目　　次

ま　え　が　き

本書は安全分析、アシュアランスケース、モデルベースシステムズエンジニアリングを解説した本です。

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。システム工学の観点から見ると、システムとは「私たちの身の回りにある、何らかのサービスを提供する物や人を抽象化した概念」と定義できます。

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。

システムが提供する「サービス」とは、システムによって私たちに提供される機能や便益のことを指します。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。

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

さらに、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

これらの手段を適切に組み合わせることで、システムのディペンダビリティを

向上させることができます。

1.3　現代のシステムとディペンダビリティの課題

1.3.1　情報システムの規模拡大とネットワーク化

近年、情報システムは急速に規模を拡大し、ネットワーク化が進んでいます。この変化は以下のような段階を経ています：

- 単機能システム
- 複合機能システム
- ネットワーク化されたシステム
- サービスポータル化されたシステム

この変化に伴い、IT システムは生活・社会インフラとしての役割を担うようになり、組込みシステムもポータル化が進んでいます。これにより、システムの複雑さと重要性が増大し、ディペンダビリティの確保がより困難かつ重要になっています。

1.3.2　大規模なシステム障害のリスク

システムの大規模化と複雑化に伴い、以下のような要因により大規模なシステム障害のリスクが高まっています：

- プログラムサイズの増大と多機能化
- ブラックボックス化したコンポーネントの増加
- 技術進化のスピードの加速
- 接続システムの多様化
- 利害関係者の変化と要求の頻繁な変更

これらの要因により、システムの完全な理解と制御が困難になっています。

1.3.3　オープンシステム（開放系）の問題

現代のシステムは、多くの場合オープンシステム（開放系）として設計され

ています。これにより、以下のような新たな課題が生じています：

- 仕様/実装の不完全さ：要求、仕様、設計、実装、テストの各段階での不完全さが避けられない
- システムの完全理解の困難さ：構成要素の論理的不透明さ（複雑化、巨大化、ブラックボックス化）により、システム全体の挙動予測が困難
- 使用環境の変化に伴う不確実さ：要求事項・レベルの変化、想定外の使われ方、ネットワークを介しての構成要素の変化
- セキュリティリスクの増大：ネットワークを介した外部からの意図的な攻撃のリスク

1.4　これからのディペンダビリティに向けて

1.4.1　不完全・不確実なシステムへの対応

従来の形式手法やテストなどの手法に加え、不完全・不確実なシステムがディペンダブルであるための新たなアプローチが必要とされています。しかし、完全に障害を排除することは不可能であり、深刻な障害が起こる可能性は以前よりも高まっています。

1.4.2　システム保証（System Assurance）の重要性

このような状況下で重要になってくるのが「システム保証」の概念です。システム保証とは、システムがどの程度ディペンダブルか、あるいはディペンダブルでないか、リスク分析などをもとに利用者などのステークホルダーに説明し、納得してもらうプロセスです。

絶対に安全である、あるいは完全にディペンダブルであることは不可能であるという事実を、ステークホルダーに理解してもらうことが重要です。

1.4.3　説明責任の必要性の高まり

近年、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。例えば、2009-2010 年のトヨタ プリウスの北米大規模リコール問題では、原因説明への準備不足が指摘されました。

また、自動車の機能安全規格である ISO 26262 の制定により、自動車メーカーはより説明しやすい形で安全性の根拠を示すことが求められるようになりました。

1.4.4　AI 技術とディペンダビリティ

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。

2 安全分析の基本手法: FTA と FMEA

2.1 未然防止の手法

障害への対応法は、応急処置、再発防止、および未然防止があります。応急処置、再発防止は発生した障害に対応する事後解析であり、リアクティブな方法（Reactive Approach）と呼ばれます。未然防止は、障害が発生してから対策を取るのではなく、計画段階や設計段階の生産の源流において、将来起こりうる障害を洗い出して、それらに対策を講じてしまうことです。潜在的な障害に対応する事前解析であり、プロアクティブな方法 (proactive approach) といわれます。未然防止の手法には、FMEA(Failure Mode and Effect Analysis), FTA(Fault Tree Analysis), ETA(Event Tree Analysis), 良品解析などがあります。信頼性工学や安全性工学では良さ加減を増強するよりも、悪さ加減を減少させる方法がとられることが多いです。

未然防止技術は多くあります（多変量解析、品質機能展開、品質工学、実験計画法、WCA、FEM、信頼性試験、故障解析、信頼性データ解析、リスク・アセスメント、ライフサイクル・アセスメント（LCA））。本講義では、最も代表的な FTA, FMEA を扱います。

2.2 FTAとFMEA

FMEA, FTA は対象とするシステム（製品、設備、プロセスなど）の故障、不具合、欠陥などの「悪さ加減」を論理的に洗い出して、内在する問題点を発見する解析手法です。見出された問題点に対しては、実際の技術活動や管理活動を通じて対策や是正措置が取られます。

FTA と FMEA の形態を示します。図図 2.1、図図 2.2は、工作用の洋ばさみについての FTA と FMEA です。(a) ははさみの部分と名称を示したもので、

図 2.1　FTA の形態（工作用はさみの事例）

内刃、外刃、留ねじの 3 つの部品からなります。(b) ははさみに対して実施した設計 FMEA を示します。部品の故障モードを洗い出して、システム（この場合ははさみ）への影響を表形式で解析する手法です。原因系から結果系を予測する方法と言えます。(c) は、はさみで紙が切れないという結果の事象を取り

図 2.2　FMEA の形態（工作用はさみの事例）

上げて、その原因を探る FTA の一部を示します。FTA は木構造の解析手法です。主な論理ゲートは AND ゲート (論理積) と OR ゲート (論理和) です。

一般にシステムはサブシステムに、サブシステムはコンポーネントに、というように、順次、下位の構成要素に分解されます。最終的にこれ以上分解できないレベルにいたります。システムとしては、階層的に分解できる構造であれば、ハードウエアでも、ソフトウエアでも、プロセスでも、あるいはそれらの複合でもよいです。FMEA は下位の階層の悪さ加減が上位の階層の悪さ加減にどのように影響するかを表形式上で論理的に解析する手法です。FTA は逆に上位の階層の悪さ加減の原因となる下位の階層の悪さ加減を木構造で論理的に解析する手法です。FMEA は単一の原因が及ぼす複数の結果を網羅的に洗い出す

のに優れた手法ですが、複数の原因によって及ぼされる結果の洗い出しは難しいです。一方、FTA は複数の原因によって及ぼされる結果も表現できますが、網羅性が十分ではありません。両者は相補的かつ相乗的に用いられています。

2.3 FTA(Fault Tree Analysis、故障の木解析)

FTA は、「なぜなぜ」を繰り返すことで、重大な故障やトラブルの発生要因を下方に向かって木構造として展開し、網羅した要因の中から重要な要因を抽出します。1979 年にスリーマイル島で発生した原子力事故の解析の際、マサチューセッツ工科大学教授の Rasmussen が原因の特定に使用したことでその有効性が評価され広まった手法です。

2.3.1 FT 図を読む

FT 図は、視覚的に故障に至るメカニズムが大変わかりやすく表現されています。FT 図の記号は、イベントを示す事象記号と、それらの間の因果関係を示す論理記号とに分けられます。表表 2.1にある４つの記号の意味を理解できれば、ほとんどの FT 図を読むことができます。さらに、表表 2.2のノードも

表 2.1 FTA で用いる基本的な記号

用いられます。FT 図の例を図図 2.3に示します。トップ事象である欠報の原因

表 2.2 FTA で用いる便利な記号

には「火災の検知信号が届かない」不具合と「ブザーが鳴らない」不具合とがあり、いずれか一方の原因で欠報に陥ることが示されています。さらに、その２つ以外に原因がないこと、あるいは他の原因は無視してよいことを示している点が重要です。この必要十分性は、熟練者でもしばしば見落とす点です。一

方の AND ゲートは、すべての下位事象が同時に発生するときに上位事象が発生することを示す記号で、並列型に対応します。「火災の検知信号が届かない」事象は、冗長に設置された 2 つのセンサ系 A,B が同時に故障しているときのみ発生する事象となります。FTA には、

図 2.3　FT 図の例

- トップ事象と基本事象との因果関係が視覚的に示され、多様な事象間関係を把握しやすいです。
- AND ゲートにより多重故障を解析できます。

などの特徴があります。

2.3.2　FT 図 の 作 成

FTA の実施は、FT 図の作成と FT 図の解析の 2 つのステップで構成されます。ここでは作成までの手順を説明します。

1. FTA 実施の準備。対象製品を熟知する技術者と品質保証担当者を含めた 3 名から 6 名程度のメンバーで実施します。設計資料、図面、材料部品リストや想定使用状況、関連するトラブル、クレーム情報、特に不具合に関する情報を集めます。
2. 解析対象の機能の理解。FTA で解析する対象製品の構造や機能について、参加者全員で十分理解します。自分の専門とする部分や分野に対象を限定することなく、周辺との関係なども理解することが重要です。
3. トップ事象の選定。信頼性や安全性を損なうような「発生することが望ましくない」トップ事象を注意深く選定します。その際、1. 明確に定義できる事象、2. 多くの下位事象の結果として発生する事象、3. 設計の中で技術的に対処できる基本事象が予想される事象、であることが望ましいです。1 が最も重要な要件であり、「明確」なとは、その事象発生の有無の判断が人によって異なることはない、との意味です。例えば「エアバッグが開かない（不動作故障）」、「エアバッグが不要のときに開く（誤作動故障）」

などは、トップ事象としてふさわしいです。しかし「***の満足度が低い」、「***の回転が不安定」などは、その範囲（基準値）が不明瞭であり適しません。また一見良さそうに見える「排気ガス規制の基準値を満たしていない」のような表現も避けるべきです。ガスの種類や基準値は時代と共に変化するので、「排気ガス CO の規制基準値**ppm を満たさない」などの具体的な基準値を明記する必要があります。2. は、FTA 解析はマンパワーが必要となるので、できるだけ重要なトップ事象を扱う、という意味です。3. は、設計時に、FT 図作成に関わる技術者が基本事象まで書ききることができるようにするためです。

4. トップ事象の 1 次要因への展開。トップ事象の 1 次要因を、製品の構造や機能、手順 1 で準備した情報などを基に列挙し、論理記号を用いて因果関係を明確に図示します。

   展開する方法は大きく分けて 2 つあります。

   (a) 構造（信頼性ブロック図）からの作成。あるシステムが信頼性ブロック図で構造が示される場合、直列系の部分を OR ゲートに、並列型部分を AND ゲートに対応させることで、FT 図を容易に作成できます。

   (b) 機能を考えて作成。実際には、信頼性ブロック図を基に FT 図全てを作成できるケースは多くありません。構成要素の機能に着目して、トップ事象の直接の原因である 1 次要因を抽出し、さらにそれらの原因である 2 次要因を抽出するという具合に、意味を考えてトップダウンに作成することになります。

   1 次要因への展開は、最も頭を悩ませるステップですが、重要な箇所であり、時間をかけるべき手順です。システムを構成するサブシステムごとに空間的に分割し、それぞれを解析するとの方針がとられることが多いですが、それよりも、エネルギーの流れに注目するなど、機能的な側面から 1 次要因を分解すると、装置間の相互作用などを見失うことが少なく、効果的な木になることが多いです。

5. トップ事象の 2 次要因以下への展開。展開可能な 1 次要因に関して、さ

らになぜなぜ分析を続け、2 次要因、３次要因を列挙、基本事象または非展開事象に至るまで論理記号を用いて展開します。

例題 図図 2.4に示されるような、２つのセンサが並列に設置された自動照明器で、「照明が点灯しない」をトップ事象とする FT 図を作成せよ。

図 2.4　照明設備の回路図と信頼性ブロック図

手順 5 までで FT 図ができあがりますが、効果的な FT 図を作成するためのコツがあります。

i)　事象発生の有無が明確な表現にすること。トップ事象にかかわらず、中間事象や基本事象でも、事象発生の有無が人によりかわることのない、明確な表現にします。「**が弱い」、「**が不安定」などの表現はさけるべきです。

ii)　基本事象では、事象を一意に特定できるように表現すること。基本事象レベルでの「接点故障」などの表現は不適切であり、それらの状況により対策が異なります。基本事象では、FT 図の作成者に聞くことなく、その状況を一意に把握できる表現まで分解することが必要です。

iii)　必要で十分な要因を漏れなく列挙しながらトップダウンで作成すること。各レベルでの要因抽出時に、思いつく要因の候補をあげればよい、という発想という発想では適切な FT 図は得られません。特に OR ゲートの下位事象では、必要十分な要因を漏れなく列挙することが求められます。

iv)　横のレベルをそろえながら分解すること。自分の得意な部分になると、いきなり詳細の部位の不具合要因がならぶことがよくあります。1 次、2 次、とトップダウンで作成しますが、それぞれのレベルに並ぶ事象の階層を合わせることが、見やすい FT 図を作るコツです。

2.4 FT 図で定量的に解析する

FT 図を作成すると、トップ事象を発生させる要因、およびその発生経路が明らかになります。それらを基に、解析のステップに移ることができます。

手順 6 では、トップ事象に対して改善効果が高いことが見込まれる基本事象を抽出するために、定量的評価法または定性的評価法を適用します。定量的評価法では、トップ事象の発生確率を求め、さらに、その確率を下げるためにはどの基本事象の発生確率を下げることが効果的かを特定します。そのためには、すべての基本事象の発生確率がわかっていることが前提となります。

手順 7 では、手順 6 で抽出された重要要因について、対策事項、対策方法を決定し、担当部署を決めます。そして、対策実施後のフォローアップを確実に行います。

手順 6 における定量的評価法では、トップ事象の発生確率を求め、さらにその確率を下げるためにはどの基本事象の発生確率を下げることが効果的かを特定します。そのためには、すべての基本事象の発生確率がわかっていることが前提となります。

(i) トップ事象の発生確率。トップ事象の発生確率は、基本事象の発生確率から容易に推定することができます。論理記号に着目して、次の方法で上位事象の確率を算出していきます。

(a) OR ゲート $\rightarrow$ 下位事象の発生確率の和

(b) AND ゲート $\rightarrow$ 下位事象の発生確率の積

例えば、図図 2.5(b) で基本事象である各パソコンの故障の確率を 0.001, プリンタの故障確率を 0.0001 とします。トップ事象の発生確率を推定する場合、パソコン故障の発生確率は、AND ゲートであるから、

$$Pr = 0.001 \times 0.001 = 1.0 \times 10^{-6}$$

となり、トップ事象の発生確率は、OR ゲートで結ばれているため、

$$Pr = 1.0 \times 10^{-6} + 1.0 \times 10^{-4} = 0.000101$$

となります。以下に注意したいです。

- 事象の発生確率とは不信頼度 F のことであり、故障率ではありません。故障率は単位時間当たりの故障発生回数で定義されるので、OR ゲートの場合は各構成要素の故障率の和で定義できますが、AND ゲートでは面倒な計算が必要となります。
- OR ゲートでの計算方法は近似式です。
- AND ゲート、OR ゲートでの計算の独立性が仮定されています。

(ii)　同一事象の排除。FT 図においては、同一の基本事象、中間事象が 2 箇所に現れても構いません。ただし、定量的解析を行う場合、同一事象を一つにまとめた FT 図に変形してから解析を進めなければ、誤った結果を導いてしまうことに注意すべきです。同一事象が複数含まれる場合には、事象間の独立性が失われるからです。図図 2.5では、Tree としては同値ですが、トップ事象の発生確率を求める場合、同一の基本事象を一つにまとめた (b) 図を用いなければなりません。より深刻な例として、2000 年 12 月の京

図 2.5　同じ意味の 2 つの FT 図

福電鉄の正面衝突事故があります。常用ブレーキも非常ブレーキも効かなかった事故ですが、通常、常用と非常用のブレーキは冗長化構造になっており、図図 2.6の (a) の FT 図が想定されます。並列の 2 つのブレーキがあるにもかかわらず同時に故障に陥る確率は、1 次要因が AND ゲートで結ばれているため、積により小さな値になることが容易に想像できます。しかしこの車両のブレーキ装置では、常用と非常用で同一のブレーキレバーの角度の違いで操作する形式であり、ブレーキ制御部は同じものを使用していました。二重化されていたのは、圧縮空気の送風装置だけです。このため (b) の FT 図でトップ事象の発生確率を算出する必要がありました。

図 2.6 常用ブレーキと非常ブレーキの多重故障の解析

(iii) トップ事象に大きな影響を与える基本事象の抽出。トップ事象の発生確率が推定されると、次に、どの基本事象の発生を抑えることがトップ事象の発生確率を下げるために効果的かを知ることができます。OR ゲートの場合、トップ事象の発生確率は（重複事象がなく独立性が成立していれば）基本事象の発生確率のトータル和となるため、発生確率の最も高い基本事象から対策を講じれば良いです。しかし AND ゲートが含まれると容易ではありません。

2.5 FT 図で定性的に解析する

多くの場合、基本事象の発生確率を推定することは難しいです。基本事象の発生確率を前提とせず、トップ事象への影響の大きな基本事象を特定するための方法として、最小カット集合を利用した方法と構造重要度を利用した方法があります。

2.5.1 最小カット集合を利用した方法

最小カット集合とは、トップ事象を発生させ得る最小の基本事象の組み合わせです。例えば図 2.7の Tree では、$\{A, B\}, \{A, C\}$ の 2 つが最小カット集合になります。この時、最小カット集合に共通の基本事象 A が存在することから、A の発生を確実に抑えることができれば、トップ事象を回避することができます。すべての最小カット集合に共通の基本事象が存在するとは限りませんが、一般には、できるだけ少ない基本事象の組み合わせで、すべての最小カット集合の発生を防止できるような組み合わせを探し出し対策を講じることで、トップ事象を防ぐことが可能となります。例えば最小カット集合が $\{A, B\}, \{B, C, D\}, \{B, E, F\}, \{D, E, F, G\}, \{G, H\}$ ならば、B と G の 2 つの事象に確実な対策を施せばトップ事象を回避することができます。最小カット

集合を得るためには、FT 図の AND ゲートを積、OR ゲートを和で表し、ブール代数を用いて積で表した項の和 (加法標準形) で全体を表現できれば、それら各項が最小カット集合です。ブール代数において、0 は事象未発生、正常、1 は事象発生、故障とします。図図 2.7では

$$T = A \cdot (B + C) = A \cdot B + A \cdot C$$

となり、$\{A, B\}$, $\{A, C\}$ の 2 つの最小カット集合が得られます。

2.5.2　構造重要度を利用した方法

構造重要度は、一つの基本事象に着目し、その事象が正規した時にトップ事象が発生する割合 (他の事象の組み合わせ増加数) を表します。図図 2.7の例で説明します。

図 2.7　FT 図とその基本事象とトップ事象

(i)　最小カット集合を求めます。$\{A, B\}, \{A, C\}$ が最小カット集合です。

(ii)　最小カット集合に基づき、基本事象とトップ事象との関係を表す真理表を作成します。$A = B = 1$(故障)、$A = C = 1$ のとき $T = 1$ とすればよいです。

(iii)　A に関する構造重要度 $I_S(A)$ を下記で求めます。

- $X =$ 事象 A が故障時のトップ事象発生の組み合わせ数、
- $Y =$ 事象 A が正常時のトップ事象発生の組み合わせ数としたとき、

$$I_S(A) = \frac{X - Y}{2^{n-1}},$$

すなわち $I_S(A) = \dfrac{3-0}{4} = \dfrac{3}{4}$ となります。また $I_S(B) = I_S(C) = \dfrac{2-1}{4} = \dfrac{1}{4}$ となり、A の構造重要度は B,C のそれよりも 3 倍となります。よって A を重点的に対策を取れば良いといえます。すべての基本事象の構造重要度を算出することで、トップ事象の回避への寄与度を知ることができますが、この値の算出は意外と面倒です。FTA 解析用のソフトウエアでは自動的に計算され便利です。

2.6 FTA 実施上の留意点

FTA を実施する際に注意すべき点は下記の通りです。

(i) 全ての基本事象の発生確率が必要であること。

(ii) 創発故障への対応。部品間の相互作用による創発故障を見落とさないためには、１次の分解で、構造ではなく、機能に着目して分解することに注意するとよいです。

(iii) 動的変化への対応。FTA は基本的に静的な解析であり、動的変化を解析しにくいです。トップ事象の発生確率も動的に変化するため、一定期間後のある時点での値で評価することに限定されます。

(iv) 事後解析での活用。未然防止での利用を念頭に説明してきましたが、故障解析や事故解析などの事後解析では確率値を利用した解析は行いません。実際に発生している真の原因を、多数の可能性がある要因の中から絞るために FT 図を利用するものであり、最小カット集合の中から、真の発生要因を絞り込むことが可能になります。

3 安全分析手法STAMP/STPA

3.1 STAMP（System-Theoretic Accident Model and Processes）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）である。従来の事故モデルが主にコンポーネントの故障（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（Safety Constraint）とその制御に注目する。

3.1.1 STAMP の基本概念

STAMP の基本要素は以下の 3 つである：

- 安全制約：安全が守られるために必要なルール
- プロセスモデル：コントローラが認識するコントロール対象の状態
- コントロールストラクチャ：コンポーネントとそれらの間の相互作用を示したシステムの設計図

STAMP では、アクシデント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る。

3.2 STPA（System-Theoretic Process Analysis）

STPA は、STAMP に基づくハザード分析手法である。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る。

3.2.1 STPA の手順

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

Step 1 ：分析目的の定義

Step 2 ：制御構造図のモデル化

Step 3 ：非安全制御動作の識別

Step 4 ：ロスシナリオの識別

以下の各項で、STPA の各ステップについて解説する。

3.2.2 Step 1: 分析目的の定義

Step 1「分析目的の定義」では、以下を項目を実施する：

- ロス（・アクシデント）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）

各用語の定義は以下のとおりである。

ロス（loss）は、ステークホルダにとって受容できない何かであり、例として、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る。アクシデント（accident）は、望まれない・計画されていないイベントで、ロスへ至るものである。例として、自車両が前方障害物に激突等が挙げられる。以前の STPA の解説ではアクシデントが識別されていたが、STPA Handbook(2018) では、アクシデントを識別するよう記載されていない。しかし、ロスと合わせ

てアクシデントを識別することで、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た。

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ある。なお、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、システム境界外を直接扱わない。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」はハザードではない。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」がハザードとなる。また、ハザードは必ずしもロスへ至るわけではないことにも注意が必要である。例えば、距離センサの値が規定値未満であったとしても、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ない。逆に「自車両が走行中である状態」は、最悪の環境下でロスへ至るが、ロスに至るまでの追加条件が多い（ロスへ至る蓋然性が低い）ため、はざーとしては扱わない。

安全制約（safety constraint）は、システムレベルの条件・動作で、ハザードを防ぐために満たす必要のあるものであり、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る。例えば、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、「要検討：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る。

ロス（・アクシデント）、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。

（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。

(i) 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））

(ii) （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）

(iii) （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（Safety Control Structure Diagram）を構築します。安全制御構造図（Safety Control Structure Diagram）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、下図 (図 3.1) のコントロール・ループにより構成されます。また、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。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、プロセスモデルは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の情報。

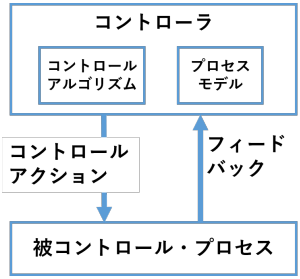

図 3.1　コントロール・ループ

下図（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。

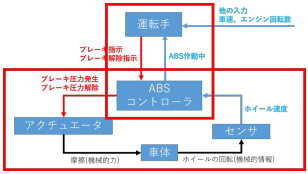

図 3.2　例：二つのコントロール・ループにより構成される安全制御構造図

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：

(i) 与えられないとハザード

(ii) 与えられるとハザード

(iii) 早すぎ、遅すぎ、誤順序でハザード

(iv)　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)			
ブレーキ圧力解放				

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

ここから再開 2024-09-17

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 が発生する主な要因としては、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。さらに、86 ページの説明と図を追加する。

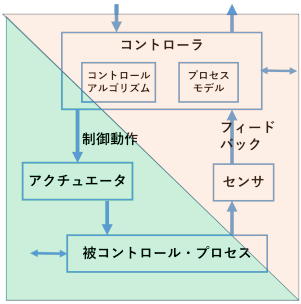

図 3.4 考えるべきロスシナリオ

（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　CAST（Causal Analysis based on System Theory）

CAST は、STAMP に基づく事故分析手法です。実際に発生したアクシデントのシナリオを識別し、システムの安全制御構造がなぜ機能しなかったかを分析します。

3.3.1　CAST の目的

CAST の主な目的は以下の通りです：

- 事故調査の際に問われるべき質問を特定する
- 事故がなぜ起こったのかを明らかにする
- 責任の所在を明らかにするのではなく、システムの改善点を見出す

3.3.2　CAST の手順

CAST は以下の手順で実施されます：

(i)　基本情報の収集

(ii)　安全コントロールストラクチャーのモデル化

(iii) 損失における各コンポーネントの分析

(iv) コントロールストラクチャーの欠陥の識別

(v) 改善プログラムの作成

3.4 事例研究: 自動運転システムへの STPA の適用

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 事例研究: 鉄道システム事故への CAST の適用

ここでは、実際の鉄道システム事故に CAST を適用する例を示します。

3.5.1 事 故 概 要

2019 年 6 月 1 日、横浜シーサイドラインの無人自動運転列車が逆走し、終端の車止めに衝突した事故を分析します。

3.5.2 基本情報の収集

- 事故日時: 2019 年 6 月 1 日 20 時 15 分頃
- 場所: 新杉田駅
- 結果: 乗客 25 名中 17 名が負傷
- システム: 自動列車運転システム（ATO）、自動列車制御システム（ATC）

3.5.3 安全コントロールストラクチャーのモデル化

[鉄道システムの安全コントロールストラクチャー図を挿入]

3.5.4 各コンポーネントの分析

例: ATO システムの分析

- 責任: 列車の自動運転制御
- 不適切な制御行動: 逆方向への出発指示
- 誤ったプロセスモデル: 正しい進行方向の認識の欠如
- コンテキスト: システムの設計上の欠陥

3.5.5 改 善 提 案

- ATO システムの進行方向認識機能の強化

- ATC システムによる逆走検知機能の改善
- 人間のオペレーターによるバックアップ体制の強化

3.6 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。

3.7 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

4 モデルベースシステムズエンジニアリング

4.1 システムエンジニアリングの基礎

4.1.1 システムエンジニアリングとは

システムエンジニアリングは、INCOSE SE HANDBOOK によると、「システムの実現を成功させることができる複数の専門分野にまたがるアプローチおよび手段」と定義されています。この分野は以下の役割を果たします：

- システム開発に必要な概念の提供
- システム開発で必要な活動の提供

システムエンジニアリングは、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。

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。

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：

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

4.1.3 システムに関する概念の定義

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（図 **??**）。

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

システムのライフサイクルに現れる活動は、ここに挙げられているどれかの

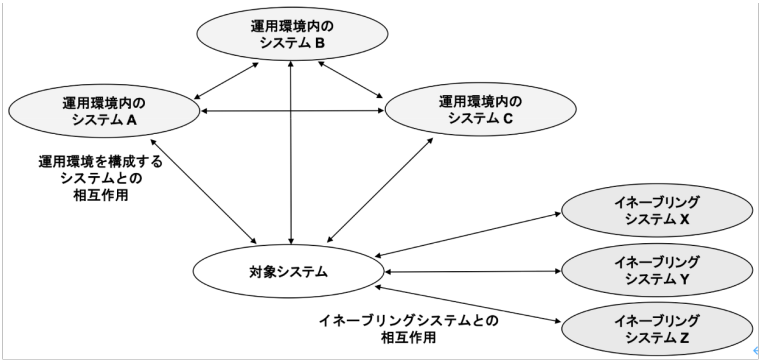

図 4.1　イネーブリングシステム

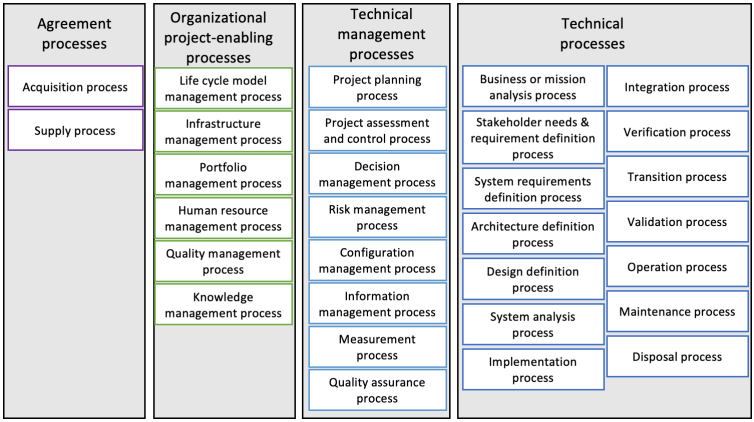

図 4.2　システムライフサイクルプロセス

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 （**??**）。

アーキテクチャビューポイントは、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、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、それらの情報を用いて、アーキテクチャビューポイントの具体例を紹介します。

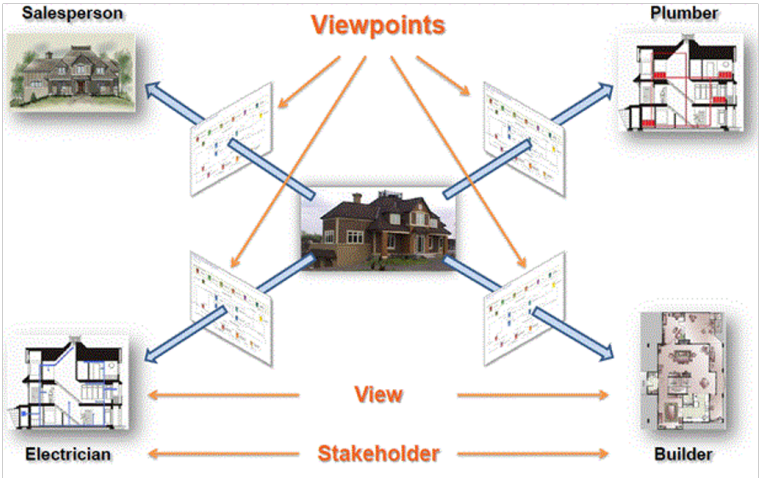

図 4.3　アーキテクチャビュー及びアーキテクチャビューポイントのイメージ

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。

4.2.2　SysML（Systems Modeling Language）

SysML は、システム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。UML（Unified Modeling Language）をベースに定義されており、システムを以下の四つの側面から記述することが可能です：

(i)　構造

(ii)　振る舞い

(iii) パラメータ間の関係

(iv) 要求

4.3 SysML を用いたシステムモデリング

4.3.1 構造のモデリング

構造のモデリングでは、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、自動車システムの場合、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。

4.3.2 振る舞いのモデリング

振る舞いのモデリングでは、システムの動作や状態遷移を表現します。例えば、自動車の運転プロセスや、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。

4.3.3 パラメータ間の関係のモデリング

パラメータ間の関係のモデリングでは、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。

4.3.4 要求のモデリング

要求のモデリングでは、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、安全性基準、燃費基準、排出ガス規制などの要求を表現します。

4.4 トレードオフ分析

4.4.1 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。

4.4.2 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。

4.4.3 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：

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

4.5　事例研究: 自動駐車システム

ここでは、自動駐車システムの設計と評価を例に、システムエンジニアリングの実践を見ていきます。

4.5.1　システム概要

想定する自動駐車システムは以下の特徴を持ちます：

- 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

4.5.2　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

4.5.3　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）

4.5.4 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

4.5.5 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。

4.5.6 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。

4.6 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。

5 アシュアランスケースとGSN

人工知能の研究開発は加速的に進み始めている。2022年に登場したChatGPTは、誰でも簡単にウエブ上で質問をすることができるチャットボットであるが、その回答の詳細さと自然さに、多くの人が驚いた。また歩行者や標識を自動認識する人工知能を持つ自動運転車は、アメリカのWaymo社や日本のTier4社など、多くの企業が開発競争を繰り広げており、アメリカではすでにカリフォルニア州において自動運転タクシーが実用化されている。しかしながら、チャットボットや自動認識を行う人工知能は、100%正しい出力をするわけではない。であるにも関わらず、その圧倒的な利便性から、人工知能が組み込まれたシステムが加速的に普及していくことはより確実になってきている。そのような状況おいて、我々を取り巻くシステムが安心して利用できるものなのか、改めて社会的な合意形成が必要な時期になっている。

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（自動車の機能安全規格）
- SOTIF（ISO/PAS 21448, UL4600）

5.2 MISRA Safety Case Guideline

MISRA（Motor Industry Software Reliability Association）は、ISO 26262 に準拠した Safety Case を作成するためのガイドラインを提供しています。このガイドラインでは、安全性の議論を以下のような層構造で捉えています：

(i) Core（核心）：適切な要求事項を得たか
(ii) Layer 1（第 1 層）：要求事項を満たしたか
(iii) Layer 2（第 2 層）：適切な手段を用いたか
(iv) Layer 3（第 3 層）：適切な環境で開発したか

この構造に基づいて、自動車の安全性に関する議論は以下のように展開されます：

- 自動車は完全で正しい安全ゴールの通りに動作する
- 自動車はハザードイベント i を低減する安全ゴール i の通りに動作する
- 安全ゴール i の妥当性
- 自動車の安全ゴール i への準拠性
- 自動車の安全ゴール i の達成手段
- 自動車の安全ゴール i の開発手段

[MISRA Safety Case Guideline の図を挿入]

5.3 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。

5.4 D-Case の概要と基本原則

5.4.1 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

5.5 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：

(i) 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

(ii) それぞれのステークホルダーの前提・主張を明確にする

(iii) 共通の目的・前提を設定し、合意形成を図る

5.5.1 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。

5.6 GSN（Goal Structuring Notation）

5.6.1 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

5.7 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]

5.7.1 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]

5.7.2 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。

5.8　D-Case ステップ

5.8.1　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：

(i)　ステークホルダの設定
(ii)　D-Case の記述
(iii)　合意形成の実施

5.8.2　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

5.8.3 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：

(i) 「前提」とトップの「ゴール」を設定する

(ii) 「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

(iii) 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

5.8.4 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

5.8.5 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。

5.9　事例紹介：自動運転システム

5.9.1　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。

5.9.2　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。

5.9.3 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]

5.9.4 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。

5.10 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。

5.11　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。

5.11.1　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

(i)　テーマ説明

(ii) ステークホルダの設定

(iii) D-Case の記述

(iv) 合意の実施

5.11.2 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

5.11.3 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。

5.11.4 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。

5.11.5 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。

5.11.6 D-Caseの評価

作成した D-Case を以下の観点から評価します：

(i)　前提の妥当性：前提が過不足なく配置されているか

(ii)　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

(iii)　規模の妥当性：ステークホルダが理解できる規模か

5.11.7 演 習 課 題

(i)　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。

(ii)　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。

(iii)　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。

(iv)　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：
  - (i) 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。
  - (ii) 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

(i) 車両は徐行速度で運転され、衝突を回避している。

(ii) 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：

(i) 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

(ii) 制御構造図のモデル化

(iii) 非安全制御動作（UCA: Unsafe Control Action）の識別

(iv) 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：

(i) 必要な制御アクションが与えられない

(ii) unsafe な制御アクションが与えられる

(iii) 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）

(iv) 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：

(i) 分析の目的と対象システムの概要

(ii) 識別したロス、ハザード、安全制約

(iii) 制御構造図

(iv) 非安全制御動作（UCA）の一覧表

(v) 主要なロスシナリオの詳細説明

(vi) 安全性向上のための推奨事項

(vii) 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

(i) 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

(ii) 対象ファイルとして「UAF_empty_template.axmz」を選択

(iii) 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）

(i) 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

(ii) 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）

(i) STAMP/STPA モデル内のコンポーネントを全て選択

(ii) 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

(i) Rs-Pr のセルから「Create Rs-Pr diagram」を選択

(ii) 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

(iii) パーティションを準備し、型を設定

(iv) 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：

(i)　STAMP/STPA を用いて自動運転システムの安全性分析を行う

(ii)　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

(iii)　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

(iv)　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

(v)　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