STAMP/STPA によるハザード分析のモデル検査を用いた支援

青木善貴† 福島祐子†

概要：STAMP/STPA は有効なハザード分析のツールである．ただし，分析者の発想に重点を置いた手法であるため，網羅的に検証する必要がある制御アルゴリズムのハザード分析については，難しい面がある．こうした部分を，形式手法の一つであるモデル検査を用いることにより，支援することができると考える．

キーワード：STAMP，STPA，モデル検査，ハザード分析

Support of Hazard Analysis by STAMP / STPA using Model Checking

YOSHITAKA AOKI† YUKO FUKUSHIMA†

Abstract:

Keywords:

1. はじめに

近年，IoT(Internet of Things)や IoT を組み込んだ CPS(Cyber Physical System)は様々な分野において活用が進んでいる．今後多くのシステムが何らかの形で IoT もしくは CPS を包含することになると思われる（以降 IoT・CPS システムと呼ぶ）．IoT・CPS システムは，外部環境へ物理的に影響を及ぼす可能性があるため安全性の確保は重要である．

しかし，IoT・CPS システムは環境情報をセンシングして振る舞いを決定するため振る舞いが複雑であることや，IoT・CPS システムが持つハイブリッド性(離散と連続の異なる性質をもつ)と不確かさ(時間的な揺らぎ／要求の予測困難)の問題[1]により安全性の検査は難しい．この問題は，昨年度ドローンの飛行の安全性を検査[2][3]した際に，センシングデータの分析結果を飛行の物理的動作に変換する振る舞いの整理・抽出が難しいことからも確認している．

IoT・CPS システムの安全性の検査には，起きてはいけないこと（アクシデント）を基にして，リスクを識別するハザード分析が有効であると仮定する．複雑なシステムの安全性解析には STAMP (Systems Theoretic Accident Model and Processes)の利用が期待されている．STAMP はシステム理論に基づく事故モデルであり，システムを環境も含めたフィードバック制御ループと捉え，そこから非安全な制御を見つけることによりハザード分析を行う. STAMP は動的な外部環境も事故モデルに含むため，システムの構成要素間の静的な関係である故障確率から安全性を検討する FTA（Fault Tree Analysis）や FEMA（Failure Mode and Effect Analysis）より，外部環境の影響が大きい IoT・CPS システムの分析に適している．

STAMP は事故モデルであり，本稿では，事前にシステムのアクシデントを見つけることを目的としているため，STPA（System-Theoretic Process Analysis）を具体的な分析手法の対象とする．

独立行政法人 情報処理推進機構（IPA）において，STAMP/STPA に関する資料の公開が行われている．我々はそれらの資料の例題を基に，簡易な IoT システムに対して STAMP/STPA の適用を試みたが，前提となる知識および技術の不足などからハザードを特定するまでに，何度も試行錯誤を繰り返さなければならなかった．

本稿はその適用時に気づいた，STAMP/STPA 適用時の課題の中から，STPA が得意としない「アルゴリズムの網羅的な検証」の部分に着目して，ハザード分析のより確実な実施を支援する手法を提案する．

網羅的な検証を行うには，形式手法の一つであるモデル検査が有効と考え，本提案手法では，ハザード分析の支援にモデル検査[4]を利用する．

IoT・CPS システムの安全性に着目している研究は少なく，検証できる方法を示すことにより，安全な IoT・CPS システムの構築に役立つと考える．

† 日本ユニシス株式会社
Nihon Unisys ,Ltd.

2. STAMP/STPA の概要と課題

2.1 STAMP/STPA の成り立ち

STAMP は，MIT の Nancy Leveson 教授が提唱したシステム理論に基づく事故モデルである．システム理論では，システムを個別の要素ごとに捉えるのではなく，相互作用する一まとまりの要素全体として捉える. STAMP はこのシステム理論に基づき,「安全」は個別の要素から出てくるものではなく要素の相互作用の中から現れるものとし,「事故」は相互作用が安全制約（システムが安全に保たれるために必要なルール）に違反したときに起きるとしている．
STAMP は, 安全性解析のための新たなモデルとして注目されている．従来の安全性解析手法（HAZOP ，FMEA など）では，数多くの構成要素同士が複雑に連携したシステムの解析には対応しきれない．従来の手法はシステムが単純な構成であった 1960 年代に作成されたもので，「事故は故障イベントの連鎖によって起こる」と定義されていた．しかし現在では個々の構成要素が正常に動作していてもシステム全体としては事故が起きる可能性がある．このような事故に対応するために，STAMP では「事故は構成要素の相互作用が安全制約に違反した場合に起きる」としている．たとえば，安全制約「電車が駅に到着するまでドアが開かない」に違反して「電車が駅に到着する前にドアが開く」ときに事故が起きるということである．従来の手法の「障害を防ぐ」という目的だけではなく，システム全体の振る舞いにおいて「安全制約を確実に守ることにより事故を防ぐ」ことが STAMP の目的である．

STPA は，STAMP モデルをベースにした具体的なハザード分析手法である．トップダウンアプローチであるため，システム開発の早い時期に適用することにより，安全性に対するハイレベルな要求を作成し，システムデザイン，個別のコンポーネントへより詳細な安全性に対する要求として反映することができる．

STPA は従来の手法と比較すると，特にソフトウェア，システムデザイン，人の振舞いに関係する原因要因やハザードシナリオを特定する上で有効である．また事故の要因を分析するための適切なガイドワードが提供されているため，対象ドメインの専門家ではなくても事故シナリオの発見が容易である．

2.2 STAMP における事故モデル

STAMP では, システムをさまざまな階層から成るコントロールストラクチャとして捉える．その目的は，安全制約を確実にし，損失を防ぐことである．

コントロールストラクチャを図 1 に示す．コントロールストラクチャでは，高レベルのコントローラから低レベルの被コントロールプロセスに対して安全性確保のためのコントロールアクション（制御指示）が実行される．また，被コントロールプロセスからコントローラへのフィードバックがなされることにより，階層間でコントロールループが形成される．

コントローラは，コントロールアルゴリズムによりコントロールアクションを決定するが，決定の際に自分が持つプロセスモデル（現在のシステムの状態）を使用して判断する．コントローラは，自動的なコンポーネントや人を想定している．

コントロールループが適切に作用することによりシステムの安全が守られる．反対に事故が起こるのは具体的には，コントローラから被コントロールプロセスへ適切なコントロールアクションが行われないためである．そして，適切なコントロールアクションが行われない原因としては, コントローラ自身が認識しているプロセスモデルが，実際のシステムの状態を正しく反映できていないこと（例：実際にはドアが開いているのに，閉まっていると認識している，など）が主な原因である．つまり，コントローラや被コントロールプロセスが故障していない場合でも，コントローラの認識の不整合により安全でないコントロールアクション（Unsafe Control Action :UCA）が実行され，最終的に事故につながるという考え方である．

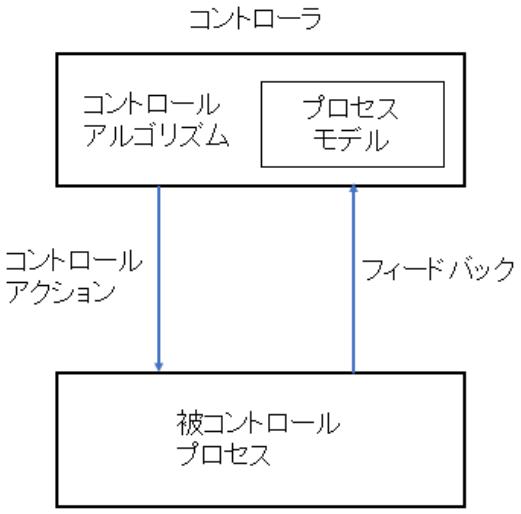

図 1 コントロールストラクチャの例

2.3 ハザード分析の進め方

ハザード分析は，安全性解析手法である STPA に基づき以下の手順で行う．

- Step0 準備 1 ：アクシデント，ハザード，安全制約の識別
- Step0 準備 2 ：コントロールストラクチャの構築
- Step1 ：安全でないコントロールアクションの抽出
- Step2 ：安全でないコントロールアクションが行われる原因であるハザード要因（Hazard Causal factor : HCF）の特定

2.4 ガイドワードによる HCF の特定

Step2 では，以下の 11 個のガイドワードを用いて分析を行うことにより HCF を特定する[5][6].

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

2.5 課題

独立行政法人 情報処理推進機構（IPA）において，STAMP/STPA の普及を図るために，資料の公開などが行われている．それらを基に実際に適用しようとしたが，どのように分析を進めればよいか分からず戸惑うことが多かった．以下の項目が，適用するに当たっての課題と考える．

- 適用事例が少ない
- 分析者のドメイン知識及び発想力への依存が大きい
- コントロールアルゴリズムやプロセスモデルの分析が難しく，関係する HCF の発見が困難である

2.2 節で述べたように，主な事故要因としては，コントローラが不正確なプロセスモデルや不適切なコントロールアルゴリズムに基づいて，コントロールアクションを実行することが考えられる．本稿では，この部分の分析を支援する方法について提案する．

コントロールアルゴリズムとプロセスモデルに関係するものは，2.4 節の（2）と（3）である．上記のガイドワードを，コントロールストラクチャ上に配置すると図 2 になる．（2）及び（3）は，コントローラ内にあり，外部からの入力を基に判断を下し，被コントロールプロセスに対して影響を与えることがわかる．

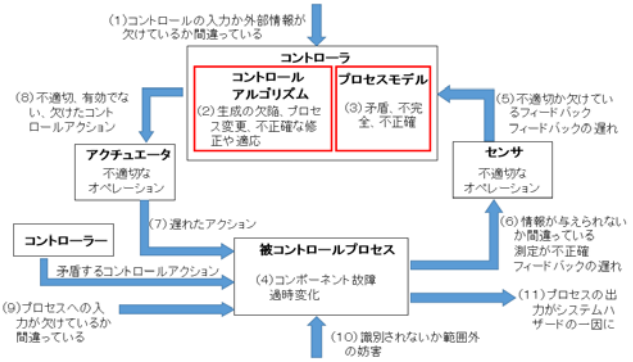

図 2 ガイドワード対応したコントロールストラクチャ

2.6 コントロールアルゴリズム・プロセスモデルの検証の難しさ

コントロールアルゴリズムは，コントローラを制御する手続きであり，プロセスモデルは，コントローラが想定する被コントロールプロセスが取り得る状態である．

これらに誤りがあると，入力情報や構成機器が正常であっても，アクシデントが発生する可能性がある．従って，コントロールアルゴリズムやプロセスモデルに誤りがないことを確認するには，想定できる入力情報や被コントロールプロセスの状態を，コントロールアルゴリズムとプロセスモデルに当てはめて，システムとして問題がないことを検証しなければならない．こういったプロセスモデルがもつ状態を網羅的に検証する方法は，STPA には定義されていない．分析者が独自に行うためハザード分析を漏れなく確実に行うことは難しいと考える．

3. 提案手法

2.6 節で述べた課題に対し，本稿では，形式手法の一つであるモデル検査を利用して，人手による作業では難しい，コントロールアルゴリズムとプロセスモデルのハザード分析を支援する手法を提案する．

STPA の Step0 および Step1 は実施済みであることが前提である．コントロールストラクチャについては，初期に作成した抽象度の高いものを使用する．抽象度が低いものしかない場合は，検証用に抽象度が高いものを再度作成する．

3.1 モデル検査の適用について

モデル検査は形式手法の一つである．モデル検査は網羅的に状態を検索し，システムが一定の状態を保つことを保証する手段の一つである．想定外のケースを探索することに向いている．

ただし，モデル検査では，予め検証対象となる状態を全て検査モデルとして定義する必要がある．そのため，検査対象のシステムの振る舞いが多すぎると，定義の対象になる状態が膨大になり，検査モデルの作成が困難になる．従

って，システムの振る舞いが非常に多くて特定がしにくいIoT･CPS システムから適切な状態を選択してモデル検査を行うことは難しいという課題がある．

状態数を減らすために抽象化などの手段もあるが，そのためには削減する状態を定める視点が必要である．STAMPのプロセスモデルは，非コントロールプロセスの状態を限定してくれるため，その状態を選択して検査モデルを作成すれば，モデル検査を行うことが可能になると考える．

3.2 提案手法の概要

本提案手法は以下の手順で行う．

①コントロールストラクチャを作成する
②各コンポーネントの状態を想定する
③各コンポーネントの状態についてのモデル図（状態遷移図）を作成する
④コントローラのコントロールアルゴリズムとプロセスモデルからモデル図（状態遷移図）を作成する
⑤モデル図に不足している機能があれば追加する
⑥検査モデル図を作成する
⑦モデル検査を実施する
⑧不具合が発見されれば，コントロールストラクチャに反映し，不具合が無くなるまで繰り返す

3.3 コントロールストラクチャ作成（5.2 ①）

コンポーネントが四つ程度の，抽象度の高いコントロールストラクチャを使用する．この理由は，まずコントローラと非コントロールプロセスの関係を明確にすることが重要だからである．抽象度が高すぎる場合は，モデル図作成の過程で不明な点がでてくるので,モデルの詳細化を行い，コントロールストラクチャに反映させる．

また，モデル図作成の過程で複数のコントローラがあると判明した場合は，個々にコントロールストラクチャを構成して連携させる．コントロールストラクチャには確定的な記述方法はないため，分析しやすいものを作成すればよい．

3.4 コンポーネントの状態（3.2 ②③）

想定したアクシデントに関係すると思われる各コンポーネントが持ち得る状態を想定する．関係性の無い状態遷移がある場合は，検査モデルは個別に分けて作成する．

3.5 コントロールアルゴリズムとプロセスモデル（3.2 ④）

コントローラが認識している非コントロールプロセスの状態を，コントロールアルゴリズムに従って制御するモデルを作成する．

3.6 不足している機能・状態の追加（3.2 ⑤）

5.5 で作成したモデルと 5.4 で想定した状態に齟齬がある可能性がある．もし齟齬があった場合には，状態の変化を起こす機能をモデル上に追加して整合性をとる．

同様にモデルを動作させるにあたり，不足していると思われる状態があれば，これも追加する．

3.7 検査モデルの構築（3.2 ⑥）

5.4～5.6 で作成したモデル図を統合して検査モデルを作成する．構成の方法は使用するモデル検査ツールにより異なる．本稿は，特定のモデル検査のツールを対象としたものではない．

3.8 検査方法（3.2 ⑦）

アクシデントが発生する状態を検証する．モデル検査の検査式は，アクシデント及び安全制約を基に作成する.検査式を実行して，想定と違う結果が出力された場合，それを基に分析を行い，HCF を特定する．ただし，複数の HCF の存在が想定されるため, 発見した HCF に対策を施したモデルで再度検証を行う必要がある．

3.9 繰り返しについて（3.2 ⑧）

修正した検査モデルの内容は，コントロールストラクチャに反映して再度検証を行う.問題が無くなるまで提案手法を繰り返す.

4. 適用事例

4.1 適用事例概要

IPA が公開している資料「はじめての STAMP/STPA～システム思考に基づく新しい安全性解析手法～」[5]にある分析実施例「単線の駅中間踏切制御装置」を基にした適用事例である．

図 3 の「センサ A」および「センサ B」は警報開始センサである．「センサ C」は警報終止センサである．警報終止センサは踏切の近くに設置される．列車通過後一定時間経過後に警報を止める．本事例では，列車の進行方向は駅 1 から駅 2 へ向かう一方向に限定する．これは提案手法を適応する事例として，シンプルで分かりやすいものにするためである．

図 3 単線の駅中間踏切制御システム概要

要求される仕様は以下のとおりである．

- 警報開始センサ A，B は列車を検知すると踏切を鳴動させる．
- 警報終止センサ C は列車を検知すると踏切の鳴動を停止させる．
- 警報終止センサ C を通過した時点で，警報開始センサ B をマスクする．
- 警報開始センサ B を通過した時点で，センサ B のマスクを解除する．

4.2 想定するアクシデント，安全制約

想定されるアクシデントは，「列車と人もしくは車などが踏切内で衝突する」である．ハザードは，「列車が通過中にも関わらず警報が鳴動していない」となる．従って安全制約は「列車が通過中は警報が常に鳴動している」である．

4.3 コントロールストラクチャの作成

三つのコンポーネントでコントロールストラクチャ（図 4）を作成した．コントローラは「踏切制御装置」であり，非コントロールプロセスが「センサ ABC」，「警報機・遮断機」である．「列車」はセンサに対する入力となる．

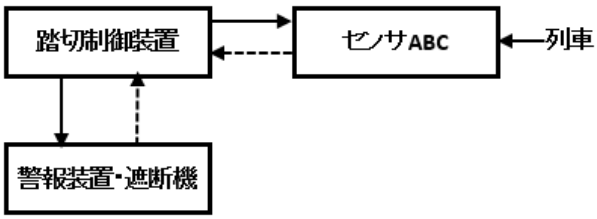

図 4 コントロールストラクチャ

4.4 コンポーネント取りうる状態のモデルの作成

上記のコントロールストラクチャの各コンポーネントがもつ状態をハザード分析の視点で抽出し，状態遷移図を作成する．状態は 4.2 節で想定したアクシデント及び安全制約を基に作成する．本稿では，検査モデルの意図が明確に伝わるようにステートマシン図によりモデルを表記する．

・踏切制御装置

コントローラなので対象外．

・警報機・遮断機

取り得る状態は，「鳴動している（遮断機が下がっている）」，「鳴動していない（遮断機があがっている）」の二つである．この二つの状態を交互に遷移する（図 5）．

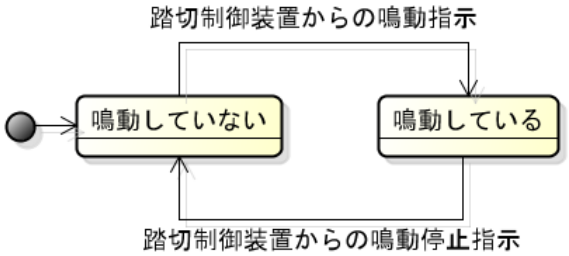

図 5 警報機・遮断機のモデル

・警報開始センサ A，B

警報開始センサ A，B が取り得る状態は，「範囲外」「通過中」「通過済」，「マスクされている」，「マスクされていない」である（図 6，図 7）．

マスクの状態と通過の状態は，状態遷移につながりはないのでモデル図は個別に作成する．センサ A，B それぞれ通過状態のモデルとマスク状態のモデルの二つをもつことになる．

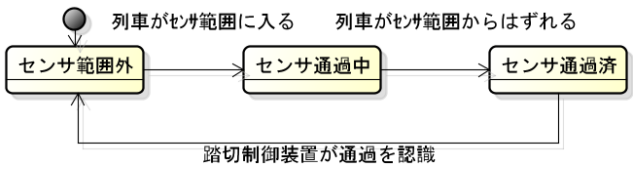

図 6 警報開始センサ A，B の通過状態のモデル

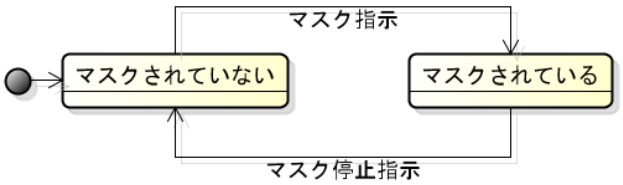

図 7 センサのマスクのモデル

・警報終止センサ C

警報終止センサ C が，取り得る状態は「範囲外」「通過中」「通過済」である（図 8）．センサ C はマスクのモデルはもたない．

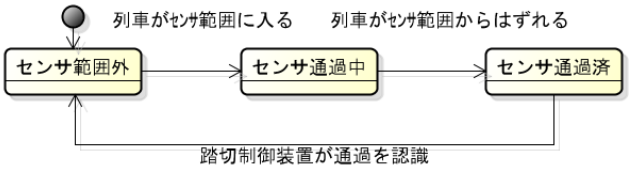

図 8 警報終止センサ C の通過状態のモデル

・列車

列車については，非コントロールプロセスではないが，環境情報としてモデル化する．

取り得る状態は，列車の位置情報である．「駅 1 通過中」，

「センサ A 通過中」,「踏切通過中」,「センサ C 通過中」,「センサ B 通過中」,「駅 1 とセンサ A の間」,「センサ A と踏切の間」,「踏切とセンサ C の間」,「センサ C とセンサ B の間」,「センサ B と駅 2 の間」の状態をとる（図 9）.

状態は，常に順番通りに遷移し，戻ることはない．複数の列車が走る場合は，当モデルを複数用意することとし，列車の追い越しは起きないものとする.

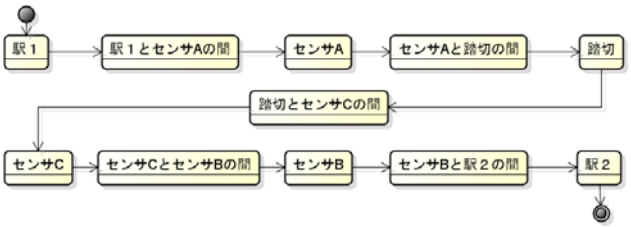

図 9 列車の位置のモデル

4.5 コントロールアルゴリズムとプロセスモデルの検討

コントロールアルゴリズムとプロセスモデルから，センサの状態と警報機・遮断機の状態を判断して，コントロールするモデルを作成する（図 10）．プロセスモデルを判断し続ける動作となるため繰り返し処理になる．列車進行方向を一方向としたモデルであるが，条件分岐を追加することにより進行方向が逆の場合にも対応可能なモデルにできる.

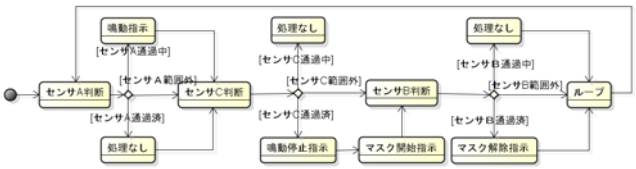

図 10 プロセスモデルのモデル

4.6 モデル検査の実施

4.6.1 想定される正常動作に関する検査

まず，検査モデルが想定した振る舞いを行うことの確認を行う．確認内容は以下の通りである.

- デッドロックしないこと
- 列車が駅 2 まで到達できること
- 警報開始センサ B がマスクされ，解除されること
- 警報機・遮断機が鳴動し，停止すること

検査を行ったが，問題は発見されなかった.

4.6.2 安全制約に関する検査

想定した安全性制約を基に検証を行う.「列車が通過中は警報が常に鳴っている」が遵守されていることを検査する．モデル検査では検査したい性質を検査式として与えて検査を行う．検査式の内容は「列車が通過中に警報機が鳴動してないことは決していない」となる.

検査式は次の通りである.

G┬(列車.踏切通過中∧警報機・遮断機.鳴動していない)

列車が一両の場合，検査結果は，「満たされる」となり，問題がないことが示された.

列車が二両の場合，検査結果は，「満たされない」となり，列車が通過中に警報が鳴動していない状態があることが判明した.

4.7 検証結果

検査式が満たされなかったためモデル検査ツールは反例を出力する．反例は満たされない状態に至る状態遷移の過程を示すものである．この反例を解析して HCF を特定する．反例には，最終的な状態に至るまでの全ての状態遷移が記録されるため，不要なデータが多い．特徴的な状態のみ選び出してグラフ化を行い，状態の遷移の動向をつかみやすくする.

グラフ化の対象にする特徴的な状態は次の通りである.

- 警報機・遮断機：「鳴動している」「鳴動していない」
- 列車：二両の位置の状態

上記の状態をグラフ化したものが図 11 である．縦軸は状態に対して任意に付けた番号である．横軸は反例のレコード番号である．状態と任意の番号の対応は表 1 に表わす.

先行する列車を列車 1，後続の列車を列車 2 とすると，列車 1 がセンサ A を通過し始めると「鳴動している」状態になる(図 11 横軸値:14)．列車 2 がセンサ A に到達してもすでに鳴動中であるため変化はない（図 11 横軸値:20）．次に列車 2 が踏切を通過中に，列車 1 がセンサ C を通過完了すると鳴動が停止されてしまうことが分かる（図 11 横軸:58）.「列車が通過中にも関わらず警報が鳴動していない」状態になることが確認できた.

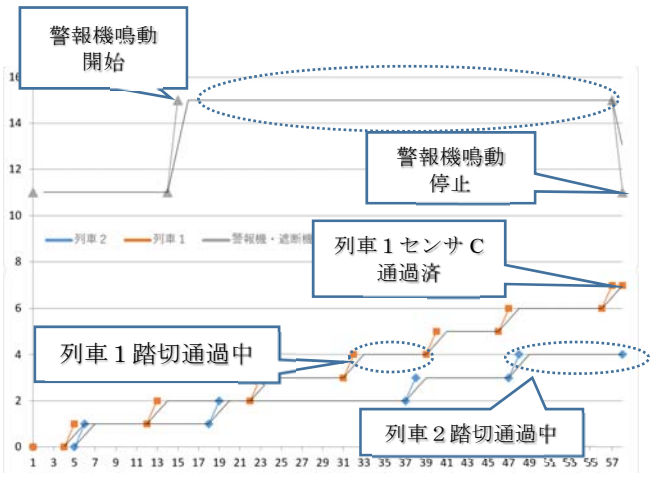

図 11 モデル検査結果

HCF は,「接近した状態で二両の列車が運行されると，片方が踏切を通過中に，もう片方の列車が警報を解除する」である.

表 1 グラフ縦軸値と状態

縦軸番号	状態の内容
0	駅 1
1	駅 1 センサ A の間
2	センサ A 通過中
3	センサ A と踏切の間
4	踏切通過中
5	踏切とセンサ C の間
6	センサ C 通過中
7	センサ C とセンサ B の間
11	鳴動していない
15	鳴動している

適用事例の参考とした，IPA の「はじめての STAMP」[5]の分析実施例において，アルゴリズムに関するハザード分析の中に，今回検出した HCF の指摘はなかった．別の章において，STPA とは別の状態変数を使う手法の紹介では，当該 HCF は指摘されている．STPA 自体は，アルゴリズムを基に網羅的に状態を確認する検証には向いていないと考えられる.

5. 関連研究

畑ら[7]は，STAMP/STPA で分析した結果を形式仕様記述で検証する方法を提案している．これは，分析結果に保証を与えるという点で重要と考える．しかし，本研究の網羅的にシステムの状態・ロジックを検証してハザードを見つける支援を行うものではない.

Asim ら[8]らは，STPA にはプロセスモデルとコントロールアクションとアクシデントの関係を表現する表記法が必要と主張している．彼らはコントローラもしくはコントロールアクションの状態と，アクシデントの状態の関係をステートマシン図で表わす表記法を提案している．コントローラのプロセスモデルを捉えて，状態のモデルに置き換える考え方は本研究と同じであるが，網羅的に検証して，検証結果から HCF を導く方法は提示されていない.

IPA の「はじめての STAMP」[5]において，状態変数を使う手法による分析が紹介されている．6.7 で指摘した HCF と同じ内容を指摘している．ただしこの手法は，表を用いて状態を整理して行うもので，分析対象が複雑になってくると分析が難しくなってくると思われる.

6. 考察，議論

6.1 IoT・CPS システムの品質特性について

システムの品質を表すものとして，JIS X25010 「システム及びソフトウェア製品の品質要求及び評価（SQuaRE）−システム及びソフトウェア品質モデル」があり，機能適合性，性能効率性などが定義されている．IoT・CPS システムもこれらの品質特性を満たすべきである.

ただし，IoT・CPS システムは，物理的なデバイスが関係するため,「ハイブリッド性」や「不確かさ」の影響が大きいと考えられ，JIS X 25010 による評価だけでは不十分であり，IoT・CPS システムの適切な評価を可能にするための評価基準やモデルリング手法が必要と考える.

6.2 STAMP における離散系と連続系について

ソフトウェア開発におけるモデルには，離散モデル（情報システム向き）と連続系のモデル（組み込み向き）がある. 離散系のモデルを表記するものとしては, UML（Unified Modeling Language）や SysML(Systems Modeling Language)があり，モデル検査による検証が可能なモデルである．一方，連続系のシステムを表記するものとしては MATLAB/Simulink がある．今後 IoT・CPS システムの様な異なる分野が連携するシステムでは, STAMP によるシステムのハザード分析が適していると思われる[9].

2.4 節で 11 個のガイドワードを示した．分析対象のシステムが IoT・CPS システムと仮定した場合，ガイドワードに対応したコンポーネントの属性を想定することにより，ガイドワードの属性が，連続系か離散系に分類できる（表 2）．両方に関係すると思われる場合は，離散系＋連続系にも分類する.

表 2 ガイドワードの連続系／離散系の分類

CPS 分類	属性	ガイドワード
Cyber	離散系	(1)(2)(3)
Physical	連続系	(1)(4)(6)(7)(9) (10) (11)
Cyber-Physical	離散系＋連続系	(1)(3)(5)(8)

今回，分析の対象にしたガイドワード(2)(3)は主に離散系のモデルであるため，モデル検査での検証は充分可能である．ガイドワード(1)も離散系と考えられるため，本提案手法が適用できるものと考える.

また，連続系に属するガイドワードについては連続系のモデル化のツールである MATLAB/Simulink などでの分析が有効ではないかと考える.

6.3 提案手法の STAMP/STPA への親和性について

STAMP/STPA は，動的に変化する環境下での安全性を保証するための分析に適している．ただし，具体的な分析法である STPA だけでは全ての局面に対応するのは難しい．分析対象に適した分析技術で支援すべきであるが，STAMP/STPA とは全く異なる表記法で分析を行うと，コントロールストラクチャの変更などへの対応が煩雑になるが，

本研究ではコントロールストラクチャを基に検証を行うため，親和性が高く，柔軟に対応できる．

6.4 提案手法の STAMP/STPA 以外への利用について

STAMP/STPA のコントロールストラクチャの考え方は，既存の UML などで表記されたモデルを，抽象化して捉えることに利用できるのではないかと考える．

既存のモデルを抽象度の高いコントロールストラクチャへ変換し，それに本提案手法を適用すれば，IoT・CPS システムの一部の安全性が検証可能になると考える．

7. まとめ，今後の展望

本稿では，状態の網羅的な検証を必要とするような HCF の発見について，STAMP/STPA を利用する分析者に対してモデル検査を適用することでハザード分析を支援する手法の提案を行った．

アクシデントを想定し，コントロールストラクチャを作って，コンポーネントがもつ状態を作れることにより，振る舞いの予測が困難な IoT・CPS システムについても，モデル検査で一定の範囲で検証できることを示した．

検査結果（反例）の解析方法についてはまだ充分検討しきれていない部分がある．直感的に理解できる表示について今後さらに検討を行いたい．

参考文献

[1] “ソフトウェアは工学の対象システムか工学的な手段か”, http://researchmap.jp/mu2paxn9j-1274/#_1274,(参照 2016-07-31).
[2] 細金万智子,青木善貴,星野隆之,モデル検査によるドローンの安全確認 , エンタテインメントコンピューティングシンポジウム 2015 論文集,2015,pp.267-273.
[3] 青木善貴,細金万智子,モデル検査によるドローンの安全確認，第 22 回 ソフトウェア工学の基礎ワークショップ FOSE2015, pp.227-228, 2015.
[4] 形式手法の実践ポータル, http://formal.mri.co.jp/db/fmtool/,(参照 2016-07-31).
[5] “はじめての STAMP/STPA ～システム思考に基づく新しい安全性解析手法～Ver1.0”, http://www.ipa.go.jp/files/000051829.pdf , (参照 2016-07-31).
[6] “STAMP 手法に関する調査報告書”, https://www.ipa.go.jp/files/000047911.pdf , (参照 2016-07-31).
[7] 畑彰拓，日下部茂，林信弘，モデル指向形式仕様記述におけるハザード解析法 STAMP/STPA の活用，SES2014，pp.33-38，2014.
[8] Asim Abdulkhaleq, Stefan Wagner, Integrating State Machine Analysis with System-Theoretic Process Analysis, Software Engineering (Workshops), pp.501-514,2013.
[9] “共通理解フレームワークとしてのサイバー・フィジカル・システムズ”,http://researchmap.jp/muj94vr0w-1274/#_1274,(参照 2016-07-31).