目　　次

ま　え　が　き

本書は安全・セキュリティ分析、モデルベースシステムズエンジニアリング、アシュアランスケースを解説した本です。自動運転やドローンシステム、さらには急激に発展している生成 AI 技術を用いたシステムなど、世界を変えうるシステムが登場しようとしています。しかしそれらのシステムの信頼性を保証することは従来のシステム以上に難しくなってきています。これらのシステムは、その内部と環境に常に変化と不確実性があるオープンシステムです。本書では自動運転システムなどのオープンシステムの安全性、信頼性（総合信頼性、ディペンダビリティ）を保証するための手法を、安全・セキュリティ分析、モデルベースシステムズエンジニアリング、そしてアシュアランスケースを結びつけて解説した、世界で初めての本です。これらの内容は、通常の大学の授業ではあまり出てこない内容ですが、特に自動車産業などの安全性や信頼性が極めて重要な分野では必須となる手法です。ぜひ本書を通して、これらの手法を習得し、将来、あるいは現在におけるシステム開発と運用に役立てていただけたら幸いです。

2024 年 10 月

松野裕, 高井利憲, 岡本圭史

1 イントロダクション

1.1 システムのディペンダビリティ：基礎概念と現代的課題

1.1.1 システムとは何か

私たちの日常生活において、「システム」という言葉をよく耳にします。しかし、この概念を正確に定義するのは簡単ではありません。ここでは、参考文献 [x] の定義を示します。

システム: 他のシステムと相互作用するもの。ここでいうシステムは、与えられた環境であり、ハードウェア、ソフトウェア、人間、および自然現象を伴う物質的世界などである。システム境界はシステムとその環境が接する境目である。

システムは互いにサービスを介して相互作用を行います。参考文献 [x] のサービスの定義を示します。

サービス：システムが (提供者の役割によって) 提供するサービスとは、その利用者から見える振る舞いです。ここで利用者とは、提供者からサービスを受ける別のシステムです。提供者のシステム境界のうちサービス提供がなされる部分を提供者のサービスインタフェースといいます。

システムは常に特定の環境に置かれており、その環境との相互作用を通じて機能します。例えば、自動車というシステムは道路という環境の中で機能し、気象条件や交通状況などの環境要因の影響を受けます。

システムとサービス、および環境の関係を図??に示します。

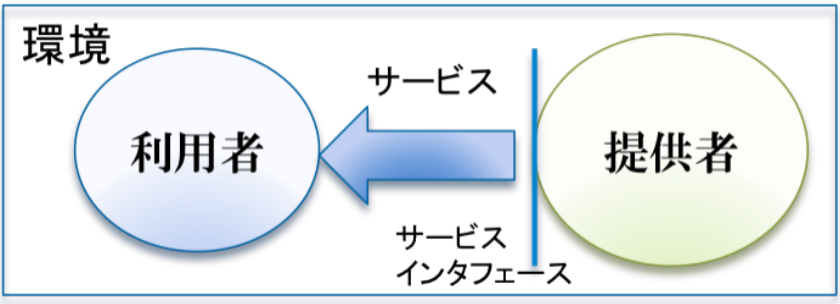

図 1.1　環境におけるシステムとサービス

上記の定義は抽象的ですが、サービスは、システムによって私たちに提供される機能や便益です。私たちユーザーは、このサービスを通じてシステムと関係を持ちます。例えば、スマートフォンというシステムは、通話、メッセージング、インターネット閲覧などのサービスを提供し、私たちはこれらのサービスを通じてスマートフォンと関わっています。

1.1.2　ディペンダビリティの概念

‘ディペンダビリティ（Dependability）は、直訳すると「依存可能性」となりますが、システム工学では「総合信頼性」と訳されることが多い重要な概念です。この概念の起源は、あるシステムが別のシステムに依存（Depend）することができる、そのシステムの性質を表すことにあります。

例えば、運転手（システム A）が自動車（システム B）に依存する場合を考えてみましょう。自動車がディペンダブル（依存可能）であるためには、どのような性質を持つべきでしょうか？安全であること、運転しやすいこと、目的地まで確実に到達できることなどが挙げられるでしょう。このように、ディペンダビリティは利用者や環境によって求められる性質が異なる、相対的な概念です。

ディペンダビリティは、日本語では現在「総合信頼性」と呼ばれています。以下にいくつかの定義を示します。

- 容認できる以上の頻度と深刻さのサービス障害を防ぐシステムの能力
- 可用性性能及びこれに影響を与える要因、すなわち信頼性性能、保全性性能及び保全支援能力を記述するために用いられる包括的な用語 (JIS Z 8115)

ディペンダビリティは JIS Z 8115 などにおけるように、可用性や信頼性など、従来のシステムの非機能要求を総合したものとして考えられてきました。しかし、もともとはシステム間の依存関係に由来し、サービス障害などに関連する概念です。可用性や信頼性は、システムがディペンダブルであるためには必要な非機能要求であると考えられますが、システムは環境に置かれ、また利用者によってサービスは使用されるため、必要となる機能、非機能などは環境や利用者によって異なります。またディペンダブルであるために必要となる機能や非機能が提供されなくなった場合、つまりシステムに障害が発生した場合、システムは復旧を行い、サービスを継続し無くてはなりません。

以上をまとめるとシステムがディペンダブルであるためには、以下の条件を満たす必要があります：

- 利用者や環境において望まれる性質を持ち続けること
- サービスを継続的に提供すること

これらを満たすためには、システムがその性質を失った場合（つまり、ディペンダブルでなくなった場合）には、速やかに復旧を行い、サービスを継続する能力も求められます。

1.2 ディペンダビリティの体系と用語

ディペンダビリティの概念は、1980 年代から国際的な研究グループ（IFIP WG 10.4 “Dependable Computing and Fault Tolerance” など）によって体系

化され、用語の整理が行われてきました。当初は「耐故障性 (Fault Tolerance)」研究から発展し、近年ではセキュリティの概念も含めて議論されています。

ここでは、Avizienis et al. (2004) による体系に基づいて、ディペンダビリティの主要な概念を紹介します。

1.2.1　ディペンダビリティ属性

ディペンダビリティ属性は、システムがディペンダブルであるために持つべき特性を表します。主な属性には以下のものがあります：

- 可用性（Availability）：正しいサービスの即応性
- 信頼性（Reliability）：正しいサービスの継続性
- 安全性（Safety）：利用者と環境へ破壊的影響をもたらさないこと
- 一貫性（Integrity）：不適切なシステム変更がないこと
- 保守性（Maintainability）：変更と修理を受け入れられること

これらの属性は相互に関連しており、時には相反する関係にあることもあります。システム設計者は、対象システムの要求に応じてこれらの属性のバランスを取る必要があります。

1.2.2　ディペンダビリティへの脅威

システムのディペンダビリティを脅かす要因は、以下の 3 つの概念で整理されています：

- 欠陥（Fault）：エラーの原因となるとみなされる、あるいは推定されるもの、こと
- 誤り（Error）：障害が起こりうるシステムの状態（ただし、エラー状態になったからといって、必ずしも障害が起こるとは限らない）
- 障害（Failure）：サービスが正しいサービスから逸脱する出来事

これらの概念は因果関係にあり、欠陥がエラーを引き起こし、エラーが障害につながる可能性があります。

1.2.3 ディペンダビリティへの脅威に対処する手段

ディペンダビリティへの脅威に対処するため、以下の 4 つの手段が提案されています：

- 欠陥防止（Fault Prevention）：欠陥の導入や発生を防ぐ
- 耐故障性（Fault Tolerance）：欠陥がある中で障害を防ぐ
- 欠陥除去（Fault Removal）：欠陥の数や深刻度を減らす
- 欠陥予測（Fault Forecasting）：欠陥の現在の数、今後の障害、影響などを予測する

ディペンダビリティの属性、脅威、対処手段をまとめると図 X のディペンダビリティとセキュリティの木になります。ディペンダビリティとセキュリティは、システムのあるべき姿に関する概念であり、図 X はシステムのあるべき姿に関する研究成果の一つと言えます。

1.3 現代のシステムとディペンダビリティの課題

IT・組込みシステムの重要性が増すにつれ、IT システムは私たちの生活および社会において欠かせないインフラとなり、さらにポータル化してきました。私たちの生活、そして社会は便利になりましたが、同時に従来のディペンダビリティや耐故障性の考え方だけでは対処できない問題が増えてきました。これらの問題は以下のようにまとめられます（参考文献 [2]）。

- システムの大規模化による問題
- プログラムサイズの巨大化
- 多機能化
- ブラックボックス化したコンポーネント
- 複雑化
- 環境の変化による問題
- 技術の急速な進化へのキャッチアップの困難さ

- 接続システムの多様化
- 利害関係者（Stakeholder）の変化による問題
- 要求の頻繁な変更
- 要求や合意に対する考え方の違い

これらは従来から問題とされてきたものですが、情報システムの規模が著しく拡大したことで、私たちはこれらの問題に対応できなくなりつつあるかもしれません。近年、深刻な情報システムの障害が報告されています。例えば、2012年6月に発生したレンタルサーバでの5000を超える企業データ消失問題では、復旧作業中に顧客データが他の顧客に流出するなど、さまざまな混乱が生じました（朝日新聞2012年7月7日）。

情報システムの規模が拡大する中で、私たちはシステムのディペンダビリティを従来通りにコントロールすることが難しい状況に直面しているかもしれません。では、どうすればよいのでしょうか。従来の耐故障技術、テスト、より精密な形式手法などは、これからますます重要になると考えられます。しかし、情報システムの規模が拡大し続ける中では、このアプローチだけでは限界があると考えます。つまり、障害を完全に回避することは難しいのです。私たちは、障害は完全には防げないという前提に立って、今後のディペンダビリティを考察しました（参考文献[3][4]）。

ディペンダビリティは、可用性や信頼性とは異なり、利用者を中心とした概念といえます。利用者や利害関係者に、障害が完全に防げなくてもディペンダブルであると感じてもらうにはどうすればよいでしょうか。私たちは以下の点が重要であると考えます。

- システム保証（System Assurance）：システムの安全性やディペンダビリティを利用者などの利害関係者に説明し、納得してもらうこと。
- 説明責任達成（Accountability Achievement）：システム開発や運用に際して、利用者などの利害関係者に必要な説明を正しく行うこと。特に障害が発生した場合には、その原因や事後対策、利用者への影響などを説明すること。

従来のディペンダビリティは、耐故障性など、主に工学的・技術的側面で研究されてきました。しかし、工学的・技術的なアプローチだけでは、今後のシステムのディペンダビリティを保つのは難しいかもしれません。そのため、他の手法も組み合わせる必要があります。私たちは、ディペンダビリティが利用者を重視した概念であることから、利用者や利害関係者に対するシステム保証と説明責任達成が重要であると考えました。

社会的にも、システム保証と説明責任の重要性は高まっています。例えば、2009 年から 2010 年に問題となったアメリカでのトヨタ車リコール問題では、最終的にトヨタ車に欠陥がなかったと報告されたにもかかわらず、トヨタは大きな批判を浴びました。この問題は政治的な背景が強かったといわれていますが [1]、参考文献 [5] は、トヨタに適切な説明体制が整っていなかったために問題が大きくなったと分析しています。また、2011 年に発行された自動車の機能安全規格 ISO26262 について、参考文献 [6] では「最大のインパクトは、安全性の根拠をより説明しやすくするよう自動車メーカーに求める点である」と指摘されています。ちなみに、ISO26262 では、D-Case のもとになったセーフティケースの提出が要求されています。

ディペンダビリティは、耐故障性の研究から始まり、安全性や信頼性を統合した概念として議論されてきました。しかし、近年の IT システムはあらゆる面で規模が拡大し続けており、そのディペンダビリティを達成するためには、従来の工学的アプローチだけでなく、システム保証や説明責任達成が重要であると考えています。本書で紹介する D-Case は、その観点から出発したものです。

最近では、AI（人工知能）技術を用いたシステム、特に自動運転システムなどのディペンダビリティが重要な課題となっています。AI 技術の不確実性や説明可能性の問題は、従来のシステムとは異なる新たなディペンダビリティの課題を提起しています。

システムのディペンダビリティは、もはや単なる技術的な問題ではなく、社会的、倫理的な問題としても捉えられるようになっています。システムの複雑化、不確実性の増大、そして AI 技術の台頭により、従来のディペンダビリティ

の概念や手法だけでは不十分になってきています。

これからのディペンダビリティ確保には、技術的な対策に加えて、システム保証と説明責任の遂行が不可欠です。また、不完全性や不確実性を前提としたシステム設計と運用の考え方を確立していく必要があります。

システム開発者、運用者、そして利用者を含むすべてのステークホルダーが、これらの課題を理解し、協力してディペンダブルなシステムの実現に取り組むことが求められています。

参 考 文 献

1.

2 安全分析の基本手法：FTA と FMEA

2.1 はじめに

本書では、ディペンダビリティ（Dependability）の新しい潮流として、STAMP/STPA などの先進的安全分析手法や、システムズエンジニアリング、アシュアランスケースといった概念を取り入れ、複雑化・高度化するシステムにどう対処していくかを議論します。しかしながら、これらの新たなアプローチを理解・活用するにあたっても、従来から用いられてきた**既存の安全分析手法**をしっかり押さえておくことは極めて重要です。そこで本章では以下の 2 つの手法を紹介します。

- **FTA (Fault Tree Analysis)**：重大事故や障害をトップ事象とし、どのような要因の組み合わせで発生するかを論理ゲートを用いて可視化する「トップダウン型」手法。
- **FMEA (Failure Mode and Effects Analysis)**：製品や工程の部品ごとに故障モードを洗い出し、それぞれがシステム全体に及ぼす影響を評価する「ボトムアップ型」手法。

これらはいずれも、システムや製品の**信頼性・安全性**を高めるうえで、長年にわたって活用されてきた「定番」の解析メソッドです。STAMP/STPA やアシュアランスケースといった新手法を導入する際にも、過去の事例や基準を踏まえたうえで、既存手法（FTA や FMEA）の成果物や解析結果を再利用した

り、組み合わせたりするケースが非常に多いと考えられます。

そこで本章では、まず **FTA** を取り上げ、その基本的な進め方や利点・留意点を解説し、その後、**FMEA** の概要を示します。これらの内容は、後続の章で扱うシステムズエンジニアリングやアシュアランスケース作成、あるいは STAMP/STPA の理解を深めるための基礎にもなるでしょう。

2.2　FTA (Fault Tree Analysis)

2.2.1　FTA の概要と目的

FTA は、発生してはならない重大な故障や不具合を「トップ事象」に設定し、なぜそれが起こるのかを下位に向かって論理的に掘り下げる**トップダウン型**の手法です。1979 年のスリーマイル島原子力発電所事故の解析で有効性が示されたことを契機に、原子力、自動車、航空宇宙、一般的な製造業など幅広い分野に広まっています。

- **特徴**：
  - 木構造（故障の木）を用いて、視覚的に「なぜ起こるか」を整理
  - AND/OR ゲートで複数要因・並列要因を表現可能
  - 大事故や重大障害の原因を重点的に深掘りするのに適している
- **注意点**：
  - トップ事象の選定を誤ると見落としが生じる
  - システム全体の網羅的な故障モード抽出には、FMEA など他手法との併用が望ましい

2.2.2　FTA の進め方

FTA は大きく以下の手順に従います。主に「故障の木 (FT 図) の作成」と「作成後の解析」の 2 段階です。

1. **FTA 実施の準備**
   - 解析対象システムを熟知するエンジニアや品質保証担当者など、3–6

名程度のチームを編成

- 関連設計資料・図面・クレーム情報などを集める

2. **解析対象の機能の理解**
   - 対象システムの構造・機能・使用環境・周辺システムとのインタフェースを共有
3. **トップ事象の選定**
   - 「明確に定義できる」「発生が望ましくない」「対策可能な要因が想定される」などの観点でトップ事象を決める
   - 例：**「エンジンが始動しない」「照明が点灯しない」「ブレーキが効かない」** など
4. **トップ事象の 1 次要因への展開**
   - トップ事象が起こる直接原因を OR/AND ゲート等で整理
   - 構成要素や機能的観点から、漏れなく候補を挙げる
5. **トップ事象の 2 次要因以下への展開**
   - さらに「なぜそうなるのか」を繰り返し、基本事象または非展開事象に至るまで木構造を描く
   - 要因の階層を揃え、曖昧な表現を避ける

2.2.3 FT 図を読む：記号とゲート

FTA では、**事象 (イベント)** を示す記号と、**論理ゲート** を使って因果関係を可視化します。

- **イベント記号**：
  - トップ事象（長方形）／中間事象（長方形）／基本事象（円）など
- **論理ゲート**：
  - AND ゲート：下位要因が「すべて発生」した場合に上位事象が成立
  - OR ゲート：下位要因の「いずれか 1 つが発生」した場合に上位事象が成立

2.2.4 FT 図の定量的解析

FT 図を作成したら、以下の定量解析を行う場合があります。

- **トップ事象の発生確率**：
  - OR ゲート → 下位事象確率の和（近似）
  - AND ゲート → 下位事象確率の積（独立性前提）
- **同一事象の排除**：
  - 同じ基本事象が複数箇所に重複する際、定量計算時には一つにまとめるなど配慮が必要
- **重要事象の抽出**：
  - OR ゲート配下では、確率が高い事象から優先的に対策
  - AND ゲートを含む複合形状では別の分析手段（最小カット集合など）と併用

2.2.5 FT 図の定性的解析

発生確率の情報がない、あるいは推定困難な場合には、定性的な方法で「主要な要因」を特定することが可能です。

- **最小カット集合 (Minimal Cut Sets)**：
  - トップ事象を発生させる「最小限の基本事象の組み合わせ」を抽出
  - 共通して含まれる基本事象があれば、そこを対策することでトップ事象発生を大きく防止可能
- **構造重要度 (Structure Importance)**：
  - ある基本事象が故障／正常化したときにトップ事象の発生パターンがどれくらい変化するかを計算
  - トップ事象の回避への寄与度が高い基本事象を抽出する

2.2.6 FTA 実施上の留意点

- **全ての基本事象の発生確率把握は難しい**
  - 信頼性データが乏しい場合、定性的評価や他手法との併用が有効

- **創発故障や相互作用に注意**
  - 単に構造分割するだけでなく、機能の流れや外部環境の影響も考慮
- **動的変化を扱いにくい**
  - FTA は基本的に静的モデルであり、時間依存のリスク変化には別途検討が必要
- **事後解析でも有効**
  - 実際に発生した故障・事故の原因を特定し、最小カット集合などで要因を絞る手段としても活用される

2.3 FMEA (Failure Mode and Effects Analysis)

続いて、FMEA について簡単に紹介します。FMEA は **ボトムアップ型** で故障モードを分析する手法であり、FTA と組み合わせることで、システム全体の安全分析をより強固にする狙いがあります。本書では後の章で扱う STAMP/STPA やアシュアランスケースとも対比しながら、従来手法との使い分け・組み合わせを検討します。

2.3.1 FMEA の概要と目的

- 故障モード (Failure Mode) を部品・工程レベルから列挙し、それが上位システムへ及ぼす影響 (Effects) を評価する未然防止手法
- NASA での宇宙開発プロジェクトや自動車、家電、医療機器など多様な領域で実績がある

2.3.2 FMEA の進め方 (概略)

1. **準備とチーム編成**
2. **解析対象の機能・構造把握**
3. **故障モードの抽出**
4. **影響評価 (Severity / Occurrence / Detection) と優先度 (RPN)**

5. **対策検討とフォローアップ**

2.3.3 FMEA の利用事例とまとめ

- **工程 FMEA**：製造工程や組立工程における不良モードを系統的に洗い出す
- **設計 FMEA**：製品の詳細設計時に、部品・サブシステムの故障モードを網羅し、上位への影響を検討
- これらの **FMEA** と **FTA** を組み合わせることで、網羅性（FMEA）と重大事象の重点把握（FTA）が両立できる

3 安全分析手法STAMP/STPA

3.1 STAMP の概要

STAMP（System-Theoretic Accident Model and Processes）は、システム理論に基づく新しい事故モデル（Accident Model）です。従来の事故モデルが主にコンポーネントと呼ばれるシステムの構成要素の故障（Failure）に焦点を当てているのに対し、STAMP はシステム全体の安全制約とその制御に注目します。

STAMP の基本要素は以下の 3 つです：

- 安全制約（Safety Constraint）：安全が守られるためにシステムやコンポーネントが守る必要があるルール。
- プロセスモデル（Process Model）：コントローラが認識するコントロール対象の状態。
- 安全制御構造（Safety Control Structure）：コンポーネントとそれらの間の相互作用（制御関係とフィードバック関係）を機能レベルで表したシステムのモデル。

STAMP では、事故を単純なイベントの連鎖ではなく、安全制約が安全制御により適切に守られなかった結果として捉えます。

3.2 STPA

STPA（System-Theoretic Process Analysis）は、STAMP に基づくハザード分析手法です。システムの設計段階や運用開始前に潜在的なハザードを特定し、安全制約を導出するために使用されます。

3.2.1 STPA の手順

STPA は以下の 4 つのステップで構成されます (STPA Handbook(2018))：

- Step 1: 分析目的の定義
- Step 2: 安全制御構造図（Safety Control Structure Diagram）のモデル化
- Step 3: 非安全制御動作（Unsafe Control Action）の識別
- Step 4: ロスシナリオ（Loss Scenario）の識別

以下の各項で、STPA の各ステップについて解説します。

3.2.2 Step 1: 分析目的の定義

Step 1「分析目的の定義」では、以下を項目を実施します：

- ロスの識別
- システムレベルのハザードの識別
- システムレベルの安全制約の識別
- ハザードの詳細化（任意）

各項目について解説します。

ロス（Loss）は、ステークホルダにとって受容できない何かを表します。ロスの例としては、人命の喪失、人の負傷、環境汚染、ミッション失敗等が挙げられます。以前の STPA の解説ではアクシデント（Accident）も識別されていましたが、STPA Handbook(2018) からは、アクシデントを識別するよう記載されていません。しかし、ロスと合わせてアクシデントを識別することで、後

述するハザードが識別しやすくなるため、ここではアクシデントについても触れることにします。アクシデントは、望まれない・計画されていないイベントで、ロスへ至るものです。アクシデントの例として、自車両が前方障害物に激突等が挙げられます。

システムレベルのハザード（Hazard）は、システムの状態または条件の集まりで、最悪の環境条件下で、ロスへ至る蓋然性が高いものです。ハザードはシステムの状態または条件の集まりであるため、システム境界外の状態または条件の集まりを直接は扱えません。例えば、「自車両が（外部環境である）前方障害物に衝突した状態」はハザードではありません。この場合、例えば「（自車両内の前方障害物検出用の）距離センサの値が規定値未満である状態」をハザードとします。また、ハザードは必ずしもロスへ至るわけではない点にも、注意が必要です。例えば、距離センサの値が規定値未満であったとしても、運転者の回避能力が高い場合には前方障害物を回避でき、人命の喪失や負傷へ至らないかもしれません。さらに、自動車である以上不可避な状態である「自車両が走行中である状態」も、最悪の環境下でロスへ至りますが、ロスへ至る蓋然性が低い（ロスに至るまでの追加条件が多い）ため、ハザードしては扱いません。

システムレベルの安全制約は、システムレベルの条件・動作で、ハザードを防ぐためにシステムが満たす必要のあるものです。素朴な安全制約はハザードの裏返しとして定義でき、またハザード発生時にロスを防ぐあるいは最小化する条件・動作としても定義できます。例えば、自動運転システムを開発・分析しているときには、ハザード「距離センサの値が規定値未満である状態」を防止するために、「自動運転システムは、距離センサの値が規定値未満になったら、自車両を安全に停止させなければならない」といった安全制約が考えられます。

ハザードの詳細化（任意）では、…

一般に、多数のロス、ハザード、安全制約が識別されるため、ロス、ハザード、安全制約は、番号を付け、ハザードには対応するロスの番号を含めることで、分析結果の整理が容易になります。

（1） 演習: ロス、ハザード、安全制約の識別 システムの説明を記述し、

図を挿入すること。

1. 上のシステムに対するロス及びアクシデントを識別しなさい。(解答例：乗員の死亡・負傷（自車両が前方障害物に衝突))
2. (i) で識別したロスに対し、ハザードを識別しなさい。(解答例：(前方障害物検出用の）距離センサの値が規定値未満)
3. (ii) で識別したハザードに対し、安全制約を識別しなさい。(解答例１：距離センサの値が規定値未満になってはならない。解答例２：自動ブレーキシステムは、距離センサの値が規定値未満にならないよう、ブレーキを指示しなければならない。)

3.2.3 Step 2: 安全制御構造図のモデル化

Step 2「安全制御構造図のモデル化」では、安全制御構造図を構築します。安全制御構造図は、システム内の構成要素（コンポーネント）間の制御関係とフィードバック関係を示すシステムのモデルで、コントロール・ループ (図 3.1) により構成されます。ここで、コントロール・ループは以下の構成要素により構成されます。

- コントローラ：制御する側のコンポーネント。
- コントロールアルゴリズム：コントローラの判断決定プロセス。コントローラは、コントロールアルゴリズムに基づき、コントロールアクションを指示します。
- プロセスモデル：コントローラの内部情報で、判断の際に参照される情報。プロセスモデルは、コントローラが認識する被コントロール・プロセスや外部環境の状態であり、フィードバックにより更新されます。
- 被コントロール・プロセス：制御される側のコンポーネント。
- コントロールアクション：被コントロール・プロセスの動作や制約を実行させるためのコントローラが出す命令・指示。
- フィードバック：被コントロール・プロセスや外部環境からコントローラに送られてくる情報。

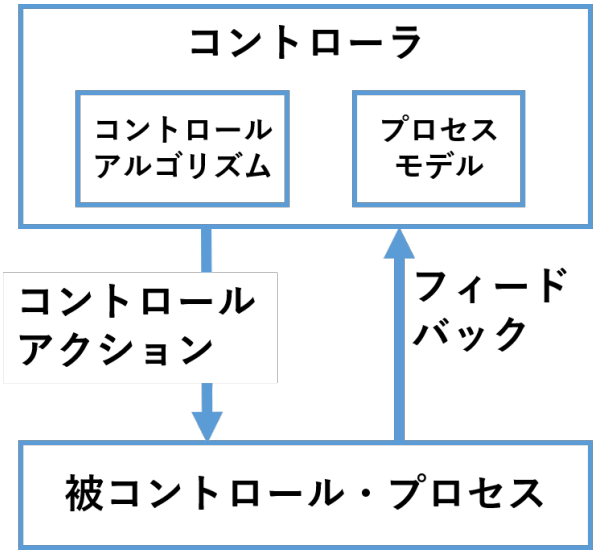

図 3.1 コントロール・ループ

図 3.2 は、赤線で囲まれた二つのコントロール・ループにより構成される安全制御構造図の例です。安全制御構造図では、制御する側を上に、制御される側を下にして記述します。また図 3.2 には、アクチュエータとセンサを記載してありますが、比較的新しい文献では、安全制御構造図を構築する際にはこれらを記載せず、後の分析（Step4）でこれらを追加して分析をするようになっています。

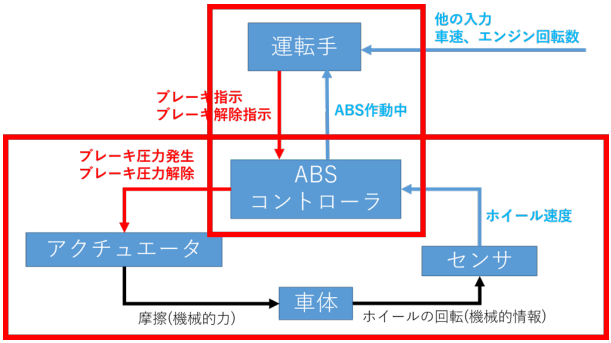

図 3.2　例：二つのコントロール・ループにより構成される安全制御構造図

安全制御構造図は、物理的設計レベルではなく、機能レベルでのシステムのモデルです。また、安全制御構造図はモデルですので、安全機構に関係無い機能は省略します。

安全制御構造図を基に分析することで、例えば、センサが故障した結果、コントローラが間違ったフィードバックを受信して、コントローラが誤った判断を下すといった事故のシナリオが識別できます。また、プロセスモデルを考えることで、実際は自車両の前方に歩行者がいるのに、コントローラ（自動運転車）は歩行者がいないと認識しており、その結果、事故に至るといったシナリオを分析しやすくなります。さらに、コントローラが判断する際に必要なフィードバックが不足していたといった重大な欠陥を早期に見つけやすくなります。

機械以外にも、人間や組織をコントローラとする場合もあります。この場合、コントロールアルゴリズムは人間の判断プロセスに、プロセスモデルは人間の認識になります。また、（分析対象システムの外側である）外部環境からのフィードバックを考えることもあります。例えば、自動運転車の安全制御構造図では、運転者を最上位のコントローラとし、運転者へのフィードバックとして、目視情報（自車両の前方に歩行者がいる・いない等）を考えます。分析の際に、安全制御構造図に人間や組織を含められるため、人間のミスや、他組織からの外圧を含めた事故のシナリオを考えられるようになります。

（1） **演習: 安全制御構造図の構築** システムの説明を記述し、図を挿入すること。

3.2.4 Step 3: 非安全制御動作の識別

Step 3「非安全制御動作の識別」では、非安全制御動作（UCA: Unsafe Control Action）を識別し、可能であればコントローラ制約を定義します。UCA は、特定のコンテキストと最悪の環境の下で、ハザードを引き起こす可能性のある制御動作で、コントローラ制約は UCA を引き起こさないために満たす必要のあるコントローラへの動作（制約）です。

UCA は文章として表すことが多いですが、

「制御動作を出すコントローラ」＋「タイプ」＋「制御動作」＋「コンテキスト」＋「ハザード」

の組として考えると、UCA を識別しやすくなります。このとき、タイプとして以下の 4 つを用います：

1. 与えられないとハザード
2. 与えられるとハザード
3. 早すぎ、遅すぎ、誤順序でハザード
4. 早すぎる停止、長すぎる適用でハザード

また、コンテキストは制御動作が非安全となる条件を表します。

UCA の例として、「運転者からのブレーキ指示があり、タイヤがロックしていないにもかかわらず、ABS コントローラがブレーキ圧力発生を指示しないため，前方障害物との距離が規定値未満になる．(H3)」を考えます。コントローラ「ABS コントローラ」に対するこの UCA 対するコントローラ制約として、この UCA を否定形である「運転者からのブレーキ指示があり、タイヤがロックしていないときには、ABS コントローラがブレーキ圧力発生を指示する。」を考えることができます。このとき、この UCA は以下のように分解されます：

- 制御動作を出すコントローラ：ABS コントローラ
- タイプ：与えない

- 制御動作：ブレーキ圧力発生
- コンテキスト：運転手からのブレーキ指示があり、タイヤがロックしていない

制御動作を出すコントローラ、タイプ、制御動作、ハザードは既に識別されているため、それらを組み合わせて考えることで、網羅的な分析が可能になります。しがって、コンテキストを識別することが最も重要となります。このとき、制御動作を出すコントローラの入力に着目すると、コンテキストが考えやすくなります。また識別した UCA に番号を付け、以下のような表形式で表すと可読性が高まり、後の分析が容易になります。

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	早すぎ、遅すぎ、誤順序でハザード	早すぎる停止、長すぎる適用でハザード
ブレーキ圧力発生	**運転者からの指示があるのに、**ABSが指示を出さない(H3)			
ブレーキ圧力解除				

図 3.3　例：UCA の表

コンテキストが無いにもかかわらず UCA となる場合や、同じコンテキストの下で与えても、与えなくても UCA となる場合には、その制御動作は設計には問題があると考えられます。

（1）　**演習:UCA の識別**　　システムの説明を記述し、図を挿入すること。

3.2.5　Step 4: ロスシナリオの識別

Step 4「ロスシナリオの識別」では、Step 3 で識別した UCA に対してロスシナリオを識別します。ロスシナリオ（Loss Scenario）は、UCA、ひいてはハザードへ至る要因を記述したシナリオです。ロスシナリオの中で具体的要因を識別するので、アクチュエータとセンサを入れたコントロール・ループを基に分析します。

大きく分けて、以下の 2 種類のロスシナリオを考えます：

- UCA へ至るシナリオ

- 制御動作の不実行・不適切な実行を表すシナリオ

「UCA へ至るシナリオ」では、図 3.2.5.1 の右上部分に着目し、なぜ UCA が発生したのかを考えます。UCA が発生する一般的な要因としては、コントローラの非安全な動作や、不適切なフィードバック・（他コントローラ等からの）入力が考えられます。また「制御動作の不実行・不適切な実行を表すシナリオ」では、図 3.2.5.1 の左下部分に着目し、なぜ制御動作は不適切に実行されたのか、なぜ制御動作は実行されなかったのかを考えます。「制御動作の不実行・不適切な実行が起こる一般的な要因としては、アクチュエータやコントロールアクションの伝達経路上での問題、被コントロール・プロセスに関する要因が考えられます。STPA Handbook(2018) やはじめての STAMP/STPA(2016) には、ロスシナリオを識別する際のヒントとして、一般的要因のリストが例示されています。このような一般的要因や過去の知見から得られている独自要因のリストは、ロスシナリオの識別に有用です。しかし、このようなリストを一通り考察したら分析を終えるのではなく、他の要因が無いかを検討することが肝要です。

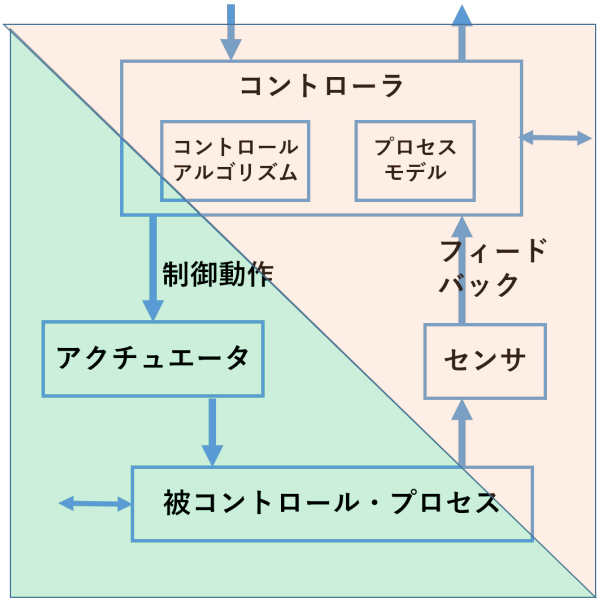

図 3.4　考えるべきロスシナリオ

（ハザードからロスシナリオまで一貫性・妥当性のある例に変更すること！）

UCA「運転者からのブレーキ指示があり、タイヤがロックしていないにもかかわらず、ABS コントローラがブレーキ圧力発生を指示しないため，前方障害物との距離が規定値未満になる。」を考えます。この UCA に対するロスシナリオ、特に UCA へ至るシナリオの例としては、「センサからのフィードバックが間違っていたため、ABS コントローラがタイヤがロックしていないと誤認識してしまい、ABS コントローラがブレーキ圧力発生を指示しない。」が考えられます。このように現実とプロセスモデル（コントローラの認識）が異なる状況は、プロセスモデルの不一致と呼ばれ、ロスシナリオを識別する際に、しばしば登場します。また、制御動作の不実行・不適切な実行を表すシナリオの例としては、「アクチュエータが故障していたため、ABS コントローラがブレーキ圧力発生を指示したにもかかわらず、アクチュエータがブレーキ圧力を発生しないため、自車両が減速せず障害物に衝突する。」が考えられます。

3.3　脅威分析手法 STPA-Sec

STPA-Sec は、STPA を拡張したハザード分析手法です。STPA-Sec に関する資料としては、STPA-Sec(2020) や福島 (202X) がありますが、STPA と比較すると参考になる資料は少なく、STPA Handbook のような詳細な工程や分析に役立つノウハウは共有されていません。そこで、この章には、上記の資料以外に、著者が STPA-Sec を実施した際に有用であった内容を追加しています。

STPA-Sec(STPA for Security) は STAMP/STPA の利点を継承しています。例えば、コントロールストラクチャ図に運転者や組織をコンポーネントとして含めることで、機械の故障以外の要因が扱いやすくなります。また、コントロールストラクチャ図を構築し、その中のコントロールアクションに対し網羅的に UCA を識別することで、分析結果の網羅性を保証できます。他方、初めに損失・ハザードを識別し、UCA 経由でロスシナリオを識別することで、事故の要因と結果を容易に結び付けらることができ、下の図に示されるように、戦略から戦術への対応が明確になります。さらに、コンセプト段階から分析できるため、攻撃を早期に軽減または排除できます。

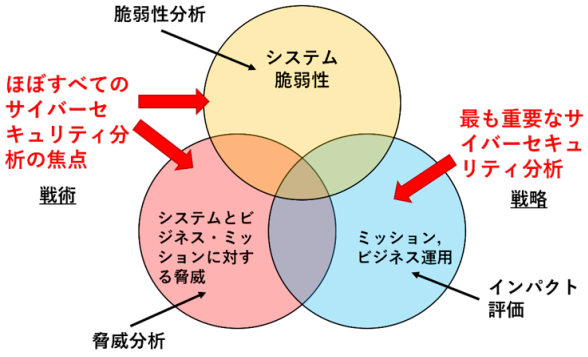

図 3.5　サイバーセキュリティ分析の 3 つの種類 (STPA-Sec2020 より引用)

STPA-Sec の手順は、STPA の手順にいくつかの工程を加えた以下の手順になります。

- Step 1 問題設定＋分析目的の定義
- Step 2 安全制御構造図のモデル化
- Step 3 非安全制御動作の識別 (Identify Hazard Control Actions)
- Step 4 ロスシナリオの識別 ＋ セキュリティシナリオの識別 ＋ ウォーゲーム（Wargaming）

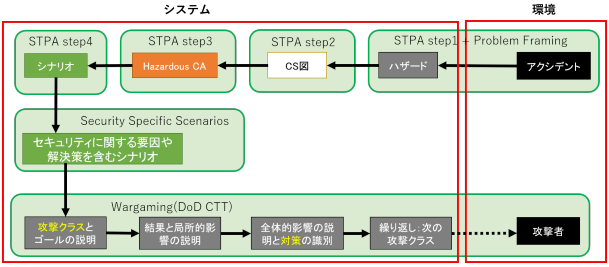

図 3.6　STPA-Sec Wargaming Using DoD Cyber Table Top

STPA-Sec Step 1 では、問題設定（Problem Framing）に加え、STPA Step 1 の分析目的を行います。STPA Step 1 では、安全に関する損失とハザードを識別しますが、STPA-Sec Step 1 では、セキュリティに関する損失とハザードも識別します。例えば、損失として会社の信用が、ハザードとして権限を持たない人が（情報システム内の）個人情報を閲覧可能な状態などがあげられます。

問題設定では、システムが何をすることになっているかを簡潔に説明する文章を構成します。この文章には、ステークホルダーとの会話と資料から抽出した目的 (purpose)、方法 (method)、目標 (goal) や制約/制限 (Constraints/Restraints) に加え、作成した機能モデル (functional model) の記述が含まれます。この文章の様式は、「{Why ＝目標} に貢献するために、{What ＝目的} を {How ＝方法} によって行うシステム。ただし、{While=制約/制限} を満たすこと。」のようになります。問題設定で構成する文章の例としては、「Why {制動距離を

短くすること｝に貢献するために、What｛タイヤがロックして滑り始めたら、ブレーキを自動的に緩めること｝を How｛ホール速度を計測し、タイヤのロックを検知し、ブレーキ圧力開放を指示 (ブレーキ圧力発生を解除)｝によって行うシステム。ただし、Where｛走行の安全の確保｝を行うこと。」等が挙げられます。

STPA-Sec Step 2 では、STPA Step 2 と同様に安全制御構造図を構築します。今回は、以下の安全制御構造図を考えることにします。

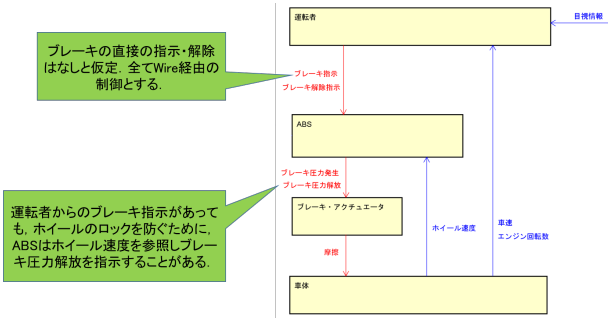

図 3.7　STPA-Sec で使用する安全制御構造図の例

STPA-Sec Step 3 では、STPA Step 3 と同様に非安全動作を識別します。STPA の場合、非安全動作（Unsafe Control Action）という用語を使いましたが、STPA-Sec では、安全とセキュリティを両方分析するため、Hazardous Control Action という用語が使われています。本書では、どちらも非安全動作という訳語としています。

上の図中のコントロールアクション「ブレーキ圧力発生」に対する UCA の例（もっと整合的・現実的な例に置き換えること）としては、

- HCA-P1：運転者からのブレーキ指示がないのに、ABS がブレーキ圧力発生を指示してしまい、後続車両に追突される。
- HCA-N1：運転者からの指示があるのに、ABS がブレーキ圧力発生を指示しないため、前方障害物に衝突する。

が考えられます。次の Step で UCA へ至るシナリオを識別しますが、HCA-N1 に対するシナリオでは、安全要因が UCA を引き起こすかもしれないし、セキュリティ要因が UCA を引き起こすかもしれません。

STPA-Sec Step 4 では、STPA Step 4 と同様に、ロスシナリオを識別します。なお、セキュリティに係る要因もロスシナリオに含めるため、セキュリティに係るヒントを追加したり、コントロールループ図に加え、セキュリティ分析で使用する図も参照すると有効です。

例えば、HCA-N1 に対するロスシナリオを識別します。このとき、プロセスモデルの不一致「ABS がホイール速度を誤認識 (ロック状態であると誤認識)」を基に、ロスシナリオの識別を進めることにします。なお、プロセスモデルの不一致「ABS がホイール速度を誤認識 (ロック状態であると誤認識)」が起こると、「走行中に、ABS がホイール速度を誤認識 (ロック状態であると誤認識) してしまい、ブレーキ圧力発生を指示せずにブレーキ圧力解放を指示を出して、HCA-N1 へ至る」となるため、プロセスモデルの不一致の要因も識別すれば、ロスシナリオが識別されます。この誤認識は不適切なコントロールアルゴリズム「ABS は運転者からのブレーキ指示があり、かつホイール回転速度からロック状態でないと認識しているときには、ブレーキ圧力発生を指示しない」により引き起こされます。さらにこの不適切なアルゴリズムが実装された理由としては、ソフトウェアの動作検証が不十分であることや、外部からの不適切なソフトウェアの更新といったセキュリティ要因が考えられます。

STPA-Sec Step 4 では、次にセキュリティシナリオを識別します。セキュリティシナリオの定義は明確ではありませんが、「しばしばロスシナリオと同じであるが、セキュリティに関連する原因 (cause) と解決策 (solution) を含むことがある」とあることから、ロスシナリオの要因と対策部分としてセキュリティに関連する要因と解決策を識別したシナリオであると考えられます。既にロスシナリオが識別されているため、ロスシナリオを含む STPA の分析結果を利用できます。さらに、セキュリティの分析で利用されている、STRIDE などの攻撃の分類を用いることで、シナリオの要因部分を識別できます。セキュリティ

と安全のかかわり方としては、セキュリティに関する要因が安全を脅かすというシナリオが主に思い浮かびますが、この逆も考えらえます。

例えば、ロスシナリオと同様に、HCA-N1 に対するセキュリティシナリオを識別します。このとき、プロセスモデルの不一致「ABS がホイール速度を誤認識 (ロック状態であると誤認識)」を基に、ロスシナリオの識別を進めることにします。今回は、コントロールアルゴリズムは適切「ABS は運転者からのブレーキ指示があり、かつホイール回転速度からロック状態でないと認識しているときには、ブレーキ圧力発生を指示する」であるとします。しかし走行中に、ABS が誤った入力「ブレーキ解除指示 (診断モード)」と「車速は低速」を受信したとします。すると ABS の仕様から、ABS はブレーキ圧力発生を指示せずに、ブレーキ圧力開放を指示してしまい、HCA-N1 へ至ります。またこの誤った入力は、入力の改ざんにより引き起こされます。

3.3.1 ウォーゲーム (Wargaming)

STPA-Sec Step 4 では、最後にウォーゲーム（Wargaming）を実施します。ウォーゲームの定義は明確ではありませんが、ウォーゲームの DoD Cyber Table Top (CTT)（参考文献）による実施例から、資産（asset）や侵害されるサイバーセキュリティ・プロパティの識別、攻撃経路 (attack path) の分析、攻撃実現性 (attack feasibility) の評価を実施していることが分かります。

ウォーゲームでは以下の内容を実施します。

- STPA-Sec(または STPA) で特定されたシナリオを引き起こす攻撃クラスを評価する。
- コントロールのクラスの潜在的な有効性を評価する。
- 攻撃クラスの実行の複雑さを理解する。
- 検討中の機能や特徴に関連する運用リスクの評価をサポートする。

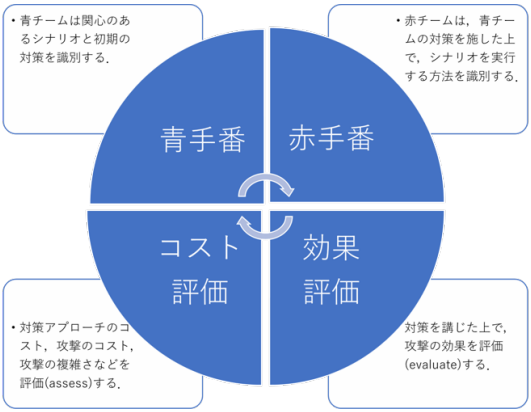

図 3.8　STPA-Sec Wargaming Approach（STPA-Sec2020 より引用）

これらの内容を実現するための、具体的な手順や作業内容については別の資料を参照することとされています。ここでは、文献 STPA-Sec で紹介されている DoD CTT（参考文献）を用いた例を紹介します。（後ろで参照しやすくするため、各項目には WG 数字の形式で番号を付けることにします）

（1）　STPA-Sec Wargaming Using DoD Cyber Table Top(CTT)

- WG1：OPFOR(Opposing FORce、仮想敵部隊、赤) が大まかな攻撃クラスとゴールを記述する。
- WG2：両チームは成果 (outcome) と効果 (effect) を記述する。
- WG3：作戦チーム (operations team、青) が作戦効果 (mission effects) を説明し、回避策 (workarounds) を記述する。
- WG4：次のクラスの攻撃を繰り返す。

さらに、WG1 のうち攻撃クラスの記述は以下の項目に細分化され、下線部を識別します。

- WG1-1：コントロールストラクチャを検証することで、潜在的な攻撃の大まかな分類についてさらなる洞察が得られる。

- WG1-2：攻撃がどの要素 (element(s)) に影響するかを検討・識別する。また、影響を受ける要素 (element impacted) と呼ばれる攻撃が影響する要素を識別する。
- WG1-3：STRIDE を使用して、影響を受ける要素に対する攻撃のうち、エフェクト (effect) の原因となるものを識別する。（このとき、エフェクトの原因となる攻撃を攻撃クラス (attack class) と呼ぶ。）
- WG1-4：シナリオに戻り、攻撃の目的（goal of the attack）は何かを検討する。
- WG1-5：OPFOR は「条件なしの HCA」を引き起こしてはならないが、条件が満たされている場合は HCA を引き起こさなければならない。

例として、HCA-N1「運転者からの指示があるのに、ABS が指示を出さないため前方障害物に衝突する」と、この HCA に対するセキュリティシナリオ「走行中に、ABS が誤ったメッセージ「ブレーキ解除指示 (診断モード)」と「車速は低速」を受信したため、ABS がブレーキ圧力発生を指示せずに、ブレーキ圧力開放を指示して、HCA3-N1 へ至る」を考えることにします。

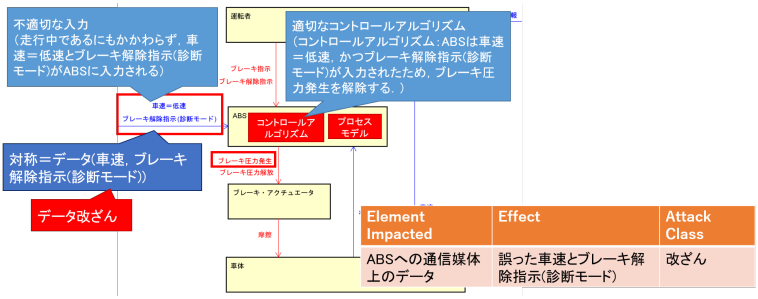

図 3.9　WG1 の攻撃クラスの識別の例

セキュリティシナリオと安全制御構造図から、影響を受ける要素として「ABS へのメッセージ（入力）」が識別され、このときの影響は「誤ったメッセージ「ブレーキ解除指示 (診断モード)」と「車速は低速」」となります。これらの影響を受ける要素と影響に対し、STRIDE を使用すると、有効な攻撃クラスとし

て「改ざん」が識別されます。他方、WG1 のゴールの識別では、攻撃成功の条件と攻撃のゴールを識別します。今回の例では、攻撃成功の条件（コンテキスト）としては「走行中」が識別でき、攻撃のゴールとしては「ABS がブレーキ指示を出さずにブレーキ解除指示を出す」が識別できます。

WG2 では、安全制御構造図に基づいて、攻撃側は「攻撃クラス」を起動するために、いつ「コンテキスト」が成立するかを決定します。このとき、基本ルール 1）制御構造は“土俵 (play ground)”である、2）現在の抽象的なレベルにとどまる、3）前提条件を明示する、4）攻撃側はコンテキスト要件を遵守する、を守るようにします。今回の例では、攻撃クラス「改ざん」とコンテキスト「走行中」としていたので、ホイール速度、車速、エンジン回転数から走行中であかどうかを決定できます。

WG3 では、作戦チームが攻撃の効果をシステム全体 (ミッションレベルまで) で評価し、回避策 (workarounds) を記述する。攻撃効果の評価には、ロスシナリオ・セキュリティシナリオに対応するハザード・損失を用いることができます。また、脅威クラスに対処するための 4 つの大まかな回避策のアプローチとしては、軽減 (mitigate)、排除 (eliminate)、転送 (transfer)、受け入れる (accept) があります。なお、回避策を策定する際には、現在の詳細レベルとの一貫性を保つ必要があります。

WG4 では、次の攻撃クラスに対しここまでの手順を繰り返します。今回の例では攻撃クラス「改ざん」を分析したので、例えば、別の攻撃クラス「DoS」を分析します。

3.4　事例研究: 自動運転システムへの STPA の適用

ここから再開 2024-09-23 ページ数的に、国際規格への対応、特にリスク評価の実施について書けばよさげここでは、自動運転システムに STPA を適用する例を示します。

3.4.1 Step 1: 分析目的の定義

- ロス: 人命の損失、車両の損傷
- ハザード: 自車両と他の物体（車両、歩行者、障害物など）との距離が安全距離未満になる
- 安全制約: 自車両は常に他の物体との安全距離を維持しなければならない

3.4.2 Step 2: 制御構造図のモデル化

[自動運転システムの制御構造図を挿入]

3.4.3 Step 3: 非安全制御動作の識別

UCA の例：

- UCA1: 前方に障害物があるにもかかわらず、自動運転システムがブレーキを適用しない
- UCA2: 安全な状況下で自動運転システムが不必要にブレーキを適用する

3.4.4 Step 4: ロスシナリオの識別

ロスシナリオの例：

- LS1: センサーの故障により、自動運転システムが前方の障害物を検知できず、ブレーキを適用しない
- LS2: ソフトウェアのバグにより、自動運転システムが安全な状況を危険と誤認識し、不必要にブレーキを適用する

3.5 ま と め

STAMP/STPA 及び CAST は、現代の複雑なシステムの安全性分析に適した手法です。これらの手法を用いることで、以下のような利点が得られます：

- システム全体の安全性を包括的に分析できる
- 人間要因を含むシステムの相互作用を考慮できる

- 事故の根本原因だけでなく、システムの改善点を特定できる
- 設計段階から運用段階まで、システムのライフサイクル全体にわたって適用できる

これらの手法を効果的に活用することで、より安全で信頼性の高いシステムの開発と運用が可能になります。

3.6 参 考 文 献

- Engineering a Safer World, 2012
- STPA Handbook, 2018
- はじめての STAMP/STPA, 2016
- はじめての STAMP/STPA(実践編), 2017
- はじめての STAMP/STPA(活用編), 2018
- STAMP ガイドブック ～システム思考による安全分析～, 2019
- STAMP Workbench
- BASIC INTRODUCTION TO STPA FOR SECURITY (STPA-SEC), 2020 SYSTEM-THEORETIC ACCIDENT MODEL AND PROCESSES (STAMP) WORKSHOP, July 22, 2020, William “Dollar” Young, Jr (Ph D)
- System-Theoretic Process Analysis for Security (STPA-SEC): Cyber Security and STPA, WilliamYoung Jr, PhD, 2019 STAMP Conference Boston, MA, March 25, 2019
- BASIC INTRODUCTION TO STPA FOR SECURITY (STPA-SEC), 2020 SYSTEM-THEORETIC ACCIDENT MODEL AND PROCESSES (STAMP) WORKSHOP, July 22, 2020, William “Dollar” Young, Jr (PhD)
- System-Theoretic Process Analysis for Security (STPA-SEC): Cy-

ber Security and STPA William Young Jr, PhD Reed Porada 2017 STAMP Conference Boston, MA March 27, 2017

- 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」, ユニシス技報, vol.41, no.2, Sep.2021

4 モデルベースシステムズエンジニアリング

4.1 システムズエンジニアリングの基礎

4.1.1 システムズエンジニアリングとは

システムエンジニアリング (systems engineering) は、分野に依存せず「システム」を構築したり発展させたりするための知見をまとめたものです。従来は、製品やサービスの領域毎に、開発プロセスは発展し、その中での最適化が行われてきました。しかし、多くのシステムが大規模化、複雑化するなかで、領域毎に発展してきた方法論では、必要な納期を満たせなかったり品質を保証することが難しくなってきました。一方で、そのような課題は、製品やサービスに依存しない現象として説明したり、さらにはその解決アプローチについても、一般化して検討できるかもしれない、と気付いた人たちがいました。そのような知見をまとめたものがシステムズエンジニアリングであり、実際多くの現場でその効果が確かめられたため、国際標準化なども含めて普及が進んでいます。

システムズエンジニアリングは例えば次のように定義されます ([JIS X 0170])。

> 利害関係者のニーズ、期待及び制約の集合を、解決するソリューションへ変換するため、及びソリューションが用いられる全期間を通じて、それを支援するために要求される，技術上及び管理上の作業の全体を統括するような、複数の専門分野を横断した取組方法。

一つ一つ見ていきましょう。

利害関係者のニーズ、期待及び制約の集合を、解決するソリューションへ変換する

はじめに、システムズエンジニアリングの目的が言及されています。ここでは、利害関係者がそれぞれのニーズを持っており、さらに明示化されていない期待や、実現に対して課せられる制約を考慮した上で、ソリューションを得ることであることが分かります。このソリューションで中心的な役割を果たすのが典型的にはシステムとなります。

及びソリューションが用いられる全期間を通じて、

「全期間」が対象になります。例えば、ニーズを集めるところや企画段階や、ソリューションの運用が終わったあとの破棄などの期間も対象となります。

それを支援するために要求される，技術上及び管理上の作業の全体を統括するような、

ソリューションを支援するための、管理上必要となる作業までが対象であることが分かります。例えば、ソリューションの運用に人系が必要であれば、人的リソースの管理なども対象に含まれます。

複数の専門分野を横断した取組方法。

例えば、機械分野や建築分野、電気分野などは、それぞれの分野に特化した方法論があるかもしれませんが、システムズエンジニアリングで目指すのは、複数の専門分野の知見が必要となるようなソリューションに対して分野横断的に適用可能な取組方法であることが分かります。

4.1.2 システムエンジニアリングの知見

システムエンジニアリングにはさまざまな知見が含まれますが、ここでは次の三つを紹介します：

1. システムに関する概念の定義
2. システムの開発や運用においてありうる活動の定義
3. システムを記述するために必要な観点の提供

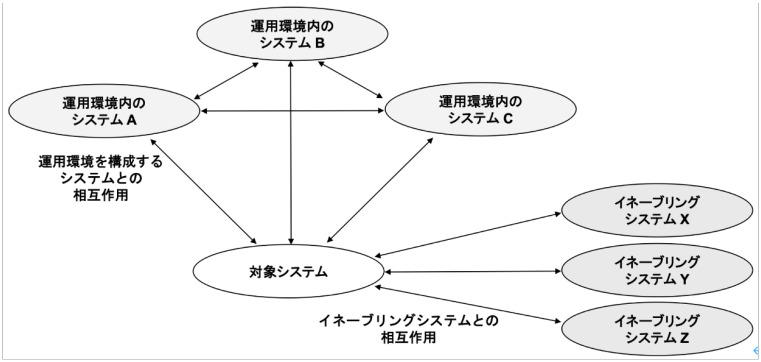

図 4.1　イネーブリングシステム

4.1.3　システムに関する概念の定義

前述の通り、システムが大規模化、複雑化すると、複数の製品やサービス領域を組み合わせることが一般的になり、これまで交流のなかった技術者も共同作業をしなければなりません。そのようなときに共通言語を与えるのも、システムズエンジニアリングの役割です。また、大規模かつ複雑なシステムは、その開発や運用プロセスも複雑になり、その計画を立てたり、人員を割り当てる際にも活動に関する共通言語が必要になります。それら開発や運用に係わる活動に関する概念も、提供する共通概念の重要な部分です。

例えば、ジェットエンジンを開発するときには、巨大な試験施設が必要になります。そのような施設もシステムであり、それ自体がライフサイクルを持ち、試験対象であるジェットエンジンのライフサイクルも意識しながら施設を開発し、運用する必要があります。開発や運用対象のシステムを対象システム（system-of-interest）と呼ぶのに対し、開発時に連携が必要となるシステムをイネーブリングシステム（enabling system）と呼びます（**??**）。

4.1.4　システムの開発や運用においてありうる活動の定義

システムエンジニアリングでは、システム開発に関わる活動を分類し、標準化したものとしてシステムライフサイクルプロセス（system lifecycle processes）

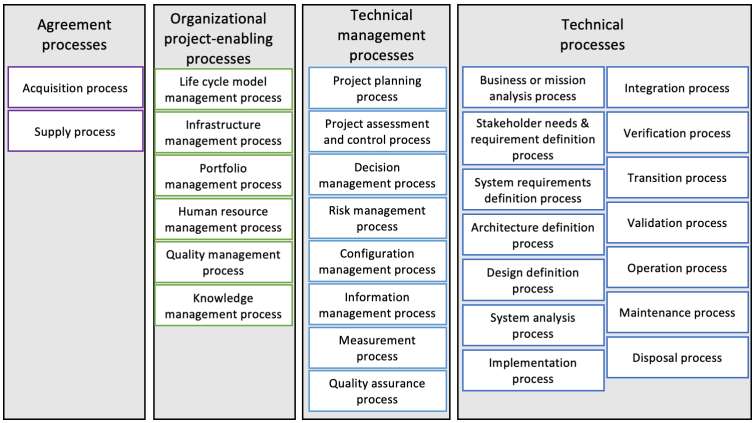

図 4.2 システムライフサイクルプロセス

が定義されています。ISO/IEC/IEEE 15288:2023 による定義を??に示します。

システムのライフサイクルを通じて、システムに係わる活動は、ここに挙げられているどれかのプロセスに相当すると考えることができます。注意点としては、これらのプロセスは順序を定義しているものではないということです。プロセスの順序は、ライフサイクルモデル（lifecycle model）として定義され、異なる概念として理解されます。

これらのライフサイクルプロセスは以下の 4 つに分類されます：

- 合意プロセス（agreement processes）
- 組織のプロジェクトイネーブリングプロセス (organizational project-enabling processes)
- テクニカルマネジメントプロセス（technical management processes）
- テクニカルプロセス群（technical processes）

これらのプロセス群には、システムの直接的な開発活動だけでなく、それを支援する活動も含んでいます。例えば、テクニカルプロセスには、技術者には馴染み深い設計定義プロセス（design process）や実装プロセス（implementation）、運用プロセス（operational process）などが含まれており、イメージしやすいと

思われます。一方で、システムズエンジニアリングで提供されるライフサイクルプロセスはそれらだけではありません。プロジェクトを計画するプロジェクト計画プロセス（project planning process）や、人員の確保や配置などの活動を含む人的リソースマネジメントプロセス（human resource management process）、さらには、契約に係わる活動である取得プロセス（acquisition process）や供給プロセス（supply process）などまで含まれています。これは、これまでのシステムズエンジニアリングの知見において、システムを開発する際には、直接的な開発活動だけでなく、それを支える活動も含めて考える必要があることを表しています。

4.1.5　システムを記述するために必要な観点の提供

システムを開発したり、運用したりするためには、対象システムを記述する必要があります。その記述方法についてもシステムズエンジニアリングは知見を与えてくれます。

一般に、大規模なシステムは、一つの設計図だけではその全貌を現すことはできません。そのため、例えば、詳細な記述の前に、その概要だけを記した文章や設計図なども作成します。従来からそのような記述を「アーキテクチャ設計」や概要設計と呼んでいました。ただし、概要を把握するにしても、目的やそれを読む関係者に応じて複数の記述を作成することが一般的です。それではどのような記述が「アーキテクチャ」とよべるのでしょうか。アーキテクチャの記述は一般にはシステムの抽象的なものです。ただし、抽象的であればアーキテクチャというわけでもありません。なぜ抽象化された記述なのに、それが利用されるかと言えば、システムに関する何らかの関心事を表していたり、懸念事項を確認できたりすることが期待されるからです。また、ここでの「関心事」や「懸念事項」は、それを持つ主体が想定されます。システムに関して関心や懸念を持つ主体を広くステークホルダー（stakeholder, 利害関係者）と呼びます。ここで、なんらかのしステークホルダーのなんらかの関心事（concern）を表現したものをアーキテクチャビュー（architecture view）と呼びます。アー

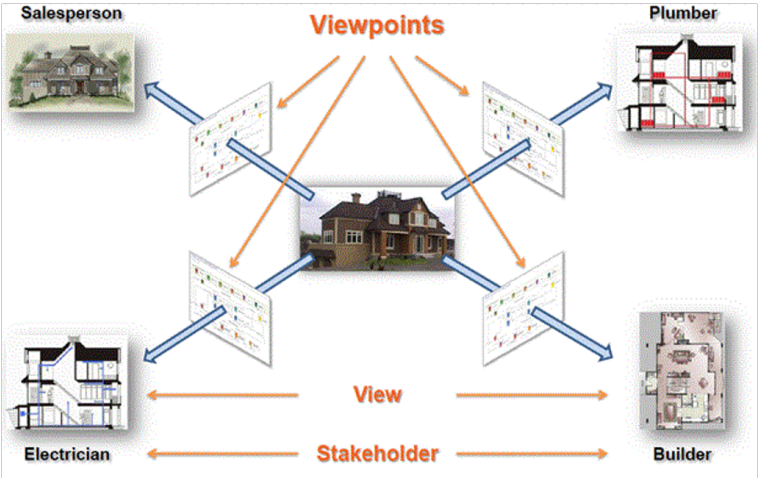

図 4.3　アーキテクチャビュー及びアーキテクチャビューポイントのイメージ

キテクチャ記述（architecture description）は、このビューの集まりであると定義されます。

さらに、ステークホルダーとその関心事は、典型的なものはパターン化しておくと便利であり、かつ、その記述方法は、おのずと適切に表現できる記法がきまってくることが多いです。このような、ステークホルダーとその関心事、その表現方法をまとめたものをアーキテクチャビューポイント（architecture viewpoint）と呼ばれます。アーキテクチャビューとアーキテクチャビューポイントのイメージは次のような図で説明されることがあります （**??**）。

原理的にはシステム毎にステークホルダーやその関心事は異なるため、アーキテクチャビューポイントはシステム開発のプロジェクト毎に構築しなければならないように見えますが、一般的に多くのプロジェクトで使用可能なアーキテクチャビューポイントの集まりもいくつか提供されています。ここでは、その中からアーキテクチャビューポイントの具体例を紹介します。

- オペレーショナルビューポイント
  - ステークホルダー ： ビジネスアーキテクト、経営層

図 4.4　モデルの例

– 関心事 : 対象の論理的なアーキテクチャを明らかにする。

ここで提供されている「オペレーショナルビューポイント」のステークホルダーは、経営層やビジネスアーキテクトと呼ばれる人たちです。関心事の「論理的なアーキテクチャ」とは、実現方法や実装方法に依存しない論理的な構造や振る舞いを記述するアーキテクチャを指します。もし対象システムのステークホルダーに経営層やビジネスアーキテクト相当の人たちが係わっている場合、まずはこのような一般的に提供されているビューポイントから、それらステークホルダーに関係しそうなビューポイントの記述を参考に、必要なビューポイントを定義することができます。

4.2　モデルベースドシステムズエンジニアリング（MBSE）

4.2.1　MBSEの概念

モデルベースドシステムズエンジニアリング（model-based systems engineering, MBSE）は、モデルを活用したシステムズエンジニアリングのアプローチです。ここでいうモデルとは、ある対象に対して、その対象の特徴を理解したり予測したりするために用いられる抽象的な表現を指します。例えば、次の図はモデルの例です。ここでは図の詳しい表記法の説明はしませんが、自動車はシャーシ、エンジン、ブレーキという構成要素からなっていることを表現しています。とても単純ですが、これによって例えば、自動車を開発するにあたって、「シャーシ」「エンジン」「ブレーキ」を開発する必要があることを伝えるためには、これで十分です。一方で、次の図もモデルの例です。このモデルは、モデルは車体と四つのタイヤから構成されていると主張しています。この図は自

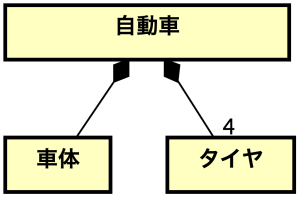

図 4.5 モデルの別の例

動車は、自動車やオートバイと比べて、四輪のタイヤからなることが特徴であることを表現しています。同じ自動車の構成要素を表すモデルでも、二通りのモデルを定義できました。これらはどちらが正しいというものではなく、「その対象の特徴を理解したり予測したりする」ことに依存してそのモデルが適切か否かが議論されるものです。

4.2.2 SysML（Systems Modeling Language）

SysML は、システムズエンジニアリングのための代表的なモデリング言語です。前節の**??**や**??**は SysML の記述例です。実際にはいくつかの図的表現の集合として定義されています。それらは次の四つに大きく分類されます。

1. 構造
2. 振る舞い
3. パラメータ
4. 要求

ここにもシステムズエンジニアリングの知見が反映されています。システムの抽象的な表現には無数の可能性がありますが、システムズエンジニアリングの目的を達成するためには、上記四つの分類で概ね必要十分であることを意味しています。

構造のモデルでは、システムを構成する要素とその関係を表現します。例えば、自動車をシステムとみなした場合、エンジン、シャーシ、車体などの構成要素とそれらの関係を表現します。

振る舞いのモデルでは、システムの動作や状態遷移を表現します。自動車の例でいえば、エンジンの始動から停止までの状態遷移などを表現します。

パラメータモデルでは、システムの性能や特性を決定する要因間の関係を表現します。例えば、自動車の場合、エンジン出力と最高速度の関係、車体重量と燃費の関係などを表現します。

要求のモデルでは、システムが満たすべき条件や制約を表現します。例えば、自動車の場合、消費者からの要望や潜在的なニーズに加え、安全性基準、燃費基準、排出ガス規制などが要求のモデルとして記述されます。

一見、これら四つに含まれていないように見えるシステムの側面について検討してみましょう。

例えば、重要な要因であることには疑いがないコストはどうでしょうか。直接コストのモデルとは書いていませんが、例えば、パラメータのモデルの一つのパラメータになります。また、パラメータの一つであるコストの評価のためには、コンポーネント毎に調達コストが決まっていれば、それらコンポーネントの構成を表す構造のモデルが必要になります。また、どの程度のコストまで許容できるかは要求に係わる要因になります。

練習問題　例えば、あるシステムではユーザビリティ（使いやすさ）が重要な要因であることが分かりました。ユーザビリティに関して、構造、振る舞い、パラメータ、要求の四つの側面から議論せよ。

4.3　トレードオフ分析

4.3.1　トレードオフ分析の概要

トレードオフ分析は、システムの異なる特性や性能間の関係を評価し、最適な解決策を見出すプロセスです。多くの場合、ある特性を向上させると他の特性が低下するという関係があり、これらのバランスを取ることが重要です。

4.3.2 記述型モデルと分析型モデルの連携

トレードオフ分析を効果的に行うためには、記述型モデルと分析型モデルの連携が重要です。

- 記述型モデル：SysML などを用いて、システムの構造、振る舞い、要求などを記述します。ステークホルダー間のコミュニケーションや合意形成に役立ちます。
- 分析型モデル：MATLAB/Simulink などのツールを用いて、システムの性能や動作をシミュレーションします。定量的な評価や予測に役立ちます。

4.3.3 トレードオフ分析の手順

以下の手順でトレードオフ分析を行います：

1. ソリューション案の検討と妥当性確認
2. 分析ケースの検討と妥当性確認
3. パラメータ項目/値の検討と妥当性確認
4. シミュレーションなどによる分析の実施
5. 分析結果の評価と意思決定

4.4 事例: 巨大ショッピングセンターの駐車場

ここでは、巨大ショッピングセンターの駐車場を例にシステムズエンジニアリングの実践を見ていきます。**??**に巨大ショッピングセンターのイメージを示します。ある巨大ショッピングセンターで以下のような課題が問題になっていると想定します。

- 駐車した場所から目的地である売り場まで遠い
- 自分の駐車した場所を忘れる
- 人のクルマにぶつける/ぶつけられる
- 子供を連れていると危ない

図 4.6　巨大ショッピングセンターの駐車場のイメージ

また、本事例においては、単独のシステム開発では無く、複数のシステムの連携についても扱いたいため、以下のような想定を置きます。

- ある程度自動運転車が普及してきている世の中を想定

4.4.1　本事例の流れ

それでははじめに、本事例におけるシステムズエンジニアリング活動の流れを示します。ここでは特に、システムズエンジニアリングにおけるアーキテクチャの記述に絞って例を示します。アーキテクチャのライフサイクルは、国際標準規格である ISO/IEC/IEEE42020[ISO/IEC/IEEE42020] や統一アーキテクチャフレームワークのエンタープライズアーキテクチャガイド [Enterprise Architecture Guide]、TOGAF などに詳しく書いてあります。ここではそれらを参考に簡単なアーキテクチャライフサイクルを定義します。

まずはじめに、アーキテクチャで記述すべき課題を明示化し、特徴付けするプロブレムフレーミングを実施します。ここでは特にモデルは使わずに、この後のアーキテクチャ活動の対象や、アーキテクチャによって明らかにしたいことなどに関する共有の理解を得ることを目指します。次のアーキテクチャ記述計画では、設定した課題に対して、どのようなアーキテクチャを記述するのか

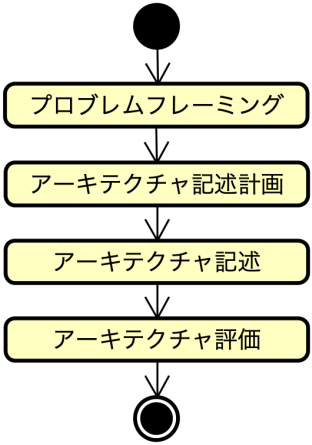

図 4.7 事例におけるアーキテクチャライフサイクル

を決めます。例えば、使用するビューポイントなどを決めていきます。あとは、実際にアーキテクチャを記述し、評価します。

（1） **プロブレムフレーミング** ここでは、アーキテクチャを記述することによって解決したい課題を簡潔に表現したプロブレムステートメント及びアーキテクチャを記述する上で前提とする仮定を定義します。

プロブレムステートメントは例えば次のように書けます。

> プロブレムステートメント: 以下の課題を解決する次期駐車場システムのアーキテクチャを明らかにする。
>
> - 駐車した場所から目的地である売り場まで遠い
> - 自分の駐車した場所を忘れる

仮定の記述例を示します。

> 仮定: ある程度自動運転車が普及してきている世の中を想定。ただし、駐車場内での自動運転には対応していない。

（2） **アーキテクチャ記述計画** アーキテクチャ記述に使用するビューポ

イントの定義などを実施します。ビューポイントの定義のためには、ステークホルダーを識別し、それぞれの関心事を記述し、関心事が表現できるモデルを決定するという手順になります。ここでは、UAF をベースに、以下のビューポイントが得られたと想定します。

- **戦略ビューポイント** 経営層などが関心のある意思決定に必要な情報を表現します。モデルは以下の種類を使用します。
  - 動機のモデル: 対象の背景となっている課題を分析したり、共通認識を得るために使用します。
  - 能力のモデル: 最上位レベルの要求の表現として、次期駐車場システム全体がもつべき能力を表現します。
  - 効果のモデル: 能力が達成した際に得られる効果を表現します。
  - 測定量のモデル: 効果に対する指標を表現するための測定量を定義します。
- **運用ビューポイント** 論理的なアーキテクチャを定義するために使用します。モデルは以下の種類を使用します。
  - 論理的な活動のモデル: 能力を実現するために必要な活動を定義します。
  - トレーサビリティのモデル: 能力と論理的な活動とのトレーサビリティを定義します。
- **リソース及び人員ビューポイント** 具体的なシステムや人員体制などのアーキテクチャを表現します。モデルは以下の種類を使用します。
  - リソース及び人員の語彙のモデル: 能力実現に必要なリソース及び人員を列挙し、自然言語で定義する。

ベースとした UAF のビューポイントはそれぞれ Strategic Viewpoint、Operational Viewpoint、Resource Viewpoint となります。

（**3**） **アーキテクチャ記述**　それでは、具体的にアーキテクチャを記述して行きます。まずは、戦略ビューポイントの動機のモデルを記述例を示します。

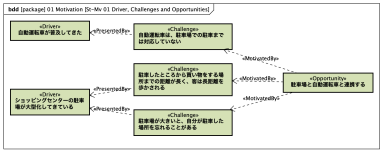

図 4.8 事例：戦略ビューポイントの動機のモデル

- ショッピングセンター近くの巨大な立体駐車場を対象
- 利用者は駐車場入口でクルマから降り、アプリで自動駐車を指示
- 車両は自律的に空きスペースまで移動し駐車
- 引き取り時も、アプリで指示後に車両が自動で出庫

4.4.2 ステークホルダーの識別

4.4.3 能力の定義と評価指標

システムに求められる能力とその評価指標を定義します：

- 能力：自動駐車機能
- 評価指標：
  - 平均入庫時間
  - 平均出庫時間
  - 平均待ち時間
  - 安全度レベル
  - 利用者ストレスポイント

4.4.4 リソースアーキテクチャの検討

システムを構成するリソースとその構造を検討します。主要なコンポーネントには以下が含まれます：

- 自動運転機能付き車両
- 駐車場管理システム
- 空きスペース確認システム（固定カメラまたはドローン）
- ユーザーインターフェース（スマートフォンアプリ）

4.4.5 分析ケースの定義

システムの評価のための分析ケースを定義します。考慮すべき要素には以下が含まれます：

- 自動運転車普及率（例：20
- 来客状況（閑散期、繁忙期）
- 駐車場の構造と容量
- 周辺交通状況

4.4.6 シミュレーションと分析

定義した分析ケースに基づき、MATLAB/Simulink などのツールを用いてシミュレーションを行います。また、STAMP/STPA などの手法を用いて安全性分析を実施します。

4.4.7 結果の評価とトレードオフ分析

シミュレーションと分析の結果を評価し、各ソリューション案のトレードオフを検討します。例えば：

- 固定カメラ方式：初期コストは低いが、スケーラビリティに課題
- ドローン方式：柔軟性が高いが、安全管理に追加コストが必要

これらの結果を総合的に判断し、最適なソリューションを選択します。

4.5 ま　と　め

システムエンジニアリングは、複雑なシステムの開発と管理を体系的に行うための重要なアプローチです。MBSE や SysML の活用、そしてトレードオフ分析の実施により、以下のような利点が得られます：

- システムの全体像と詳細の両方を把握できる
- ステークホルダー間のコミュニケーションが促進される
- 定量的な評価に基づく意思決定が可能になる

- システムの品質、信頼性、安全性の向上につながる

今後のシステム開発者は、これらの手法と考え方を効果的に活用し、より良いシステムを設計・開発することが求められます。

4.5.1 参 考 文 献

- [JIS X 0170] JIS X 0170: 2020 システムライフサイクルプロセス (ISO/IEC/IEEE15288:2015)
- [ISO/IEC/IEEE42020] ISO/IEC/IEEE42020:2019 Architecture Processes
- [UAF] Unified Architecture Framework(UAF) Domain Metamodel, Version 1.2, OMG, December 2021.
- [Enterprise Architecture Guide] Enterprise Architecture Guide for UAF (Informatice), Appendix C, Version 1.2, OMG, July 2022.

5 アシュアランスケースと GSN

5.1 アシュアランスケースとは

アシュアランスケース（Assurance Cases）は、システムまたは製品の特性（安全性、セキュリティ、信頼性など）に関して、構造化された議論を明示的に示すドキュメントです。このドキュメントは、最上位の主張を下位の証拠および前提条件に結びつける形で構成されます。

特に安全性に焦点を当てたアシュアランスケースは、セーフティケース（Safety Case）と呼ばれます。セーフティケースは、機能安全や自動運転開発の分野で広く推奨されており、以下のような規格で言及されています：

- ISO 26262: 自動車の機能安全に関する国際規格です。
- SOTIF（ISO/PAS 21448):
- UL4600:

5.2 セーフティケースの重要性

セーフティケースの重要性は、過去の事例からも明らかです。例えば、2009 年から 2010 年にかけて発生したトヨタ自動車のスロットル制御システムの問題では、安全性に関する明確な説明や証拠の提示が不足していました。この事例は、「それらを明確に用意できていれば、もっと簡単に解決していた（かもし

れない)」という教訓を残しました。

このような経験から、システムの安全性や信頼性に関する説明責任の重要性が高まっています。アシュアランスケースは、この説明責任を果たすための効果的なツールとなります。

5.3 D-Case の概要と基本原則

5.3.1 D-Case と は

D-Case は、JST CREST DEOS プロジェクトの D-Case コアチームによって 2009 年から 2013 年にかけて開発された手法です。名称の「D」は「Dependability (信頼性)」を表しています。

D-Case の主な目的は以下の通りです：

- 合意形成のための手法・ツールの提供
- 開発・運用を通じたアシュアランスケースによるディペンダビリティの合意形成

5.4 D-Case の目的: ミニマムの合意形成

D-Case の核心的な目的は、異なる立場や背景を持つステークホルダー間での「ミニマムの合意形成」を実現することです。以下の図は、この概念を視覚的に表現しています：

[D-Case の合意形成の図を挿入]

この図が示すように、D-Case は以下のプロセスを促進します：

1. 異なる立場・関心・目的、経験・価値観を持つステークホルダーを特定する
2. それぞれのステークホルダーの前提・主張を明確にする
3. 共通の目的・前提を設定し、合意形成を図る

5.4.1 D-Case の基本的な考え方

D-Case を効果的に活用するための基本的な考え方は以下の通りです：

- GSN（Goal Structuring Notation）自体は基本的な構造にとどめ、大きすぎないようにする
- コンテキストには要求分析結果、安全分析結果、テスト結果などの詳細なドキュメントを配置する
- GSN 自体は様々なドキュメントを紐付ける論理的な骨組みとして機能させる
- 他のドキュメントが充実していれば、GSN 自体は迅速に作成できるようにする

この考え方により、D-Case は複雑なシステムの信頼性を効率的に議論し、合意形成を促進するツールとなります。

5.5 GSN（Goal Structuring Notation）

5.5.1 GSN の 概 要

GSN（Goal Structuring Notation）は、アシュアランスケースを視覚的に表現するためのグラフィカル記法です。GSN は以下のような特徴を持ちます：

- 議論のモデル化を可能にする
- 様々な目的で利用できる柔軟性がある
- 元々はシステムの安全性を保証するためのセーフティケースを記述する目的で開発された
- D-Case において中心的な役割を果たす

5.6 GSN の基本要素

GSN は以下の基本的なノードを使用して構成されます：

ゴール（Goal） ステークホルダ間で合意したい主張

戦略（Strategy） 上位のゴールの分解の仕方を説明

前提（Context） 議論の前提となる情報

証拠（Evidence） ゴールが達成できていることを示す証拠（テスト結果など）

未達成（Undeveloped） まだ具体化できていないゴールや説明であることを示す

[GSN の基本要素の図を挿入]

5.6.1 GSN のノード接続ルール

GSN のノードを接続する際は、以下のルールに従います：

- 「前提」に接続する場合はコンテキストリンク（点線）を使用
- それ以外の接続にはサポートリンク（実線）を使用
- 終端は必ず「証拠」か「未達成」
- 「ゴール」は「戦略」に基づきサブゴールに分解する

[GSN のノード接続ルールの図を挿入]

5.6.2 GSNの作成例

以下に、GSN の簡単な作成例を示します。この例では、システムの安全性を主張するための GSN を構築しています。

[GSN の簡単な例の図を挿入]

この例では、以下の要素が含まれています：

- トップゴール：「システムは安全である」
- 戦略：「ハザードごとに議論する」
- サブゴール：「ハザード A に対処できる」「ハザード B に対処できる」
- 前提：「ハザードリスト A,B」
- 証拠：「テスト結果」

このように、GSN を用いることで、システムの安全性に関する議論を構造化し、視覚的に表現することができます。

5.7 D-Case ステップ

5.7.1 D-Case ステップの概要

D-Case ステップは、システム開発や日常の場で異なるステークホルダが手軽に合意形成を行うためのプロセスです。このプロセスは、システム開発における様々なミスコミュニケーションを減らし、ディペンダビリティを向上させることを目的としています。

D-Case ステップは以下の 3 つのステップから構成されます：

1. ステークホルダの設定
2. D-Case の記述
3. 合意形成の実施

5.7.2 ステップ 1：ステークホルダの設定

このステップでは、プロジェクトや議論に関わる全てのステークホルダを特定します。ステークホルダには、開発者、利用者、運用者など、システムに関わる全ての人々が含まれます。

ステークホルダを明確化することで、以下の利点があります：

- 各ステークホルダの立場・関心・考え・経験・価値観を理解できる
- ステークホルダ間の潜在的な対立や誤解を事前に特定できる
- 合意形成のプロセスをスムーズに進められる

5.7.3 ステップ 2：D-Case の記述

D-Case の記述は、以下の 3 段階で行います：

1. 「前提」とトップの「ゴール」を設定する
2. 「戦略」を設定し、トップゴールを分割してサブの「ゴール」を設定する

3. それぞれの最終ゴールのための「証拠」（または「未達成」）を設定する

この過程で、以下の点に注意します：

- これまでの D-Case があれば参照する
- 設定したステークホルダの情報を考慮する
- 論理的な構造を保ちながら、詳細化していく

5.7.4 ステップ 3：合意形成の実施

合意形成の実施には、以下の 2 つの場合があります：

ステークホルダ全員の場合 プロジェクタ等で D-Case を表示しながら、合意ができるか議論する

一部のステークホルダのみの場合 D-Case 記述不参加のステークホルダにもわかるよう合意形成を行う。必要であれば D-Case と同等の情報量を持つ絵や文章を用意する

5.7.5 D-Case の評価基準

作成した D-Case は、以下の 3 つの観点から評価します：

前提の妥当性 前提が過不足なく配置されているか

議論の妥当性 議論が論理的でステークホルダが理解できるか

規模の妥当性 ステークホルダが理解できる規模か

評価の結果、改善が必要な場合は、これらの基準に基づいて D-Case を修正します。一般的に、スライド 1 枚で見えるくらいの規模が目安となります。

5.8 事例紹介：自動運転システム

5.8.1 レベル 4 自動運転システムの事例

ここでは、レベル 4 自動運転システムを継続的に保証するための枠組みを提

案した事例を紹介します。この事例は、SafeComp 2024 で発表された「A Case Study of Continuous Assurance Argument for Level 4 Automatic Driving」に基づいています。

主な特徴：

-
- 塩尻駅から塩尻市庁舎の周回コース (2km) を対象とする
- 特に市役所へ入るための右折にフォーカス
- STAMP/STPA の分析結果などをもとに GSN を記述

この事例研究では、自動運転シャトルバスが直面する様々な信頼性の課題に焦点を当てています。例えば、道路上の物体に対する過度に保守的な安全マージンは、車両が無期限に右折できなくなる可能性があり、交通システム全体の可用性を低下させる可能性があります。

5.8.2　継続的アシュアランスの重要性

この研究では、静的なアシュアランスケースだけでなく、継続的なモニタリングデータを組み合わせた動的なアプローチの重要性を強調しています。UL4600（自動運転車の国際標準）で導入されている安全性能指標（SPIs）は、この考え方を反映したものです。

SPIs は、設計、シミュレーション、テスト、展開の各段階で安全性主張が反証されていないかを検出する手段を提供します。著者らが提案するモニタリングシステムを含むツールチェーンは、この SPI メカニズムの一例と言えます。

5.8.3　アシュアランスケースのトップレベル構造

レベル 4 自動運転システムのアシュアランスケースのトップレベル構造は、以下のような要素で構成されています：

- システムの安全性に関する最上位の主張
- 運用条件、環境条件などの前提
- サブシステムごとの安全性主張

- 妥当性検証と評価に関する主張

[アシュアランスケースのトップレベル構造の図を挿入]

5.8.4 特定のユースケースの妥当性検証

この事例研究では、塩尻市での特定のユースケース（市役所への右折）に焦点を当てた妥当性検証の GSN 図も提示されています。この図は、以下のような要素を含んでいます：

- ユースケースの安全性に関する主張
- 安全性要求事項の充足性
- テストシナリオの網羅性
- 実環境でのテスト結果

[特定のユースケースの妥当性検証の GSN 図を挿入]

この詳細な GSN 図は、特定のユースケースに対する安全性の議論を構造化し、必要な証拠と論理的つながりを明確に示しています。

5.9 ま と め

本書では、システムのディペンダビリティを確保するための重要なツールであるアシュアランスケースと GSN について学びました。主な内容は以下の通りです：

- アシュアランスケース：システムが信頼できることを示すドキュメント
- GSN (Goal Structuring Notation)：グラフィカルなアシュアランスケースの表記法、モデル
- D-Case：ステークホルダ間の合意形成を促進するための手法
- 自動運転システムなどの最近のシステムでの実践例

これらの手法と概念は、今日の複雑化するシステム開発において、信頼性、安全性、セキュリティを確保するために不可欠なものとなっています。特に、自動運転技術や AI システムなど、新しい技術の導入に伴い、システムの振る舞

いの予測が困難になる中で、アシュアランスケースの重要性はますます高まっています。

今後のシステム開発者は、これらの手法を効果的に活用し、システムの信頼性を体系的に示す能力を磨くことが求められます。同時に、継続的なモニタリングと評価を通じて、システムの安全性と信頼性を維持・向上させていく必要があります。

アシュアランスケースと GSN は、単なる文書化ツールではなく、システム開発のプロセス全体を通じて、安全性と信頼性に関する思考を構造化し、ステークホルダ間のコミュニケーションを促進する強力な手段です。これらを適切に活用することで、より安全で信頼性の高いシステムの開発が可能となるでしょう。

5.10　D-Case の実践演習：スマート内覧システム

本節では、D-Case 手法を用いた合意形成の可視化を体験し、D-Case ステップを実践的に学ぶための演習を行います。

5.10.1　演 習 の 概 要

(1) 演習の目的

- D-Case 手法による合意形成の可視化を体験する
- D-Case ステップを実践的に確認する

(2) 演習の流れ

1. テーマ説明
2. ステークホルダの設定
3. D-Case の記述
4. 合意の実施

5.10.2　テーマ：スマート内覧システム

本演習では、「スマート内覧」というサービスを題材とします。これは、ユー

ザーの携帯電話を利用してセルフ（ひとり）で不動産物件の内覧ができるサービスです。

（1） スマート内覧システムの特徴

- 事前予約による無人内覧
- ユーザー認証による鍵のロック解除
- 専用タブレットによる案内
- 制限時間内は自由に見学可能
- 終了 5 分前にタブレットによる通知
- カメラ越しの監視付き（事前連絡あり）
- 終了後、ユーザーが施錠を確認

5.10.3 ステップ 1：ステークホルダ分析

（1） ステークホルダの特定 このシステムに関わる全てのステークホルダを特定します。例えば：

- 開発会社
- 管理会社
- ユーザー（物件内覧者）
- 不動産所有者
- 地域住民

（2） ステークホルダ関係の設定 どのステークホルダ間の関係性について D-Case を描くかを決定します。例：

- 開発会社 から ユーザー
- 管理会社 から 開発会社
- 開発会社 から 管理会社

注意点： どういったステークホルダが存在するかを明確にし、ステークホルダ間の説明責任を分析することが重要です。

5.10.4 ステップ 2：D-Case の記述

（1）**「前提」の抽出**　選択したステークホルダ関係に基づいて、以下の項目を抽出します：

- 要求（2 つ以上）
- 懸念事項（2 つ以上）

例：

要求:
- システムは安全に動作する
- 誤動作が少ない

懸念事項:
- システムに問題はないのか
- セキュリティ、動作の問題

注意点：「誰」から「誰」に対して「どのような」説明責任があるかを意識してください。

（2）**「トップゴール」の設定**　要求や懸念事項に対して「何の」説明責任があるかを意識し、トップゴールを設定します。

例：「スマート内覧システムによって懸念される事項が起こらない」

（3）**「前提」の設定**　トップゴールに対して関連する前提を接続します。これらの前提は、後の展開のガイドとなります。

例：

- セキュリティー面（窓、鍵の施錠）
- ユーザーによる破損行為等
- システムが問題なく動作する
- 管理会社へのメリット

（4）**「戦略」の決定**　トップゴールをどのようなサブゴールに分割するかという方針を「戦略」として記述します。

例：「懸念事項ごとに議論する」

他の戦略の例：

- 安全性の 10 項目で議論する

- ユーザーの懸念事項と期待で分けて議論する
- サブシステムごとに安全性を議論する
- ハザードごとに議論する

（**5**）**「サブゴール」の設定**　戦略に基づいて、トップゴールをサブゴールに細分化します。

例：

- セキュリティー面（窓、鍵の施錠）の対策
- システムが問題なく動作する
- 管理会社へのメリットがある
- 破損行為等への対策がある

（**6**）**「証拠」の設定**　各サブゴールに対して、それを達成したことを証明できる証拠を設定します。

例：「施錠は完璧にされる」というサブゴールに対して、「施錠実験結果」を証拠として設定。

5.10.5 ステップ 3：合意形成の実施

- 説明相手を想定し、作成した D-Case を用いて合意を実施します。
- GSN を理解していない相手もいるため、必要に応じて別の表現方法も用意します。
- わかりやすく、伝わりやすい説明を心がけます。

5.10.6 D-Caseの評価

作成した D-Case を以下の観点から評価します：

1. 前提の妥当性：前提が過不足なく配置されているか
2. 議論の妥当性：議論が論理的でステークホルダが理解できるか
3. 規模の妥当性：ステークホルダが理解できる規模か

5.10.7 演 習 課 題

1. スマート内覧システムについて、開発会社からユーザーへの説明責任を想定した D-Case を作成してください。
2. 作成した D-Case について、グループ内で相互評価を行い、改善点を議論してください。
3. 改善した D-Case を用いて、ユーザー役の人に対して説明を行い、フィードバックを得てください。
4. 最終的な D-Case と、演習を通じて学んだことをレポートにまとめてください。

この演習を通じて、アシュアランスケースの実践的な作成方法と、ステークホルダ間のコミュニケーションにおけるその有用性を理解することができるでしょう。

6 総合演習：自動運転システムの信頼性分析

本章では、これまでに学んだ知識を活用し、STAMP/STPA（System-Theoretic Process Analysis）を用いて自動運転システムの信頼性分析を行います。この演習を通じて、実際のシステム開発における安全性分析の手法を体験的に学びます。

6.1 演習の概要

6.1.1 対象システム

自動運転車の事例：L4 自動車駐車誘導（L4 Car Park Pilot，L4 CPP）

6.1.2 想定シナリオ

自動運転車サービスを提供する会社を想定します。要求分析の結果、以下のような要求仕様が固まってきました：

- なんらかの認証がなされた駐車場又は駐車エリア内において、無人での移動を行う。
- 最大速度は 10km/h。
- 危険を検知したときのアクション：
    1. 車速を徐行速度まで落とす。ただし、交差地点や合流箇所には進入しない。
    2. 安全地帯で停止し、（もしいれば）遠隔オペレーター、又は、ドライバーに通知する。

6.1.3 目指す安全状態

1. 車両は徐行速度で運転され、衝突を回避している。
2. 安全な場所で停止し、セキュアな状態にある。遠隔オペレーター又はドライバーは情報を通知され、今後の対処について決定している。

6.2 システム要素の概要

本演習では、以下のようなシステム要素を考慮します：

6.2.1 Localization（位置推定のためのセンシングシステム）

駐車場内の駐車場所を特定するのに十分な精度を持つ。実装方法は、例えば、駐車場の地図情報と、人工的なランドマークを配置するなどの方法が考えられる。

6.2.2 Environmental Perception Sensor（物標検知のためのセンシングシステム）

前方などの障害物や他車両、歩行者、駐車場構造物、路上の標識やラインなどを検知する。必要に応じて駐車場との通信も可能。

6.2.3 Interpretation and Prediction（解釈と予測システム）

他車両や歩行者、その他障害物などの動きの意味を解釈し、未来の動きを予測する。

6.2.4 Driving Planning（運転計画システム）

走行経路を計画し、計画された走行経路に対し、走行路の条件や自車の幅、他の物標などの制限を考慮し、車両の縦方向及び横方向の運動に変換する。

6.2.5 ADS Mode Manager（自動運転モード管理システム）

自動運転モードの起動条件と解除条件をチェックする。縮退モードへの遷移

も担う。

6.2.6 User State Determination（ユーザー状態決定システム）

ユーザー（任意の搭乗者）が運転機能の委譲を要求しているか検出する。

6.2.7 Monitors（監視システム）

長時間のオペレーションになる場合、電源又は燃料を監視する。

6.3 STAMP/STPA による安全分析の手順

STAMP/STPA を用いた安全分析は、以下の手順で行います：

1. 分析目的の定義
   - ロス、アクシデント、ハザード、安全制約の識別
2. 制御構造図のモデル化
3. 非安全制御動作（UCA: Unsafe Control Action）の識別
4. ロスシナリオの識別

6.4 演 習 課 題

以下の手順に従って、L4 自動車駐車誘導システムの安全分析を行ってください。

6.4.1 ロス（アクシデント）とハザードの識別

システムが引き起こす可能性のあるロス（アクシデント）とハザードを特定してください。

例：

- ロス：人身事故、車両損傷
- ハザード：他の車両や歩行者との衝突、駐車場構造物との接触

6.4.2 制御構造図のモデル化

システムの制御構造図を作成してください。各コンポーネント間の制御アクションと、フィードバック情報を明確に示してください。

6.4.3 非安全制御動作（UCA）の識別

各制御アクションに対して、以下の 4 つの類型に基づいて UCA を識別してください：

1. 必要な制御アクションが与えられない
2. unsafe な制御アクションが与えられる
3. 制御アクションのタイミングまたは順序が適切でない（早すぎる、遅すぎる、正しくない順序）
4. 制御アクションの継続時間が不適切（停止が早すぎる、適用され過ぎる）

UCA 表の例：

表 6.1　UCA 表の例

制御アクション	与えない	与える	タイミング/順序
ブレーキ制御	ブレーキを適用しない場合、衝突の危険がある	不必要なブレーキにより後続車との衝突の危険がある	ブレーキが遅すぎると突の危険がある

6.4.4 ロスシナリオの識別

特定した UCA に対して、それがなぜ発生する可能性があるのかを分析し、ロスシナリオを作成してください。

ロスシナリオの例：

- センサーの故障により障害物を検知できず、ブレーキが適用されない。
- ソフトウェアのバグにより、不適切なタイミングでブレーキが適用される。

6.5 レポート作成

演習の結果を以下の構成でレポートにまとめてください：

1. 分析の目的と対象システムの概要
2. 識別したロス、ハザード、安全制約
3. 制御構造図
4. 非安全制御動作（UCA）の一覧表
5. 主要なロスシナリオの詳細説明
6. 安全性向上のための推奨事項
7. 分析プロセスの振り返りと学んだこと

この総合演習を通じて、STAMP/STPA を用いた実際のシステム安全分析の手法を体験し、自動運転システムの複雑さと安全性確保の重要性について理解を深めることができるでしょう。

6.6 MBSE によるアーキテクチャ設計

本節では、STAMP/STPA で作成したコントロールストラクチャーを利用し、安全なシステムを実現するために必要な機能の抽出を行います。

6.6.1 MBSE モデリングの手順

以下の手順に従って、MBSE モデルを作成します：

（1） ステップ 1: STAMP/STPA ファイルと UAF テンプレートファイルのマージ

1. メニューの「ファイル」から「プロジェクトの比較とマージ」を選択
2. 対象ファイルとして「UAF_empty_template.axmz」を選択
3. マージ画面で「マージ」を選択（優先順位はデフォルトのまま）

（2） ステップ 2: STAMP/STPA モデルの再利用（準備 1）

1. 構造ツリーからコントロールストラクチャーのコンポーネントを全て展開
2. Rs-Tx Resource Taxonomy(empty) を開き、図の名前を Rs-Tx Resource Taxonomy に変更

（3） ステップ 3: STAMP/STPA モデルの再利用（準備 2）

1. STAMP/STPA モデル内のコンポーネントを全て選択
2. 選択したコンポーネントを Rs-Tx Resource Taxonomy の図上へドラッグ＆ドロップ

（4） ステップ 4: STAMP/STPA モデルの再利用（準備 3）　UAF で用意されている概念を用いてモデルを整理します：

- Post, Person ← Personnel View にあります
- System, Resource Artifact ← Resource View にあります
- Environment ← Parameters にあります

（5） ステップ 5: 振る舞いモデルの作成を通じた機能の抽出

1. Rs-Pr のセルから「Create Rs-Pr diagram」を選択
2. STAMP/STPA で分析したシナリオの名前（例：「Rs-Pr 障害物発見時の振る舞い」）を付ける
3. パーティションを準備し、型を設定
4. 色分けを行い、シナリオに必要な機能を定義

6.6.2 MBSE モデリングの補足

- 今回は安全なシステムを実現するために必要な機能の抽出という目的のためにモデルを記述しました
- 機能の抽出には、プロセスのモデルだけでなく、状態のモデルややりとりされる情報のモデルなども記述し、シミュレーションを通じて妥当性を確認する必要があります
- 安全性以外の観点（戦略やオペレーションなど）もモデル化しながら最終的な機能を確定していきます
- MBSE は、多くの観点から検討したソリューションが満足するものであ

ることを客観的に説明するためのエビデンスを提供します

6.7　アシュアランスケースによるシステム保証

本節では、アシュアランスケースを用いてシステムの安全性を保証する演習を行います。

6.7.1　演習 1: ステークホルダーの分析

このシステムにどのようなステークホルダがいるか、挙げてみましょう。

6.7.2　演習 2: 前提とトップゴールの設定

- 開発企業とシステムの安全性認証者（アセッサー）間の合意形成を想定します
- 安全性に関するトップゴールを設定します
- 必要十分な前提をトップゴールにつけます

6.7.3　演習 3: サブゴールへの分割

トップゴールの分割が最も重要です。以下のような分割方法を考えてみましょう：

- システムの構造による分割
- システムのプロセスによる分割
- システムの要求による分割
- システムのハザードによる分割

[自動運転システムの GSN の図を挿入]

6.8　ま　と　め

本講義では以下の内容を学びました：

- 信頼性の概念
  - ディペンダビリティ、安全性、セキュリティ、など
- 安全性、セキュリティ分析手法
  - FTA, FMEA などの従来の安全分析手法
  - STAMP/STPA による安全、セキュリティ分析
- モデルベースシステムズエンジニアリング (MBSE)
- アシュアランスケースによるシステム保証
  - Goal Structuring Notation (GSN)
  - D-Case 手法

6.9　総合演習の課題 2

以下の手順に従って、自動運転システムの安全性分析と保証を行ってください：

1. STAMP/STPA を用いて自動運転システムの安全性分析を行う
2. 分析結果を基に MBSE モデルを作成し、必要な機能を抽出する
3. 抽出した機能と安全性要求を基にアシュアランスケース（GSN）を作成する
4. 作成したアシュアランスケースの妥当性を評価し、必要に応じて改善する
5. 一連のプロセスと結果をレポートにまとめる

この総合演習を通じて、システムの安全性分析から保証までの一連のプロセスを体験し、実際のシステム開発における安全性確保の重要性と方法論について理解を深めることができるでしょう。

7 まとめ

引用・参考文献

1) Yutaka Matsuno, Assurance Carrying Code