

影響分析のための STAMP/STPA への動的コナーセンスの導入

谷口 智哉
九州工業大学

taniguchi.tomoya132@mail.kyutech.jp

日下部 茂
九州工業大学

kusakabe@csn.kyutech.ac.jp

要旨

システムの大規模化・複雑化に伴い、派生開発などでの影響分析の難度が上昇している。大規模化や複雑化したシステムでの影響分析では、コードレベルでの影響範囲の特定に加え、システムレベルでの影響範囲の特定を行う必要がある。システムレベルの分析をトップダウンに行う手法の一つとして、STAMP/STPA がある。しかし STAMP/STPA に基づく依存関係の特定において、静的な依存関係はそのコントロールストラクチャ上で明確であるが、動的な依存関係の特定は自明ではない。本研究では、そのような STAMP/STPA での動的依存関係の分析に動的コナーセンス(実行順序, タイミング, アイデンティティ, 値)の概念を導入し、コントロールストラクチャ上に可視化する手法を提案する。提案手法を事例に適用し、従来の STAMP/STPA 単独では分析者の見落としやスキル不足によって見落とされていた可能性のある影響関係を識別可能であることを示した。

1. はじめに

派生開発では、変更や追加機能の実装に伴う影響分析は重要である。しかし、システムの大規模化・複雑化により、変更による影響範囲の特定がより難しくなっている。影響分析が適切に行われなかった場合、欠陥が入る可能性が高まる上に、過剰な検証や検証不足が発生しやすくなり、開発コストの増大やシステムの安全性低下といった問題を引き起こす可能性がある。

影響分析には、コードレベルでの変更の影響を分析する手法が活用されており、関数間の依存関係やデータフローの変化を特定することが可能である。しかし、これらは主にソースコードやモジュール単位の解析に焦点を当てており、システム全体の動作や上流工程での影響を考慮するものではない。システム全体の影響を考慮するためには、より抽象度の高いレベルから影響分析を行う必要がある。特に、上流工程における影響分析が適切に行われなければ設計や要求仕様の変更がもたらす影響を正しく把握できず、不要な修正や不十分な検証を引き起こす可能性がある。そのため、システム全体の構造や相互作用を考慮した影響分析の手法が求められている。

事故因果関係モデルとその安全性解析手法である STAMP/STPA (System-Theoretic Accident Model and Processes/ System-Theoretic Process Analysis)[1]はトップダウンの解析手法であり、システム全体の相互作用を考慮した安全性分析が可能である。そのため、大規模・複雑化したシステムにおける、過剰または不足した検証の問題をトップダウンに分析する手法としても期待される。

しかしながら、STAMP/STPA に基づいて依存関係を分析するとき、コントロールストラクチャ上でコントロールアクションとフィードバックによる静的な依存関係は明確であるものの、動的な依存関係は必ずしも明確でない上に確立したその分析法もないという課題がある。従来の STAMP/STPA での動的な依存関係の分析は、ガイドワードの利用などをベースにしたシナリオの展開でなされるが、その詳細は分析者に委ねられており、属人性が高いという課題がある。

本研究では、このような動的な依存関係の課題に対して STAMP/STPA に動的なコナーセンスを導入することで解決する。動的コナーセンスは、システム実行時の呼び出し関係に着目し、動的な依存関係を「実行順序」「タイミング」「アイデンティティ」「値」の4つの観点[2]で分類する手法である。これにより、影響範囲の特定をより精確に行い、従来の STAMP/STPA では暗黙的に処理されていた依存関係の識別を明示的に行うことが可能となる。

本研究を通じて、動的コナーセンスを活用した STAMP/STPA の適用可能性を明らかにし、属人的な判断に依存しない、動的コナーセンスに沿った動的な依存関係の特定手法の確立を目指す。

本論文は、以下のような構成である。第2章では、本研究に関連する既存の影響分析の手法や研究について紹介する。第3章では、STAMP/STPA の概要について説明する。第4章では、STAMP/STPA に動的コナーセンスを導入し、影響範囲の特定を体系的に行う手法を提案する。第5章では、提案手法をネット通販システムの例題に適用し、従来の STAMP/STPA 単独では見落とされる可能性のある影響関係を識別できることを示す。第6章では、研究のまとめと今後の課題について述べる。

2. 関連研究

2.1. 影響分析について

影響分析は、システムの変更が他の要素に与える影響の特定と、その適切な対応に重要な技術である。代表的な分析手法として、コードレベルの解析がある。例えば、依存関係解析や静的解析を用いることで、関数やクラス間の関係を把握し、変更が及ぼす影響範囲を特定する手法がある。これにより、ソースコード内の影響範囲を明確にし、変更による不具合を防ぐことが可能となる。

しかし、コードレベルでの影響分析だけでは、システム全体の設計や要求仕様の変更が及ぼす影響を十分に捉えることは難しい。特に、大規模かつ複雑なシステムでは、設計や要求段階での影響を適切に把握しないと、開発後期の想定外の問題発生リスクが高まる。そのため、上流工程における影響分析の必要性が指摘されている。

このような課題に対応するため、システム設計ドキュメントと変更要求をどのように関連付けるかについての手法が提案されている[3]。提案された手法では、変更要求とシステム設計の関係を分析し、変更が及ぼす影響を体系的に管理する方法について検討している。これにより、上流工程における影響分析の精度を向上させ、変更に伴うリスクを低減することが期待される。

この手法は、変更要求のトレーサビリティに重点を置き、設計ドキュメントのどの部分が影響を受けるのかを明確化することで、設計変更の影響範囲を特定することを目的としている。しかし、このアプローチでは主に静的な依存関係に基づく影響分析が中心であり、動的な依存関係を明示的に考慮する手法は示されていない。また、変更要求と設計要素の対応付けが適切に行われない場合、適切な変更箇所を特定を正確に行えない可能性がある。

本研究では、STAMP/STPA を活用した影響分析の支援を目的として、動的コナーセンスの概念を導入し、動的な依存関係を明示的に可視化する手法を提案する。関連研究が変更要求と設計の関係を静的に整理するのに対し、本研究はシステムの動作時に生じる影響関係を分析し、より詳細な影響分析を可能にする点で異なる。

従来の研究では、影響分析は変更要求の管理や設計要素のトレーサビリティに依存することが多い。本研究では、STAMP/STPA によるシステム安全性解析の枠組みを活用し、動的コナーセンスを導入することで、影響範囲の特定をより体系的に行う方法を提案する。このアプローチにより、設計変更時の影響範囲の特定の精度を向上させ、影響範囲をより正確に可視化することが可能になる。

2.2. コナーセンス

コナーセンスは、1996年に Meilir Page-Jones が提唱した概念で[2]、システム内のコンポーネント間の結びつきを評価するための尺度を提供する。システム全体の正しさを維持するために、あるコンポーネントの変更が別のコンポーネントの変更を必要とする場合、これらのコンポーネントはコナーセント(接続)されていると定義される。

コナーセンスには、大きく静的コナーセンスと動的コナーセンスの2種類がある。静的なコナーセンスはコードレベルでの結びつきを表し、主にソースコードの解析を通じて評価される。例えば、複数のコンポーネントが同一の名前を共有する必要がある場合の結びつきの名前コナーセンス、特定のデータ型に依存する結びつきの型のコナーセンス、共有する値の意味に依存する結びつきの意味のコナーセンスがある[2]。

動的コナーセンスは実行時の振る舞いによる結びつきを指し、コンポーネント間の依存関係を評価する[2]。本研究では以下の4つのコナーセンスを扱う。

- 実行順序のコナーセンス: あるコンポーネントの実行が他のコンポーネントの実行との順序がある。
- タイミングのコナーセンス: あるコンポーネントの動作の時間的なタイミングが他のコンポーネントの動作に影響する。
- アイデンティティのコナーセンス: あるコンポーネントの参照するオブジェクトを他のコンポーネントも参照。
- 値のコナーセンス: あるコンポーネントの参照する値を他のコンポーネントも参照し、値が一致していなければ正常に動作しない。

3. STAMP/STPA の概要

STAMP は、Nancy Leveson 教授によって提唱された新しい事故因果関係モデルである。STAMP は安全性を「故障防止の問題」ではなく、動的なコントロールの問題として捉え、システムの振る舞いを制約することで安全性を確保するというアプローチである[1]。STAMP の特徴として、コンポーネント間の非安全な相互作用や動的な関係に注目することやソフトウェア、人間、組織、安全文化など、事故の要因を総合的に扱うことがあげられる。

次に STAMP のハザード分析手法の1つである STPA は、システム全体を俯瞰しながらトップダウンで安全性を分析する。この手法は特にソフトウェア集約型システムや複雑な相互作用を持つシステムにおいて有効である[1]。STPA の利点として、運用でしか発見することができなかった「想定外の想定外」を開発プロセスの初期から識別することができる点が挙げられる。

STPA の分析の手順は STAMP/STPA を提唱した Nancy Leveson 教授によって作成されたハンドブックであ

る「STPA Handbook」[4]や、IPA の発行した「はじめての STAMP/STPA」[5]でまとめられている。本研究では、前者の「STPA Handbook」での手順をもとに説明する。STPA の手順は以下の 4 つのステップで構成される[4]。

1. 解析目的の定義

最初のステップでは、システムの安全性を分析する目的を明確にする。これは STPA の方向性を決定し、分析結果を効果的に活用するために重要である。さらにこのステップでの分析は 3 つに分かれる[4]。

・損失(Loss)の特定

システムにおいて回避すべき望ましくない結果を特定する。損失とは利害関係者にとって受け入れることのできない人間の生命の損失や人間の傷害、物的損害、ミッションの損失などが含まれる。

・システムレベルのハザードの識別

損失を引き起こす可能性のある状態や状況を明確化する。ハザードは損失を防ぐためにコントロールすべき対象となる。システムとはいくつかの共通の目標や目的を達成するために一体として動くコンポーネントの集合である。

・システム制約の定義

システムが安全性を維持するために遵守すべき制約を設定する。システムレベルの制約はハザードを防ぎ最終的に損失を防ぐために必要があるシステムの条件や動作のことである。ハザードが特定できていれば、システムレベルの制約はハザードの状態の逆の状態となる。

2. コントロールストラクチャのモデル化

コントロールストラクチャはコントロールする要素(コントローラー)とコントロールされる要素(コントロールプロセス)の相互作用を示す視覚的なモデルである[4]。これには、システムの重要な構成要素と、それらの間で行われるコントロールとフィードバックの関係が含まれる。

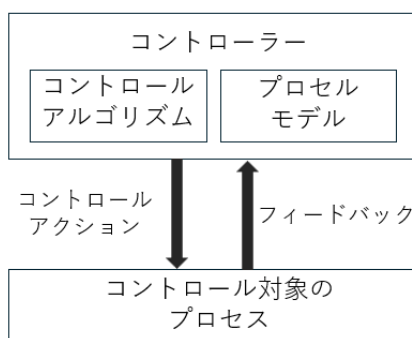


図 3-1 一般的なコントロールストラクチャ

図 3-1 は、一般的なコントロールストラクチャを示している。コントローラーは、コントロール対象のプロセスに対してコントロールアクション(指令)を与えることで、その挙動のコントロールを試みる。コントローラーがどのようなコントロールアクションを出すかは、コントロールアルゴリズムと

呼ばれる意思決定の仕組みによって決定される。

また、コントローラーは、コントロール対象が現在どのような状態にあるかを自ら推定するための内部的なモデル(プロセスモデル)を持っており、そのモデルに基づいてどのようなコントロールアクションを出すかを決定する。このプロセスモデルは、コントロール対象の状態を観測して得られるフィードバックによって、更新される仕組みになっている。

3. 非安全なコントロールアクション(UCA)の特定

コントロールアクションがハザードを引き起こす状況を明確にする。これにより、非安全な状態を防止する方法が見えてくる。コントロールアクションがハザードを引き起こすことにつながるパターンとして以下の 4 つを考える[4]。

- ・ コントロールアクションを与えないことがハザードにつながる
- ・ コントロールアクションを与えることがハザードにつながる
- ・ 潜在的には安全なコントロールアクションを与えるが、早すぎる、遅すぎる、または間違った順序である
- ・ コントロールアクションがあまりにも長く続いているか、早く止まる

4. 損失シナリオの識別

非安全なコントロールアクション(UCA)がどのようにしてハザードを引き起こすのかを詳細に分析し、損失の発生原因を特定する[4]。この分析により、システムの設計段階でリスクを軽減するための対策を講じることができる。

損失シナリオ識別の重要な視点は以下の 2 つである。

1. なぜUCAが発生するのか
コントローラーの意思決定プロセスや、システム内の影響要因(センサの誤動作、データ遅延、ヒューマンエラー)を分析する。
2. なぜコントロールアクションが正しく実行されないのか
システムの構造や外部環境の影響により適切なタイミングでコントロールが行えない可能性を検討する。

システムの構造や外部環境の影響により、適切なタイミングでコントロールが行えない可能性を検討する。損失シナリオを識別する際には、以下のような一般的なパターンを考慮する。

- ・ 不適切なフィードバック
センサの異常や通信遅延により、コントローラーが正しい情報を受け取れない。
- ・ 設計上の制約不足
システム設計時に考慮されなかった状況が実際の運用で問題を引き起こす。

- ・ ヒューマンエラーや誤操作
操作マニュアルの不備や警告表示の誤解により、意図しないアクションが実行される。

損失シナリオを特定することで、システムの設計段階で安全性向上のための制約やフィードバックループの強化が可能となる。これにより、損失の発生リスクを低減し、より安全なシステム設計が実現できる。

4. 動的コナーセンスを導入した STAMP/STPA

4.1. 動的コナーセンスを導入した手法の概要

STAMP/STPA はコントロールアクションとフィードバックを通じた依存関係を表現するが、明示的にコントローラー間の動的な依存関係を記述する仕組みがない。そのため、動的な依存関係を特定し、分析を行うことが系統的でなく、属人的なものであった。なお、動的な依存関係がある部分では、タイミングや順序のずれによって UCA が生じる可能性があるため、STAMP/STPA のスコープ内で扱うことは合理的であると考え。本研究では、それらのような動的な依存関係を、潜在的に UCA が発生する関係として可視化・分析するための枠組みとして動的コナーセンスを導入する。本研究では、動的コナーセンスの概念を導入し、STAMP/STPA のコントロールストラクチャに動的な依存関係を反映することで、影響分析に必要な依存関係の可視化を支援する手法を提案する。

本提案手法では、STAMP/STPA のコントロールストラクチャを拡張し、動的コナーセンスの 4 つの観点(実行順序、タイミング、アイデンティティ、値)を用いて依存関係を特定し、影響範囲を可視化する。

4.2. 動的コナーセンスを導入した手法の手順

本手法は従来の STAMP/STPA でのコントロールストラクチャの構築のあとに以下の 2 つのステップを追加する。コントロールストラクチャの詳細化は従来の STAMP/STPA の手順としてあるが、本手法では、コントロールストラクチャの詳細化は機能レベルまで行っておく必要がある。

1. コントローラー間のコナーセンスを特定する

詳細化したコントローラーについて、動的コナーセンスの観点から依存関係を特定する。

(1) 実行順序のコナーセンス

「あるプロセス A の実行が、別のプロセス B より先に行われる必要がある場合、A→B の実行順序のコナーセンスが成立する。」

以下の 2 点の条件を満たす部分を探す。

- ・システム仕様書に明示的な順序制約がある場合
- ・A の実行が完了しないと B が開始できない

(2) タイミングのコナーセンス

「あるプロセス X が、Y に対して特定の時間内に実行されないとシステムが正常に動作しない場合、X→Y のタイミングのコナーセンスが成立する。」

プロセスやコントロールアクションの遅延時間や応答時間を考慮し、特定の経路に時間制約があるか確認する。

(3) アイデンティティのコナーセンス

「複数のコントローラーが、同じオブジェクト(エンティティ、データ)に対して操作を行う場合、アイデンティティのコナーセンスが成立する。」

どのコントローラーが、どのデータにアクセスするかを確認し、共通するデータにアクセスするコントローラーを探す。

(4) 値のコナーセンス

「複数のコントローラーが、同じ変数やデータ値を参照し、その値が一致しないとシステムが正常に動作しない場合、値のコナーセンスが成立する。」

値の一貫性が必要なデータを特定し、共通するデータにアクセスするコントローラーを探す。

2. コントロールストラクチャ上に反映する

特定した動的コナーセンスによる依存関係を、コントロールストラクチャへ追加し、可視化するために、依存関係のあるコントローラー間を線で結び、どの種類のコナーセンスが存在するかを示すラベルをつける。

この手法を適用することで、従来の STAMP/STPA では属人性の高かった動的な依存関係の分析を系統的に実施した上で、明示的でなかった動的な依存関係を可視化することができるようになる。

5. ネット通販システムへの適用と分析

5.1. ネット通販システムの概要

本研究では、IPA の発行している「はじめての STAMP/STPA(実践編)」[6]に例題として掲載されている、「ネット通販システム」を対象に手法を適用する。

利用者からの注文を電子的なネットワーク通信によって受け、注文された商品を利用者に配送する一般的なネット通販業務を想定している。ここでは、試行の例題として簡単化するために、以下のように限定されている[6].

- ・代金の決済などの金銭授受に関する業務は含まない

- ・商品の入荷に関する業務は含めない
 - ・利用者が注文した後のキャンセルや変更はない
- 機能としては利用者からの注文を受けると、在庫管理機能に引当てを指示する「販売管理機能」、商品の在庫データを管理する「在庫管理機能」、出荷指示を受けると、指示された商品をピックアップし、配送機能に引き渡す「出荷機能」、出荷機能から商品を受け取り、指示された宛名に配送する「配送機能」からなる。

5.2. 基本的な STAMP/STPA と動的コナーセンスの適用

対象システムに対して、提案手法による分析を行った。まず以下図 5-1 に従来の STAMP/STPA の手法でモデル化されたコントロールストラクチャの例を示す。

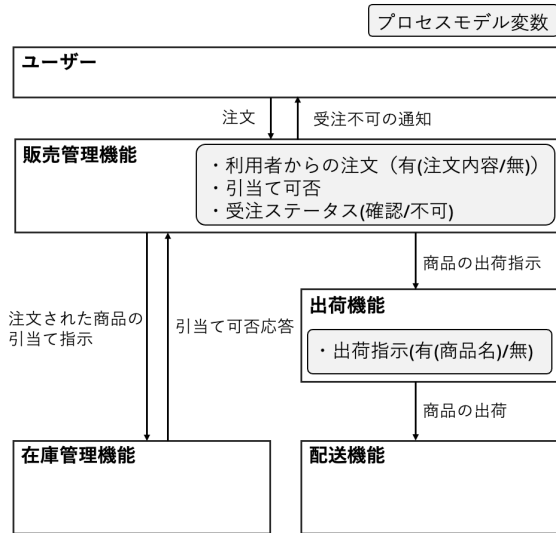


図 5-1 構築したコントロールストラクチャ

図 5-1 は、IPA のハンドブック[6]に基づいて構築されたものであり、一部のフィードバック経路が省略されている。図 5-1 における、コントロールストラクチャでは、コントローラー間の依存関係は、コントロールアクションとフィードバックによる静的な依存関係は明示的に表現されているものの、実行時の動的な依存関係は表現されていない。例えば、このシステムでは、販売管理機能が利用者からの注文を受け、在庫管理機能に対して商品の引当て指示を行い、在庫管理機能が引当て可否の応答を返し、そのあとに商品の出荷指示を行う、という流れでシステムは動作しなければならないが、図 5-1 ではその流れを明示的に表現されていない。そこで、4.2 節での手法をネット通販システムに適用し、分析を行った。またコナーセンスのあるコントローラー同士を線で結び、ラベル付けを行った。

図 5-2 は図 5-1 のコントロールストラクチャに対してコナーセンスを識別し、ラベル付けを行ったコントロールス

トラクチャである。ネット通販システムにおいては、実行順序のコナーセンスと値のコナーセンスが識別された。実行順序のコナーセンスは、販売管理機能と在庫管理機能との間で成立し、注文された商品の在庫を確認し、注文された商品の引当てが可能かどうかを判定した後に商品出荷指示のコントロールアクションが与えられなければならないため成立する。同様に値のコナーセンスも、販売管理機能と在庫管理機能との間で成立し、在庫管理機能で在庫数による引当て可否と、販売管理機能のプロセスモデル変数の引当て可否が一致していないと正常な動作をすることができないため成立する。

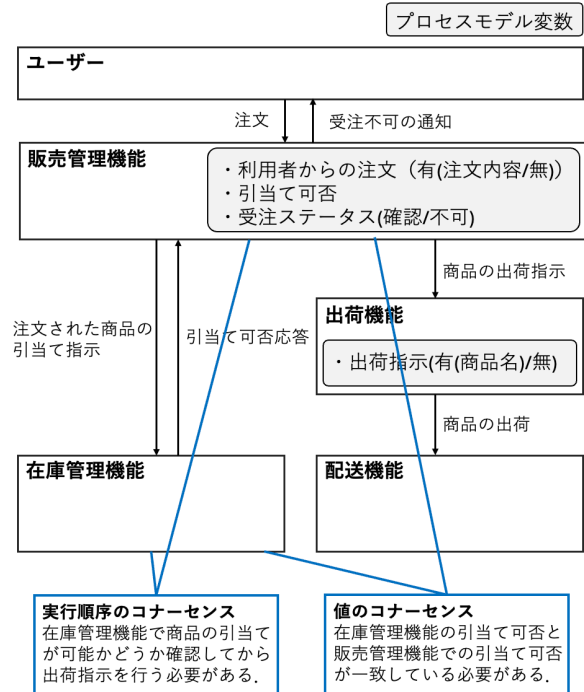


図 5-2 ラベル付けしたコントロールストラクチャ

図 5-2 のコントロールストラクチャから STAMP/STPA の従来手法である非安全なコントロールアクション(UCA)の識別の手順を行った。まずは、IPA のハンドブックで例題として示されていたUCAの識別した表を示す[6]。

表 5-1 IPA のハンドブックによるUCAの抽出結果

コントロールアクション	与えられないことがハザード	与えられることがハザード	早すぎる、遅すぎる、間違った順番
商品の出荷指示	受注ステータスが確定と決定された注文に対して出荷指示が発行されない。そのため、出荷が行われない。	実際には出荷可能な商品が注文数より少ない状況で受注ステータスが確定と決定され、出荷指示が発行される。商品の現物が不足し、出荷ができない。	受注ステータスが確定と決定された後、出荷指示の発行が遅れ、速やかに出荷されない。

表 5-1 では IPA のハンドブックが例題として示している UCA を識別した表である。ここでは、コントロールアクションが与えられないことがハザードにつながる、与えられることがハザードにつながる、早すぎる・遅すぎる・間違った順番、コントロールアクションの適用が短すぎる・長すぎる、という 4 つのガイドワードに沿って分析されているが、コントロールアクションの適用が短すぎる・長すぎるというガイドワードによる識別はなかったため省略している。これをもとに、図 5-2 のコントロールストラクチャで識別される UCA を抽出したところ、表 5-1 では足りない UCA があつたため、それを以下表 5-2 として示す。

表 5-2 では、赤太字で表 5-1 と比べて追加された UCA を示している。「在庫管理機能から出される在庫数による可否応答と販売管理機能の引当て可否が一致せず商品が出荷されない」という UCA は図 5-2 の値のコーナーセンスをもとに抽出された。また「注文された商品の引当て可否応答が返ってくる前に商品の出荷指示が出され、正しく商品が出荷されない。」という UCA は図 5-2 の実行順序のコーナーセンスをもとに抽出された。

表 5-2 追加した UCA を含めた抽出結果

コントロールアクション	与えられないことがハザード	与えられることがハザード	早すぎる、遅すぎる間違った順番
商品の出荷指示	受注ステータスが確定と決定された注文に対して出荷指示が発行されない。そのため、出荷が行われない。	実際には出荷可能な商品が注文数より少ない状況で受注ステータスが確定と決定され、出荷指示が発行される。商品の現物が不足し、出荷ができない。 在庫管理機能から出される引当て可否応答と販売管理機能の引当て可否が一致しない状態で出荷指示が出され商品が正しく出荷されない。	受注ステータスが確定と決定された後、出荷指示の発行が遅れ、速やかに出荷されない。 注文された商品の引当て可否応答が返ってくる前に商品の出荷指示が出され、正しく商品が出荷されない。

動的コーナーセンスの導入によって、従来の STAMP/STPA 単独で明示的に分析する手法がなかった動的な依存関係の特定を行い、動的な依存関係をもとに新たな UCA を識別した。識別された UCA から損失シナリオを抽出することで、設計段階で影響分析のためのシ

ステムの変更箇所の特定の精度が向上する可能性がある。

6. おわりに

本研究では STAMP/STPA に動的コーナーセンスを導入し、動的な依存関係の明示的な特定を可能にすることで、影響分析の精度向上を目指した。従来の STAMP/STPA では、静的な依存関係の分析が中心であり、動的な依存関係は分析者の裁量に委ねられていたが、本研究の手法により、実行順序やタイミングなどの観点から依存関係を分類し、より体系的な分析が可能となった。

本研究の成果として、影響範囲の特定を構造的に支援するための手法を提案し、その有効性を一事例を通じて示した。STAMP/STPA に動的コーナーセンスを導入することで、従来では明示されていなかった依存関係の一部を可視化できることを確認した。

しかしながら、本研究にはいくつかの課題が残されている。その一つとして、影響分析を行う際に要件定義書やユースケースの理解に依存している点が挙げられる。これにより、分析の属人性を完全に排除するには至らず、分析者の解釈による影響が残る可能性がある。今後の研究では、このような要件定義やユースケースの解釈の属人性を低減する方法についても検討する必要がある。

また、提案手法の実用性を高めるためには、実際のシステム開発に適用し、他の安全性分析手法や影響分析手法との比較評価を行うことが重要である。今後は、異なるシステムや開発プロジェクトにおいて本手法を適用し、その有効性をさらに検証していくことが求められる。

本研究を通じて、STAMP/STPA における影響分析の課題に対する新たなアプローチを提案し、その可能性を示した。今後の研究を通じて、本手法をより実用的かつ汎用的なものへと発展させることが期待される。

参考文献

- [1] N.G. Leveson: *Engineering a Safer World: Systems Thinking Applied to Safety*, MIT Press, (2011).
- [2] M.Richards, N. Ford: 「ソフトウェアアーキテクチャの基礎」, 島田浩二(訳), O'REILLY JAPAN, (2022).
- [3] 高橋加寿子, 塚本良太, 磯田誠: *変更要求に対するシステム設計書修正箇所抽出法の検証*, 情報処理学会研究報告, Vol.2018-SE-200, No.5
- [4] N.G. Leveson, J.P. Thomas: 「*STPA Handbook*」, MIT, (2018), <<https://psas.scripts.mit.edu/home/books-and-handbooks/>>, (参照 2025-02-13).
- [5] IPA: はじめての STAMP/STPA, <<https://www.ipa.go.jp/digital/stamp/ug65p9000000>

11xs-att/000055009.pdf>, (参照 2025-05-26).

- [6] IPA: はじめての STAMP/STPA(実践編), <
<https://www.ipa.go.jp/publish/qv6pgp00000010x0-att/000059652.pdf>>, (参照 2025-05-26).