修士論文

統計的モデル検査法を用いたSTAMP/STPAにおけるシナリオのリスク分析

辻 光顕
奈良先端科学技術大学院大学
先端科学技術研究科
情報理工学プログラム

主指導教員: 飯田 元 教授
超高信頼ソフトウェアシステム検証学 研究室（情報科学領域）

令和2年3月13日提出

本論文は奈良先端科学技術大学院大学先端科学技術研究科に
修士 (工学) 授与の要件として提出した修士論文である.

辻 光顕

審査委員：

飯田 元 教授	（主指導教員，情報科学領域）
松本 健一 教授	（副指導教員，情報科学領域）
片平 真史 教授	（副指導教員，情報科学領域）
石濱 直樹 准教授	（副指導教員，情報科学領域）
高井 利憲 准教授	（副指導教員，情報科学領域）

統計的モデル検査法を用いたSTAMP/STPAにおけるシナリオのリスク分析*

辻 光顕

内容梗概

近年，ソフトウェアを中心としたシステムに対する新たなハザード分析手法としてSTAMP/STPAが注目されている．STAMP/STPAでは，システムに対するコントロールストラクチャを作成し，そこからハザードに至るシナリオおよび安全対策を導出する．STAMP/STPAは，FTAやETAといった従来型のハザード分析手法では識別ができないような，システムの機能間における認識の齟齬を要因としたハザードも識別することができるという特長がある．一方で，システムの開発期間やコストが限られた中で効率的にシステムの品質を高めるためには，ハザードの発生確率と深刻度の積であるリスクに基づくシナリオの優先付けによる対策の検討が必要であると考えられる．そこで本研究では，統計的モデル検査法を利用して，STAMP/STPAで導出されたシナリオに対するリスクを分析する手法について述べる．特に，STAMP/STPAにおけるコントロールストラクチャを形式化し，シナリオを形式モデル上で表現する方法について提案する．本研究では，鉄道における踏切制御システムを題材に手法を適用し，提案手法の評価を行った．

キーワード

リスク分析, 形式手法, 統計的モデル検査法, ハザード分析, STAMP/STPA

*奈良先端科学技術大学院大学 先端科学技術研究科 修士論文, 令和2年3月13日.

Risk Analysis of Scenarios based on STAMP/STPA by Using Statistical Model Checking*

Mitsuaki Tsuji

Abstract

Recently, a new hazard analysis technique called STAMP/STPA for software-intensive systems has been widely accepted. In STAMP/STPA, a control structure diagram of a system is defined, and then, scenarios that can lead to a loss and safety measures will be derived. An important advantage of STAMP/STPA is that it allows to identify not only hazards related to component failure, but also software-related and non-failure hazards, which can't be analyzed by traditional hazard analysis techniques such as Fault Tree Analysis (FTA) and Event Tree Analysis (ETA). However, in order to improve system quality efficiently, scenarios need to be prioritized and safety measures will be considered in terms of risk (combination of probability and severity of hazards) when development resources, such as development cost or development period, are limited. Therefore, in this study, the author proposes a method to evaluate risk of scenarios based on STAMP/STPA by using statistical model checking. Especially, proposes a method for systematically transforming a control structure diagram to a formal model in order to express scenarios on a formal model. The proposed method was evaluated by using an example of the railway gate control system. As a result, it

*Master's Thesis, Graduate School of Science and Technology, Nara Institute of Science and Technology, March 13, 2020.

could confirm that scenarios are expressed on a formal model and could prioritize them by analyzing risk.

Keywords:

Risk Analysis, Formal Method, Statistical Model Checking, Hazard Analysis, STAMP/STPA

目 次

1. はじめに **1**

2. 本研究が取り組む課題と目的 **4**

3. ハザード分析手法 STAMP/STPA **7**

3.1 事故モデル 7

3.1.1 従来の事故モデル 7

3.1.2 システム理論に基づく事故モデル STAMP 8

3.2 STAMP/STPA におけるハザード分析の手順 10

3.3 分析目的の定義 10

3.3.1 損失の識別 11

3.3.2 システムレベルにおけるハザードの識別 11

3.3.3 システムレベルにおける安全制約の識別 11

3.3.4 システムレベルにおけるハザードの精密化 12

3.4 コントロールストラクチャのモデル化 12

3.5 非安全なコントロールアクションの識別 13

3.6 ハザードに至るシナリオの識別 14

3.6.1 非安全なコントロールアクションにつながるシナリオ . . . 16

3.6.2 コントロールアクションが不適切な実行をされる，または実行されずにハザードに至るシナリオ 19

4. 統計的モデル検査法 **23**

4.1 統計的モデル検査法の概要 23

4.2 Uppaal SMC における形式的なモデルの定義 24

4.3 Uppaal SMC におけるクエリの記述 29

5. 提案手法 **31**

5.1 手順 1：STAMP/STPA によるハザード分析 31

5.2 手順 2：振る舞いの識別と合意 32

5.3 手順3：シナリオ生成用モデルの作成 33
5.3.1 シナリオ生成用モデルの形式的定義 33
5.3.2 コントロールアクションとフィードバックの形式化 38
5.3.3 コントローラの形式化 40
5.3.4 被コントロールプロセスの形式化 41
5.3.5 アクチュエータとセンサの形式化 41
5.3.6 外部環境の形式化 42
5.3.7 ハザードの形式モデルへの反映 42
5.4 手順4：シナリオ観測用モデルの記述 49
5.5 手順5：シナリオに対する形式検証 51
5.6 手順6：シナリオに対するリスク分析 53

6. 事例による提案手法の評価 **54**
6.1 踏切制御システムの概要 54
6.2 提案手法による分析 56
6.2.1 STAMP/STPA による分析例 (手順1) 56
6.2.2 振る舞いの識別と合意の例 (手順2) 60
6.2.3 シナリオ生成用モデルの記述例 (手順3) 62
6.2.4 シナリオ観測用モデルの記述例 (手順4) 65
6.2.5 シナリオに対する検証例 (手順5) 67
6.2.6 シナリオに対するリスク分析例 (手順6) 68
6.3 事例に対する評価 69

7. 関連研究 **71**

8. 結論と今後の課題 **73**

謝辞 **75**

参考文献 **78**

付録 **83**

A. SysML で記述された振る舞いのモデル **84**

B. Uppaal SMC で記述された形式モデル **91**

図 目 次

1 リスクマネジメントプロセス 1
2 フォルトツリーの例 [13] 4
3 イベントツリーの例 [13] 5
4 ドミノモデル 8
5 システム理論におけるシステムの概念 9
6 STAMP における相互作用の概念図 9
7 STAMP/STPA の手順 10
8 コントロールストラクチャ 13
9 コントロールループ 15
10 非安全なコントローラの動作と不適切なフィードバックの範囲 . . 17
11 コントロール経路とコントロール対象プロセスにおける要因 . . . 20
12 確率時間オートマトンとその確率分布の例 (1) 26
13 確率時間オートマトンとその確率分布の例 (2) 27
14 確率時間オートマトンの並行合成の例 28
15 アージェントブロードキャストチャネルの例 29
16 本研究における提案手法の手順 31
17 手順 2 の流れ 32
18 コントロールストラクチャインターフェースの例 35
19 シナリオ生成用モデルの例 37
20 図 19 のシナリオ生成用モデルにおける各オートマトンの例 37
21 シナリオ生成用モデルの概念図 39
22 故障に対する形式的な表現 43
23 不適切なコントロールアルゴリズムの表現例 44
24 シナリオ観測用モデルの例 51
25 テンプレートの集合の例 52
26 想定する踏切制御システムの構成 54
27 想定する踏切制御システムのコントロールストラクチャ 57
28 識別されたコントロールループ 59

29 システム全体の構成要素に関するブロック定義図 60
30 連動装置の信号制御に関する振る舞い記述 61
31 列車 (TrainA) のステートマシン図による振る舞い記述 62
32 連動装置におけるコントロールアクションの出力側オートマトンの例 63
33 フィードバックの入力側 (左) と出力側 (右) に対応するオートマトンの例 64
34 列車 (TrainA) のモデルの例 64
35 センサである踏切制御子 (左) と故障を制御する環境 (右) に対応するオートマトンの例 65
36 HS1 のシナリオ観測用モデル 66
37 踏切制御システムの構成 (1) 84
38 踏切制御システムの構成 (2) 85
39 踏切制御システムの構成 (3) 86
40 遮断器の振る舞い 87
41 在線情報の更新に対する振る舞い (1) 87
42 在線情報の更新に対する振る舞い (2) 88
43 独占権の情報の変更に対する振る舞い 88
44 踏切制御装置の振る舞い 89
45 踏切制御子の振る舞い 89
46 信号機の振る舞い 90
47 アクチュエータである信号機 (1R) に対応するオートマトン 91
48 アクチュエータである信号機 (2R) に対応するオートマトン 91
49 アクチュエータである信号機 (2L) に対応するオートマトン 92
50 アクチュエータである信号機 (3L) に対応するオートマトン 92
51 環境に対応するオートマトン (1) 92
52 環境に対応するオートマトン (2) 93
53 遮断器における C_{in} 型オートマトン 93
54 踏切制御装置における C_{out} 型オートマトン (1) 94

55 踏切制御装置における C_{out} 型オートマトン (2) 94
56 踏切制御装置における C_{out} 型オートマトン (3) 95
57 踏切制御装置における C_{out} 型オートマトン (4) 95
58 踏切制御装置における F_{in} 型オートマトン (1) 95
59 踏切制御装置における F_{in} 型オートマトン (2) 95
60 踏切制御装置における F_{in} 型オートマトン (3) 96
61 踏切制御装置における F_{in} 型オートマトン (4) 96
62 踏切制御装置における F_{in} 型オートマトン (5) 96
63 踏切制御装置における F_{in} 型オートマトン (6) 97
64 踏切制御装置における F_{in} 型オートマトン (7) 97
65 踏切制御装置における F_{in} 型オートマトン (8) 97
66 踏切制御装置における F_{in} 型オートマトン (9) 98
67 踏切制御装置における F_{in} 型オートマトン (10) 98
68 踏切制御装置における F_{in} 型オートマトン (11) 99
69 センサーである軌道回路に対応するオートマトン (1) 99
70 センサーである軌道回路に対応するオートマトン (2) 99
71 列車 (TrainA) における C_{in} 型オートマトン 100
72 列車 (TrainA) における F_{out} 型オートマトン 100
73 列車 (TrainB) における C_{in} 型オートマトン 100
74 列車 (TrainB) における F_{out} 型オートマトン (1) 101
75 列車 (TrainB) における F_{out} 型オートマトン (2) 101
76 列車 (TrainC) における C_{in} 型オートマトン 101
77 列車 (TrainC) における F_{out} 型オートマトン (1) 102
78 列車 (TrainC) における F_{out} 型オートマトン (2) 102
79 シナリオ (HS2) に対応するシナリオ観測用モデル 102
80 シナリオ (HS3) に対応するシナリオ観測用モデル (1) 102
81 シナリオ (HS3) に対応するシナリオ観測用モデル (2) 103
82 シナリオ (HS3) に対応するシナリオ観測用モデル (3) 103
83 シナリオ (HS4) に対応するシナリオ観測用モデル (1) 103

84 シナリオ (HS4) に対応するシナリオ観測用モデル (2) 103
85 シナリオ (HS4) に対応するシナリオ観測用モデル (3) 103
86 シナリオ (HS4) に対応するシナリオ観測用モデル (4) 104

表 目 次

1 UCA 表の例 14
2 非安全なコントローラの動作に対するハザード要因のモデル化の方針 43
3 不適切なフィードバックと情報の原因に対するハザード要因のモデル化の方針 46
4 コントロール経路に関連するシナリオに対するハザード要因のモデル化の方針 47
5 コントロール対象のプロセスに関連するシナリオに対するハザード要因のモデル化の方針 48
6 シナリオ例に対するイベント列 50
7 イベント列に対応するオートマトンの状態遷移 50
8 識別された損失，ハザード，安全制約 56
9 非安全なコントロールアクションの分析結果 58
10 HS1 のイベント列 66
11 検証結果 67
12 発生確率と深刻度によるリスクレベルの例 68
13 各シナリオに対するリスク分析結果 68

1. はじめに

近年，鉄道や自動車，人工衛星，家電製品などの多くのシステムが内部のコンピュータやネットワーク機能などを持って制御されている．そして，開発されるシステムの高機能化に伴い，ますます大規模化，複雑化している．こうしたシステムを開発するためには，設計段階のハザード分析によるハザードの識別と，ハザードに対する安全対策によるリスクの低減が重要である．ここで，ハザードとは，システムがその利害関係者に対して損失をもたらす可能性がある状態を指し，リスクとは，ハザードの深刻度と発生確率の積として定義される [4]．また，ここでの損失とは，人命の喪失や物的損害，評判の喪失といったシステムの利害関係者にとって受け入れられないようなものを指す．ハザードの例として，システムの部品レベルであればコンデンサの故障や電線の断線，システム全体のレベルでは列車の走行中に踏切が遮断されないといった状態や，航空機のニアミス等が挙げられる．こうしたハザードの特定およびリスクの管理は，システム開発の過程で正しく行われることが重要である．図 1 に，国際規格である ISO31000[4] で定義されたリスクマネジメントの手順を示す．

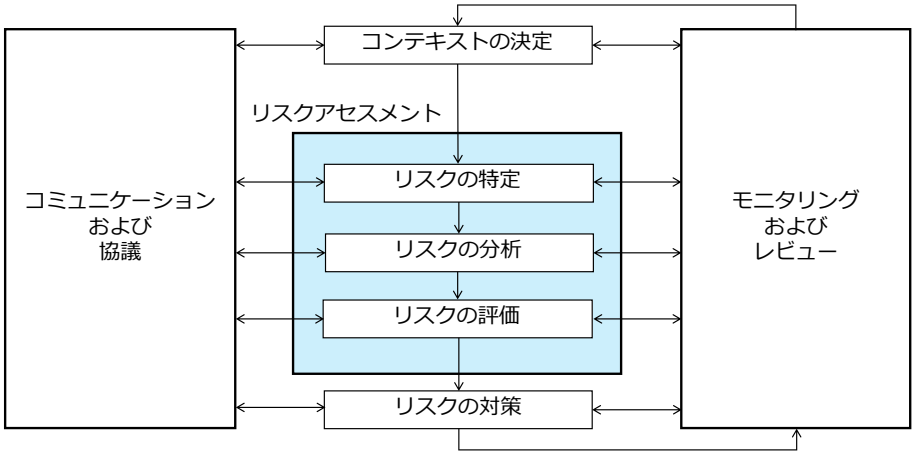

図 1　リスクマネジメントプロセス

ISO31000 の定義によれば，リスクマネジメントプロセスはコンテキストの決定から始まる．すなわち，解決すべき課題や目標，投入資源の確定，適用範囲等の，リスクマネジメントを進める上で前提となる条件の決定を行う．それに続くリス

クの特定，分析，評価は，合わせてリスクアセスメントと呼ばれる活動である．まず，リスクの特定では，利害関係者にとってリスクとなりうる事象を発見，認識し，記述することを行う．続いて，リスクの分析では，識別した事象の起こりやすさなどの特徴から，リスクの大きさに対する評価を行う．最後に，リスクの評価では，リスク分析の結果から対策を行うか，あるいはさらにリスク分析を行うか等の判断を行う．このようなリスクアセスメントを行って，実際にリスクの対策を行う．こうした一連の活動は，常にモニタリングやレビューが行われ，専門家等を交えたコミュニケーションや協議といった活動で情報共有がなされる．

本研究では，リスクアセスメントにおけるリスク分析を行うことを目的としている．こうしたリスク分析の活動は，通常システム開発の上流工程から実施される．例として，鉄道分野における国際規格である RAMS(IEC62278)[2] では，鉄道システムの開発においてリスク分析を行う段階が定義されている．このリスク分析は，システムの安全性および可用性，信頼性，保守性を達成するために開発初期段階から継続的に実施される．具体的には，まずハザードを識別を行い，各ハザードに対する発生頻度と深刻度からリスクの見積もりを行う．そして，リスクを低減するための方針の決定を行う．

ハザードやリスクを分析するための分析手法は複数存在し，代表的なものとしてフォルトツリー分析 (Fault Tree Analysis, FTA)[24] やイベントツリー分析 (Event Tree Analysis, ETA)[5] などがある．これらの手法は，ハザード発生の原因となる故障等から最終的に危険事象に至るまでの因果関係を記述することができる．また，各事象に対して故障率を仮定することにより，ハザードの発生確率およびリスクを求めることが可能となっている．一方で，人間やソフトウェアといった故障がないシステムに対して，FTA や ETA を用いてハザード分析やリスク分析を行うことは難しい．そこで，近年では新たなハザード分析手法として STAMP/STPA[26] が注目されている．STAMP/STPA では，システムが持つ機能間の相互作用から発生するハザードと，ハザードに至るまでのシナリオを識別することが可能となっている．そして，各シナリオに対してハザードが発生しないようにするための対策を検討する．

しかしながら，実際にシステムを開発する上では，開発期間やコストといった

制約から，識別された全てのシナリオに対する対策を検討し反映することが難しい場合があると考えられる．また，STAMP/STPA は，リスク分析を目的とした手法ではないため，手法単体ではシナリオのリスクによる優先付けを行うことはできない．

そこで本研究では，統計的モデル検査法 [23] を利用して，STAMP/STPA で識別されたシナリオに対するリスク分析を行う手法を提案する．特に，統計的モデル検査法でシナリオの発生確率を計算するために，STAMP/STPA の分析結果からモデル検査で検証を行うための形式モデルを系統的に変換する手法の提案を行う．また，提案手法を鉄道の踏切制御システムの例に適用して，手法の有効性を評価する．その結果，STAMP/STPA で識別されたシナリオを形式モデルで表現し，シナリオの発生確率およびリスクを分析することができた．

本論文の構成は以下の通りである．まず，第 2 章で本研究が取り組む課題と目的について述べる．次に，第 3 章で本研究が対象とするハザード分析手法 STAMP/STPA について説明する．第 4 章では，本研究で対象とするハザードの発生確率を計算するために利用する統計的モデル検査法について述べる．また，第 5 章では，本研究の提案手法について述べる．次に，第 6 章では，鉄道システムの事例を用いて提案手法を評価した結果について述べる．第 7 章で本研究の関連研究について説明し，最後に第 8 章で結論と今後の課題についで述べる．

2. 本研究が取り組む課題と目的

安全性や可用性などのシステムの品質を効果的に高めるためには，システムの開発段階におけるハザード分析が重要である．既存の代表的なハザード分析手法には，第 1 章で述べたように FTA や ETA がある．以下に，FTA と ETA による分析例 [13] を図 2 と図 3 にそれぞれ示す．

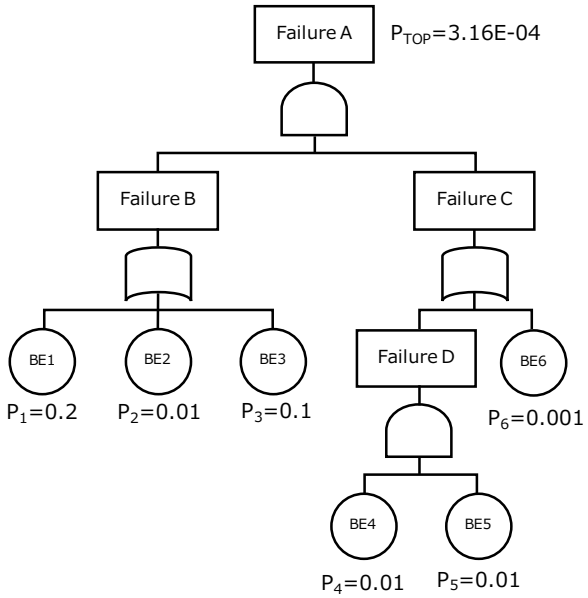

図 2 フォルトツリーの例 [13]

FTA では，最小単位である基本事象から，論理積や論理和を表す記号を用いて，頂点の事象が発生するまでの因果関係を樹形図によって表現する．また，樹形図に含まれる基本事象にその故障確率等を与えることによって，頂点の事象に対する発生確率を算出することが可能となっている．図 2 の例では，BE1 から BE5 までの 5 つの基本事象が存在し，それぞれの故障確率が P_1 から P_6 まで仮定されている．そして，それぞれの故障確率が最終的にトップ事象である Failure A に対する故障確率 P_{TOP} として得られる．

ETA では，時系列で発生するイベントの分岐に対する確率を仮定し，最終的な到達点に至るまでの確率を算出する．図 3 の例では，初期イベントである Initial

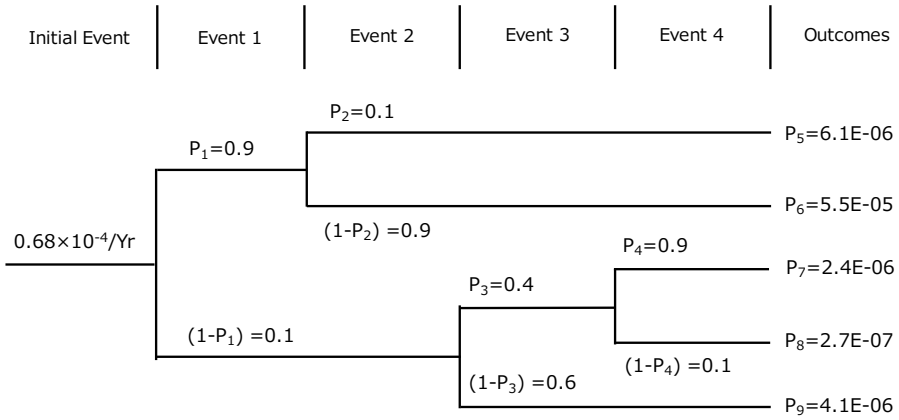

図 3 イベントツリーの例 [13]

Event から，分析結果である Outcomes までの各分岐が示されている．また，各分析点には事象の発生確率 P_1 から P_4 が設定され，最終的な確率値として P_5 から P_9 が計算される．このようにして，イベントツリー分析では，事象の発生順序や分岐を考慮した分析が可能である．FTA や ETA を行うことで，故障等を原因とした事象からハザードに至る確率を求めることが可能であり，確率値とハザードの深刻度に対する値を用いてリスクを計算することができる．

しかしながら，近年ではハードウェアを中心としたシステムではなく，人間やソフトウェアを中心としたシステムが多く開発されるようになっている．そのため，このような故障を持たない要素を含むシステムに対して，FTA や ETA を用いてハザード分析やリスク分析を行うことは難しい．そこで，近年では新たなハザード分析手法として STAMP/STPA が注目されている．STAMP/STPA では，システムが持つ機能間の相互作用から発生するハザードと，ハザードに至るまでのシナリオを識別することが可能となっている．そして，各シナリオに対してハザードが発生しないようにするための対策を検討する．

一方で，実際にシステムを開発する上では，開発期間やコストといった制約から，識別された全てのシナリオに対する対策を検討し反映することが難しい場合がある [1]．また，STAMP/STPA では，識別されたシナリオに対するリスクを求めることは目的としていない．さらに，STAMP/STPA で識別される故障を前提としないようなシナリオのリスク分析を FTA や ETA によって行う場合，機能間

の動的な相互作用が伴う複雑な振る舞いを全て樹形図として表現する必要があるため，これは現実的に困難である [7].

そこで本研究では，STAMP/STPA で識別されたシナリオに対してリスク分析を行うことによる，シナリオの優先付けを行うための手法の提案を目的とする．具体的には，形式手法の 1 つである統計的モデル検査法を用いてシナリオの発生確率に対する定量的な分析を行う．統計モデル検査法では，シミュレーションベースで振る舞いの検証を行うため，故障を前提としない動的な相互作用によるシナリオのリスクを分析することが可能である．本研究では，特に，STAMP/STPA におけるシナリオ生成用モデルに対し，統計的モデル検査法による形式的検証が可能となるようにモデルを変換するための手法を提案する．また，鉄道の踏切制御システムに対して提案手法を適用し，その有効性を評価する．

本研究の提案手法によって，リスク評価の考え方が用いられている分野に対する効用が期待できる．先に述べたような，鉄道分野の RAMS 規格で定義されているシステム開発の初期段階におけるリスク分析のアプローチとして応用することができる．また，自動車分野の規格である ISO26262 における自動車安全水準 (ASIL) でも同様にハザード分析とリスク分析とその対策の必要性について定義されている．特に，自動運転分野では，ドイツの自動運転戦略 (Pegasus Project)[1] が要求定義の段階からリスク分析を行い，リスクの高いシナリオから優先的にテストを行うための必要性を述べている．本研究は，こうしたシステムにおけるリスク低減を行うためのテストに向けたシナリオの優先順位付けのアプローチの 1 つになると考えられる．

3. ハザード分析手法STAMP/STPA

この章では，本研究が対象とするハザード分析手法であるSTAMP（Systems-Theoretic Accident Model and Processes）/STPA（System-Theoretic Process Analysis）について述べる．STAMP/STPAは，従来型のハザード分析手法における事故発生に対する考え方とは異なる考え方を基礎としたハザード分析手法となっている．そこで，まず第3.1節でその違いについて説明する．続いて，第3.3節以降でSTAMP/STPAによる具体的なハザード分析の手順について述べる．

3.1 事故モデル

事故モデルとは，事故が発生する仕組みを説明するための基礎となる考え方である．ここでは，従来型の事故モデルや，本研究で利用する安全分析手法の基礎となる事故モデルSTAMP[22]について述べる．

3.1.1 従来の事故モデル

これまでに，システムのハザードやリスクを分析するための手法として，FTAやETAがあると述べた．これらの手法は，事故は機械の物理的な故障や人間のオペレーションミスを原因として，それに関連するイベントが連鎖的に発生することによってもたらされるものであるという考え方に基づくものである．こうした事故モデルはドミノモデル[19]と呼ばれ，従来のハードウェアを中心としたシステムに対する基本的な考え方となっている．図4にドミノモデルの概念図を示す．

このように，ドミノモデルは事故に対する決定的な原因が1つ存在し，事故の防止のためにはその原因を突き止めて対策することが重要であるという考え方に基づくものである．しかしながら，現代のソフトウェアを中心とする複雑化したシステムに対しては，必ずしもドミノモデルでは説明できない事例が増加している[22]．そこで，近年ではSTAMPと呼ばれる事故モデルが注目されている．第3.1.2節で，STAMPの概要について説明する．

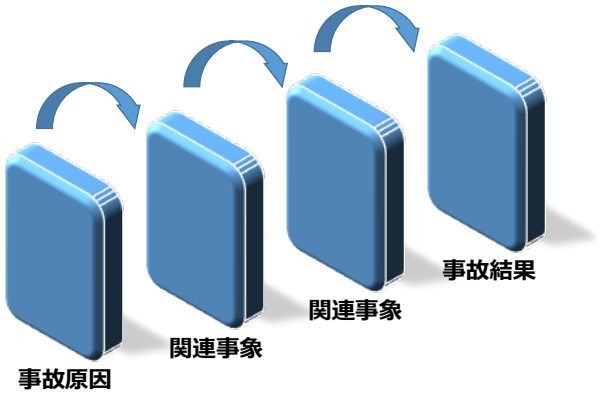

図 4　ドミノモデル

3.1.2　システム理論に基づく事故モデル STAMP

第 3.1.1 節で述べたように，現在開発されている多くのシステムは，その基幹的な役割を担う存在が従来までのハードウェアを中心としたものではなく，ソフトウェアを中心とするものに変化してきている．また，システムや人間が相互に接続されることによって，それらのコミュニケーションの齟齬による事故が増加している．現在，多くのシステム開発において，こうした物理的な故障がなくても発生する事故を防ぐことが重要となっている．そこで近年，システム理論に基づく新たな事故モデルとして STAMP と呼ばれる考え方が注目されている．システム理論におけるシステムの概念を図 5 に示す．

システム理論は，システムの構成要素を個別に捉えるのではなく，全体の振る舞いを扱うことによってその複雑な現象を説明しようとするものである．また，現象はシステムを構成している要素間の相互作用によって創発的に発生するものであることを述べており，これは創発特性と呼ばれている．STAMP は，このシステム理論の考え方に基づき，事故はシステムを構成する機能間における相互作用の中で創発的に発生するとしている．したがって，システムの安全性や運用性などを達成するためにはシステムの創発特性を制御する必要がある．STAMP における相互作用の概念図を図 6 に示す．STAMP では，機能間の相互作用の中で，コントローラと呼ばれる要素がコントロールアクションと呼ばれる指示を出力し，それによってシステムの創発特性が制御されると考える．また，システムはコン

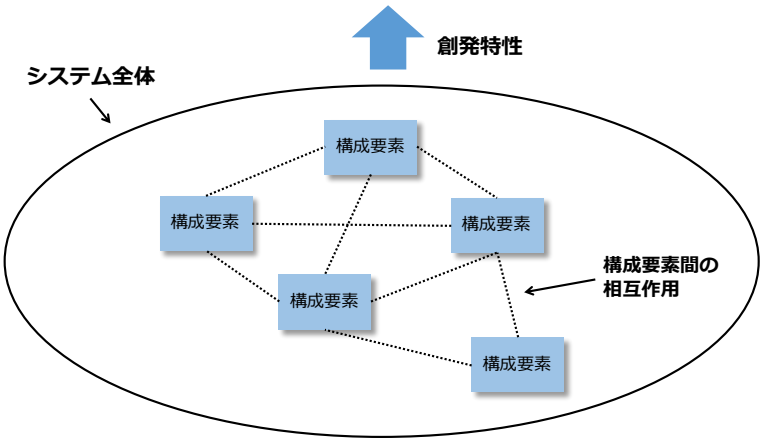

図 5　システム理論におけるシステムの概念

トローラにフィードバックを送信し，コントローラはその情報をもとに自身のシステムに対する認識を更新する．コントローラは自身が持つシステムに対する認識に基づいて安全のための制御を行う．こうした STAMP の考え方に基づいたハザード分析手法として提案されているものが STAMP/STPA である．

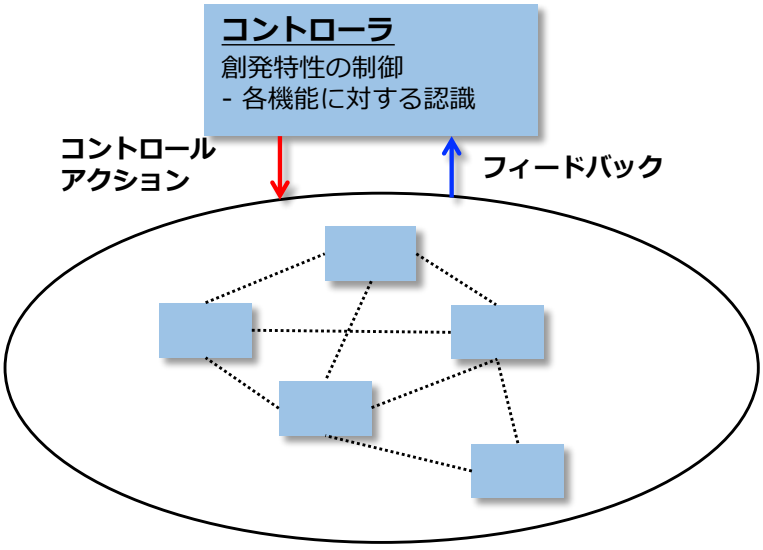

図 6　STAMP における相互作用の概念図

3.2 STAMP/STPA におけるハザード分析の手順

ここでは，STAMP/STPA のハンドブック [26] に基づいた，ハザード分析手順の概要について説明する．図 7 に，ハザード分析の手順を示す．

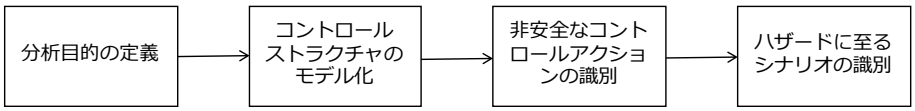

図 7 STAMP/STPA の手順

STAMP/STPA によるハザード分析は，図 7 に示すように 4 つの手順から構成される．分析目的の定義では，分析対象となるシステムやその境界について定義し，前提とするハザードや安全制約などを決定する．この詳細は第 3.3 節で述べる．また，コントロールストラクチャのモデル化では，分析対象の制御構造を抽象的に記述する．詳細は第 3.4 節で述べる．次に，モデル化したコントロールストラクチャをもとに，表を用いて非安全なコントロールアクション網羅的に識別する．この具体的な方法は，第 3.5 節で説明する．最後に，非安全なコントロールアクションから，ハザードに至るまでのシナリオをいくつかのハザード要因の分類をもとに識別する．この詳細は，第 3.6 節で述べる．

3.3 分析目的の定義

まず，STAMP/STPA の最初に行われる分析目的の定義について述べる．分析目的の定義は次の 4 ステップから構成される．以下では，各ステップの詳細について説明する．

1. 損失の識別
2. システムレベルにおけるハザードの識別
3. システムレベルにおける安全制約の識別
4. ハザードの精密化 (状況に応じて実施する)

3.3.1 損失の識別

まず初めに，損失の識別についての概要を説明する．損失とは，システムに関わる利害関係者にとって受け入れられないような損害を意味する．損害の例としては，人命の喪失やミッションの喪失，評判の損失など様々なものがある．こうした対象システムにおける損失が分析の初期段階で定められる必要がある．本研究では，列車の踏切制御システムを対象として，人命に関わる損失や，踏切周辺における交通渋滞の発生といった経済的な損失等を定義している．

3.3.2 システムレベルにおけるハザードの識別

次に，システムレベルにおけるハザードの識別について述べる．ここでのシステムとは，分析対象における機能 (コンポーネント) の集合であり，ハザードとは，先に述べたような損失につながる可能性を持つ状態の集合を指す．システムレベルのハザードを識別するためには，システムの境界を要求仕様として定義する必要がある．要求仕様を定めることによって，前節で述べた損失に至る可能性があるハザードを識別する．ただし，ここではシステムレベルにおけるハザードを識別することを目標とするため，物理的なコンポーネントに対する故障等の詳細な原因を含めないものとする．システムレベルのハザードの例としては，「踏切における遮断器が下がらない状態で列車が通過する」などが考えられる．また，ハザードは分析が進むことで随時追加される場合がある．

3.3.3 システムレベルにおける安全制約の識別

第 3.3.2 節で述べたようなシステムレベルのハザードを識別した後は，それらに対するシステムレベルの安全制約を導出する．ここで，システムレベルの安全制約とは，システムレベルのハザードを防ぎ，最終的にはそれらに対応する損失が発生しないようにするための制約である．この段階におけるシステムレベルの安全制約は，先に識別されたシステムレベルのハザードと同程度の抽象度であるとする．識別される安全制約もハザードの場合と同様に，安全分析の過程で都度追加される可能性がある．前節で示したシステムレベルのハザードに対応する安

全制約の例としては，「列車が踏切を通過する際には遮断器が下がっていなければならない」といったものが考えられる．

3.3.4 システムレベルにおけるハザードの精密化

ここでは，システムレベルのハザードを精密化することについて述べる．システムレベルの各ハザードが識別された後に詳細化を行うことで，より詳細な安全制約を検討することができる．これらは，安全対策を反映したコントロールストラクチャのモデル化に役立てることができる場合がある．第 3.3.2 節におけるハザードの例を詳細化すると，「列車が踏切に接近する速度に対して遮断器が下がる動きが遅い」などが考えられる．一方で，こうした精密化されたハザードに対する安全制約の例として，「遮断器を下げる動作は列車の最高速度に合わせなければならない」などが考えられる．

3.4 コントロールストラクチャのモデル化

STAMP/STPA では，システムは安全のための制御を行う要素であるコントローラ (controller) と，制御される対象となる要素の被コントロールプロセス (controlled process) から構成されており，これらが適切に動作することによってシステムの安全が実現されるとする．各要素は機能単位でモデル化され，コンポーネント (component) と呼ばれる．さらに，外部環境 (external environment) もコンポーネントとして表現することもある．こうしたコンポーネント間の制御構造をコントロールストラクチャ (control structure) と呼ぶ．図 8 に典型的なコントロールストラクチャの例を示す．

コントロールストラクチャに含まれるコントローラは，被コントロールプロセスに対する認識を表すプロセスモデル (process model) と，プロセスモデルに基づいて対象の制御を行うためのアルゴリズムであるコントロールアルゴリズム (control algorithm) を持つ．プロセスモデルは，被コントロールプロセスから受け取るフィードバック (feedback) によって更新されることを基本とする．コントローラは，このプロセスモデルとコントロールアルゴリズムをもとに，コントロー

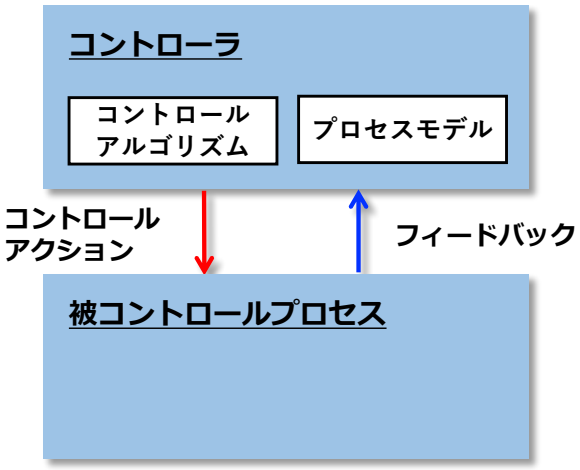

図 8　コントロールストラクチャ

ルアクション (control action) を出力して被コントロールプロセスを制御する．このような，コントロールアクションとフィードバックによる機能間のループ構造をコントロールループ (control loop) と呼ぶ．STAMP/STPA におけるフィードバックの用語は制御理論に由来する．しかし，コントロールストラクチャでは，分析対象の情報の流れやエネルギーの伝達をコントロールアクション，それ以外の流れをフィードバックと呼ぶため，例えば，フィードバックは必ずしもコントロールアクションに対応するものではない．一般的に，コントロールストラクチャはコンポーネント間の相互作用を表現する手段であるため，それらの振る舞いに対する情報は完全には含まれないことに注意する．本研究では，リスク分析を行うことを目的として，各コンポーネントに対して振る舞い情報を与える．この詳細については，第 5 章で説明する．

3.5　非安全なコントロールアクションの識別

ここでは，作成されたコントロールストラクチャに対して，非安全なコントロールアクション (Unsafe Control Action, UCA) を識別する．非安全なコントロールアクションの識別は UCA 表を用いて行う．表 1 に，本研究における UCA 表の一部を例として示す．

表 1 UCA 表の例

コントロールアクション	与えられないとハザード	与えられるとハザード	早過ぎ，遅過ぎ，誤順序	早すぎる停止，長すぎる適用
鳴動開始	警報が鳴らずに列車が通過する	列車が存在しないが警報が鳴動する	警報が鳴動する前に列車が踏切に到達する	列車通過後も警報が鳴動を続ける

UCA 表はコントロールストラクチャに含まれる全てのコントロールアクションに対して行う．この例では，鉄道の踏切システムを想定したものとなっている．表に含まれているコントロールアクションの「鳴動開始」とは，踏切が列車の接近を検知して警報を鳴らすための指示である．STAMP/STPA では，選択したコントロールアクションに対して，4 つのガイドワードをもとに非安全な事象を識別する．表 1 では，赤で記述された 4 つの UCA が存在する．「与えられないとハザード」では，本来出力されるべきコントロールアクションが出力されないことによって，システムがハザードに至る過程を記述する．「与えられるとハザード」では，コントロールアクションが出力されることによってハザードに至る過程について検討する．「早過ぎ，遅過ぎ，誤順序」では，コントロールアクションが出力されるタイミングのずれを原因としてハザードが発生する場合について記述する．最後に，「早過ぎる停止，長過ぎる適用」では，コントロールアクションが適用される長さによって生じるハザードについて検討する．ここで識別した各 UCA に対して，後のステップでシナリオを検討する．

3.6 ハザードに至るシナリオの識別

非安全なコントロールアクションを識別した後，システムがハザードに至るまでのシナリオを識別する．シナリオを識別するために，図 9 に示すようなコントロールループを考える．

ここで識別されるコントロールループには，具体的なシナリオの識別を行うためにアクチュエータ (actuator) とセンサ (sensor) が含められる．アクチュエータとは，コントローラが出力したコントロールアクションを物理的な動作に変換す

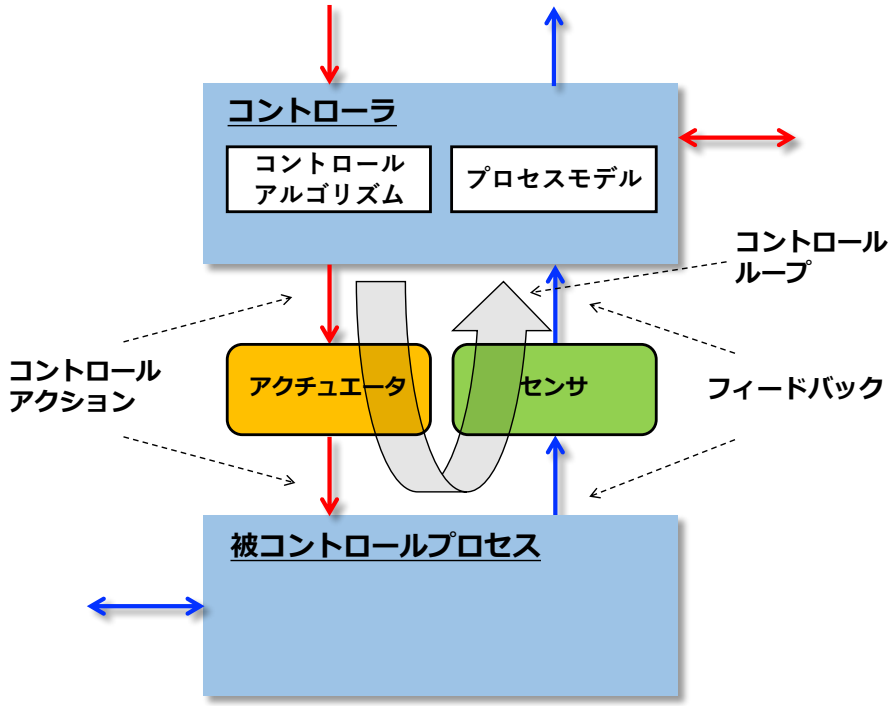

図 9　コントロールループ

るための機能である．また，センサはコントローラがフィードバックを受信するための装置である．こうしたコントロールループをもとに，STAMP/STPA のハンドブック [26] では，以下に示す 2 つの種類のシナリオを考慮する必要があると述べている．

(a) 非安全なコントロールアクションにつながるシナリオ

(b) コントロールアクションが不適切な実行をされる，または実行されずにハザードに至るシナリオ

シナリオを識別するためには，図に示すようなコントロールループをもとに検討し，その後にハザード要因を導く．以降では，**(a)** と **(b)** のそれぞれに対応するシナリオの識別するために検討するべきハザード要因の分類について述べる．

3.6.1 非安全なコントロールアクションにつながるシナリオ

ここでは，コントローラが非安全なコントロールアクションを出力するための原因について述べる．非安全なコントロールアクションを出力する原因は，以下の 2 つの分類をもとに検討する．

(a-1) 非安全なコントローラの動作

(a-2) 不適切なフィードバックと情報の原因

ここで，**(a-1)** の非安全なコントローラの動作とは，コントローラが含むコントロールアルゴリズムやプロセスモデル，他のコントローラの入出力等によって，不適切なコントロールアクションを出力することに対する分類である．**(a-2)** における不適切なフィードバックと情報の原因では，被コントロールプロセスや他のコンポーネントから入力されるフィードバックによって，結果的にコントローラが非安全なコントロールアクションを出力することの分類である．図 10 に，コントロールループにおけるそれぞれの分類が対象とする範囲を示す．

まず最初に，上記における非安全なコントローラの動作について述べる．非安全なコントローラの動作は，一般的に以下の 4 つに分類される．

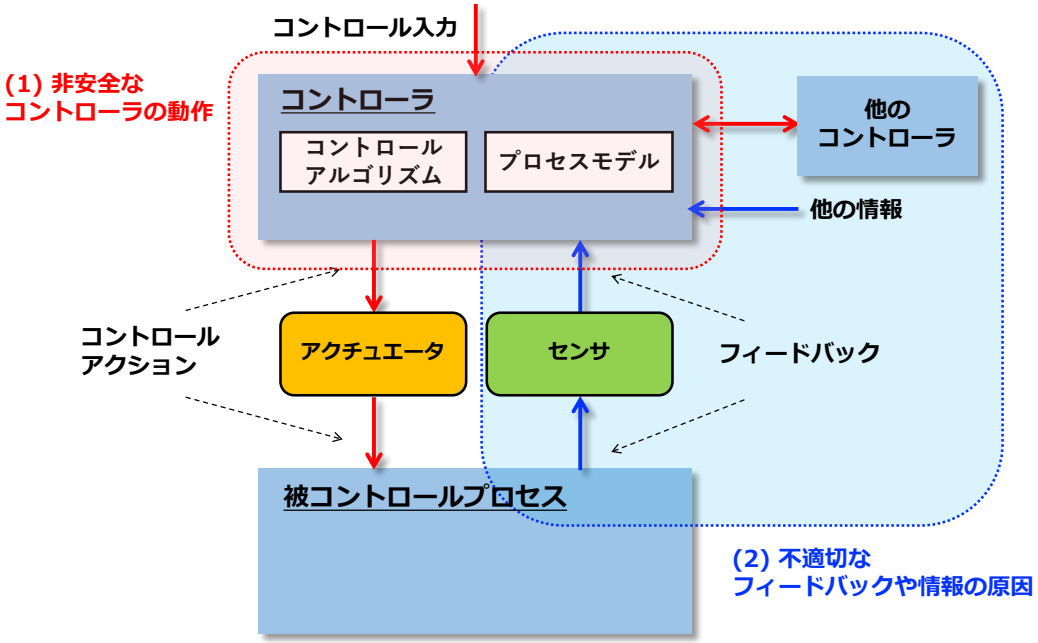

図 10　非安全なコントローラの動作と不適切なフィードバックの範囲

(a-1-1) コントローラの物理的な故障

(a-1-2) 不適切なコントロールアルゴリズム

(a-1-3) 非安全なコントロール入力

(a-1-4) 不適切なプロセスモデル

ここで，コントローラの物理的な故障とは，コントローラの経年劣化や外部環境からの衝撃等でハードウェアが故障し，適切なコントロールアクションが出力されないことを意味している．次に，不適切なコントロールアルゴリズムでは，コントロールアクションを出力するためのアルゴリズムに問題があり，結果として非安全な制御を行ってしまうことによる分類である．一般的に，コントロールアルゴリズムの欠陥には，以下のような原因がある．

(a-1-2-1) コントロールアルゴリズムの誤った実装

(a-1-2-2) コントロールアルゴリズムの仕様に対する欠陥

(a-1-2-3) 環境の変化により，コントロールアルゴリズムが不十分となる

また，非安全なコントロール入力は，他のコントローラからのコントロールアクションを受け取ることで対象コントローラが非安全な動作に至ることに対する分類である．

最後に，不適切なプロセスモデルに対する原因の分類を示す．プロセスモデルは，コントローラが持つ被コントロールプロセスに対する認識のモデルである．以下は，そのプロセスモデルが現実と一致しない，あるいはプロセスモデル同士に齟齬があることに対する分類である．一般的には，以下のような原因があるとされる．

(a-1-4-1) コントローラが間違ったフィードバックや情報を受けとる

(a-1-4-2) コントローラが正しいフィードバックや情報を受けとっているが誤って解釈したり無視する

(a-1-4-3) コントローラが必要に時にフィードバックや情報を受けとっていない (遅れている，あるいは全く受けとらない)

(a-1-4-4) 必要なコントローラのフィードバックや情報が存在しない

次に，不適切なフィードバックと情報の原因について示す．不適切なフィードバックと情報の原因は大きく 2 つに分類される．

(a-2-1) フィードバックや情報を受信しない

(a-2-2) 不適切なフィードバックを受信する

上記の **(a-2-1)** では，被コントロールプロセスからのフィードバックがコントローラやセンサの問題で受信されないことを意味している．また，**(a-2-2)** では，フィードバックを受信したが，性能等の問題でコントローラが正しく受信できない場合に相当する．**(a-2-1)** のフィードバックや情報を受信しないことの詳細な原因として，以下がある．

(a-2-1-1) センサがフィードバックや情報を送信したが，コントローラが受信しない

(a-2-1-2) フィードバックや情報が送信されていないにも関わらず，まるでセンサやコントローラが受信したかのように振る舞う

(a-2-1-3) フィードバックや情報が送信されているにも関わらず，センサやコントローラが受信しない

(a-2-1-4) フィードバックや情報がコントロールストラクチャーに存在しない，またはセンサが存在しない

最後に，**(a-2-2)** の不適切なフィードバックを受信することの詳細な原因として，以下の分類がある．

(a-2-2-1) センサは適切に応答しているが，コントローラが不適切なフィードバックや情報を受信する

(a-2-2-2) センサがフィードバックや情報に対して不適切に応答する

(a-2-2-3) センサに必要なフィードバックや情報を提供するような性能がない，あるいは，そのように設計されていない

3.6.2 コントロールアクションが不適切な実行をされる，または実行されずにハザードに至るシナリオ

続いて，**(b)** の分類である，コントロールアクションが不適切な実行をされる，または実行されずにハザードに至るシナリオの識別について示す．ここでは，以下の 2 つの原因について検討する．

(b-1) コントロール経路を含むシナリオ

(b-2) コントロール対象のプロセスに関連するシナリオ

ここで，コントロール経路とは，コントロールアクションがコントローラから出力されて，アクチュエータを経由した後，被コントロールプロセスまで作用する経路のことである．また，コントロール対象プロセスに関連するシナリオでは，被コントロールプロセスにおける入出力や外乱について考慮する．図 11 に，コントロールループにおける分類 **(b-1)** と **(b-2)** の範囲を示す．

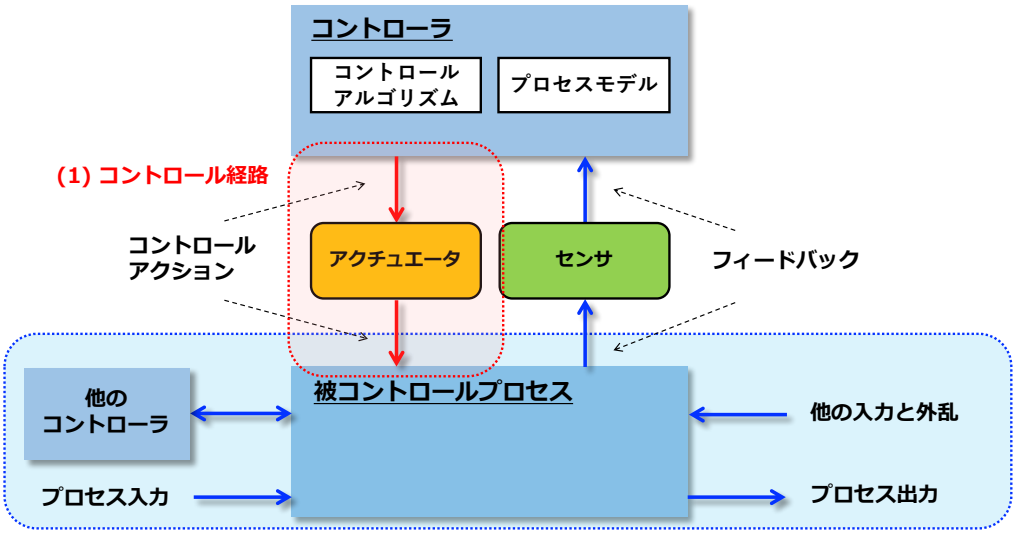

図 11　コントロール経路とコントロール対象プロセスにおける要因

上記 **(b-1)** のコントロール経路を含むシナリオには，大きく分けて 2 つの分類がある．

(b-1-1) コントロールアクションが実行されない

(b-1-2) コントロールアクションが不適切に実行される

すなわち，**(b-1-1)** は，コントロールアクションがコントローラから出力されても，そのアクションがアクチュエータの問題によって被コントロールプロセスに作用しない場合に相当する．また，**(b-1-2)** では，コントローラからのコントロールアクションは受信されても，アクチュエータが意図した通りに動作しないことを述べている．**(b-1-1)** の詳細な原因として，以下がある．

(b-1-1-1) コントローラはコントロールアクションを送信したが，アクチュエータが受信しない

(b-1-1-2) コントロールアクションをアクチュエータは受信したが，アクチュエータが応答しない

(b-1-1-3) アクチュエータが応答するが，コントロールアクションが適用されない，あるいはコントロール対象のプロセスが受信しない

また，**(b-1-2)** の詳細な分類として，以下が存在する．

(b-1-2-1) コントローラはコントロールアクションを送信したが，アクチュエータは不適切な受信をする

(b-1-2-2) コントロールアクションをアクチュエータが受信したが，アクチュエータが不適切に応答する

(b-1-2-3) アクチュエータが適切に応答するが，コントロールアクションが不適切にコントロール対象のプロセスで適用または受信される

(b-1-2-4) コントロールアクションをコントローラが送信していないのに，まるで既に送信しているかのようにアクチュエータまたは他の部分が応答する

続いて，**(b-2)** であるコントロール対象のプロセスに関連するシナリオに関しては，大きく分けて 2 つの分類がある．

(b-2-1) コントロールアクションが実行されない

(b-2-2) コントロールアクションが不適切に実行される

すなわち，**(b-2-1)** では，コントロールアクションがアクチュエータを経由して正しく伝達されているが，被コントロールプロセスの問題によって実行されない場合を説明している．また，**(b-2-2)** では，**(b-2-1)** と同様にコントロールアクションが正しく受信されるが，その動作がコントローラの意図した振る舞いではないことを意味している．**(b-2-1)** の詳細な原因として，以下があるとされる．

(b-2-1-1) コントロール対象のプロセスによって，コントロールアクションが適用される，または受信されるが，コントロール対象のプロセスは応答しない

また，**(b-2-2)** では，以下に示す原因があるとする．

(b-2-2-1) コントロール対象のプロセスによって，コントロールアクションが適用される，または受信されるが，コントロール対象のプロセスは不適切に応答する

(b-2-2-2) コントロール対象のプロセスによって，コントロールアクションが適用または，受信されないが，まるでコントロールアクションが適用または受信されたかのようにプロセスが応答する

4. 統計的モデル検査法

この章では，本研究でハザードに至るシナリオの分析を行うために用いる統計的モデル検査法の概要について説明する．まず，一般的なモデル検査法 [8] の概要について説明し，その後に統計的モデル検査法の形式的な定義やモデル化および，検証方法について説明する．

4.1 統計的モデル検査法の概要

モデル検査法 (model checking)[8] とは，形式検証技術 (formal verification) の 1 つであり，システムを数学的にモデル化してその状態を網羅的に探索して検証する手法である．モデル検査法では，まずシステムの設計仕様をもとに，形式モデルを作成する．その一方で，システムに対する要求仕様から論理式を導出する．そして，その両方を入力として，形式モデルが論理式を満たすか否かを検証し，満たされない場合は反例を出力する．検証で得られた反例は，設計仕様を修正するために用いられ，反例が出力されなくなるまでモデル化と検証が繰り返し実施される．

システムやソフトウェア開発において，一般的に行われている検証活動である動的テスト (dynamic testing) は，モデル検査法のような形式検証技術の対照的な手法とみなすことができる (以下，動的テストを単にテストと呼ぶ)．テストの基本的な考え方は，人が事前にテストのための入力データを用意し，期待される出力データが得られるか否かを確認するというものである．テストではシステムを動的に実行させて検証することが可能であるため，最終的な実装が要求仕様を満たしていることを確認できるという利点がある．一方で，テストのみでは全ての入力値を網羅的に用意することは現実的に難しいため，システムに欠陥が含まれていないことを証明することは困難である．モデル検査法では，先に述べたように，システムをモデル化して振る舞いを網羅的に確かめる手法であるため，欠陥が含まれていないことの証明を行うことができる．しかしながら，システムの規模が大きくなると，モデルの状態が非常に大きくなり検証ができなくなる状態爆発 (state space explosion)[8] と呼ばれる問題がある．こうした問題を回避するた

めの一般的な方法として，検証する範囲を限定することによってモデルの状態数を減らす抽象化と呼ばれる手法が存在する．また，その他の手法として，本研究が対象とする統計的モデル検査法が提案されている．

統計的モデル検査法 (statistical model checking)[23] とは，一般的なモデル検査法における厳密な状態探索を行わずに，シミュレーションベースで状態を探索して検証する手法である．より具体的には，モデル化された状態空間に対してモンテカルロシミュレーションを行い，得られた標本パスの集合から統計的な推定を行うものとなっている．統計的モデル検査法では，シミュレーション可能であれば状態が有限でなくても検証を行うことが可能であり，代表的なツールとして，PLASMA Lab[17] や Uppaal SMC[10] がある．ここでは，本研究で利用する Uppaal SMC について述べる．

Uppaal SMC とは，Uppaal[25] と呼ばれるモデル検査ツールに対して統計的分析を行うための拡張を行ったものである．Uppaal SMC では，Uppaal と同様に実時間システムをモデル化して検証することができるという特長がある．実時間システムは，確率時間オートマトンによってモデル化する．以降では，この確率時間オートマトンの形式的な定義について説明する．

4.2 Uppaal SMC における形式的なモデルの定義

以下では，Uppaal SMC におけるモデルの形式的な定義は，確率時間オートマトンの一般的な定義 [11] に基づいて行う．ただし，本研究は STAMP/STPA で識別されたシナリオを形式モデル上で再現することを目的としているため，それに適する定義を行う．

X をクロック変数 (clock variable) の集合とする．クロック変数とは，時間経過を表現するための非負の実数値を取るリセット可能な変数である．クロック変数を用いたガード (遷移条件) は $x \sim n$ という形式で表される．ただし，$x \in X, n \in \mathbb{N}, \sim \in \{<, \leq\} (\sim \in \{>, \geq\})$ である．また，X を用いたガードの上界と下界をそれぞれ $\mathcal{U}(X), \mathcal{L}(X)$ とする．

定義 1. 確率時間オートマトンは，組 $(L, \ell_0, X, \Sigma, E, R, I)$ で表される．ここで，

各要素は次のもとである．

1. L はロケーション (location) の集合，
2. $\ell_0 \in L$ は初期ロケーション，
3. X はクロック変数の集合，
4. $\Sigma = \Sigma_i \uplus \Sigma_o \uplus \{\varepsilon\}$ はアクション (action) の集合，Σ_i は入力アクション (input action)，Σ_o は出力アクション (output action)，ε は空のアクション (empty action)，
5. $E \subseteq (L \times \mathcal{L}(X) \times \Sigma \times 2^X \times L) \cup \bigcup_{n\geq 2} L \times (L \times \mathbb{N})^n$ は遷移関係 (transition relation) の集合，
6. $R\colon L \to \mathbb{Q}_{>0}$ は各ロケーションに対する指数分布のパラメータの割り当て，及び，
7. $I\colon L \to \mathcal{U}(X)$ はロケーションに対する不変条件の割り当て．

ロケーションは状態 (state) とも呼ぶ．また，遷移関係の定義における $\bigcup_{n\geq 2} L \times (L \times \mathbb{N})^n$ は Uppaal SMC における確率的遷移を表すための重み付き辺であり，各辺に重みが割り当てられるものとなっている．遷移関係 $(\ell, g, a, \{x_1, \ldots, x_n\}, \ell')$ は，$\ell \xrightarrow{g,a,x_1,\ldots,x_n} \ell'$ と表現される．ここで，ℓ は遷移元のロケーション，g は遷移のガード，a はアクション，$x_1, \ldots, x_n$ はリセットするクロック変数，ℓ' は遷移先のロケーションをそれぞれ表す．アクションが存在しない場合，すなわち，$a = \varepsilon$ であるとき $\ell \xrightarrow{g,x_1,\ldots,x_n} \ell'$ と記述する．また，クロック変数がリセットされない場合は $\ell \xrightarrow{g,a} \ell'$ と記述する．重み付き辺 $(\ell, ((\ell_1, w_1), \ldots, (\ell_n, w_n)))$ は $\langle \ell \xrightarrow{w_1} \ell_1, \ldots, \ell \xrightarrow{w_n} \ell_n \rangle$ と記述する．ただし，$n \geq 2$ であり，$\ell, \ell_1, \ldots, \ell_n \in L$ である．また，$w_1, \ldots, w_n$ は各辺 $\ell_1, \ldots, \ell_n$ に割り当てられる重みである．

例 1. 図 12 に，確率時間オートマトン $(\{\mathsf{INIT}, \mathsf{S1}, \mathsf{S2}, \mathsf{END}\}, \mathsf{INIT}, \{x\}, \{\varepsilon\}, \{\langle \mathsf{INIT} \xrightarrow{1} \mathsf{S1}, \mathsf{INIT} \xrightarrow{5} \mathsf{S2}\rangle, \mathsf{S1} \xrightarrow{x\geq 2} \mathsf{END}, \mathsf{S2} \xrightarrow{x\geq 6} \mathsf{END}\}, \emptyset, \{\mathsf{INIT} \mapsto x < 1, \mathsf{S1} \mapsto x \leq 4, \mathsf{S2} \mapsto x \leq 8, \})$ の Uppaal SMC による表現とその確率分布の例を示す．

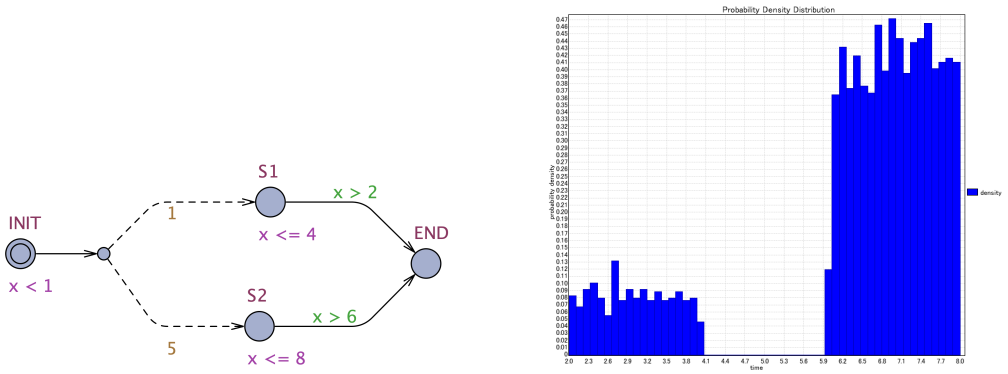

図 12　確率時間オートマトンとその確率分布の例 (1)

上図左側において，丸は状態，矢印は状態遷移を表している．また，二重丸で描かれている状態 INIT は初期ロケーションを表している．この例では，状態 INIT から状態 S1 と状態 S2 への遷移が可能となっている．点線の矢印は確率的な状態遷移を表現している．例えば，この例では状態 S1 と状態 S2 がそれぞれ $1/6$ と $5/6$ の確率で遷移することができる．また，状態の下に描かれている条件式は不変条件である．例えば，状態 S1 では，x<=4 という不変条件が満たされる必要がある．ただし，変数 x はクロック変数と呼ばれる時間経過を表す変数である．したがって，状態 S1 は 4 単位時間以内の滞在が許可されることを示す．そして，状態 S1 は，状態 END に向かって遷移することができる．ただし，辺の上に示されている条件式が満たされている場合に状態遷移が可能である．そのため，例えば状態 S1 から状態 END への遷移は，x>2 が満たされる場合，すなわち，2 単位時間よりも多く時間が経過している場合に遷移することができる．図 12 の右側は，この確率時間オートマトンに対する確率密度分布を表している．グラフの縦軸は確率密度，横軸は時間を表している．この例では，4 単位時間から 6 単位時間の間は状態 S1 に到達することはないので確率密度が 0 となっており，それ以外では確率的遷移に応じた一様分布が形成されていることが確認できる．

例 2. 次に，確率時間オートマトン $(\{\mathsf{INIT}, \mathsf{END}\}, \mathsf{INIT}, \{x\}, \{\varepsilon\}, \{\langle \mathsf{INIT} \to \mathsf{END} \rangle\}, \{\mathsf{INIT} \mapsto 4/1\}, \emptyset)$ と，その確率分布の例を図 13 に示す．

この図の例では，状態 INIT から状態 END への遷移が可能である．状態 INIT の下に示されている $4:1$ は連続型確率分布である指数分布に対するパラメータを

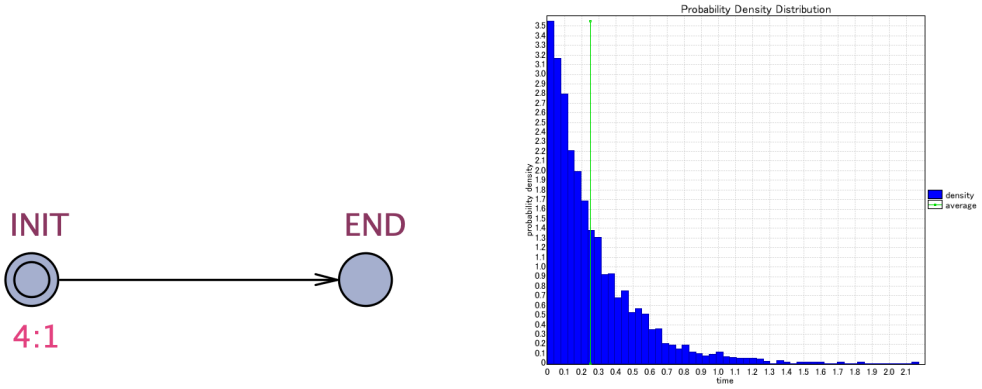

図 13　確率時間オートマトンとその確率分布の例 (2)

表現している．指数分布は，ある事象が一定の発生率で起きることを表現可能な確率分布である．例えば，図 13 では，1 単位時間に平均して 4 回状態 END に遷移することを表現している．このオートマトンに対する確率分布を図 13 の右側に示す．この図から指数分布の形成が確認できる．また，緑色で示された縦棒は平均を表しており，その値が 1/4 単位時間に近い値であることが示されている．

次に，確率時間オートマトンの並行合成について説明する．$A_j = (L^j, \ell_0^j, X^j, \Sigma^j, E^j, R^j, I^j)$，ただし，$\Sigma^j = \Sigma_i^j \uplus \Sigma_o^j$ かつ $1 \leq j \leq n$ となる確率時間オートマトン A_j を考える．確率時間オートマトンの並行合成は $A_1 \mid \cdots \mid A_n$ と表現され，以下が成立する場合に並行合成可能である．

1. $1 \leq j, k \leq n$ において，$j \neq k$ のとき $X^j \cap X^k = \emptyset$,
2. $\bigcup_{1 \leq j \leq n} \Sigma_i^j$ と $\bigcup_{1 \leq j \leq n} \Sigma_o^j$ の間に 1 対 1 の写像が存在する．

定義 2. 共通変数付き確率時間オートマトンネットワーク (network of priced timed automata) とは組 $(V, \mathcal{A})$ である．ここで，V は変数の有限集合，$\mathcal{A}$ は確率時間オートマトンの並行合成である．

状態空間の離散的範囲は $\prod_{x \in V} \mathsf{dom}(x) \times \prod_{(L,\_,\_,\_,\_,\_,\_) \in \mathcal{A}} L$ である．本論文では，確率時間オートマトンは並行合成可能である．出力アクションおよび入力アクションは a!, b!, c!... および，a?, b?, c?... とそれぞれ記述する．図 14 に確率時間オートマトンの並行合成の例を示す．

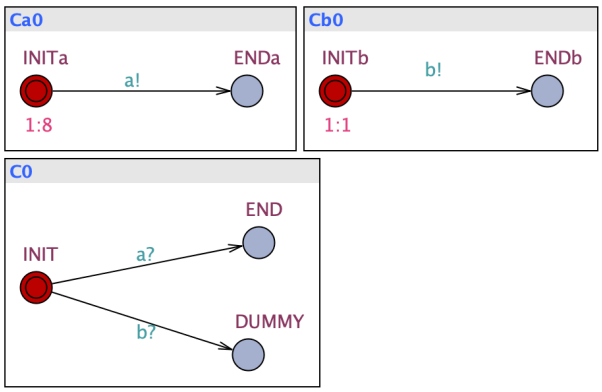

図 14　確率時間オートマトンの並行合成の例

図 14 の例では，3 つの確率時間オートマトン Ca0，Cb0，C0 が並行合成されている．また，Ca0 と Cb0 はそれぞれアクション a!と b!を出力することが可能であり，C0 はどちらか一方のアクションを入力することができる．

確率時間オートマトンにおけるアクションは，UPPAAL においてはチャネル (channel) に対応する．UPPAAL SMC では，**ブロードキャストチャネル** (broadcast channel) と**アージェントブロードキャストチャネル** (urgent broadcast channel) と呼ばれる 2 種類のチャネル型を利用することができる．ブロードキャストチャネルでは，チャネル名を a と仮定すると，1 つのメッセージ送信 a!を 0 個以上の a?によって受信する通信方式である．すなわち，送信側は受信側の状態にかかわらずメッセージを送信することが可能である．ただし，受信可能な場合は全ての a?によってメッセージが受信される．こうしたブロードキャスト通信以外にも，通常の 1 対 1 の通信を表現する必要がある場合には，UPPAAL SMC のチュートリアル [10] に示されている変換方法を行うことによって表現することが可能である．アージェントブロードキャストチャネルは，通常のブロードキャストチャネルの性質に加えて，メッセージを送信可能な状態になった場合，それ以上の時間経過を許さずに同期を行う．例として，図 15 にアージェントブロードキャストチャネルの例を示す．

図 15 では，アージェントブロードキャストチャネルを用いた送信側のオートマトンを示している．この例では，状態 INIT では 10 単位時間までの滞在が許され

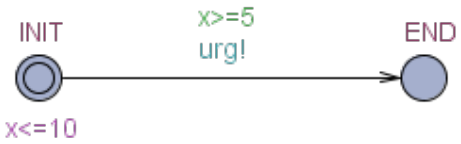

図 15　アージェントブロードキャストチャネルの例

ており，5 単位時間以上で状態 END に遷移可能である．また，エッジには時間に関するガードとして x>=5 があり，チャネルとしてはアージェントブロードキャストチャネルである urg!がある．ここで，仮に urg!が定義されていない場合，状態 INIT から状態 END への遷移は 5 単位時間以上 10 単位時間以下であり，5 単位時間経過直後に遷移するとは限らない．しかし，図のようにエッジにアージェントブロードキャストチャネルを定義することで，ガードの条件式が満たされた瞬間に遷移するといった振る舞いを表現することができる．したがって，この例では 5 単位時間が経過した時点で状態 INIT から状態 END に遷移する．

4.3 Uppaal SMC におけるクエリの記述

Uppaal SMC では，通常のモデル検査法と同様に，モデルを検証するために論理式で記述されたクエリ (query, 問い合わせ式) を与える必要がある．クエリは検証時間の範囲を表す *bound* を用いて以下のように記述できる．

$$\Pr[bound](\varphi)$$

ここで，$\Diamond$ は性質 φ がいつか満たされることを表す．ここで，φ は，以下のように BNF 記法で定義される [10].

$$\begin{aligned} \varphi \ ::= \ & \texttt{BExpr} \mid \varphi \ \texttt{and} \ \varphi \mid \varphi \ \texttt{or} \ \varphi \\ \mid \ & \varphi \ \texttt{U[a,b]} \ \varphi \mid \varphi \ \texttt{R[a,b]} \ \varphi \mid \texttt{<>[a,b]} \ \varphi \mid \texttt{[][a,b]} \ \varphi \end{aligned}$$

上記の定義において，$a, b \in \mathbb{N}$ かつ $a \leq b$ である．また，BExpr はクロック変数やロケーションに関するブール演算式である．例えば，あるモデルに対して，100

単位時間以内に状態 state==true が成立する確率を計算したい場合，モデル検査器に与えるクエリは以下のように記述することができる．

Pr[<=100] (<> state==true)

次に，確率時間オートマトンに含まれるある状態に到達する確率を計算したい場合のクエリの記述方法について説明する．Uppaal SMC では，確率時間オートマトンはテンプレートとして定義される．例えば，先に示した図 14 の例では，3 つのテンプレートが並行合成されており，それぞれのテンプレート名は Ca0，Cb0，C0 である．この場合，例として，100 単位時間以内にテンプレート C0 における状態 END に到達する確率を計算するために必要なクエリは以下のように記述される．

Pr[<=100] (<> C0.END)

検証結果は，ある確率値 $p = Pr(\varphi)$ に対して，信頼係数 $1-\alpha$ の信頼区間 $[p-\varepsilon,\ p+\varepsilon]$ で与えられる．ここで，α と ε は検証時に設定するパラメータであり，α は信頼パラメータ (confidence parameter)，ε は近似パラメータ (approximate parameter) と呼ばれる [15].

5. 提案手法

この章では，本研究の提案手法について述べる．図 16 に本研究における提案手法の流れを示す．

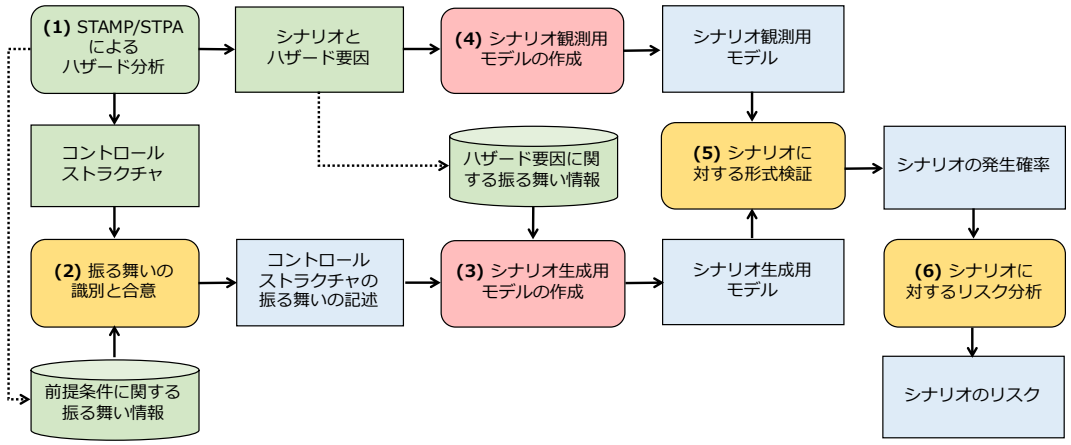

図 16　本研究における提案手法の手順

図中の角丸四角形は手続きを示し，四角は手続きによる成果物を表している．また，円柱は外部のデータを表す．提案手法は大きく 6 つの手続きによって構成される．本研究では，認識の齟齬によるシナリオを形式的モデルに表現するため，特に図の赤で示された (3) と (4) の形式化に関する手法を提案する．以下に，提案手法の各手順を示す．

5.1 手順 1：STAMP/STPA によるハザード分析

まず最初に，図 16 の (1) に示されている STAMP/STPA によるハザード分析を行う．基本的には，第 3 章で説明した STAMP/STPA の手順に従って分析を行う．すなわち，システムレベルのハザードと安全要求を識別してからコントロールストラクチャを作成する．その後，非安全なコントロールアクションとハザードに至るシナリオを識別し，ハザード要因について検討する．STAMP/STPA による識別結果であるコントロールストラクチャとハザードに至るシナリオ，ハザード要因に関する情報は次の手順で用いられる．

5.2 手順 2：振る舞いの識別と合意

ここでは，UML や SysML で対象システムの振る舞いを記述し合意する．手順の流れを図 17 に示す．

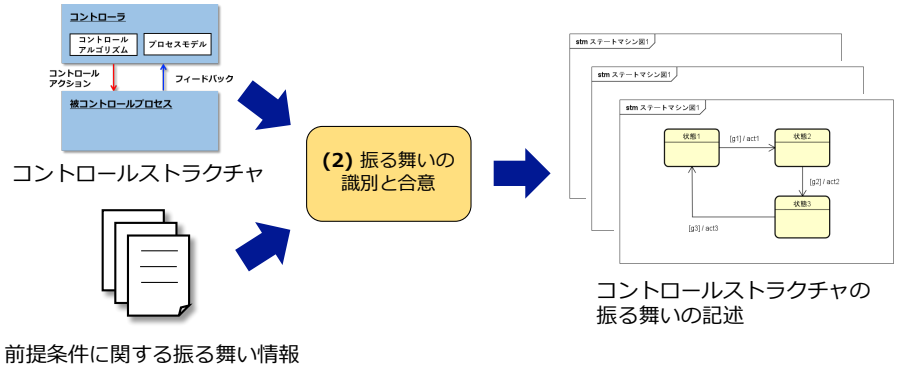

図 17　手順 2 の流れ

第 3.4 節でも述べたように，コントロールストラクチャはハザード分析を行うためのモデルであるため，機能に対応する振る舞いは持たない．そこで，形式的モデルを作成するためには，外部の情報をもとに機能に対する振る舞いを与える必要がある．しかしながら，一般的に，STAMP/STPA のコントロールストラクチャに含まれる機能の振る舞いは，厳密に検討すれば非常に複雑なものとなる．例えば，本研究では鉄道システムを分析対象として「列車」を 1 つのコンポーネントとしてみなすが，列車には走行機能や通信機能など様々な機能が含まれる．本来であれば，こうした機能の全てについて検討すべきであるが，そうした振る舞いを全て列挙してモデル化することは現実的に難しい．そこで，本研究では，コントロールストラクチャに含まれる機能に対して，手順 1 で識別された前提条件に関連する部分に限定して振る舞いを与える．振る舞いは UML[1]や SysML[2]を用いて記述しておき，事前に他の利害関係者とレビュー等を行い合意を得る．

また，振る舞いに関する情報の他に，本研究では振る舞いに関連する確率的な振る舞いを持たせる必要がある．例えば，第 5 章で説明する事例では，列車が定

[1]http://www.uml.org/
[2]http://www.omgsysml.org/

期的に出現する頻度や信号の切り替えタイミングなどを確率値で表現している．ただし，本来はこうした振る舞いは，鉄道のダイヤや信号のロジック等を基準にすれば確率値ではなく決定的なものとなる．しかし，上記でも述べたように，全ての振る舞いを詳細にモデル化，分析することは難しい．そこで，本研究では，振る舞いに含まれる確率値を，これまでの運用から得た統計データをもとに近似して表現する．ただし，確率値によって近似する箇所やその程度によって，モデル上の振る舞いと本来の振る舞いの乖離が大きくなってしまう可能性がある．したがって，分析の目的に応じて，厳密に振る舞いを与える部分と確率値によって近似する箇所を検討して決定する必要がある．一方で，システムの新規開発を行う場合には，モデル化する上で過去の運用データなどを直接参照できない可能性がある．このような場合は，新規開発部分で再利用されている既存のコンポーネントなどを調べ，類似システムの運用データ等から適切に見積もる必要があると考えられる．

5.3 手順 3：シナリオ生成用モデルの作成

次に，手順 2 で得られた，コントロールストラクチャの振る舞いの記述をもとにシナリオ生成用モデルを作成する．形式化の方法は，ハザードに至るシナリオをモデル上に表現することを目的として，STAMP/STPA のハンドブック [26] を参考に検討した．まずは，シナリオ生成用モデルの形式的な定義とその例を示す．その後，シナリオ生成用モデルの概念図を示し，シナリオを生成するためのハザード要因を形式的に表現するための考え方について説明する．

5.3.1 シナリオ生成用モデルの形式的定義

ここでは，シナリオ生成用モデルを形式的に定義するにあたり，まずはコンポーネントの相互作用のみの情報を含むモデルを導入する．

定義 3. コンポーネントインタフェース (component interface) は，組 $(CA_{in}, CA_{out}, FB_{in}, FB_{out}, PM)$ で表される．ただし，CA_{in}，CA_{out}，FB_{in}，FB_{out} は集合であり，CA_{in} および CA_{out} の要素をコントロールアクション，FB_{in} および FB_{out} の要素

をフィードバックと呼ぶ．PM は値の範囲が有限の変数であり，プロセスモデル変数と呼ぶ．

コンポーネントインタフェース $C = (CA^0_{in}, CA^0_{out}, FB^0_{in}, FB^0_{out}, PM^0)$ とすると，$CA_{in}(C)$ は CA^0_{in}，$PM(C)$ は PM^0 を表す．また，$CA(C)$ は $CA_{in}(C) \cup CA_{out}(C)$ を表し，$FB(C)$ は $FB_{in}(C) \cup FB_{out}(C)$ を表す．ただし，ここでのコンポーネントインタフェースは STAMP におけるコントロールアルゴリズムを含んでいないものとする．また，$CA(C) \cup FB(C)$ における要素は**コンポーネント間チャネル** (inter-component channel) と呼び，コンポーネント C におけるコンポーネント間チャネルは $ICC(C)$ と記述し，$ICC(C) = CA(C) \cup FB(C)$ である．

定義 4. コントロールストラクチャインタフェース (control-structure interface) CSI は，以下を満たすコンポーネントインタフェースの有限集合である．

- CSI におけるコンポーネントインタフェース C および，$CA_{out}(C)$ におけるコントロールアクション a に対して，$C' \neq C$ かつ $a \in CA_{in}(C')$ を満たすようなコンポーネントインタフェース C' が存在する．
- CSI におけるコンポーネントインタフェース C および，$FB_{out}(C)$ におけるフィードバック f に対して，$C' \neq C$ かつ $f \in FB_{in}(C')$ を満たすようなコンポーネントインタフェース C' が存在する．

変数 x に対して，x が取り得る値の範囲を $\mathsf{dom}(x)$ と記述する．

例 3. 図 18 にコントロールストラクチャインタフェースの例を示す．ただし，$\mathsf{Controller1} = (\emptyset, \{\mathsf{ca1}\}, \{\mathsf{fb1}\}, \emptyset, \{\mathsf{u}\})$，$\mathsf{Controller2} = (\emptyset, \{\mathsf{ca2}\}, \{\mathsf{fb2}\}, \emptyset, \{\mathsf{v}\})$，$\mathsf{ControlledProcess1} = (\{\mathsf{ca1}, \mathsf{ca2}\}, \emptyset, \emptyset, \{\mathsf{fb1}, \mathsf{fb2}\}, \emptyset)$，$\mathsf{dom}(u) = \{0, 1\}$，$\mathsf{dom}(v) = \{\mathsf{true}, \mathsf{false}\}$ である．

次に，コンポーネントの形式的な定義を与える．直感的には，コンポーネント一つに対して，オートマトンを一つ対応させることにより，コントロールアルゴリズムなどの振る舞いを含むコンポーネントを表すことができる．ただし，第 3.6 節で述べたようなハザード要因も含むようなシナリオを系統的に表現するため，

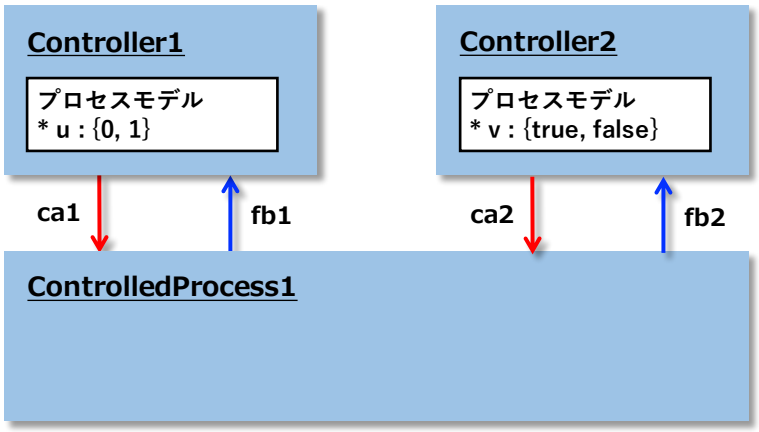

図 18　コントロールストラクチャインターフェースの例

本研究では，コンポーネントの振る舞いを入出力単位で分割して記述できるような形式的な定義を導入する．考え方の詳細については第～節で述べる．

定義 5. コンポーネント (component) は，組 $(C, S, \mathcal{A})$ で表される．ただし，C はコンポーネントインタフェース，S は値の範囲が有限の**状態変数** (state variables)，$(PM(C) \cup S, \mathcal{A})$ は以下を満たす共通の状態変数を持つ確率時間オートマトンの並行合成 $\mathcal{A} = A_1 \mid \cdots \mid A_n$ である．ただし，$n \geq 1$ に対して，$CA_{in}(C) \cup FB_{in}(C) \subseteq \bigcup_{1 \leq j \leq n} \Sigma_i^j$ かつ，$CA_{out}(C) \cup FB_{out}(C) \subseteq \bigcup_{1 \leq j \leq n} \Sigma_o^j$ である．$1 \leq j \leq n$ に対し，$\Sigma^j = \Sigma_i^j \uplus \Sigma_o^j$ となるような確率時間オートマトンを $A_j = (L^j, \ell_0^j, X^j, \Sigma^j, E^j, R^j, I^j)$ とする．

- $\Sigma_i^j \cap ICC(C) \subseteq CA_{in}(C)$ かつ，$\Sigma_o^j \cap ICC(C) = \emptyset$ の時，A_j を C_{in} 型オートマトンと呼ぶ．
- $\Sigma_o^j \cap ICC(C) \subseteq CA_{out}(C)$ かつ，$\Sigma_i^j \cap ICC(C) = \emptyset$ の時，A_j を C_{out} 型オートマトンと呼ぶ．
- $\Sigma_i^j \cap ICC(C) \subseteq FB_{in}(C)$ かつ，$\Sigma_o^j \cap ICC(C) = \emptyset$ の時，A_j を F_{in} 型オートマトンと呼ぶ．
- $\Sigma_o^j \cap ICC(C) \subseteq FB_{out}(C)$ かつ，$\Sigma_i^j \cap ICC(C) = \emptyset$ の時，A_j を F_{out} 型オートマトンと呼ぶ．

コンポーネントが上記の各構成要素のように確率時間オートマトンの並行合成から構成される場合，これを**シナリオ生成用コンポーネント** (scenario-generating component) と呼ぶ．本研究では，シナリオの生成を目的とするため STAMP/STPA におけるアクチュエータやセンサ，環境などを導入するが，これらもここで定義するようなコンポーネントの 1 つと考える．

ここで定義したシナリオ生成用コンポーネントが，上で述べた振る舞いを入出力単位で分割して記述できるようにしたコンポーネントの形式的な定義である．

上記で定義したコンポーネントは，STAMP/STPA におけるコントロールアルゴリズムを含んだコンポーネントに対応する．コンポーネント $\mathcal{C} = (C_0, S_0, \mathcal{A}_0)$ とすると，$\mathcal{A}(\mathcal{C})$ は $\mathcal{A}_0$，$C(\mathcal{C})$ は C_0 を表す．

シナリオ生成用コンポーネントを用いて，シナリオ生成用モデルを定義する．

定義 6. コントロールストラクチャ (control structure) $\mathcal{CS}$ は，以下を満たすものの有限集合である．

1. 集合 $\{C(\mathcal{C}) \mid \mathcal{C} \in \mathcal{CS}\}$ は，コントロールストラクチャインタフェース，
2. 集合 $\{A(\mathcal{C}) \mid \mathcal{C} \in \mathcal{CS}\}$ は，並行合成可能な確率時間オートマトンを構成する．

また，コントロールストラクチャがシナリオ生成用コンポーネントから構成される場合，これを**シナリオ生成用コントロールストラクチャ** (scenario-generating control structure)，または，**シナリオ生成用モデル** (scenario-generating model) と呼ぶ．

例 4. 図 19 にシナリオ生成用モデルの例を示す．ただし，$C = \{\mathsf{Controller1}, \mathsf{ControlledProcess1}\}$, $S = \{s\}, \mathcal{A} = (\{u, s\},\ \mathsf{STA1} \mid \mathsf{STA2} \mid \mathsf{STA3} \mid \mathsf{STA4})$ とする．ただし，$\{\mathsf{Controller1} = (\emptyset, \{ca1, ca2\}, \{fb1, fb2\}, \emptyset, \{u\}),\ \{\mathsf{ControlledProcess1} = (\{ca1, ca2\}, \emptyset, \emptyset, \{fb1, fb2\}, \emptyset), \mathsf{dom}(u) = \{0, 1\}$ である．

ここで，STA1 から STA4 は確率時間オートマトンであり，それぞれが並行合成されたものとなっている．図 19 では，例として STA2 に対応する確率時間オートマトンが描かれている．STA2 に現れる入力アクションは fb1 および fb2 なので，

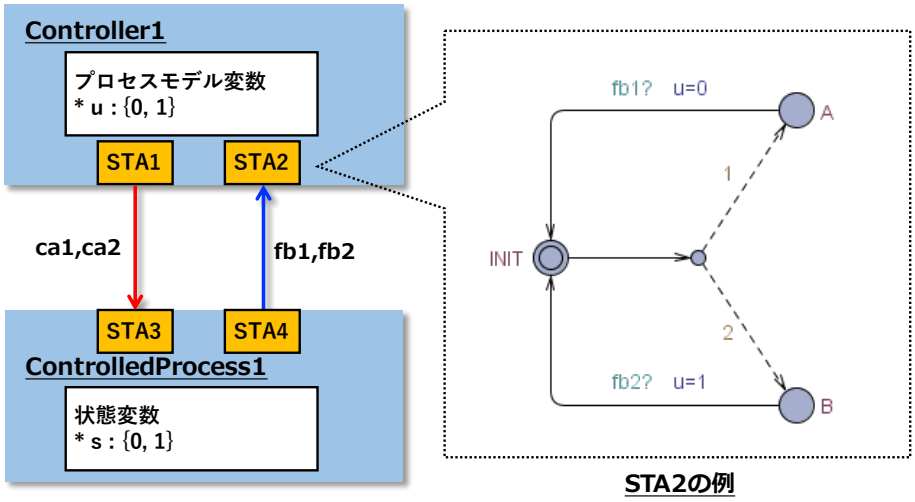

図 19　シナリオ生成用モデルの例

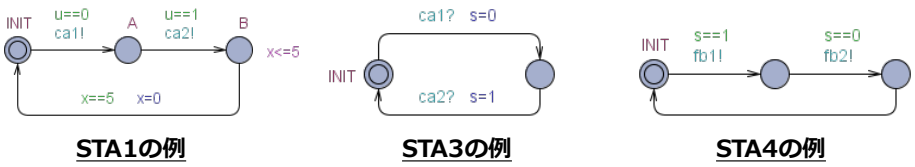

図 20　図 19 のシナリオ生成用モデルにおける各オートマトンの例

Controller1 の入力フィードバックの集合に含まれ，かつ出力アクションは存在しないため，F_{in} 型オートマトンである．図 20 に，STA1，STA3，STA4 の例を示す．

これらは STA2 と同じように STA1 は C_{out} 型オートマトン，STA3 は C_{in} 型オートマトン，STA4 は F_{out} 型オートマトンである．したがって，Controller1 と ControlledProcess1 はシナリオ生成用コンポーネントであることがわかり，全体としてシナリオ生成用コントロールストラクチャであることがわかる．

これまでに，シナリオ生成用モデルの形式的な定義について説明した．以降では，形式化に関する詳細な説明を行う．図 21 にシナリオを生成するためのモデルの概念図を示す．

モデルを作成するための方法は，第 3.6.1 節で述べたシナリオの要因を表現できることを目的として定義した．具体的には，コンポーネントの入出力，コントロールアルゴリズム，プロセスモデル等に関連するハザード要因である．これらの要因と形式モデルを対応させるため，コンポーネントを分割してモデル化する．また，コンポーネント間のコントロールアクションやフィードバックの経路におけるハザード要因も表現するため，アクチュエータやセンサ等も形式化を行う．以下に，各部分のモデル化に関して詳しく述べる．

5.3.2 コントロールアクションとフィードバックの形式化

コントロールアクションとフィードバックは UPPAAL SMC 上のチャネルとして表現する．コントロールアクションやフィードバックなどの作用は，チャネルを経由したメッセージによって実現される．また，コントロールアクションやフィードバックはブロードキャストチャネルで定義される．したがって，メッセージの送信側は，必ずしも受信側が受信可能ではなくてもメッセージを送信することが可能である．しかし，受信可能な場合は必ずメッセージを受信するものとなっている．また，ブロードキャストであるため，1 つのセンサから複数のコンポーネントへ同時にフィードバックを送信するといった振る舞いも表現することができる．

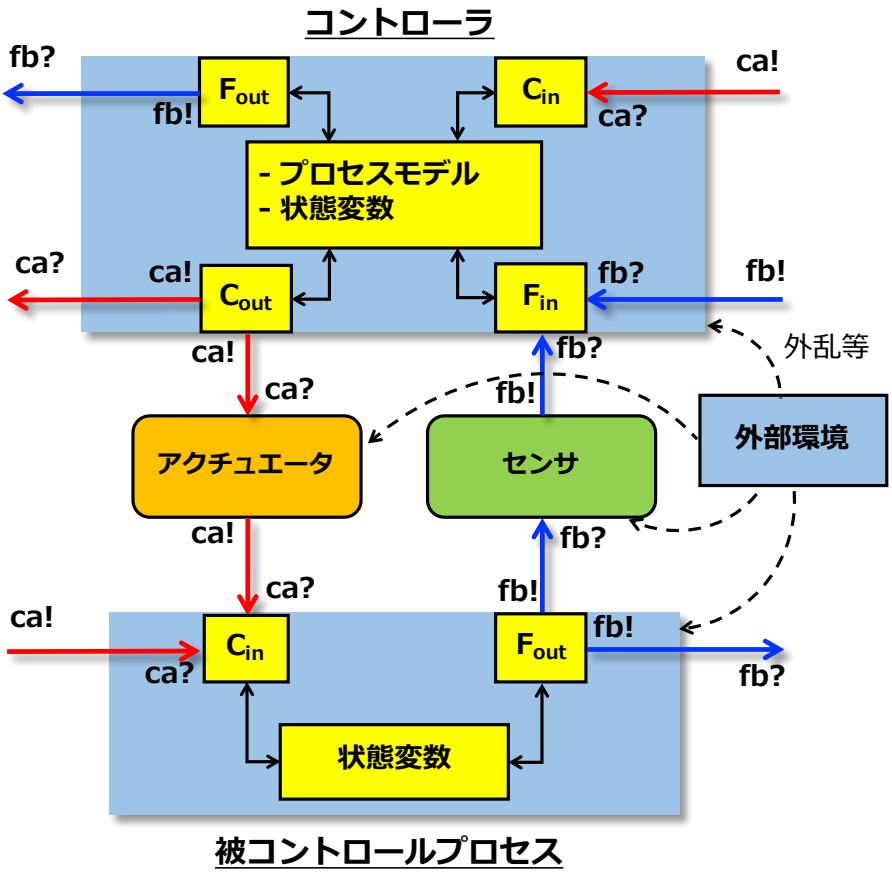

図 21　シナリオ生成用モデルの概念図

5.3.3 コントローラの形式化

STAMP/STPA におけるコンポーネントは，ある特定の機能単位でモデル化されたものが複数個存在するのが一般的である．モデル検査法では，最終的に対象となる形式モデルは 1 つの状態遷移系であり，その状態遷移系に対する探索を行うことによって検証を行う．しかし，STAMP/STPA で識別されたコントロールストラクチャやその相互作用，プロセスモデル等の変数を初めから 1 つの状態遷移系に変換することは難しい．一方で，Uppaal SMC では，オートマトンをテンプレートとして定義する．オートマトンを複数のテンプレートに分割して記述することで，複数の処理が並列して動作するような並行性を表現できる．本研究では，コンポーネントを入出力単位に分割して形式化することを検討した．これによって，1 つのコンポーネントが複数のコントロールアクションやフィードバックの送受信を同時に実行するような振る舞いも形式的に表現することができる．

コントローラには，他のコンポーネントに対する認識を表すプロセスモデルに対応するプロセスモデル変数と，コントローラの振る舞いを表すための状態変数が定義される．また，コントローラにおけるコントロールアクションの入出力は，入力部分である C_{in} と，出力部分である C_{out} となるオートマトンの集合にそれぞれ分割される．C_{in} は，他のコントローラからコントロールアクションを受信するためのオートマトンであり，コントロールアクションの受信動作に対する振る舞いが記述される．C_{out} は，コントロールアクションを他のコンポーネントに対して送信するためのオートマトンである．ここで，C_{out} には，コントロールアクションを送信するためのアルゴリズムであるコントロールアルゴリズムが表現される．このオートマトンは，状態遷移に伴ってチャネルを経由したメッセージ送信を実行することによって，コントロールアクションに相当するメッセージを出力することができる．このメッセージによって，他のコンポーネントはコントローラから制御が行われる．ただし，オートマトンの状態遷移は，コントローラが内部に持つプロセスモデル変数値や，コントローラが持つ状態変数などによって制御される．

続いて，フィードバックの入出力について述べる．フィードバックの入出力は，入力部分である F_{in} と，出力部分である F_{out} となるオートマトンの集合にそれぞ

れ分割される．F_{in} では，他のコンポーネントからのフィードバックを受信するためのオートマトンであり，フィードバックに対応するチャネルからのメッセージ受信動作が定義される．ここでは，フィードバックの受信に対する解釈および，プロセスモデルへの反映に対する振る舞いが記述される．F_{out} は，コントローラの持つ状態変数の値に応じて，他のコンポーネントにチャネルを経由してフィードバックを送信するためのオートマトンである．ここでは，コントローラが保持する状態に応じて，他のコンポーネントにフィードバックを送信するための振る舞いが記述される．

5.3.4 被コントロールプロセスの形式化

被コントロールプロセスは，コントローラからコントロールアルゴリズムやプロセスモデル，コントローアクションの出力部分，フィードバックの入力部分がないものとして考えることができる．したがって，被コントロールプロセスの入出力には，コントロールアクションを受信するためのオートマトンの集合 C_{in} と，フィードバックを送信するためのオートマトンの集合 F_{out} の 2 種類のみが存在する．また，被コントロールプロセスの振る舞いを表すための状態変数が定義される．C_{in} には，コントローラにおける定義と同様に，他のコントローラからコントロールアクションを受信に関する振る舞いが定義され，振る舞いを表す状態変数に作用することができる．また，F_{out} には，被コントロールプロセスが他のコンポーネントに対してフィードバックを送信するための振る舞いが表現される．

5.3.5 アクチュエータとセンサの形式化

アクチュエータとセンサについても形式的な定義の上では，コンポーネントの特殊な場合として扱う．

コントローラが出力したコントロールアクションはアクチュエータを経由して被コントロールプロセスに作用する．また，被コントロールプロセスから出力されるフィードバックはセンサを経由してコントローラに入力される．ここでは，アクチュエータやセンサをそれぞれオートマトンの集合として定義する．先に述

べたコンポーネントの形式化と同様に，これらのオートマトンには，アクチュエータやセンサにおける情報伝達に対する振る舞いが記述される．アクチュエータやセンサは，基本的にコントローアクションやフィードバックの中継を行う存在であるため，入出力単位で分割をせずにオートマトンを記述して簡易的に表現する．

5.3.6 外部環境の形式化

外部環境も形式的な定義の上では，コンポーネントの特殊な場合として扱う．

外部環境に含まれるオートマトンには，コントローラや被コントロールプロセス，アクチュエータやセンサに対して作用するための振る舞いが表現される．具体的には，ハザード要因として識別された外乱等の発生に対する振る舞いが記述される．図 21 における黒色の点線矢印は，外部環境によるハザード要因がコンポーネントの状態変数等を経由して擬似的に作用する概念を表している．

5.3.7 ハザードの形式モデルへの反映

ここでは第 3.6 節で述べた，ハザードに至るシナリオの要因に対する分類を形式モデルで表現する方法について述べる．ここでは，**(a-1)**，**(a-2)**，**(b-1)**，**(b-2)** の分類について，それぞれ順に説明する．

非安全なコントローラの動作

表 2 に，第 3.6 節の **(a-1)** である「非安全なコントローラの動作」に対する各ハザード要因の分類をモデルに反映させるための方針を示す．

まず，非安全なコントローラの動作に繋がる原因として，コントローラの故障がある．こうした振る舞いは，コンポーネントに含まれるオートマトンの状態変数を用いて，オートマトンが状態遷移できなくなることを利用して擬似的に表現できる．図 22 に故障を形式的なモデルで表現する場合の概念図を示す．

図 22 において，丸がオートマトンの状態であり，それらを接続する矢印は状態の遷移を表現する辺である．また，辺に示された遷移条件 Guard の右側は，状態遷移が発生するための条件式であり，これが満たされて真である場合，矢印の方向

表 2 非安全なコントローラの動作に対するハザード要因のモデル化の方針

ハザード要因	変更対象	変更するモデル	変更方針
(a-1-1)	コントローラ，環境	C_{in}, C_{out}, F_{in}, F_{out}	状態変数を各オートマトンの遷移条件として定義して故障を表現し，環境に対応するオートマトンによって変数の値を制御する
(a-1-2-1)	コントローラ	C_{out}	出力側のオートマトンの振る舞いを変更することで誤ったコントロールアルゴリズムの実装を表現する
(a-1-2-2)	コントローラ	C_{out}	出力側のオートマトンの振る舞いを変更することで誤ったコントロールアルゴリズム仕様を表現する
(a-1-2-3)	コントローラ	C_{out}	出力側のオートマトンの振る舞いを変更することでコントロールアルゴリズムの劣化を表現する
(a-1-3)	コントローラ	C_{in}	入力に対応するオートマトンの振る舞いを変更して非安全な入力がされたことの振る舞いを表現する
(a-1-4-1)	コントローラ，被コントロールプロセス	F_{out}	フィードバックの送信元に対応するオートマトンの振る舞いを変更することで誤ったフィードバックの送信を表現する
(a-1-4-2)	コントローラ	F_{in}	フィードバックの受信側に対応するオートマトンの振る舞いや，プロセスモデル変数の定義や更新方法を変更して表現する
(a-1-4-3)	コントローラ	F_{in}	フィードバックの受信側に対応するオートマトンの振る舞いを変更することで受信しない，あるいは遅延する状態を表現する
(a-1-4-4)	コントローラ，被コントロールプロセス	F_{out}	フィードバックの送信元に対応するオートマトンの振る舞いを変更することでフィードバックの欠落を表現する

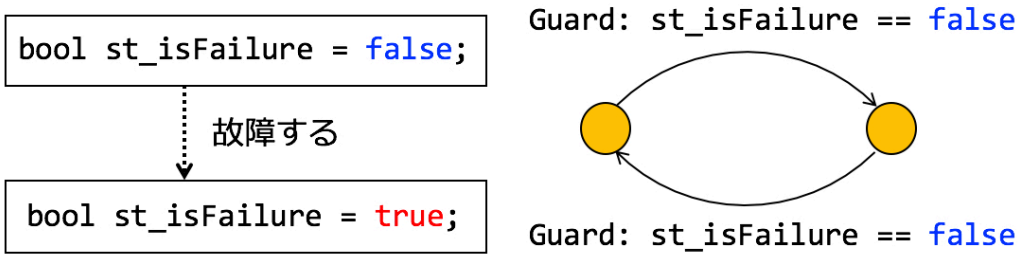

図 22 故障に対する形式的な表現

へ状態遷移可能となる．変数 st_isFailure は，故障の有無を表す状態変数であり，右側のオートマトンは変数の値が false の間動作を続けることができる．オートマトンをこのように表現しておくことで，擬似的に故障状態の有無をモデル化することができる．例えばハザード要因として外乱による故障を考える場合には，コンポーネントに含まれる各オートマトンにこうした状態変数を追加し，外部環境に対応するオートマトンが変数の値を特定のタイミングで変数 st_isFailure の値を true に変更することで表現できる．外部環境によって状態変数を変更するタイミングは，外部環境が持つオートマトンの状態遷移に確率値を与えることによって制御することができる．

不適切なコントロールアルゴリズムは，コントローラの出力側オートマトンにおける状態間の遷移や，その時間制約の欠陥によって表現される．図 23 にアルゴリズムの欠陥に対する表現例を示す．

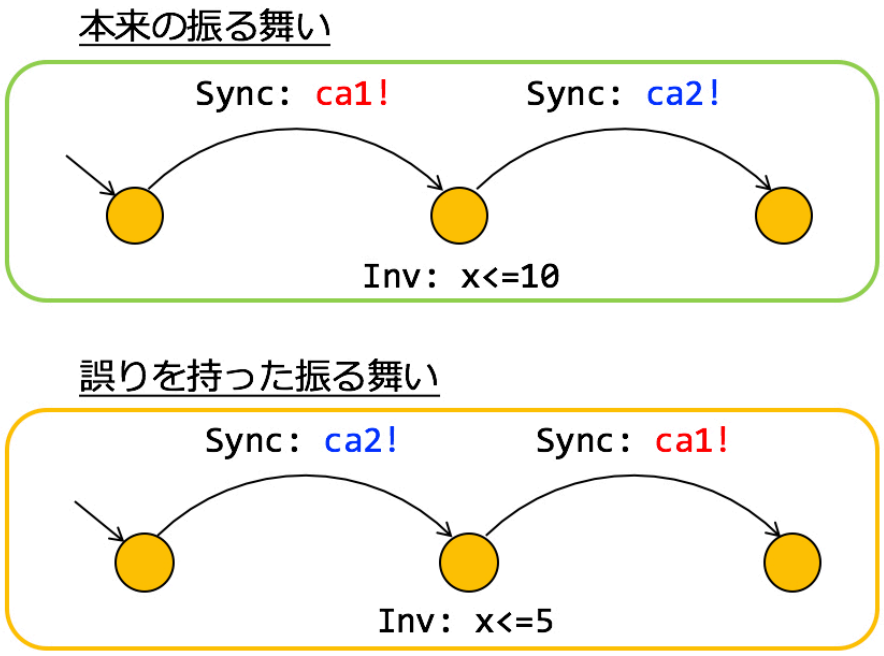

図 23　不適切なコントロールアルゴリズムの表現例

図 23 における同期チャネル Sync の右側はメッセージを送信するためのチャネルを表し，不変条件 Inv の右側は時間制約を表している．ここでは，本来であれ

ばコントロールアクション ca1 が出力された後に ca2 が出力されるはずであるが，実際は ca2 の後に ca1 が出力されるような欠陥を持ったアルゴリズムであることを表している．また，中央に位置する状態に対する時間制約も異なっている．このように置くことで，コントロールアルゴリズムの欠陥を表すことができる．ただし，分類の **(a-1-2-1)** である「コントロールアルゴリズムの誤った実装」と，**(a-1-2-2)** に対応する「コントロールアルゴリズムの仕様に対する欠陥」を明確に区別して表現することはできない．その理由として，アルゴリズムの実装に関する作業工程等をオートマトンでモデル化することは難しいと考えられることなどが挙げられる．しかし，これまでに説明した形式化の方法によって，分析を行う上で代表的なハザード要因を表すことは可能であると考えられる．

続いて，**(a-1-3)** である「非安全なコントロール入力」に対する表現は，コントロールアクションを出力するコントローラの出力部分に対応するオートマトンや，それが入力される入力側のオートマトンの状態遷移や時間制約を用いて表現することができる．

最後に，**(a-1-4)** の「不適切なプロセスモデル」に対するハザードの表現は，先に述べたプロセスモデルに対応する変数の定義によって表現される．不適切なプロセスモデルの主な原因として，コントローラが受信するフィードバックの内容に関する誤りや，そもそもフィードバックが存在しないといった問題がある．こうした原因は，コントローラのフィードバック受信用オートマトンの振る舞いによって表現，記述することが可能であると考えられる．

不適切なフィードバックと情報の原因

次に，**(a-2)** の分類である「不適切なフィードバックと情報の原因」に対する表現について検討する．表 3 に，各ハザード要因の分類をモデルに反映させるための方針を示す．

この分類は，**(a-2-1)** の「フィードバックや情報を受信しない」および，**(a-2-2)** の「不適切なフィードバックを受信する」に分解される．**(a-2-1)** は，基本的にセンサや，コントローラがフィードバックを正しく受信しないことを原因としたハザード要因に関する分類となっている．形式的なシナリオ生成用モデルでは，センサを含めて形式化しているため，センサの遅延や故障等の振る舞いは，これまで

表 3 不適切なフィードバックと情報の原因に対するハザード要因のモデル化の方針

ハザード要因	変更対象	変更するモデル	変更方針
(a-2-1-1)	コントローラ	F_{in}	フィードバックの受信側に対応するオートマトンの振る舞いを変更することで，フィードバックを受信できないことを表現する
(a-2-1-2)	コントローラ, センサ	F_{in}, センサ	フィードバックの受信側やセンサの振る舞いに対応するオートマトンを変更してフィードバックを受信したかのような振る舞いを表現する
(a-2-1-3)	コントローラ, センサ	F_{in}, センサ	フィードバックの受信側やセンサのオートマトンを変更することでフィードバックを受信できない振る舞いを表現する
(a-2-1-4)	フィードバック, センサ	フィードバックチャネル, センサ	フィードバックチャネルの定義やセンサに対応するオートマトンの定義を変更することでフィードバックやセンサの欠落を表現する
(a-2-2-1)	被コントロールプロセス	F_{out}	フィードバックの送信側に対応するオートマトンの振る舞いを変更することで不適切なフィードバックの送信を表現する
(a-2-2-2)	センサ	センサ	センサに対応するオートマトンを変更することでセンサの不適切な振る舞いを表現する
(a-2-2-3)	センサ	センサ	センサに対応するオートマトンを変更することで不十分な性能に対する振る舞いを表現する

に述べた時間制約や状態変数を用いて表現することが可能である．また，**(a-2-2)** も，**(a-2-1)** で述べた方法と同様に表現できる．

コントロール経路を含むシナリオ

続いて，**(b-1)** である「コントロール経路に関連するシナリオ」の表現について検討する．表 4 に，各ハザード要因の分類をモデルに反映させるための方針を示す．

表 4　コントロール経路に関連するシナリオに対するハザード要因のモデル化の方針

ハザード要因	変更対象	変更するモデル	変更方針
(b-1-1-1)	アクチュエータ	アクチュエータ	アクチュエータに対応するオートマトンがコントロールアクションを受信しないことを表現する
(b-1-1-2)	アクチュエータ	アクチュエータ	アクチュエータに対応するオートマトンを変更することでコントロールアクションを受信しても応答しない振る舞いを表現する
(b-1-1-3)	被コントロールプロセス	C_{in}	コントロールアクションの受信側に対応するオートマトンを変更することでコントロールアクションが受信されない振る舞いを表現する
(b-1-2-1)	アクチュエータ	アクチュエータ	アクチュエータに対応するオートマトンを変更することでコントロールアクションが不適切に受信される振る舞いを表現する
(b-1-2-2)	アクチュエータ	アクチュエータ	アクチュエータに対応するオートマトンを変更することで不適切に応答することの振る舞いを表現する
(b-1-2-3)	被コントロールプロセス	C_{in}	コントロールアクションの受信側に対応するオートマトンを変更することでコントロールアクションが不適切に受信される振る舞いを表現する
(b-1-2-4)	アクチュエータ, 被コントロールプロセス	アクチュエータ, C_{in}	アクチュエータやコントロールアクションの受信側に対応するオートマトンを変更することでコントロールアクションを受信したかのような振る舞いを表現する

まず，**(b-1-1)** は，基本的にはコントロール経路に含まれるアクチュエータの不適切な動作を原因としたシナリオに関する分類である．アクチュエータはシナリオ生成用モデルにオートマトンとして定義される．アクチュエータがコントロールアクションを受信しない，応答しないといった振る舞いは，オートマトンの状態遷移やチャネルの記述，時間制約等で表現することができる．また，コントロー

ルアクションに対して，アクチュエータが応答するが，コントロール対象プロセスが受信しないといった振る舞いは，被コントロールプロセスの受信側オートマトンの振る舞いによって表現される．

また，**(b-1-2)** の「コントロールアクションが不適切に実行される」ことに対する原因は，**(b-1-1)** と同様にアクチュエータに関連する分類となっている．**(b-1-1)** では，アクチュエータや被コントロールプロセスが応答しないといった内容に対して，**(b-2-2)** では，それらの誤動作も含まれた分類である．誤動作についても，各オートマトンの振る舞いを変更したり，外部環境やアクチュエータに対応するオートマトンの振る舞いを組み合わせることによって表現することができる．

コントロール対象のプロセスに関連するシナリオ

最後に，**(b-2)** である「コントロール対象のプロセスに関連するシナリオ」に対する表現について検討する．表 5 に，各ハザード要因の分類をモデルに反映させるための方針を示す．

表 5　コントロール対象のプロセスに関連するシナリオに対するハザード要因のモデル化の方針

ハザード要因	変更対象	変更するモデル	変更方針
(b-2-1-1)	被コントロールプロセス	F_{out}	フィードバックの送信側に対応するオートマトンを変更することで被コントロールプロセスが応答しない振る舞いを表現する
(b-2-2-1)	被コントロールプロセス	F_{out}	フィードバックの送信元に対応するオートマトンを変更することで被コントロールプロセスが不適切に応答する振る舞いを表現する
(b-2-2-2)	被コントロールプロセス	F_{in}, F_{out}	フィードバックの受信側と送信側に対応するオートマトンを変更することでフィードバックが受信されない状態と，まるで応答したかのような振る舞いを表現する

(b-2) は，**(b-2-1)** である「コントロールアクションが実行されない」および，**(b-2-2)** の「コントロールアクションが不適切に実行される」に分類される．ここで，**(b-2-1)** の原因では，コントロールアクションが正しく被コントロールプロセスによって受信されるが応答しないことが述べられている．こうした原因は，

被コントロールプロセスのコントロールアクション入力用オートマトンが正しく振る舞う一方で，フィードバックを出力するオートマトンが正しく動作しないことによる原因として表現できる．こうした原因が外乱等によるものであれば，外部環境に対応するオートマトンの振る舞いを組み合わせることによって，それらを表現することが可能である．

(b-2-2) では，**(b-2-1)** と同様にコントロールアクションが被コントロールプロセスによって適切に受信されるが，想定した振る舞いをしないことに対する分類である．こうした誤動作は，**(b-2-1)** の場合と同様に，被コントロールプロセスのフィードバック出力部分に対応するオートマトンの振る舞いで記述できる．外乱等の影響による場合も同様である．

5.4 手順 4：シナリオ観測用モデルの記述

次に，ハザードに至るシナリオを観測するためのモデルに対する記述を行う．シナリオを観測するモデルは UPPAAL SMC におけるテンプレートとして定義しておく必要がある．こうしたテンプレートを定義しておくことにより，形式的なシナリオ生成モデル上でハザードに至るシナリオが発生した場合に，それを検出する役割を果たす．ここでは，こうしたオートマトンのテンプレートを定義する流れについて述べる．一般的に，STAMP/STPA で識別されたシナリオは自然言語で記述されるため，それらをオートマトンに変換する必要がある．ハザードに至るシナリオはイベントの列として見なすことができる．また，イベントの発生は状態の遷移として考えることができる．ここでは，以下のような抽象的なシナリオの例を考える．

> イベント E_1 が発生し，その後，イベント E_2 が起きて，最終的にイベント E_3 によってハザードに至る．

STAMP/STPA における自然言語によるシナリオ記述に規則はない．そのため，イベント列における事象に対応する要素を明確化するため，ここでは表を用いて記述することにする．表 6 に上記のシナリオ例に対するイベント列を示す．

表 6　シナリオ例に対するイベント列

番号＼対応要素	F_1	F_2	F_3
1	事象E_1が発生		
2		事象E_2が発生	
3			事象E_3が発生

ハザードの状態に到達

表 6 における対応要素とは，シナリオに関連するコンポーネント，センサ，アクチュエータ，外部環境等である．また，番号ではイベント列の順番を表している．表では，例えば，イベント E_1 は機能 F_1 における事象であることを表している．こうしたイベント列に対して，第 5.3 節でモデル化したオートマトンとの対応関係を考える．各イベントは表 7 に示すように，オートマトンにおける状態遷移と対応させることが可能である．

表 7　イベント列に対応するオートマトンの状態遷移

番号＼対応要素	F_1	F_2	F_3
1	状態遷移T_1		
2		状態遷移T_2	
3			状態遷移T_3

例えば，シナリオにおける事象 E_1 は状態遷移 T_1 に対応する．こうした状態遷移は，コントロールアクションやフィードバックの送信のようにチャネルを経由したメッセージ送信や，コンポーネントが持つ状態変数の変化を利用して観測することができる．この表を利用して検討されたシナリオ観測用モデルの例を図 24 に示す．

図 24 のシナリオ観測用モデルは，一番左側の状態から遷移が開始され，一番右

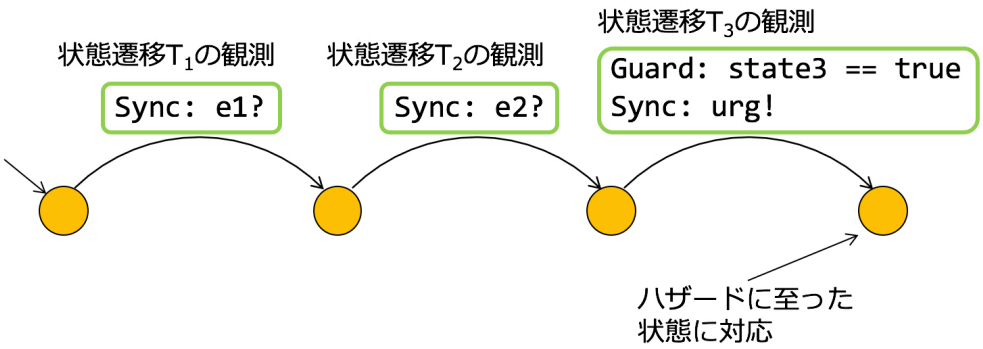

図 24　シナリオ観測用モデルの例

の状態に到達したとき，システムがハザードの状態であることを意味する．まず，状態遷移 T_1 の観測が e1 というチャネルにおけるメッセージの受信によって行われる．e1?は，e1 というチャネルにおけるメッセージの受信動作を意味する．同様に，この例では状態遷移 T_2 がメッセージ受信 e2?によって対応付けられている．状態遷移 T_3 は，メッセージ受信による観測ではなく，コンポーネント等が持つ状態変数の観測によって行われる．図の例では，st_state3 という状態変数が true であることの観測によって状態遷移するものとなっている．ここで，同期 urg!はアージェントブロードキャストチャネルのダミーである．ここでは，st_state3 という状態変数が true となった瞬間にハザードに到達していることを表現するために，アージェントブロードキャストチャネルによるダミーを定義している．こうしたオートマトンをモデル化した後，手順 3 で得たモデルと合成し，次の手順でハザードに至るための確率を求めるための検証を行う．

5.5　手順 5：シナリオに対する形式検証

次に，手順 3 と手順 4 で得られた形式的モデルに対してシナリオの発生に関する検証を行う．ここでは，ハザードの発生確率に関する定量的に検証を行う．図 25 に手順 3 と手順 4 で得られるシナリオ生成用モデルとシナリオ観測用モデルの例を示す．図 25 の例では，シナリオ生成用モデルとして 2 つのオートマトンテンプレート T1 と T2 が存在する．また，シナリオ観測用モデルのテンプレートと

して T3 がある．UPPAAL SMC では，こうした各テンプレートを合成した状態遷移系に対して論理式を与えて検証を行う．

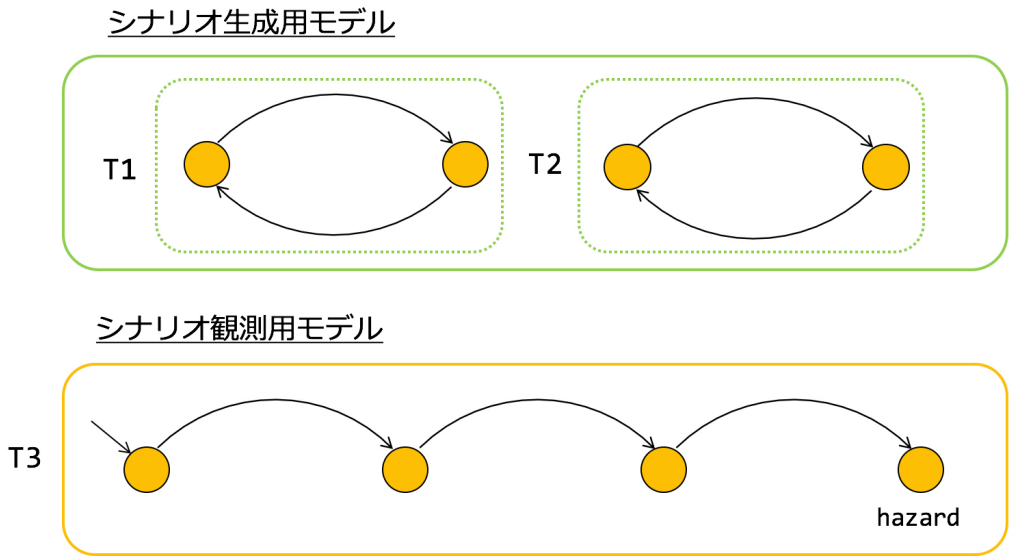

図 25　テンプレートの集合の例

シナリオ観測用モデルのテンプレート T3 において，ハザードの状態に対応する状態名を hazard とする．この場合，ハザードの発生確率は，例えば以下の論理式で検証することができる．

Pr[<=100] (<> T3.hazard)

ここで，Pr は確率を計算するときに用いるクエリである．<>は時相演算子であり，「将来どこかの時点で」を意味する．また，Pr[<=100] は，100 単位時間内で論理式の性質を検証するという意味を持つ．すなわち，上記の論理式は「100 単位時間内にテンプレート T3 における状態 hazard に到達する確率値」を計算する．また，UPPAAL SMC では，確率値を求めた後に，性質が満たされるまでのトレース (状態遷移列) を出力することができるため，モデル上でシナリオが成立するまでの具体的な振る舞いを確認することができる．

5.6 手順6：シナリオに対するリスク分析

最後に，手順5で得られたシナリオに対する確率値をもとに，リスクを分析する．リスクの値はハザードの発生確率とその深刻度によって以下のように求めることができる [4].

$$R = P \times I$$

ここで，R はリスク値であり，P はハザードの発生確率，I はその深刻度である．この深刻度は，ハザードによって発生する可能性のある損失の深刻度をもとに検討する必要がある．すなわち，手順1で識別した損失の深刻度である．その大きさは，システムの利害関係者の基準によって独自に決定される．その基準は，その組織が何を重視するかに依存するため，その利害関係者間で妥当な値を検討して合意する必要がある．ここで得られたハザードのリスク値および，先に検討していた対策の基準となるリスク値と比較を行い，どのハザードから優先的に対処するべきかが決定される．

6. 事例による提案手法の評価

この章では，第 5 章で述べた提案手法をもとに，鉄道における踏切制御システムを対象として手法の評価を行う．具体的には，提案手法によって，STAMP/STPA で識別されたシナリオが形式モデルで再現されること，および，得られた形式モデルによってリスクが分析できることを確認する．まず，第 6.1 節で対象システムの概要について述べ，その後，提案手法の各手順を実施した結果について説明する．最後に，第 6 節で全体としての評価について述べる．

6.1 踏切制御システムの概要

ここでは，踏切制御システムに対する STAMP/STPA の適用報告 [29] を参考に，例題として用いる踏切制御システムの概要について述べる．まず，図 26 に，想定する踏切制御システムの構成を示す．

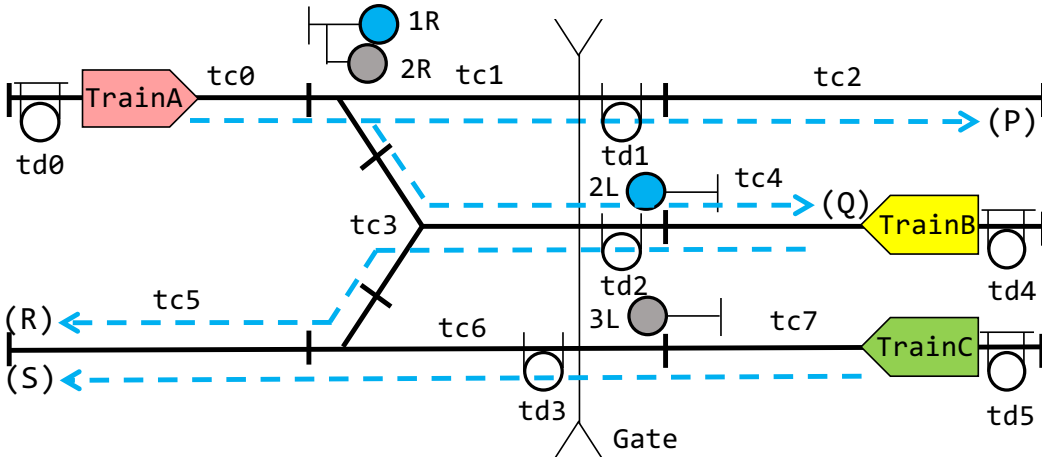

図 26　想定する踏切制御システムの構成

図 26 では，直線は線路を表し，区切られた区間は軌道回路を表している．td0 から td5 で示された記号は，踏切制御子を表す．また，図中央の縦棒 Gate は踏切を示している．さらに，青色や灰色で示された丸は信号機を示している．以下では，システムの各構成要素について説明する．

軌道回路　軌道回路とは，線路上の特定区間に列車が存在するか否かを検知する電気的な装置である．図中の tc0 から tc7 は軌道回路を示している．軌道回路によって得られた在線情報は，信号を制御するための連動装置や，踏切を制御するための踏切制御装置に伝達される．各進路の始端には信号機が存在し，その進路上の軌道回路に列車が在線している場合は赤，在線していない場合は青を表示する．これにより，同一進路上に複数の列車が在線することを防いでいる．

踏切制御子　踏切制御子とは，踏切の警報機や遮断機を動作させるために、踏切に列車が接近したことを検知する装置である．踏切制御子は始動点と終止点がある．始動点とは，列車が踏切に接近していることを検知するための装置である．また，終止点とは，列車が踏切を通過したことを検知するための装置であり，警報を停止するために用いられる．

列車　ここでは，列車の種類がその進路に応じて TrainA，TrainB，TrainC の 3 つある．TrainA は図中の矢印で示された進路の中で，進路 (P) と (Q) に対応する進路を取ることができる．同様にして，TrainB は進路 (R)，TrainC は進路 (S) に向かって進行することができる．各列車の進路は，連動装置によって信号の切り替えが行われることによって決定される．

連動装置　分岐のある駅構内には連動装置と呼ばれる装置があり，これによって各信号の制御が行われる．1R，2R，2L，3L は各進路を表示するための信号であり，図中の矢印で示された P，Q，R，S の進路にそれぞれ対応する．連動装置は軌道回路から得た在線情報をもとに進路の切り替えを行う．連動装置は，進路制御装置からの進路構成指示を受けて, 軌道回路から得た在線情報をもとに進路の切り替えを行い信号機を制御する．ただし，進路制御装置などの詳細は文献 [30, 32] の説明に従うものとする．

踏切制御装置　踏切制御装置は，警報機や遮断器を制御して踏切に出入りする人や自動車などを制御するための装置である．図 26 の中央に存在する縦棒が踏切を表している．踏切制御装置は，踏切制御子や軌道回路などの情報をもとに列車

の進行方向などを判断して制御を行う．また，踏切に異常があると判断した場合は，安全のために遮断器を降ろす機能を持っている．

6.2 提案手法による分析

ここでは，第 6.1 節で述べた踏切制御システムに対して提案手法を適用した結果について述べる．本研究では，STAMP/STPA による安全分析や SysML による振る舞い記述が同時に行える分析ツールである astah System Safety[3]を用いて，ハザード分析および振る舞いの記述を行った．

6.2.1 STAMP/STPA による分析例 (手順 1)

まず，手順 1 では，対象システムにおける基本的な前提条件，損失，システムレベルのハザード，安全制約を識別する．ここでの前提条件は，第 6.1 節で説明した通りである．また，損失および，システムレベルのハザードと安全制約の識別結果を表 8 に示す．

表 8 識別された損失，ハザード，安全制約

損失	ハザード	安全制約
(L1) 列車と人 (車) が衝突する	(H1) 列車が踏切に接近中に警報の鳴動が開始されない	(S1) 列車が踏切に接近したら警報の鳴動が開始されなければならない
(L1) 列車と人 (車) が衝突する	(H2) 列車が踏切を通過中に警報の鳴動が停止される	(S2) 列車が踏切を通過中は警報の鳴動が停止してはならない
(L2) 交通渋滞が発生する	(H3) 列車が踏切に接近していないのに警報が鳴動を開始する	(S3) 列車が接近していない場合は警報が鳴動を開始してはならない
(L2) 交通渋滞が発生する	(H4) 列車が踏切を通過したのに警報が鳴動を続ける	(S4) 列車が踏切を通過したら警報は鳴動を停止しなければならない

この例では，(L1) のような人命に関わるものと，損失 (L2) のようなシステム運用上の可用性に関わる損失について定義している．次に，図 27 に今回対象とするシステムのコントロールストラクチャを示す．

[3]http://astah.change-vision.com/ja/product/astah-system-safety.html

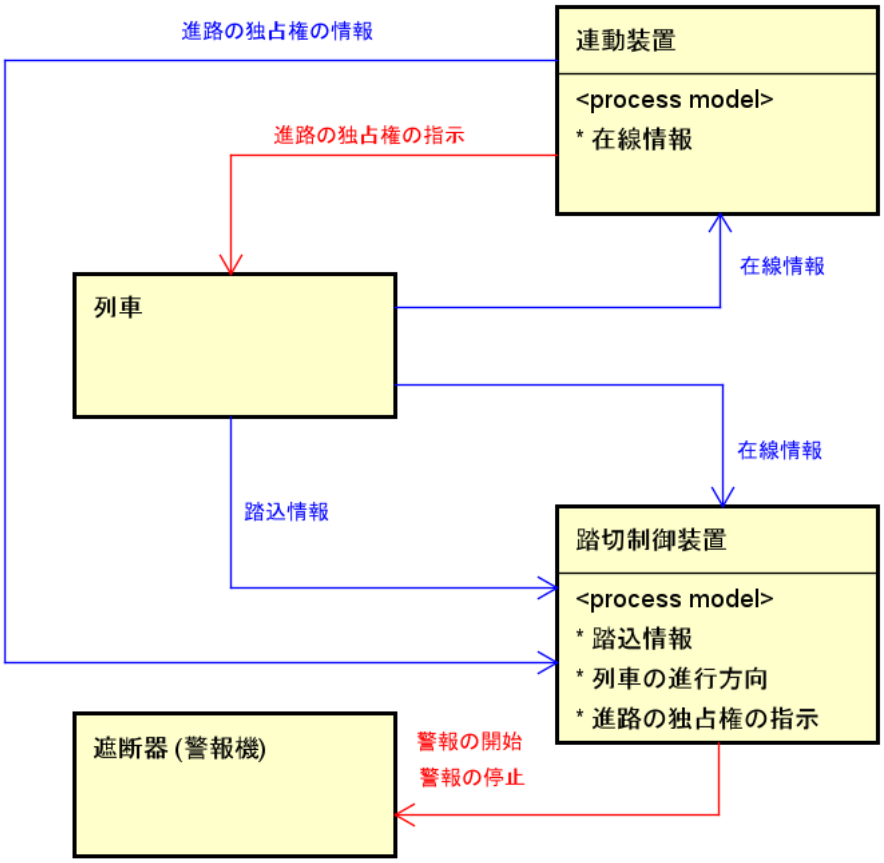

図 27 想定する踏切制御システムのコントロールストラクチャ

ここで，各コンポーネントが持つ基本的な役割は第 6.1 節で述べた通りである．また，連動装置と踏切制御装置はコントローラであり，列車と遮断器 (警報機) は被コントロールプロセスである．連動装置は列車からの在線情報をプロセスモデルとして保持している．また，連動装置は在線情報をもとに進路を設定し，列車に指示する．踏切制御装置は，列車からの踏込情報をもとに，列車が踏切に進入したか否かに関する情報と，在線情報から得た列車の進行方向に関する情報をプロセスモデルとして持っている．また，連動装置から受け取る進路の独占権の情報によって，進路設定の情報が共有されている．こうした情報をもとに，警報機に対して警報の開始や停止の指示を出すものとなっている．　次に，表 9 に非安全なコントロールアクションを分析した UCA 表の例を示す．ただし，ここでは本研究で対象とする「警報の開始」と「警報の停止」に関するコントロールアクションに限定した結果について示す．

表 9　非安全なコントロールアクションの分析結果

CA	From	To	与えられないとハザード	与えられるとハザード	早過ぎ，遅過ぎ，誤順序	早すぎる停止，長すぎる適用
警報の開始	踏切制御装置	遮断器 (警報機)	(U1) 警報が鳴動せず列車が踏切を通過する (S1)	(U2) 列車がいない状態でも警報が鳴動を開始する (S3)	(U3) 警報の鳴動が開始する前あるいは鳴動直後に列車が踏切を通過する (S1)	(U4) 鳴動開始指示が継続し，停止指示が出ても鳴動が継続する (S4)
警報の停止	踏切制御装置	遮断器 (警報機)	(U5) 列車が踏切通過後も警報が鳴動を続ける (S4)	(U6) 列車が踏切を通過中に警報の鳴動が停止する (S2)	(U7) 列車が踏切を通過する前に警報の鳴動が停止する (S1)	(U8) 鳴動停止指示が継続し，次の列車が接近しても鳴動開始しない (S1)

表に含まれている (U1) から (U8) は，非安全なコントロールアクションの識別子である．また，(S1) から (S4) はそれぞれの非安全なコントロールアクションによって違反される表 8 における安全制約の識別子である．これら 2 種類のコントロールアクションに対してそれぞれコントロールループが識別される．識別されたコントロールループを図 28 に示す．ただし，図 28 では，2 つの非安全なコントロールアクションがまとめて表現されている．

次に，識別されたコントロールループに対してハザードに至るシナリオを考える．シナリオは，(U1) から (U8) までの全てに対して網羅的に分析されるが，ここでは北村の報告 [29] を参考に，(U1)，(U2)，(U6)，(U7) に対応するシナリオ

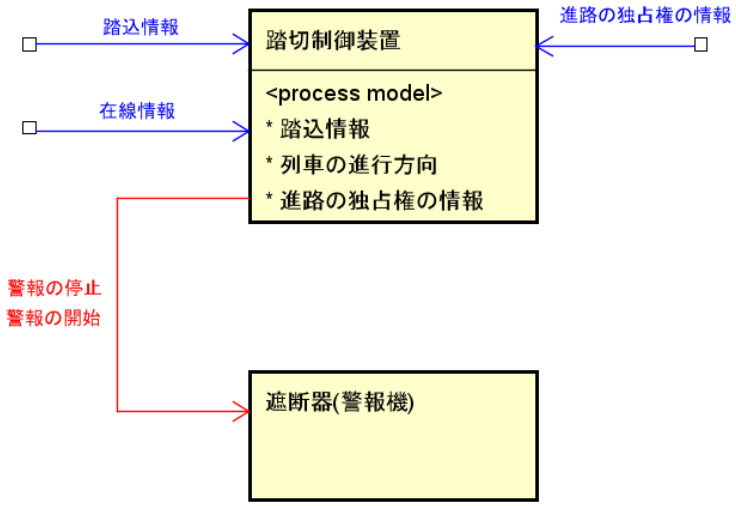

図 28　識別されたコントロールループ

を考える．以下に，各コントロールループと対応する各シナリオおよび，そのハザード要因について説明する．

(HS1) 信号 (1R,2R,2L,3L)=(青, 赤, 青, 赤) とする．TrainA が始動点 td0 に進入し警報開始する．その後，TrainB が tc3 に進入するが，踏切制御装置が 15 秒以内に TrainA が tc3 に進入したと誤判断．異常検知を行い警報の鳴動が継続する．このシナリオは，表 9 の非安全なコントロールアクション (U2) に対応する．ハザードの要因は，踏切制御装置が列車の進路 (1R と 2R) を区別していないという，踏切制御装置におけるプロセスモデルの欠陥である．

(HS2) 信号 (1R,2R,2L,3L)=(赤, 青, 赤, 青) とする．TrainA が始動点 td0 を通過後に tc3 に進入するが，その後に TrainC が tc6 に進入したことで，踏切制御装置は TrainA がバックして tc3 から tc6 に進行したと誤判断．踏切に接近する列車がいないと判断して警報の鳴動が停止し，無遮断状態で列車が踏切を通過する．このシナリオは，非安全なコントロールアクション (U6) に対応する．ハザードの要因は，踏切制御装置が列車の台数に対する認識を持っていなかったことによるプロセスモデルの欠陥である．

(HS3) 列車が始動点に進入し警報開始した後，終止点に到達する前に後続の列車が進入する．このシナリオは，非安全なコントロールアクション (U7) に対応する．ハザードの要因は，列車が適切な距離を保てなかったことによる，被コントロールプロセスの不適切な応答である．

(HS4) 列車が踏切に接近しても踏切制御子が列車を検知せずに警報が鳴動しないため，無遮断の状態で列車が踏切を通過する．このシナリオは，非安全なコントロールアクション (U1) に対応する．ハザードの要因は，落雷や落ち葉が軌道回路上に堆積することによる装置の不具合である．

6.2.2 振る舞いの識別と合意の例 (手順 2)

図 29 に，SysML によって記述したブロック定義図を示す．

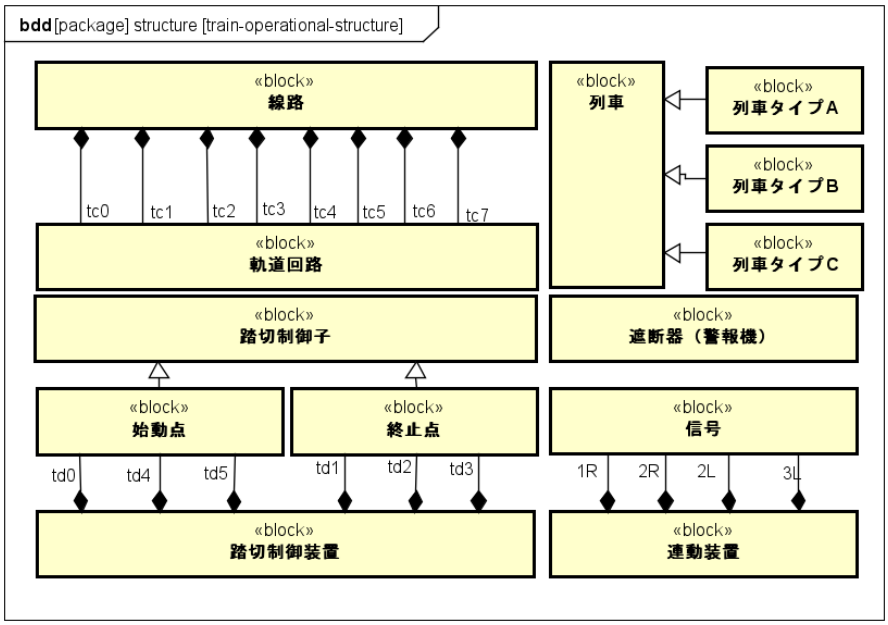

図 29 システム全体の構成要素に関するブロック定義図

図 29 では，コントロールストラクチャに含まれるコンポーネントおよび，コントロールループに含まれるアクチュエータやセンサを含めたシステムの構造が記

述されている．例えば，列車は 3 種類の列車との汎化の関係を持つ．また，信号機は連動装置とコンポジションの関係を持つ．次に，振る舞いの記述例として，図 30 に連動装置の信号制御に関する振る舞いアクティビティ図によって示す．

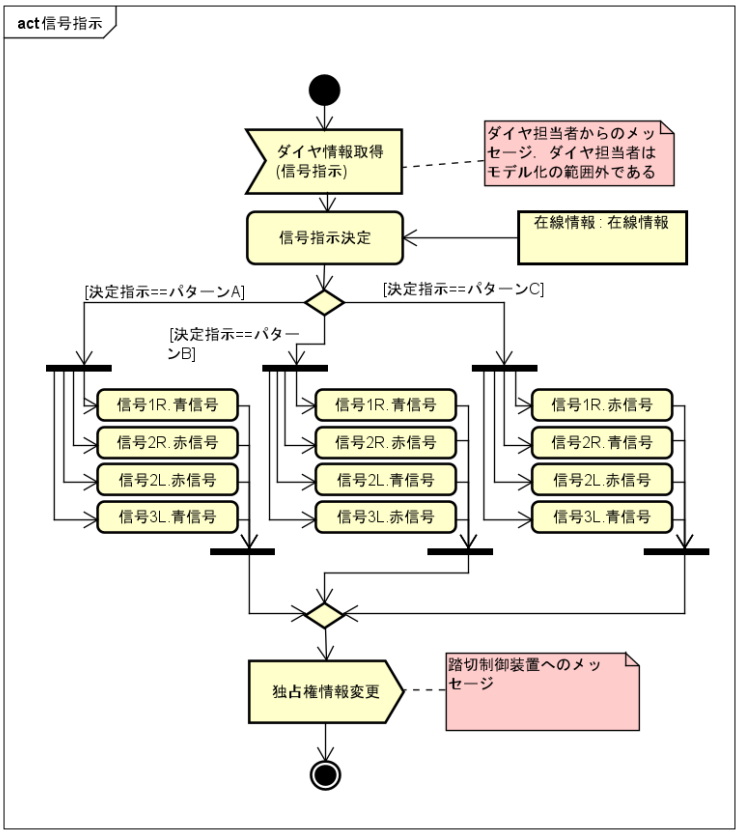

図 30　連動装置の信号制御に関する振る舞い記述

図 30 では，信号機の切り替えパターンの振る舞いが記述されている．ここでは，信号機の組み合わせが (1R,2R,2L,3L)={(青, 赤, 赤, 青),(青, 赤, 青, 赤),(赤, 青, 赤, 青)}であり，それぞれが 1/3 の確率で選択される振る舞いとなっている．ただしここでは，システムの運用に関するデータをもとに近似した振る舞いという想定である．このようにして SysML や UML 等で振る舞いを記述しておくことによって振る舞いに対するレビューをすることが可能になり，その妥当性を確認することができる．

次に，列車の振る舞いを SysML のステートマシン図によって記述したものを図 31 に示す．ここでは，例として TrainA の振る舞いを示す．

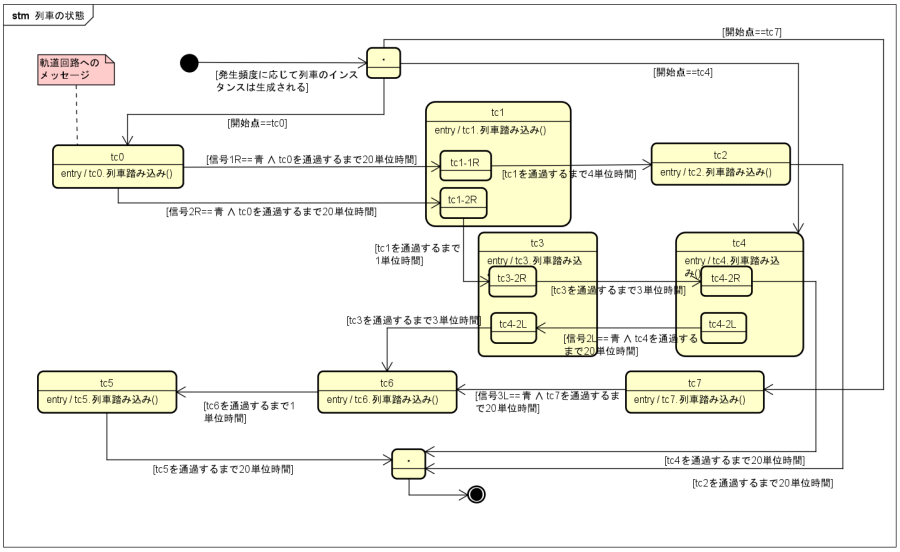

図 31　列車 (TrainA) のステートマシン図による振る舞い記述

図には，列車が移動する軌道回路の情報や，移動に必要な時間といった振る舞いが含まれている．

6.2.3　シナリオ生成用モデルの記述例 (手順 3)

ここでは，手順 2 で得られたコントロールストラクチャの振る舞いの記述をもとに，各コンポーネントの入出力やコントロールアルゴリズム，プロセスモデル等をそれぞれ UPPAAL SMC でモデル化する．図に，連動装置に対応するコンポーネントにおける，コントロールストラクチャの出力側オートマトンを示す．

このモデルは，先に述べた図 30 の振る舞いをもとに記述されている．例えば，初期ロケーション INIT からの遷移は，点線で示された矢印によって確率的な振る舞いが表現されたものとなっている．また，ロケーションの内部に C が示された状態はコミットロケーションと呼ばれ，時間経過を許さない振る舞いを表していしたがって，図 30 に示されているような，4 種類の信号が同時に設定されて，

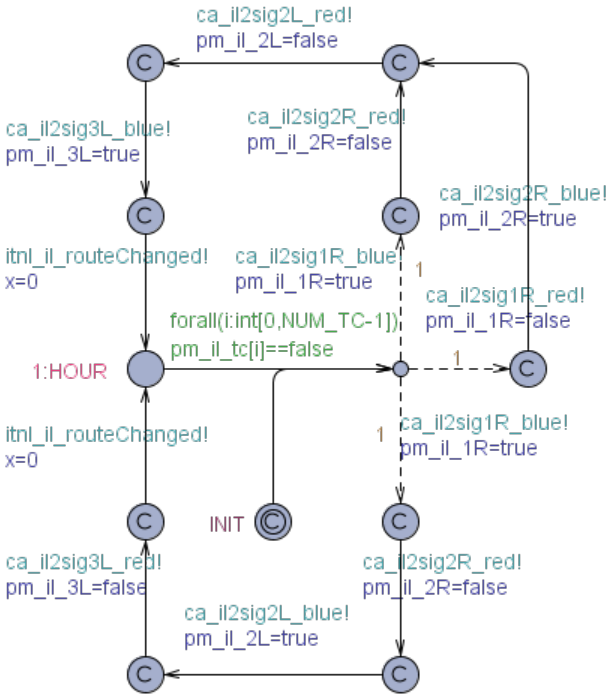

図 32　連動装置におけるコントロールアクションの出力側オートマトンの例

各信号にコントロールアクションとしてメッセージが送信されている振る舞いが表現されている．

次に，連動装置におけるフィードバックの受信側と送信側に対応するオートマトンを図 33 に示す．

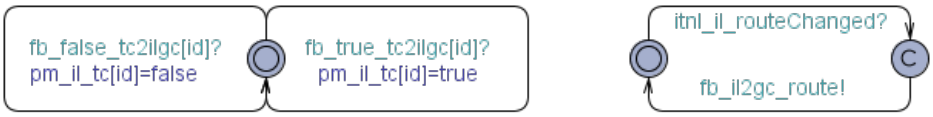

図 33　フィードバックの入力側 (左) と出力側 (右) に対応するオートマトンの例

図 33 に示されたフィードバックの受信側に対応するオートマトンは，センサである軌道回路からのフィードバックを受け取ることで，プロセスモデル変数が更新されている振る舞いを表している．また，フィードバックの送信側に対応するオートマトンは，コンポーネント間チャネル itnl_il_routeChanged?によって同期を行いながら，踏切制御装置に対してフィードバック fb_il2gc_route!を送信している．

次に，被コントロールプロセスである列車のモデルを図 34 に示す．ここでは，例として TrainA のフィードバック出力側に対応するオートマトンを示している．

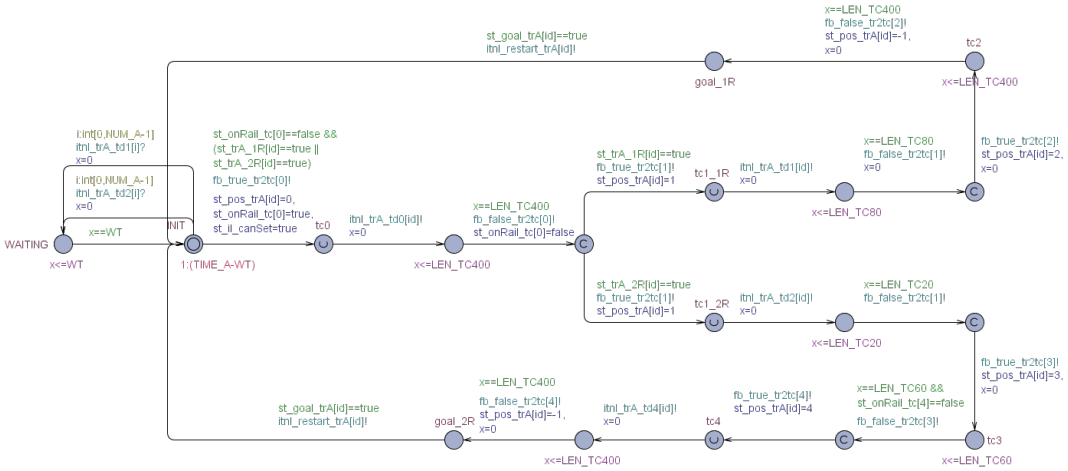

図 34　列車 (TrainA) のモデルの例

このモデルでは，ロケーション INIT に列車がランダムに出現することを表現するためのレートが定義されている．ここでは，平均して 900 単位時間に一回列車が出現するモデルになっている．本研究では，他の種類の列車である TrainB と TrainC についても同様の確率値を与えている．

次に，センサである踏切制御子と，環境に対応するオートマトンの一部を図 35 にそれぞれ示す．

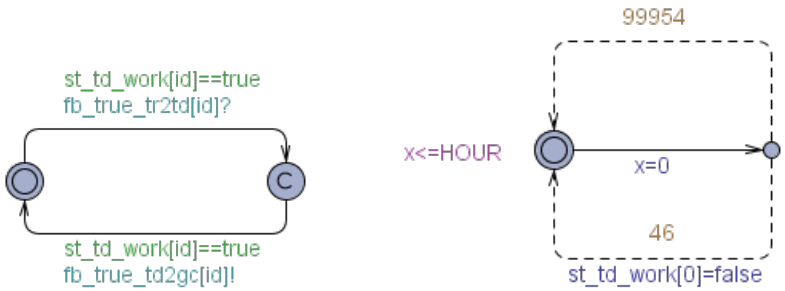

図 35　センサである踏切制御子 (左) と故障を制御する環境 (右) に対応するオートマトンの例

ここで，センサには故障状態の有無を表す状態変数 st_td_work[] が定義されており，この値が true の間動作することが可能である．一方で，右側のオートマトンは環境に対応するオートマトンであり，一定の確率で変数の値を false にすることでセンサの確率的な故障を表現している．この例では，過去の鉄道におけるインシデント情報 [33] をもとに (HS4) における故障の確率を設定している．

6.2.4　シナリオ観測用モデルの記述例 (手順 4)

次に，手順 1 で得られたシナリオをもとに，各シナリオを観測するためのモデルを作成する．ここでは，例として HS1 のイベント列を表 10 に示し，それに対応するシナリオ観測用オートマトンを図 36 に示す．

HS1 のイベント列は 5 つのイベントに分けて考えることができる．各イベントは図 36 に示したシナリオ観測用オートマトンの各辺に対応している．例えば，表におけるイベント番号 1 の連動装置の状態は，シナリオ観測用オートマトンのロ

表 10　HS1 のイベント列

イベント番号	TrainA	TrainB	連動装置	踏切制御装置
1			1R 青，2R 赤， 2L 青，3L 赤	
2	始動点 td0 に進入			
3				下り列車 (TrainA) を検知し 踏切鳴動開始
4		軌道回路 tc3 に進入		
5				TrainA が tc3 に進入したと 誤判断し，それが 15 秒以内 だったため，異常検知を行い 遮断継続

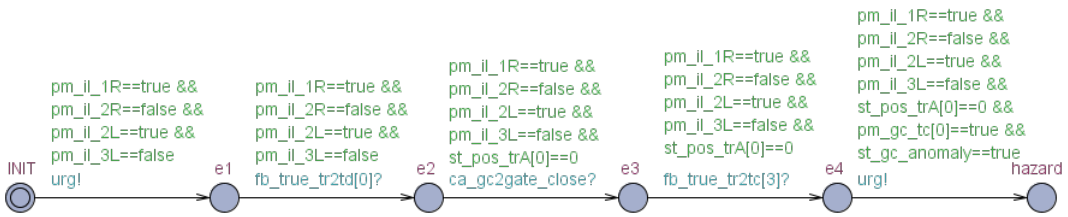

図 36　HS1 のシナリオ観測用モデル

ケーション INIT から e1 までの遷移に対応している．他のシナリオも同様にしてシナリオ観測用オートマトンを定義する．

6.2.5 シナリオに対する検証例 (手順 5)

HS1 を観測するための UPPAAL テンプレート名を scenario1 とすると，検証式は以下のように記述できる．

Pr[<=3600] (<> scenario1.hazard)

ここで，HOUR は 3600 単位時間を表している．同様にして，HS2 と HS3 も 3600 単位時間の範囲でクエリを与えて検証した結果を表 11 に示す．

表 11 検証結果

シナリオ	発生確率	検証時間 [s]
HS1	[2.56e-2, 2.76e-2]	495.187
HS2	[1.09e-3, 3.09e-3]	41.984
HS3	[0.00e+0, 2.00e-6]	6989.395
HS4	[3.24e-4, 2.32e-3]	25.781

統計的モデル検査法によって得られた発生確率は 95%信頼区間として得られている．また，近似パラメータは $\varepsilon = 0.001$ としている．定性的なモデル検査によって反例を分析したところ，シナリオがモデル上で再現できることが確認できた．また，各シナリオの検証時間は合計で約 7552.347s≃126m となった．このことから，UPPAAL SMC では，しばしば一般的なモデル検査法で問題となる状態爆発問題 [8] を引き起こすことなく，現実的な時間内で検証を完了できることが確認できる．HS3 の検証時間が長い理由は，確率値の範囲を狭めるためにシミュレーションの数を増やしているためである．確率値の範囲を狭める理由については後述する．

6.2.6 シナリオに対するリスク分析例 (手順 6)

まずは，表 12 に RAMS 規格 [2] を参考に作成したイベントの確率と深刻度からリスクのレベルを評価するための表の例を示す．この例では，ハザードの発生頻度が Incredible から Frequent の 6 段階であり，Frequent が最も発生頻度が高い．また，深刻度は Insignificant から Catastrophic の 4 段階で，Catastorophic が最も深刻度が大きいことを意味している．発生頻度と深刻度の組み合わせからリスクレベルは 1 から 4 まで定義され，4 が最も優先的に対策される必要があるものとなっている．

表 12 発生確率と深刻度によるリスクレベルの例

発生頻度	発生確率	リスクレベル			
Frequent	1e-1 $\leq$	3	4	4	4
Probable	1e-2 $\leq$ to $<$ 1e-1	2	3	4	4
Occasional	1e-3 $\leq$ to $<$ 1e-2	2	3	3	4
Remote	1e-4 $\leq$ to $<$ 1e-3	1	2	3	3
Improbable	1e-5 $\leq$ to $<$ 1e-4	1	1	2	2
Incredible	$<$ 1e-5	1	1	1	1
		Insignificant	Marginal	Critical	Catastrophic
		深刻度			

検証結果と表 12 からシナリオの優先順位付けの例を表 13 に示す．

表 13 各シナリオに対するリスク分析結果

シナリオ	発生確率	深刻度	リスクレベル
HS1	[2.56e-2, 2.76e-2]	Marginal	3
HS2	[1.09e-3, 3.09e-3]	Critical	3
HS3	[0.00e+0, 2.00e-6]	Catastrophic	1
HS4	[3.24e-4, 2.32e-3]	Critical	3

ここで仮定したシナリオの深刻度は，ハザードから予想される損失をもとに検討した．この結果から，HS1 がリスクレベル 3 と評価され，深刻度はどのシナリ

オよりも低いにも関わらず，HS2 と HS3 と同等のリスクレベルとなった．また，HS3 は最も深刻度が高いが，リスクレベルが 1 から 4 となり，他のシナリオのリスクとの比較が難しい結果となった．この結果については後に第 6.3 節で考察する．こうしたリスクレベルの結果をもとに，対策すべきハザードの優先順位や，どの程度コストをかけるのかといった計画が検討される．

6.3 事例に対する評価

事例を通じて，提案手法により，STAMP/STPA で識別されたシナリオが形式モデル上で表現できることが確認できた．具体的には，第 6.2.3 節で示した通り，図 32 から 35 に示したようなシナリオ生成用モデルおよび図 36 に示したようなシナリオ観測用モデルのような形式モデルが得られた．

本研究では，これまで述べたように STAMP/STPA で分析した結果得られるコントロールストラクチャやシナリオ，ハザード要因から系統的にモデル化する方法を示した．モデル化には，まず振る舞いモデルを SysML などの図で表現してから，それらを手作業で確率時間オートマトンに直して検証する方法について述べた．モデル化するために作成した SysML のモデルは合計で 14 個であり，Uppaal SMC におけるテンプレート数は 47 個となった．形式的なモデルを生成した後の検証は Uppaal SMC によって自動的に行われる．そのため，今後はモデルの規模が大きくなった場合でも手法を現実的な時間内で適用可能にするために変換作業の効率化を行うことが求められる．これまでに UML や SysML から形式的なモデルを自動的に生成する研究が多くなされているため，自動化は原理的に可能であると考えられる．

次にシナリオの優先順位付けについて述べる．Uppaal SMC による検証の結果，各シナリオに対する発生確率が表 11 のような形で得られた．これらは，各シナリオのリスクにおける発生確率として用いることができるので，原理的にはシナリオの優先順位付けができることが確認できた．ただし，今回仮定したリスクレベルを用いてリスクを評価したところ，当初の設定していた近似パラメータでは，リスクレベルが決定できないものが存在した．

具体的には，今回与えた近似パラメータである $\varepsilon = 0.001$ で検証を行った結果，

HS3 は確率の範囲が [0.00e+0, 2.00e-3] と広いためリスクレベルが 1 から 4 の範囲となり，他のシナリオとリスクを比較することが難しかった．この場合，リスクレベルを決定するために確率値の範囲をより狭める必要がある．これは，今回の検証で与えた近似パラメータである $\varepsilon = 0.001$ をより小さな値に変更することで実現できる．パラメータを $\varepsilon = 0.000001$ と変更して再度検証を行ったところ，表 13 の結果となり，リスクレベルは 1 であることが確認できた．このことから，HS3 は最も深刻度が高いシナリオであるが，リスクレベルは最も低いことがわかる．この理由としては，今回対象としたモデルでは，単一の軌道回路上に複数の列車が進入することがない振る舞いとなっているためである．こうした排他制御による安全のための仕組みは，鉄道システムにおける最も基本的なものとなっている．

リスクレベルの数は，一般的には 4 段階から 5 段階程度で十分であると考えられている．例えば，本研究で参照した鉄道分野の RAMS 規格では，表 12 に示したように，4 段階のリスクレベルを決定して対策が検討される．また，自動車分野の機能安全に関する国際規格である ASIL では，5 段階でハザードのリスクレベルが決定される [3]．本研究では RAMS を利用したが，提案手法は統計的モデル検査法によってハザードに対する任意の確率値を計算することが可能であるため，より詳細なリスクレベルを仮定した場合にも応用することが可能であると考えられる．

7. 関連研究

リスク分析に関連する研究として，従来型のハザード分析手法である FTA や ETA を用いたリスク分析を提案した研究がある．Leitner ら [21] は，鉄道システムにおける事故シナリオのリスクを FTA と ETA で分析している．リスクの算出には，人間の振る舞い等に関する確率等も含めて過去の運用に基づく統計データを利用して計算している．しかし，本研究では，STAMP/STPA で識別されるような，故障がなくても機能間の動的な相互作用によって発生するシナリオの分析を行うことを目的としている．また，本研究の事例として扱うようなリアルタイム性を持つシステムを対象とした分析を FTA や ETA によって分析することは困難である．

また，Hallerbach ら [16] は，テストによって自動運転システムの安全性を効率的に高めるための手法を提案している．具体的には，自動運転システムの運用において危険なシナリオを識別し，テストシナリオの優先付けを行う方法について述べている．シナリオの危険度は，車同士が衝突するまでの時間を求めるための計算式を利用して決定される．そのため，シナリオの危険度を厳密な基準によって測ることが可能である．一方で，この基準は自動運転システムにおける運用シナリオに特化しているため，STAMP/STPA を用いて識別されたシナリオの評価を行うことができるとは限らない．本研究の提案手法は自動運転システムに特化したものではないため，この手法と同様の精度を保証することは難しいが，自動運転システムに限らず様々な対象に適用してシナリオの優先付けを行うことができると考えられる．

次に，STAMP/STPA を応用した関連研究について説明する．Dakwat ら [12] は，STAMP/STPA で分析したコントロールストラクチャを Uppaal のモデルに変換し，STAMP/STPA で識別された安全制約を満たすか否かを検証する手法について提案している．また，Abdulkhaleq ら [6] は，ソフトウェアの実装を形式的モデルに変換し，STAMP/STPA で分析した安全制約を実装が満たすか否かを検証する手法を提案している．また，形式検証したモデルをもとに安全制約を満たしているテストケースを生成し，実装上でテストするための方法についても述べている．一方で，本研究が目的とする STAMP/STPA で識別されたシナリオを形

式的に検証するためのモデルを系統的に変換したり，それによってシナリオの優先付けを行うための方法は提案されていない．髙田ら [31] は，STAMP/STPA で識別されたシナリオに対して，FTA を用いたリスク評価を行うことで優先付けを行う手法を提案している．具体的には，鉄道の踏切システムを対象に，過去のインシデント情報からシナリオに含まれる事象の発生確率を見積もることで，最終的にハザードに至るリスクを求めている．しかしながら，本研究が対象とするような動的な相互作用を含む振る舞いを直接検証することは難しいと考えられる．

8. 結論と今後の課題

本研究では，STAMP/STPA で識別されたハザードのシナリオのリスクを求めるため，STAMP/STPA の結果を系統的に形式化する方法について提案した．また，鉄道の踏切制御システムを対象に提案手法を評価し，シナリオがモデル上で表現されていることが確認できた．また，各シナリオのリスク分析を行い，各シナリオに対して RAMS 規格を参考にリスク値を計算した．

第 6.3 節でも述べたように，本研究の提案手法は振る舞いモデルを記述してから形式モデルを作成する必要がある．したがって，対象システムの規模が大きい場合，記述される振る舞いモデルが大きくなるにつれて形式モデルに変換するための作業量も増えることになる．したがって，今後は本手法をより効率的に行うための自動化が必要になる．具体的には，UML や SysML で記述された振る舞いモデルから形式モデルを自動的に変換するツールを実装することなどが考えられる．先行研究として，Huang ら [14] は，SysML のステートマシン図から Uppaal における時間オートマトンを自動的に変換する方法について提案している．しかしながら，本研究が対象としている確率時間オートマトンに対する自動変換手法はこれまでに提案されていない．本研究における今後の課題としては，こうした手法を参考にしてモデルの変換を効率的に行う手法の検討が考えられる．

また，Uppaal SMC では検証時に与える近似パラメータ ε の値によって計算結果の精度が異なる．精度の高い確率値を求める場合，すなわち，信頼区間の幅を狭めるためには近似パラメータの値を小さくする必要がある．一方で，計算の精度を高めるためには，モデルに対してより多くのパスを生成して検証を行う必要があるため，検証時間が増えるという問題がある．したがって，ある性質の発生確率が非常に小さい場合，その値を観測するために精度を高める必要があるため，現実的な時間内に計算が終わらない可能性がある．これらはモンテカルロシミュレーションを基本とする手法に共通する課題となっている [18, 9]．

この問題に対する代表的な解決手法として，重点的サンプリング (importance sampling) や，重点的分割法 (importance splitting) などが提案されている [18]．本研究で対象としたシナリオの検証では，こうした手法を使うことなく各シナリオの発生確率を求めることができている．しかしながら，対象の発生確率がより小

さいものに分析対象を変更した場合に検証が困難になる可能性がある．そのため，今後は上記に述べたような解決手段を本研究の提案手法に適用するための方法を検討する必要があると考えられる．具体的には，重点的分割法を応用し，シナリオを複数の分割してそれぞれの発生確率から計算する方法などが考えられるが，こうした手法についても今後検討する予定である．

最後に，UPPAAL SMC では，オートマトンのロケーションや辺にテストコードを埋め込むことで抽象的なテストケースを生成することができる．テストケースの生成は，モデルに対してある一定の深さまでシミュレーションを行うことで，その実行パスからテストケースを生成することができる．その他に，本研究でもシナリオを検証するために用いたクエリを利用して，クエリが満たされる，すなわち，ハザードに到達するまでのシナリオに対応したテストケースを生成することも可能となっている．今後は，こうしたモデルを利用したテストケース生成への応用についても検討する予定である．

謝辞

本研究を進めるにあたり，ご指導，ご協力いただきました方々に心より御礼申し上げます．

奈良先端科学技術大学院大学 先端科学技術研究科 情報科学領域 ソフトウェア設計学研究室 飯田元 教授 には，研究に対する姿勢から論文のまとめ方，発表の仕方にいたるまで多岐にわたりご指導をいただきました．また，ソフトウェア工学に関する知識が浅かった私に対して，基礎から丁寧にご指導をいただきました．修士課程の二年間，自由な雰囲気の中で楽しく研究室生活を過ごすことができたのは，ひとえに先生のお人柄のおかげだと思います．ここに厚く御礼申し上げます．

奈良先端科学技術大学院大学 先端科学技術研究科 情報科学領域 ソフトウェア工学研究室 松本健一 教授 には，研究の方針や内容に関する多くのご指導を賜りました．特に，コロキアムにおける発表の場では研究の構成や，その後の方針に関する有意義なご意見をいただきました．心から感謝申し上げます．

奈良先端科学技術大学院大学 先端科学技術研究科 情報科学領域 超高信頼ソフトウェアシステム検証学研究室（宇宙航空研究開発機構 第三研究ユニット）片平真史 教授には，修士課程の二年間を通して，研究方針や発表方法，論文執筆などに関する非常に多くのご指導を賜りました．研究を進めるにあたって，実際の航空宇宙分野におけるシステム開発や最新の安全分析手法等の様々な技術に関する知見をいただきました．また，就職活動の際には，私が希望していた宇宙業界全般に関する貴重なアドバイスをしていただきました．ここに厚く御礼申し上げます．

奈良先端科学技術大学院大学 先端科学技術研究科 情報科学領域 超高信頼ソフトウェアシステム検証学研究室（宇宙航空研究開発機構 第三研究ユニット）石濱直樹 准教授には，ご多忙の身にも関わらず，研究の方針や論文執筆の方法等について熱心にご指導をいただきました．特に，研究構成の立て方について悩んでいたときには，常に適切なアドバイスをしていただきました．また，コロキアムなどの発表前には日夜親身になって資料の添削指導やご助言をいただきました．心より感謝申し上げます．

奈良先端科学技術大学院大学 先端科学技術研究科 情報科学領域 ソフトウェア設計学研究室（株式会社チェンジビジョン） 高井利憲 准教授には，研究活動全体を通してご指導をいただきました．ご多忙の身にも関わらず，研究方針や発表の方法，論文執筆をしていた際には，まるで自分の研究のように考察や議論をしていただきました．特に，形式的な技術について議論を行った際には，非才な身である私に対して，常に熱心にご指導をしていただきました．心より感謝致します．

奈良先端科学技術大学院大学 先端科学技術研究科 情報科学領域 ソフトウェア設計学研究室 市川昊平 准教授には，普段のミーティングや輪講の場を通して研究の進め方や発表の仕方についてご助言をいただきました．また，研究室へ所属する前で何も分からない状態だった私に対して丁寧に対応していただき，当時は非常に心強かったことを覚えています．深く感謝申し上げます．

奈良先端科学技術大学院大学 先端科学技術研究科 情報科学領域 ソフトウェア設計学研究室 高橋慧智 助教には，普段の研究活動を通して適切なご助言をいただきました．特に，コロキアムの発表に向けた練習の場では，資料の作成方法に関するご指導や研究に関する有意義なご質問をいただきました．厚く御礼申し上げます．

京都工芸繊維大学 情報工学・人間科学系 ソフトウェア工学研究室 崔恩瀞 助教には，研究に対する心構えや進め方，研究方針に対して多くのアドバイスをいただきました．特に，ミーティングやコロキアムの発表練習などを通して，発表資料の作成方法や論文の読み方といった研究活動の基本についてご指導を賜りました．心より感謝申し上げます．

三菱電機エンジニアリング株式会社 中村 哲 様，新井摩衣子 様 には，本研究の方針や内容について多くのご助言を賜りました．実際の開発現場における技術者としてのご意見は大変貴重であり，PBL 型授業であるアシュアランス演習を通して様々な知見をいただきました．心より感謝いたします．

東日本旅客鉄道株式会社 JR 東日本研究開発センター 北村知 様には，本研究を進める上で数多くの貴重なご助言を賜りました．特に，急なお願いにも関わらず迅速にご対応していただき，専門家としての立場から実際の鉄道システムにつ

いて多くのことを学ばせていただきました．厚く御礼申し上げます．

仙台工業高等専門学校 総合工学科 岡本圭史 教授には，本研究の方針について多くのご助言をいただきました．特に，安全分析手法や形式検証技術に関する議論の場では，非常に有意義なご意見を賜りました．心から感謝申し上げます．

日本大学 工学部 情報工学科 高信頼性システム研究室 関澤俊弦 准教授には，本研究の方針についてご助言をいただきました．特に，本研究で対象としている形式検証技術や統計的手法に関して，専門家としての立場から非常に有意義なご意見を賜りました．深く感謝申し上げます．

超高信頼ソフトウェアシステム検証学研究室（有人宇宙システム株式会社）柿本和希 様には，研究を進める上で数多くのご助言を賜りました．ご多忙の中，定例のミーティングに参加していただき，研究方針の立て方や研究活動に対する姿勢など，多くのことを学ばせていただきました．厚く御礼申し上げます．

奈良先端科学技術大学院大学 小川暁子 様，仲田綾奈 様には，ご多忙にもかかわらず，出張や留学での事務処理を通して大変お世話になりました．心より感謝申し上げます．

ソフトウェア設計学研究室および，超高信頼ソフトウェアシステム検証学研究室の皆様には，ミーティングや輪講，日々の議論を通して多くのことを学ばせていただきました．非常に明るい雰囲気の研究室で，この二年間の研究室生活を非常に楽しく過ごさせていただきました．特に，超高信頼ソフトウェアシステム検証学研究室の先輩である入江琴子 様には，研究発表資料の添削や論文添削等を行って頂きました．深く感謝申し上げます．また，超高信頼ソフトウェアシステム検証学研究室の後輩である千田将也 様には，研究について深く議論を交わしたり，多忙にもかかわらず修士論文の確認を行っていただきました．心より感謝致します．

最後に，大学院に進学して研究を行うための機会を与え，経済的にも支援していただいた家族に感謝の意を表して，謝辞とさせていただきます．

参考文献

[1] https://www.pegasusprojekt.de/en/home

[2] IEC62278: Railway applications – specification and demonstration of reliability, availability, maintainability and safety (RAMS), IEC, Standard, 2002.

[3] ISO26262: Road vehicles–Functional safety, ISO, Standard, 2011.

[4] ISO31000: Risk management – Principles and guidelines, ISO, Standard, 2009.

[5] J. D. Andrews and S. J. Dunnett. Event-tree analysis using binary decision diagrams. *IEEE Transactions on Reliability*, vol.49, no.2, pp.230–238, 2000.

[6] A. Abdulkhaleq, S. Wagner, and N. Leveson. A Comprehensive Safety Engineering Approach for Software-Intensive Systems Based on STPA. *Procedia Engineering*, vol.128, pp.2–11, 2015.

[7] G. Cicotti and A. Coronato. Towards a Probabilistic Model Checking-based approach for Medical Device Risk Assessment. *International Symposium on Medical Measurements and Applications (MeMeA2015)*, pp.180–185, 2015.

[8] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*, The MIT Press, 2000.

[9] F. Cérou, A. Guyader, and M. Rousset. Adaptive multilevel splitting: Historical perspective and recent results. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol.29, no.4, 043108, 2019.

[10] A. David, K. G. Larsen, A. Legay, M. Mikučionis, and D. B. Poulsen. UPPAAL SMC Tutorial. *International Journal on Software Tools for Technology Transfer*, vol.17, no.4, pp.397–415, 2015.

[11] A. David, K. G. Larsen, A. Legay, M. Mikučionis, D. B. Poulsen, J. van Vliet, and Z. Wang. Statistical Model Checking for Networks of Priced Timed Automata. *International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS 2011)*, pp.80–96, 2011.

[12] A. L. Dakwat and E. Villani. System safety assessment based on STPA and model checking. *Safety Science*, vol.109, pp.130–143, 2018.

[13] R. Ferdous, F. Khan, R. Sadiq, P. Amyotte, and B. Veitch. Fault and Event Tree Analyses for Process Systems Risk Analysis: Uncertainty Handling Formulations. *Risk Analysis*, vol.31, no.1, pp.86–107, 2011.

[14] X. Huang, Q. Sun, J. Li, and T. Zhang. MDE-Based Verification of SysML State Machine Diagram by UPPAAL. *International Conference on Trustworthy Computing and Services (ISCTCS 2012)*, pp.490–497, 2012.

[15] T. Hérault, R. Lassaigne, F. Magniette, and S. Peyronnet. Approximate Probabilistic Model Checking. *International Workshop on Verification, Model Checking, and Abstract Interpretation (VMCAI 2004)*, pp.73–84, 2004.

[16] S. Hallerbach, Y. Xia, U. Eberle, and F. Koester. Simulation-Based Identification of Critical Scenarios for Cooperative and Automated Vehicles. *SAE International Journal of Connected and Automated Vehicles*, vol.1, no.2018–01–1066, pp.93–106, 2018.

[17] C. Jegourel, A. Legay, and S. Sedwards. A Platform for High Performance Statistical Model Checking – PLASMA. *International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2012)*, pp.498–503, 2012.

[18] C. Jegourel, A. Legay, and S. Sedwards. Importance Splitting for Statistical Model Checking Rare Properties. *Computer Aided Verification (CAV 2013)*, pp.576–591, 2013.

[19] F. I. Khan and S. A. Abbasi. Models for domino effect analysis in chemical process industries. *Process Safety Progress*, vol.17, no.2, pp.107–123, 1998.

[20] M. Kwiatkowska, G. Norman, and D. Parker. PRISM: Probabilistic Symbolic Model Checker. *International Conference on Modelling Techniques and Tools for Computer Performance Evaluation (TOOLS 2002)*, pp.200–204, 2002.

[21] B. Leitner. A General Model for Railway Systems Risk Assessment with the Use of Railway Accident Scenarios Analysis. *Procedia Engineering*, vol.187, pp.150–159, 2017.

[22] N. G. Leveson. *Engineering a Safer World: System Thinking Applied to Safety.* The MIT Press, 2012.

[23] A. Legay, B. Delahaye, and S. Bensalem. Statistical Model Checking: An Overview. *International Conference on Runtime Verification (RV 2010)*, pp.122–135, 2010.

[24] W. S. Lee, D. L. Grosh, F. A. Tillman, and C. H. Lie. Fault Tree Analysis, Methods, and Applications – A Review. *IEEE Transactions on Reliability*, vol.R-34, no.3, pp.194–203, 1985.

[25] K. G. Larsen, P. Pettersson, and W. Yi. UPPAAL in a nutshell. *International Journal on Software Tools for Technology Transfer*, vol.1, no.1–2, pp.134–152, 1997.

[26] N. G. Leveson and J. P. Thomas. STPA Handbook. 2018.

[27] M. Tsuji, T. Takai, K. Kakimoto, N. Ishihama, M. Katahira, and H. Iida. Prioritizing Scenarios based on STAMP/STPA Using Statistical Model Checking. *4th International Workshop on Testing Extra-Functional Properties and Quality Characteristics of Software Systems (ITEQS 2020)*, 9 pages, 2020.

[28] P. Yang, R. Karashima, K. Okano, and S. Ogata. Automated inspection method for an STAMP/STPA – Fallen Barrier Trap at Railroad Crossing –. *Procedia Computer Science*, vol.159, pp.1165–1174, 2019.

[29] 北村知. JR 東日本における STAMP 活用の取り組み. *SEC journal*, vol.13, no.4, pp.30–37, 2018.

[30] 中村英夫. 列車制御–安全・高密度運転を支える技術–. オーム社. 2011.

[31] 髙田哲也, 望月寛, 高橋聖, 中村英夫. 鉄道信号の安全性解析に関する手法の提案, 横幹連合カンファレンス, B-4-3, 2019.

[32] 梅原淳. 鉄道ダイヤを支える技術. 株式会社秀和システム, 2016.

[33] 安全報告書 2012, 北海道旅客鉄道株式会社, 2012.

発表リスト

- 辻光顕, 高井利憲, 片平真史, 石濱直樹, 柿本和希, 飯田元. ハイブリッドシステムの安全検証に向けた形式的な STAMP モデルの提案. 電子情報通信学会技術研究報告, vol.118, no.471, pp.91–96, 2019 年 3 月.

- 辻光顕, 高井利憲, 柿本和希, 石濱直樹, 片平真史, 飯田元. 統計的モデル検査法を用いた STAMP/STPA におけるシナリオのリスク分析. ウィンターワークショップ 2020・イン・京都 論文集, pp.1–2, 2020 年 1 月.

- Mitsuaki Tsuji, Toshinori Takai, Kazuki Kakimoto, Naoki Ishihama, Masafumi Katahira, and Hajimu Iida. Prioritizing Scenarios based on STAMP/STPA Using Statistical Model Checking. *4th International Workshop on Testing Extra-Functional Properties and Quality Characteristics of Software Systems (ITEQS 2020)*, 9 pages, March 2020.

付録

A. SysMLで記述された振る舞いのモデル

B. Uppaal SMCで記述された形式モデル

A. SysML で記述された振る舞いのモデル

ここでは，第 6.2.2 節で述べた振る舞いモデルの記述結果を示す．

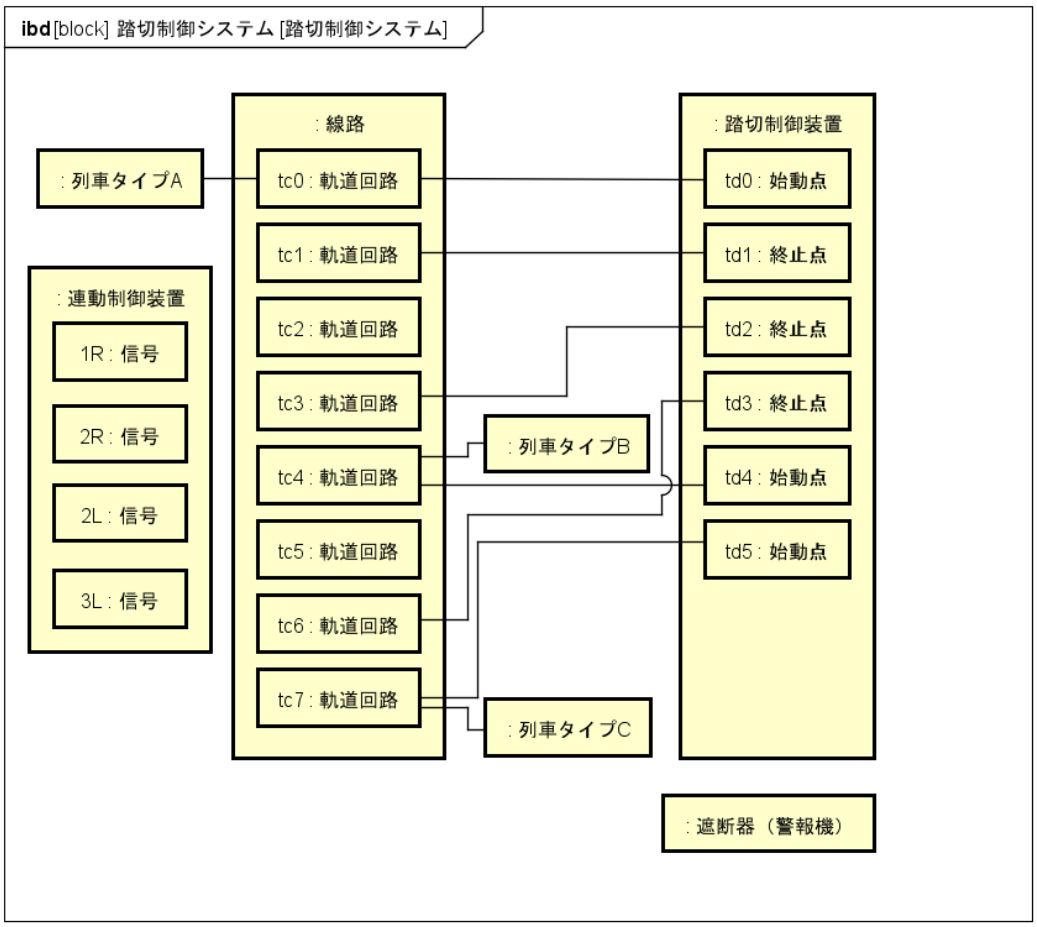

図 37 踏切制御システムの構成 (1)

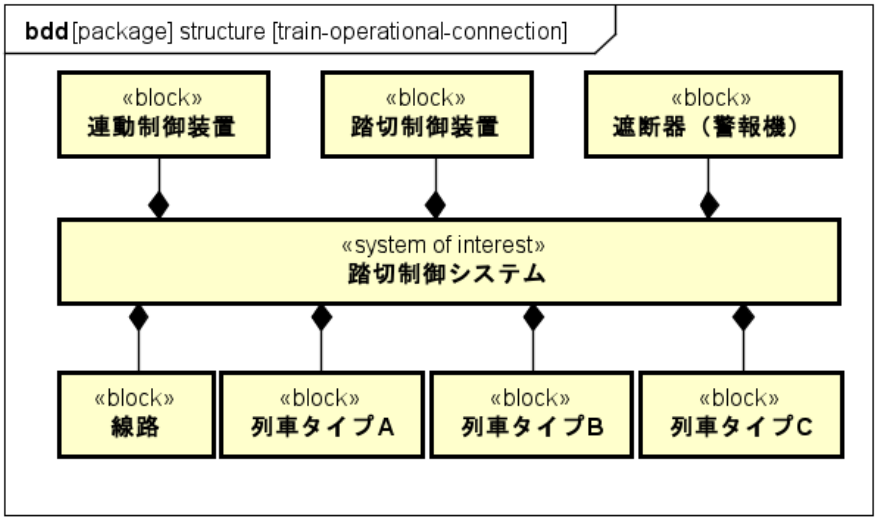

図 38 踏切制御システムの構成 (2)

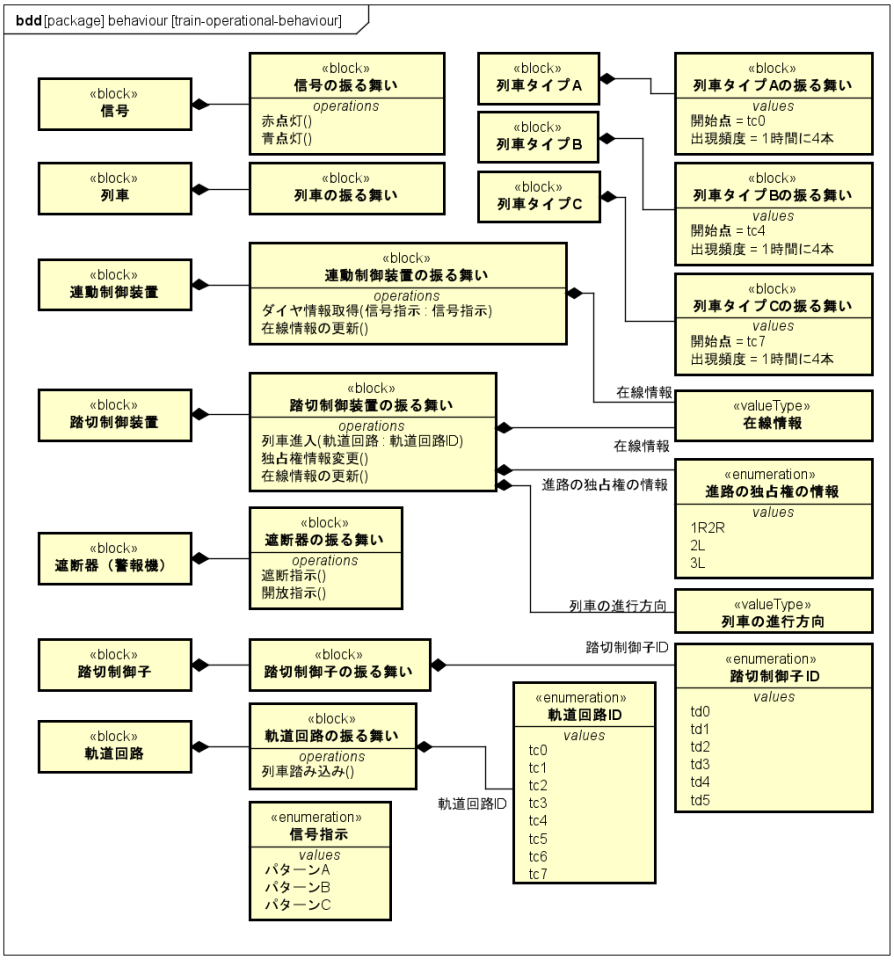

図 39 踏切制御システムの構成 (3)

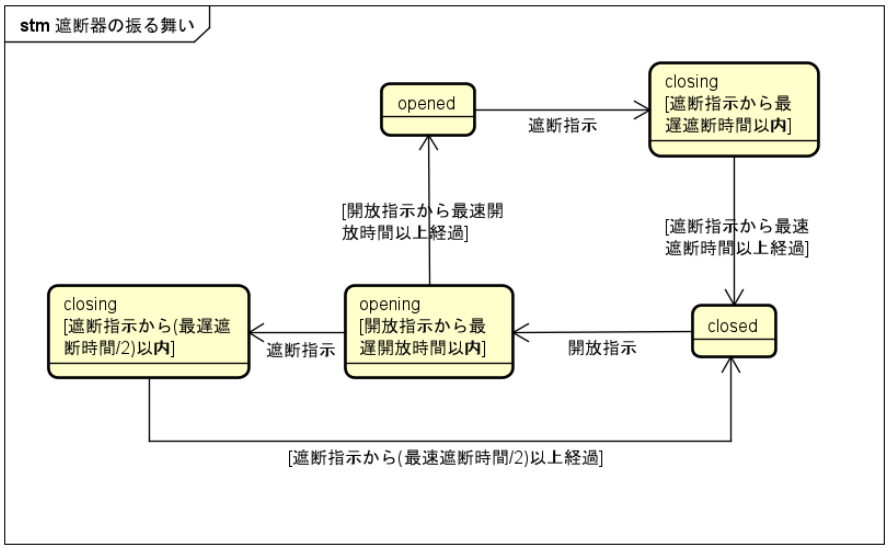

図 40　遮断器の振る舞い

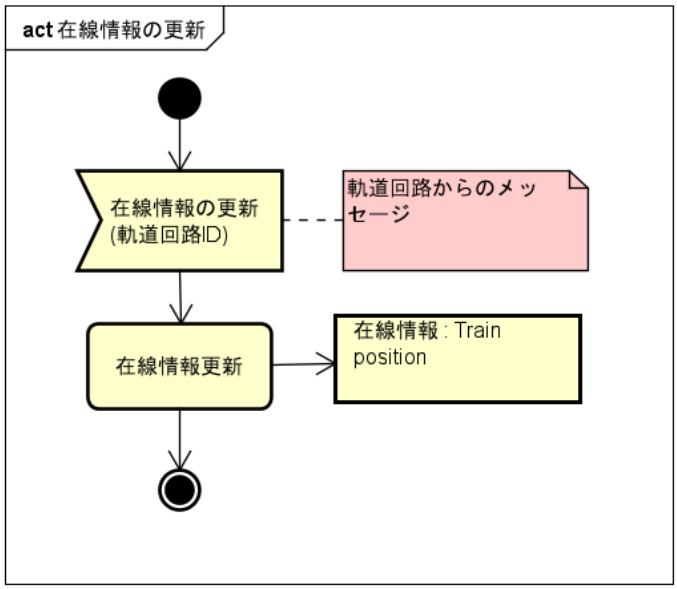

図 41　在線情報の更新に対する振る舞い (1)

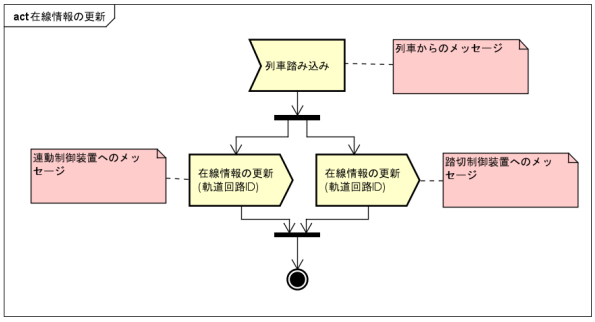

図 42　在線情報の更新に対する振る舞い (2)

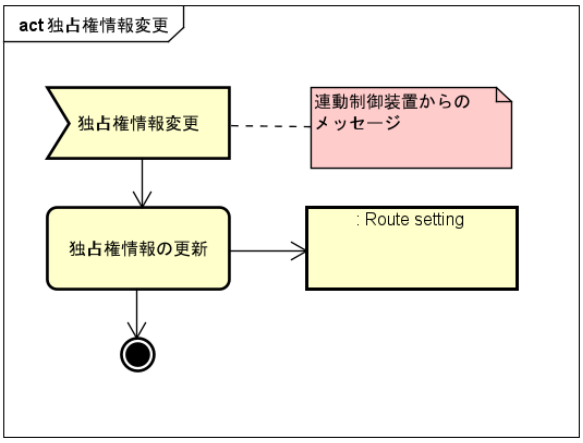

図 43　独占権の情報の変更に対する振る舞い

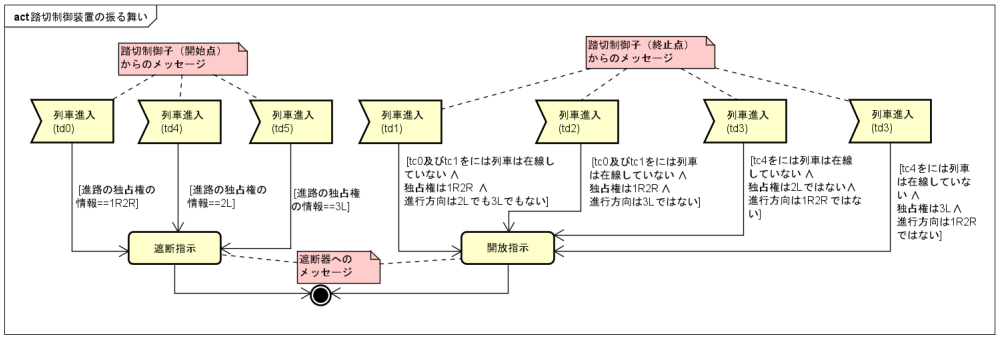

図 44 踏切制御装置の振る舞い

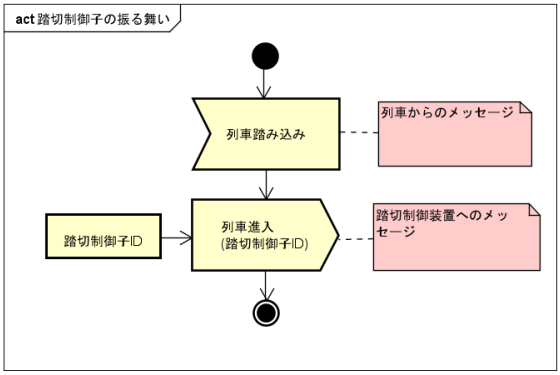

図 45 踏切制御子の振る舞い

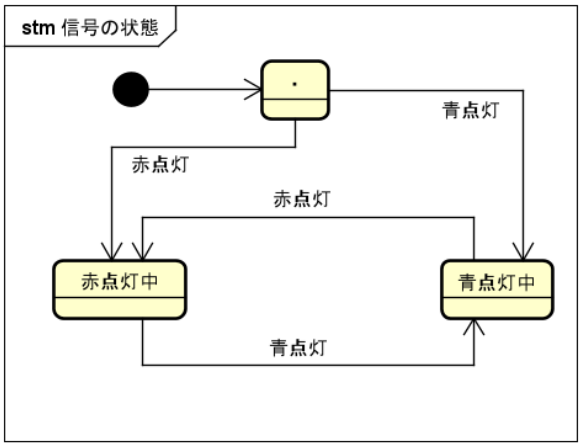

図 46 信号機の振る舞い

B. UPPAAL SMC で記述された形式モデル

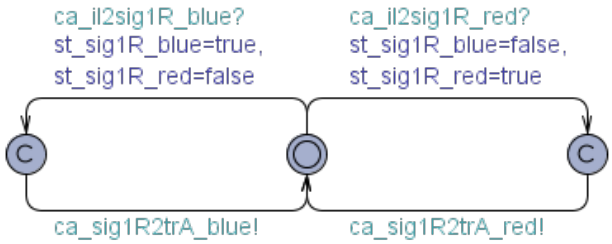

図 47　アクチュエータである信号機 (1R) に対応するオートマトン

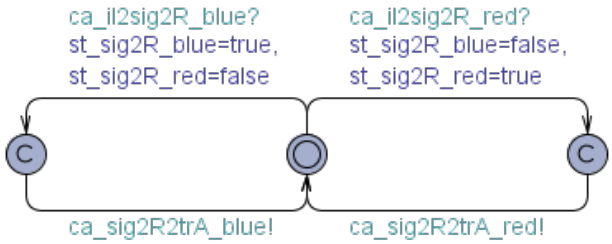

図 48　アクチュエータである信号機 (2R) に対応するオートマトン

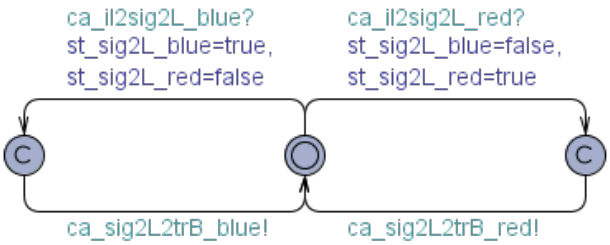

図 49　アクチュエータである信号機 (2L) に対応するオートマトン

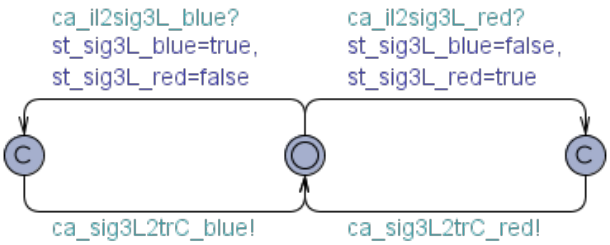

図 50　アクチュエータである信号機 (3L) に対応するオートマトン

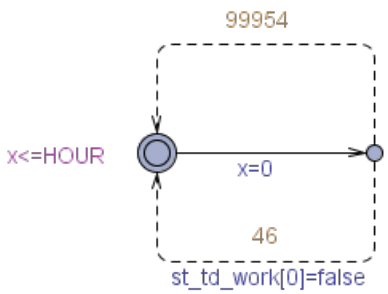

図 51　環境に対応するオートマトン (1)

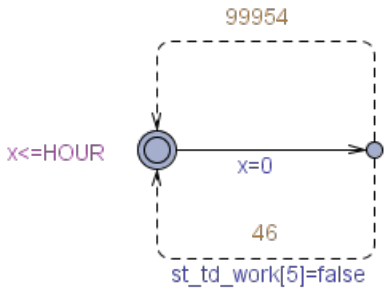

図 52　環境に対応するオートマトン (2)

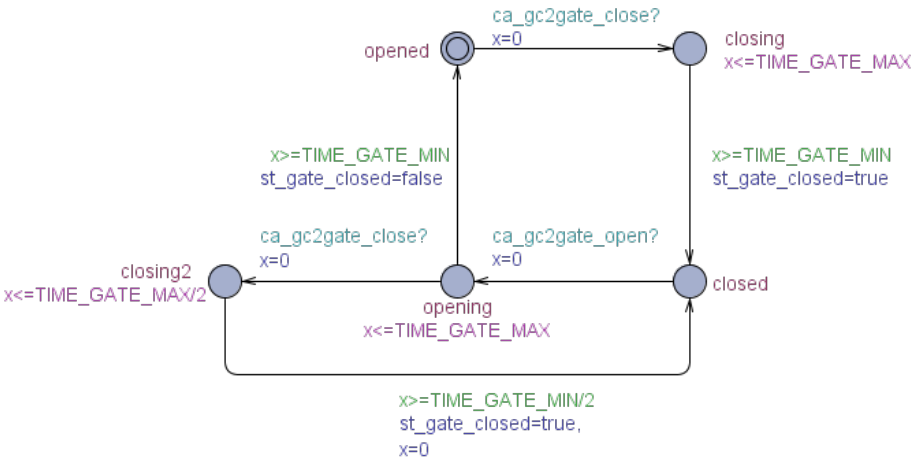

図 53　遮断器における C_{in} 型オートマトン

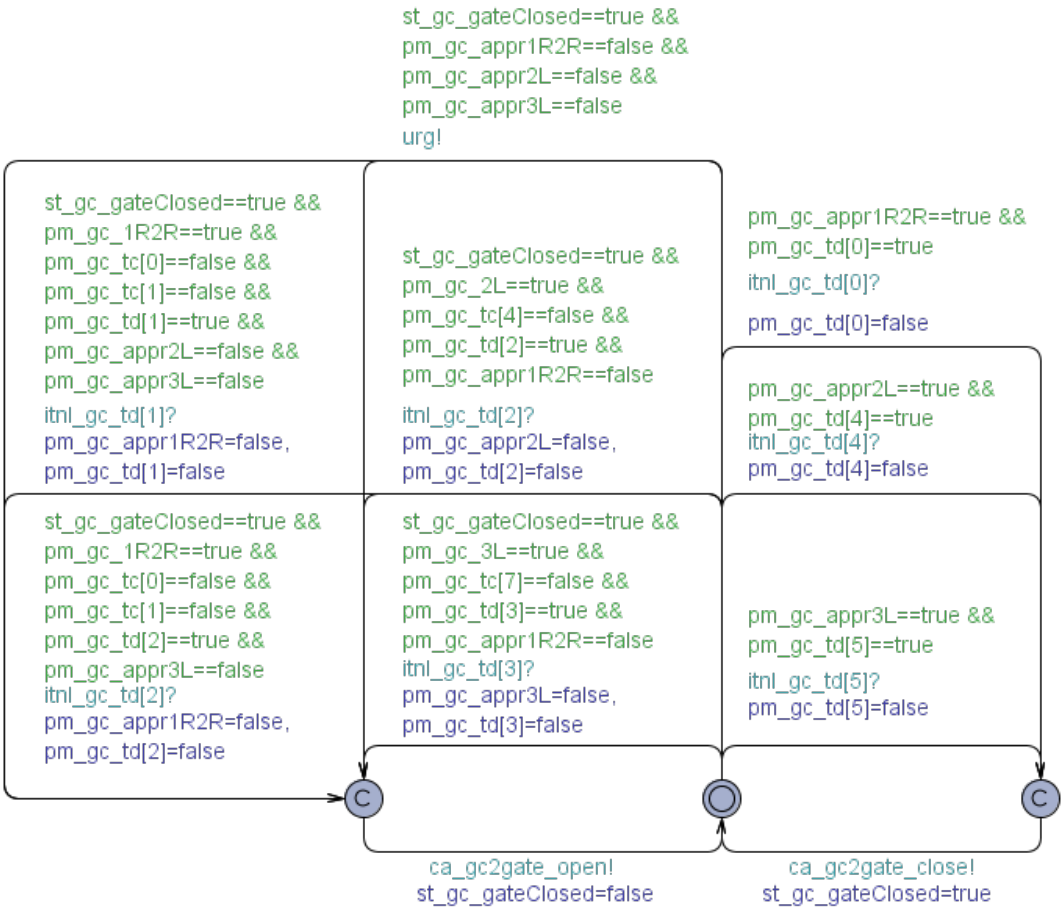

図 54　踏切制御装置における C_{out} 型オートマトン (1)

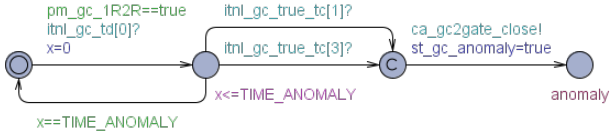

図 55　踏切制御装置における C_{out} 型オートマトン (2)

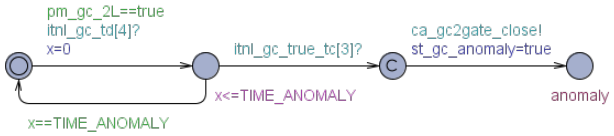

図 56　踏切制御装置における C_{out} 型オートマトン (3)

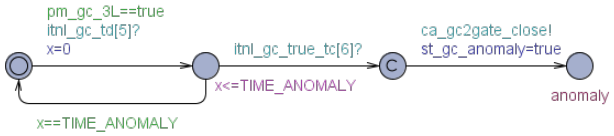

図 57　踏切制御装置における C_{out} 型オートマトン (4)

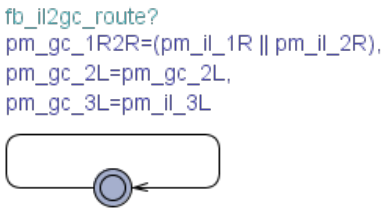

図 58　踏切制御装置における F_{in} 型オートマトン (1)

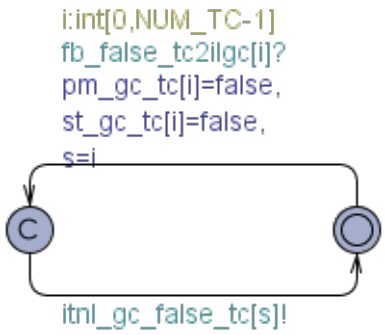

図 59　踏切制御装置における F_{in} 型オートマトン (2)

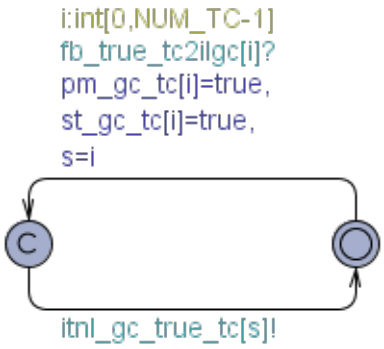

図 60　踏切制御装置における F_{in} 型オートマトン (3)

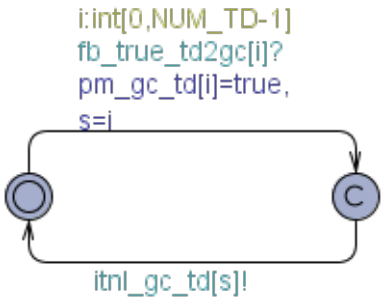

図 61　踏切制御装置における F_{in} 型オートマトン (4)

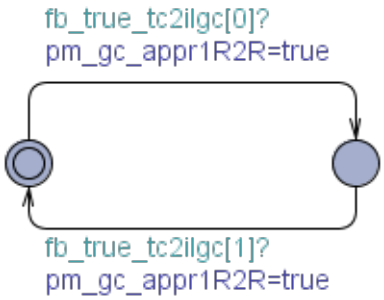

図 62　踏切制御装置における F_{in} 型オートマトン (5)

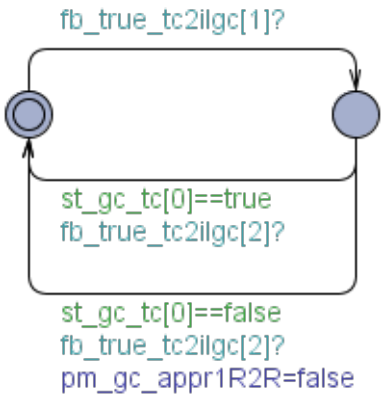

図 63　踏切制御装置における F_{in} 型オートマトン (6)

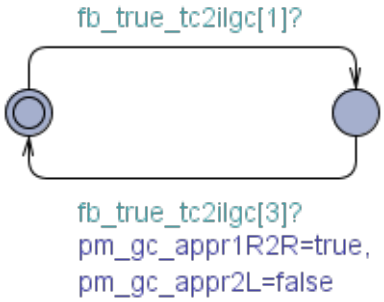

図 64　踏切制御装置における F_{in} 型オートマトン (7)

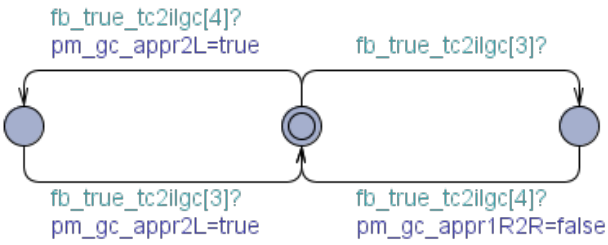

図 65　踏切制御装置における F_{in} 型オートマトン (8)

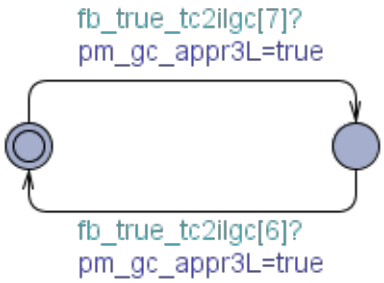

図 66　踏切制御装置における F_{in} 型オートマトン (9)

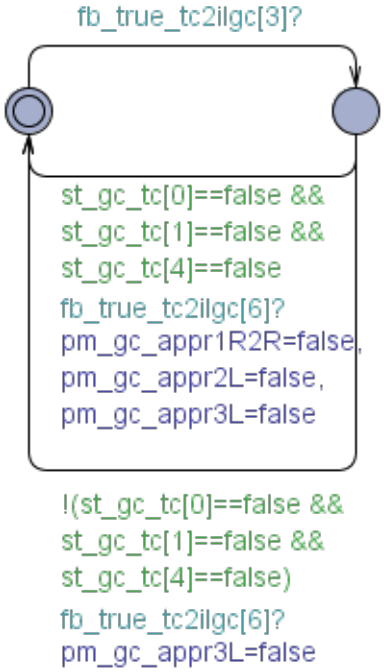

図 67　踏切制御装置における F_{in} 型オートマトン (10)

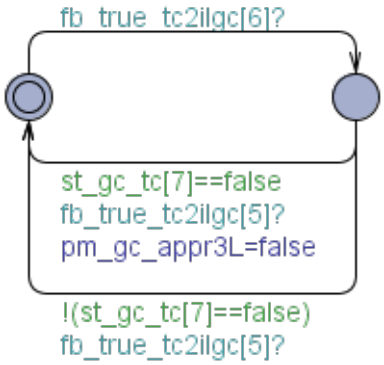

図 68　踏切制御装置における F_{in} 型オートマトン (11)

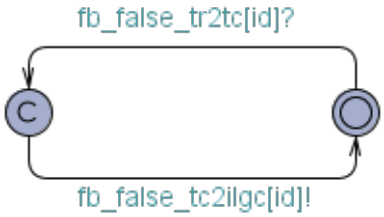

図 69　センサーである軌道回路に対応するオートマトン (1)

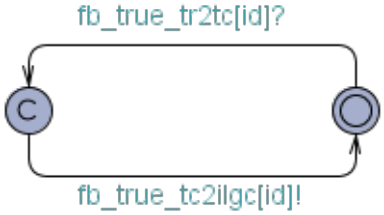

図 70　センサーである軌道回路に対応するオートマトン (2)

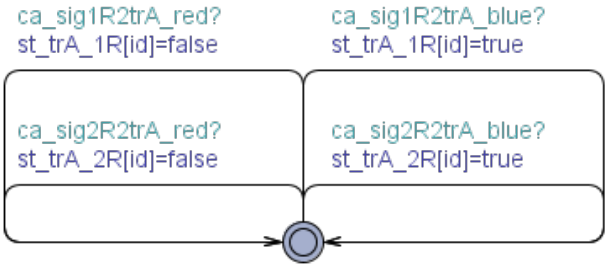

図 71　列車 (TrainA) における C_{in} 型オートマトン

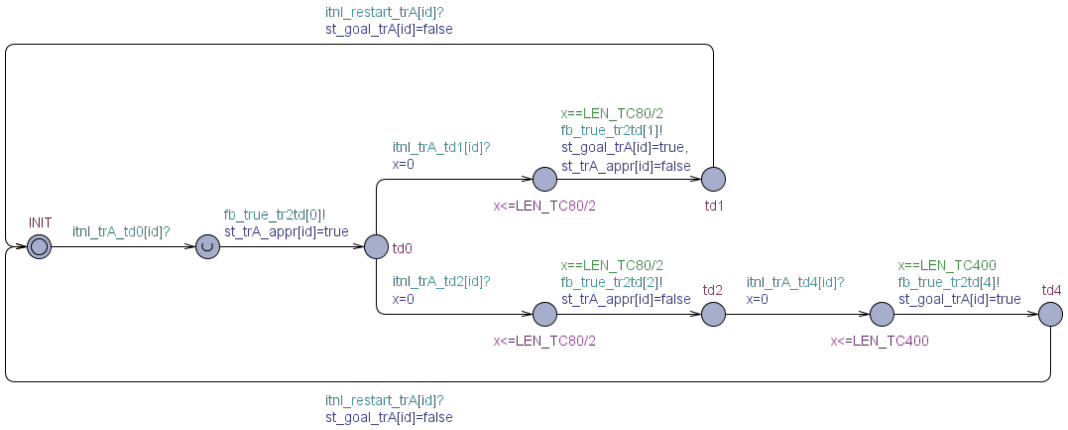

図 72　列車 (TrainA) における F_{out} 型オートマトン

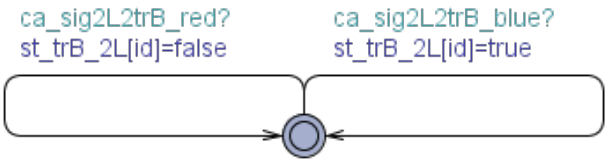

図 73　列車 (TrainB) における C_{in} 型オートマトン

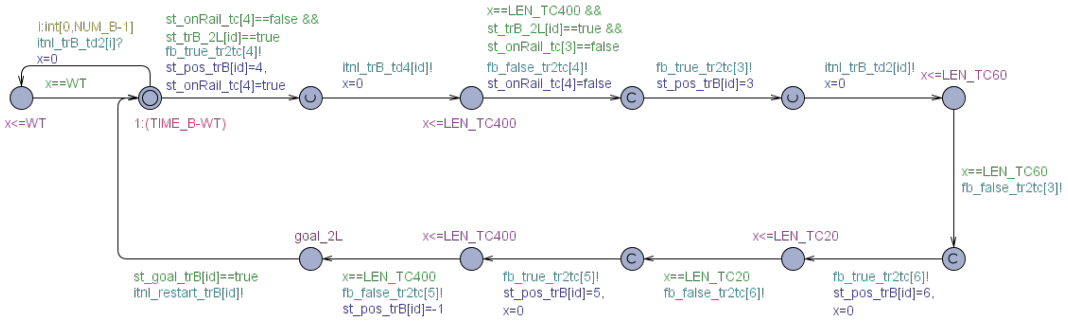

図 74　列車 (TrainB) における F_{out} 型オートマトン (1)

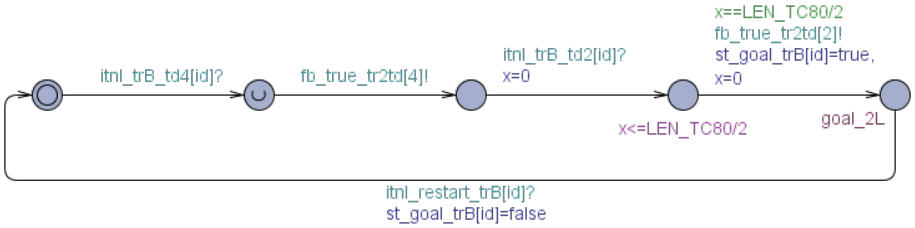

図 75　列車 (TrainB) における F_{out} 型オートマトン (2)

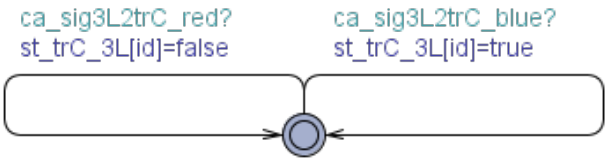

図 76　列車 (TrainC) における C_{in} 型オートマトン

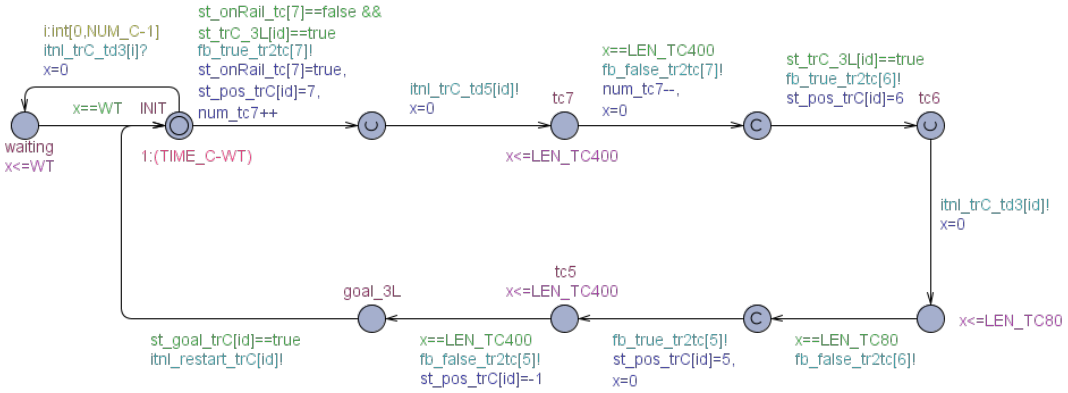

図 77　列車 (TrainC) における F_{out} 型オートマトン (1)

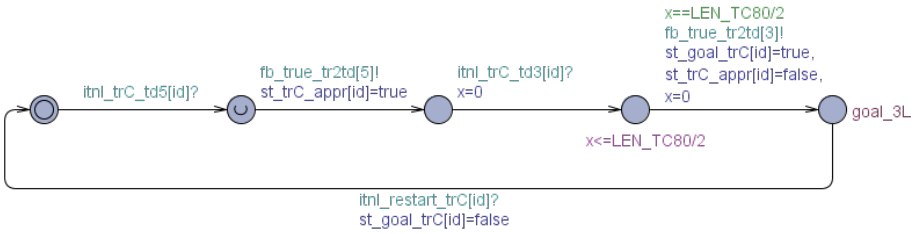

図 78　列車 (TrainC) における F_{out} 型オートマトン (2)

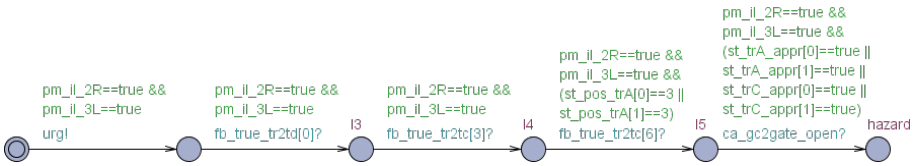

図 79　シナリオ (HS2) に対応するシナリオ観測用モデル

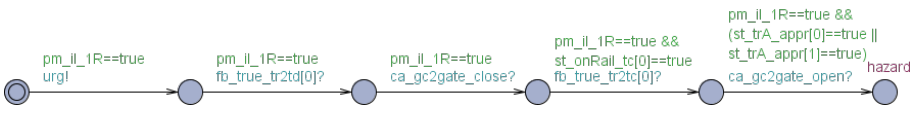

図 80　シナリオ (HS3) に対応するシナリオ観測用モデル (1)

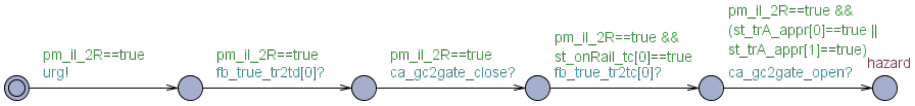

図 81　シナリオ (HS3) に対応するシナリオ観測用モデル (2)

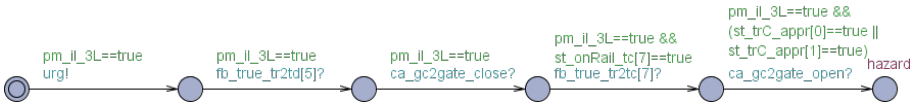

図 82　シナリオ (HS3) に対応するシナリオ観測用モデル (3)

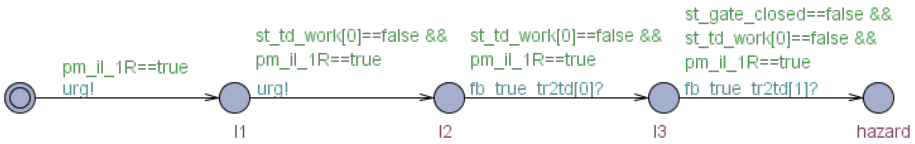

図 83　シナリオ (HS4) に対応するシナリオ観測用モデル (1)

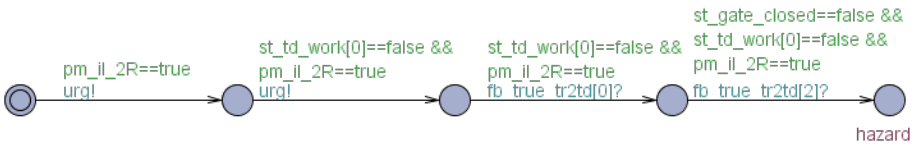

図 84　シナリオ (HS4) に対応するシナリオ観測用モデル (2)

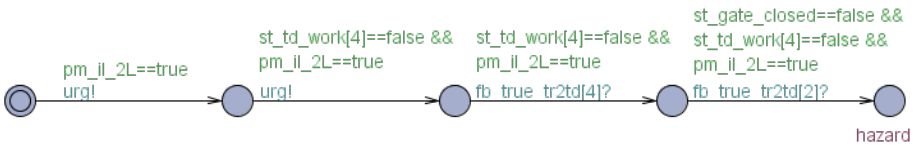

図 85　シナリオ (HS4) に対応するシナリオ観測用モデル (3)

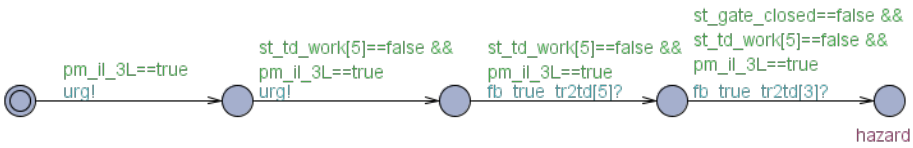

図 86　シナリオ (HS4) に対応するシナリオ観測用モデル (4)