システム理論に基づいたモデリングと分析手法を用いたCo-simulationのシナリオ作成手法

岩田 紘成[1,a)]　日下部 茂[2,b)]　荒木 啓二郎[1,c)]　大森 洋一[1,d)]

概要：組込みシステム開発において，ハードウェアに対するプラントモデルとソフトウェアに対するコントローラーモデルを協調的にモデル化し (Co-model)，そのシミュレーション（Co-simulation）を行うモデルベース開発手法が提案されている．その支援ツールとして，形式的仕様記述言語 VDM-RT によるコントローラモデルと，ボンドグラフによるプラントモデルを対象とした Crescendo Tool が提案されている．我々は，形式手法を実際の開発プロセスで効果的に行うための方法論について研究を行っており，本稿では，前述のような形式的仕様を用いた Co-simulation の効果的な導入について議論する．

Generating Senarios for Co-simulation using Systems-Theoretic Modeling and Analysis STAMP/STPA

Hironari Iwata[1,a)]　Shigeru Kusakabe[2,b)]　Keijiro Araki[1,c)]　Yoichi Omori[1,d)]

Abstract: In developing embedded systems, there proposed an emerging approach in which we develop and simulate a collaborative model cosisting formal models of software for embedded systems and continuous models of physical plant. The Crescendo tool supports such a model-based approach with formal models in VDM-RT and continuous models in Bond Graph. Using formal methods such as VDM is an approach to effectively developing high quality software, and we have been investingating effective introduction of formal methods in actual development process.

1. はじめに

近年，組込みシステムはシステムの高度化・多機能化が進み，開発後期の統合段階で問題が発覚すると，大きな手戻りが発生することになる．このような背景から，組込みシステム開発の初期段階から領域間の協調的開発を行うことへの要求が高まっている．

組込みシステム開発におけるハードウェアとソフトウェアの協調的開発の方法の一つに Collaborative-model(Co-model) を用いたモデルベース開発手法がある．Co-model はハードウェアに対するプラントモデルとソフトウェアに対するコントローラーモデル, およびそれら 2 つのモデルの間の相互作用に関する契約（Contract）からなるモデルである．Co-model を用いた協調的シミュレーション Collaborative-simulation（Co-simulation）を行うことで設計の分析を行うことができる．そのような開発手法の支援ツールの一例として，形式仕様記述言語 VDM-RT によるコントローラモデルと，ボンドグラフで表現されたプラントモデルによる Co-model を対象とした Crescendo Tool が提案されている．しかし, 特に安全性要求に関して, 設計段階の Co-simulation で対象システムのどのような性質を検証すれば良いかは明らかでない.

我々は，形式手法を開発プロセスで効果的に実施するための方法論について研究を行っており，本稿では，Co-simulation を対象にその効果的な導入について議論する．

本稿では, 第 2 節で本研究で組込みシステムの開発プロ

[1] 九州大学, Kyushu University, 744 Motooka Nishi-ku Fukuoka 819-0395, Japan
[2] 長崎県立大学, University of Nagasaki, 1-1-1 Manabino, Nagayo-cho, Nagasaki 851-2195, Japan
[a)] h.iwata@nanotsu.ait.kyushu-u.ac.jp
[b)] kusakabe@sun.ac.jp
[c)] araki@csce.kyushu-u.ac.jp
[d)] yomori@ait.kyushu-u.ac.jp

セスとして想定した組込みシステム開発の参照的なプロセスである ESPR（Embedded System development Process Reference）について述べる．第 3 節では，Co-simulation をサポートするツールである Crescendo Tool について述べる．第 4 節では, ESPR を用いた開発への Crescendo Tool の対応付け結果と問題点について述べる．第 5 節では, 問題改善のための提案手法と, 手法に用いるハザード分析手法である STAMP/STPA について述べる．第 6 節では， 提案手法の評価について述べる．最後に第 7 節では， 本研究のまとめと今後の課題について述べる．

2. ESPR の概要

2.1 ESPR について

ESPR（Embedded System development Process Reference）は IPA/SEC[1] が組込みソフトウェア開発を円滑に進めるために標準的な作業やベストプラクティスをまとめたものである文献 [2]. ESPR は, システムの開発プロセスに関する国際規格を参考にし， 具体的な作業のレベルで開発プロセスを整理した開発ガイドであり，個々の作業の入力と成果物を明示して作業内容についても具体的で理解容易な表現を用いることで高品質の組込みソフトウェアを効率的に開発することを目的としている．

2.2 ESPR の作業内容

本稿で対象とする形式仕様記述に関する ESPR のプロセスは, システム・エンジニアリング，ソフトウェア・エンジニアリング，セーフティ・エンジニアリングプロセスである．

2.2.1 システム・エンジニアリング・プロセス

システム・エンジニアリング・プロセス（SYP）は， 組み込みシステム開発に関する作業の中で，ソフトウェアとハードウェアを含む製品としての要求定義からシステムテストまでの作業を定義したプロセスである．この工程のより具体的な作業をまとめた作業群（アクティビティ）は以下の通りである．

- SYP1 システム要求定義
- SYP2 システム・アーキテクチャ設計
- SYP3 システム結合テスト
- SYP4 システムテスト

2.2.2 ソフトウェア・エンジニアリング・プロセス

ソフトウェア・エンジニアリング・プロセス（SWP）は，組込みソフトウェア開発に関する作業の中でも，直接ものづくりに関係するソフトウェアとして要求定義からソフトウェア総合テストまでの作業を定義したプロセスである．この工程のアクティビティは以下の通りである．

- SWP1 ソフトウェア要求定義
- SWP2 ソフトウェア・アーキテクチャ設計
- SWP3 ソフトウェア詳細設計
- SWP4 実装および単体テスト
- SWP5 ソフトウェア結合テスト
- SWP6 ソフトウェア総合テスト

2.2.3 セーフティ・エンジニアリング・プロセス

セーフティ・エンジニアリング・プロセス（SYP）は，システムとして想定されるあらゆる利用形態を考慮し，その状況下でシステムに求められる安全性を定義し， それが確実に実現されていることを確認するための作業を定義したプロセスである．この工程のアクティビティは以下の通りである．

- SAP1 安全性要求定義
- SAP2 安全性テスト

3. Co-simulation の概要

3.1 Crescendo Tool

Crescendo Tool[3] は， Co-modelling と Co-simulation をサポートするツールであり, 連続時間モデル/Continuous-Time model（CT-model）と離散イベントモデル/Discrete-Event model（DE-model）を結びつけ，障害のモデリングやフォールトトレラントメカニズムを含む複数の側面からのモデリングを Co-simulation を通してサポートするツールである．

Crescendo Tool ではこの Co-model を動作させ Co-simulation を行う．Contract とは，DE-model と CT-model の間の相互作用に関する契約で以下の 3 つの要素からなる．

・設計パラメータ
　CT-model と DE-model が共有している値． シミュレーション中は一定である．

・変数
　シミュレーション中にそれぞれのモデルで書き換えられる．

・イベント
　モデルからもう一方のモデルへ引き起こされるアクション．

Crescendo Tool では， 以上の 3 つの要素をモデル間でやり取りをすることで Co-simulation を行う．

3.1.1 連続時間モデル/CT-model

CT-model（Continuous-Time model）は， 時間の経過に応じてシステムの状態が連続的に変化するモデルである．Crescendo Tool では， システムの制御対象（プラント）をボンドグラフによる CT-model で表現し，20-sim[4] というツールを用いてモデリング， シミュレーションを行う．

3.1.2 離散イベントモデル/DE-model

DE-model はシステムの状態が遷移する時間ポイントだけが表現されたモデルである．この DE-model を Crescendo Tool では, 形式仕様記述言語 [5] である VDM-RT を用いてモデリングを行い，Overture Tool[6] を用いてシミュレーションを行う．

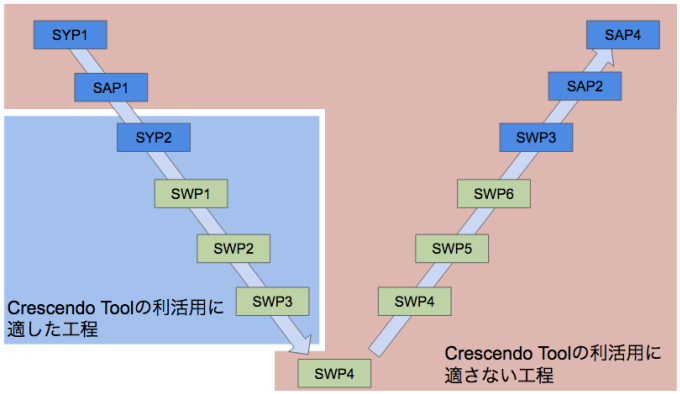

図 1　ESPR と Crescendo Tool の対応付けの結果
Fig. 1　Result of associating Crescendo Tool with development process based on ESPR

4. ESPR を用いた開発への Crescendo Tool の適用結果の分析

本節では， Crescendo Tool を ESPR をベースにした開発プロセスで活用することを想定し，各工程の作業内容や成果物と Crescendo Tool の支援内容の対応関係を分析する．本研究では， Contract を決定した後に， CT-model, DE-model をそれぞれ並列に作成し，2 つを統合することで Co-model を構築する方法である Contract-first の手順に従って Co-model を構築することを想定している．

4.1 Crescendo Tool と ESPR の対応付け

まず, ESPR の作業手順に従って Crescendo Tool を用いたモデルベース開発を行う際の，各工程における Co-model の抽象度と Crescendo Tool の役割を確認し，ツールを開発プロセスで使用する際の問題点を指摘し， その問題点における改善手法を提案する．

4.2 Crescendo Tool が適用できる開発工程

この節では， ESPR を用いた開発において Crescendo Tool を効果的に適用できる工程とその工程でツールで確認できることについて述べる．また, 開発工程における適用範囲を図 1 に示す.

システム・アーキテクチャ設計（SYP2）
この工程ではシステムを構成する機能をハードウェア， ソフトウェアの視点から整理し，機能実現のためのハード/ソフトの役割分担を決める際に Contract を確立することができる．ただし具体的な Co-model を構築することはまだできない．

ソフトウェア要求定義（SWP1）
この工程では， 前工程である SYP からの入力を受けて， ソフトウェアに必要となる機能/非機能要求を VDM-RT で記述し DE-model を構築することで， システムをソフトウェアで実現する際の機能/非機能面の要求と制約条件を明確にすることができる．したがって， この工程では機能ブロック単位でソフトウェアに要求される機能・非機能を記述した状態の DE-model と CT-model を統合した抽象度の高い Co-model を得ることができる．

ソフトウェア・アーキテクチャ設計（SWP2）
この工程では， 機能ユニットの振る舞い・構造まで詳細化した状態の DE-model と CT-model を統合した Co-model を得ることができる．また， 得られた Co-model を用いて， 以下の内容を確認することができる．

- 設計した機能ユニットの振る舞い・構造の妥当性
- ソフトウェアの非機能要求が十分に反映されているかの評価
- ソフトウェア要求との対応（トレーサビリティ）が取れているかの評価

ソフトウェア詳細設計（SWP3）
この工程では， プログラムユニット単位（実装可能なレベル）の振る舞い・構造まで詳細化した DE-model と CT-model を得ることができる．また， 得られた Co-model を用いて， 以下の項目を確認することができる．

- 詳細化されたユニットの処理の振る舞い・構造の妥当性
- 詳細化された処理がソフトウェアとハードウェア両方の側面から仕様の整合性

4.3 Crescendo Tool が適用できない開発工程

Crescendo Tool の利活用に適さない工程は以下の 2 つに分類できる．

SYP1 以前の工程
ESPR において， システムに必要な機能を実現するためのハードウェアとソフトウェアの役割分担を検討する SYP1 以前の工程では， ハードとソフトの間の相互作用に関する契約である Contract を確立することはできない．SYP1 以前の工程では Co-moddelling を行うことができず， Crescendo Tool を適用することはできない．

実装以降の工程
Crescendo Tool は，DE-model と CT-model を協調的に動作させるツールである．実装以降の工程の検証では， 実装したソフトウェアを用いるため， DE-model を用いた検証は必要ない．

5. Co-simulation シナリオを利用したプロセスの改良提案

前節の分析を受けて， 本研究では， SYP1 以前の工程

において，Co-model を作り込む過程において SYP1 以前の工程で行った要求分析が Co-model に反映されているかを Crescendo Tool を用いて確認することができるシナリオを作成する手段として, ハザードシナリオの利用法を提案する．

5.1 提案する手法

本研究では， 安全性要求定義（SAP1）の工程においてシステムの要求仕様から，ハザード分析手法である STAMP/STPA を用いて，障害モデルやハザードシナリオを作成し，下流工程で活用することを提案する．抽出した障害モデルやハザードシナリオを活用し，システム・アーキテクチャ設計（SYP2）以降の工程において Co-model を開発し，Crescendo Tool を用いることで検証を行う．ハザード分析を行って得られたハザードシナリオを効果的に用いることで，SYP2 以降の工程で作られる Co-model に対し Co-simulation を行い，SAP1 で行った安全性要求定義の内容が， Co-model に反映されているかどうかを確認することができる．

5.2 STAMP/STPA の概要

STAMP（Systems-Theoretic Accident Model and Process/システム理論に基づく事故モデル）は，システム論を利用した事故モデルであり, システムを構成するコンポーネント間の相互作用に着目してシステム全体の流れを表す事故モデルとなっている．このため， STAMP では事故は単純なコンポーネントの故障のみを原因とせず， コンポーネントの振る舞いやコンポーネント間の相互干渉が， システムの安全制約を違反した場合に起こると考える．

STPA（System Theoretic Process Analysis）は STAMP を利用したハザード分析手法であり， 事故が起きる前に事故を引き起こす潜在的な要因を見つけ出すことを目的としている．文献 [7] に従い，ここでは大きく分けて以下の 4 つの順序に従って STPA の分析を行う．

5.2.1 準備 1 : アクシデント， ハザード， 安全制約の識別

準備 1 ではアクシデント， ハザード， 安全制約の 3 つを作成する必要がある．これらはシステムが回避すべき事象を事前に設定することで目的に沿ったハザード分析を行うためのものである．それぞれの言葉の定義は以下に記す．

アクシデント　損失を伴う， システムの事故
ハザード　アクシデントにつながるシステムの状態
安全制約　システムが安全に保たれるために必要なルール

5.2.2 準備 2 : コントロールストラクチャの作成

準備 2 ではコントロールストラクチャを作成する．コントロールストラクチャは， システムを制御する各機能の相関関係を示した図である．コンポーネント間でやり取りされる制御の指示やフィードバックなどを矢印で結んで表す． 各コンポーネントはコントローラー（Controller）と呼ばれ， その機能や役割に応じて階層構造になっている． 上位のコントローラーからはコントロールアクション（Control Action）と呼ばれる指示が出され， センサーやアクチュエータなど下位のコントローラーからはフィードバック（Feedback）として情報が送られる． さらに各コントローラーには， そのコントローラーがどのような処理や指示を行うかというプロセスモデル（Process model）が含まれており，特に人間が行うプロセスモデルはメンタルモデル（Mental model）と呼ばれている．これらのコントローラー間のやり取りを表したものをコントロールループと呼ぶ．

5.2.3 STPA Step1 : 安全でないコントロールアクションの識別

STPA Step1 では安全でないコントロールアクション（Unsafe Control Action : UCA）の識別を行う．UCA の識別は， ハザードにつながる恐れのあるコントロールアクションの不具合を明確にすることを目的としており，大きく分けて， 以下の 4 つの種類に分類される．

- 「Not Providing」与えられないとハザード
  - 安全のためのコントロールアクションが与えられない．
- 「Providing」与えられるとハザード
  - ハザードにつながる恐れのある， 安全ではないコントロールアクションが与えられる．
- 「Wrong Timing/Order」早すぎ、遅すぎ、誤順序でハザード
  - コントロールアクションのタイミングが遅すぎる， 早すぎる， または定められた順序に設置していない．
- 「Stopping Topp Soon / Applying Too Long」早すぎる停止、長すぎる適用でハザード
  - コントロールアクションがすぐに止まる， もしくは適用が長すぎる．

上記 4 つの種類以外に 5 番目のシナリオとして，「要求されたコントロールアクションが提供されているが， それに従っていない」という UCA が考えれる．この原因については， コントロールループ内での不具合や遅れなど不適切な動作が含まれる． 5 番目のシナリオに関しては，STPA Step2 で分析していくことになる．

5.2.4 STPA Step2 Causal factor（誘発要因）の特定

STPA 最後の段階として，STPA Step1 で識別した UCA の原因となる Causal factor と， 予想される事故シナリオの特定を行う．Causal factor の特定には， UCA の引き金になる 11 個のガイドワードを使って， コントロールストラクチャ内の各コントロールループを分析していく．このプロセスでは， UCA に対してガイドワードの適用を行い，どのようにハザードが発生するかという部分にまで詳細に

分析を進める必要がある．

Step1 での 4 つの標準的な分類（ガイドワード）をコントロールストラクチャに適用し，コントロールループの基本コンポーネントを調べて，誤った操作を分類し，その標準の不適切な制御の原因となりうるかを決定する．11 個のハザード要因（ガイドワード）は以下の通りである．

(1) コントロール入力や外部情報の誤りや喪失．
(2) 不適切なコントロールアルゴリズム．
（作成時の欠陥，プロセスの変更，誤った修正や適用）
(3) 不整合，不完全，または不正確なプロセスモデル．不適切な操作．
(4) コンポーネントの不具合．経年による変化．
(5) 不適切なフィードバック，あるいはフィードバックの喪失．フィードバックの遅れ．
(6) 不正確な情報の供給，または情報の欠如．
測定の不正確性．フィードバックの遅れ．
(7) 操作の遅れ．
(8) 不適切または無効なコントロールアクション，コントロールアクションの喪失．
(9) コントロールアクションの衝突．プロセス入力の喪失または誤り．
(10)未確認または範囲外の障害
(11)システムにハザードを引き起こすプロセス入力．

6. 提案した手法の事例適用

本節では，前節で提案した手法を Crescendo Tool の例題として含まれている Line Following Robot[8] に適用し，ハザードシナリオを作成した結果について述べる．

6.1 モデルの説明

この節では，提案手法を適用した Crescendo Tool の例題モデルの Line Following Robot モデルの概要について述べる．Line Following Robot は，車体に取り付けられたセンサーで地面に描かれたラインに沿って走行するロボットである．ロボットは，いくつかのセンサーを使うことで地面の明るさを読み取り，読み取った情報を元に走行する．また，車輪は 2 つ備わっており，それぞれの車輪のモーターに別々に出力を伝えることで，きめ細かい方向転換を行うことができる．本研究ではこのモデルのケース記述（Case Description）を要求仕様書とし，ハザードシナリオを抽出した．

6.2 STAMP/STPA を用いたハザードシナリオ作成

この節では，先述した STAMP/STPA のハザード分析手順に沿って Line Following Robot の Case Description からハザードシナリオを作成した際の作業内容を以下に述べる．

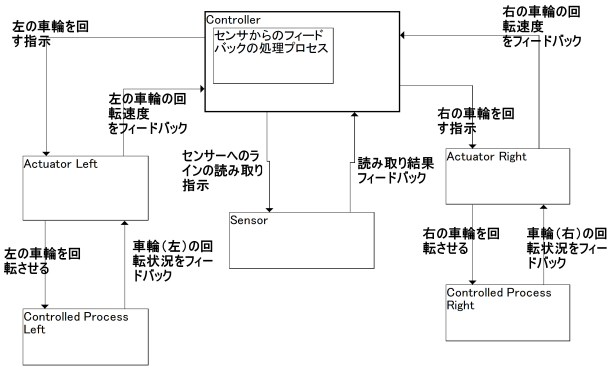

図 2 Line Following Robot の コントロールストラクチャ
Fig. 2 Control Structure of Line Following Robot

6.2.1 アクシデントとハザード，安全制約の識別

この段階では Line Following Robot システムのアクシデントとハザード，安全制約の識別を行った．実際に作成したアクシデント，ハザード，安全制約を表 **??**に示す．

6.2.2 コントロールストラクチャの構築

この段階では Line Following Robot システムのコントロールストラクチャを作成した．実際に構築したコントロールストラクチャを図 **2** に示す．また，Line Following Robot のコントロールストラクチャを構成するコンポーネントと，それらの役割について述べる．

Controller
Controller は，Line Following Robot の制御を行うコンポーネントである．Controller はセンサーへラインの読み取り指示を出し，地面の状況を受け取る．その情報をコントローラーで処理し，センサーの下がライン上かどうか判定する．判定に従って次の車体の動作を処理し，Actuator に出すべき出力を伝える．

Sensor
Sensor は Controller からの指示を受け地面の状態（明るさ）を読み取り，その情報をフィードバックするコンポーネントである．

Actuator（Left/Right）
Actuator はコントローラーから車輪を回す指示を受けると，車輪を回転させ，現在の車輪の回転状況（速度）をコントローラーにフィードバックするコンポーネントである．

Controlled Process（Left/Right）
Controlled Process は，Line Following Robot における車輪に当たるコンポーネントで，Actuator によって回転させられる．

6.2.3 STPA Part1. UCA の識別

STPA Step1 では UCA の識別を行った．実際に適用した結果を表 **2** に示す．

システム	アクシデント	ハザード	安全制約
LineFollowingRobot	目的地までたどり着く事ができない	ラインを見失いコースからはずれてしまう	・センサーが正常に作動している ・駆動系が正常に作動している

表 1　Line Following Robot システムのアクシデント， ハザード， 安全制約
Table 1　Accidents,Hazards,Safety constraints of Line Following Robot

表 2　STPA Step1 分析結果
Table 2　result of STPA Step1

Unsafe Control Actions

Control Action	Not providing causes hazard	Providing causes hazard	Wrong timing or order causes hazard	Stopped too soon or applied too long
センサーへのラインの読み取り指示	センサーでラインを読み取る指示が出されず、ラインを読み取ることができず、ラインを見失う [H-1]	センサーで読み取った情報を正しく処理できずラインを見失う [H-1]	センサーで読み取るタイミングが遅く、ラインの変化に対応できず、ラインを見失う [H-1]	
右の車輪を回転させる		車輪の回転速度が速すぎてラインの変化を見落として、コースから外れてしまう。 [H-1]	車輪を回転させるタイミングが遅く、ラインの変化に対応できず、コースから外れてしまう [H-1]	
左の車輪を回転させる		車輪の回転速度が速すぎてラインの変化を見落として、コースから外れてしまう。 [H-1]	車輪を回転させるタイミングが遅く、ラインの変化に対応できず、コースから外れてしまう [H-1]	
右の車輪を回す指示		車輪の回転速度が速すぎてラインの変化を見落として、コースから外れてしまう。 [H-1]	車輪を回転させるタイミングが遅く、ラインの変化に対応できず、コースから外れてしまう [H-1]	
左の車輪を回す指示		車輪の回転速度が速すぎてラインの変化を見落として、コースから外れてしまう [H-1]	車輪を回転させるタイミングが遅く、ラインの変化に対応できず、コースから外れてしまう [H-1]	

表 3　STPA Step2 分析結果
Table 3　result of STPA Step2

STPA パート1分析内容	車輪の回転速度が速すぎてラインの変化を見落として，コースから外れてしまう．
STPA パート1分類	Providing causes hazard
STPA パート2該当項目	不十分な制御アルゴリズム（作成上の不具合，プロセス変更，不正確な修正）
	コントロールアクションの衝突，プロセス入力の喪失
考えられるハザード要因 ・速度の設定が不適切である（不十分な制御アルゴリズム） ・アクチュエーターからの出力がコントローラーから指示されたものと違う（プロセス入力の喪失）	

STPA パート1分析内容	車輪の回転速度が速すぎてラインの変化を見落として，コースから外れてしまう．
STPA パート1分類	Wrong timing or order causes hazard
STPA パート2該当項目	不適切または無効なコントロールアクション，コントロールアクションの喪失
	動作の遅れ
考えられるハザード要因 ・コントローラから指示された出力をアクチュエーターから車輪へ伝えるのが遅い（動作の遅れ） ・コントローラーがアクチュエーターに指示を伝えるのが遅い（不適切なコントロールアクション）	

STPA パート1分析内容	センサーでラインを読み取る指示が出されず，ラインを読み取る事ができず，ラインを見失う
STPA パート1分類	Not providing causes hazard
STPA パート2該当項目	不十分な制御アルゴリズム（作成上の不具合，プロセス変更，不正確な修正）
考えられるハザード要因 ・コントローラーの処理設計が間違っている．（不十分な制御アルゴリズム）	

STPA パート1分析内容	センサーで読み取った情報を正しく処理できず，ラインを見失う．
STPA パート1分類	Providing causes hazard
STPA パート2該当項目	不十分な制御アルゴリズム（作成上の不具合，プロセス変更，不正確な修正）
	プロセスモデルの不一致
	フィードバックの不十分，欠落，遅延
考えられるハザード要因 ・センサーからのフィードバックが間違っている．（不十分な制御アルゴリズム / プロセスモデルの不一致） ・センサーが故障している．（フィードバックの不十分，欠落，遅延）	

STPA パート1分析内容	センサーで読み取るタイミングが遅く，ラインの変化に対応できず，ラインを見失う．
STPA パート1分類	Wrong timing or order causes hazard
STPA パート2該当項目	不十分な制御アルゴリズム
考えられるハザード要因 ・コントローラーの処理の仕方が適切ではない．（不十分な制御アルゴリズム）	

6.2.4　STPA Part2. Causal factor（誘発要因）の特定

STPA Step2 として， STPA Step1 で識別した UCA の原因となる Causal factor と， 予想される事故シナリオの特定を行った．実際に適用した結果を**表 6.2.4** に示す．

6.2.5　ハザード分析によって得られたハザードシナリオ

事故モデルに対し STPA を適用した結果， 以下に示すようなハザードシナリオが得られた．アクチュエーターは左右に 1 つずつで合計 2 つずつあり， 同じ構造なのでそれぞれのアクチュエーターが原因のハザードシナリオは同じものとなるため，重複を省略したものにしている．

ハザードシナリオ（HS）

HS1. 速度の設定が不適切で， 車輪の回転速度が速すぎるためラインの変化を見落として， コースから外れてしまう．

HS2. アクチュエーターが故障しており， アクチュエーターからの出力がコントローラーから指示されたものと違い，結果として車輪の回転速度が速くなりラインの変化を見落としてコースからはずれてしまう．

HS3. コントローラーからのコントロールアクションが欠落し， 指示がアクチュエーターに届かなかったため， ラインの変化に対応できず， コースから外れてしまう．

HS4. コントローラーから指示された出力をアクチュエーターから車輪へ伝えるのが遅く， ラインの変化に対応できず， コースから外れてしまう．

HS5. コントローラーの処理が不適切で， センサーでラインを読み取るという指示が出されず， ラインを読み取ることができず， ラインを見失う．

HS6. コントローラーのセンサーからのフィードバックの処理が間違っており， センサーで読み取った情報を正しく処理できずラインを見失う．

HS7. センサーが故障していて， 正しくラインを読み取ることができず， ラインを見失う．

HS8. コントローラーの処理が不適切で， センサーで読み取るタイミングが遅く， ラインの変化に対応できず，ラインを見失う．

6.3　提案手法の考察

6.3.1　ハザードシナリオに対する考察

提案手法によって作成されたハザードシナリオに基づいた Co-simulation により以下の内容を確認できると考える．

HS1

SWP2 において，速度を Co-model における設計パラメーターとし，さまざまな値のパラメーターでシミュレーションを行うことにより，ロボットがラインに沿って走行するために必要な制約条件を確認する．実装後の SWP5 において，得られた制約条件を用いて実装した機能が正しい振る舞いをするかを確認する．

HS2

アクチュエーターが故障しているという障害を 20-sim でモデリングする．モデリングした障害をシミュレーション中に発生させるケースを Crescendo Tool のシナリオ記述言語で記述しアクチュエーターが故障した状況を再現する．これにより，障害が発生した際のロボットの振る舞い方を確認することができる．この障害モデルを用いることにより SWP2 において，想定される障害に対する非機能要求を設計する際に，シミュレーションによって，要求された機能を設計できているか確認する．また，SWP5 において正しく実装されているかを確認する．

HS3

SWP2 において，設計段階でコントローラーからのコントロールアクションがアクチュエーターに確実に届いているかをシミュレーションを行い確認する．また，SWP5 において機能ユニットが正しく実装されているかを確認する．

HS4

SWP2 において，CT-model と DE-model の同期時間を変えてシミュレーションを行うことでコントローラーからアクチュエーターに指示が伝わるタイミングが適切であるかを確認する．また，SWP5 において設計どおりのタイミングで動作するよう実装されているかを確認する．

HS5

SWP3 においてシミュレーションを行い，設計段階でセンサーでラインを読み取るという指示を出す機能がコントローラーの設計に含まれていることを確認する．また，実装後の SWP4 でコントローラーが正しく実装されているかを確認する．

HS6

SWP3 においてシミュレーションを行うことで，センサーからのフィードバックに対するコントローラーの振る舞いが正しく設計されているかを確認する．また，SWP4 においてコントローラーの振る舞いが正しく実装されているかどうかを確認する．

HS7

センサーが故障しているという障害を 20-sim でモデリングする．モデリングした障害をシミュレーション中に発生させるケースを Crescendo Tool のシナリオ記述言語で記述しセンサーが故障した状況を再現する．これにより障害が発生した際のロボットの振る舞い方を確認することができる．SWP2 において，想定される障害に対する非機能要求を実現するため，設計時にシミュレーションを行うことで，要求された機能を確実に設計できているか確認する．また，SWP5 において正しく実装されているかを確認する．

HS8

SWP2 において，CT-model と DE-model の同期時間を変えてシミュレーションを行うことでコントローラーからセンサーに指示が伝わるタイミングが適切であるか確認する．また，SWP5 において設計通りのタイミングで動作するよう実装されているかを確認する．

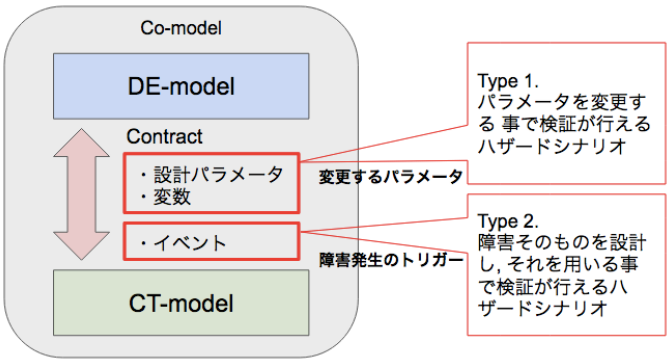

図 3　ハザードシナリオと Co-model の関係性
Fig. 3　Relationship of the hazard scenarios and Co-model

提案手法によって作成したハザードシナリオは，Crescendo Tool で検証可能であり，以下の 3 つのタイプに分類することができることが分かった．

Type1

パラメーターを変更することで検証が行えるハザードシナリオ

- 該当するハザードシナリオ：HS1，HS4，HS8

Type2

検証を行うために障害そのものを設計する必要があるシナリオ

- 該当するハザードシナリオ：HS2，HS7

Type3

Co-model を動かすことで検証が行えるシナリオ（DE-model の設計の検証）

- 該当するハザードシナリオ：HS3，HS5，HS6

本研究で作成したハザードシナリオは，Line Following Robot の Co-model における Contract と対応している．先述した Type1，Type2 のハザードシナリオと Contract

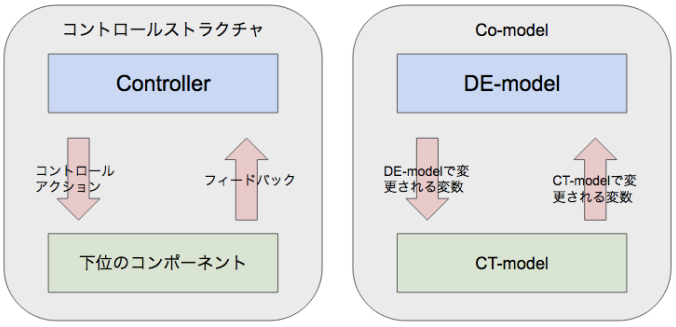

図 4　コントロールストラクチャと Contract の関係性
Fig. 4　Relationship of the Control Structure and Contract

の間には図 **3** のような対応関係がある．

6.3.2　STAMP と Co-model の関係について

提案手法における STAMP/STPA の作業成果物である Line Following Robot のコントロールストラクチャにおいて，コンポーネント間の関係は， Co-model における DE-model と CT-model 間の Contract に対応する．図 **4** のように， コントロールストラクチャにおける Controller は， Co-model における DE-model にあたり， Controller に制御されるコンポーネントは CT-model に相当する．

コントロールアクションは DE-model から CT-model に渡される変数と対応があり，下位のコンポーネントからのフィードバックは， CT-model から DE-model へ渡される変数と対応がある．コントロールストラクチャのコンポーネント間でやり取りされる情報は， Contract の構成要素の一つである変数と対応付けられる．提案手法の作業成果物から， 下流での Co-modelling における Contract の候補を作成できると考えられる．

7.　まとめと今後の課題

本研究では， 組込みシステムの協調的開発において，Co-simulation による設計分析を効果的に行うために，組込みシステムの開発プロセスガイドである ESPR に， Co-simulation をサポートする Crescendo Tool の活用を対応付けた．各工程で作成するモデルの抽象度とそのモデルに対する分析内容を整理した．その結果， 想定されるシステム障害について SAP1 で分析しておく必要があることが分かった．そのため， システム論に基づいたハザード分析手法である STMAP/STPA を適用する手法を提案した．提案手法を Crescendo Tool の例題の一つである Line Following Robot の要求仕様書に適用した結果, 後の工程において作成されるモデルにおいて検証すべき内容を, Co-simulation による設計レベルでの検証が可能なシナリオとして作成できた．また, 提案手法は Co-modelling における Contract の確立にも有効であることが分かった．SAP1 で扱うことができる Co-model の抽象度が高いため, 提案手法によって作成されたハザードシナリオも抽象度の高いものとなった．形式仕様によるモデリングの効果とそのオーバーヘッドの間の定量的なトレードオフの評価については今後の課題である．

また, 本研究で作成したハザードシナリオは自然言語で記述されているため, 曖昧さを含んでおり, 読み手によってハザードシナリオの解釈が異なる可能性がある．ハザードシナリオを形式手法である VDM を用いて表現することで, そのような曖昧さを排除したハザードシナリオが得られると考えられる．VDM を利用した曖昧さを排除したハザードシナリオの記述と, そのハザードシナリオの利活用についても, 今後の課題である．

謝辞　本研究の一部は JSPS 科研費 24220001, 基盤研究 (S)「アーキテクチャ指向形式手法に基づく高品質ソフトウェア開発法の提案と実用化」の成果による．

参考文献

[1] : IPA 情報処理推進機構，https://www.ipa.go.jp/.
[2] 独立行政法人情報処理推進機構 ソフトウェア・エンジニアリング・センター編著：【改訂版】組込みソフトウェア向け開発プロセスガイド，翔泳社 (2007).
[3] : The Crescendo Tool, http://crescendotool.org/.
[4] : 20-sim, http://www.20sim.com/.
[5] : VDM Information web Site, http://fmvdm.org/vdmtools/index-ja.html.
[6] : Overture, http://www.overturetool.org/.
[7] 独立行政法人日本貿易振興機構：STAMP 手法に関する調査報告書，技術報告，独立行政法人情報処理推進機構ソフトウェア高信頼化センター (2015).
[8] Ingram, C., Pierce, K., Gamble, C., Wolff, S., Christensen, M. P. and Larsen, P. G.: Crescendo Examples Compendium, Technical report, Aarhus University, Department of Engineering Finlandsgade 22, DK-8200 Aarhus N, Denmark (2014/1).