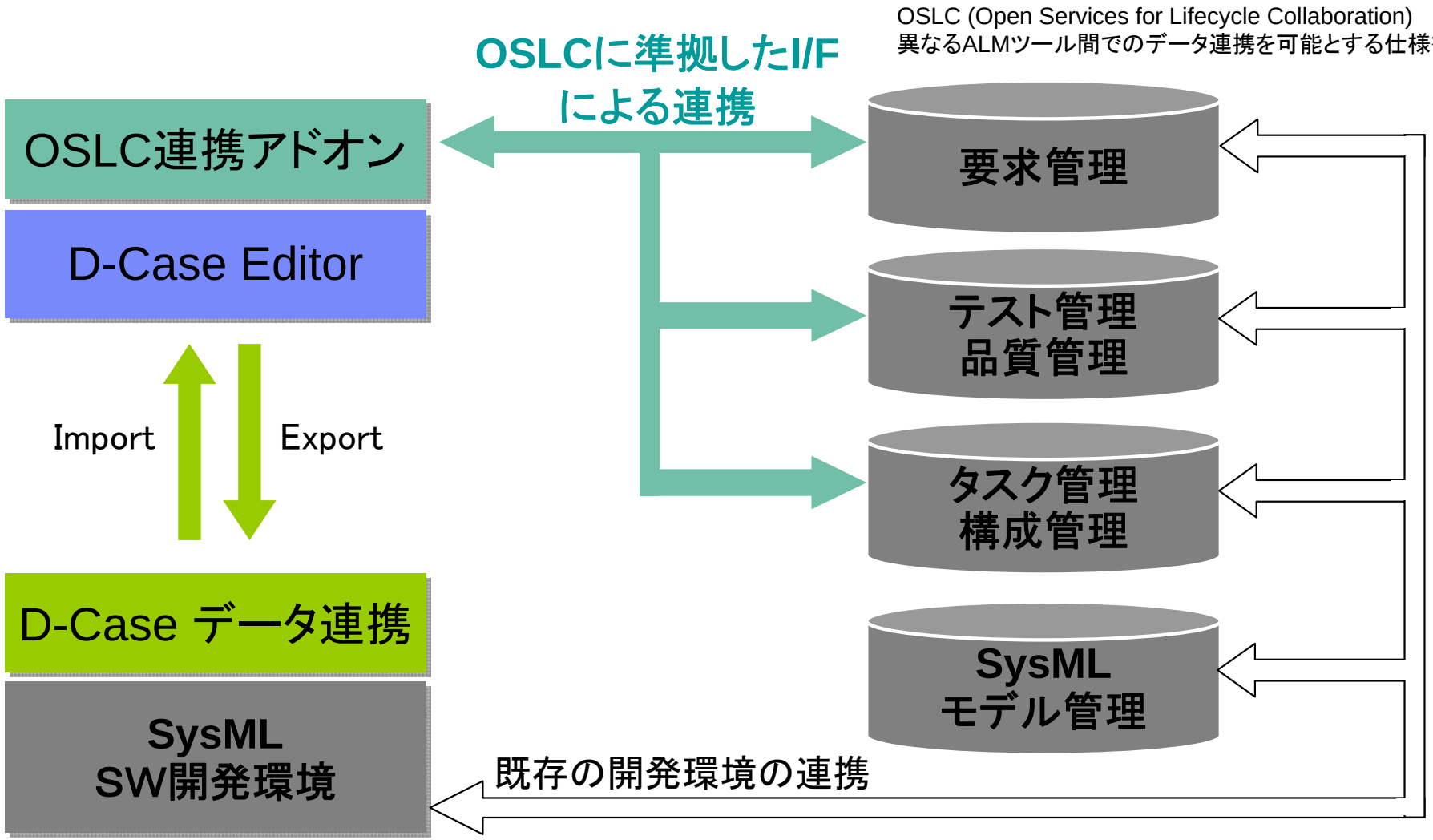
D-Case Driven Development with SysML の全体像

D-CaseとSysMLモデリングとの連携によるシステム開発



D-Case Driven Development with SysML を実現する開発環境

D-CaseとSysMLモデリングとの連携によるシステム開発環境



具体例

ISO26262に準拠した車載システムへの適用

ISO26262に準拠した車載システムへの適用

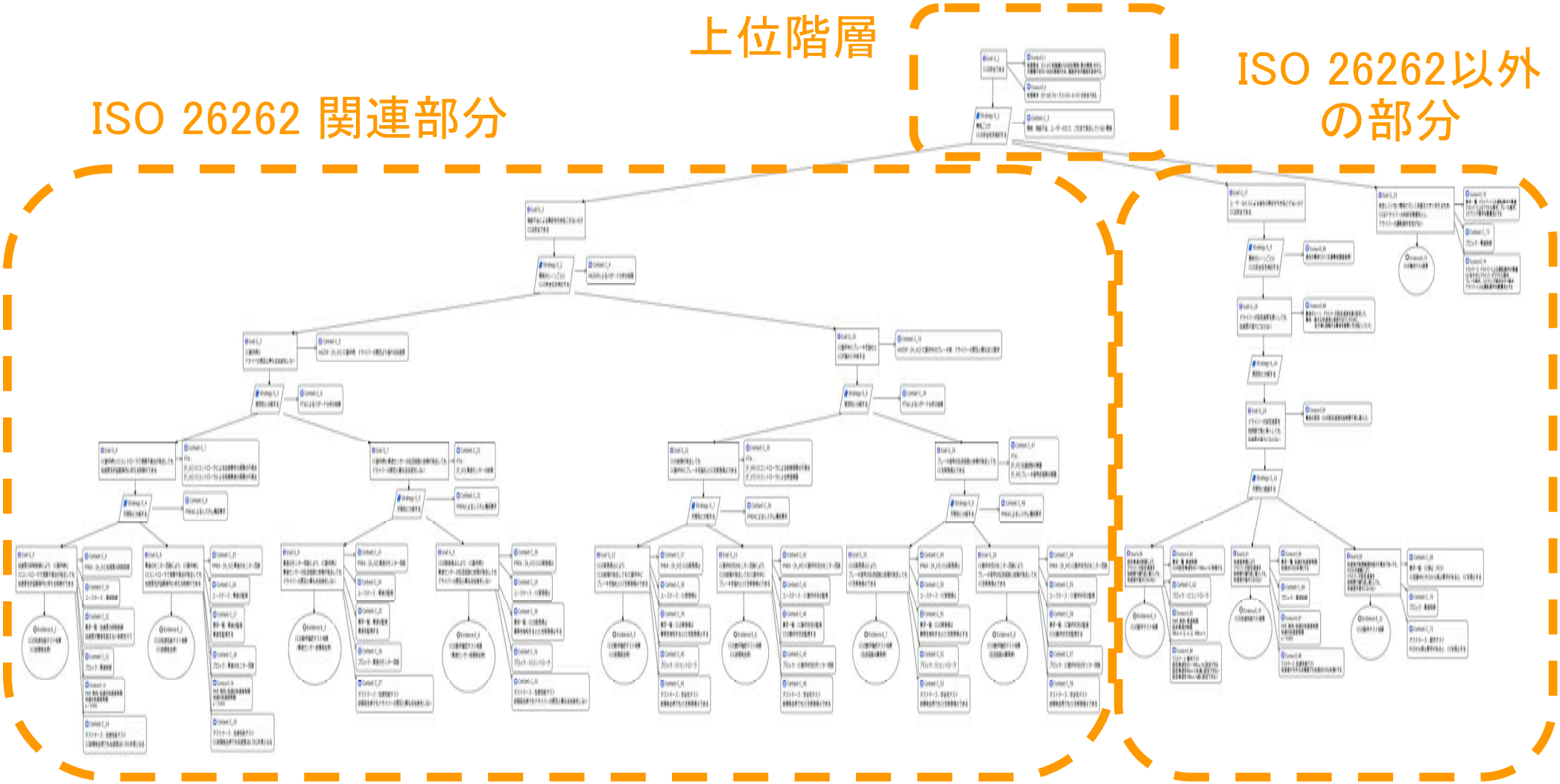
本手法はISO26262の安全ライフサイクルと対応

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	
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	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

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

例. 車載クルーズコントロール(CC)システムの安全性



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

D-Case

トップゴールの設定

安全性の観点に基づき
トップゴールと前提を設定

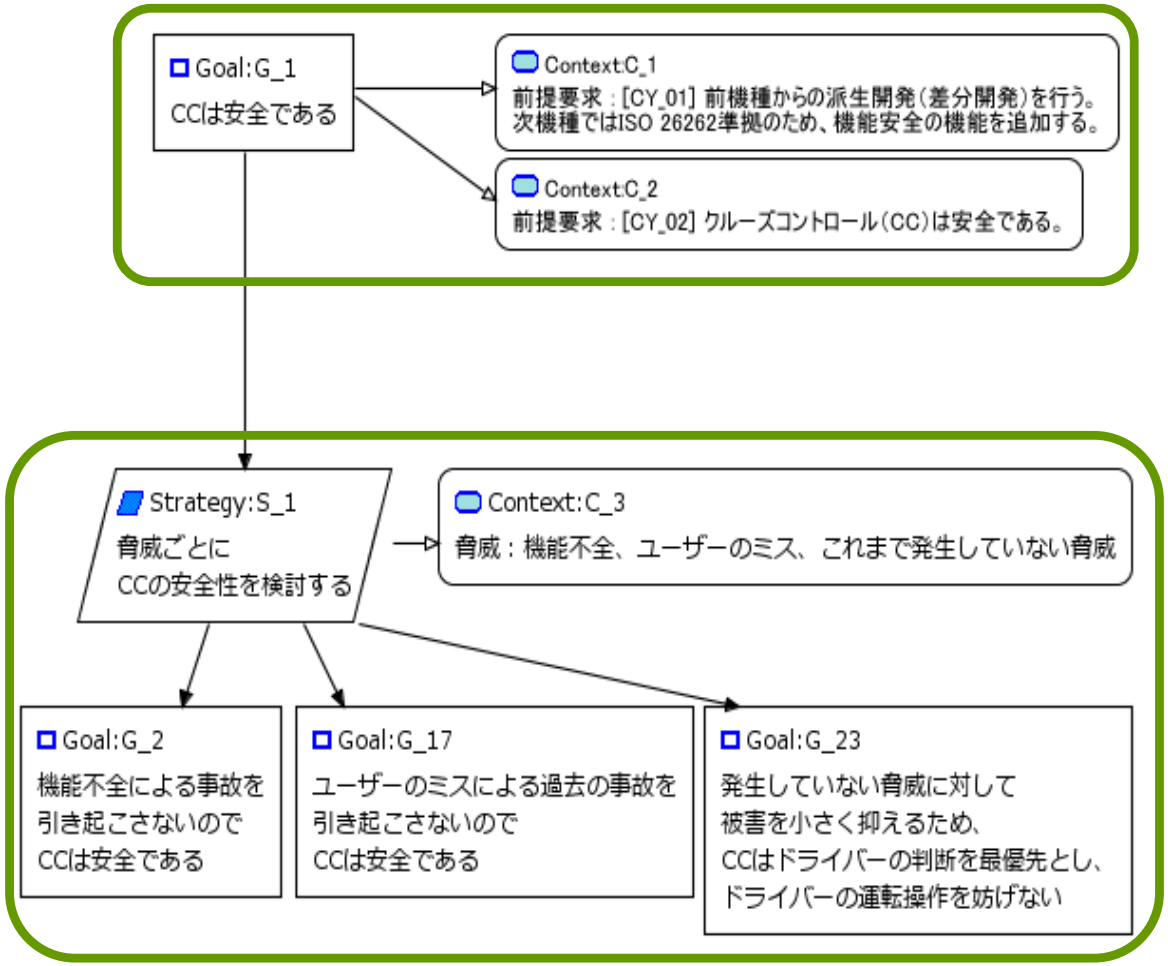
システムが満たすべき性質と
前提を設定

ディペンダビリティへの脅威の明確化

安全性を阻害する
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



開発フロー ～ ハザードの識別

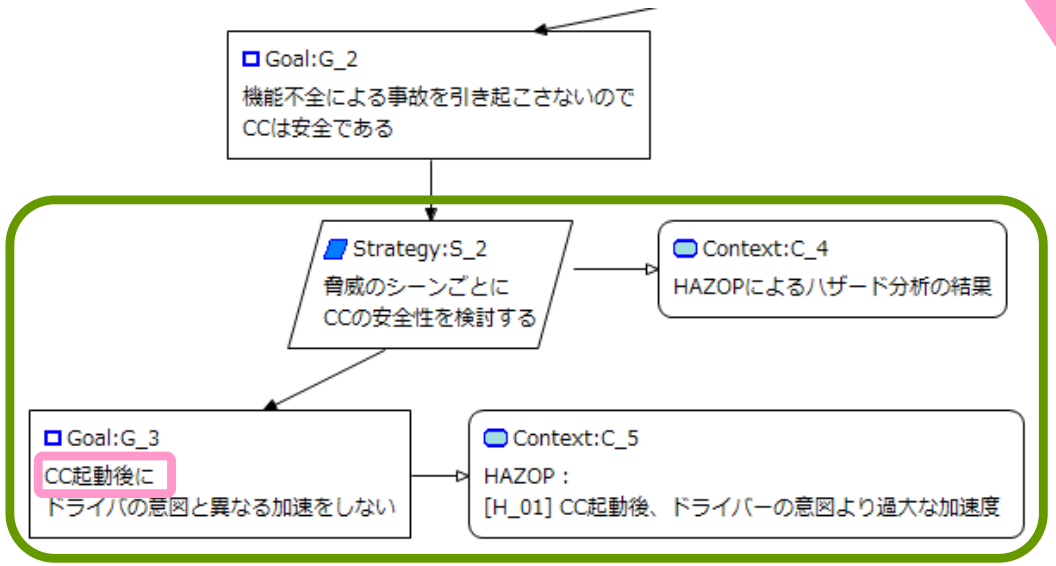
安全性を阻害するすべてのシーンを抽出

D-Case

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



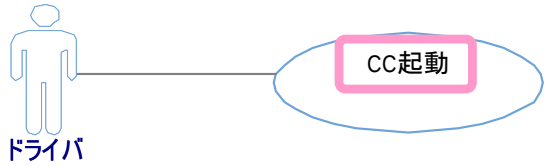
反映

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

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

D-Case

原因の明確化

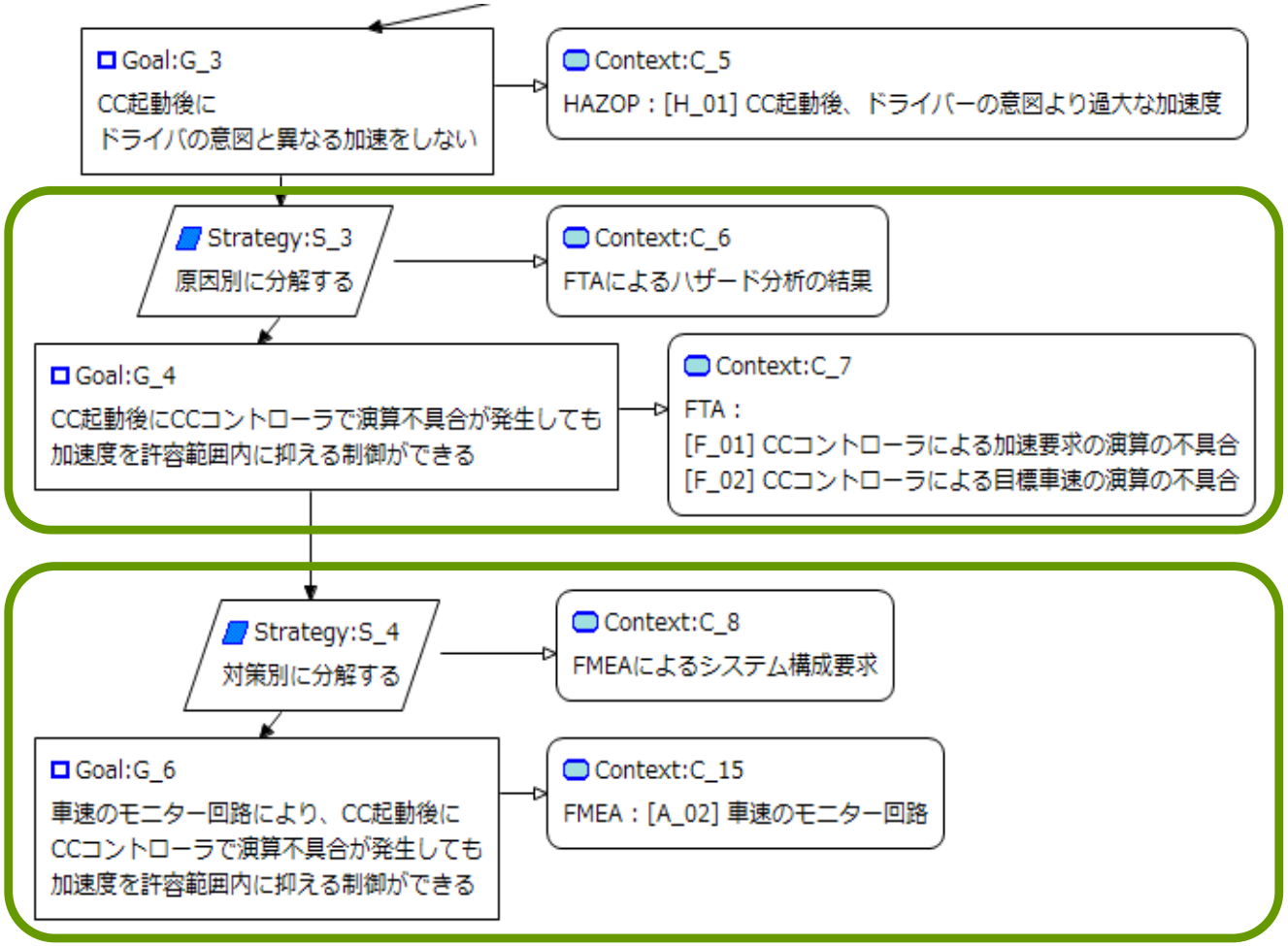
対策を立てられる粒度に至るまで原因を深堀り

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



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、要件の導出

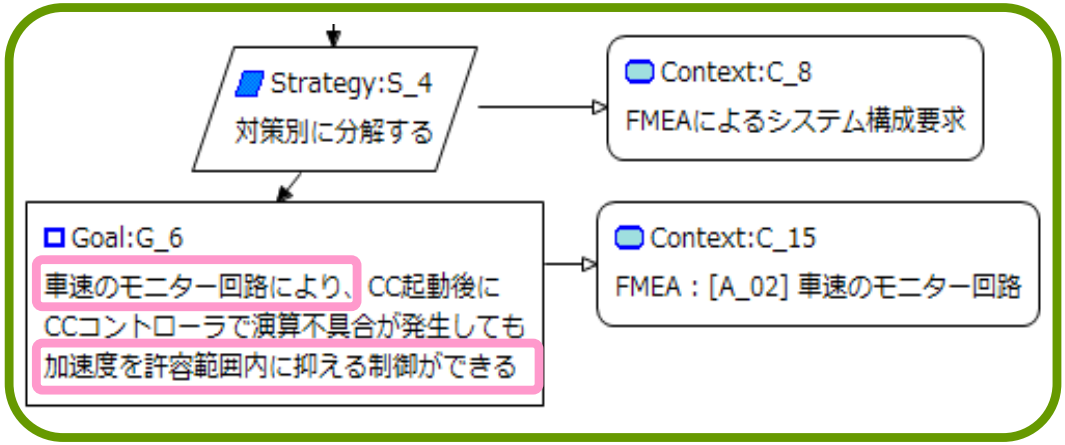
D-Case

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

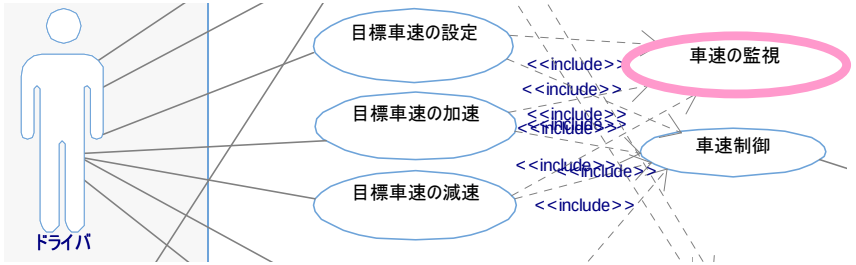


SysML

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

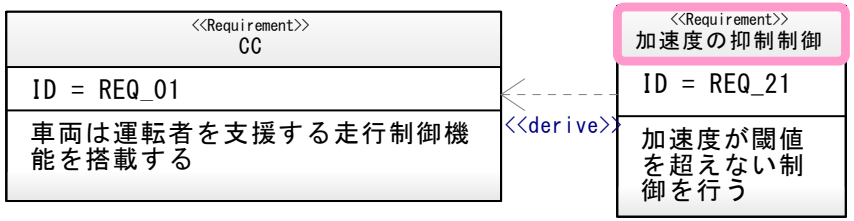
機能要求を基にユースケースを更新



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



開発フロー ～ 技術安全要求に基づく分解

D-Caseの記述を基にSysMLモデルを更新

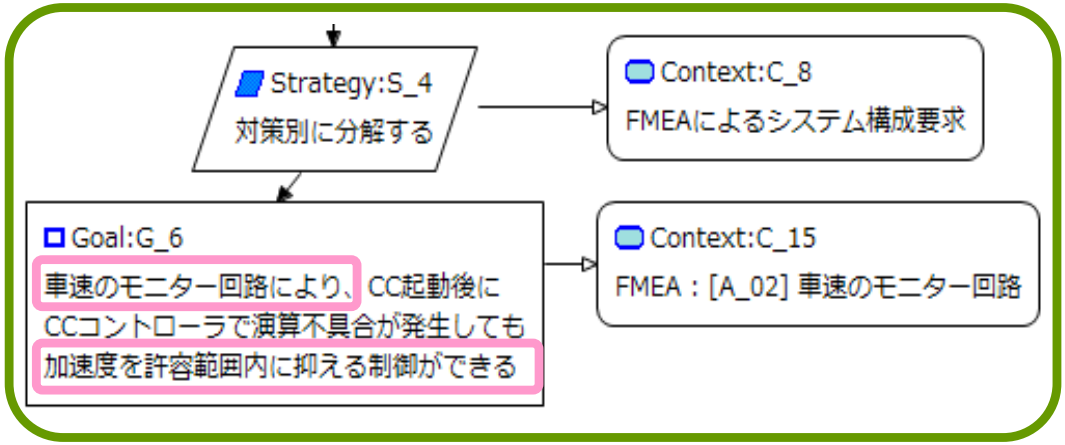
D-Case

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



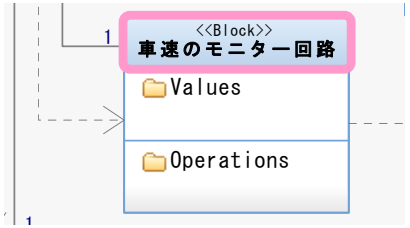
SysML

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

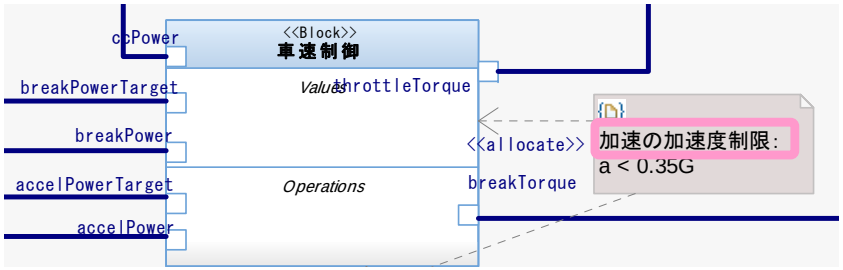


パラメトリック図

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



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

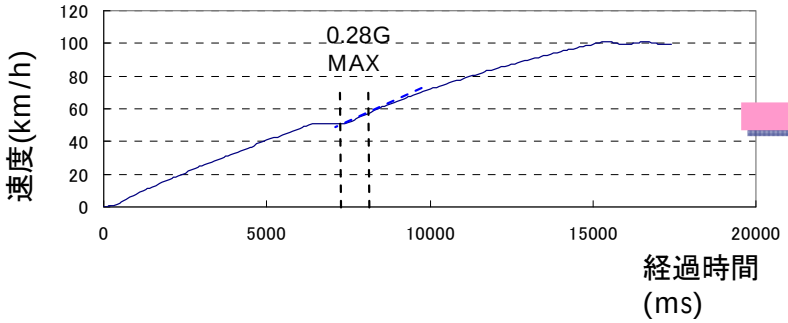
SysML	D-Case

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

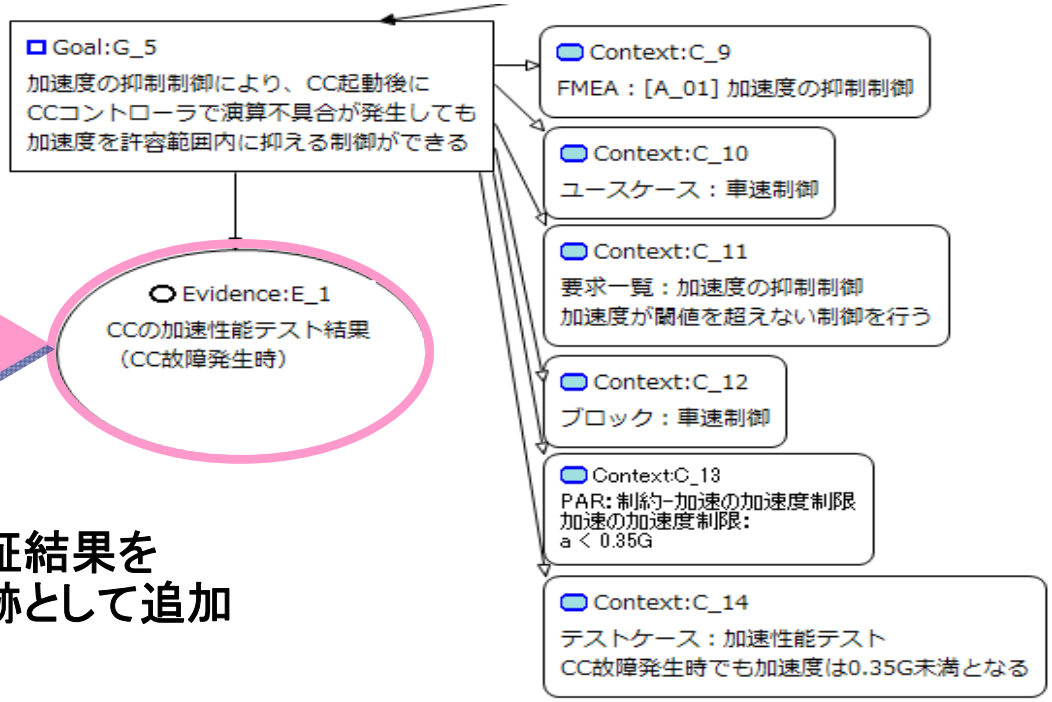


検証結果を
証跡として追加

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



まとめ

- D-Case Driven Development 手法を実現
- D-Case と SysML 開発環境とのデータ連携を実現
- 具体例として本手法を車載システム開発に適用

今後の展開

- 本開発手法の開発現場での実証

END OF PACKAGE