STAMP/STPA を用いたセキュリティ分析設計

：生成 AI を活用した反復開発プロセスへの導入

STAMP/STPA-Based Security Analysis and Design

: Leveraging Generative AI in the Iterative Development Process

リーダー：安樂　啓之（インフォテック）　　吉村　隆広（アイホン）
研 究 員：藤井　広宣（ブライシス）　　石原　佑一（東京海上日動システムズ）
佐々木　瑛太（アズビル）　　戸田　優作（アズビル）
池上　達也（デンソー）　　永野　哲(ミラクシアエッジテクノロジー)
紫藤　紀行（三菱電機ソフトウエア）
主　査　：金子　朋子（創価大学）
副 主 査：髙橋　雄志（東日本国際大学）
アドバイザー：佐々木　良一（東京電機大学）

研究概要

STAMP/STPA を用いたセキュリティ分析の有効性は先行研究により示されている．しかし，システムの大規模化・複雑化が進む一方で開発期間の短縮が求められる現状において，分析の専門性の高さと所要時間の長さから，反復的な分析の実施が困難であるという課題が存在する．本稿では，生成 AI により STAMP/CAST による事故分析効率化が期待できるとした過去の研究を踏まえ，STAMP/STPA に基づくセキュリティリスク分析に生成 AI を統合することでその課題解決を図った．そして実験により，プロンプトを用いた STAMP/STPA 分析が実用的に機能し，開発工程の進行に応じて反復的に分析できることを確認した．このことから，セキュリティバイデザイン原則に基づき反復的に分析・設計を行う開発プロセスの実現可能性についての知見を得た．

1. はじめに

2010 年代以降，複雑化した情報システムとそれらの相互接続が発展した[1]ことにより，従来の構成要素の分析ではとらえきれない要素間の相互作用に起因するセキュリティリスクが顕在化している[2]．例えば，2015 年のウクライナ大規模停電では，電力設備への侵入と遠隔操作により送電が遮断され，複数システムの相互作用を悪用した攻撃により約 22 万 5 千世帯が停電した[3]．また，2010 年のマルウェア Stuxnet 攻撃では，制御システムと情報系システムの接続点を介してマルウェアが侵入し，個々の構成要素は正常に動作していたにもかかわらず，それらの相互作用により核燃料施設の遠心分離機が破壊された[4]．このような要素間の相互作用に起因するリスクの分析手法として，セーフティ分野では STPA（Systems-Theoretic Process Analysis）が複雑なシステムの安全性分析に有効であることが示されている[5]．STPA は，システムを構成要素の相互作用として捉え，安全制約の違反状態を体系的に見つける手法として，航空宇宙や自動車など多様な分野で有効性が示されている[6][7]．この STPA の枠組みをセキュリティ分野に応用した STPA-Sec により，個別の脆弱性に依存せず制御構造の観点から分析を行うことができることが示されている[8]．

Pope はアジャイルアプリケーション開発時に STPA によるセキュリティ分析を行う際に，早いタイミングからネットワークや IT の専門家を加えて開発する有用性をコストや設計上の理由をもとに主張した [9]．それに加え，ウォーターフォール開発では進行に伴

い，システム設計の詳細化に伴う分析の精緻化が必要であること，またアジャイル開発では要件の動的な変更に応じた設計成果物等の更新が必要である．

しかし，STPA 分析は高度な専門知識を要し，分析対象の理解や制御構造図の作成に多大な工数を要するため，反復的な分析の実施が困難であることが指摘されている[10][11]．我々は，これまでの研究で STAMP/CAST による安全性分析に生成 AI を利用することで分析効率が向上することを示した[12]．しかしこの先行研究は一度の分析を対象としており，開発プロセスにおける反復的な分析を対象としていない．

本稿では STPA を用いたセキュリティ分析に生成 AI を活用し，開発工程の進行に伴って分析を反復的に実施することで分析精度を段階的に向上させる手順を提案する．また，実際のシステム開発を模擬した事例により，提案手順の実行可能性を確認する．

2. 関連技術／研究

2.1 関連技術

2.1.1 STAMP (System-Theoretic Accident Model and Processes)

STAMP は，Leveson により提唱されたシステム理論に基づく事故モデルである[13]．STAMP では，システム安全は構成要素だけでなく，コントローラと被制御プロセス間の相互作用が適切に機能することで実現されると考える．このため，たとえ構成要素が仕様通りに動作していても，不適切な制御や情報伝達によって事故が発生し得るとし，制御構造（Control Structure）を用いて制御および情報の流れを明示的に表現できる．

2.1.2 STPA / STPA-Sec

STPA（System-Theoretic Process Analysis）は，STAMP に基づく代表的なハザード分析手法であり，安全性を故障の問題ではなく制御問題として捉える[14]．システムレベル，のアクシデントから出発し，トップダウンに分析を進める点が特徴である．以下の 4 段階を基本手順として進める．

(1) 分析目的の定義
(2) 制御構造のモデル化
(3) 非安全な制御アクション（Unsafe Control Action ： UCA）の識別
(4) 損失シナリオの識別

STPA-Sec（System-Theoretic Process Analysis for Security）は，STPA をセキュリティ分析へ拡張した手法であり，サイバーセキュリティを対象とする．STPA と同様の手順に従い，攻撃者による不正な制御入力や制御アルゴリズムの改変といったセキュリティ誘発要因を考慮する点に特徴がある．

これらは複雑なシステムにおいて，従来手法では捉えにくいシステミックな問題に対応可能である．一方，高度な専門知識を要し分析負担や主観性が課題とされている．

2.2 先行研究

2.2.1 セキュリティ分析の実態

セキュリティバイデザインの観点では，セキュリティ分析を開発工程の初期にあたる企画・要件定義工程から行うことが望ましい． Rindell らは，セキュリティに関するエンジニアリング活動をアジャイル開発に適用することに関するアンケートをフィンランドのソフトウェアエンジニア，ソフトウェアセキュリティの専門家，およびソフトウェア開発プロセスの関係者に対して行っている[15]．その中で要件定義工程ではセキュリティ要件のレビューが非常に重要であると評価している一方，半数近い関係者が実施については後ろ向きであるとの実態が示されている．

2.2.2 STPA-Sec によるセキュリティ分析

我々は STPA-Sec によるセキュリティ分析を自動運転システムに適用し，情報の改ざんや DoS 攻撃などのセキュリティ要因と，運転者の危険察知遅れなどのセーフティ要因を同時に考慮した分析ができることを実証した[16]．また，金子らは脅威分類モデルである Microsoft の STRIDE モデルを用いて，具体的かつ網羅的なセキュリティリスク分析を可

能にする手法を提案した[17]．これらの研究により，大規模化・複雑化したシステムにおいて，STPA-Sec によるセキュリティ分析が有効性であることを示している．

2.2.3　生成 AI による STPA 分析

大規模言語モデル（LLM）を中心とする生成 AI の発展に伴い，STPA を含む安全・ハザード分析への適用可能性が検討され始めている．Qi らは，ChatGPT を用いた STPA のケーススタディを通じて，ハザード識別や損失シナリオ記述といった工程において LLM が分析者を補助し得ることを示している[18]．しかしながら，生成結果には漏れや不正確さが含まれる可能性があることを指摘し，人間の専門家による確認・修正を前提とした活用が不可欠であると述べている．また，Diemert らは，LLM を用いてハザード原因の抽出を支援する枠組みを提案し，安全分析における生産性向上の可能性を示している[19]．

これらの先行研究は一度の分析作業を前提とした生成 AI の支援効果に主眼を置いており，生成 AI による分析の質や能力の評価を目的としている．このため，セキュリティに関するエンジニアリング活動をアジャイル開発に適用するといった具体的な運用方法についてまでの言及はされていない．あわせて，開発工程の進行に伴い分析，設計を反復的に実施することについても同様に言及されていない．

3.　提案手法

STAMP/STPA によるセキュリティ分析は，システム全体の相互作用を考慮した体系的な手法として有用であると考えた．しかし，(1)詳細なモデル化に要する時間と労力，(2)セキュリティや対象業界など幅広い専門知識の必要性，という 2 つの課題により，開発プロセスにおける反復的な実施は困難であった．特に，大規模・複雑化したシステムを対象とする場合，この負担はさらに増大する．

提案手法は，上記の課題を解決するために STAMP/STPA の分析手順とシステム構成をもとに生成 AI が実行可能なプロンプトをテンプレート化する．テンプレート化により，生成 AI は数分程度で体系的な初期分析を実行できるようになり，(1)で示した問題から困難であった反復的なセキュリティ分析を実用的にできると期待される．

図 1 に提案手法の全体像を示す．

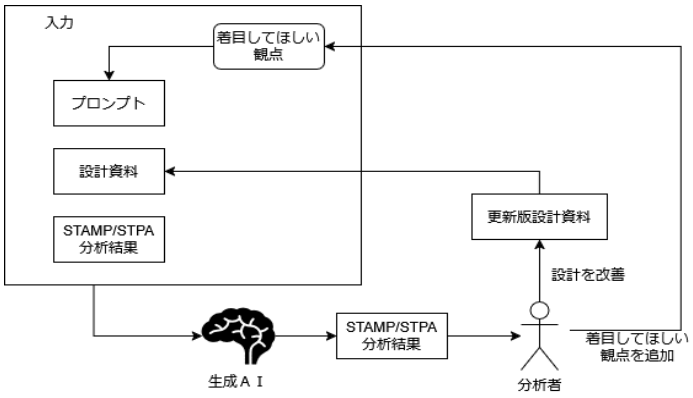

図 1　提案手法の全体像

3.1　提案手法の特徴

本手法は，プロンプトのテンプレート化を核とし，これを実用化するための要素の追加と反復プロセスを組み合わせることで，以下の特徴を有する．

(1)　分析時間の短縮

テンプレート化により，生成 AI は数分程度で初期分析を完了できる．これが，反復的なセキュリティ分析を実用的に可能にする最も重要な要素である．

(2) 分析品質の確保
テンプレートに分析手順とセキュリティ観点を明示することで一定の品質を確保する.
(3) 反復的な分析の実現
生成 AI による反復的な分析は，システム変更に伴う軽量な再分析を実現する．これにより，開発プロセス全体を通じた継続的なセキュリティ改善を実現する.
(4) 専門知識の補完
分析者がテンプレートに専門知識（規格，脅威情報）,および分析観点などを組み込む.これにより生成 AI が専門知識やシステム固有の文脈に基づく分析を可能とする.
(5) セキュリティバイデザインの実現
開発の各段階で実用的にセキュリティ分析,および改善の実施により，企画・要件定義段階からセキュリティを組み込むセキュリティバイデザイン原則を実現する.
これらの特徴により，従来は時間と専門性の制約から困難であった，開発プロセス全体を通じた反復的なセキュリティ分析・改善が実用的に可能となる.

4. システム開発ドキュメントの STAMP/STPA によるリスク識別と反復適用実験

実験の目的は，以下の課題についての検証である．なお，本実験では，要件定義書や設計書などのシステムを説明するドキュメントを作成することを設計と表現する.

① システムの要件定義や設計の序盤から STAMP/STPA による分析を行い，設計対象のリスク識別を短時間(数分程度)で行うことができるか.
② ①はリスク対策済みの設計情報の再評価にも適用できるか.
③ ①と②より設計，リスク識別を繰り返し実施することで，システム内のリスクコントロールを改善することができるか.

実験の概要について図 2 に示す.

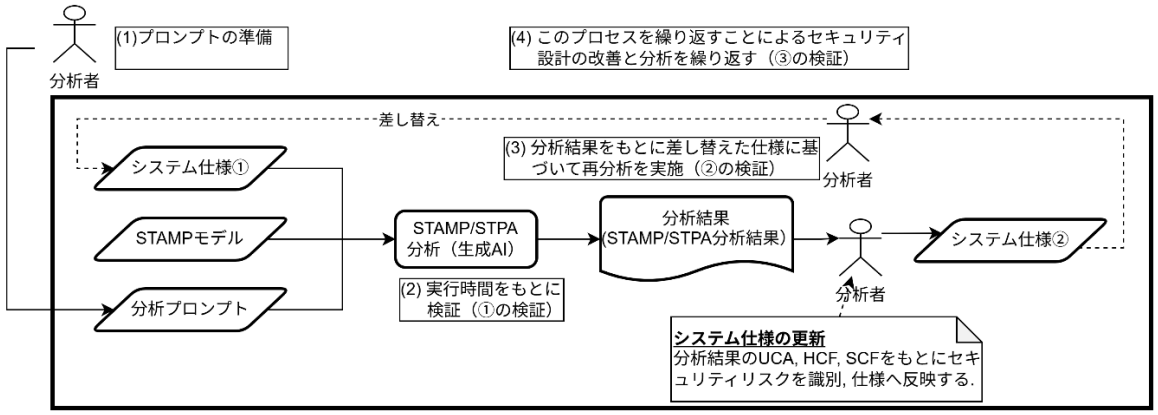

図 2 実験の概要

4.1 実験手順

(1) 実験準備
システム仕様， STAMP モデル,およびこれらの内容を元に STAMP/STPA に基づきリスクの識別を行う UCA,HCF,および SCF をリストアップするプロンプトを準備する.
(2) 生成 AI による分析の実施と分析実行時間の測定
(1)で用意したプロンプトを用いてシステム仕様,STAMP モデルに対する STAMP/STPA 分析を生成 AI により実行する． 分析結果として， UCA,ハザード要因(Hazardous Causal Factor ： HCF），セキュリティ要因(Security Causal Factor ： SCF)が導き出されることを確認する．またその分析実行時間は数分程度で分析できることを確認する.
(3) リスク対策後のシステム仕様に対する再分析の実施（1 周目）

(2)の分析結果に含まれている HCF, SCF からリスクを識別し, システム仕様にリスク回避,あるいはリスク削減となる対策を施す. その後,見直したシステム仕様, (1)で用いた STAMP モデル,およびプロンプトを用いて再度生成 AI による分析を行うことが出来ることを確認する,このことからリスク対策済みの設計情報の再評価にも適用できることを確認する.

(4) リスク対策後のシステム仕様に対する再分析の実施（2 周目）

(3)をもう一回繰り返すことが出来ることを確認する.(3)同様に対策をとったリスクが分析結果に反映されていることを確認する.

4.2 実験結果

各手順における結果を(1)から(4)に示す.

(1) 実験準備

本実験にて用意したものを表 1 **表 1** に示す.

表 1 実験にて用意したもの

No	用意したもの	説明
1	プロンプト	予め用意したシステムの仕様記述,およびSTAMPモデルをもとに,生成AIを用いてSTAMP/STPAによる分析を行うプロンプト。利用した生成AIはGoogle Gemini(モデル：Gemini2.5 Flash thinking,2025/12/19実施)を用いた.
2	STAMPモデル	STAMP/STPA分析に必要な分析情報. 実験ではスマートロック装置におけるアクシデント,ハザード,および安全制約を記載したものを用意した.
3	システム仕様	システムに関する仕様や設計資料. 今回の実験ではスマートロック装置の設計を用意した.

(2) 生成 AI による分析の実施と分析実行時間の測定

生成 AI による分析実行時間はおおむね数分程度であり,少なくとも 5 分を上回ることはなかった.これにより,課題①について可能であることが分かった.

(3) リスク対策後のシステム仕様に対する再分析の実施（1 周目）

リスク対策前後の生成 AI によるリスク識別結果をまとめたものを表 2 に示す.

表 2 識別リスクとリスク対応後の再分析（1 周目）

分析・対処前		対策実施	対策内容	分析・対処後		リスク変化	
UCA番号	識別されたリスク（HCF/SCF）			UCA番号	識別されたリスク（HCF/SCF）	発生頻度	影響度
UCA-1	・フィードバックの不十分・欠落・遅延	○	開錠操作の動作定義を改善	-	-	解消	解消
UCA-2	・プロセスモデルの不一致・不完全	○	開錠操作の動作定義を改善	-	-	解消	解消
UCA-3	・不十分な制御アルゴリズム	○	開錠操作の動作定義を改善	-	-	解消	解消
UCA-4	・上位からの指示や外部情報の誤り・欠落	○	施錠ロジックの制御異常への対策を実施	UCA-8	・プロセスモデルの不一致	減少	変化なし
UCA-7	上位からの指示や外部情報の誤り・欠落	○	FWアップデート処理の改善	-	-	解消	解消
UCA-11	・コントロールアクションの不適切・無効・欠落	○	開錠時の認証処理を改善	UCA-5	・モバイルアプリまたはクラウドサーバーの認証情報管理の不備	減少	変化なし
-	-	ー	新たに検知された	UCA12	・ 不十分な制御アルゴリズム	ー	ー
-	-	ー	新たに検知された	UCA15	・ 不正確な情報（ペアリング認証の不足）	ー	ー
UCA-1	Denial of Service（サービス拒否）	○	クラウドサーバの通信要件を具体化	UCA-1	Information Disclosure / Repudiation（情報漏洩/否認）	減少	減少
UCA-4	Tmpering（改ざん）	○	通信の暗号化対象を全体と明示	-	-	解消	解消
UCA-7	Denial of Service（サービス拒否）	○	FWアップデート処理の定義を追加	-	-	解消	解消
UCA-9	Spoofing（なりすまし）	○	スマートロックの設定機能の定義を追加	UCA-15	Elevation of Privilege（権限昇格）	変化なし	減少
UCA-11	Spoofing（なりすまし）	○	認証情報管理について具体化	UCA-5	Spoofing（なりすまし）	減少	変化なし
UCA-14	Information Disclosure（情報漏洩）	ー		UCA-17	Information Disclosure（情報漏洩）	変化なし	変化なし
-	-	ー	新たに検知された	UCA-4	Denial of Service（サービス拒否）	ー	ー
-	-	ー	新たに検知された	UCA-12	Tampering / Elevation of Privilege（改竄/権限昇格）	ー	ー

分析前後の UCA を識別する番号(UCA 番号),識別されたリスク(HCF/SCF), 対策実施には対策を実施したか(○:実施/ー:実施無し),リスク変化列には対策前後に識別された同一リスクについてその変化を発生頻度,および影響度の観点で記載した.

例えば,UCA-11 は,モバイルアプリ利用者の認証情報を盗み出してなりすましを行うリスクを指している. 設計を改善することにより, モバイルアプリ利用者の認証情報を盗み出してなりすましを行う前にモバイル機器やクラウドサービスの脆弱性を利用

しなければならなくなった．そのため，情報漏洩のための手順が困難になったと判断し,発生頻度の減少と記載した．影響度については，その影響範囲の変化を評価している．例えばUCA-9は不正アクセスによるすべての利用権限を奪取されることが問題であったが,対策を施すことにより被害範囲がモバイル機器のみに限定されることから,減少と記載した.これらのことから生成AIによる分析は，リスク対策済みの設計情報の再評価にも適用できることが分かった.

(4) リスク対策後のシステム仕様に対する再分析の実施（2周目）

(3)同様，2周目の実験を行い,リスク対策前後の生成AIによるリスク識別結果をまとめたものを表3に示す．前項同様に識別されたリスクに対して対処を行ったものは,再分析後にリスクが減少していることが分かった.

表3 識別リスクとリスク対応後の再分析（2周目）

分析・対処前		対策実施	対策内容	分析・対処後		リスク変化	
UCA番号	識別されたリスク（HCF/SCF）			UCA番号	識別されたリスク（HCF/SCF）	発生頻度	影響度
UCA-8	・プロセスモデルの不一致	○	スマートロック利用条件としての認証要件を明示	UCA-2	・プロセスモデルの不一致	変化なし	変化なし
UCA-5	・モバイルアプリまたはクラウドサーバーの認証情報管理の不備	○	多要素認証による認証エラー対策	–	–	解消	解消
UCA-12	・ 不十分な制御アルゴリズム	○	FWの署名実施を追加	UCA-4	・不十分な制御アルゴリズム	減少	変化なし
UCA-15	・ 不正確な情報（ペアリング認証の不足）	○	セキュアなBluetooth通信を要件に追加	UCA-7	・不適切なコントロールアクション	減少	変化なし
–	–	ー		UCA-1	・動作の遅れ	–	–
–	–	ー		UCA-5	・意図しない外乱	–	–
–	–	○		UCA-8	・不十分な制御アルゴリズム	–	–
UCA-1	Information Disclosure / Repudiation（情報漏洩/否認）	○	通信のセッション管理を厳格に変更/暗号要件追加	–	–	減少	減少
UCA-15	Elevation of Privilege（権限昇格）	○	アプリケーションの権限を最小にする要件追加	UCA-7	・Information Disclosure（情報漏洩）	減少	減少
UCA-5	Spoofing（なりすまし）	○	サービスの認証要件を具体的に記載	–	–	解消	解消
UCA-17	Information Disclosure（情報漏洩）	○	セキュアなBluetooth通信を要件に追加	UCA-8	・Denial of Service（サービス拒否）	減少	変化なし
UCA-4	Denial of Service（サービス拒否）	○	多様所認証など認証の厳格化	–	–	解消	解消
UCA-12	Tampering / Elevation of Privilege（改竄/権限昇格）	○	アプリケーションの権限を最小にする要件追加	UCA-4	・Tampering（改ざん）	減少	変化なし
–	–	ー		UCA-2	・Spoofing（なりすまし）	–	–
–	–	ー		UCA-9	・Denial of Service（サービス拒否）	–	–

4.3 考察

4.2(3),および(4)より，設計，リスク識別，およびリスク対策のための設計改善を繰り返し実施することでシステム内のリスクコントロールを改善できることが分かった．本実験の設計改善は分析者が実施したが，生成AIによる改善提案を行うことによって，より効率的にセキュリティ仕様の改善につながることが期待できる．また，4.2(2)より,分析自体も短時間で行うことができたことから，人手による分析より手軽にSTAMP/STPA分析を行うことができるので，システム開発の上流の段階から反復的なリスク分析による開発プロセスの改善につながると考えられる.

4.4 妥当性への脅威

実験の分析前後におけるリスクの傾向については，発生頻度と影響度の増減について,分析者の主観に基づいて評価を行っている．そのため，リスクの評価について主観的な評価が含まれている．また，生成AIについてもGeminiのみでの検証により比較を行っている.より多くのモデルによる評価を行うことにより評価の傾向が異なる可能性が考えられる.

5. 今後の展望と実用化に向けた提案

本稿では，STAMP/STPAの反復的な分析・改善の有効性を検証した．これによって，この手法は企画・要件定義から，ITシステムのライフサイクル全般における有効性が期待できる.

5.1 ITシステムのライフサイクル全般における改善

提案手法により，企画・要件定義といった開発の初期段階から，STAMP/STPAに基づく

セキュリティリスクの識別およびリスク対応を通じた反復的な設計改善が可能となった．このような改善活動を反復的に実施することにより，システムに内在するセキュリティリスクを早期段階から容易に識別できるようになり，それらに対して早期に対処することで，セキュリティバイデザインの実現が期待できる．

また，本稿案手法は運用フェーズ以降においてもその仕組みの維持をすることが重要である．それはサービス提供開始後も新たな脅威は継続的に増大することにより運用時においてもシステムのセキュリティリスク識別と設計改善が求められるためである．例えば，生成 AI により本提案手法を実装したエージェントを分析基盤として運用段階においても維持できれば，組織や担当者の変更に依存しない一貫した判断基準に基づくセキュリティリスクの識別および設計改善の仕組みを，システムライフサイクル全体にわたって維持することが可能となる．これによって，開発時のセキュリティ設計の意図を維持しながら，新たな脅威に対応し続けることが可能になる．こうした仕組みが DevSecOps に統合されることにより,システムのライフサイクル全体においてセキュリティ改善ができることが期待できる．

5.2　今後の研究課題

本稿で示した手法の実用化と適用範囲の拡大のため,取り組むべき課題を以下に示す．

(1) 多様なシステムでの汎用性検証
本稿ではスマートロックシステムを対象とした検証にとどまっている．異なる規模や複雑度のシステム，多様なドメイン（IoT，クラウドサービス，組込みシステムなど）への適用を通じて，提案手法の汎用性の検証が課題である．

(2) AI エージェント化と組織展開
構築したプロンプトテンプレートを AI エージェント化し，組織内で共有することで，担当者による分析結果のばらつきを抑制し，組織全体での分析品質の均一化が期待できる．この実装と運用効果の測定が課題である．

(3) 複数生成 AI モデルの比較評価とテンプレート可搬
本稿では Gemini を用いたが，構造化されたテンプレートは特定の AI モデルに依存しない設計となっている．Claude，ChatGPT など複数モデルでの出力品質の比較検証と，プロンプトテンプレートの可搬性の評価が課題である．

(4) RAG による知識更新の実装
脅威はサービス提供開始後も進化し続けるため,RAG(Retrieval Augmented Generation)を活用し,最新の脅威情報やセキュリティ規格を分析に取り込む仕組みの実装と,その効果の検証が課題である．

(5) 経済性を考慮したレビュー体制の確立
生成 AI は網羅的に脅威を抽出できる一方,過度に多くの対策を提示する可能性がある．守るべき資産の優先順位付けと，インシデント発生確率および影響度を考慮した効率的なレビュー体制の確立が課題である．

6　おわりに

本稿では，STAMP/STPA によるセキュリティ分析の専門性の高さと所要時間の長さが開発サイクルでの反復的な分析を困難にしているという課題に対し，生成 AI とプロンプトテンプレートを組み合わせた分析手法を提案し，その有効性を検証した．

提案手法は，STAMP/STPA の分析手順とシステム構成をプロンプトテンプレートとして構造化し，生成 AI による初期分析と人間のレビューを組み合わせた反復的アプローチである.実験対象システムへの適用により,(1)生成 AI による分析が 5 分以内で完了すること,(2)前回の分析結果を活用した効率的な再分析が可能であることを確認した．また，対策実施後のリスク評価から,反復的な分析とリスク低減のサイクルが機能することが示された．

本研究の成果により，専門知識を持たない開発者でも，プロンプトテンプレートと生成 AI の支援によって STAMP/STPA に基づくセキュリティ分析を実施できることが示された．これは，開発の早期段階からセキュリティリスクを継続的に評価する開発プロセスを可能にし，セキュリティバイデザインの実践を支援するものであることから，今後は 5 章で述べた課題解決に取り組むと共に，手法のさらなる改善につとめていきたい．

参考資料

[1] 独立行政法人 情報処理推進機構(IPA)，制御システムのセキュリティリスク分析ガイド 第 2 版，2023 年
[2] 経済産業省商務情報政策局，第 8 回産業サイバーセキュリティ研究会 事務局説明資料，2024 年
[3] 独立行政法人 情報処理推進機構(IPA)，制御システム関連のサイバーインシデント事例 1：2015 年ウクライナ大規模停電，2019 年
[4] 独立行政法人 情報処理推進機構(IPA)，制御システム関連のサイバーインシデント事例 3：Stuxnet 攻撃，2020 年
[5] 独立行政法人 情報処理推進機構(IPA)，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～ Ver.1.0，2016 年
[6] Nancy G. Leveson, “Engineering a Safer World”, MIT Press, 2012
[7] 西澤賢一ほか，セーフティ&セキュリティ開発における STAMP/STPA の有効性検証 日本科学技術連盟 SQiP 2018 年
[8] 福島 祐子，CPS のサイバーセキュリティに求められる安全分析と STPA-Sec の有効性，(日本ユニシス技報 41(2))，2021
[9] Gregory M. Pope, Systemic Theoretic Process Analysis (STPA) Used for Cyber Security and Agile Software Development, Lawrence Livermore National Laboratory, 2021
[10] 森川聡久ほか，「開発 SE が使える!今注目のリスク分析手法 STAMP/STPA のシステム開発への適用～システム開発での STAMP/STPA の実践を通じて得られたテーラリングのエッセンス～」，ソフトウェアプロセス改善カンファレンス (SPI Japan 2020)，2020
[11] 独立行政法人 情報処理推進機構(IPA)，はじめての STAMP/STPA（実践編）～システム思考に基づく新しい安全性解析手法～，2017 年
[12] 安樂啓之ほか，日本科学技術連盟 SQiP，STAMP/CAST 分析における生成 AI の支援～羽田港宇高航空機衝突事故を題材として～，2025 年
[13] Nancy G. Leveson, Engineering a Safer World: Systems Thinking Applied to Safety, The MIT Press, 2012
[14] 独立行政法人 情報処理推進機構(IPA)，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～ Ver.1.0，2016 年
[15] Kalle Rindell et al., Security in agile software development: A practitioner survey, Information and Software Technology Volume 131, 2021.
[16] 大森淳夫ほか，日本科学技術連盟 SQiP，セーフティ＆セキュリティ開発のための技術統合提案と事例作成 ～STAMP/STPA とアシュアランスケースの統合～，2018 年
[17]金子朋子ほか，情報処理学会研究報告，安全性解析手法 STAMP／STPA への脅威分析（＝STRIDE）の適用，2018 年
[18] Yi Qi et al., Safety Analysis in the Era of Large Language Models: A Case Study of STPA using ChatGPT, Machine Learning with Applications, 2025
[19] Simon Diemert, Jens H. Weber, Can Large Language Models assist in Hazard Analysis?, arXiv preprint, 2023