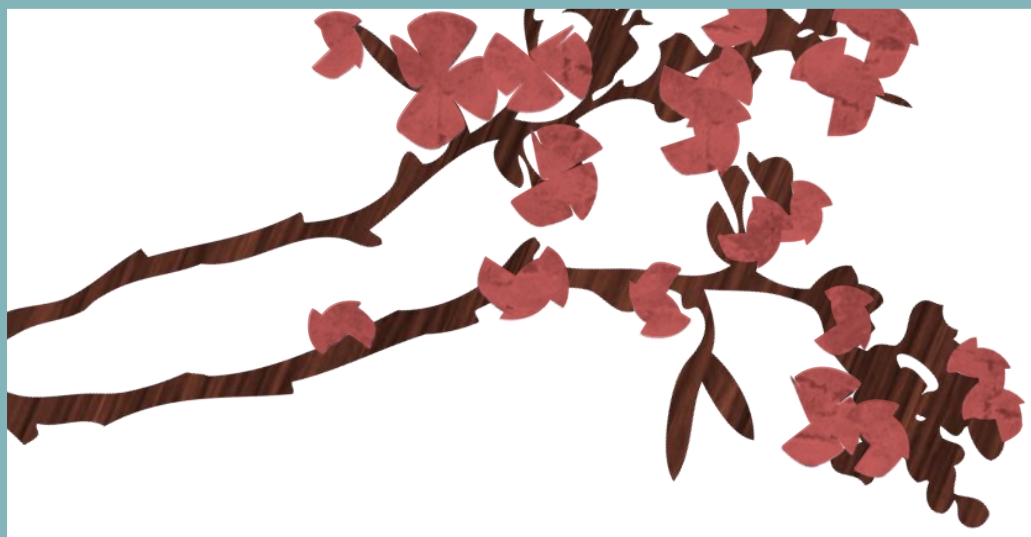


STPA HANDBOOK

日本語版 Ver.0.2

NANCY G. LEVESON
JOHN P. THOMAS

MARCH 2018

このハンドブックは、実際のシステムに対して、STPA の利用を考えている方に向けたものです。他で紹介されているような、理論的基礎を解説するものではありません。実際のプロジェクトでSTPA を使い始める方々にその望ましい使い方を示したり、STPA を教育する講義で補完教材として利用したりするためのハンドブックです。

Translators into Japanese

Seiko Shirasaka / KEIO Univ.

Masa Katahira, Naoki Ishihama, Yasushi Ueda, Naoko Okubo, Ai Noumi, Kazuki Kakimoto / JAXA

COPYRIGHT © 2018 BY NANCY LEVESON AND JOHN THOMAS. ALL RIGHTS RESERVED. THE UNALTERED VERSION OF THIS HANDBOOK AND ITS CONTENTS MAY BE USED FOR NON-PROFIT CLASSES AND OTHER NON-COMMERCIAL PURPOSES BUT MAY NOT BE SOLD.

目次

序文 **3**

第 1 章:はじめに **4**

第 2 章:基本的な STPA 解析の実行方法 **14**

第 3 章:システムエンジニアリングプロセスへの STPA の統合 **56**

第 4 章:STPA を用いた職場の安全 **74**

第 5 章:組織と社会分析 **87**

第 6 章:STPA を用いた先行指標の識別 **102**

第 7 章:安全マネージメントシステム **117**

第 8 章:あなたの組織への STPA の導入 **141**

付録 A:ハザードの例 **145**

付録 B:機能的コントロールストラクチャーの例 **147**

付録 C:UCA 表の例 **155**

付録 D:安全に関するコントロールストラクチャーに係る責任 **160**

付録 E:ソフトウエア集約型システムの分析的分解における限界 **164**

付録 F:非エンジニアのための工学とシステムエンジニアリングの基本的概念 **166**

付録 G:因果関係シナリオ生成を支援するコントロールモデル **175**

*謝辞:*多くの方々にこのハンドブックの初期草案についてコメントを頂きました。以下の皆様のご支援に感謝申し上げます。（アルファベット順)

Matthew Borsch (Ford), Diego Silva Castilho (MIT and Brazilian Air Force), Christopher Deign (Akamai), Mikala Chatzimichailidou (Imperial College), Rashmi Hedge (Ford), Stephen Johnson (Fluor), Ioana Koglbauer (University of Graz), Galvani Lacerda (Embraer), Simon Lucchini (Fluor), Shem Malquist (FedEx), Maj. Dan Montes (U.S. Air Force), Sandro Nuesch (Ford), Felipe Oliveira (Embraer), Todd Pawlicki (U.C. San Diego Medical Center), Kep Peterson (Akamai), Martin Rezjek (Zurich University of Applied Sciences), Lori Smith (Boeing), Michael Stone (Akamai), Maj. Sarah Summers (U.S. Air Force), Mark Vernacchia (General Motors), Sarra Yako (Ford), Col. William Young (U.S. Air Force), and members of the U.K. National Cyber Security Centre.

また、日本語版作成にあたり、以下の皆様のご支援に感謝申し上げます。

慶應義塾大学　白坂 成功

宇宙航空研究開発機構　片平 真史　石濱 直樹　植田 泰士
大久保 梨思子　能美 亜衣　柿本 和希

序文

このハンドブックは、STPA を使い始めたい方や、単純なハザード解析以上の目的で使用したい方の助けとなることを目的としています。そのために、例を提供することを心がけ、更に多くの例を付録に含めました。また、教育を受けていない人がハンドブックを理解し使用するために必要となる、基本的な工学的概念について説明するための付録も提供されています。ハンドブックの本編には、他の新しい概念が紹介されています。

『はじめに』では、STPA の基礎となる事故因果関係モデルである STAMP について簡単に紹介します。また、事故の例を示し、現在の複雑なソフトウエアインテンシブシステムに STPA が必要な理由を説明します。

次の章は、STPA の実行方法に関する詳細なチュートリアルとなっています。初心者だけでなく、STPA を試してその結果を改善したい人にとっても便利です。

ハンドブックの他の章では、STPA を標準的なシステムエンジニアリングプロセスに統合する方法、職場の安全に使用する方法、組織的分析や安全以外のシステムの創発特性に STPA を使用する方法、リスク増加や効果的な安全マネージメントシステム及びサイバーセキュリティの設計の先行指標の提供に STPA を使用する方法について説明します。

最後の章では、STPA を大規模な組織へ統合することで学んできたこと、及び、最も効果的で事業への混乱を最小とする STPA プロセスの構成方法について説明します。

このハンドブックを書く上での私たちの目標は、大学での入門書を書くのではなく、実際に STPA を使用する人のためにガイドを提供することです。したがって、他の研究への参照を省略しています。多くの他の情報源ではこの種の情報を提供していますが、STPA を使用するためのインストラクションや方法に焦点をしぼるものはほとんどありません。多くの例、発表論文、学位論文などを探す場合は、http：//psas.scripts.mit.edu/home/を参照してください。

ハンドブックのユーザーには、ハンドブックを使用する際に、その業界に適切な例で作業することをお勧めします。

第 1 章：はじめに

ナンシー・レブソン

STPA（System-Theoretic Process Analysis）は、拡張した事故の因果関係モデルに基づいた、比較的新しいハザード解析手法です。コンポーネントの故障に加え、STPA では、事故は、コンポーネントが故障していなくても、システムコンポーネント間の非安全な相互作用によっても引き起こされると想定しています。従来のハザード／リスク分析手法に優る STPA の利点には以下のようなものがあります：

- 非常に複雑なシステムを分析することができる。以前は運用でしか発見できなかった「想定外の想定外（Unknown unknowns）」が、開発プロセスの初期に識別され、除去または低減することができる。意図している機能と意図しない機能の両方を取り扱うことができる。
- 従来のハザード解析方法とは異なり、STPA は、安全要求や制約の識別を補助するために、初期の概念分析から開始できる。これらは、システムアーキテクチャやシステム設計の中に、安全性（またはセキュリティ）を設計するために使え、開発後期や運用中に識別される設計の欠陥による費用のかかる手戻りを除去することができる。設計が洗練され、より詳細な設計判断がなされると、STPA 分析もまた洗練され、更なる詳細な設計判断を助ける。要求から全てのシステム成果物に対する完璧なトレーサビリティを簡単に保持でき、システムの保守性と発展を促進できる。
- STPA は、ソフトウエアと人のオペレーションを分析に含み、損失に対する全ての潜在的な要因がハザード解析に含まれることを保証する。
- STPA は、大規模で複雑なシステムでは、多くの場合欠落している、または抽出が困難なシステム機能をドキュメントで提供する。
- STPA は、システムエンジニアリングプロセスとモデルベースシステムエンジニアリングに容易に融合することができる。

STPA に対しては、フォルトツリー解析（FTA）、故障モード・影響及び致命度解析（FMECA）、イベントツリー解析（ETA）や HAZOP（操作性や運用性に着目したハザード解析）等の、従来のハザード解析手法とのたくさんの評価や比較が行われてきました[1]。これらの評価の全てにおいて、STPA は従来の分析によって識別された因果関係シナリオを識別できただけでなく、従来の分析では発見できなかったようなソフトウエアに関連する、故障してない場合のシナリオを多く識別しました。いくつかのケースでは、事故について分析者が知らない状態で、STPA だけが事故の原因を特定できました。加えて、STPA は従来の手法と比較して、はるかに少ない時間と人的リソースで実施できることがわかりました。

基礎理論

STPA を利用するには、基礎的な理論、数学的基礎が少し必要ですが、直感的な理解が役立ちます。このトピックに興味がない場合、次のセクションへ読み飛ばしてください。

STPA はシステム理論に基づいています。システム理論は、高度な技術を利用し、次第に複雑になっていったシステムに対応するために、第二次世界大戦後に開発されました。はじめに、何が置き換わったかを理解することが重要です。

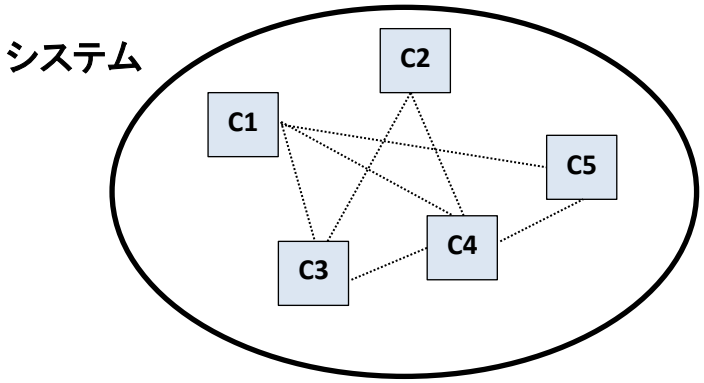
---

[1] これらのうちいくつかの情報は、過去の MIT STAMP/STPA workshops で紹介されています。プレゼンテーションは、http://psas.scripts.mit.edu/home/でご覧になれます。

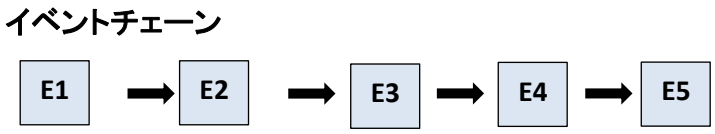
解析的分解（従来のアプローチ）

何世紀にもわたり複雑さは、システムを小さいコンポーネントに分け、各々のコンポーネントを独立に評価及び分析し、次に構成要素の振る舞いを理解するためにそれぞれの結果を結合することで、扱われてきました。

物理的なまたは機能的なコンポーネントは、既知の方法で直接作用すると想定できる個別のコンポーネントに分解されます。例として、航空機の機能的コンポーネントは、推進、ナビゲーション、姿勢制御、ブレーキやキャビン環境制御等々となるかもしれません。また、航空機は同様に物理的コンポーネント、例えば、機体、エンジン、翼、スタビライザーやノズル等に分解できます。機能的コンポーネントと物理的コンポーネント間のマッピングを作成することはできますが、マッピングは直接的で既知であることに注意してください。



対照的に、振る舞いは、個々のイベントが先行するイベント（複数可）の直接的な結果となる場合に、経時的に分割されたイベントとして伝統的にモデル化されます。（AND/OR 論理を利用して）結合された複数の先行または後に続くイベントがあるかもしれません。それは、単に複数のチェーンになります（AND/OR 論理は、ツリーに統合して特定すべき必要な連鎖の数を削減するために使用されます）。



イベントは、例えば、航空機では、プッシュバック、タクシー、離陸、上昇、クルーズ、アプローチや着陸のようなオペレーションの各段階にグループ化できるのではないでしょうか。

いったんシステムがコンポーネント（ピース）に分解されると、コンポーネントは別々に分析され、システムとしての分析をおこなうために、個別の分析結果が組み合わされます。例えば、複雑な物体の重量が分析のゴールだった場合、個別のピースが計量され、システムの重量を得るために結果を組み合わせます。他の共通的な例としては、システムの信頼度は、通常個々のコンポーネントの信頼度を評価し、そしてコンポーネントの信頼度を数学的に組み合わせ、評価されます。

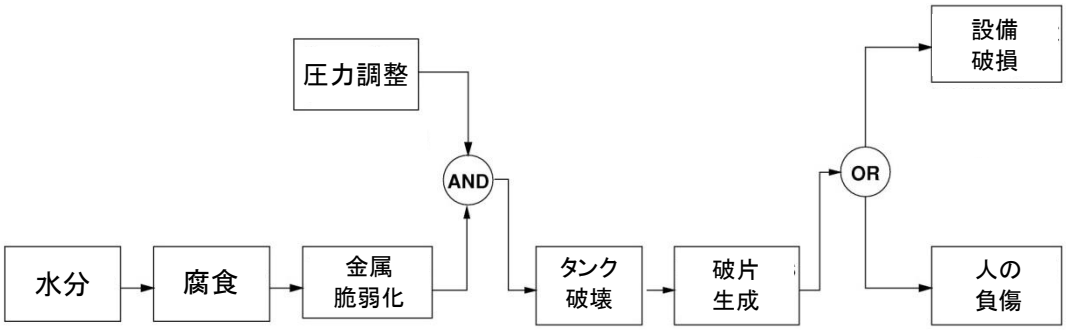
このような分解的(decompositional)または還元主義的(reductionist)アプローチの成功は、分離と個々の分析が、知りたい現象や特性を歪曲していないという仮定に依存します。具体的には、以下のような場合にアプローチは上手くいきます：

- 各コンポーネントやサブシステムが独立で動作する。イベントがモデル化された時、それらは先行イベント、後行イベントの直前直後を除いて、独立である。
- コンポーネントが、全体の中で自らの役割を果たす時、個別に（単独で）評価した時と同じように動作する。
- コンポーネントとイベントが、フィードバックループや他の間接的な相互作用を目的としていない。
- コンポーネントまたはイベントにまたがる相互作用は、対で評価でき、複合的な特徴として結合できる。

これらの仮定が満たされていない場合、個別の分析による単純な合成は正確にシステムを反映しません。

なぜこれら全てが重要なのでしょうか？なぜなら、従来のハザード解析は、分解と更にはこれらの仮定に基づいているからです。関連する基本的なアプローチは、安全性やリスクの指標とすることを想定し、システムをコンポーネントに分割し、事故はコンポーネントの故障によって引き起こされていると想定し、故障の確率をコンポーネントごとに個別に計算し、後でシステムの信頼度計算に分析結果（起こる可能性のあるコンポーネント間の相互作用の種類についての想定に基づき）を結合します。これは安全のリスクを表すものと想定されます。例としては、故障モード影響解析（FMEA）と故障モード、影響、及び致命度解析です。後者は全ての故障を考慮するのではなく、重大な損失となる故障のみを考慮しています。

その代わりに、直接関連する物理的または論理的（機能的）な損失を引き起こす故障イベントのチェーンが識別され、また各々の確率はイベントチェーンが発生する確率に結合されます。以下は、タンクの爆発についてのイベントチェーンの例です。



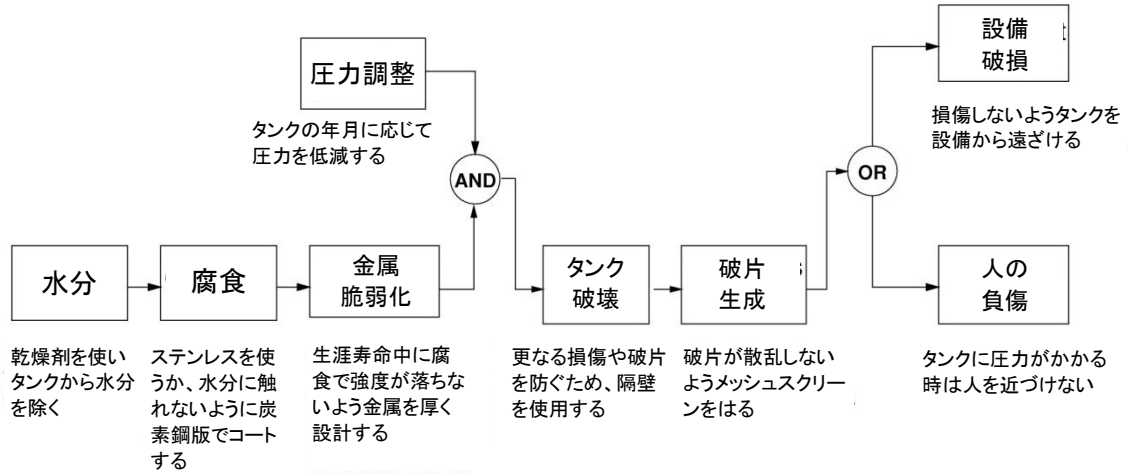
この例では、水分がタンクに入ることで腐食を引き起こします。腐食は金属を脆くし、特定の動作圧力により、タンクの破裂を引き起こします。その結果、破片が生成され、設備の損傷か人の負傷となります。

事故が故障イベントによって引き起こされると想定される場合、事故を防ぐには、１つの故障がチェーンの下流にある他のイベントを引き起こさないように、イベントを除去するかイベントの間にバリアをおくというアプローチは道理にかないます。タンク破裂モデルのイベントは、冗長系、バリア、コンポーネントの高度な完全性(integrity)とマージン(overdesign)設計、フェールセーフデザインを含む典型的な事故防止のための設計テクニックや、人に対しては、運用手順、チェックリストとトレーニングを用いて以下のように注解されます。これらの設計特性は故障イベントの発生か、伝搬の確率を低減するために利用されます。

故障イベントは、もちろん、可能性(probability)や起こりやすさ(likelihood)によって決定される、確率論[2]が仮定であるべきです。残念ながら、ソフトウエアや人間はこの仮定は満たしません。この

[2] 統計的に解析して平均的な行動や予想される行動範囲を理解するための確率やパターンによって記述できるが、正確には予測できないものは、何かが確率論的である。

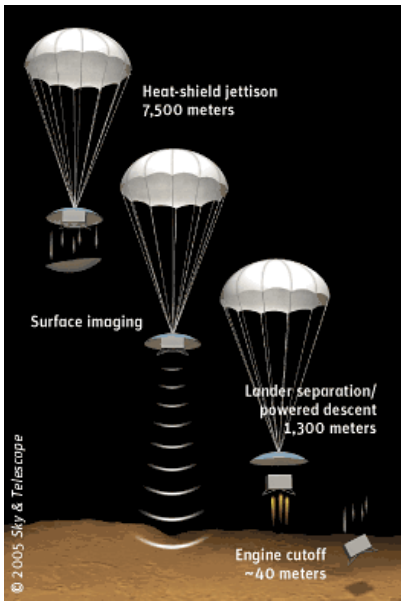
２つ目のイベントチェーンをハザード解析とするアプーチは、フォルトツリー解析（FTA）、イベントツリー解析（EAT）、フォールトハザード解析（FHA）と HAZOP（操作性や運用性に着目したハザード解析、故障を、考慮すべきイベントまたは状態でなく偏差として利用）の基礎となります。



従来の分解的なアプローチであるハザード解析の基礎となる前提は、（従来）構築している電気機械システムのタイプについては真(true)（または十分に近い）であり、新たなハイテクのソフトウエア集約型システムもある特性においても真(true)です。しかし、今日の複雑さが増したシステム（主にソフトウエアの使用により可能になる）は、このアプローチが効果的でないシステムとして構築されています。今日では、システムの予測、理解、計画及び運用前に全ての潜在的なシステムの振る舞いに対して防御をすることは困難です。複雑さはイベントチェーンによりシステムの振る舞いを分割して識別することができない「想定外」を作り上げています。加えて、複雑さは重要なシステム特性（安全性のような）に対し、個別のシステムコンポーネントの振る舞いは関係せず、コンポーネント間の相互作用が影響します。事故は、事実として要求を満たし、かつ故障していないコンポーネント間の非安全な相互作用によって発生します。

いくつか例が役に立つかもしれません。

ある海軍の航空機がミサイルをある地点から別の地点へ運んでいた。あるパイロットが、前の航空機を狙って（彼がそうやるように言われていたように）ダミーミサイルを発射して、計画されたテストを実行した。「賢い」ソフトウエアは　発射を指令されたミサイルが良い位置にない場合、代替のミサイルが発射されるように設計されていたことをどうやら誰も知らなかった。このケースでは、ダミーミサイルと照準の間にアンテナがあった。そして、ソフトウエアは代わりに異なる（良い）位置にある、実弾のミサイルを発射するよう判断した。ここでは航空機のどのコンポーネントが故障したのか？



同様な損失はマースポーララランダーにもあった。安全に着陸するためには、宇宙機は速度を落とす必要がある。これを行うには、火星の大気、パラシュートと降下エンジン（ソフトウエアによる制御）が用いられる。宇宙機が着陸次第、ソフトウエアは、宇宙機への損傷を避けるために、降下エンジンを直ちにシャットダウンする必要がある。着陸のための脚に付いている非常にセンシティブなセンサがこの情報を提供する。しかし、脚が展開されたときに、ノイズ（誤ったセンサ信号）が生成されてしまった。この時の振る舞いはソフトウエア要求にはなかった。おそらく、ソフトウエアはこの時に動作することになっていなかったため（要求には）含まれていなかったが、ソフトウエアエンジニアがプロセッサの負荷を均等にするために早期に動作を始めることに決めた。ソフトウエアは、惑星の地表か 40 メートルのところにいたにも関わらず、宇宙機が着陸したと判断し、降下エンジンを停止させた。ここでは、宇宙機のどのコンポーネントが故障したのか？

航空機がまだ空中にいる最中に、逆噴射（タッチダウンした後に航空機を減速させるために使用される）することは危険である。航空機が地上にいない時に、人間のパイロットが誤って逆噴射させてしまうことを回避するために、防御がソフトウエアに設計されている。詳細は省くが、ソフトウエアにとって飛行機が着陸したことを判断するための手がかりの一部は、車輪に係る重量と、車輪の回転レートであるが、今回のケースでは多くの理由で有効ではなかった。例えば、滑走路が非常に濡れて、車輪がハイドロプレーニングになった。結果として、パイロットは逆噴射を行うことができなくなり、航空機は滑走路を走り切り、小高い丘にオーバーランした。ここでは航空機のどのコンポーネントが故障したのか？

この事故は 2012 年に Tuvelov 航空機がモスクワに着陸する時に起こった。ソフトなタッチダウンにより、通常より少し遅めに滑走路に接触した。その時横風により、車輪の重量スイッチが起動せず、逆噴射が展開しなかった。パイロットは、普段行っているように、逆噴射は行われていると思っていた。それにより、航空機を止めるのに限られた滑走路の長さしかないため、早く止まるように、素早く高いエンジン出力にした。代わりに、これは航空機を前方に加速させ、高速道路の盛土に衝突した。ここでは、航空機のどのコンポーネントが故障したのか？

ワシントン州の海岸には、多くの水と島々がある。カーフェリーは人々の移動のために必要である。ある日、フェリーシステムは、フェリーが港に着いた後にもレンタカーが降りることができずに、立ち往生させられた。地元のレンタカー会社が、盗難を防止するため、エンジンが停止した時に車を動かした場合に、車を使えなくするセキュリティデバイスをインストールしたことによって起こった。フェリーが動いて車が走っていない時に、車は全て自分自身で使えなくし、フェリーシステムはレッカー車がフェリーから車をレッカーするまで動くことができなかった。ここでは、車のどのコンポーネントが故障したのか？

この事件に関係するリチウムイオン電池の製造業者は、バッテリーの破裂は 1 千万飛行時間に一度発生すると断定したが、2013 年にたった 2 週間の間で 2 個の電池が故障した。これには様々な要因が含まれているが、一つの要因はここでの興味深い例となる。電池の故障のイベント中、環境制御システムは、中央バルブを駆動させることにより、冷却ダクトのファンを通して煙を排出するように設計されていた。しかしながら、環境制御システムに電源を供給するユニットが、電池の故障により瞬時にシャットダウンした。その結果、バルブは駆動できず、APU バッテリーによって生じた煙が、客室とバッテリーコンパートメントの外に排出することができなかった。

航空機のアビオニクスシステムでのもっと複雑な例を、付録 A に示しています。加えて、グラーツ大学（オーストリア）の Dr. Ioana Koglabauer は、システム的アプローチの有用さを強調する例を提供していました。彼らの大学にある、認定された fixed-base のフライトシミュレーターは、認証された航空機と同じようなコクピットのコンポーネント（例：ラダーペダル）を有しているが、シミュレータのペダルはしばしば壊れ、交換する必要があります。この問題をコンポーネントレベルで理解し、解決しようとすると、損傷を与えたパイロットを非難し、懲罰を与える、パイロットにこのシミュレータを取り扱うための教育をする、ペダルの改善に投資することもあるでしょう。コンポーネント間（人、ハードウェア、ソフトウエア）の相互作用に注目して、なぜパイロットがペダルをシミュレータと実際の航空機とは異なるようにペダルを使うのかを理解することにより、異なる問題定義と解決策がみつかります。パイロットがシミュレータでブレーキをかける時、シミュレータのソフトウエアが、航空機が止まる時の典型的な小さなピッチダウンの動きを示すようにプログラミングはされていないとわかります。この動きは車を駐車する時にも感じられます。航空機シミュレータにこのフィードバックがない場合、パイロットはブレーキを必要以上に長く、強くかけます。この理由から、シミュレータのペダルは壊れるが、実際の航空機では起こりません。最善の解決策は、ソフトウエアを改善し、パイロットに彼らが必要とするフィードバックを与えることです。

これら全てのケース（そしてその他何百もの）では、問題は個々のコンポーネント故障から由来するものでなく、欠陥のあるシステムデザインに帰結する、システムエンジニアリングプロセスにおける欠陥にあります。分解的な分析では、ヒューマンエラー、ソフトウエア要求の誤りやシステム設計の欠陥などのこれらの事故原因を識別できません。ここで別の方法が必要となります。STPA の根底にある、別の論理的基礎はシステム理論とよばれます。

システム理論

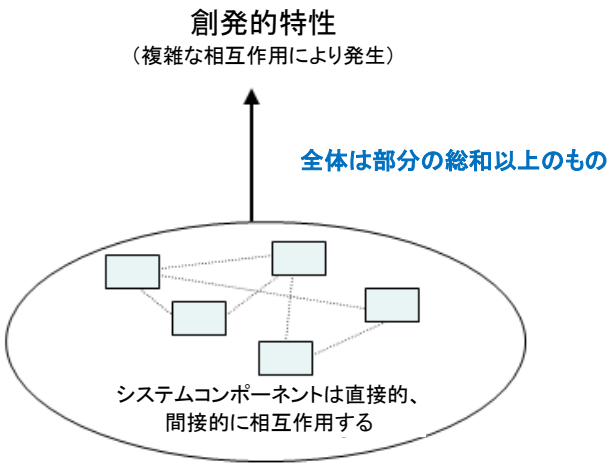
上述の通り、エンジニアリングで用いられるシステム理論は、戦後に作られた複雑さが増したシステムを扱うために、第二次世界大戦の後に構築されました[3]。また、システム理論は、生物学的システムの複雑さをうまく理解するためにも構築されました。これらのシステムでは、相互作用をするコンポーネント（サブシステム）はコンポーネントの振る舞いが明白でない方法で結合されるため、分離や分解による分析は、システム全体としての結果を歪めてしまいます。この新しいアイディアを用いた最初のエンジニアリングは、1950 年代と 1960 年台のミサイルや早期警戒システムでした。

システム理論のユニークな側面は以下の通りです：

- システムは、部品の集合ではなく、全体として捉えられる。「全体は部分の総和以上である」という通説を聞いたことがあると思う。
- 主な関心事は*創発特性*である。それは、個々のコンポーネントの総和ではなく、コンポーネントが相互作用する時の「創発」する特性である。創発特性は全ての技術的、社会的側面を考慮することによってのみ、適切に取り扱うことができる。
- 創発特性は、システムの部品同士の関係によって生じる。すなわち、どのように相互作用し調和するのか、による。

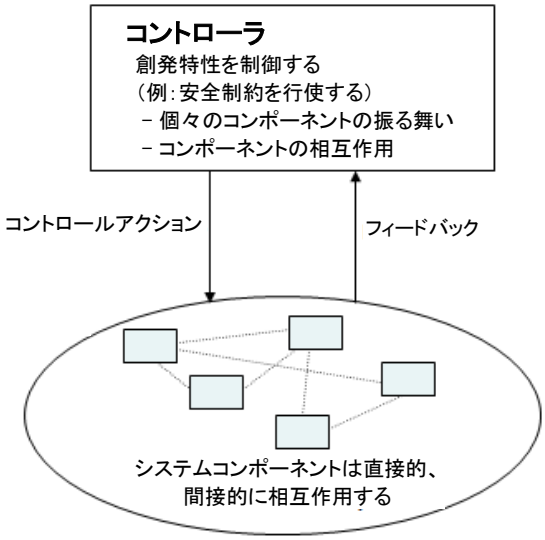
[3] 更に学びたい方には、次の論文をお勧めします。Peter Checkland, *System Thinking, System Practice*, John Wiley & Sons (1981), Gerald Weinberg, *An Introduction to General Systems Thinking*, John Wiley & Sons (1975).

歴史的な文献に興味のある方は、次をお勧めします。W. Ross Ashby, *An Introduction to Cybernetics*, Chapman and Hall (1956).



創発特性が個々のコンポーネントの振る舞いやコンポーネント間の相互作用によって生じる場合、例えば安全性、セキュリティ、保守性や運用性などの創発特性を制御するには、個々のコンポーネントの振る舞いやコンポーネント間の相互作用の制御が必要となることは理解できます。コントローラを図に入れることもできます。コントローラは、コントロールアクションをシステムに提供し、コントロールアクションによる影響を決定するためのフィードバックを得ます。もしこれが、標準的なフィードバックのコントロールループに見える場合、あなたはまさに正しいです。これがそのものです。

コントローラはシステムの振る舞いの制約を強要します。安全制約の例としては、航空機や自動車は最小間隔を取っていなくてはいけない、深井戸の圧力は安全レベルを下回っていなければいけない、航空機は着陸する時以外空中にいるためには十分な浮力を維持しなければならない、武器の偶発的な爆発は避けなければならない、毒性の物質は工場から決して流出してはならない、といったことです。



簡単な例を用いて上記のモデルを説明するために、国内または国際空域を考えてみましょう。もし、各航空会社がいつでもどのルートで飛んでも、各自のスケジュールを最適化するようにできたら、誰もが午後 5 時にシカゴ、ニューヨーク、ヒースローに着陸しようとすると、混乱を生じるでしょう。このような衝突を避けるため、２つの創発特性であるスループットと安全性を制御するシ

ステムレベルの航空管制コントロール（ATC）が導入されています。ATC はシステム上の全てのスループットを最適化する責任を持ち（各機体のルートを最適化するわけでない）、航空機間の十分な間隔を維持します。

このモデルには、私たちが安全工学で行う全てを含んでいます。コントロールは、広義に解釈されます。例えば、コンポーネントの故障と非安全な相互作用は、冗長性、インターロック、バリアまたはフェールセーフデザインのような、設計によってコントロールされます。安全性もまた、開発プロセス、製造プロセスや手順、保守プロセスや通常のシステム運用プロセス等のプロセスを通してコントロールされます。最後に、安全性は、政府の規制、文化、保険、法律や裁判所または個人の関心を含む、社会的なコントロールを用いてコントロールできるかもしれません。人間の振る舞いは、社会や組織のインセンティブ構造の設計を通して、部分的にコントロールできます。

STAMP とは？

STAMP[4] （System-Theoretic Accident Model and Processes）は、システム理論[5]に基づいた新しい事故因果関係モデルの名称であり、前セクションでの説明の通り、STPA の基礎理論になっています。これ（STAMP）は、もっと複雑なプロセスやシステムコンポーネント間の非安全な相互作用を含む直接関連する故障イベントチェーンやコンポーネント故障の域を超えて、従来の因果関係モデルを拡張しています。それが、STPA やその他のツールの根幹となっています。

STAMP では、安全は、故障防止の問題というよりも、動的なコントロールの問題として扱われています。いかなる原因も STAMP から省かれることはありませんが、そこにはもっと多くのものが含まれ、故障を防止することから、システムの動作上の制約を強制することに重点を変化させます。

STAMP を使用するいくつかの利点は以下の通りです：

- ボトムアップでなくトップダウンのため、とても複雑なシステムでもうまくいきます。
- ソフトウエア、人間、組織、安全文化のような、事故やその他の損失の因果関係要因を、異なる方法や別々に扱うことなく含みます。
- STPA や事故解析（CAST）、増加するリスクの先行指標（leading Indicatiors）の識別や管理、組織リスクの分析等のより強力な手法を創造することができます。

STAMP はいかなる創発特性にも適用できるため、STPA は、サイバーセキュリティも含む、いかなるシステム特性にも使用することができます。

STAMP は分析手法ではありません。代わりにどのように事故が発生したのかという一つのモデルであり、一連の想定です。STAMP は、従来の安全解析手法（例えば、フォルトツリー解析、イベントツリー解析、HAZOP、FMECA や HFACS）の根幹である「故障イベントのチェーン」（またはドミノ倒しまたはスイスチーズモデルまたはその他本質的に等価であるもの）の代替です。従来の分析手法は、なぜ事故が「故障イベントのチェーン」モデルの中で発生したかの仮定によって構築されているだけですが、STAMP をベースにすることで、新しい分析手法が構築できます。「故障イベントのチェーン」は STAMP のサブセットであることから、STAMP を元に作られた手法は、従来の安全解析手法を用いて得られる結果が全て含まれることに注目してください。

---

[4] Nancy G. Leveson, *Engineering a Safer World*, MIT Press (2012)

[5] STAMP の根底をなす基本のシステム理論は、のちに基本のシステム理論から自然や社会組織の適応や非線形行動を説明するため進化した複雑性理論と混同するべきではありません。エンジニアリングには、組織のエンジニアリングや設計でさえ、基本のシステム理論で十分です。詳しくは、付録 F をごらんください。

今日、最も広く使用されている２つの STAMP をベースとしたツールは、STPA (System Theoretic Process Analysis) と CAST (Causal Analysis based on Systems Theory) です。STPA は、開発中に潜在的な事故の要因を分析する、事前の解析手法であり、ハザードは除去または制御することができます。CAST は、発生した事故/インシデントを評価し、関与していた原因因子を識別するための、事後的な解析手法です。このハンドブックは、STPA の利用について焦点を当てます。将来、CAST についての同様のハンドブックを計画しています。

本ハンドブックの構成

ハンドブックの第２章は、STPA の基礎をどのように実施するのかを説明します。航空宇宙の例が使用されていますが、他の産業の例は付録に含まれています。

第３章から７章では、STPA を様々タイプの共通的なエンジニアリング活動にどのように使用するのかを説明します。STPA はいかなる創発特性（システム）にも適用できるため、5 章では STPA を安全以外、ここではシステムエンジニアリングの有効性に適用しています。産業界では、STPA の適用先は多くは安全に関連するものでしたが、企業では品質、セキュリティや生産エンジニアリング等、他のシステム特性に使われ始めています。

最後に、第８章では大きな組織でどのように STPA を組み込んでいくのかのアドバイスを提供します。付録では追加の例として、組織の安全管理システムを分析するガイド、なぜ分解（的なアプローチ）が今日のソフトウエア集約型で、複雑なシステムでうまくいかないのか、またエンジニアではない方のために、基本的な概念について追加説明（例を含む）を行なっています。

第 2 章：基本的な STPA 解析の実行方法

ジョン・トーマス

STPA メソッドの概要

基本的な STPA のステップを、グラフィカルに*図 2.1* に示します。

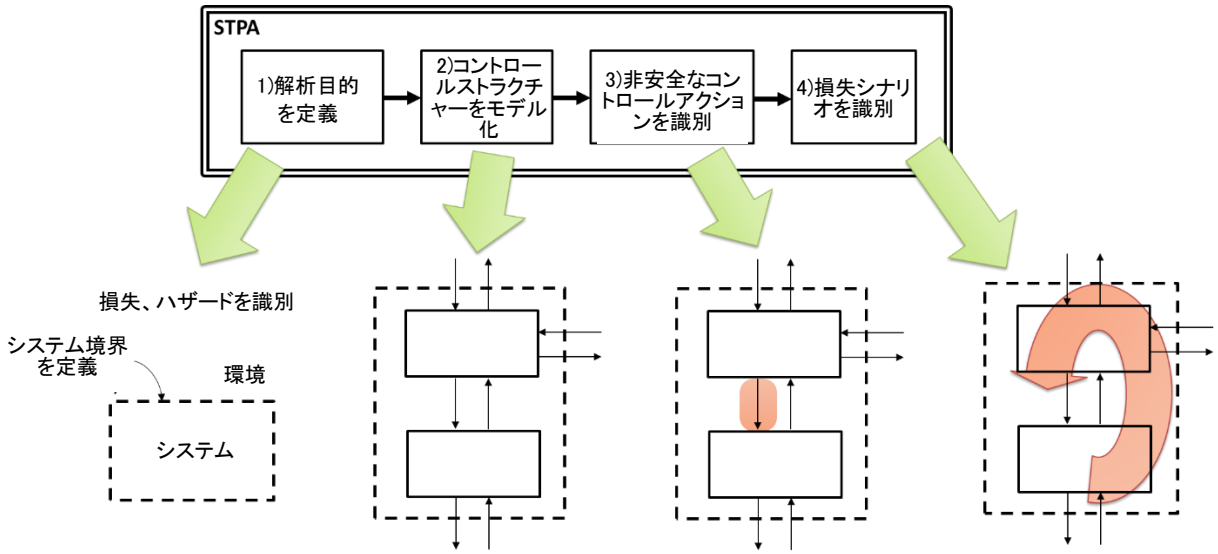


図 2.1：基本的な STPA メソッドの概要

解析の目的を決めることは、いずれの解析方法でも最初のステップとなります。どのような損失をその解析で防ぐことを狙っているのか？ STPA を人命の喪失を防ぐような伝統的な安全目標に対して適用するのか？もしくは、セキュリティ、プライバシー、性能、及びその他のシステム特性に対して、より広く適用するのか？解析対象となるシステムは何か？システムの境界は何か？これら及び他の基本的な質問が、このステップで問われることになります。

第２のステップは、コントロールストラクチャーとよぶシステムのモデルを構築することになります。コントロールストラクチャーは、一連のフィードバックコントロールループとして、システムをモデル化し、機能的関係性及び相互作用を表現しています。コントロールストラクチャーは、通常、非常に抽象的なレベルで始まり、システムの更に詳細な情報を表現するために、繰り返し改良されていきます。このステップは、STPA が安全性、セキュリティ、プライバシー、または他の特性に適用されるか否かで変わるものではありません。

第３のステップは、彼らが最初のステップで定義された損失につながる可能性があるかを調べるために、コントロールストラクチャーのコントールアクションを分析することです。これらの非安全なコントールアクションは、システムの機能要求や制約を作成するために使われます。このステップは、STPA が安全性、セキュリティ、プライバシー、または他のプロパティに適否で変わるものではありません。

第４のステップでは、システムで非安全なコントロールが発生する可能性のある理由を識別します。以下を説明するために、シナリオが作成されます：

1. 間違ったフィードバック、不適当な要求、設計ミス、コンポーネントの故障、及びその他の要因が、どのように非安全なコントールアクションを引き起こし、最終的に損失につながる可能性があるのか。
2. 安全なコントールアクションが提供されるが、後に伝わらない、または適切に実施されない場合に、どのように損失につながる可能性があるのか。

シナリオが識別されると、開発中に STPA を使用する場合には、追加要求を作成し、軽減策を識別し、アーキテクチャ（開発）を推め、推奨設計と新設計の決定を行うために使用することができ、設計が完了した後に STPA を使用する場合には、すでにある設計の決定を再検討/評価し、ギャップを特定し（、テストケースを定義し、テスト計画を作成し、リスクの先行指標指標を作成し、このハンドブックの後の章で説明するように、他の用途に使用されます。

解析目的を定義

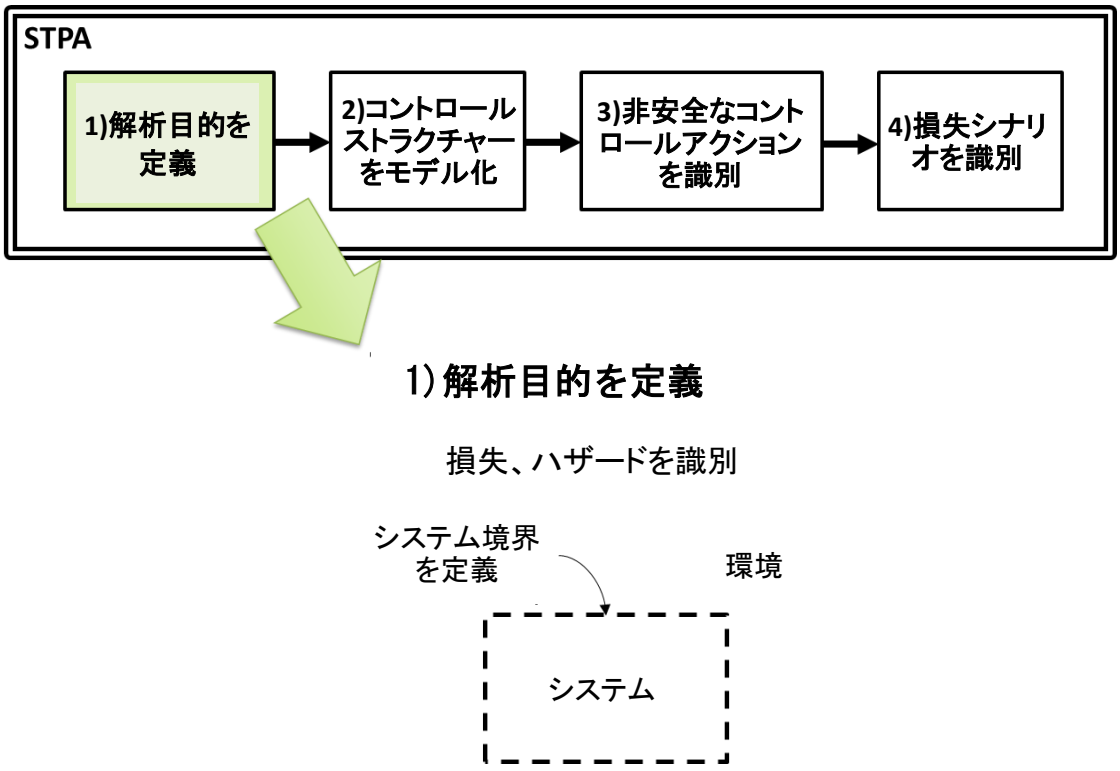


図 2.2：解析目的の定義

図 2.2 に示すように、STPA を適用する最初のステップは、解析の目的を定義することです。解析の目的を定義することは、４つの部分で構成されます：

1. 損失を識別する
2. システムレベルのハザードを識別する
3. システムレベルの安全制約を識別する
4. ハザードの精密化（オプション）

図 2.3 に、本項で説明した 4 つの部分をまとめます。

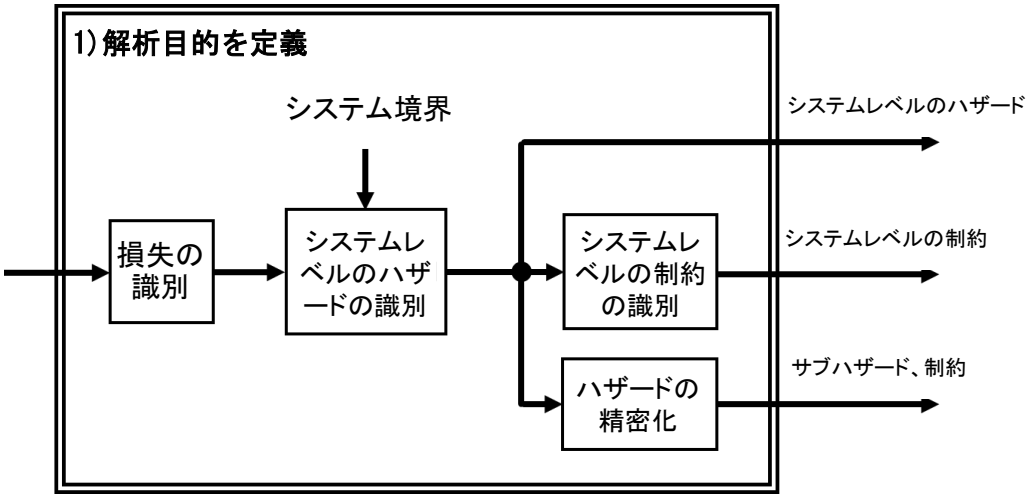


図 2.3：解析目的の定義の概要

損失(loss)の識別

定義：損失は、利害関係者にとって価値のあるものに関わります。損失には、利害関係者に受け入れられないような、人間の生命の喪失や人間の傷害、物的損害、環境汚染、ミッションの喪失、評判の喪失、機密情報の喪失もしくは漏洩、またはその他の損失が含まれます。

異なる産業ではハザード解析の目標を識別するために異なる言葉が使われ、例えば、防止する対象として、（不慮の）事故(accident)、（不運な軽微な）事故・災害(mishap)、または有害事象（adverse event）、といったものです。様々な業界での実務現場の人に使われる複数の用語についての混乱を避けるため、この章では「損失(loss)」という用語を使っており、STPA の目的は損失を防ぐことです。

STPA は、利害関係者に受け入れられないどのような損失も対象として使用できます。1 つ以上の損失が含まれる場合は、ランク付けし、優先順位を付けることができます。全ての STPA 結果は、1 つまたは複数の損失にトレース可能になりますので、解析結果は、それらが参照先の損失に基づいて、簡単にランク付けし、優先順位を付けることができます。

解析を始める前に、利害関係者は、どの損失にフォーカスして解析したいのかを識別しなければなりません。考慮すべき損失は、マネージメントによって、政府の規制によって、または、顧客によって、定義することができます。損失を識別する一般的なアプローチは、以下の通りです。

1. 利害関係者を特定する。例えば、ユーザ、製造者、顧客、操作者など
2. 利害関係者が、システム内の「利害関係」を識別する。彼らは何を重んじるのか？例えば、人間の生活、使用可能な全航空機、発電、輸送など。目標は何か？例えば、使用可能な全航空機を整備する、輸送を提供するために、医療を提供するために、電力を供給するなど。
3. それぞれの価値や目標を喪失に結びつける。例えば、人命の喪失、航空機の喪失、発電の損失、輸送の損失など

多くの解析のユーザが、防止したいと考える、損失の例は、以下の通りです：

L-1：人命の喪失または人々への負傷
L-2：車両の損失または損害
L-3：車両外部のものの損失または損害
L-4：ミッションの喪失（例えば、輸送ミッション、監視ミッション、科学的ミッション、防衛ミッション、など）
L-5：顧客満足度の損失

L-6：機密情報の喪失
L-7：環境の損失
L-8：発電の損失

損失を識別する時の一般的なミスを防ぐためにいくつかのヒント：

1. 損失に、利害関係者に受け入れられないような損失を含めることがある
2. 損失には、「ヒューマンエラー」や「ブレーキの故障」のような個々のコンポーネントまたは特定の原因を引用してはならない
3. 損失には、システム設計者によって、直接、制御されない環境の側面を含めてもよい
4. 明示的に除外されている損失など、特別な注意事項やなされた仮定を全て文書化する

解析において、関心のある損失が識別された後、次のステップでは、これらの損失に関連するハザードを定義します。

システムレベルのハザードの識別

定義：ハザードとは、一組の最悪ケースの環境条件で、損失につながるような、システム状態、または、一組の条件である。

定義：システムとは、いくつかの共通の目標、目的、または、その達成するために、一体として合わせて動くコンポーネントの集合である。システムには、サブシステムを含んでもよく、また、より大きなシステムの一部であってもよい。

システムレベルのハザードを識別するには、最初に解析されるシステム及びシステム境界を識別する必要があります。システムは、アナリストによって考え出される抽象概念です。システムに含まれるもの、及び、システム境界が何であるかを決めなければなりません。解析目的のためにシステム境界を定義するには、システム設計者が対象とするシステムの各部位を入れることが、工学的な点から最も有効な方法となります。これが、ハザードと損失を区別するための最も重要な理由となります。この損失には、システム設計者やオペレータが、部分的にコントロールする、またはまったくコントロールしないような環境の側面が入るかもしれません。安全工学の目標は、解析対象システムにおけるハザードの影響を排除または軽減することです。そのため、いくつかのレベルのコントロールが必要です。*図 2.4* は、その環境からシステムを分離するシステム境界を、抽象概念として示しています。

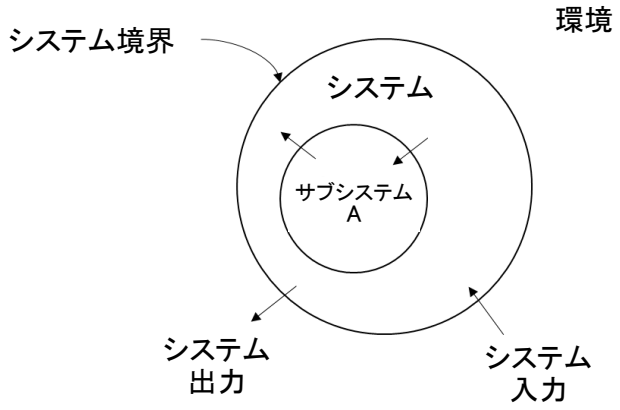


図 2.4：システム、システム境界、及び環境の関係性

例として、原子力発電所を考えてみましょう。放射性物質の放出、近くの住民や都市の近さ、風向は、全て生命の潜在的な喪失につながる重要な要因になるかもしれません。しかし、エンジニアとして、風をコントールすることができない、また、都市の位置をコントロールすることができないかもしれないが、プラントの内外への放射性物質の放出（システムレベルハザード）はコントールすることができます。

システム及びシステム境界が識別されると、次のステップは、システムレベルのハザードを定義することです。これは、最悪の環境条件で損失につながるシステムの状態又は条件を識別することによって行われます。システムレベルのハザードのいくつかの例を以下に示します：

H-1：航空機が、飛行中の最小間隔の基準に違反する[L-1、L-2、L-4、L-5]
H-2：航空機の機体の完全性が失われる[L-1、L-2、L-4、L-5]
H-3：航空機が、地上で、指定された誘導路、滑走路、または駐機場を外れる[L-1、L-2、L-5]
H-4：航空機が、地上で、他の物体に接近しすぎる[L-1、L-2、L-5]
H-5：人工衛星が、科学的データを収集することができない[L-4]
H-6：車両が（軍事的）地域や他の障害物から安全な距離を保てない[L-1、L-2、L-3、L-4]
H-7：無人航空機(unmanned aerial vehicle: .UAV)が監視ミッションを完結できない[L-4]
H-8：原子力発電所が、危険な(hazardous)物質を放出する[L-1、L-4、L-7、L-8]

一般的に、ハザードは一つ以上の損失につながり、それぞれのハザードは結果として生じた損失にトレースできなければなりません。このトレーサビリティは、一般的に、ハザードの説明の後の括弧[]に記載されます。上記の例では前のページの損失例へのトレーサビリティを示しました。

システムレベルのハザードを定義するための 3 つの基本的な基準があります：

- ハザードは、システム状態または条件である（コンポーネントレベルの原因または環境状態ではない）
- ハザードは、いくつかの最悪ケースの環境で損失につながる
- ハザードは、防止されるべき状態または条件を言い表さなければならない

まず、システムレベルのハザードは、システムの状態や条件であり、設計者のコントロール外にある外部環境の状態ではありません。加えて、システムレベルのハザードは、物理的なコンポーネントの障害（例えば油圧リーク、不十分なブレーキ液、等）のような詳細なコンポーネントレベルの原因を表してはいけません。このステップでコンポーネントレベルの原因を参照することは、過度に、分析に制限をかけることになります。後のステップであまり目立たない原因を見落としやすくなるためです。代わりに、予防すべきシステムレベルの状態または条件（ハザード）を識別し、後の STPA ステップにおいて、ハザードのコンポーネントレベルの原因をシステムマティックに識別してください。

次は、ハザードが損失に至る最悪の環境について考えなければなりません。これは、必ずしもハザードが常に損失に至ることを示しているわけではありません。例えば、化学プラントは有毒物質を放出するかもしれませんが、風や気象条件が、近隣の人及び人の住む地域に対して、有毒物質からの影響を防ぎます。しかし、最悪の環境下で、有害物質は、人口密集地域に届き、損失につながる可能性があります。

最後は、ハザードは防止されるべき状態または条件です。「飛行中の航空機」は、最悪の環境下で間違いなく損失につながる可能性があるシステム状態ですが、除去すべきまたは防止すべき条件でありません（さもなければ、航空機を作らないでしょう）。ハザードはそれが防止すべき状態であり、システムを決して陥らせてはならない状態のことです。それは、システムが、普通の方法では目標(goal)を達成できない状態です。

システムレベルのハザードを識別する時の共通間違い

ハザードの原因とハザードの混同

ハザードの定義でよくある間違いは、ハザードの原因とハザードを混同することです。例えば、「ブレーキの故障」、「通知のないブレーキの失敗」、「オペレータの動揺」、「エンジンの故障」、及び「油圧のリーク」は、システムレベルのハザードではなく、ハザードの潜在的な原因です。この間違いを避けるために、識別されたハザードが、システムにある、ブレーキ、エンジン、油圧ライン、などのような個々のコンポーネントに関係していないことを確認してください。その代わりに、ハザードが全体のシステムやシステム状態に関係づけられているべきです。言い換えれば、それぞれのハザードが以下を含んでいることをチェックしてください：

<ハザードの仕様> = <システム>＆<非安全な状態>＆<損失へのリンク>

例　H-1 =航空機　飛行中の最小間隔の基準に違反する[L-1、L-2、L-4、L-5]

正確な順序は重要ではありません。ただ簡単に「飛行機の最小間隔の基準に違反する[L-1、L-2、L-4、L-5]」と書くことができます。重要なのは、システムレベルのハザードが、これらの要素を含んでいることです。

不必要な詳細を含んだ多すぎるハザード

損失と同様に、含まれるシステムレベルのハザードの数には厳しい制限はありません。経験則として、約 7〜10 以上のシステムレベルのハザードがある場合は、より扱いやすいリストを作るために、ハザードのグループ化や結合することを考えてください。不必要な詳細を含めてしまい、扱いにくい、レビューのしにくい、そして欠落していることを見つけにくいリストを作ることになるかもしれません。 その代わりに、（以下のハザードの精密化の項で説明するように）システムレベルのハザードの、より抽象的で扱いやすい組合せから始め、必要に応じて後でサブハザードにそれらを精密化します。

曖昧な、または、繰り返しの表現

システムレベルのハザードは「非安全な」がシステムレベルで何を意味するか正確に定義します。よくある間違いは、ハザード自身で「非安全な」という表現を使用することです。そうすることで、再帰的な定義を作り上げ、分析に情報や価値を追加することになりません。例えば、「H-1：航空機が非安全な飛行を経験する[L-1] 」と書きたくなるかもしれません。それは確かに危険のように思われるので＜定義から非安全な飛行はハザーダスに違いない＞、そう書きたくなりますね？その問題点は、それがあまりにも曖昧で、非安全である実際の条件を識別する助けにはならないということです。いくつかは、「H-1：航空機が非安全な飛行を経験する[L-1] 」の言い方と同じような過ちを犯してしまっています。残念ながら、分析の残りに苦闘し、重要なケースを見落とすだけです。簡単な解決策は、ハザードそのものの中で「非安全な」という言葉の使用を避け、代わりに「非安全な」によって意味されることを正確に特定することです。どのようなシステム状態や条件が、それを非安全にさせるのでしょうか？例えば、コントロールされていない、または他の飛行機と近すぎる飛行機は非安全になるでしょう。お察しの通り、このような実際の条件を特定することが、後述の STPA のステップに極めて役に立ちます。

故障とハザードの混同

他のハザード解析の方法で経験をしている専門家は、時々、特定の技術的な機能からの潜在的なディビエーション(deviation)を説明した、または、物理的なコンポーネントの故障を表した STPA ハザードを記載するという過ちを犯してしまいます。皆さんは、技術的なシステムの一連のディビエーション(deviation)、欠陥(faults)、または機能故障のセットを探すことから始まる伝統的な技法をよく知っているかもしれません。 STPA において幅広い一連の原因を指定するには、定義され仕様化された機能は安全で正しいであろう、ヒューマンオペレータが期待通りに振る舞うであろう、自動化

動作がヒューマンエラーや混乱を誘発しないであろう、オフノミナルなケースは起きないであろう、もしくは、技術的な設計、仕様、要求が正しいであろう、と仮定することはできません。例えば、「（軍事的）地域への航空機のコントールされた飛行」というハザードは STPA に含めることができます。このハザードは、単に、技術的な機能故障を調べることでは省かれてしまうかもしれません。

STPA におけるハザードの識別は、原因にかかわらず、本質的に非安全であるシステム状態と条件についてです。実際には、システムのハザードは、技術的な故障、設計エラー、不備のある要求、またはヒューマンプロシジャー(human procedure)に関わる原因と相互作用(インタラクション）を区別しない、十分に高いレベルで特定されるべきです。

ハザードをレビューする時に、何を探すべきか？

ハザードを識別する際の共通的なミスを防ぐためのヒント

- ハザードは、システムの個々のコンポーネントを参照すべきではありません
- すべてのハザードは、システム全体およびシステムの状態を参照すべきです
- ハザードは、システム設計者およびオペレータによってコントロールできる、またはマネージできる要因を参照する必要があります
- すべてのハザードは、それを防止するためのシステムレベルの条件を記述する必要があります
- ハザードの総数は、通常は 7 から 10 以上にせず、比較的小さくなければなりません
- ハザードは、「非安全な」、「意図しない」、「偶然の」などのような、あいまいな、または再帰的な言葉を含めるべきではありません

STPA が反復的な方法でありハザードはこの時点では、確定されている必要はないことに注意してください。この後の STPA ステップで、新たなハザードを発見することができ、このリストは必要に応じて再検討し、改訂することができます。

システムレベルの制約の定義

定義：システムレベルの制約は、ハザードを防ぐ（そして最終的に損失を防ぐ）ために満たす必要があるシステムの条件や動作を細かく特定する

システムレベルのハザードが識別されると、強制されるべきシステムレベルの制約を特定することは容易です。ただ単に状態を逆にします。

<ハザード>　= <システム> & <非安全な条件> & <損失へのリンク>
<安全制約>　= <システム> & <強制する条件> & <ハザードへのリンク＞

H-1：航空機　最小間隔の基準に違反する[L-1, L-2, L-4, L-5]
SC-1：航空機　他の航空機や物体からの最小間隔の基準を満足しなければならない[H-1]

H-2：航空機　機体の完全性が失われる [L-1, L-2, L-4, L-5]
SC-2：航空機　機体の完全性は、最悪ケースの状態下でも保たれなければならない[H-2]

各制約は、一つ以上のハザードに追跡可能であり、それぞれのハザードは、一つ以上の損失に追跡可能となります。一般的には、トレーサビリティは、1 対 1 である必要はありません。単一の制約が一つ以上のハザードを防止するために使われ、複数の制約が単一のハザードに関連している可能性があり、各ハザードが一つ以上の損失につながる可能性があります。

また、制約は、ハザードが発生した場合に損失をどのようにシステムが最小限に抑えるかを定義できます。例えば、航空機が最小間隔に違反した場合に、その違反が検出されなければならず、航空機が衝突しないようにするために、対策が取られなければなりません。化学プラントが有毒な化学物質を放出した場合に、有毒な環境が検出され、適切な対策が取られなければなりません。これらの制約は、一般的に次のように記述できます：

<安全制約> = <ハザード＞が発生した場合に、<損失を防止または最小限に抑えるためになすべきこと>

SC-3：航空機が最小間隔に違反した場合に、違反が検出され、衝突を防ぐために対策が取られなければならない

安全制約は、特定の解決策や実装を指定すべきではありません。例えば、衝突回避システムを取り付けるような解決策を指定する代わりに、SC-3 では、違反が検出され、衝突を防止する何らかの方法がなければならないということを、単に、示します。特定の解決策を指定することは、初期の段階では、通常は時期尚早で、新たな潜在的に優れた解決策を見落とす結果となる可能性があります。

STPA 解析の残りの部分は、体系的にシステムレベルの危険や損失につながる、これらの制約に違反する可能性がシナリオを識別します。

システムレベルのハザードの精密化（オプショナル）

システムレベルのハザードのリストが特定され、レビューされた後、これらのハザードは、適切な場合には、サブハザードに精密化することができます。サブハザードは多くの STPA の適用(アプリケーション）[6]は必要ではありませんが、大規模な分析労力と複雑な適用のためには役立てることができます。それらは、コントロールストラクチャーをモデル化するなどの将来のステップをガイドできるからです（コントロールストラクチャーに関する次章を参照）。

システムレベルのハザードを精密化する最初のステップは、システムのハザードを防止するようにコントロールされる必要のある基本的なシステムプロセスまたはアクティビティを識別することです。例えば、商業航空に対して前に識別したシステムレベルのハザードを考えてみましょう：

H-4：航空機が、地上で、他の物体に接近しすぎる[L-1、L-2、L-5]

サブハザードを導出する一つの方法は、次のことを聞くことです：このハザードを防ぐためには何をコントロールすることが必要でしょうか？地上で航空機をコントロールするために、航空機の減速、加速、及び操舵（ステアリング）をコントロールする方法が必要になります。これらが適切にコントロールされていない場合（例えば、減速が不十分である場合）は、システムレベルのハザードにつながる可能性があります。

以下のサブハザードが、H-4 に対して導出されます：

H-4：航空機が、地上で他の物体に接近しすぎる[L-1、L-2、L-5]

減速

H-4.1：減速が、着陸時、離陸中止、または地上走行中に不十分である

H-4.2：非対称減速が、他の物体に向かって航空機を動かす

H-4.3：減速が、離陸時の V1 ポイント後に発生する

加速

H-4.4：地上走行中に過度な加速をする

[6] STPA を初めて使う方は、この手順を必要としない小さなアプリケーションから開始すると便利です。

H-4.5：非対称加速によって他の物体に向かって航空機を動かす
H-4.6：離陸中に加速が不十分である
H-4.7：着陸中にまたは駐機時に加速する
H-4.8：離陸中止中に加速され続ける
操舵（ステアリング）
H-4.9：誘導路、滑走路、または駐機場（エプロン）経路の途中で向きを変える不適当な操舵
H-4.10：操舵により、航空機を誘導路、滑走路、または駐機場（エプロン）経路から外れる

これらのサブハザードはそれぞれ、より特定の制約を生成するために使われます。*表 2.1* に、上記のサブハザードに基づいて、減速に対して導出される制約を示します。

表 2.1：サブハザードからの特定のプロセス制約の導出

H-4 から導出されるサブハザード	**制約の例**
H-4.1：減速が、着陸時、離陸中止、または地上走行中に不十分である	SC-6.1：減速は、着陸の TBD 秒以内に、または最低 TBD m/s^2 で離陸中止する際に、行わなければならない
H-4.2：非対称減速によって他の物体に向かって航空機を動かす	SC-6.2：非対称減速は、方向制御の喪失、または、航空機の誘導路、滑走路、駐機場（エプロン）からの離脱、に至ってはならない
H-4.3：減速が、離陸時の V1 ポイント後に発生する	SC-6.3：減速は、離陸時の V1 ポイント後に与えられてはならない

コントロールストラクチャーをモデル化

STPA の次のステップは、*図 2.5* に示すように、階層的なコントロールストラクチャーをモデル化することです。

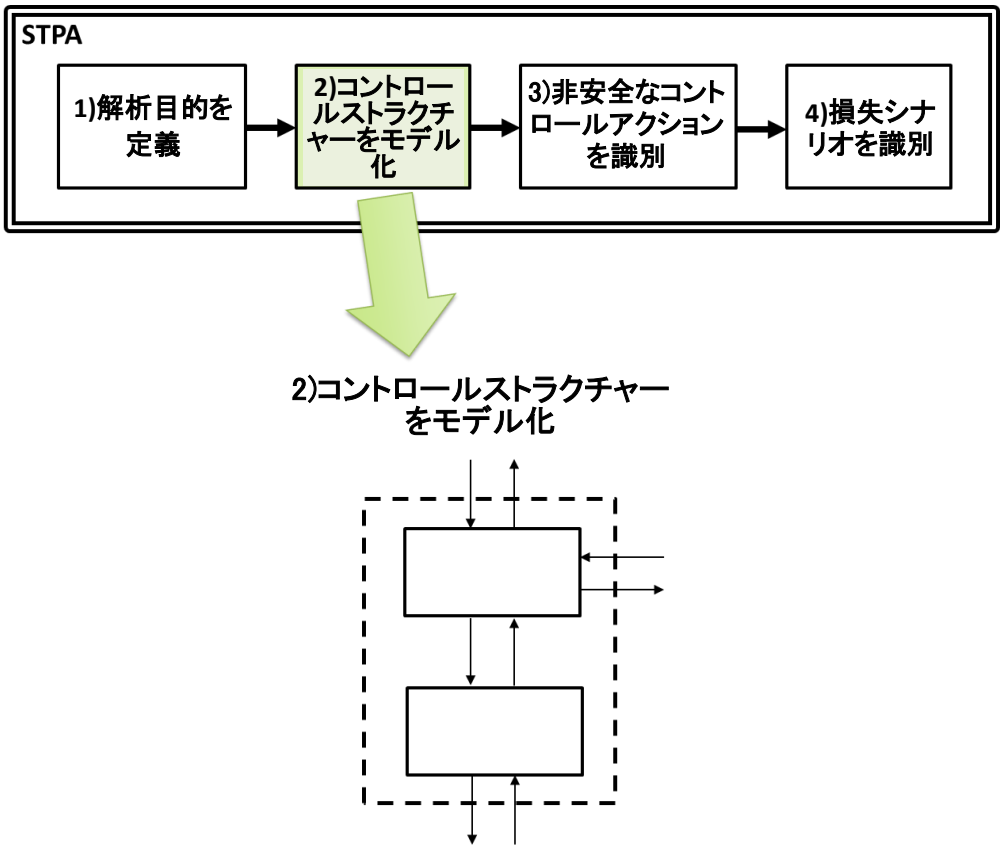


図 2.5：コントロールストラクチャーのモデル化

コントロールストラクチャーとは何か？

定義：階層的なコントロールストラクチャーは、フィードバックコントロールループで構成されたシステムモデルです。有効なコントロールストラクチャーは、システム全体の動作に制約を強制します。

階層的なコントロールストラクチャーは、*図 2.6* に示すようなコントロールループから構成されています。一般に、コントローラは、いくつかのプロセスをコントロールし、コントロール対象のプロセスの動作に対する制約を強制するために、コントロールアクションを与えることができます。コントロールアルゴリズムは、コントローラの意思決定プロセスを表しています。それが、与えるコントールアクションを決定します。コントローラは、また、意思決定を行うために使われるコントローラ内部の信じていること(belief)を表すプロセスモデルを有しています。プロセスモデルは、コントロール対象のプロセスについての信じていることやシステムまたは環境の他の関連のある側面を含むことができます。プロセスモデルは、コントロール対象のプロセスを観測するために使われるフィードバックによって部分的に更新できます。

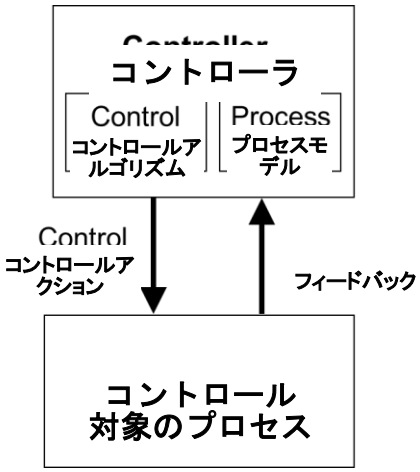


図 2.6：一般的なコントロールループ

問題は、*図 2.6* のどの時点でも発生する可能性があります。例えば、プロセスモデルは、現実と一致しない場合があります（例えば、コントローラが、飛行機が本当は上昇している時に降下していると考えたり、自動車が本当は後退(reverse)に入っている時に駐車(park)に入っていると考えたり、空港の滑走路が空いている、など）。これは、非安全なコントロールアクションにつながります。センサの故障が、誤ったフィードバックを引き起こし、非安全な動作につながる可能性があります。ある設計では、必要なフィードバックが欠落しているかもしれないし、プロセスモデルの欠陥と非安全な動作をもたらすような遅延フィードバックを与えているかもしれません。お察しの通り、STPA は、これら、及び損失につながる他のシナリオを体系的に識別する方法を提供しています。

図 2.6 の一般的なコントロールループは、損失につながるような、複雑なソフトウエアと人間の相互作用を説明し、予測するために使用できます。それが最新エンジニアリングの最大の課題の２つです。人間に対して、通常、プロセスモデルはメンタルモデルとよばれ、コントロールアルゴリズムは、オペレーション手順(procedure)や意思決定ルールとよばれることもあるが、基本的な考え方は同じです。

もちろん、一般的に、ほとんどのシステムには、コントロールループ間のいくつかの重複や相互作用があります。*図 2.7* に示すように、コントロールループの複数の相互作用が、階層的なコントロールストラクチャーでモデル化できます。航空のための簡単な例を*図 2.8* に示します。

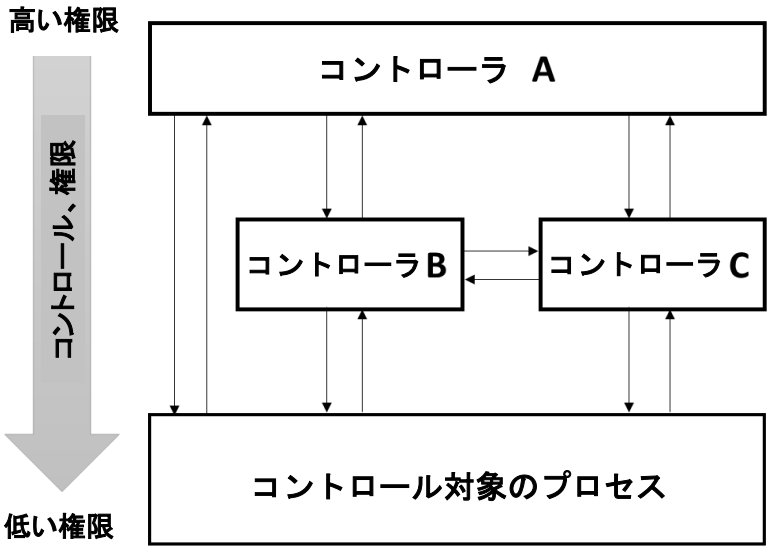


図 2.7：一般的な階層的なコントロールストラクチャー

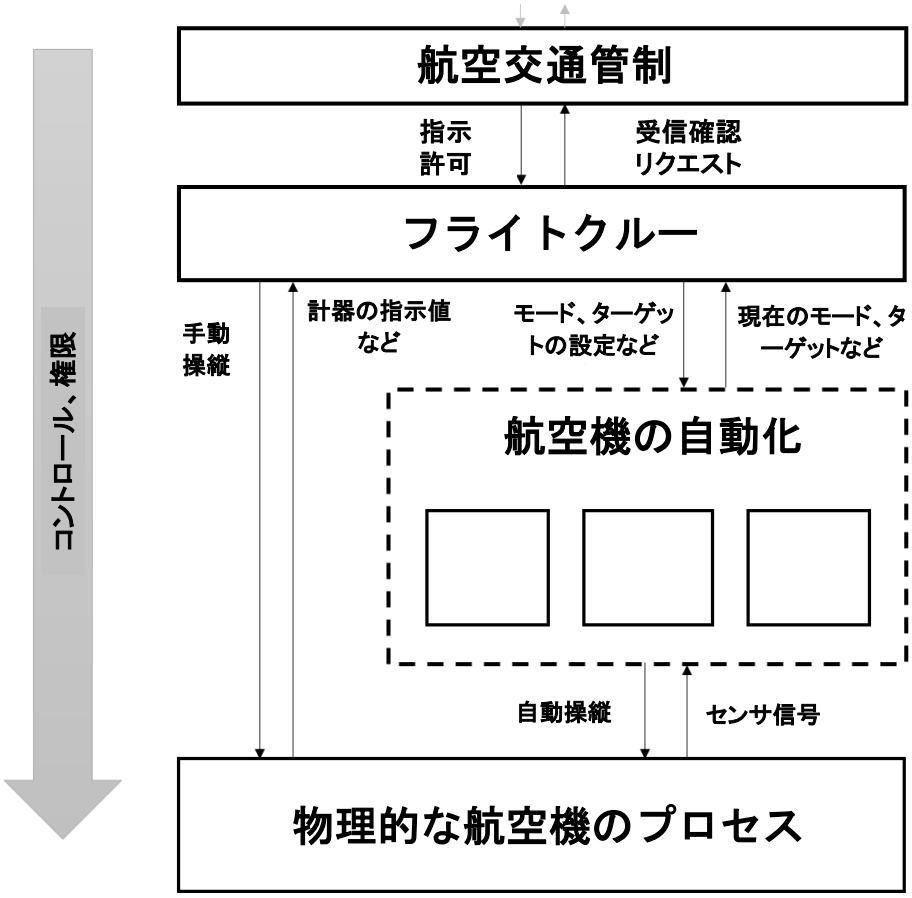


図 2.8：航空のための階層的なコントロールストラクチャーの単純な例

一般的に、階層的なコントロールストラクチャーは、少なくとも 5 つのタイプの要素[7]が含まれています：

- コントローラ
- コントロールアクション
- フィードバック
- コンポーネントからの他の入力及び出力（制御やフィードバックではない）
- コントロール対象のプロセス

階層的なコントロールストラクチャーの縦軸には意味があります。それは、システム内のコントロールと権限を示しています。垂直の配置は、上部にある高いレベルのコントローラから底部にある最も低いレベルのエンティティへのコントロールの階層を表します。各エンティティは、その直下のエンティティ対するコントロールと権限を有し、かつ各エンティティは、同様にすぐ上位のエンティティからのコントロールと権限の対象となります。例えば、図 2.8 の航空機の自動化は、物理的な航空機システムにコントロールアクションを送信し、フィードバックを観測することによって、コントローラとして動作できます。同時に、航空機の自動化は、フライトクルーからのコントロールアクションを受信して、実行し、クルーにフィードバックを送るといった、コントロール対象のプロセスでもあります。

言い換えると、上向きの矢印はフィードバックを表しているのに対し、全ての下向きの矢印はコントロールアクション(コマンド）を表します。これらの約束ごとは、複雑さを管理し、認識しやすいコントロールの関係性とフィードバックループを作成する助けとなります[8]。 いくつかのケースでは、単にコントロールストラクチャー・ダイアグラムを描くことで、以前に発見されていない欠陥を、いやというほど明らかにできます。例えば、安全なコントロールアクションを選ぶのに必要なフィードバックを持たないエンティティによってコントロールアクションが与えられるかもしれない、フィードバックがそれについて何かをする能力を持たないエンティティに与えられるかもしれない、競合を検知したり、解決する能力を持たない同じコンポーネントへ、複数のコントローラが競合するコマンドを与えられるかもしれません。この段階で、もし欠陥を把握していなくても、心配しないでください。STPA 方法論の残りの部分で、体系的にこれら及びその他の問題を識別するでしょう。

競合の共通点

コントロールストラクチャーは物理モデルではない

STPA に使用される階層コントロールストラクチャーは、物理ブロック図、概略図、又は配管及び計装図のような物理モデルではなく、機能モデルです。接続は、情報をコマンドやフィードバックとして、送信することができ、これらは必ずしも物理的な接続に対応する必要はないことを示します。例えば、運航乗務員と航空管制の間の相互作用は、物理的な性質のものではないですが、それらは機能的なコントロールストラクチャーでモデル化されます。

コントロールストラクチャーは実行可能なモデルではない

コントロールストラクチャーは、実行可能なモデルやシミュレーションモデルではありません。実際には、コントロールストラクチャーには、（人間のような）実行可能なモデルが存在していないコンポーネントが頻繁に含まれています。その代わりに、STPA は必要な動作の制約、要求、及び望まれる仕様を慎重に導出するために使用できます。それは、望ましいシステム特性を守るために欠かせません。例えば、*図 2.8* のコントロールストラクチャーは、航空交通管制が必要な時にいつ

[7] コントロールストラクチャーはアクチュエータやセンサも含むが、後のシナリオ識別ステップで追加されます。

[8] コントロールストラクチャーは必要に応じて左から右へ描画されるが、ここでの議論を故意に簡略化します。

もフライトクルーに指示を送ること、いつも指示を送るための能力を有する（例：無線が常に動作可能である）こと、正しい指示が常に送ること、もしくは、常にパイロットが指示に従うこと、を想定していません。単に航空交通管制からフライトクルーに指示を送れるシステムが作られた/作られるということです。 STPA の次のステップでは、どのように非安全な動作（送信したが受信していない指示や非安全な指示の送信などを含み）が発生するかを慎重に検討します。コントロールストラクチャー自体は実行可能なモデルではありませんが、STPA の方法論は実行可能なモデルと仕様を作成するために使用できるような、正確な要求や他のアウトプットを生成します。

コントロールストラクチャーは、従うことを想定していない

コントローラとコントロールアクションを従われるものだと勘違いしないでください。コントローラがコントロールアクションを送るからといって、実際にそれが常に守られると意味するものではありません。同様に、フィードバック経路は、コントロールストラクチャーに含まれているからといって、必要な時に、実際にフィードバックが常に送信されることやそれが正確であることを意味するものではありません。コントロールストラクチャーのコントロールアクションとフィードバックは、単に、この情報を送信するようにメカニズム（システム設計の中で記載されることになる）が作られることを示しています。これは、暗示やコントローラとプロセスが実際にどのように動作するかについては何も暗に意味したり、想定したりしていません。つまり、STPA の主要な目標は、コントロールストラクチャーを分析し、各要素がどのように非安全で潜在的に期待しないような動作をするかを予測することです。

複雑さを管理するために抽象化を用いる

ハザード解析における最大の課題の 1 つは、システムの複雑さを管理することです。コントロールストラクチャーは、複雑さを管理しやすくするいくつかの方法で抽象化を用います。例えば、商用フライトではフライトクルーは、2、3 人の個々のパイロットから構成されることがあります。まず 3 人の別々のパイロットから始めてコントロールストラクチャーを乱雑にするよりもむしろ、一緒に単一のフライトクルーとして、正しく、コントロールアクションを与え、フィードバックを集めるといった、グループ化をすべきです。同様に、全ての個々の航空機サブシステムをはっきりとリスト化するよりむしろ、コントロール階層の２つのレベルとしてコントロールする航空機の自動化と物理的なプロセスをモデル化することによって、より抽象的なレベルで始めるべきです。

抽象化の原理は、コントロールストラクチャーでコマンドやフィードバック経路に適用することができます。コックピット内の個々のボタン、スイッチ、及びレバーをリスト化するよりむしろ、上昇の操作(マヌーバ)のような、より幅広いアクションで始めるのがよいかもしれません。その後、これらの幅広いアクションを、必要に応じて、ピッチ、推力(スラスタ)、または他のコマンドに精密化できます。個々のコマンドやセンサがまだわからない開発の初期段階では、この原則は特に便利です。

コントロールアクションの経路には、コントローラがコントロール対象のプロセスに作用するメカニズム（アクチュエータとよぶ）や、コントローラがコントロール対象のプロセスからのフィードバックを感知するメカニズム（センサとよぶ）を含むかもしれません。これらの詳細は、最初にコントロールストラクチャーを作成するまでは通常は抽象化されていますが、シナリオ作成ステップにおいてアクチュエータやセンサを含めるように後でコントロールストラクチャーを精密化します。

追加タイプの抽象化がコントロールストラクチャーで使用されます。航空機の自動化システムにフライトクルーから送られたフライトコマンドを考えてみましょう。遠隔操縦の UAV アプリケーションでは、これらのコマンドは、様々な異なるコンポーネントを通過するかもしれません。コマンドコンソール上の物理的ボタンから、デジタルパケットにコマンドをエンコードする組込みシステムを介して、UAV 上のネットワークスイッチ、無線送信機、衛星、無線受信機へ、などです。これは、初期のコントロールストラクチャーではコントロール経路に沿って、これらの詳細化ステップ

の全てを示す必要はありません。重要なのは、遠隔操縦が UAV にフライトコマンドを送信するいくつかの方法を持っているということです。

実際には、STPA を適用するための最も効率的な方法は、これらの設計の意思決定をする前、そのような詳細がわかる前に開始することです。上記の抽象的なコントロールストラクチャーは、STPA を使い始め、システムのコミュニケーション経路や他の部分に対する要求や制約を識別するために利用できます。その後、STPA 結果は、アーキテクチャ、基本及び詳細設計を進め、実装上の意思決定を行い、コントロールストラクチャーを改良するために使用できます。詳細がわかり、設計の意思決定がなされた場合でも、最初に、より詳細なコントロールストラクチャーのモデルを分析する前に、まず、より高い抽象度レベルで STPA を適用し、迅速に結果を出し、より広範な問題を識別するために役立ちます。

コントロールストラクチャーのモデル化

コントロールストラクチャーのモデル化は、抽象的なコントロールストラクチャーを使って始め、繰り返し詳細を追加します。多くの場合、システム内のコントロールストラクチャー及びコントロールループは明らかかもしれないし、前のアプリケーションから再利用することができます。コントロールストラクチャーが明らかにされていない場合、機能的なコントロールストラクチャーをモデル化し、詳細を追加する 1 つの方法を以下のガイダンスで説明します。付録 B には、様々な業界で使われてきたコントロールストラクチャーの例を入れています。

一つの開始方法は、制約をかけ、以前に識別されたハザードを防ぐために必要となる基本的なサブシステムを識別することです。例えば、十分な減速に関連するハザードと制約を導出しました。ホイールブレーキサブシステム、逆噴射(reverse thrust)、及び他のサブシステムを使用することによって、これらの制約がかけられます。これらのサブシステムを用いた、初期のコントロールストラクチャーを*図 2.9* に示します。

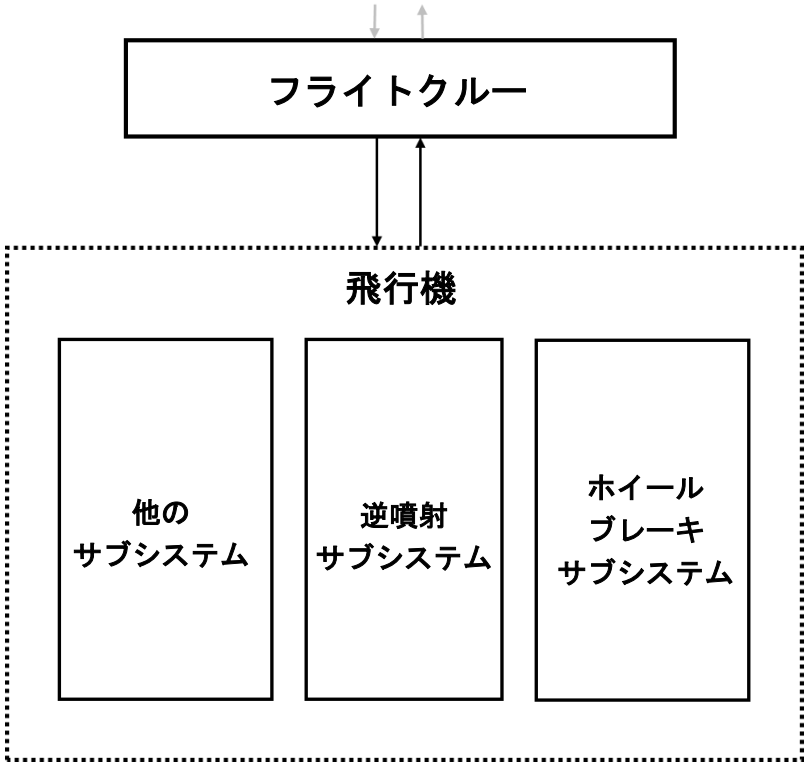


図 2.9：サブシステムを用いた精密化したコントロールストラクチャーの例

サブシステムが一度識別されると、どのようにコントロールするかを定義して、コントロールストラクチャーを精密化できます。この例として、ホイールブレーキサブシステムを改良してみましょう。ホイールブレーキサブシステムは、フライトクルーによってだけ直接手動で制御されていますか？ブレーキをコントロールすることのできる制御ユニット、自動化コントローラ、または他の人間も存在しますか？STPA を開発の後期段階で適用する場合には、これらの答えはわかっており、明らかかもしれません。STPA を早期のコンセプト開発段階で適用する場合には、上記のハザードと制約はこの決定のガイドに利用できます。例えば、着陸又は離陸中止時に自動的にブレーキをかけるために、ホイールブレーキサブシステムに自動ブレーキコントローラを含むことができます。

ホイールブレーキサブシステム内にブレーキシステム制御ユニット（BSCU）を入れた精密化したコントロールストラクチャーを*図 2.10* に示します。慎重にコントロールストラクチャーのより詳細を追加するように、「ズームイン」することに注意してください。

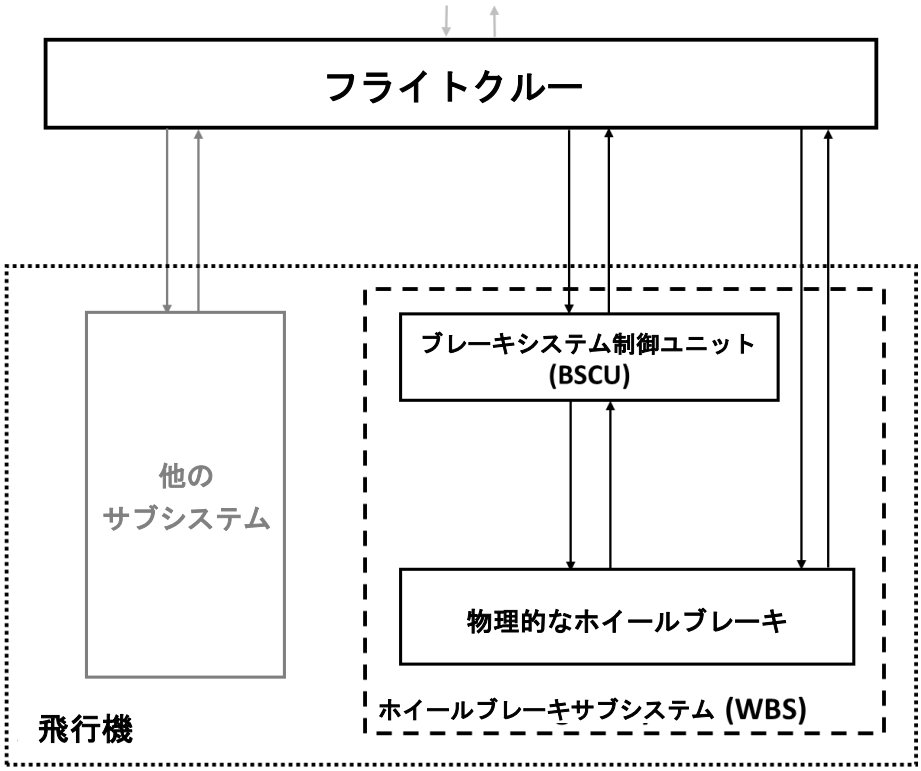


図 2.10：サブシステムコントローラを用いて精密化したコントロールストラクチャー

コントローラが一度識別されると、システムエンジニアは責任を割り振ることができます。これらの責任は安全制約の精密化です。安全制約を付加するために、各エンティティが一緒に何をすべきでしょうか？例えば、ブレーキがスリップを起こした時に、BSCU は自動的にブレーキをパルス化するという責任（アンチスキッド防止機能）があり、フライトクルーはブレーキをかける必要があるときを決める責任があるといったことです。

ホイールブレーキに関連する責任の例：

物理的なホイールブレーキ

- R-1： BSCU またはフライトクルーによって指示された時にホイールを減速する[SC-6.1]

BSCU

- R-2： フライトクルーから要求された時にブレーキを作動する[SC-6.1]
- R-3： スリップの場合のブレーキをパルス化する（アンチスキッド）[SC-6.2]

- R-4： 着陸又は離陸中止時にブレーキを自動的にかける（自動ブレーキ）[SC-6.1]

フライトクルー

- R-5： ブレーキが必要とされる時を決定する[SC-6.1, SC-6.3]
- R-6： ブレーキをどのようにかけるかを決定する：自動ブレーキ、通常ブレーキ、または手動ブレーキ[SC-6.1]
- R-7： ブレーキをかける準備のため、BSCU や自動ブレーキを設定する[SC-6.1]
- R-8： ブレーキや無効の BSCU をモニタし、故障時には手動ブレーキをかける[SC-6.1, SC-6.2]

次に、各コントローラのコントロールアクションは、これらの責任に基づいて定義できます。例えば、フライトクルーは、R-5 及び R-6 を満足するように、手動ブレーキ manual braking のコントロールアクションを送信する能力が必要になります。R 6 及び R-7 を満足するように、BSCU を arm 及び set する方法が必要となるでしょう。R-8 を満足するように、BSCU を disarm する必要があるかもしれません。責任に基づいてラベル付けしたコントロールアクションとともに修正したコントロールストラクチャーを*図 2.11* に示します。

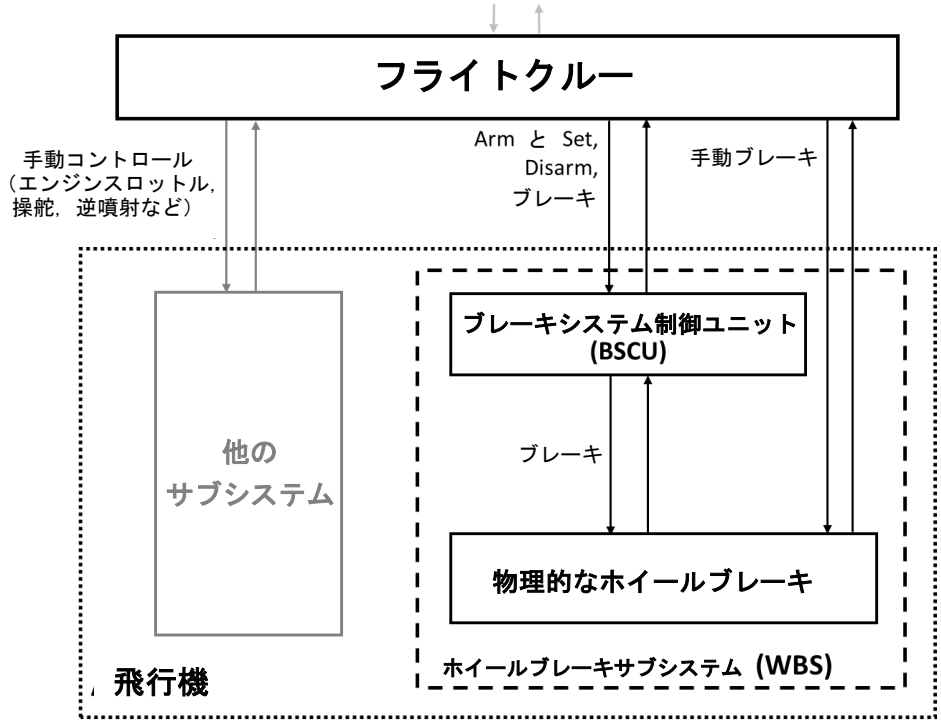


図 2.11：サブシステムへのプロセス割付け後の精密化したコントロールストラクチャー

この時点で、コントローラとコントロールアクションを識別し、ラベル付けしましたが、フィードバックはどうでしょうか？フィードバックは、コントローラが意思決定を行う必要があるようなプロセスモデルをまず識別することにより、コントロールアクションと責任からのフィードバックを得ることができます。その後、フィードバックと正確なプロセスモデルを作り出すために必要となる、フィードバックやその他の情報を識別することができます。例えば、R-3 は、スリップの場合に BSCU がブレーキをパルス化する必要があることを識別します。これを行うには、BSCU は、スリップが起きたこと（BSCU のプロセスモデルに含むべき情報）を知る必要があります。スリップを検出するためには、どのようなフィードバックが必要でしょうか？ホイールスピードのフィードバックを使用できるでしょう。同様に、R-8 からは、フライトクルーズは、故障時に BSCU を無効にする必要があるかもしれないことを識別します。これを行うには、フライトクルーは BSCU が故障していること（再度、プロセスモデルに入るべき情報）を知る必要があります。故障を検知するため

にはフライトクルーはどのようなフィードバックを使うことができるのでしょうか？ BSCU 欠陥(fault)についてのフィードバックを得ることができるでしょう。自動的に着陸又は RTO*(離陸中止)*（R-4）時のブレーキをかけるためには、BSCU は、航空機が着陸した時、または、いつ RTO が発生している時を知る必要があるでしょう。ホイールスイッチの重みや他の入力情報を使って、着陸や RTO 状態を検出することができます。これらの責任からフィードバックをどのように推論するのかを*表 2.2* に示します。

表 2.2：フィードバックを責任からどのように引き出すかを示す例

BSCU の責任	プロセスモデル	フィードバック
フライトクルーから要求された時ブレーキを作動する[SC-6.1]	フライトクルーによりブレーキをリクエストする	ブレーキペダルが作動した
スリップの場合にブレーキをパルス化する[SC-6.2]	飛行機がスリップしている	ホイールスピード 慣性基準装置(IRU)
着陸又は RTO 時にブレーキを自動的にかける（自動ブレーキ）[SC-6.1]	飛行機が着陸した 離陸を中止する	ホイール上の重み スロットルレバーの角度

コントロールストラクチャーは、このような責任を使って、再び「ズームイン」し、補助的な詳細情報を追加することで、更に精密化することができます。例えば、物理的なホイールブレーキには、コマンド（R-1）で、ホイールを減速する責任があります。これは、油圧で行うことができます。 R-6 では、BSCU は、ノーマルブレーキと自動ブレーキの両方を実行する必要があることを示しています。これらの 2 つの動作をコントロールするために BSCU 内の 2 つのコントローラを使用することができます。自動ブレーキコントローラと油圧コントローラです。

ラベルを付けたフィードバックと特定された BSCU 内部コントローラとともに、精密化したコントロールストラクチャーを*図 2.12* に示します。

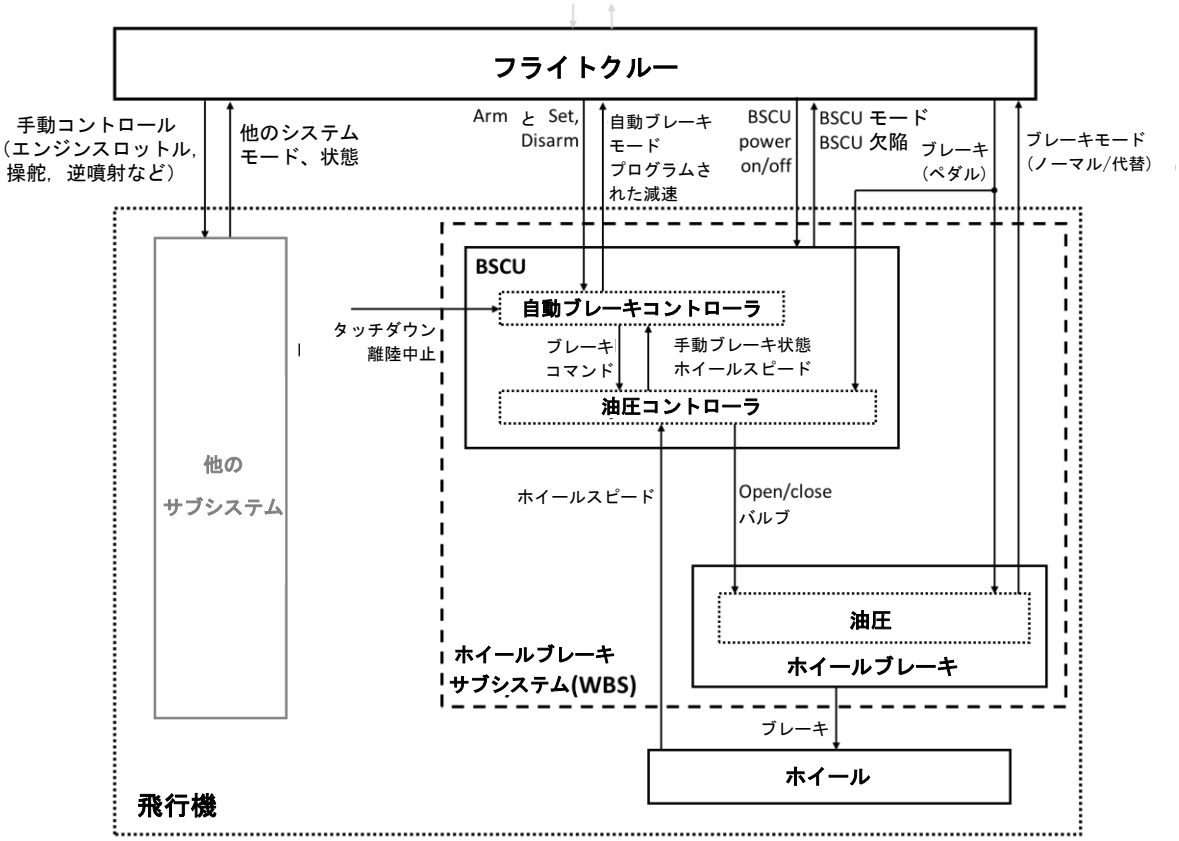


図 2.12：責任に基づいた精密化後のコントロールストラクチャー

コントロールストラクチャーをモデル化する時の共通質問

コントロールストラクチャーは、先に進める前に完全である必要があるか？

開発が終わる前の早期に STPA を適用している場合、一部の情報がわかっていない可能性もあり、コントロールストラクチャーが不完全である場合があります。分析は不完全なコントロールストラクチャーを用いて始めることができ、STPA が潜在的に欠けているフィードバック、コントロール、及び他の不一致を識別するのに役立ちます。それにより、開発に沿って、コントロールストラクチャーを精密化できます。次のステップを開始するために必要な最低限なものは、少なくとも、1 つのコントローラ、コントロールアクション、及びコントロール対象のプロセスです。しかし、一般的に、関連情報が意図的にコントロールストラクチャーから欠落していない場合には、STPA は、より簡単かつ効率的になります。

コントローラ、コントロールアクション、及びフィードバックの決定がなされた後で、STPA を開発段階の後期に適用する場合には、コントロールストラクチャーから高いレベルの情報を意図的に省略する理由はありません。見落とされた重要なフィードバックのような潜在的な欠陥を識別するために、STPA をその設計に適用できます。

矢印のラベルをどのように特定すべきか？

単に「コマンド」として全てのコントロールアクションを、単に「フィードバック」または「ステータス」として全てのフィードバックを、単に「コンピュータ」または「コントローラ」としてコントローラをラベリングするなどのコントロールストラクチャーにおける曖昧かつ漠然としたラベルを使用しないでください。コントロールアクションのラベルは、「Open/Close バルブ」のようなコマンドの種類（既知の場合）を示すべきです。そして、フィードバックは、「ホイールスピード」のような送信される情報の種類（既知の場合）を示すべきです。コントロールアクションとフィードバックを送信するために使われる特定の物理媒体は、この時点では重要ではありません。重要なのは、送信される機能的な情報です。例えば、「BSCU ボタン」、「共有バス」、「エンコードされたデジタルパケット」より、むしろ「BSCU power on/off」を使ってください。同様に、コントローラのラベルは、物理的な実装ではなく、動作の機能的な種類またはコントローラの役割を示すべきです。例えば、「シングルボードコンピュータ」より、むしろ「自動ブレーキコントローラ」のようなコントローラのラベルを使用してください。また、操縦(pilot)機能が搭載または地上のいずれで実施されても、「フライトクルー」や「パイロット」のようなラベルを使用してください。

コントロールストラクチャーには、全てのアクチュエータやセンサを入れるべきか？

具体的なアクチュエータやセンサは STPA を開始するためには必要でなく、まだ含める必要はありません（後のステップでそれらを入れます）。複雑さを管理しやすくするために、これらの詳細が STPA の後のシナリオ作成ステップ中に考えられるまで後回しにするのに役立ちます。例えば、*図 2.12* に示すホイールブレーキサブシステムのコントロールアクションとフィードバックは、電磁弁の異なる種類やいくつかの他の全く異なる実装を使用して実現されてもよいです。このフェーズで重要なのは、具体的な実装（わかっているかもしれないし、そうでないかもしれない）ではなく、与えられるコマンドやフィードバックの種類です。

物理的プロセス及び物理的相互作用をどのようにコントロールストラクチャーに差し込むか？

全てのモデルと同様に、コントロールストラクチャーモデルは、実世界の特定の側面を、他の部分を抽象化し強調抑制しながら、強調しています。コントロールストラクチャーは、設計上の欠陥、要求の欠陥、ヒューマンエラー、ソフトウエアエラー、更には従来の物理的なコンポーネント故障のような問題を識別するために非常に役に立つ、機能的関係性及び機能的相互作用を重要視します。コントロールストラクチャーモデルは、典型的には、コンポーネントの物理的近接や火災伝播のような、純粋に物理的または幾何学的関係をうまく表現してはいません。

上記の全てのレベルで、意思決定を行い、直接または間接的に物理的プロセスをコントロールするような、機能的なコントローラを特定しますが、物理的プロセスは、一般的にコントロールストラクチャーの最も低いレベルで識別されコントロールされます。

コントロールストラクチャーは、線形階層を必要としているか？

いいえ。いくつかのシステムでは、コントロールストラクチャーは、明らかに垂直線形階層の梯子のように表現されるかもしれません。他のシステムのコントロールストラクチャーは、同じ垂直レベルに、互いにコントロールしあわない複数のコントローラを入れることができます。コントローラは全てで、同じ物理的プロセスをコントロールすることができる、または、異なるプロセスをコントロールすることができます。

同じレベルのコントローラは、コントロール/フィードバックの関係性の範囲外でお互いにコミュニケーションすることもできます。そのようなコミュニケーションは、コントロールストラクチャー図の水平の矢印で表すことができます。一般的に、コントロールストラクチャーにおける相互作用（矢印）は、３つのカテゴリのいずれでも表すことができます。コントロールアクション、フィードバック、及びその他の情報です。

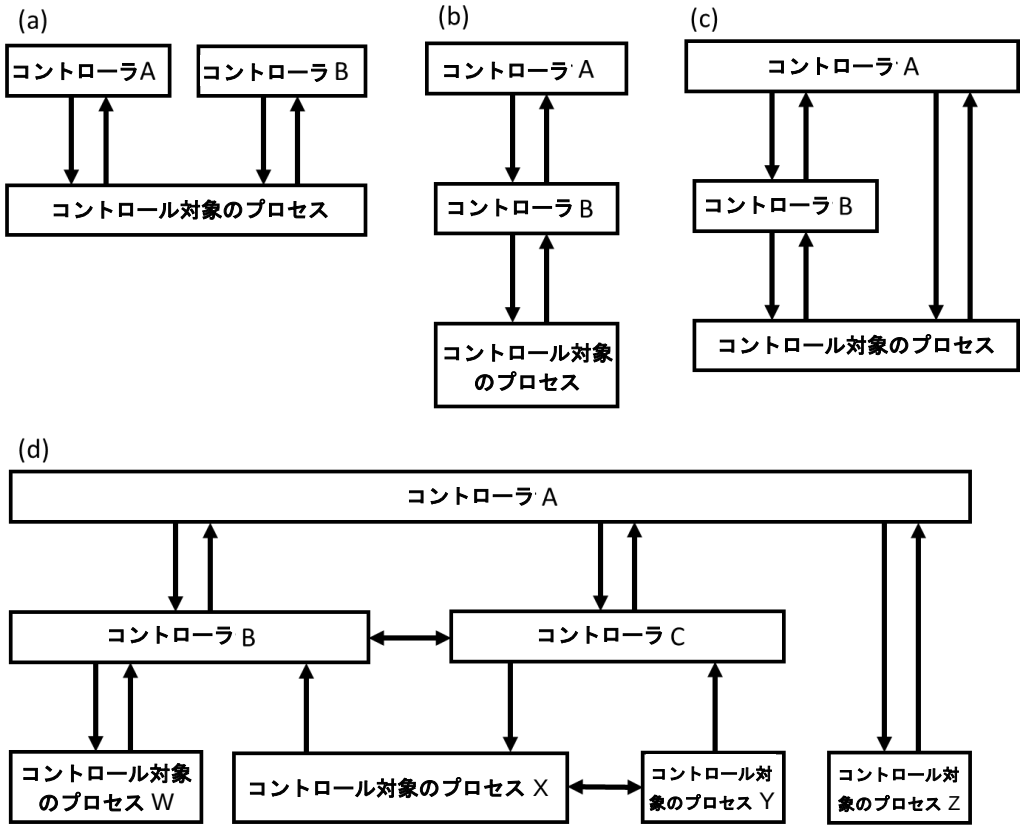


図 2.13：コントロールストラクチャーの異なる種類

いくつかの異なった種類のコントロールストラクチャーモデルを図 2.13 全体に示します。それぞれ、システムを設計し分析する際に考慮すべき、異なる利点とトレードオフ(tradeoff)を合わせて示します。（a）のコントロールストラクチャーは、共有プロセスをコントロール及び監視するために並列に動作する 2 つのコントローラを示しています。コントローラは、互いに相互作用はしておらず、どちらもが（STPA で識別する損失のシナリオにつながる可能性がある）その他のものをコントロールすることができません。このタイプのストラクチャーは、コントローラが素早くまたは互い

に独立して動作する必要がある場合には、有利である可能性があります。 例えば、非安全な状態が確認された場合にはいつでも、一連のコマンドを行わずに、オペレータに直接オペレーションを中止(アボート）できる能力を与えることが好都合かもしれません。しかし、このストラクチャーは、責任の拡散や他のコントローラのアクションに関する想定などの連携上の課題があるでしょう（後のステップでは、より詳細に、これらのことや他の問題を分析します）。

(b）におけるコントロールストラクチャーは、明確な一連のコマンドとして直線的なコントロール階層を示しています。この構造は、調整の問題に対処し、各コントローラの責任の違いを明確にさせるかもしれないが、コントローラ A のフィードバックやコントロールアクションへの反応時間の点から課題を示している可能性があり、コントローラ B の動作により依存しています。

コントロールアクションとフィードバックは、レベルを「省く」ことができます。コントローラがある上のレベルと下のレベルを超えた相互作用するのを妨げるモデルの制限はありません。例えば、(c)のように、高いレベルコントローラが他のコントローラをバイパスでき、低いレベルのプロセスを直接コントロールまたは監視することができます。コントローラ A は、通常はコントローラ B でブレーキを作動させるオートメーションを使用しているパイロットを表すことができます。しかし、緊急時にパイロットはコントローラ B をバイパスし、直接（手動で）ブレーキを作動させることができかもしれません。

（d）のような、より複雑なストラクチャも可能です。コントローラとプロセス間の 1 対 1 の対応付けが必要条件ではありません。コントローラは、1 つ以上のプロセスをコントロールができ、また、プロセスは、0 以上のコントローラによってコントロールできます。典型的には、各コントロールアクション経路は、並列したフィードバック経路とペアになるでしょうが、常にではありません。例えば、コントローラ C は、直接プロセス Y をコントロールできないかもしれません。その代わりに、プロセス X をコントロールし、プロセス Y で、X の効果をモニタすることができます。

どのようにして誰が誰をコントロールするのかを把握するか？

多くの場合、コントロールの関係性はシンプルで、かつ、わかりやすいです。それは、下位層に指示を出すスーパーバイザーやバルブの open/close のためにコマンドを送信するコンピュータのようなものです。コントロールの関係性は、また、あまり直接的ではありません。これは、優先順位を割り当てるマネージャ、標準的な操作手順を提供する航空会社、などのようなものです。これらは、全て、コントロールの異なる形態です。

前述したように、コントロールは、従うこととは違います。コントロールの関係性が存在するという理由だけでコントロールアクションを常に実行し、適切に動作することを意味するものではありません。実際に、私たちも、全てのコントロールアクションが、いつでも守られると期待していないかもしれません。いくつかの予期しない状況での損失を防ぐために必要となる時、コントロールアクションが、もっともな理由で無視されるかもしれません。 STPA は従うことを想定していません。また、これらの問題は慎重に後のステップで検討します。

同様に、別のコンポーネントの応答や影響動作を引き起こす能力は、必然的にコントロールを意味するものではありません。バルブを open するためのコントロールアクションはバルブが開く引き金になるが、一方で、高い温度を示すフィードバック信号が、コントローラにファンをオンにする引き金となります。影響を与える、または、応答する能力は、コントロールアクションをフィードバックと区別するのに十分ではありません。

コントロールには、目標を達成するために意図的な意思決定を必要とします。フィードバックは、特定のより高いレベルの目標を認識する（またはそれに責任を有する）ことなく、与えられますが、コントロールアクションは、常に目標を達成するために与えられます。コントロールはまた、典型的には、監督することを含んでいます、それは、より低いレベルのエンティティをモニタ

し、その動作や影響を評価するためにより良い能力や情報を有し、そして、高いレベルの目標の達成を確実にするためにガイドしたり、介入したりするものです。

コントロールの階層は、目標と責任の階層に密接に関連しています。上記の自動 BSCU コントローラは、着陸が検出された時にブレーキの引き金に関連する非常に限られた責任を有しているかもしれません。フライトクルーは、いつ着陸するかを決定するより高いレベルの責任、及び、ラフな着陸を避けるより高いレベルの目標を有しています。 BSCU は、必要かもしれないが、これらのより高いレベルのフライトクルーの目標を達成するには不十分な要素かもしれません。すなわち、フライトクルーは、より高いレベルの目標を達成するために、BSCU 及び他の多くの要素を監督しなければなりません。同様に、航空交通管制も、確実に複数の航空機の最小間隔を確保するような、より高いレベルの責任を有しています。また、多くのフライトにわたって、スループットを最大化するような、より高い目標を有しています。単一のフライトクルーは、目標を達成するために航空交通管制で使われる単に一つの要素であり、フライトクルーも、より高いレベルの目標にどのように貢献するか、または航空交通完成によってより高いレベルの目標が到達されるかどうか、ということに気づいていないかもしれません。

最後に、コントロールは、権限に密接に関係しています。航空交通管制とパイロットとの関係を考えてみましょう。航空交通管制は、パイロットの上位の権限を持っており、パイロットが一般的に従わなければならない、指示と離着陸許可を出します。パイロットは航空交通管制の上位の権限は有しておらず、単に従うように航空交通管制に指示を出すことができません。パイロットは緊急事態を宣言することができ、航空交通管制は、そのフィードバックに適切に応答する必要があるでしょう。ただパイロットは応答のトリガをかけることができるだけで、コントロールを意味するものではありません。フライトの安全を確保するために必要な時に、パイロットも、航空交通管制の指示に背くことができますが、従うことは、コントロールとは異なります。パイロットは航空機上で最終的かつ直接的な権限を持っていますが、彼らは ATC を上回る最終的な権限を持っておらず、全体的な空域を管理しません。 ATC は、一般的に、指示と離着陸許可を発して間接的に航空機をコントロールします。逆ではありません。第 5 章では、組織や社会分析に関連するような、コントロールストラクチャーやコントロールの関係性について更なる議論を行います。

初心者は時々これらの区別に苦労しますが、経験を積むことで、すぐに簡単にかつ自然にできようになります。また、コントロールストラクチャーに誰が誰をコントロールするかについての間違いは、通常その分析の結果に重大な影響はないことに気付かされます。例えば、コントロールアクション X がフェードバック X として誤って特徴づけられたと想定してみましょう。それがフィードバックとして特徴づけられたため、非安全なコントロールアクションを識別するステップでは、欠落または遅延したコントロールアクション X がどのようにハザードにつながるかを考慮しないでしょう。しかし、次のステップでは、潜在的なフィードバックの問題を調べ、欠落または遅延したフィードバック X がハザードにどのようにつながるか、同じシナリオを特定するでしょう。

コントロールストラクチャー図以外のものを文書化する必要がありますか？

コントロールストラクチャーを理解するのに役に立つ任意の追加情報を文書化することをお勧めします。基本的な説明、目的、特別な機能は、コントローラの責任、プロセスモデルなどのコントローラに関する追加情報を、コントロールストラクチャーをしっかり具体化するために文書化する必要があります。コントロールアクションとフィードバックに関する情報を明確にすることも役立ちます。特に、短いラベルからは重要な側面が明らかではないかもしれない時に役立ちます。同じことが、コントロール対象のプロセスにも当たります。一般的に、重要な明確にする情報や仮定はきちんとコントロールストラクチャー図と一緒に文書化すべきです。

コントロールストラクチャーをレビューする時は何をみつけるべきでしょうか？

コントロールストラクチャーの共通の間違いをみつけるには、次のヒントが役に立ちます：

コントロールストラクチャーにおける共通の間違いを防ぐためのヒント

- ラベルが特定の物理的な実装ではなく、送られた機能的な情報を説明していることを確認してください
- 情報の種類がわかっている場合には、単に「コマンド」または「フィードバック」のような曖昧で漠然としたラベルを避けてください
- すべてのコントロール対象の物理的プロセスは、1 つ以上のコントローラによってコントロールされていることを確認してください（必ずしも必要ではないが、しばしば間違いを指摘します）
- 競合とのギャップのために（トレーサビリティを含む）責任をレビューしてください
- 責任を満たすために必要なコントロールアクションが含まれていることを確認してください
- 責任を満たすために必要なフィードバックが含まれていることを確認してください（フィードバックが不明な、コンセプト開発の初期に適用する場合はオプション、後のステップで欠落しているフィードバックを識別することができます）

非安全なコントロールアクションの識別

コントロールストラクチャーをモデル化したら、次のステップ（*図 2.14* に示す）は、非安全なコントロールアクションを識別することです。

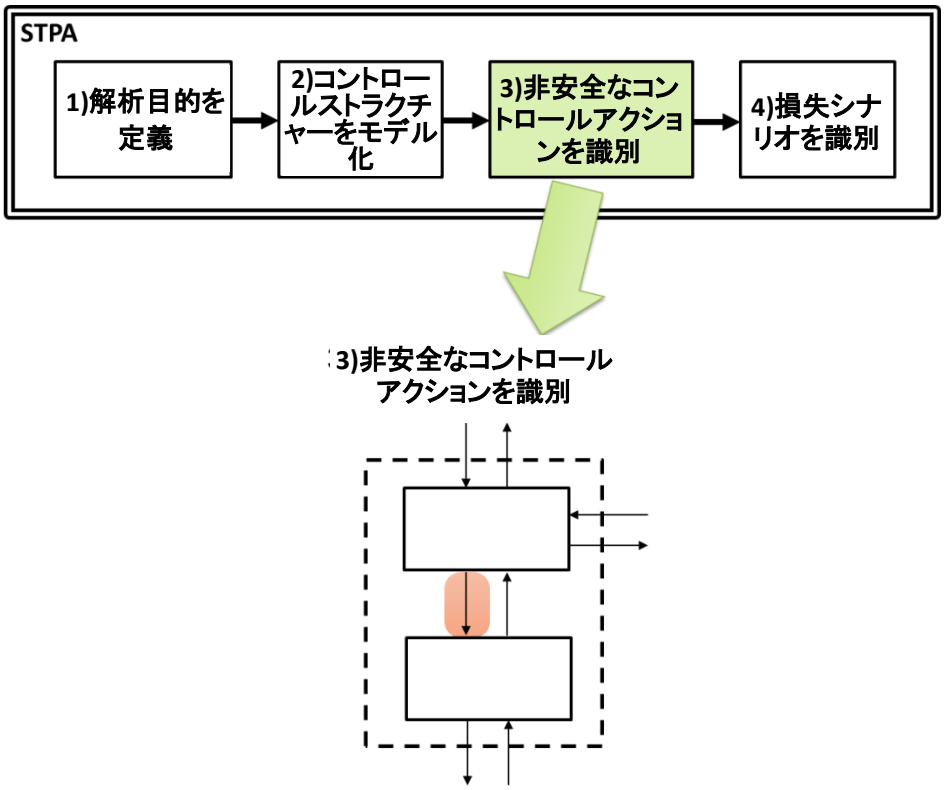


図 2.14：非安全なコントロールアクションの識別

定義：非安全コントロールアクション（UCA）は、ある特定のコンテキストと最悪の環境下で、ハザードにつながるコントロールアクションのことである[9]

以下の*表 2.3* に、BSCU コントローラのための UCA の例を示します。ホイールブレーキサブシステムのための UCA の大きなセットは、付録 C の例で見ることができます。

表 2.3：BSCU のための非安全コントロールアクションの例（一部分の例）

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	早過ぎ、遅過ぎ、誤順序	早過ぎる停止、長過ぎる適用
ブレーキ	UCA-1: 着陸滑走中 BSCU が作動している時、BSCU 自動ブレーキがブレーキコントロールを出さない[H-4.1]	UCA-2: 通常離陸中に BSCU 自動ブレーキがブレーキコントロールアクションを出す[H-4.3, H-4.6] UCA-5:着陸滑走中に BSCU 自動ブレーキが、不十分なブレーキレベルでブレーキコントロールアクションを出す [H-4.1] UCA-6: 着陸滑走中に BSCU 自動ブレーキが、方向性のある、または非対称のブレーキとなるブレーキコントロールアクションを出す [H-4.1, H-4.2]	UCA-3: 着陸後 BSCU 自動ブレーキがブレーキコントロールアクションを出すのが遅過ぎる(>TBD 秒) [H-4.1]	UCA-4: 着陸時に BSCU 自動ブレーキが、ブレーキコントロールアクションを止めるのが早過ぎる(TBD タクシー速度に達する前) [H-4.1]

コントロールアクションが非安全になるには、4 つの側面があります（上表の列に示す）。

1. コントロールアクションを与えないことがハザードにつながる。
2. コントロールアクションを与えることがハザードにつながる。
3. 潜在的には安全なコントロールアクションを与えるが、早過ぎる、遅過ぎる、または間違った順序である
4. コントロールアクションがあまりにも長く続いている（連続コントロールアクションであり離散的なものではない）あるいは、あまりにも早く止まる。

UCA-2 について考えてみましょう：

UCA-2：通常離陸中に BSCU 自動ブレーキがブレーキコマンドを出す [H-4.3, H-4.6]

H-4.3：減速が離陸時 V1 ポイント後に発生する、そして、H-4.6：離陸中に加速が不十分である、に至るため、UCA は非安全となります。全ての UCA は、一つ以上のハザード（またはサブハザード）にさかのぼることができ、全ての UCA の最後に括弧でトレーサビリティを記載することをお勧めします。

UCA では、コントロールアクションが非安全となるコンテキストを識別しなければなりません。このコンテキストが重要となります。ある航空機の BSCU がブレーキコマンドを出すことができることをわかっている、と仮定しましょう。このコマンドは、非安全なのでしょうか？

[9] 「非安全」という用語は STPA で識別されたハザードに関係するものです。前に議論したように、人間の死や負傷（伝統的な安全性）に関する問題を含んでいます。しかし、ミッションの損失、性能の喪失、環境の損失のような他の損失などを含めて幅広く定義できます。

そのコンテキストを考慮せずに評価することは不可能です。 UCA-2 には「通常の離陸中に」というコンテキストが含まれており、これがコントロールアクションを非安全にさせるのです。

あるコントロールアクションが常に非安全であった場合には、エンジニアは、システム設計に決してこれを含めることはないでしょう。全ての UCA は（どのコンテキストで）どのような条件下だとコントロールアクションが非安全であるのかを明確にする必要があります。その後、システムの設計からそれらの事例を排除したり、軽減する方法を見つけることができます。どのような関連コンテキストでも、環境条件、コントロール対象のプロセスの状態、コントロール状態、前のアクション、パラメータ（例えば、特定の降下速度がプログラムされている）を含め、UCA で参照することができます。 UCA の構築で、「～の時」、「‥の間」、「‥中に」のような言葉を使用すると、コンテキストを作成するのに役立つことがよくあります。

全ての UCA には、5 つの部分が含まれています：

UCA-2： BSCU Autobrake provides Brake command during a normal takeoff [H-4.3]
<ソース> <タイプ> <コントールアクション> <コンテキスト> <ハザードへのリンク>

最初の部分は、コントロールアクションを与えることができるコントローラです。２つ目の部分は、非安全なコントロールアクション（与えられる、与えられない、早過ぎるか遅過ぎる、早すぎる停止または長すぎる適用）のタイプです。３つ目の部分は、（コントロールストラクチャーからの）コントロールアクションまたはコマンド自体です。４つ目の部分は、上述したコンテキストであり、そして最後の部分は、ハザード（またはサブハザード）へのリンクです。

UCA は、たいてい上に示したのと同じ順序で各部分を書いていますが、場合によっては、別の順序で使うことが、もっと明確になり、あるいはもっと自然であるかもしれません。この順序は重要ではありません。キーポイントは、UCA は、これらの四つの部分を含有していることです。

UCA に関する一般的な質問

UCA はハザードが常に発生することを保証しているか？

*いいえ。*最良の場合のシナリオでは、UCA-2 が離陸中の早めの時期に発生する可能性があり、パイロットは UCA-2 が起こっていることを認識して、彼らはすぐに BSCU を無効にし、何も事故にはなりません。最悪のシナリオでは、パイロットが時間内に反応して回復することはできないかもしれません。あるいは BSCU を無効にするような彼らの行動には効果がないかもしれない、相当大きな追い風があり、V1 決定速度の直後に UCA-2 が発生する、あるいは他の要因が発生して航空機が滑走路を外れて走行する（H-4）などです。 STPA の目標は、最良や最悪の場合のシナリオはどれが起こる可能性が高いのか議論することでも、パイロットの能力と応答についての想定をすることでもありません。 BSCU をエンジニアリングする時に、そのフライトが最良あるいは最悪の場合の環境で発生するかどうかに関わらず、通常の離陸時にブレーキコマンドを出すことを防止したいと思います。このステップの目標は、ただ単に、防止されるべき行動を識別することです。 STPA は、最悪の場合の解析方法ではなく、最良の場合、平均的な場合、または最も可能性の高い場合のメソッドです。

既にセーフガードがある時、UCA を識別することがあるか？

システムには、とりわけハザードの原因となる非安全なコントロールアクションを防ぐための保護機能、冗長性、バックアップシステムのようなセーフガードがあるかもしれません。例えば、UCA-1「BSCU が作動している時に、BSCU 自動ブレーキが、着陸滑走中にブレーキコントロールアクションを出さない」は、ハザードになりません。それは、効果的に BSCU をバイパスする、独立した代替ブレーキシステムが設計に含まれており、これがいつでもマニュアルのブレーキを許可するからです。最良のシナリオでは、セーフガードは、意図した通りに動作し、有効かつ十分になり、

ハザードを回避することができます。しかし、最悪の場合のシナリオでは、セーフガードが意図したように働かない、十分ではない、目下の状況では有効ではないかもしれません。上記の理由と同様に、STPA は、最悪の場合の解析方法であり、セーフガードが存在する時も私たちは UCA を省略することはできません。

この論拠を理解する他の方法としては、我々はセーフガードが存在しない場合でも、UCA を防ぎたいのだと気づくことです。例えば、代替的なブレーキシステムを設計に入れることができるにもかかわらず、我々は意図的に UCA を準備するために BSCU を設計することはないでしょう。別のブレーキシステムなどのセーフガードが用意されていても、（例えば UCA-1 などを防ぐため）BSCU 自動ブレーキが適切なブレーキコントロールアクションを与えるということを、私たちは確認したいと思います。

実際には、セーフガードがわかり、設計に組み込まれる前の早い段階で、理想的には STPA が適用されます。 STPA は、防止しなければならない UCA を識別し、その後、UCA を防止または軽減するために、機能要求を導出して、設計上の決定を行うことに、UCA を使用します。潜在的に非安全な動作を識別した後、（デザインがまだ存在していない場合）具体的な設計機能を作成してセーフガードを追加します。あるいは（デザインが既に存在する場合）、既存の設計上の決定及びセーフガードの妥当性を明らかにすることができます。

非安全なコントロールアクションの最後の２種類は、両方ともタイミングである。この違いは何か？

UCA の３つ目の種類は、早過ぎる、遅過ぎる、順序が違うといった、間違ったタイミングで与えられたコントロールアクションについてです。非安全なコントロールアクションの４つ目の種類は、ある継続時間でのみコントロールアクションに適用されるもの、すなわち、連続または非ディスクリートなコントロールアクションのことです。

まず、BSCU 自動ブレーキにより出されるブレーキコマンドが連続コントロールアクションとして実装されているとします。換言すれば、自動ブレーキは、ブレーキコントロールアクションを出し始めた時にブレーキが適用されて、そのブレーキは、自動ブレーキがブレーキコントロールアクションを与えることをやめるまで、適用され続けます。ブレーキコントロールアクションの初期タイミングは、ハザードを引き起こす可能性があります：

> UCA-3：着陸後に BSCU 自動ブレーキがブレーキコントロールアクションを出すのが遅過ぎる（> TBD 秒）[H-4.1]

ブレーキコントロールアクションも、着陸時に遅延なく（TBD 秒以内）正確に与えられますが、その後、ただちに自動ブレーキがコントロールアクションを出すのをやめて、即座にブレーキは適用されなくなります。ブレーキが、あまりにも早く停止して効き目がありませんでした。この動作は、UCA-4 に反映されているように、コントロールアクションはもともと、適切な状況下で時間どおりに出されていても H-4.1 を引き起こす可能性があります：

> UCA-4： 航空機の着陸時、BSCU 自動ブレーキが、ブレーキコントロールアクションを止めるのが早過ぎる（TBD タクシー速度を達成する前） [H-4.1]

今度は、別の状況を考えてみましょう。：BSCU 自動ブレーキは、「Start Braking ブレーキを開始」と「Stop Braking ブレーキを停止」の 2 つの異なった別々のコントロールアクションを出します。個々のコントロールアクションは、有意義な継続時間を持っていないので、「早過ぎる停止/長すぎる適用」は、該当しません。代わりに、コントロールアクションのいずれかが、どのようにして早過ぎたり遅過ぎたりすることがあるのか、検討したいと思います。

> UCA-3： 着地後に BSCU 自動ブレーキがスタートブレーキ Start Brake コントロールアクションを出すのが遅過ぎる（> TBD 秒） [H-4.1]

UCA-4： 航空機の着陸時、BSCU 自動ブレーキがブレーキ停止 Stop Brake コントロールアクションを出すのが早過ぎる（TBD タクシー速度を達成する前）[H-4.1]

コントロールアクションが連続または不連続であることを分析することによって、同様の問題を網羅していることに注意してください。

非安全なコントロールアクションの種類ごとに、正確に一つの UCA を特定する必要があるか？

いいえ。4 種類すべての UCA を考慮するべきではありますが、全ての場合に適用できないかもしれません。上記の UCA-2、UCA-5、及び UCA-6 に示したように、あるタイプの UCA をいくつか識別することも可能です。一般的に、各カテゴリは、0、1、2、またはそれ以上の UCA を含んでいてもよいのです。

非安全なコントロールには 4 つ以上種類があるか？別のカテゴリが必要か？

これらの 4 つのカテゴリは、おそらく全部揃っているものです。非安全なコントロールアクションには他のカテゴリがありません。しかし、サブカテゴリがあります。例えば、二番目の種類の UCA が起こるかもしれない三つの異なった方法があります。

2. 非安全なコントロールアクションを与えること
    a. コントロールアクションが決して安全ではないかもしれない状態のコンテキストを考える
    b. 不足や過度のコントロールアクションが非安全になるコンテキストを考える
    c. コントロールアクションの方向性が非安全になるかもしれない状態のコンテキストを考える

最後の２つの例は、一つ以上のパラメータを含むコントロールアクションに関連するものです。例えば、BSCU ブレーキコントロールアクションは、典型的には、単純なバイナリの on/off コントロールアクションではありません。BSCU は、適用されるブレーキの特定量を指示することにより、ブレーキをコントロールします。UCA-2 は、通常の離陸中に（BSCU の指定するブレーキ量に関わらず）ブレーキコマンドを出すことが非安全である、と特定します。 BSCU が正しく着陸滑走時にブレーキコマンドを与えているにもかかわらず、BSCU は、不十分なレベルのブレーキを指示する場合、まだ非安全である可能性があるので、UCA-5 が存在します。ブレーキコマンドが非対称に送られて、航空機が間違った方に向きを変えてしまうことがあるので、UCA-6 が存在します。コントロールアクションが、1 つ以上のパラメータを特定できる時はいつでも、どの程度与えられたコンテキストに対して、これらのパラメータが、不十分、過度である、又は間違った方向であるか、検討することが重要です。

どんなシステムレベルのハザードにも関連していない、重要な UCA を特定した。どうすればよいか？

全ての UCA は、1 つまたは複数のシステムレベルのハザードにまでトレースできなければなりません。あなたが識別したハザードに一つも関連していない UCA を特定した場合は、何かハザードを見逃している可能性があります。UCA によって引き起こされるシステムレベルの状態や条件を識別して、新しいハザードを追加したり、新しい状態や条件が含まれるように既存の一連のハザードを見直したりすることを考えてください。 STPA は反復的であり、厳しく直線的に行う必要はありません。分析が進むにつれて、また、より多くの情報が利用できるにつれて、以前の結果を更新することができます。

説明されている UCA の成果や結果はどこにあるか？

全ての UCA は、システムレベルの影響と UCA の成果を取り込み、それが引き起こす可能性のあるハザードまたはサブハザードを参照する必要があります。多くのシステムでは、このような UCA

とハザードとの間の関連性は、明確かつ明白です。BSCU が着陸滑走時に不十分なブレーキしか与えないことは、明らかに H-4.1「着陸時の不十分な減速」に関連しています。

しかし、いくつかのケースでは、UCA とハザードとの間の関連性は、より複雑でわかりにくいかもしれません。特に、それが明らかでない、より複雑なアプリケーションでは、UCA とハザードとの関係の背後にある特別な推論を文書化することをお勧めします。推論は時々UCA にいくつかの単語を追加することによって文書化することができますが、推論が単純ではない場合、これは困難になることがあります。一般的な解決策は、必要に応じて、各々の UCA のコメントを文書化することです。 UCA のコメントは、その UCA のための理論的根拠、どのようにその UCA がハザードに至るのか、UCA の根拠とする想定、UCA の影響や結果、あるいは UCA を理解するために必要な他の情報、そしてそれがどうやってハザードに至る可能性があるのか、を記述することができます。

UCA の成果、結果または他の推論と、UCA コンテキストを混同しないように注意してください。よくある間違いは、UCA コンテキストを省略して、代わりに UCA 結果を含めることです。例えば：

正しい UCA： 通常離陸時に BSCU 自動ブレーキがブレーキコマンドを与える [H-4.3]

誤った UCA： BSCU 自動ブレーキはブレーキコマンドを出して衝突が生じる

UCA のコンテキスト（現在の状態や条件）が識別されない場合、要求を作成してシナリオを識別するという次のステップを実行することが困難または不可能になります。全ての UCA は、UCA のコンテキストを含んでいなければなりません。明確にするために、結果が含まれていてもよいです。

UCA のコンテキストにあるプロセスモデルの欠陥を特定するべきか？

UCA の原因と UCA を混同しないように注意してください。UCA のコンテキストは、特定のコントローラプロセスモデルや信じていること（真であってもなくても）ではなく、コントロールアクションを非安全にするかもしれない実際の（真の）状態や条件を特定するべきです。STPA の次のステップは、プロセスモデルの欠陥やその他の要因といったような、UCA の原因を特定します。例えば：

正しい UCA： BSCU 自動ブレーキは通常の離陸時にブレーキコマンドを出す

誤った UCA： 航空機が着陸していると間違えて考えている時に BSCU 自動ブレーキはブレーキコマンドを出す

人間の UCA の識別

同じアプローチは、UCA を識別するためのヒューマンコントローラに適用することができます。例えば、BSCU の電源を OFF にするフライトクルーのコントロールアクションについて考えます。*表 2.4* は、このコマンドに対応するフライトクルーの UCA の例をいくつか示します。ホイールブレーキに関連するフライトクルーのための追加の UCA の例を付録 C に記載しています。

表 2.4：フライトクルーのための非安全なコントロールアクション（例の一部）

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	早過ぎ、遅過ぎ、誤順序	早過ぎる停止、長過ぎる適用
BSCU 電源オフ	UCA-1: 異常な WBS 動作の場合にクルーが BSCU 電源をオフにしない[H-4.1, H-4.4, H-7]	UCA-2: アンチスキッド機能が必要で、WBS が正常に動作している時に、クルーが BSCU の電源をオフにする[H-4.1, H-7]	必要とされる時に自動ブレーキやアンチスキッド動作が完了する前に、クルーが早過ぎる BSCU 電源オフを行う [H-4.1, H-7]	N/A

電源オフのコントロールアクションは、この場合に継続時間はないので、最後の列は N / A（該当なし）であると考えています。クルーは単に、BSCU の状態をトグルする電源オン及び電源オフのコマンドを出しています。電源オンコマンドが出ていない、あるいはあまりにも遅く出てしまうことを考えている場合、あまりにも長く電源をオフにしたままにする BSCU に関連する問題が捕らえられます。人間がシステムコンポーネントの他のタイプと同じように処理されて、全体の分析に簡単に組み込むことができることに注意してください。

一般的なミスを防ぐためのヒント

次のようなヒントによって、UCA を識別する時の一般的な間違いを防ぐことができます：

UCA を識別する際の一般的なミスを防ぐためのヒント

- すべての UCA が、コントロールアクションが非安全な状態になるコンテキストを特定していることを確認してください。
- UCA のコンテキストが、実際の状態についての潜在的に信じていることではなく、コントロールアクションを非安全にさせるであろう実際の状態や条件を特定していることを確認してください。
- UCA のコンテキストが明確に定義されていることを確認します。
- UCA のコンテキストが、将来の影響や成果に置き換えられないことを確認します。
- すべての UCA が 1 つのまたは複数のハザードとリンクするようにトレーサビリティが文書化されていることを確認します。
- N/A であると想定するコントロールアクションの種類をすべてレビューして、それらが適用されないことを確認します。
- パラメータを持つ連続コントロールアクションすべてに対して、それらのパラメータの過剰、不足や方向の間違いを考慮していることを確認します。
- UCA の背後にある仮定や特別な推論が、すべて文書化されていることを確認します

コントローラの制約を定義する

定義：コントローラの制約は、UCA を防ぐために満足されるべき必要なコントローラの動作を特定する

UCA が識別されると、各コントローラの動作上の制約に変換することができます。例えば、BSCU コントロールアクションを分析する際に、通常の離陸時におけるブレーキコントロールアクションを出している BSCU は、ハザードにつながる可能性がある、と判断しました。したがって、BSCU は、そのコンテキストでブレーキコントロールアクションを出してはなりません。一般的に、各 UCA は、各コントローラのための制約を定義するように転化させることができます。*表 2.5* では、*表 2.4* にある UCA に由来することのできる BSCU 動作制約を示します。

表 2.5：フライトクルーのための非安全なコントロールアクション例（未完成）

非安全なコントロールアクション	コントローラ制約
UCA−1: 着陸滑走中 BSCU が作動している時、BSCU 自動ブレーキがブレーキコントロールアクションを出さない[H−4.1]	C-1： 着陸滑走中 BSCU が作動している時、BSCU 自動ブレーキは、ブレーキコントロールアクションを出さなければならない[UCA-1]
UCA−2: BSCU 自動ブレーキが通常離陸中にブレーキコントロールアクションを出す [H−4.3, H−4.5]	C-2： BSCU 自動ブレーキは、通常離陸中にブレーキコントロールアクションを出してはならない [UCA-2]
UCA−3:着地後に BSCU 自動ブレーキがブレーキコントロールアクションを出すのが遅すぎる(> TBD 秒) [H−4.1]	C-3： 着地後に TBD 秒以内に、BSCU 自動ブレーキは、ブレーキコントロールアクションを出さなければならない [UCA-3]
UCA−4: 着陸滑走時に BSCU 自動ブレーキが、ブレーキ停止コントロールアクションを出すのが早過ぎる(TBD タクシー速度に達する前に) [H−4.1]	C-4： 着陸滑走時に TBD タクシー速度に達する前に、BSCU 自動ブレーキは、ブレーキコントロールアクションを停止してはならない[UCA-4]
UCA−5:着陸滑走中に BSCU 自動ブレーキが、不十分なブレーキレベルでブレーキコントロールアクションを出す [H−4.1]	C-5： 着陸滑走中に BSCU 自動ブレーキは、TBD レベルより弱いブレーキをかけてはならない[UCA-5]
UCA−6: 着陸滑走中に BSCU 自動ブレーキが、方向性のある、または非対称のブレーキとなるブレーキコントロールアクションを出す [H−4.1, H−4.2]	C-6： 着陸滑走中に BSCU 自動ブレーキは、方向性のある、または非対称のブレーキをかけてはならない[UCA-6]

コントロールアクション分析のための入力と出力

図 2.15 に、コントロールアクション解析のための入力と出力をまとめます

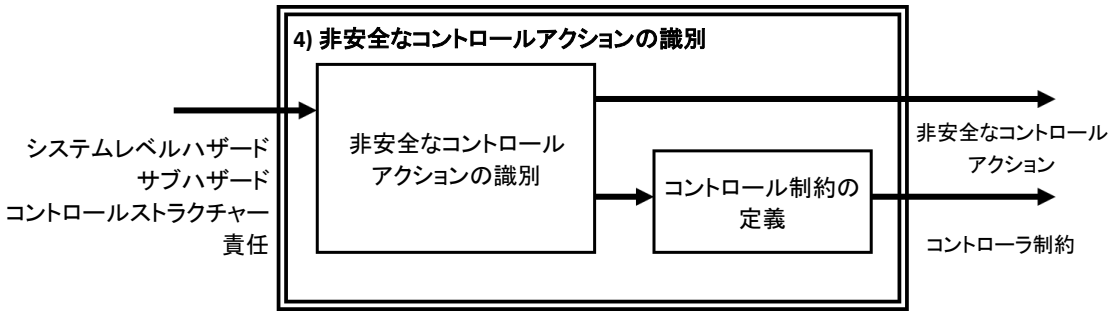


図 2.15：コントロールアクション解析の概要

損失シナリオの識別

非安全コントロールアクションが識別されたら、次のステップ（*図 2.16* に示されている）は、損失シナリオを識別することです。

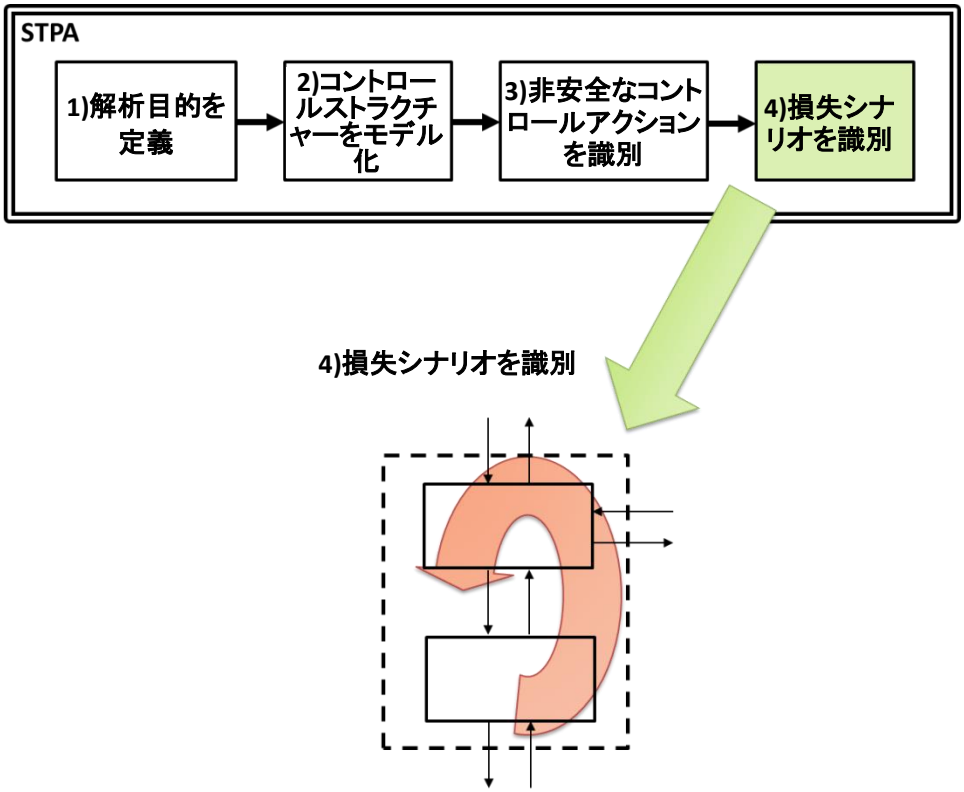


図 2.16：損失のシナリオの識別

定義：損失のシナリオは、非安全なコントロールアクション及びハザードに至る可能性のある、因果関係要因を説明するものである。

図 2.17 に示すように、損失のシナリオは２つのタイプを考慮する必要があります。

a) なぜ、非安全なコントロールアクションが起こるのか？

b) なぜ、コントロールアクションは、不適切に実行される、または実行されず、ハザードに至るのか？

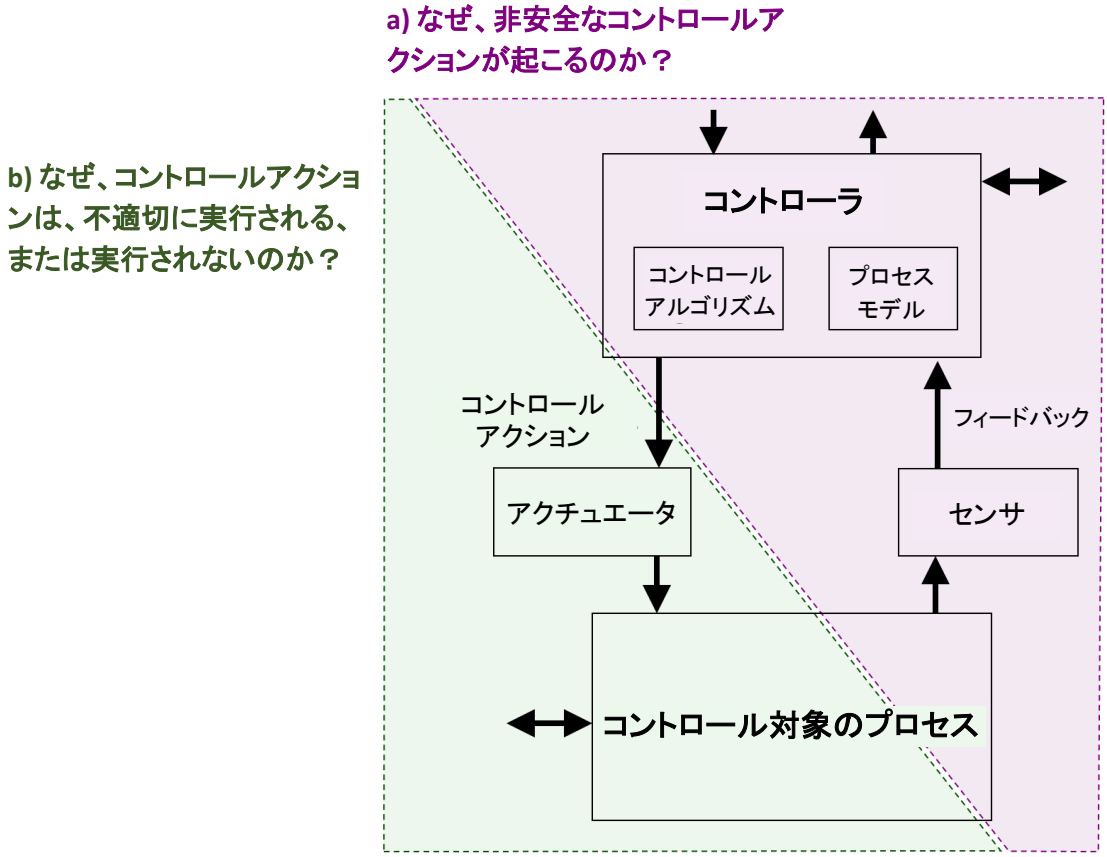


図 2.17：考慮すべき二つのタイプのシナリオ

図 2.17 は、センサ及びアクチュエータを含むことに注意してください。この時点までの分析で、私たちは存在し得るコントロールアクションとフィードバックを検討してきましたが、フィードバックが（例えばセンサで）測定または検出する方法や、コントロールアクションが実行される方法（例えばアクチュエータで）については検討していません。シナリオは非安全なコントロール及びフィードバックの特定の原因を識別するため、センサ及びアクチュエータを含むようにコントロールストラクチャーを改良することが役に立ちます。

a) 非安全なコントロールアクションにつながるシナリオの識別

シナリオのこのタイプは、コントローラが、そのコントロールアクションを与える（あるいは与えない）原因となりえるものは何なのかを説明するために、UCA から始めて逆さまに作業することによって作成することができます。一般的に、UCA につながるシナリオには以下のものが含まれます。

- コントローラに関連する故障（物理的なコントローラに対する）
  - コントローラ自体の物理的な故障
  - 電源の故障
  - など
- 不適切なコントロールアルゴリズム
  - コントロールアルゴリズム仕様の欠陥のある実装

    - コントロールアルゴリズム仕様に欠陥がある
    - コントロールアルゴリズム仕様は、時が経つにつれて、変更または劣化して不十分となる
- 非安全なコントロール入力
    - 別のコントローラから受けとった UCA（他のコントローラからの UCA を考慮する際に既に抽出されたもの）
- 不十分なプロセスモデル
    - コントローラが、間違ったフィードバック/情報を受けとる
    - コントローラが、正しいフィードバック/情報を受けとるが、誤って解釈する、または、それを無視する
    - コントローラが、必要な時に、フィードバック/情報を受けとらない（遅れる、または、全く受けとらない）
    - 必要なコントローラフィードバック/情報が存在しない

非安全なコントロールアクションを伴うシナリオを作成するために、*図 2.18* に示すような要因を考慮する必要があります。その要因は、UCA を引き起こした非安全なコントローラの動作から始まります。

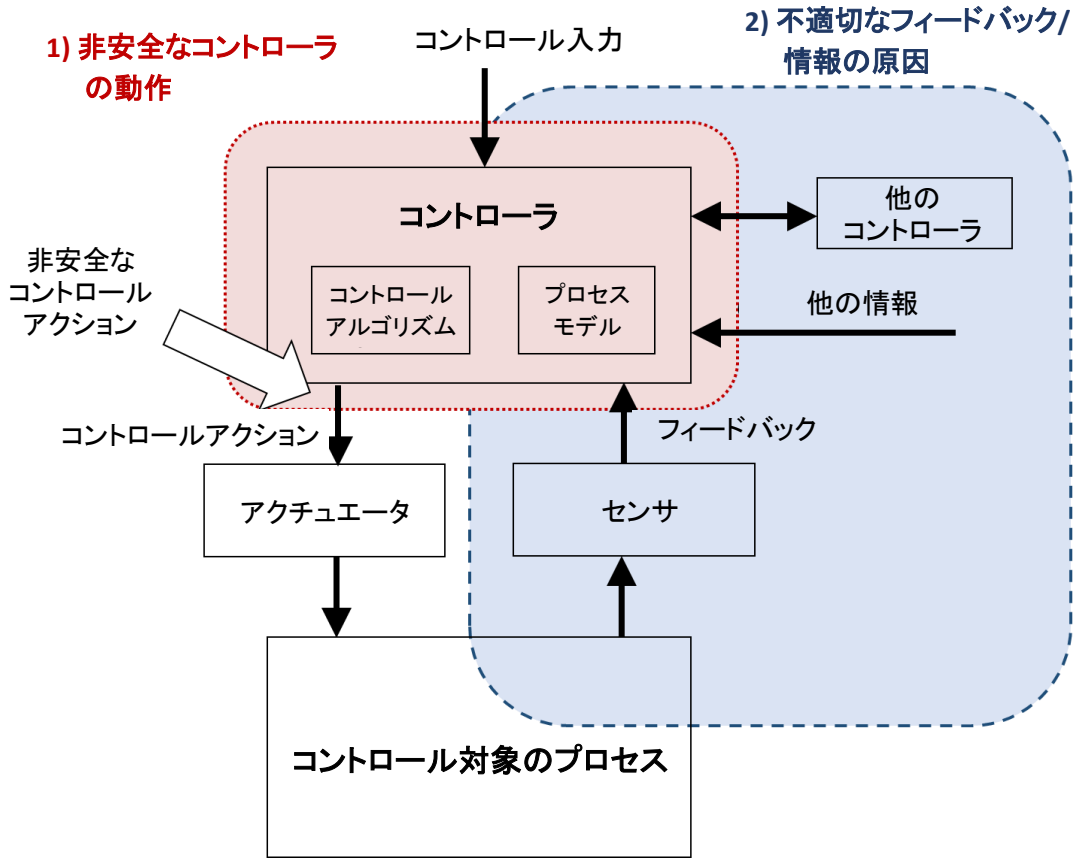


図 2.18：非安全なコントロールアクションが、(1)非安全なコントローラの動作及び(2)不十分なフィードバック及び他の入力、によって引き起こされる

1) 非安全なコントローラの動作

コントローラが非安全なコントロールアクションを与える（または与えない）ことに対して、一般的に４つの理由があります：

- コントローラを含む故障（物理的なコントローラに対する）
- 不適切なコントロールアルゴリズム
- 非安全なコントロール入力（別のコントローラからの）
- 不適切なプロセスモデル

物理的なコントローラの場合、UCA は、コントローラに関する故障により生じる可能性があります。 例えば、BSCU コントローラが故障するため、BSCU 電源が故障するために、BSCU は、ブレーキコマンドを出さないかもしれません。これらのシナリオを識別するには、UCA から始めて、コントローラを識別して、USA を説明することのできるコントローラに関連する物理的故障を識別します。例えば：

> UCA-1： 着陸滑走中に BSCU が作動している時、BSCU 自動ブレーキがブレーキコントロールアクションを与えない[H-4.1]
>
> UCA-1 のシナリオ 1：BSCU 自動ブレーキ物理コントローラが、ブレーキコントロールアクションが与えないことを原因として、着陸滑走中に BSCU が作動している時、失敗する[UCA-1]。結果として、着陸時には不十分な減速が与えられるかもしれない[H-4.1]

不適切なコントロールアルゴリズムも、UCA を引き起こす可能性があります。コントロールアルゴリズムは、コントローラのプロセスモデル、前のコントロール入力と出力、及びその他の要因に基づいて、コントロールアクションがどのように選択されるのか、特定します。ヒューマンコントローラの場合、時として、コントロールアルゴリズムは意思決定と呼ばれ、これは訓練、手順、及び過去の経験のような様々な要因によって形づくることができます。

これらのシナリオを特定するには、UCA から始めて、コントロールアルゴリズムがどのようにして UCA を引き起こす可能性があるのか、識別します。例えば：

> BSCU 自動ブレーキの例：
> UCA-3：着地後に BSCU 自動ブレーキがブレーキコントロールアクションを出すのが遅過ぎる（> TBD 秒）[H-4.1]
> UCA-3 のシナリオ 1：航空機は着陸するが、BSCU 内のプロセッシングが遅延して、ブレーキコントロールアクションが出されるのがあまりにも遅くなってしまう[UCA-3]。結果として、着陸時の減速が不十分になるかもしれない[H-4.1]
>
> 人間のクルーの例：
> Crew-UCA-1：異常な WBS 動作が発生した時に、クルーが BSCU Power off を出さない[H-4.1、H-4.4]
> Crew-UCA-1 のシナリオ 1：異常な WBS 行動が発生し、クルーに BSCU フォールト表示が出される。クルーが BSCU [Crew-UCA-1]を Power off しない。これは、操作手順で、BSCU フォールト表示を受け取った時に、クルーが BSCU を Power off しなければならないことが明記されていないためである。

一般的には、コントロールアルゴリズムの欠陥は、以下から起因する可能性があります：

- コントロールアルゴリズム仕様の欠陥のある実装
- コントロールアルゴリズム仕様に欠陥がある
- コントロールアルゴリズム仕様は、時が経つにつれて、変更または劣化して不十分となる

これらのタイプの欠陥それぞれに特有の根拠は、検討されているアプリケーションに依存するかもしれません。

コントロールアルゴリズムが、前のコントロールアクションは適切に実行されていると仮定した時に、ある共通のコントロールアルゴリズムの欠陥が生じます。コントロールアクションが成功したかどうかを示すためのフィードバックがない場合、この欠陥は特に重要です。例えば、直前のブレーキコントロールアクションが成功して飛行機は既にブレーキをかけていると間違って推定するために、BSCU 自動ブレーキが、ブレーキコントロールアクションを与えないかもしれません。

セキュリティに関連するシナリオを含めるには、ある追加の可能性を考慮する必要があります。コントロールアルゴリズムの欠陥が攻撃者(adversary)によって注入されたか、そしてその方法についてです。

他のコントローラからの非安全なコントロール入力も、UCA を引き起こす可能性があります。他のコントローラに対して非安全なコントロールアクションを識別する時に、これらは、前のステップの間に見つけることができます。

最後に、不十分なプロセスモデルは、非安全なコントロールアクションを引き起こす可能性があります。上記で説明したように、プロセスモデルは、コントロールアクションを決定するためのコントロールアルゴリズムによって使用されるコントローラの内部の信じていることを表します。コントローラのプロセスモデルが現実とは一致しない場合、プロセスモデルの欠陥が発生します。プロセスモデルの欠陥が起きるかもしれない理由は以下の通り：

- コントローラが間違ったフィードバック/情報を受けとる
- コントローラが、正しいフィードバック/情報を受けとっているが、誤って解釈する、または、それを無視する
- コントローラが、必要に時に、フィードバック/情報を受けとっていない(遅れている、または、全く受けとらない)、
- 必要なコントローラのフィードバック/情報が存在しない

これらの問題は、アプリケーションに応じて、様々な方法で発生する可能性があります。不正確なフィードバック/情報がコントローラによって受けとられるかもしれません。解決することのできない競合している情報、あるいは誤って解決されるかもしれないコンフリクトなど、正しくないフィードバック/情報がコントローラによって受けとられるかもしれません。コントローラが無効になっている、切られている、他のタスクで忙しい、更新するための必要条件でプロセスモデルを見落としている、など何か他の理由で、正しいフィードバック/情報を受けとったが、コントローラが無視していることがあります。プロセスモデルが誤って更新される、誤ったプロセスモデルが更新される、フィードバック/情報は何か他のもの、または他のエラーを表していると考えられている場合、コントローラの解釈の問題が発生することがあります。フィードバックが全く受けとられない、特に、コントローラはフィードバックの代わりにデフォルト値を推定しているかもしれない場合、あるいは、ばらばらの順序で情報を受けとっているなどでフィードバックが遅延している場合、フィードバック・情報は、必要な時に受信されていないかもしれません。最後に、必要なフィードバック/情報がコントロールストラクチャーまたは設計は存在しないかもしれず、不適切なプロセスモデルという結果に終わるかもしれません。

これらのシナリオを特定するには、UCA から始め、UCA を引き起こす可能性のあるコントローラのプロセスモデルを識別しましょう。 現在の状態またはモードに関する信じていること(belief)、前の状態、能力、動的な動作(dynamic behavior) [10]、前の動作やアクション、将来の状態や動作（予

[10] ここでコントロール理論の調整に関する問題を捉えることができます。プロセスモデルには、コントロール対象の動的な特性に関するコントローラの考えが含まれており、これらの考えが正しくない時に調整の問題が起こります。例えば、PID コントローラの、比例、 積分、微分の用語は、コントロール対象のダイナミクスに関する考えをあ

測）、現在コントロールされているプロセスについての信じていること、他のコントロール対象のプロセス、システムにある他のコントローラ（特に調整のために必要な信じていること）、アクチュエータ、センサ、あるいはシステムや環境の関連する他の側面、を考えてください。

UCA を引き起こす可能性のある、関連するプロセスモデルが識別されると、受け取ったフィードバックやその他の情報（またはその欠如）によって、どのようにプロセスモデルが思い浮かぶのかを識別します。

例：

UCA-2： BSCU が作動している時に、BSCU 自動ブレーキが、着陸滑走中にブレーキコントロールアクションを出さない[H-4.1]

UCA を引き起こす可能性があるコントローラのプロセスモデル（信じていること）：コントローラは、航空機が既に地上で停止していると信じている

コントローラが正しいフィードバックを受信するが誤って解釈：ホイール速度信号が瞬間的にアンチスキッド動作中にゼロに達するかもしれず、欠陥のあるプロセスモデルをもたらす。

UCA-2 のシナリオ 1：BSCU が作動していて、航空機が着陸滑走を開始する。 BSCU は、航空機が既に停機場所(stop)に来ていると間違えて考えているため、BSCU は、ブレーキコントロールアクションを出さない[UCA-2]。受信したフィードバックが、着陸滑走中、一時的にゼロ速度を示している場合、この欠陥のあるプロセスモデルが思い浮かぶ。受信したフィードバックは、航空機が停止していなくても、アンチスキッド操作中にゼロ速度を瞬間的に示すことがある。

UCA を引き起こす可能性があるコントローラプロセスモデル（信じていること）：航空機がフライト中である

必要な時にコントローラが情報を受信しない：タッチダウン表示を受信していない

UCA-2 のシナリオ 2：BSCU が作動して航空機が着陸滑走を始める。航空機は空中にあり、タッチダウンしていないと BSCU が誤って考え、BSCU がブレーキコントロールアクションを出さない。[UCA-2] タッチダウンの表示がタッチダウン時に受信されていない場合は、このような欠陥のあるプロセスモデルが思い浮かぶ。 <Why？このシナリオの検討は、まだ次のセクションに続きます。>

不適切なフィードバック/情報を含んでいる任意のシナリオは、フィードバック/情報が不十分であるかもしれない理由を説明するために、完了させるなければなりません。不十分なフィードバックと情報が、その理由を理解しないままでは防ぐことはできません。

2) 不適切なフィードバックと情報の原因

UCA を引き起こす可能性があるフィードバックや情報（またはその欠如）を、シナリオが識別するときにはいつでも、我々は、これらの問題を引き起こす可能性があるかを説明するために、フィードバック/情報は、どこから来るのか検討する必要があります。フィードバックは、（通常はセンサを介して）コントロール対象のプロセスから入ってきて、他の情報は、他のプロセス、他のコントローラ、あるいはシステムや環境内の他のソースから来るのかもしれません。

一般的に不適切なフィードバックや情報に関連するシナリオには以下のようなものがあります：

らわしています。これらの信念が正しくないとき、安定性その他の問題が起こり、コントロールアルゴリズムは、結果として非安全なコントロールアクションを作り上げるかもしれません。

- フィードバックや情報を受信しない
  - センサ（複数可）はフィードバック/情報を送信したが、コントローラが受信しない
  - センサ（複数可）はフィードバック/情報を送信していないが、受信される、または、センサ（複数可））に適用する
  - フィードバック/情報が受信されず、またはセンサ（複数可）に適用する
  - フィードバック/情報はコントロールストラクチャーに存在しない、またはセンサー（複数可）が存在しない
- 不適切なフィードバックを受信する
  - センサ（複数可）は適切に応答しているが、コントローラが不適切なフィードバック/情報を受信する
  - センサ（複数可）が、（受信される、またはセンサ（複数可）に適用する）フィードバック/情報に対して不適切に応答する
  - センサ（複数可）に必要なフィードバック/情報を提供するような性能がない、あるいは、そのように設計されていない

送信したが受信していなかった、あるいは受信が不十分であるようなフィードバック/情報を含んだシナリオは、伝送エラー、コミュニケーションの喪失、コミュニケーションの遅れ（フィードバック/情報が送信されているが違った順序で受信されていることを含める）、その他の問題によって引き起こされるかもしれません。不十分なセンサ応答または無応答のセンサがあるシナリオは、センサの故障、センサへの電力の損失、センサ動作または測定の不正確さ、センサエラーまたは誤作動、センサ応答の遅延、不適切な構成、継時的なセンサの劣化や変更、センサ環境の予期しない条件、他の問題によって引き起こされるかもしれません。更に、設計のエラー、仕様上の欠陥、コントロール対象のプロセスに関する誤った仮定、間違った条件を測定するセンサ、正しく報告してはいるが誤解を招くようなセンサ（ホイールがロックされているのに航空機が動いている時のゼロホイール速度など）、その他の問題によって、センサは、必要なフィードバックを与えることができないのかもしれません。

システムの実際の真実の状態（UCA のコンテキストで識別されている）を示す、そのフィードバック/情報を受信できる理由を特定することにより、フィードバック/情報が入ったシナリオを完成させることができます。

例えば、前述の UCA-2 のシナリオ 2 を完了してみましょう：

<u>UCA コンテキストからの真実の状態</u>：航空機は着陸滑走の状態にある（上記のシナリオ 2 を参照）。

<u>受信情報</u>：タッチダウン時にタッチダウン表示を受信しない（上記のシナリオ 2 参照）

<u>どのようにして、これが真実の状態とされてしまうのか</u>：報告されたホイールの速度が不足している、報告されたホイール上の重量が十分でない、ホイールの速度あるいはホイール上の重量の表示が遅れる、など。

<u>UCA-2 のシナリオ 2</u>：BSCU が作動して航空機が着陸滑走を始める。航空機は空中にあり、タッチダウンしていないと BSCU が誤って考え、BSCU がブレーキコントロールアクションを出さない。[UCA-2] タッチダウンの表示がタッチダウン時に受信されていない場合は、このような欠陥のあるプロセスモデルが作られる。以下のいずれかが発生した場合、必要なときにタッチダウン表示が受信されないかもしれない：

- 濡れた滑走路によるホイールのハイドロプレーン
- 使用されるフィルタリングによるホイール速度フィードバックの遅延
- 着陸時の横風による空中/地上の表示の不一致
- ホイール速度センサの故障

- 空中/地上のスイッチの故障
- など

結果として、着陸時に与えられる減速が不十分になるかもしれない[H-4.1]

セキュリティに関連した原因を含めるには、追加の可能性を考慮する必要があります。それは、識別されたフィードバックやその他の情報が、攻撃者によってどのような影響を受ける可能性があるのかを識別することです。さらに具体的には、どのようにして、それらのものが投入されて、不正に改ざんされ、傍受されて、あるいは敵に開示されているのでしょうか？例えば、セキュリティが含まれているならば、次のような原因が上記のシナリオ 2 に追加されるかもしれません。

- 攻撃者が不十分なホイール速度を示すフィードバックを偽装する
- 攻撃者の DoS 攻撃の実行により、ホイール速度フィードバックが遅れる
- 攻撃者によって正しいホイール速度フィードバックが傍受されて遮断される
- 攻撃者がホイール速度センサへの電力をディスエーブルにする

b) コントロールアクションが不適切な実行をされる又は実行されないシナリオの識別

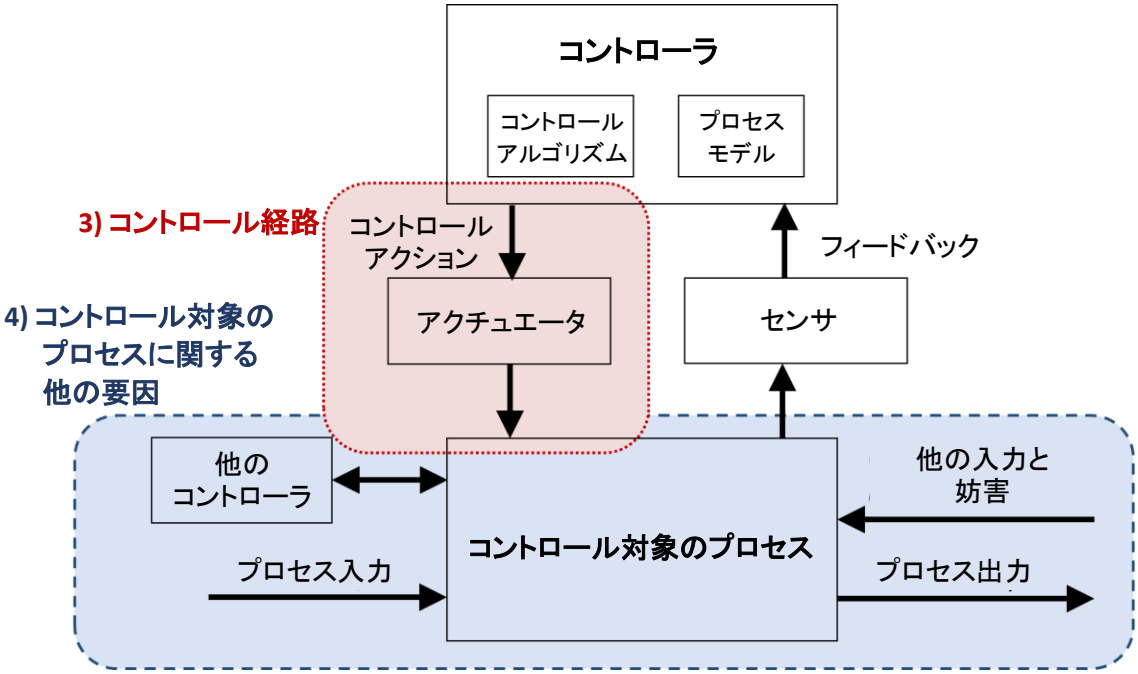


図 2.19： コントロールプロセスに影響を与えるような 1) コントロール経路と 2）他の要因を表す一般的なコントロールループ

ハザードは UCA が原因で発生する可能性がありますが、コントロールアクションが正しく実行されない、または、実行されていない場合は、UCA がなくても発生することがあります。これらのシナリオを作成するために、コントロール経路と同様に、*図 2.19* に示すようにコントロール対象のプロセスに影響を与える要因に影響するファクターを考慮しなければなりません。

1) コントロール経路を含むシナリオ

コントロール経路は、コントロールプロセスにコントロールアクションを伝送します。このコントロール経路は、単純なアクチュエータで構成できるかもしれませんし、一連のアクチュエータを含んでいるかもしれません。あるいは、スイッチ、ルータ、衛星、または他の機器との複雑なネットワークを介して、コントロールアクションを伝送できるかもしれません。それがどのように実装

されるかに関わらず、どのようにして問題は、この経路に沿って不適切に実行された、または実行されなかったコントロールアクションを生じる可能性があるのか、識別する必要があります。

一般的には、コントロール経路を含むシナリオには次のものが含まれることがあります：

- コントロールアクションが実行されない
  - コントローラはコントロールアクションを送信したが、アクチュエータ（複数可）が受信しない
  - コントロールアクションをアクチュエータ（複数可）は受信したが、アクチュエータ（複数可）が応答しない
  - アクチュエータ（複数可）が応答するが、コントロールアクションが適用されない、あるいはコントロール対象のプロセスが受信しない
- コントロールアクションが不適切に実行される
  - コントローラはコントロールアクションを送信したが、アクチュエータ（複数可）は不適切な受信をする
  - コントロールアクションをアクチュエータ（複数可）が受信したが、アクチュエータ（複数可）の不適切に応答する
  - アクチュエータ（複数可）が適切に応答するが、コントロールアクションが不適切にコントロール対象のプロセスで適用または受信される
  - コントロールアクションをコントローラが送信していないのに、まるで既に送信しているかのようにアクチュエータまたは他の部分が応答する

不適切に受信された或いは受信されていないコントロールアクションのあるシナリオは、（送信されたが異なる順序で受信されているコントロールアクションを含む）通信の遅延、伝送エラー、通信の喪失、その他の問題によって引き起こされる可能性があります。不適切なアクチュエータ応答または無応答のシナリオは、アクチュエータの故障、アクチュエータへの電力の損失、アクチュエータ動作の不正確さ、アクチュエータエラーまたは誤作動、アクチュエータの応答の遅れ、アクチュエータにより受信されたその他のコマンド（他のコントローラからの潜在的に競合しているコマンドを含む）、アクチュエータによって使用される誤った優先順位スキーム、誤った構成、アクチュエータへの継時的な劣化や変更、アクチュエータ環境における予期しない条件、又は他の問題によって引き起こされているかもしれません。

これらのシナリオを作成するには、コントロールアクションから始めて、不適切な実行または何も実行しなかったことがアプリケーションのために何を意味するのかを識別して、その動作にコントロール経路がどのように貢献できるのかを識別します。

例:

コントロールアクション: BSCU がブレーキコマンドを送る

実行なし：ブレーキがかけられない

不適切な実行：不十分なブレーキをかける

シナリオ 1：BSCU は着陸時にブレーキコマンドを送信するが、ブレーキはアクチュエータの故障で、かけられない。その結果、着陸時に与えられた減速は不十分になる可能性がある[H-4.1]

シナリオ 2：BSCU は着陸時にブレーキコマンドを送信するが、アクチュエータの応答の遅れによって、不十分なブレーキングがかけられる。その結果、着陸時に与えられた減速は不十分になる可能性がある[H-4.1]

シナリオ 3：BSCU は着陸時にブレーキコマンドを送信するが、配線の問題によりアクチュエータが受信しない。その結果、着陸時に与えられた減速は不十分になる可能性がある[H-4.1]

どうしてコントロールアクションが送信されず、まるでコントロールアクションが送信されたかのようにアクチュエータまたは他の部分が応答するかを検討することも重要です。これらのシナリオは、与えられたコントロールアクションに対する UCA と類似しているでしょう。

例：

コントロールアクション：BSCU がブレーキコマンドを送信しない

不適切な実行：通常の離陸中にブレーキをかける（UCA-2 と同様）

シナリオ 4：BSCU がブレーキコマンドを送信しないが、油圧バルブの故障によってブレーキがかけられる。その結果、離陸時に加速が不足する可能性がある[H-4.6]

セキュリティ上の問題を入れるためには、ここでは追加の可能性を考慮する必要があります。それは、識別されたコントロールアクションが、攻撃者によってどのような影響を受ける可能性があるのかを識別することです。さらに具体的にどのようにして、それらのものが投入されて、不正に改ざんされ、傍受されて、あるいは敵に開示されているのでしょうか？例えば：

シナリオ 5：BSCU がブレーキコマンドを送信するが、攻撃者がブレーキコマンドをブロックする、サービス拒否攻撃(DoS 攻撃)を実行したため、ブレーキがかけられない。その結果、着陸時に与えられた減速が不十分になる可能性がある[H-4.1]

2) コントロール対象のプロセスに関連するシナリオ

コントロールアクションをコントロール対象のプロセスに伝送または適用する場合でも、効き目がないかもしれないし、また他のコントローラによってオーバーライドされることがあります。

一般的にコントロール対象のプロセスに関連するシナリオに以下が含まれます：

- コントロールアクションが実行されない
  - コントロール対象のプロセスによって、コントロールアクションが適用される、または受信されるが、コントロール対象のプロセスは応答しない
- コントロールアクションが不適切に実行される
  - コントロール対象のプロセスによって、コントロールアクションが適用される、または受信されるが、コントロール対象のプロセスは不適切に応答する
  - コントロール対象のプロセスによって、コントロールアクションが適用される、または受信されるが、まるで、コントロールアクションが適用または受信されたかのようにプロセスが応答する

これらのシナリオは、プロセス入力によって引き起こされる可能性があります。これは、欠落していたり、不適切であったり（不適切な油圧など）、外部または環境の外乱、コンポーネントの故障、プロセスによる応答遅延、プロセスのエラーまたは不正動作、他のコントローラから受信した潜在的に競合するコマンド、コントロール対象のプロセスへ適用された、または受信された前のコントロールアクション、プロセスによって使われる誤った優先順位スキーム、誤ったコンフィグレーション、プロセスや環境への経年劣化や変化、プロセス環境における予期しないまたは処理できない条件、または他の問題などといったものです。

これらのシナリオを作成するために、コントロールアクションを選択し、どの要因がコントロール対象のプロセスにコントロールアクションを無効にするような影響を与えるかを識別します。

例：

コントロールアクション：BSCU がブレーキコマンドを送信する

シナリオ 6：BSCU がブレーキコマンドを送信するが、ホイールブレーキシステムは、前もって（BSCU をバイパスするような）代替のブレーキングモードへのコマンドを指示されたため、ブレーキがかからない。その結果、着陸時に与えられた減速が不十分である可能性がある。 [H-4.1]

シナリオ 7：BSCU がブレーキコマンドを送信するが、不十分な油圧（ポンプ故障、油圧リークなど）によりブレーキがかからない。その結果、着陸時に与えられた減速が不十分である可能性がある。[H-4.1]

シナリオ 8：BSCU がブレーキコマンドを送信し、ブレーキがかけられるが、滑走路が濡れているため（ホイールのハイドロプレーン）に、航空機は減速しない。その結果、着陸時に与えられた減速が不十分である可能性がある。 [H-4.1]

セキュリティの問題を入れるには、同じ追加の可能性を再度考慮する必要があります。それは、攻撃者がどうやって同じ問題を引き起こすコントロールアクションに相互作用するのかを識別することです。例えば：

シナリオ 9：BSCU はブレーキコマンドを送信するが攻撃者がホイールブレーキングシステムを代替のブレーキングモードに入れるようなコマンドを入れ込んだため、ブレーキがかけられない。その結果、着陸時に与えられた減速が不十分である可能性がある[H-4.1]

共通のミスを防止するためのヒント

共通のミスの多くは、シナリオよりもむしろ、個々の因果関係要因を識別することにあります。例えば、「ホイール速度センサの故障」、「ホイールの速度フィードバックが遅れる」、「電源喪失」のような要因のリストを作成したくなるかもしれません。シナリオのコンテキストを超えて個々の要素をリストにすることの問題は、複数個の要因がどのように相互作用を及ぼすのかを簡単に見落としてしまうことにあります。また、間接的に UCA やハザードにつながる、つまらないことではないが明白にわからない要因を見落とすことになりますし、これらの要因の組み合わせがハザードに至ることを考えなくなるでしょう。単一の要因群を考慮することは、本質的に、コンポーネントの単一故障だけを考慮するという FMEA になってしまいます。

シナリオを識別するための入力と出力

図 2.20 に、シナリオの識別ステップのための入力および出力をまとめます。

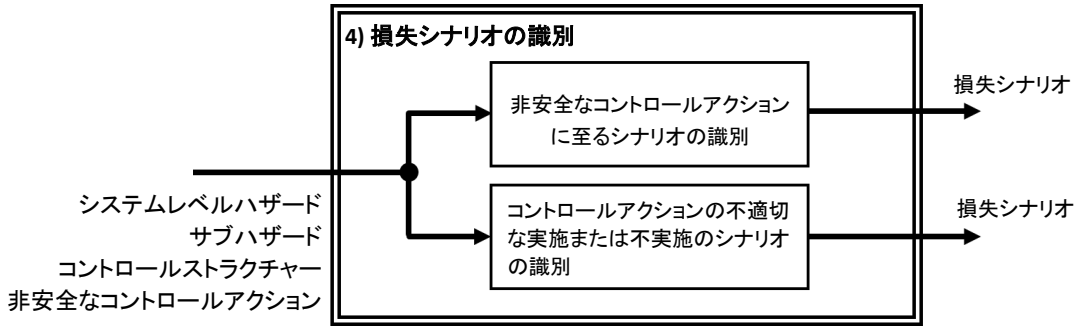


図 2.20：シナリオの識別の概要

STPA 出力およびトレーサビリティ

図 2.21 に、各種の STPA 出力間で保たれるトレーサビリティを示します。

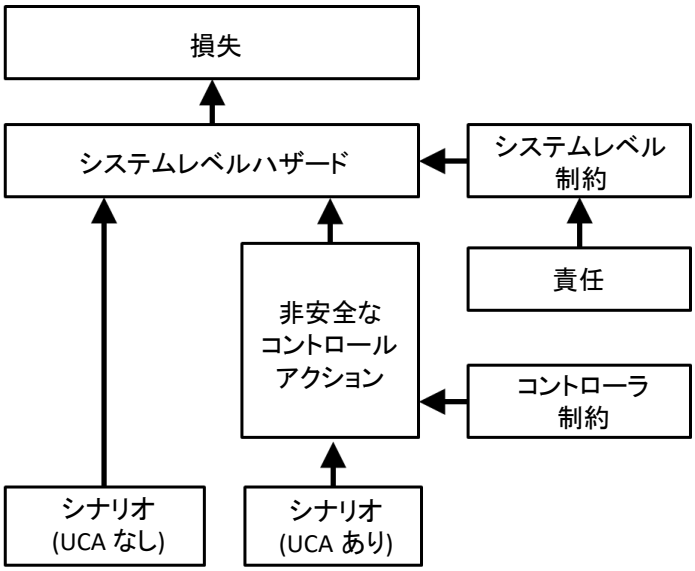


図 2.21 ：STPA 出力間のトレーサビリティ

コントロールストラクチャーは、すべての STPA 出力に密接に関連しており、したがって簡単にするため、*図 2.21* に明確には示されていません。 これらの STPA 出力は、以下を含めて多くの異なる方法で使用することができます：

- システムアーキテクチャを進める
- 実行可能な要求を作成する
- デザインの推奨事項を識別する
- 必要な緩和策やセーフガードを識別する
- テストケースを定義し、テスト計画を作成する
- 新しいデザインの決定(STPA が開発中に使用されている場合)を進める
- 既存の設計上の決定を評価して(デザインが完成した後に STPA が使用されている場合)、ギャップと必要な変更を識別する
- リスクの先行指標を作る
- より効果的な安全マネージメントシステムを設計する
- など

ここまでで、STPA がどのように実施されるか、基本的な理解をされたはずです。あなた自身のエンジニアリング問題のいくつかに分析を試みることをお勧めします。付録には、役に立ちそうな追加の例をいくつか入れてあります。

まとめと考察

この章では、STPA で使用される基本的な考え方を説明しました。次の章では、STPA をシステムエンジニアリングに関わる様々なフェーズや活動に適用する方法について説明します。第 4 章では、職場の安全に STPA を使用するための追加情報と例を提供します。この時点まで、すべての例には、技術的なシステムを必要としていますが、STPA は、どのような社会技術システムにも使用することができます。第 5 章では、安全性以外にも、組織の分析や、創発的、システム全体の特性について STPA を使用する方法を示します。第 6 章では、STPA は、リスクの先行指標を識別するために使用することのできる方法を示します。第 7 章では、より効果的な安全管理システムを設計する STAMP と STPA の使い方を説明します。最後に、第 8 章では、これまでに大規模な組織やプロジェクトに STPA を組み入れる方法について学んできたことを説明します。

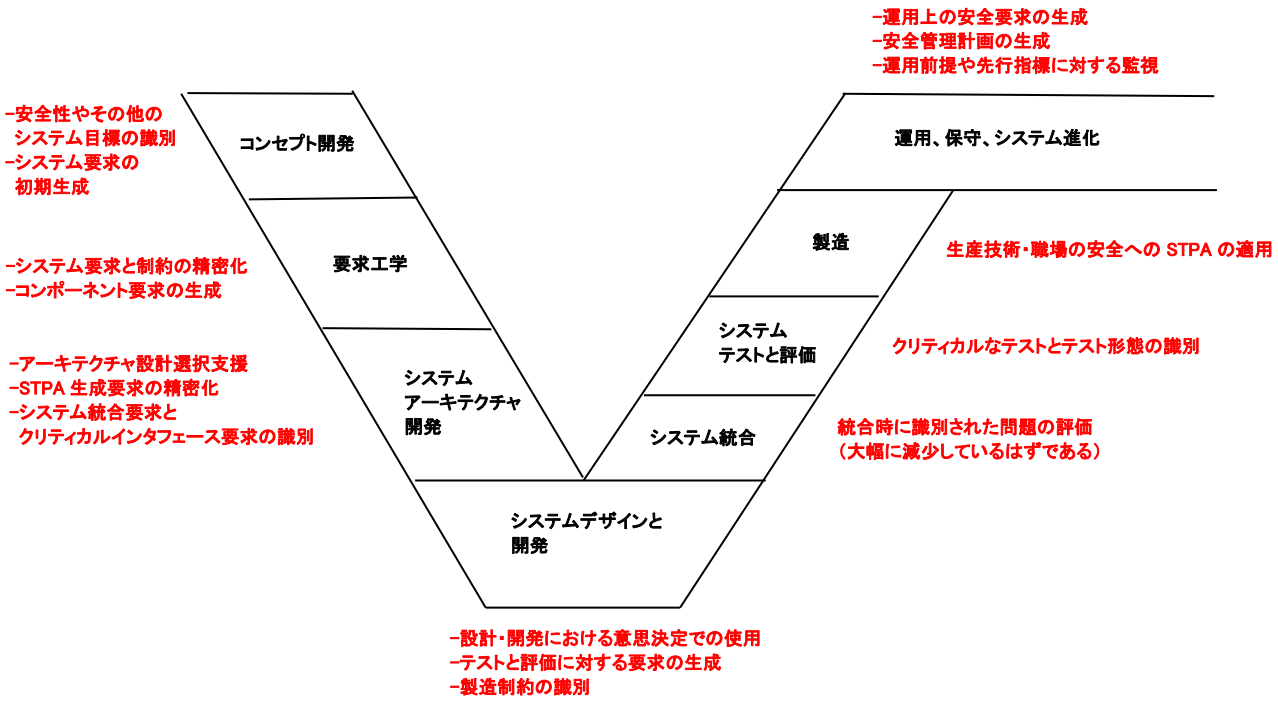
第 3 章：システムエンジニアリングプロセスへの STPA の統合

ナンシー・レブソン

システム安全は、多くの場合、システムエンジニアリングプロセスから独立または分離して扱われます。最もよくあるのが、安全が事後の保証活動として扱われてしまうことです。安全をシステム内で保証することはできませんが、安全を考慮して設計しなければいけませんので、事後で安全を扱ってしまうと、安全に関する設計上の欠陥は、いつも後工程で発見され、是正することができないこととなります。その結果として、欠陥を是正する必要がないことの論拠を見つけ出すことに労力が注がれ、さらにその論拠が破綻し始めると、システムに冗長性を持たせる、あるいはオペレータに好ましくない運用手順で問題の検知と回避させるような、高価で非効果的な解決策へ労力が注がれることとなります。

本章では、システム全体のエンジニアリングプロセスにシステム安全性解析を密に統合する方法について述べます（システムエンジニアリングの基礎は付録 F を参照）。その統合をすることができれば、大きな効果をもたらすだけでなく、安全に関するエンジニアリングコストは大幅に削減でき、うまくいけば無駄のないものとなります。また手戻りが減り、コスト低減や期間短縮へもつながるでしょう。

以下に示す図は、標準的なシステムエンジニアリングプロセスの V 字モデル（またはウォーターフォールモデル）を簡略化したもので、フィードバックループは簡素化のために省略しています。本図は、STPA を標準的なシステムエンジニアリングプロセスにどのように統合するかを示し、その中で STPA の役割候補を赤で示しています。様々な V 字モデルや他の開発プロセスモデルがあるかもしれませんが、この標準的な V 字モデルから置きかえて解釈することは難しいことではないでしょう。ただし最上流（V 字 左上部の 2 つの活動）が省略されているようなプロセスはその限りではなく、そのようなプロセスでセーフティクリティカルなシステムを構築すべきではありません。



STPA は、最も初期のコンセプト開発段階から標準的なシステムエンジニアリングプロセス全体を通して使用することができます。基本的に STPA はコンセプト開発の早い段階で上位の安全要求を生成し、システム要求分析段階でそれらを改良化することに使用することができます。そのシステム要求と制約は、システムアーキテクチャ設計や詳細なシステム設計に役立ちます。そして STPA は、設計・開発プロセスの中で意思決定が行われる流れに沿って解析結果が使われる形で一体となって進められていくこととなり、一連の保証活動や製造プロセスを通して役立ち、システムの運用に対しても重要な情報を提供していくこととなります。

STPA のモデルは昨今のモデルベースシステムエンジニアリングにおいて提案されることの多いアーキテクチャモデルと異なりますが、STPA はそれらのシステムモデル（設計意思決定に沿って精密化される）上でも機能し、モデルベースシステムエンジニアリングプロセスにも適用できます。最小限の再解析で意思決定や設計を変更することができるようにするため、STPA では開発プロセス全体のトレーサビリティがとられていることを奨励しています。

最後に、本書を通して述べているように、STPA は、単に安全だけでなく、システムエンジニアリングや製品ライフサイクルにおけるいかなる創発特性に対しても適用することができます。

全体プロセス

STAMP と STPA は、システムエンジニアリングにおける全ての活動に寄与し、その解析や解析結果は以下のとおり使用します。

1. 開発・運用における損失の定義
2. システム設計における外部制約の識別（市場や規制からの要求を含む）
3. システムレベルのハザード識別と、システム動作に関する関連要求や制約の識別
4. システムコントロールストラクチャのモデル化
5. ハザードや制約の精密化とシステムコンポーネントへの機能割り当て
6. アーキテクチャ、設計、実装上の意思決定支援
7. システム統合支援
8. システムテストに対する要求生成
9. 製造管理（生産技術、職場の安全）
10. 運用上の安全要求（リスクを高める先行指標を含む）の生成と安全管理計画の生成
11. 先行指標監視を含む運用上の安全管理

1. 考慮すべき損失に関する意思決定

初期のコンセプト開発プロセスは V 字モデルによって異なりますが、この段階では通常、利害関係者や利用者の分析（ニーズ分析）、顧客要求の生成、規制要求審査、実現性検討、トレードオフ検討、および設計を進め決定していくための評価基準の確立のような活動を実施します。また最終的には運用コンセプトも検討するかもしれません。

多くの場合、その初期段階に必要な配慮を欠いたり、労力が割かれず、すぐにアーキテクチャ仕様や上位レベル設計の検討へ入ってしまいがちです。コンセプト開発が不十分であると、どのように開発をしたとしても、顧客にとって使い物にならなかったり、部分的にしか利害関係者のニーズを満たせなかったり、あるいは保証、保守、運用も難しいシステムへつながる可能性があります。その初期段階の検討不足の埋め合わせを行うために後工程で変更を行うかもしれませんが、後工程での変更は、開発が進むほどより高価なものとなり、開発を崩壊させていくものとなります。

特に安全性とセキュリティについては初期のコンセプト検討において十分検討されていないことがよくあります。*図 3.1* にこれらの創発特性の典型的な扱われ方を示します [Young 2017]。多くの場合は、運用中に損失が発生した後の対応に重きが置かれます。加えて、設計やシステムエンジニア

リングの大半が既に行われた後の追加（例えば、保護システムや侵入者検知の追加）を行うことにも注力されています。このような後工程における変更は、最初からシステムに安全性とセキュリティを作り込んでいた場合に比べ、より高価なものとなるばかりでなく、通常あまり効果的なものとはなりません。

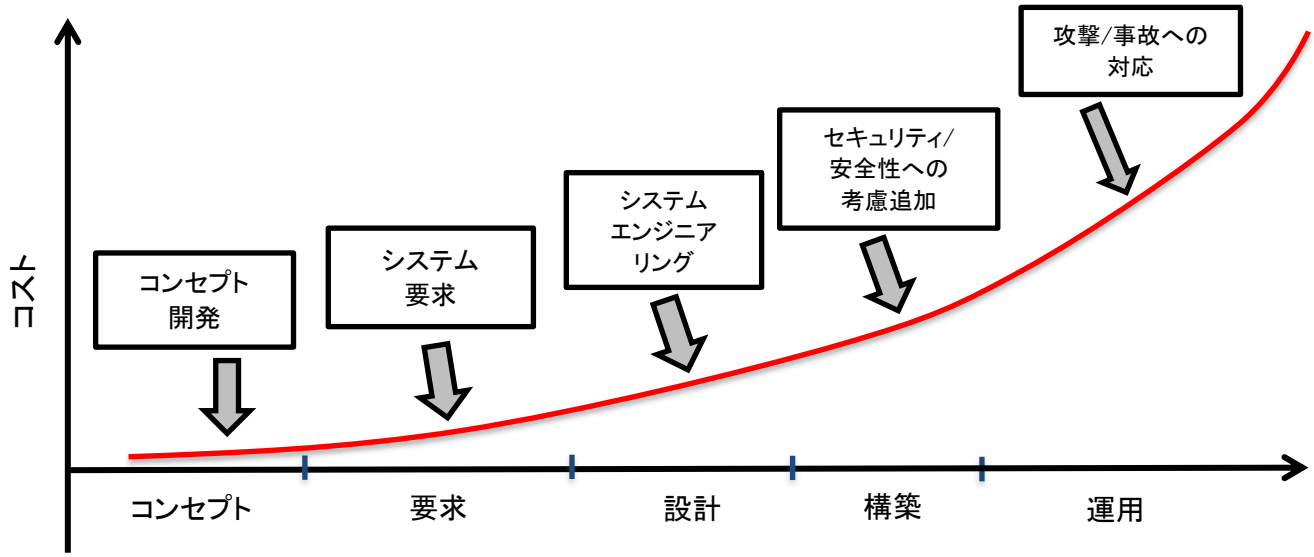


図 3.1: 安全とセキュリティを開発初期からシステムに作りこむ必要性[11]

安全とセキュリティへは後工程で検討するのではなくコンセプト開発段階で検討を開始すべきで、その検討結果はシステム全体やコンポーネントに対する安全とセキュリティに関する要求を作るために使用されるべきです。Frola と Miller（1984）[12]は、安全に関連する設計の 70～90%はコンセプト開発段階で決まり、それ以降の変更は不可能であるか非常に高価なものとなると主張しています。またセキュリティにおいても同じことが言えます。

多くのシステムエンジニアリングプロセスでは、安全性はその初期段階で検討されることとなっていますが、通常予備ハザード解析（Preliminary Hazard Analysis：PHA）の形態をとっています。予備ハザード解析では、識別したハザードに対してどれぐらいの労力を開発や運用において割くべきかや、ハザードをどのように除去または制御すべきかの予備的な提案を決定するようなリスクアセスメントが行われます。

[11] William Young, *A System-Theoretic Security Analysis Methodology for Assuring Complex Operations Against Cyber Disruptions*, Ph.D. dissertation, MIT, 2017.

[12] F. Ronald Frola and C.O. Miller, System Safety in Aircraft Acquisition, Technical Report, Logistics Management Institute, Washington D.C., January 1984.

Table 2.1. 典型的な PHA フォーマット[13]

プログラム：＿＿＿＿＿＿ エンジニア：＿＿＿＿＿＿				日付：＿＿＿＿＿＿ ページ：＿＿＿＿＿＿		
項目	ハザード条件	要因	影響	リスク評価符号	評価	提案
割り当て番号	条件の特質を記入	何が条件を発現させるかを記述	是正しない場合何に影響するか、ハザード条件の影響	ハザードレベルの割り当て	発生確率、発生可能性： － 見込み － 度合い － 大きさ	ハザードを除去または制御するための活動提案

[Vincoli, 2005]

表中の情報の多くは初期のコンセプト開発段階で利用することができますが、詳細設計を行う前にハザード条件の発生確率(probability)やその起こりやすさ(likelihood)を知ることはできません。あるいはそのシステムがソフトウエア集約型であった場合には詳細設計後であったとしてもそれらを知ることはできません。また過去のシステムと大きく異なるシステムを開発する場合には過去の履歴情報も使うこともできません。完全に合理的な因果シナリオを有する実際のプロジェクトにおいても、多くの場合、あるハザードが誤ってリスクが低いまたは極めて低いと初期段階で見逃されていることがこれまで STPA で見つかっています。心理学研究においては、まれな事象の発生確率やその起こりやすさに対する人間の推定精度は非常に悪いということが示されています。[14]

この種の PHA における最大の問題は、初期のコンセプト分析における主目的の一つである機能的な安全性とセキュリティに関する要求を識別するために必要な情報を提供しない点となります。従来のハザード解析技術は理論的にはその目的のために使われることとなっているかもしれませんが、かなり詳細な設計情報が必要となるため後工程にならなければ適用することができず、設計作業に機能安全要求を提供するには遅すぎることとなります。

その代替手段として航空分野でよく用いられているのは、システムが設計として充足しなければならない確率的要求です[SAE ARP 4761]。例えば「着陸中または離陸中断時における全ての車輪制動の喪失確率は 1 フライト当たり 5E-7 未満でなければならない」や「離陸時において 1 つの車輪をロックできない不慮の検出されない車輪制動の発生確率は 1 フライト当たり 5E-9 未満でなければならない」などです[SAE ARP 4761]。ブレーキシステムのほとんどが故障確率が既知のハードウエア部品だけで構成されていた時代にはこのような要求は合理的だったかもしれませんが、ほとんど全てのシステムにおいてソフトウエアが広範囲に使用されている昨今のソフトウエア集約型システムにおいては、これらの要求の充足を保証することはできません。

利害関係者や利用者のニーズ（ニーズ分析）を行い、顧客視点目標・要求の生成、規制要求審査、実現性評価、概念の探索やトレードオフ検討、および設計を進め決定していくための評価基準の確立のような活動を実施するコンセプト開発段階において STAMP と STPA は必要とされる情報を創り出すことができます。失敗または問題のある多くのシステムエンジニアリング活動における主要な問題は、利害関係者のニーズと適切なシステム目標を限定的にしか理解できていないことから生じます。

2. システム設計における外部制約の識別（市場や規制からの要求を含む）

[13] Jeffrey Vincoli, Basic Guide to System Safety, John Wiley & Sons, 2005.

[14] the work of Tversky and Kahneman on heuristic biases を例として参照

社会システムの安全に関するコントロールストラクチャーは、システムにおける外的な（環境的な）社会制約を理解するために使用することができます。規制が設けられている多くの産業に共通的に同様である社会技術のコントロールストラクチャーの例を*図 3.2* に示します。左側には開発のためのコントロールストラクチャー、右側には運用のためのコントロールストラクチャーを示しています。おそらく各産業それぞれ独自の社会技術のコントロールストラクチャーがあり、国によっても異なるかもしれません。あくまで例として国家構造がここに示されていますが、いくつかの産業（例えば、航空や原子力）においては、上層に国際的なコントロールコンポーネントもあります。

一度このような構造をモデル化すれば、各開発プロジェクトの間で多少の差はあるかもしれませんが、おそらく各開発プロジェクトのためにほとんど変更の必要もなく適用できるでしょう。この構造は、開発中のシステムのための規制やその他の外部要求を識別し、文書化するために使用することができます。

3. 考慮すべき損失の識別

非常に初期のコンセプト開発段階では、利害関係者は、許容できない損失を特定しなければいけません。多くのハザード解析やシステム安全に関する技術は、人の死亡、負傷や資産の損害だけを扱いますが、STPA は環境汚染、ミッション損失、経済的損失、さらには企業の評判に関する被害を含め、あらゆる損失を扱うことができます。言い換えれば、損失とは、利害関係者が避けたい任意の創発特性を意味します。また正式には、事故（軍事用語では mishap）とは、利害関係者が損失であると考えることをもたらす任意の望ましくない、または計画外の事象または状態を意味します。本書では、航空機の例を用いて、以下の事故（損失）から説明します：

A1. 乗客や航空機に関わる人の死亡または重傷

A2. 航空機や航空機外部の物に対する許容できない損傷

何が許容できないかは利害関係者によって定義される必要があります。利害関係者が損失を回避するために労力を割く価値があると考えている次のような新たな損失が利害関係者によって特定されるかもしれません。

A3: 運航遅延から生じる経済的損失

A4: 航空機や航空会社の評判に関する被害による売上減

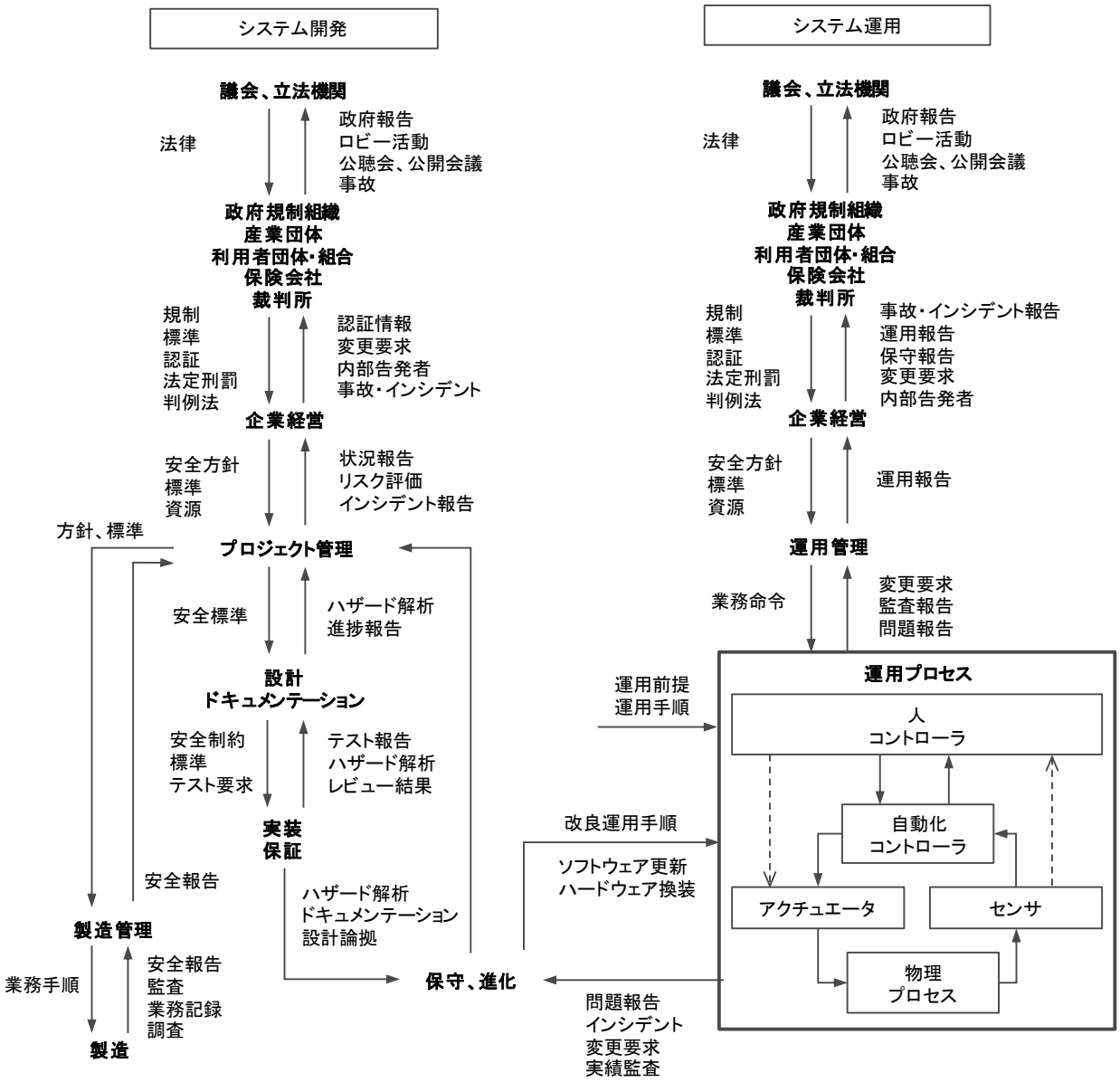


図 3.2:安全性に関するコントロールストラクチャーの例

4. システムレベルのハザード識別と、システム動作に関する関連制約の識別

許容できない損失が利害関係者によって識別されると、エンジニアはその損失に関連付けてハザードを定義することができます。その最終的な目標は、ハザードを除去するか、除去することができない場合は、システム設計の中で識別されたハザードを制御するか減らすこととなります

まず最初の問題は、どうやってハザードを定義するかです。ハザードという用語は時折、航空機における悪天候など、システム境界外のものを参照するために使用されていますが、STPA におけるハザードは、システム設計者が制御できるシステム状態に限定されます。例えば、ハザードは悪天候ではなく、むしろ悪天候による悪影響を受ける航空機となります。制約やコントロールによって、天候から影響を受けないようにしたり、天候からの影響に耐えることができる航空機を設計することなどができるかもしれません。システムエンジニアリングは、システムの設計を考えるもの

ですのでシステムそのものの動作だけに影響を与えることができ、システム境界外の物（つまりシステムの運用環境）を変えることはできません。

また事故や事故の前提条件につながる可能性がある条件がハザードとして定義される時があります。しかし、そのような定義は、そのシステムが全体で動作するようなものであればほとんど全てのハザードを含んでしまうため、除去や制御をすることができません。例えば、着陸しようとしている航空機は、その後に事故に至る可能性がありますが、決して着陸しない航空機を作ることはできません。別の例としては、着陸中の減速（安全な運用として必要なもの）はハザードではなく、着陸後に減速できないことがハザードとなります。

よって実用的な理由からハザードはシステム設計者が決して発生させたくない事故につながる前兆状態として定義されていますので、設計プロセスで除去または制御すべきものとなります。正式な定義は前章で述べています。

システム設計においてハザードを除去するために、ハザードはシステム開発プロセスの最初に識別される必要があることは明らかです。特定されたハザードは、ソフトウエアやオペレータを含むさまざまなシステムコンポーネントに対しての挙動に関する（機能的だが確率的ではない）安全要求を作成するために使用されることとなります。

前述の事故 A1 と A2 に関しては以下のハザードが含まれます：

H1: コントロールされた飛行を維持するのに不十分な推力[A1、A2]

H2: 機体の完全性の喪失[A1、A2]

H3: 航空機と固定・移動物体との間の最小間隔違反[A1、A2]

H4: 地上における航空機の移動・静止物体への過接近または誘導路からの不慮の離脱[A1、A2]

H5: 運航予定時刻に離陸できない航空機[A3、A4]

H6:その他

これらのハザードは、次のような上位レベルのシステム制約[15]につながります：

SC1: コントロールされた飛行を維持するのに十分な推力が利用可能でなければならない[H1]

SC2: 機体の完全性は、最悪の条件下で維持されなければならない[H2]

SC3: 航空機は固定・移動物体との間の最小間隔基準を満たさなければならない[H3]

SC4: 地上において航空機は移動・静止物体との間で安全な距離を常に保ち、誘導路などの安全領域に留まらなければならない

SC5: 航空機は出発予定時刻のTBD分以内に離陸することができなければならない[H6]

SC6: その他

5. 機能的なシステムコントロールストラクチャのモデル化

前述した最初のコントロールストラクチャーではシステムが動作する環境（システム境界の外側）を含むことができましたが、本ステップにおけるシステムコントロールストラクチャは、システム設計空間内に存在するものについてのさらなる詳細を提供します。 モデル化する範囲（考慮すべきシステム境界の定義を意味する）は、システムエンジニアリングプロセスの目的に依存します。概念設計が進むにつれて詳細情報は増加しますので、それに応じて複数の異なるモデルを構築

[15] 否定的な要求は受け入れられない場合が多いですが、重要な情報を損なうことなく安全制約を肯定的表現に変更することができないことがあるため、ここでは要求と制約が区別されています。 制約は「shall」ではなく「must」または「must not」を使用して記述されます。この記述は要求（システム目標またはミッション）とそれらの目標の達成方法に関する制約が区別できるという利点があり、トレードオフの決定を行う必要がある場合に役に立ちます。

することはよいかもしれません。またコントロールストラクチャーモデルの構築に関する詳細は前章で述べていますが、ここでは本章が独立するよう繰り返しを含めて述べます。

図 3.2 のコントロールストラクチャーであれば、設計されるべきシステムは、図中右下で「運用プロセス」と示されたボックス内における機能的なシステムコントロールストラクチャかもしれません。通常、最初にとても上位のシステム要求や制約を識別するため、STPA はこのシステムコントロールストラクチャ上で行われますが、コンセプト設計プロセスにおいて、コントロールストラクチャーは精密化され、STPA は、関連付けられたシステム要求と制約を改良化することに使用されることとなります。

例えば、図 3.3 はパイロット、自動制御システム（おそらく複数のコンピュータで構成されますが、後の設計段階で決定されます）、および物理航空機部品のわずか 3 つのコンポーネントで構成される、航空機のとても上位の機能的コントロールモデルを示しています。航空機のような複雑なシステムにおいては、現在検討しているコントロールストラクチャー部位に応じ、抽象化レベルを変えることで、その部分に着目することができます。このようなトップダウン方向の精密化は、航空機の全体的な動作の理解や、コンポーネント間の相互作用を識別するのにも役に立ちます。

このモデルには、システムは航空機であろうこと、またその航空機は一般的な航空機の機能・設計が含まれているという事実以上の設計上の意思決定は含んでいないことに注意してください。したがって、このモデルは初期のコンセプト開発段階での使用に適しています。システムエンジニアリング上、この段階で、モデルのいずれかの部分が特定のアプリケーションのために詳細化されすぎている場合、それらを除去することができます。また注意いただきたいのは、コンセプト開発のこの段階ではパイロットは人（搭乗している人か地上の人）かもしれませんし、自動化（無人航空機）されるかもしれませんし、航空機も固定翼機、垂直上昇機、あるいは垂直離着陸機であってもよいことで、区別は必要ありません。この普遍性を持つことによって、より多くの情報が入手できるまでの意思決定の先送りとモデルと分析の再利用が可能となります。昨今設計されている航空機システムでは、操縦モードとして、これらの一部または全部を含むかもしれません。

パイロットの役割は、図 3.3 のコントロールストラクチャーに示されるように、航空機の自動化を管理し、航空機の設計に応じて、直接または間接的に離陸をコントロールし、飛行、着陸、地上で航空機を操縦することです。パイロットと自動化コントローラには、それらがコントロールしているシステムまたはサブシステムのモデルを含んでいます。自動化が航空機をコントロールしていますので、自動化は現在の航空機の状態のモデルを含んでいなければなりません。パイロットもまた航空機の状態のモデルを必要とし、それに加えて、パイロットは、自動化の状態と、それらが動作している空港や空域のモデルを必要としています。多くのパイロットの過失は、自動化がどう動くかの理解や自動化の現在の状態理解の不足に原因があります。

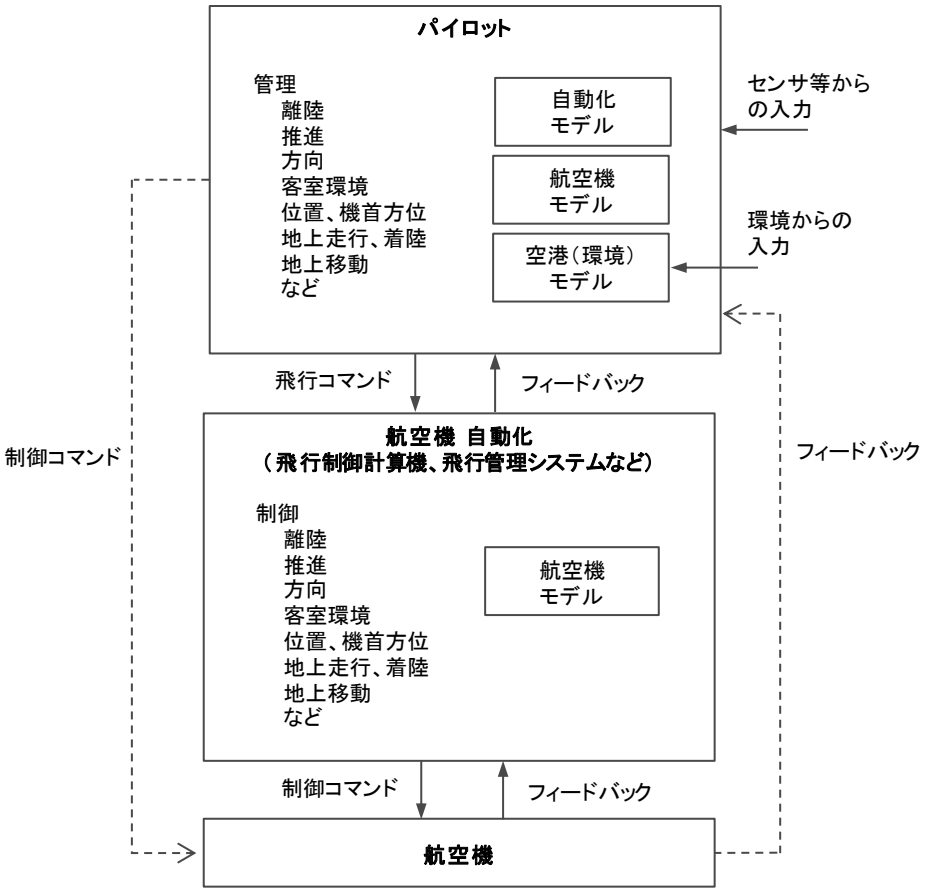


図 3.3: 航空機レベルでの上位のコントロールストラクチャー

パイロットは、自動化へ飛行コマンドを供し、自動化から自動化や航空機の状態に関するフィードバックを受けます。設計によっては、パイロットは航空機のハードウエアに直接コントロール動作を提供（つまり自動化されたシステムを介さない）と直接フィードバックを受けることができます。図中の点線は、この直接コントロールとフィードバックを表しています。設計が精密化され、より詳細な設計上の決定が行われると、これらの点線は除去されるか、特定の内容で具体化されるかもしれません。パイロットは様々な知覚手段を通した航空機の状態や環境 に関する直接の知覚フィードバックを常に持っています（無人航空機システムのようにパイロットが地上にいる場合を除く）。同時に、パイロットは、多くの場合、自動化と自動制御装置がコントロールしている物理的なコンポーネントの状態に関して、直接的に全てを知覚することはできませんので、パイロットに利用可能な情報には、エンジニアがパイロットが知っておくことが重要と判断する情報に制限されます。 STPA は、この重要な情報が何であるかを識別するのに役立ちます。昨今の高度な飛行制御においては、パイロットは全ての情報を必要としていないという仮定をおいているか、他の理由による省略により、パイロットへ全くフィードバックを提供していない状態や入力が含まれています。

STPA では、このコントロールストラクチャーを使用しながら、システムのハザードから作られた非常に大まかなシステムレベルの安全要求と制約（例は前述）を精密化していきます。これらの新たに精密化された安全要求によって、上位のコントロールストラクチャーにおける 3 つのコンポーネントそれぞれに対して、航空機の安全な運航を維持するために必要となるプロセスモデルの内容やフィードバック要求だけでなく、責任、権限、および説明責任が特定されることとなります。例えば、パイロットは各段階で安全な飛行を確保するのに十分な推力を指令すること[SC1]、あるいはパイロットは航空機を地上で移動させるための安全なコマンドを与えられなければならない[SC4]

というような要求です。システムレベルの安全要求は、この段階から、安全に関するコントロールストラクチャー内の個々のコンポーネントに割り当てが始まることに注意してください。

6. ハザードや安全制約の精密化とシステムコンポーネントへの割り当て

コンセプト設計プロセスの進行に応じ、要求が精密化されていくようにコントロールストラクチャも精密化されていきます。この精密化は、その時点での設計詳細度に応じた非安全なコントロールアクションと因果シナリオの識別により行います。その結果は、その後の開発プロセスで精密化し使用することができる安全関連の一連の要求となります。

システム要求と安全要求は、それらを満たすシステムアーキテクチャの設計よりも先に分析しなければいけません。もちろん安全が、システムの唯一の目標ではなく、他の要求もアーキテクチャの設計において考慮される必要があります。しかしながら、我々の経験では、多くの場合、安全要求（大抵はシステム要求）が特定される前にアーキテクチャが設計されています。この順序の逆転は、アーキテクチャがシステム要求や安全性要求に対して最適にならないばかりか時にシステムの目的に対してもアーキテクチャが最適にならないことが起こる可能性を高めてしまいます。

アーキテクチャの設計にはシステムコンポーネントへのシステム要求の割り当てが含まれます。このプロセスにおいて、STPA では、システムレベルで作られた非安全なコントロールアクションに対する因果関係シナリオの生成を行います。この精密化プロセスの例として減速を使用します。関連する上位のシステムハザードは以下の通りです：

H4: 地上における航空機の移動・静止物体への過接近または誘導路からの不慮の離脱[A1、A2]

航空機が地上またはその近くで動作している場合にH4に関連する特定の事故が起こり、結果として飛行機が滑走路から外れてしまったり、飛行機を滑走路上かその近くの物体へ衝突させてしまうようなことを引き起こすかもしれません。このような事故においては、許容できない損害や怪我や人命損失が発生する速度でフェンス、他の飛行機、滑走路上やその端におかれている物体へ衝突してしまうことも起こりえます。

H4は次の通り、減速に関するハザードへ精密化することができます：

H4-1: 着陸時、離陸中断時、または地上走行時における不適切な航空機の減速

H4-2: 離陸時V1[16]ポイント後の減速

H4-3: 駐機中の航空機の作動

H4-4: 意図しない航空機の方向制御（差動制動）

H4-5: 安全域（誘導路、滑走路、ターミナルゲート、ランプなど）外での航空機操縦

H4-6: 着陸装置格納時（または格納後）における主たる車輪の回転非停止

これらのハザードに関する上位のシステム安全性制約は設計上の制約として、これらのハザードを簡易に修正し再記述することで定義できます。

SC1: 着陸時、離陸中断時、または地上走行時において、前進動作は制動コマンドよりもTBD秒以内遅延しなければならない（H4-1）

SC2: SC2:航空機はパイロットの直接操作なくV1後に減速してはいけない（H4-2）

SC3: 駐機中はコマンドなしで動作してはいけない（H4-3）.

SC4: 差動制動は航空機の方向制御の損失や意図しない制御へ至ってはならない（H4-4）

---

[16] V1 ポイントは、離陸を継続するよりも離陸を中断することの方が危険な離陸シーケンスのポイントです。 ただし非常にまれには、離陸を継続するより V1 ポイント後に減速する方が安全ですが、分析結果を数ページに収めるため、そのような状況（およびその他）は、本例の分析では考慮いたしません。

SC5: 航空機は安全域（誘導路、滑走路、ターミナルゲート、ランプなど）外で意図せず操縦されてはならない（H4-5）

SC6: 車輪が格納されている際には、主たる車輪の回転は停止しなければならない（H4-6）

これら全て非常に上位の安全要求であることに注意してください。つまり安全要求は航空機全体としての考慮から始まり、前章で述べた通り、これらを、乗組員やソフトウエアやコンポーネントの相互作用を含む特定のシステムコンポーネントに関連付けながら、より詳細な機能的な安全要求へ反復しながら精密化していきます。システム安全要求はSTPAで識別した因果関係シナリオを防ぐために生成され、このプロセスは全てが設計の中で適切に分析し扱われるようになるまで継続します。重要なことは、システム全体としての動作が最初に考慮されているため、後からコンポーネント間の全ての相互作用を考慮して、それらがハザードであるかどうかを判断しなくても潜在的に非安全な相互作用を識別できることです。

図3.4は、減速が発生する地上でのコントロール機能に着目したコントロールストラクチャーモデルです。本モデルでは、逆推進スラスタ、スポイラ、車輪制動装置の3つの基本的な物理的なコンポーネントがあります。この1つのコントロールストラクチャーだけでなく、より大きな機能的なコントロールストラクチャーモデルを再度分析に含めることによって、STPAは、分析対象のハザードに関連する制動コンポーネント間のすべての相互作用（意図したものと意図しないものの両方）を考慮することができます。ここで制動装置は、逆推進スラスタ、スポイラ、車輪制動装置で構成することが決定されているようにいくつかの設計上の決定事項がこの段階で発生しています。しかしながらまだ設計詳細度はとても一般的かつ上位レベルのもので多くの航空機において適用可能なレベルとなっています。

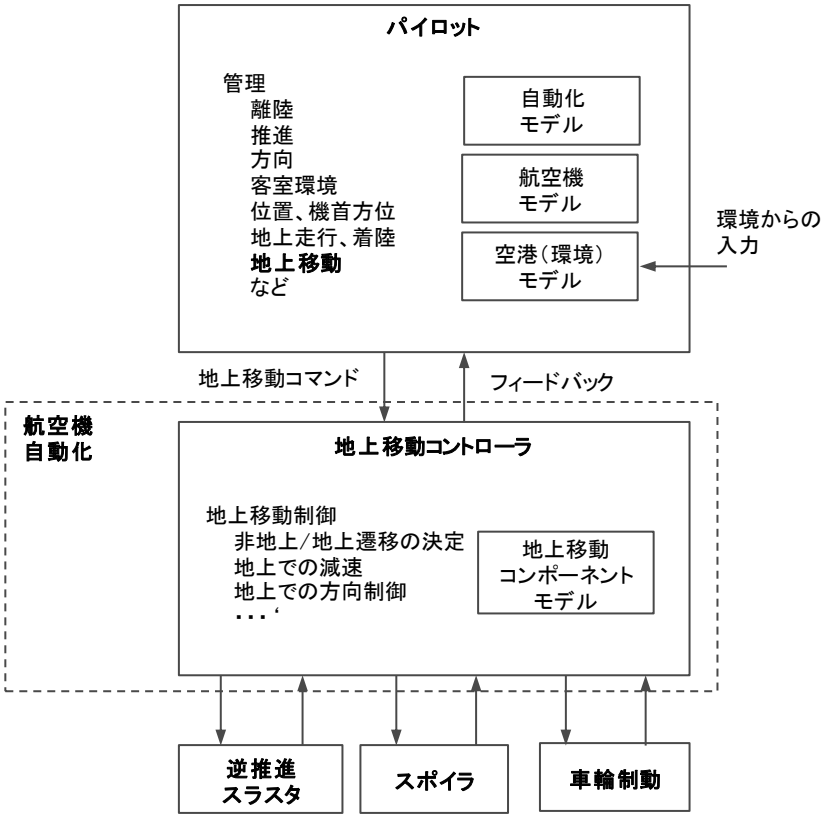


図3.4:減速コントロールストラクチャー

開発のこの時点において、STPA によって生成することができるブレーキシステムのコンポーネント間の相互作用の安全制約の例は以下の通りです：

SC-BS-1: 速度が TBD を超えた状態でホイールブレーキを手動または自動で駆動させる場合には、スポイラを展開しなければならない

SC-BS-2: ホイールブレーキは、着陸装置格納時に駆動しなければならない

SC-BS-3: 地上でのスポイラの駆動においては、作動可能な状態となっている自動ブレーキシステムを駆動しなければならない

SC-BS-4: 前方推進スラスタの稼働時に自動ブレーキシステムはホイールブレーキを駆動してはいけない

SC-BS-5: 前方推進スラスタの稼働時に自動スポイラ装置はスポイラを格納しなければならない

STPA は、オペレータ、保守者、さらには経営者のような人間を含んで分析するため、STPA はシステム設計の影響によって発生する、モード混乱や状況認識喪失のような人為的な過失を特定することができる構造化された方法であることに注意してください。このような人為的な過失は、実際の自動化状態とオペレータが認知している状態の間での不整合（同期の欠如）が起こることによって発生し得ます。

ホイールブレーキシステムの機能的な構造に着目した減速コントロールストラクチャーを図 3.5 に示します。さらに図 3.5 においては、ブレーキシステムのうち 1 つのコンポーネントだけが考慮されているが、物理設計・実装についての仮定は存在しません。 STPA は、潜在的な問題への特定の設計解なしで始めることができます。その代わりに、必要な基本機能の動作から分析が始まり、その動作が危険となり得る条件を特定していきます。設計者は、この分析によって得られた安全要求を満たすのに必要な冗長性や機能の再設計などの特定の設計解に基づきこの後工程で意思決定をすることができます。

図 3.5 は、現代の航空機で一般的に採用されている自動ブレーキシステムを含める、という一つの重要な設計上の決定を含んでいます。一貫性あるブレーキが行われることを確認するためパイロットは着陸前に自動ブレーキを事前に設定することができます。自動ブレーキは、突風や氷のような条件下でのパイロットの作業負荷を軽減し、大幅にブレーキ摩耗を低減することができます。自動ブレーキを設定した場合、自動ブレーキは、指定された時間にホイールブレーキを行うこととなります。

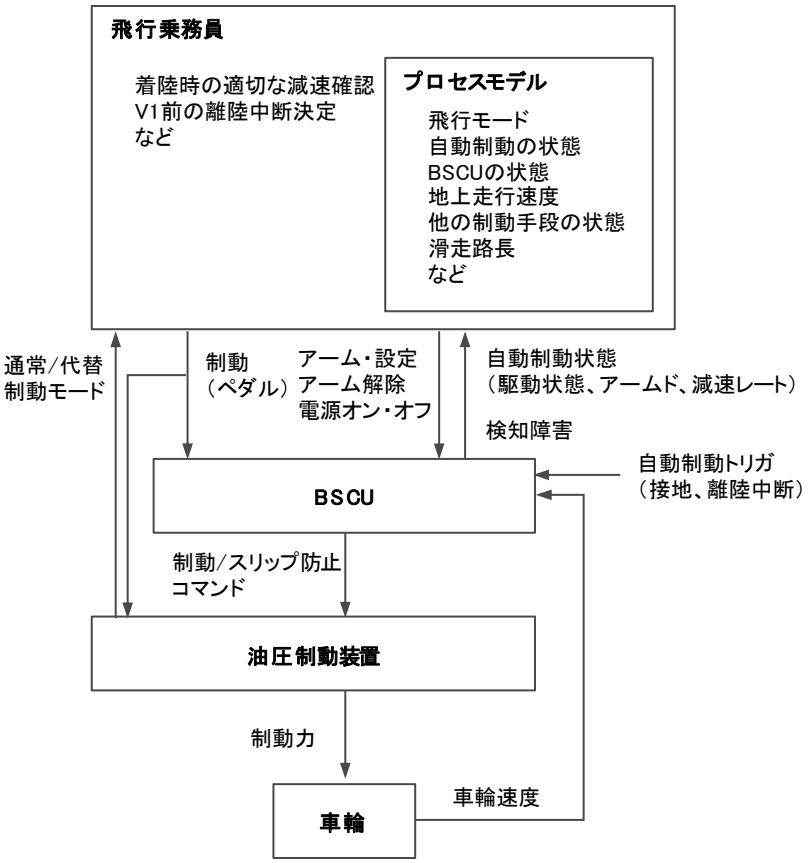


図 3.5: ホイールブレーキシステムのコントロールストラクチャー

表 3.1 の乗務員（FC）とブレーキシステム制御ユニット（BSCU）に対する STPA によって導出したシステム安全制約を表 3.1 に示します。これらは、本書のこれまでの章で説明したように UCA のコンテキストテーブルから導出されおり、表 3.1 には、後工程で非安全な機能が導入されてしまうことなく設計変更ができるようにするためのトレーサビリティ情報として論拠が含まれています。

表 3.1:STPA によるシステムレベル安全制約の例

非安全なコントロールアクション	説明	論拠
FC-R1	乗務員は接地前に手動ブレーキをかけてはならない [CREW.1c1]	車輪固定、制御喪失、タイヤの破裂を引き起こし得る
FC-R2	乗務員は、安全な地上走行速度に達する前にTBD秒以上、手動ブレーキを止めてはならない [CREW.1d1]	速度超過、滑走路オーバーシュートを引き起こし得る
FC-R3	乗組員は自動ブレーキ中にBSCUの電源を切ってはならない [CREW.4b1]	自動ブレーキが非アームド状態となる
BSCU-R1	離陸中断中は常にブレーキコマンドを与える必要がある [BSCU.1a1]	滑走路内で停止できない可能性がある
BSCU-R2	接地前にブレーキコマンドを与えてはならない [BSCU.1c1]	タイヤの破裂、制御喪失、怪我、その他の損傷を引き起こし得る
BSCU-R3	離陸後、着陸装置格納前に車輪は固定しなければならない [BSCU.1a4]	飛行中に車輪が回転することにより操作マージンの減少を引き起こし得る

開発のこの段階になるとシステム要求と安全制約を使用して上位のシステムアーキテクチャを定義することができます。STPA で導出されたこれらの上位の安全制約に違反する因果関係シナリオは、詳細なアーキテクチャやシステムの設計プロセスにおいて役立つ、詳細な設計制約やその他の情報を提供することとなります。

設計が継続的に反復して精密化されていくように、STPA 解析も同様に精密化されていき、設計者がトレードオフや効果的な設計を検討する際にその結果は役立ちます。例として、ブレーキにおける設計の決定（油圧制御装置によって駆動するバルブを含む）を示すより詳細なコントロールストラクチャーを*図 3.6* に示します。油圧制御装置に関連するバルブやコマンドの追加を伴う設計上の決定は、すでに特定されているハザードシナリオを少なくとも部分的には解決した結果であるかもしれません。

より詳細な設計レベルで STPA 解析を継続し続けることによって、非安全なコントロールアクションの識別、あるいは精密化された設計（*表 3.2*）に含まれる 3 つの個々のバルブをコントロールする BSCU 油圧制御装置(HC）に関係する非安全なコントロールアクションに関連付けられた安全制約の識別をしていくことができます。

表 3.2:ホイールブレーキシステムのコントロールストラクチャーに関する STPA によるシステムレベル安全制約の例

非安全なコントロールアクション	説明	論拠
HC-R1	HCは、代替ブレーキを必要とする障害がある場合、緑色の油圧遮断弁を開けてはならない [HC.1b1]	通常ブレーキと代替ブレーキの両方が無効になる可能性がある
HC-R2	HCは、スリップが発生した場合、スリップ防止弁を駆動しなければならない [HC.2a1]	スリップを回避し、濡れた状態や凍った状態で完全に止まるには、スリップ防止機能が必要
HC-R3	HCは、ブレーキコマンドが受信できていない場合、緑色の計測弁を開く位置コマンドを供給してはならない [HC.3b1]	コマンドによらないブレーキが適用されていることに乗務員が気付かない可能性がある
HC-R4	HCのスリップ防止装置は、低速地上走行中、ブレーキを解除してはならない	スリップ防止装置は車輪低速回転、車輪回転速度の急激な変化を感知することでスリップを感知するため、低速地上走行中に不慮のブレーキ解除を引き起こし得る

繰り返しになりますが、機能的な設計制約を導出した非安全なコントロールアクションとシナリオによってトレーサビリティは維持されています。よってこれらのコントロールアクションとシナリオは上位の設計と解析へ遡ることができます。その解析結果は設計プロセスを通して実施したSTPAに沿った精緻なツリー構造となっていますので、変更を行う際やある制約がどこから来たのかを理解をすることに役立ちます。

また、非安全なコントロールアクションに関連したより詳細な因果関係シナリオは、どうやって新しいシナリオを除去したり軽減するかについての詳細設計の流れに沿って生成することができます。別の設計解とのトレードオフを解決するために必要となる情報が提供され分析の役に立つかもしれません。このプロセスはシステム開発プロセス全体を通して続きます。

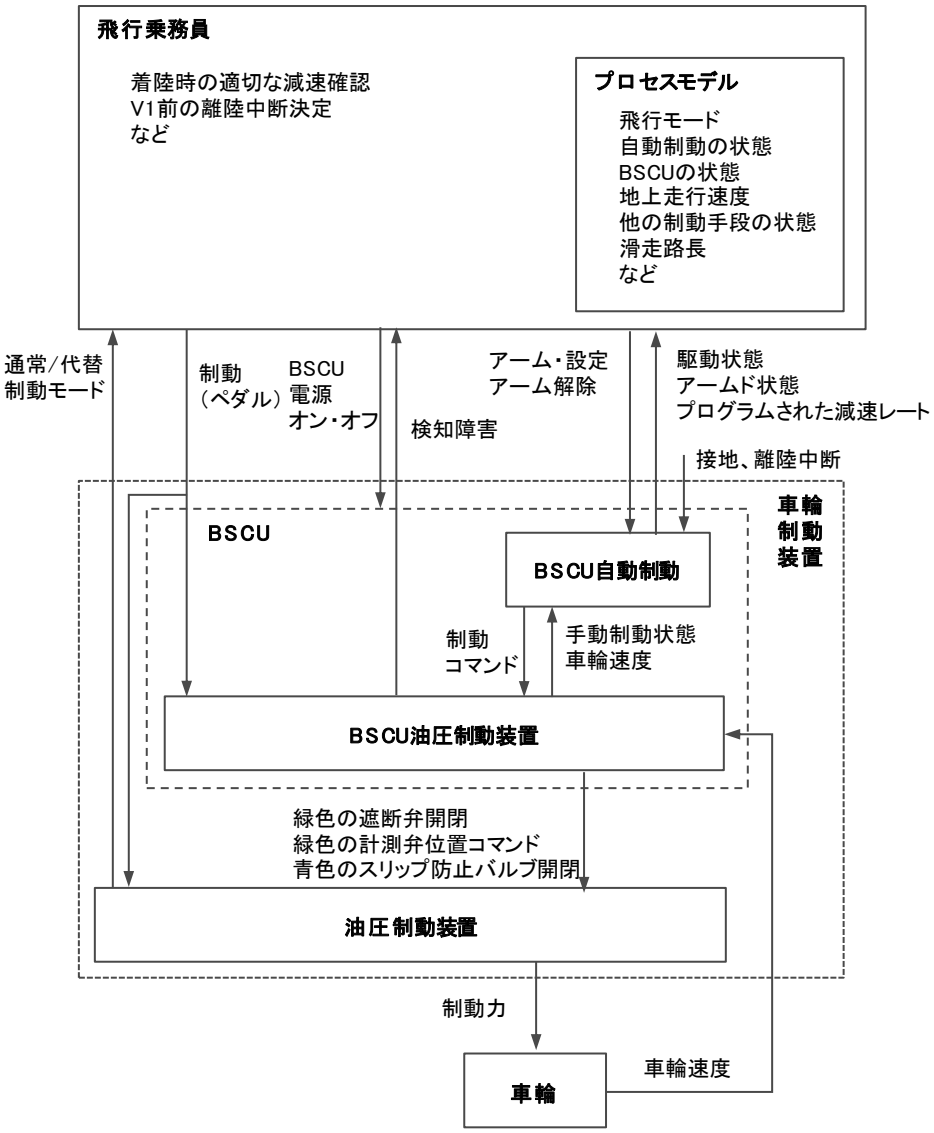


図 3.6:詳細なホイールブレーキシステムのコントロールストラクチャー

7. システム統合

STPA が開発プロセスの初期段階での統合システムの安全要求を識別するために使用されている場合は、システム統合がはるかに容易になるはずです（少なくとも、安全の観点からは）。コンポーネント間の非安全な相互作用は、むしろこの段階で発見されるのではなく、この開発後期段階の前に、システムの設計対象外となっている必要があります。この時点で統合システムの安全に関する欠陥が見つかるような場合は、将来開発プロジェクトのために開発プロセスの欠陥を見直すべきです（前述の通り）。もちろん見つかった欠陥は欠陥を排除するために、制約を識別する STPA 解析に組み込むことが必要になります。

8. システムテストと評価に対する要求生成

因果関係シナリオと V 字の左側のプロセスにおける STPA 解析で導出された他の情報は、テスト要求や評価プログラムを生成するために使用することができます。テスト要求や計画は、システムの種類とすでに実施されているテストに依存します。

一例として、米空軍テストパイロットであるダン・モンテスの MIT での博士論文において、STPA が航空機の飛行試験に使用することができる方法を述べています。図 3.7 は、図 3.2 で示した安全に関するコントロールストラクチャーの例に対して開発と運用のコントロールストラクチャーの間にテストのコントロールストラクチャーを別途追加したものです。新たなハザードや非安全なコントロールアクションや因果関係シナリオが飛行試験プログラムに存在する可能性があることに注意してください。例えば、通常の運用環境と飛行試験環境は異なるかもしれないので新しいコンテキストを考慮する必要があるかもしれません（例えば、航空機に異常なストレスを与えるテストや、極端なコントールアクションを適用して回復可能性を確認するテスト）。STPA は、任意のコントロールストラクチャーでそうであるのと同様に、システムテストのコントロールストラクチャーに適用することができ、その結果は安全な飛行試験の計画と飛行試験計画の生成に使用できます。

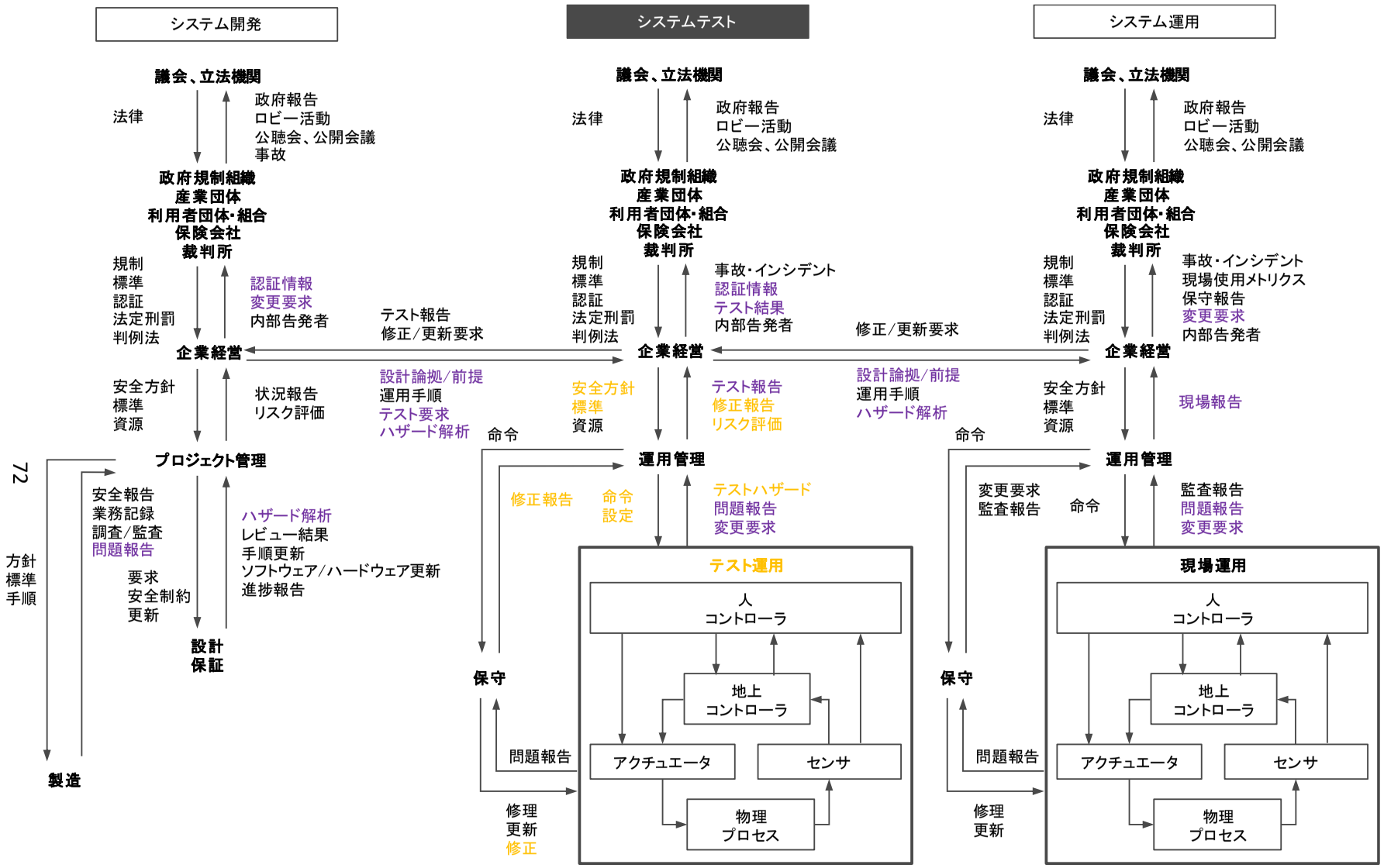
9. 製造管理（生産技術、職場の安全）

STPA は、これまで生産技術や職場の安全にもうまく適用されてきています。もし製造と製造可能性に関する設計が設計しているシステムにおいて重要であるならば生産可能性（特に製造における安全に関して）への考慮は最初の STPA 解析から含まれているべきです。職場の安全への STPA の適用は、組織に関する分析や経営に関する分析と同様に本書の別の章において述べています。

10. 運用上の安全要求の生成（リスクを高める先行指標と安全管理計画を含む）

システム設計において除去や適切な制御をすることができない因果関係シナリオは、システムオペレータが運用による制御を行うことができるようにシステムオペレータに渡さなければなりません。本書の 6 章に述べるリスクを高める先行指標は、開発プロセス中に行われた STPA 解析や運用上の安全解析として行われた STPA 解析から導出することができます。これらの先行指標の目的は、うまくいけば重大な損失が発生する前に設計時に使用環境として設定した前提の欠陥を識別したり、またシステムの動作が時間の経過とともに変化し、使用状況に関する元々の設計前提がいつ崩れるのかを特定することです。

システムオペレータや保守に関する要求は、それらがシステム設計内のコンポーネントとして含まれているため、開発中に行われた STPA 解析から直接導出することができます。因果関係シナリオは、運用要求への入力となるでしょう。


システム開発
システムテスト
システム運用
議会、立法機関
法律
政府報告
ロビー活動
公聴会、公開会議
事故
政府規制組織
産業団体
利用者団体・組合
保険会社
裁判所
規制
標準
認証
法定刑罰
判例法
認証情報
変更要求
内部告発者
テスト報告
修正/更新要求
企業経営
安全方針
標準
資源
状況報告
リスク評価
設計論拠/前提
運用手順
テスト要求
ハザード解析
命令
プロジェクト管理
安全報告
業務記録
調査/監査
問題報告
方針
標準
手順
要求
安全制約
更新
ハザード解析
レビュー結果
手順更新
ソフトウェア/ハードウェア更新
進捗報告
設計
保証
製造
事故・インシデント
認証情報
テスト結果
内部告発者
修正/更新要求
運用管理
修正報告
命令
設定
テストハザード
問題報告
変更要求
テスト運用
人
コントローラ
地上
コントローラ
アクチュエータ
センサ
物理
プロセス
保守
問題報告
修理
更新
修正
現場報告
変更要求
監査報告
現場運用

11. 先行指標監視を含む運用上の安全管理

運用上のリスク管理や安全管理システムは、すべての運用環境に存在しなければいけません。STPA の結果は運用上の安全計画や安全要求を作成するために使用することができます。またリスク管理や安全管理システムは先行指標を特定し、リスクが高い状態となってしまわないかシステムの運用を監視する際に主に役立ちます（このトピックに関する本書の別章を参照してください）。

STPA はまたインシデントや事故を分析し変更を管理する際にも役立ちます。システム設計変更（アップグレードや改造）のような変更、コンテキストの変更（異なる環境や異なる目的のためにシステムを使用）、あるいは運用支援系の変更（物流サプライチェーンや保守の実行）は他と同様に扱われなければいけません。

計画の変更は、常にハザード解析に考慮されなければなりません。このプロセスは、その変更の影響が、特定のシステムのハザードに到達し得るのかそうでないのかが分かるよう STPA に埋め込まれたトレーサビリティによって用意に実施できるようにされるべきです。計画外の変更は STPA 解析から生成された先行指標を介して扱われることとなります。

第 4 章：STPA を用いた職場の安全

ナンシー・レブソン

作業現場や職業上の安全は、しばしば作業者の振る舞いに着目して行われ、それ以上の分析は行われないことが多いようです。しかし職場安全へのシステムの取り組みにおいては、人間の行動はシステムに組み込まれており、システムの影響を受けると考えます。その目標は、ヒューマンエラーを減らすための作業環境をどのようにデザインするのか明らかにすることです。この環境での STPA のアプリケーションは、技術的なシステム上の STPA の標準アプリケーションと同じですが、公開された文献には例が多くはないので、ハンドブックに本章を入れました。

本章にて二つの分析例を紹介します。一つ目の例は、人間が高度に自動化された機械と働く現場です。もう一つはロックアウト/タグアウト（LOTO）を用いる、技術が複雑ではない職場ハザードの例です。両方の場合で、作業現場を安全化するための従来手法と STPA の主な相違は、従来手法が手順を作成して作業者を従わせるのに対して、STPA ではシステム全体を見てハザードシナリオを識別し、これらを排除あるいは低減するための方法を明確にすることです。

以下に挙げる最初の例は、実験的に実在する製造現場に対して STPA を適用したものなので、実際に使われた評価手順や費用を提示することができます。

半自動製造プロセスに対する STPA 適用例

ここで紹介するのは、Nathaniel Peper の修士論文[17]から直接持ってきたものであり、これは、ロボットにより高度に自動化されている作業場での実際の航空機組立プロセスに対して STPA を適用している事例です。修士論文中では以下のように説明されています。

> 「*…プロセスは、作業者の操作する車両が航空機のサブコンポーネントを搭載して工場に入ってきたときに始まる。工場には、さらに二人の作業者が、それぞれ自動地上車両（AGV）を動かして、工場内にある AGV ドッキングと充電をする位置から入ってくるサブコンポーネントのための製品輸送車両（PTV）のある接続ポイントへと、ハンドヘルドコントローラで運転する責任を負っている。この PTV に接続した後、二人のドライバーは、その結合した製品輸送システム（PTS）を、最初に述べたサブコンポーネントの位置まで、工場内を協力して運転しなければならない。この時点で、別のグループにいる作業者は皆で一緒に働いて、サブコンポーネントを、車両位置から PTS まで移動させる。AGV のオペレータは、そのサブコンポーネントが自動製造セルへの受け取りを準備することのできる位置まで、工場内を通って PTS を運転する。このセルの中には、サブコンポーネントに穴を開けることや追加コンポーネントで穴を埋めることを実行する、自動化プロセスを行うための複数のロボットや関連する支援装置がある。サブコンポーネント、PTS、ロボットセル、オペレータ、すべての準備が整ったら、セルオペレータとコントローラは、PTS のコントロールを想定して、その作業命令を実行するために自動化プロセスの間を動かす。この間に、ロボットセル境界内にないサブコンポーネントの部分で同時に起こる作業もある。全体の自動化プロセスが完了すると、作業者は PTS のコントロールを再度想定して、さらにマニュアルの作業プロセスのある工場のラインの後工程へと PTS を移動させる。*」

[17] Nathaniel Peper, Systems Thinking Applied to Automation and Workplace Safety, MIT Masters Thesis (Leaders for Global Operations Program), June 2017.

事故または損失は以下の通りです。

A: 人間の死、怪我、病気

作業現場における一般的なハザードは以下の通りです。

H1: 制御されていないエネルギーへの曝露(または損失につながり得るレベルのエネルギー)

H2: 怪我につながる可能性のある人体に有害な（または人体に応力を与える）動作

H3: 安全なレベルを超える毒性物質への曝露

H4: 聴力に影響を与えるレベルの騒音への曝露

H5: 人間の健康に必要な基礎的要求を満たさないその他環境に長時間晒される

チームは上記の一般的なハザードを対象システムに関連付け、詳細化しました。

H1: 制御されていないエネルギーへの曝露(または損失につながり得るレベルのエネルギー)

H1.1: AVG がその他の物体に最低限保持しなければならない距離を越えて接近する

H1.2: PTV がその他の物体に最低限保持しなければならない距離を越えて接近する

H1.3: いずれかの AGV と PTV が最低限保持しなければならない距離を越えて接近する

H1.4: ロボットがその他の物体に最低限保持しなければならない距離を越えて接近する

電気的、熱的、空気圧、水圧、重力、機械及び非電離放射線（レーザー）などを含む他のエネルギー源は H1 に関連して制御されなければなりません。

H2 と **H3** は以下のように詳細化されます:

H2: 怪我につながる可能性のある人体に有害な（または人体に応力を与える）動作

H2.1: 定型作業中の、怪我につながる可能性のある人体に有害な（または人体に応力を与える）動作

H2.2: メンテナンス中及び点検、トラブルシューティング中の、怪我につながる可能性のある人体に有害な（または人体に応力を与える）動作

H2.3: 設備の導入及び修理、オーバーホール中の、怪我につながる可能性のある人体に有害な（または人体に応力を与える）動作

H3: 安全なレベルを超える毒性物質への曝露

H3.1: 作業者がシステムの操作中に安全なレベルを超える毒性物質に曝される

H3.2: 作業者が装備の点検中に安全なレベルを超える毒性物質に曝される

H3.3: 作業者が製造プロセス中に安全なレベルを超える毒性物質に曝される

ハザードを安全制約に変更できることは自明であるため、ここでは述べません。

実際の分析は 8 人の分野横断的なチームによって実施された。チームメンバーは統合、エンジニアリング、オペレーション、保守部門から集められました。分析の関係者らは全員、システムに対する彼らの責任領域におけるリーダー、またはエキスパートでした。分析の取りまとめ（ファシリテータ）は MIT の修士学生が行いました。学生は STPA の教育を受けているが、プラントに導入されている新たな生産プロセスに対してはあまり習熟していませんでした。

分析には 3 週間を要しました。全体の評価作業ミーティングは 2 回、各回 3 時間未満で実施した。また、残りの分析については、並行して、取りまとめとシステムの各分野の熟練者が一対一で議論を行いました。

最初の評価作業ミーティングでは、STAMP/STPA が会社や分析チームにとって新しいものであったため、手法の紹介を行いました。またチームが防ぎたい事故や損失、ハザードとシステム境界、

安全コントロールストラクチャー図の初期ドラフト版を作成しました。初期ドラフト版のコントロールストラクチャー図を作成した後、各部分の詳細を熟練者と作成しました。図を完成させた後で評価作業を開始することで、STPA の評価をコントロールストラクチャー図上の特定のコントロールループに絞って実施することができました。モデリングと分析のプロセスを分離したことによって、多くのミーティングを分析グループのメンバが都合の良い時間に容易に設定することができました。ミーティングによってコントロールアクション、プロセスモデル、フィードバックやその他の詳細が追加され、競合や問題が関係者間で解決されました。

適切な詳細度の安全コントロールストラクチャー図を作成した後も、STPA の Step1 と 2 を通して、一対一または少数のグループで分析を継続しました。分析結果に対しては、個々の会議で明らかになった所見の確認や潜在的な低減策の議論を目的に別のグループでも議論を行いました。このプロセスは 3 週間に亘って行われ、グループ作業と一対一のミーティング、ファシリテータの個人作業を合計した作業時間は約 300 時間でした。

ハザードを特定した後、4 つの機能、構造、及び RAAs（Resposibilties, Accountability, and Authority）のコントローラに対してコントロールストラクチャー図を作成しました。STAMP/STPA において、RAAs は以下のように定義されています。

責任（Responsibilities）: 基本的なコントロール責任及びプロセスモデル、プロセスモデル変数

権限（Authority）: コントローラによって提供され得るコントロールアクション、どのように、いつ、どこへコントロールアクションが送出されるか。

説明責任（Accountability）: どんなフィードバックが、いつ、どこのコントロールストラクチャーに、どのように送られるか

チームは STPA に慣れていなかったため、全体の詳細をつかむためにグループでコントロールストラクチャー図を作成しました。モデルはシステムの部分ごとに、技術的な専門家を含むグループまたは個人によって修正されました。

4 つのシナリオベースのモデルを図 4.1, 4.2, 4.3, 4.4 に示します。

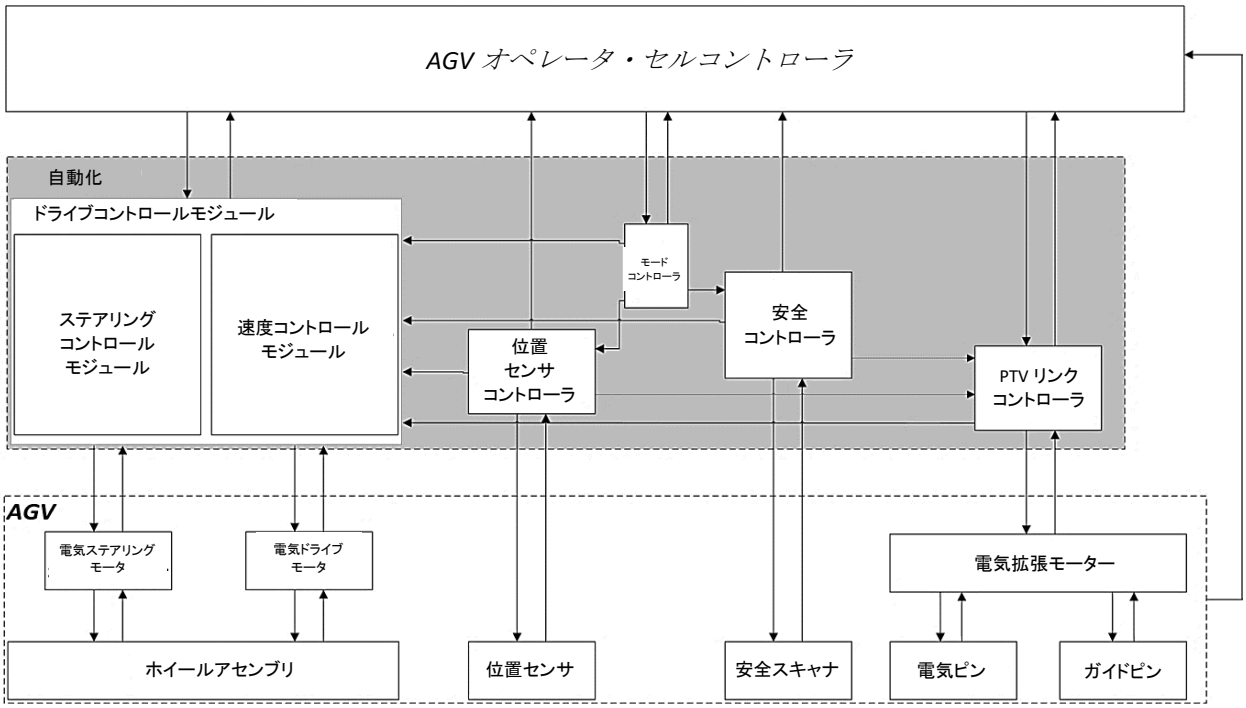


図 4.1：自動地上車両（AGV）安全コントロールストラクチャー

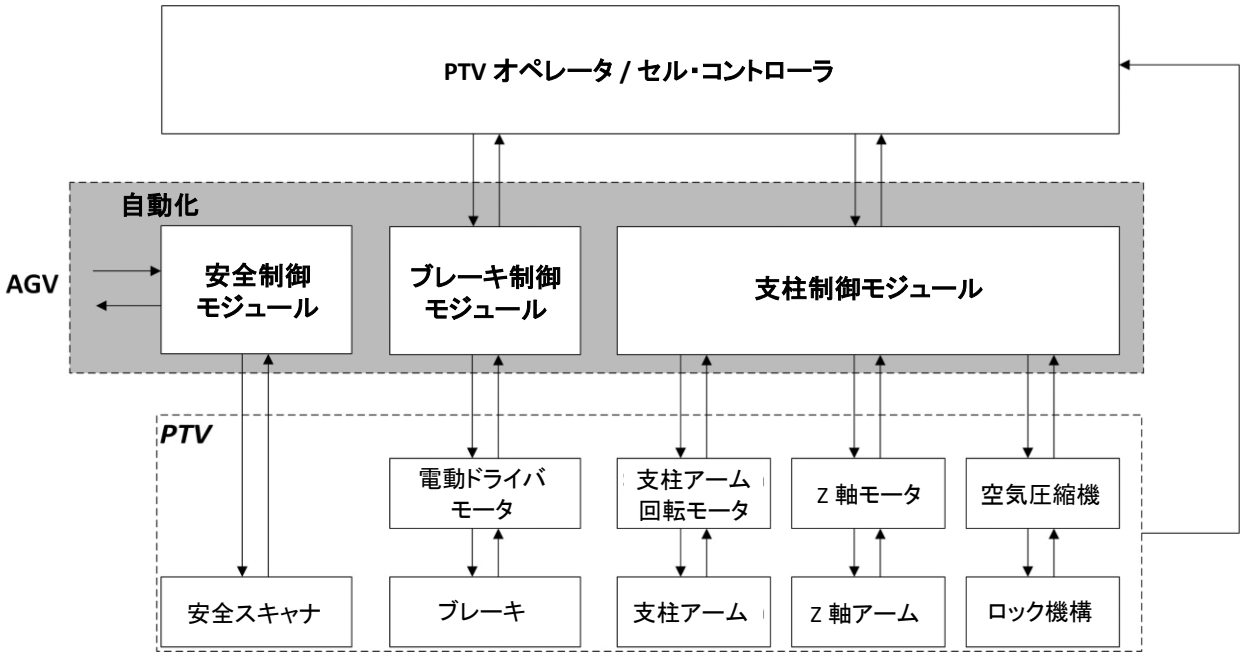


図 4.2：製品輸送車両（PTV）コントロールストラクチャー

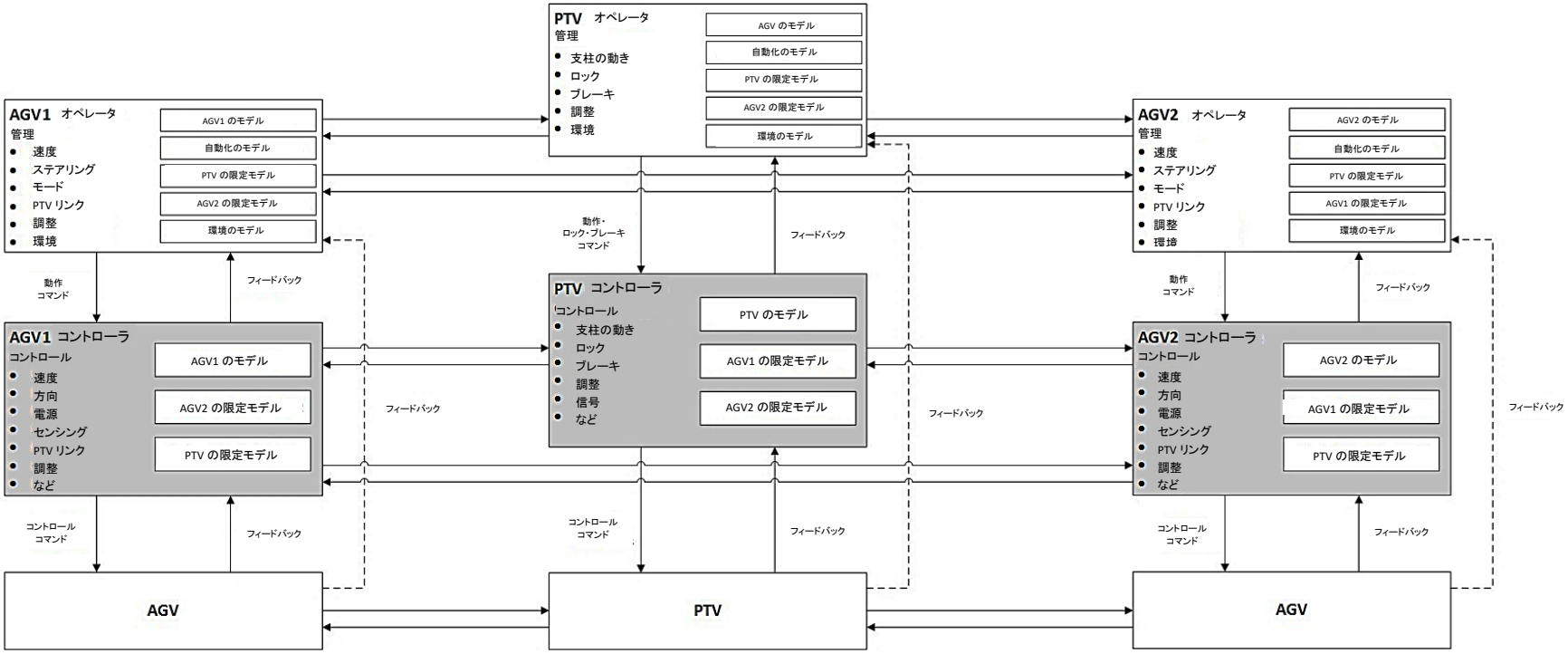


図 4.3：結合した製品輸送システムの安全コントロールストラクチャー

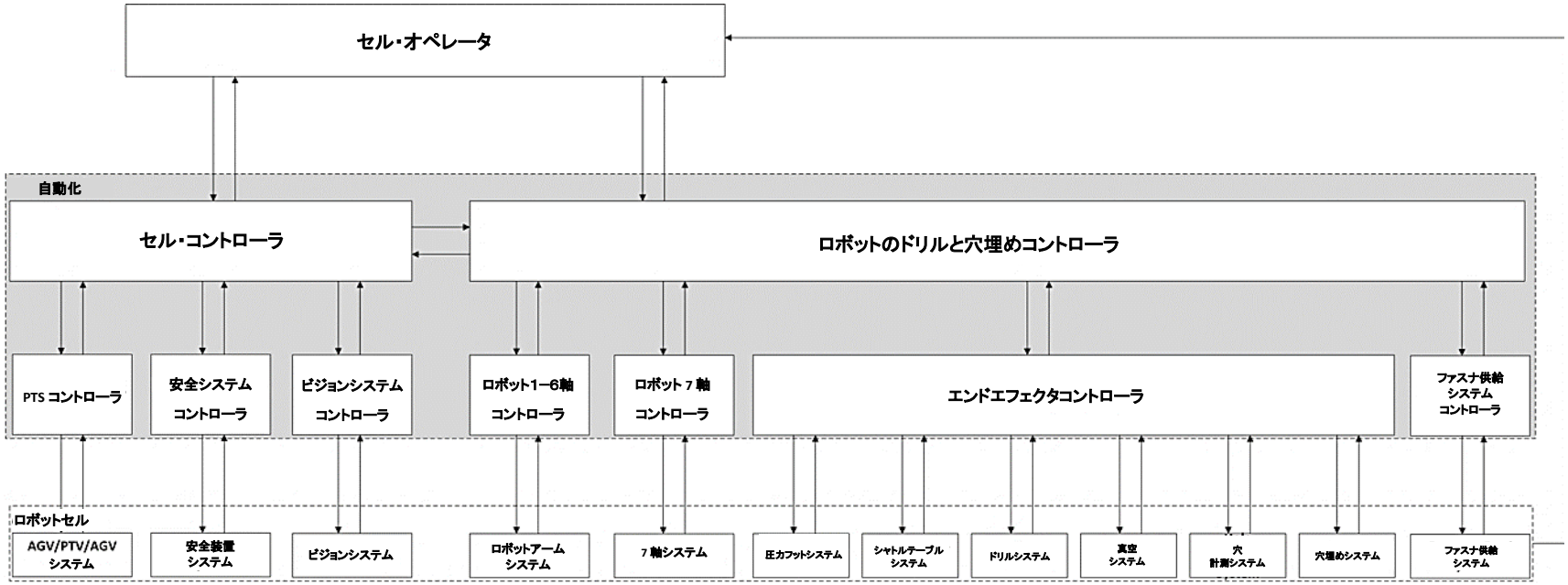


図4.4：ロボットシステムの安全コントロールストラクチャー

ここに示した STPA 分析結果は一例です。より詳細な AFV セーフティコントロールストラクチャの一部を図 4.5 に示します。

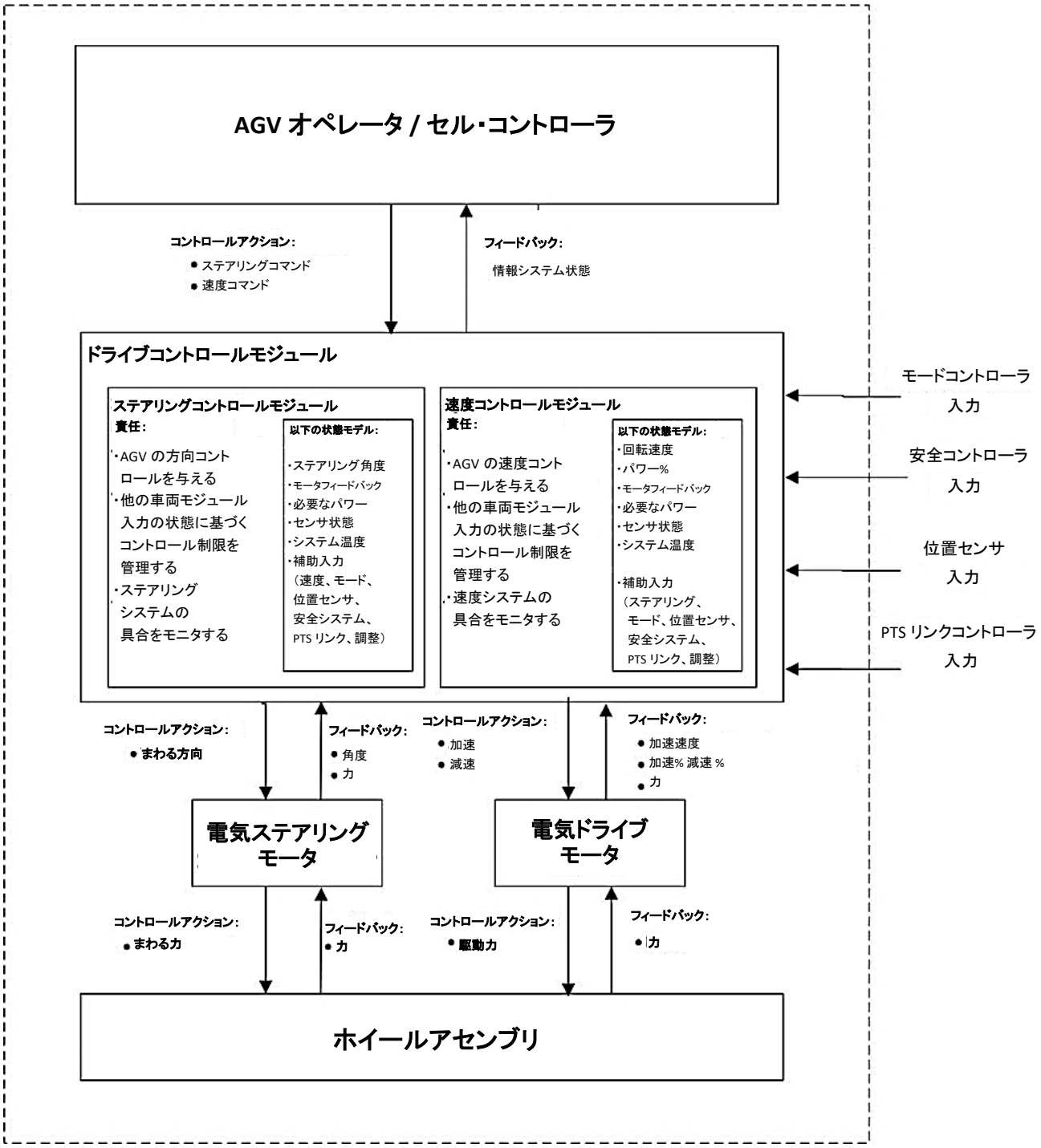


図 4.5 : AGV 安全コントロールストラクチャー・セグメント

AGV に対する UCA テーブルの例を以下に示します。

コントロールアクション	与えられるとハザード	与えられないとハザード	不正確なタイミング /誤順序	早すぎる停止/ 長すぎる適用でハザード
オペレータが駆動コマンドをドライブコントロールモジュールに送信する	**UCA1:** 動作が他の物体との最低限保持すべき距離を越えるときにドライブコントロールモジュールが駆動コマンドを受信する[H1.1]	**UCA3:** 現在の動作が他の物体と最低限保持すべき距離を越える際にドライブコントロールモジュールがコマンドを受信しない[H1.1]	**UCA4:**安全な経路の方向の手前、または後でコマンドを受信する[H1.1]	**UCA7:** 動作が他の物体との最低限保持すべき距離を越えるときにドライブコントロールモジュールが駆動コマンドを長く受信する[H1]
	UCA2: 人が動作する可能性のあるコンポーネントを操作しているときにドライブコントロールモジュールが駆動コマンドを受信する[H1]		**UCA5:** 人が動作する可能性のあるコンポーネントの操作をやめる前にドライブコントロールモジュールが駆動コマンドを受信する[H1]	
			UCA6: 人が動作する可能性のあるコンポーネントの操作を始めた後にドライブコントロールモジュールが駆動コマンドを受信する[H1]	

UCA1（ドライブコントロールモジュールが他の物体との最低限保持すべき距離を越える駆動コマンドを受信する）の因果関係シナリオの例をいかに示します。

因果関係シナリオ1: 不慣れなオペレータが AGV を不適切に駆動する

- 新しいオペレータが十分な訓練を受けていない

因果関係シナリオ2: オペレータがオブジェクトを見ていない、または安全な経路を誤って判断することで AGV が他の物体に向かって駆動する

- オペレータの働きすぎ、環境の変化、他の要因による不注意
- 雑然とした/制限された場所で操作を行った
- 他の作業者、車体、 AGV, PTV, PTS 複合体の翼桁（スパー）に視界がふさがれる

因果関係シナリオ3: AGV のスピード/ステアリングの設定が変更されたことで AGV が他の物体に向かって駆動される

- コントローラの変更に対してオペレータが変更を知らなかった、または変更内容に詳しくなかったため、変更される前の理解で操作を行った。
- コントローラが劣化していたが、オペレータが劣化していることを知らなかった、または慣れていなかったため、変更される前の理解で操作を行った。
- AGV システムの変更に対してオペレータが変更を知らなかった、または変更内容に詳しくなかったため、変更される前の理解で操作を行った。

因果関係シナリオ 4:コントローラのハードウエア故障によって AGV が他の物体に向かって駆動する

- オペレータがコマンドを変更しても、同じ入力が継続して提供される

因果関係シナリオ 5:コマンドに対して車両が動き出すのが遅かったため、オペレータがコマンドを長く送信し続けることで AGV が他の物体に向かって駆動される

- 処理の遅延、計算機のオーバーロード
- AGV へのコントロール入力に対するタイムアウトが存在しない

因果関係シナリオ 6:ナビゲーションする上でクリティカルなシステム（セーフティスキャナ、位置センサ）がオペレータに誤った情報を提供する、または情報を提供しないことで AGV が障害物に向かって駆動する

- HMI(Human Machine Interface) が AGV の側面、地面の近くの容易に確認できない場所にあるため、オペレータがフィードバックを見逃す
- スキャナの電源が入っているのに、または入っておらずステータスを送信しない
- 位置センサの電源が入っているのに、または入っておらずステータスを送信しない

因果関係シナリオ 7:スキャナによって検知されなかった障害物に向かってオペレータが駆動する:

- 障害物がセーフティスキャナの範囲外に存在する
- 工場内の PTV あるいは翼桁の高さにある障害物
- PTV が AGV 上のレールに誘導される
- AGV が想定外の使われ方をする（搬入物がスキャナの視界をふさぐなど）。
- 物体がスキャナの視界内に存在するが、検知の閾値よりも低い
- 物体が AGV の反応速度やスキャンレートを越えて接近し、衝突する。
- スキャナの性能が時間経過によって劣化する
- スキャナが非安全な状態に陥る

因果関係シナリオ 8:スキャナによって発見されていたが、AGV を止められなかったためオペレータが AGV を障害物に向かって駆動する:

- ソフトウエアコーディング上のエラーによって車両が意図通りに停止しない
- CPU オーバーロードによる信号処理の遅れがある
- 不適切な PTV スキャナとのインタフェース、または PTV スキャナが AGV の安全制御モジュールに信号を送信しない

因果関係シナリオ 9:故意にスキャナがオフにされている際に、AGV オペレータが AGV を他の物体に向かって駆動する

- スキャナが無効化されており、オペレータはそれをしならない。AGV は停止をスキャナにたよっている
- PTV のスキャナが無効化され、それが AGV のオペレータにフィードバックされない
- オペレータが物体を見ていなかった

- オペレータは障害物回避を位置センサにたよっている
- オペレータが車両のモード誤って認識しており、車両に異なる動作を期待する
- AGV のシステムが変更されており、オペレータがそのアップデートを知らなかった
- コマンドが遅延し、オペレータがコマンドを長すぎた送信をする

一般的に、識別されたシナリオはハードウエアの故障や（フィードバックの誤りや見逃しによる）プロセスモデルの誤り、システムコンポーネントのインタラクション、工場内でのスケジュール圧力とマネジメント及び規範の組み合わせを含んでいます。分析チームは STPA の Step1 と Step2 を完了した後、ハザーダスな状態に陥ることを防ぐ、または排除するための新しいシステム制御を開発するために分析結果を使用しました。

一例として、システムが AGV と PTV 及び作業者を含む物体との衝突を回避するために、セーフティスキャナに大きく依存していることが挙げられます。オブジェクトがシステムの動作線上にあってかつオペレータかスキャナによって見つけることができなかった場合、事故を防ぐ、又は低減するためにも制御設計が存在していません。対象システムの巨大さ、長さ、重さ、AGV を駆動するモータの力から、特定の状況下では非安全なコントロールアクションが重大な事故を引き起こす可能性があります。オペレータやワーカーはエリア内では AGV の動作に気を付けており、また AGV はセーフティスキャナによってコントロールされる設計でしたが、現状ではいくつかの危険なシナリオが存在することから現状の設計が不十分であることがわかります。

まず、作業者はセーフティスキャナが故障、またはキャリブレーションされていない、正しく設計されていない場合、生産計画の遅れや仕事が完了しない事態を防ぐためにセーフティスキャナを無視する可能性があることが、同社の他の分析によって実証されています。セーフティスキャナが使われていない場合、オペレータや周囲の作業者に対して AGV が安全システムなしで動作していることが通知されることはなく、セーフティスキャナはシステムに統合されていません。ある作業者が仕事を完遂させるためにセーフティスキャナをオフにして、もしオンに戻すことをしなかった場合、次のユーザーはスキャナが無効化されていることを知らずに AGV を操作することになります。

もう一つのシナリオは、スキャナが保護していない障害物に AGV が遭遇することです。車両の駆動モータはドライブコントローラに対して実際に受けた力をフィードバックしていません。これは AGV のドライブコントローラが異なる駆動コマンドをオペレータから受け取るまで、AGV は障害物に対して全力で突っ込み続けるという結果を示しています。この例としては、スキャナのスキャン範囲の外側に何かある場合です。それは、車両を超える上側に障害物があるが依然として輸送システムの上に載せている製品の経路にある、車両の周りの領域で発生しているずれにスキャナが調整されている、障害物が依然としてスキャン範囲内にある時にフォールトと障害物をクリアするスキャナのコントロールロジック、統合された製品輸送システムで複数のコントローラが複数のスキャンシステムを管理する、などが考えられます。

AGV はパワフルで大きく、作業員の安全を守るためのフェンスを設けることができないモバイルシステムです。セーフティスキャナを意図的に機能させなくする方法も多く存在するため、分析チームは実際に開発を行うためにいくつかの新しいコントロール方法を議論しました。一つは作業員がスキャナをオフにした場合、周辺エリアにスキャナがオフになったことを音や光で通知する機能を付けることです。加えて、チームは AGV から周辺エリアの作業員へ通知するための方法に焦点を当てました。オペレータは自分の下した命令を信じていることや、周囲の作業者はロボットがすべきことを信じていることだけでなく、そのシステムが実際に行なっていることを知るための有益で読みやすいフィードバック情報の表示がなくても、大部分が元来設計されたようにオペレータの意図しているように AGV が動くだろうと本来の設計は仮定しているようです。

もう一つの議論すべき点は、多くの協調ロボットのアプリケーションにみられるような AGV の駆動モータのフィードバックロジックをプログラミングすることです。安全スキャナが定位置のみのコントロールをしていることになったシナリオや、スキャナが適切なコントロールを維持してい

ない可能性があることから、システムのためのモバイル機能がある第二のコントロール方法を考慮します。このコントロールでは、ハザードとして定義される最小保持距離の違反を認める一方で、AGV は動きをやめて、事故を避けるあるいは事故の過酷度を減少させることになるでしょう。

これらは STPA プロセスと結果から生じた議論のほんの一例ですが、残りの結果も、この例にとてもよく似ていました。定義されたシステムレベルのハザードを回避するために、協調やコントロールを要するシステム間プロセスモデルの不確定要素についてのコミュニケーション、多様な可動部品のシステムモータフェイードバックロジックや、物理システムから人間のオペレータや周囲にいる作業者達へのフィードバックレベルの増加に対して、推奨される設計を変更しました。

チームは、どのようにシステムが、その場所の安全コントロールを侵害する可能性があるのか、潜在的に事故になり、どのように既存のコントロールを変更する必要があったのか、どんな新しいコントロールを取り付ける必要があったのか、明確にすることに焦点を当てました。

ロックアウト/タグアウト（LOTO）手順に適用した STPA

二つ目の簡単な例は STPA がより伝統的な作業現場の安全ハザードにも適用可能であることを示しています。この例は危険な機械が適切に停止され、メンテナンスや保守作業が完了する前に再起動できないようにするための手続きである伝統的なロックアウト/タグアウト（LOTO）を対象としています。隔離された電源はロックされ、ロックした作業者を識別するためのタグがロックに貼り付けられます。作業者はロックの鍵をもっており、確実に彼または彼女だけが機械を始動できます。LOTO の目的は、マシーンが危険な状態のとき、または作業者が直接機械と接触するような状態で機械が始動する事故を防ぐことです。

このプロセスは比較的フールプルーフなものにみえますが、全米自動車労働組合の調査によると 1973 年から 1995 年の間に構成メンバー内で発生した死亡者数の 20%（83/414）が不適切なロックアウト/タグアウト手順に起因していたことがわかっています。

事故:

A1: 人が負傷する

A2: 製品が損傷する

A3: 製造設備が損傷する

A4: 出荷が遅延する

ハザード:

H1: 人が危険なエネルギーに曝される[A1]

H2: 製品が過剰な、または不適切な危険なエネルギーに曝される[A2]

H3: 製造設備が過剰な、または不適切な危険なエネルギーに曝される[A3]

H4: 作業が重要な局面で遅延する

安全制約(SC):

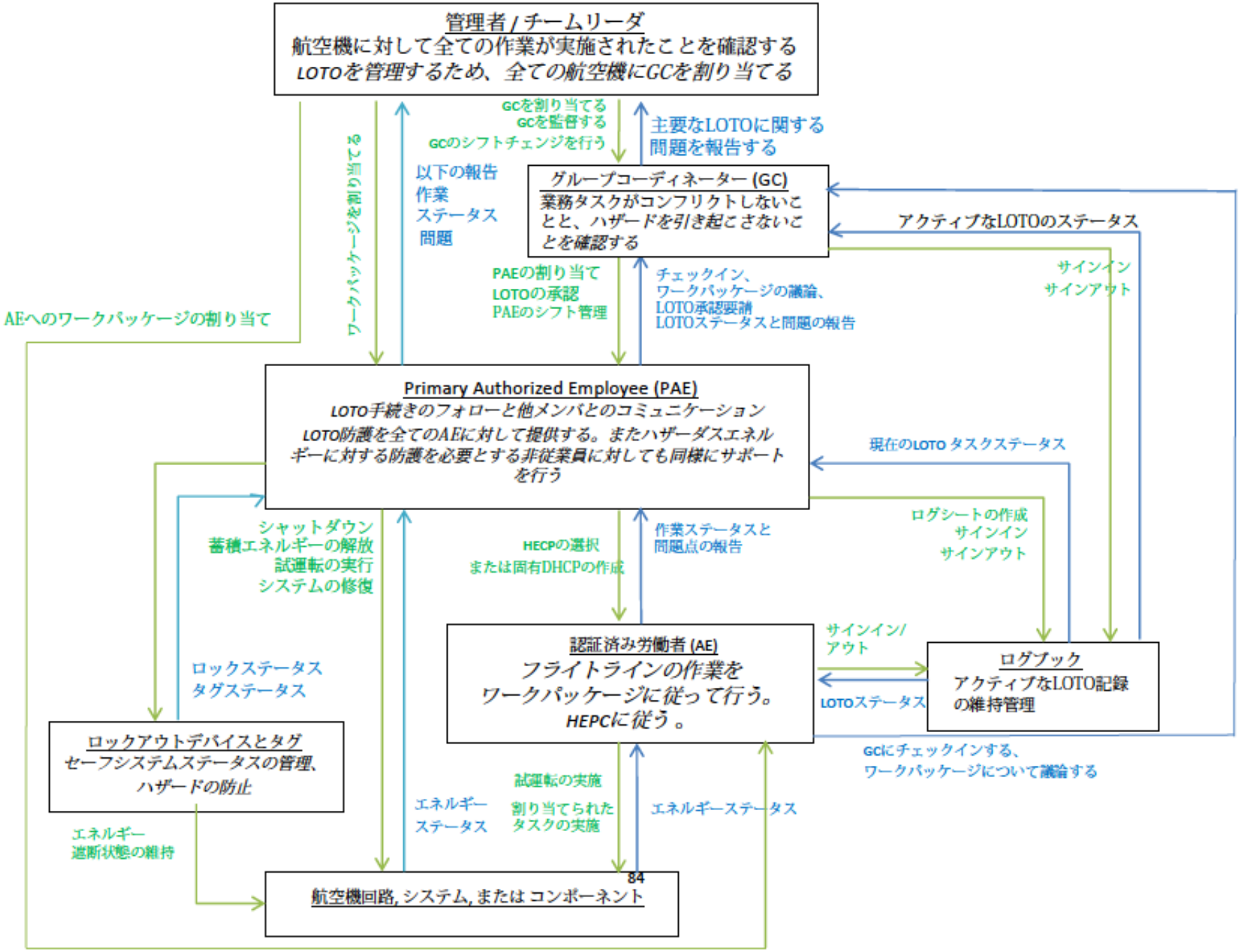
SC1: 人々は過剰なまたは不適切で危険なエネルギーに曝される状況下で労働してはいけない [H1]

SC2: 危険なエネルギーは適切な、確立された閾値の下でしかきどうしてはいけない [H2]

SC3: 製造設備は過剰なまたは不適切で危険なエネルギーに曝される状況下で使用されてはいけない。 [H3]

SC4: 作業はプロセス上の重要な局面で遅延してはいけない（サインオフ、ロックの適用、除去など） [H4]

LOTO の一般的なコントロールストラクチャーの例を図 4.6 に示します。


管理者 / チームリーダ
航空機に対して全ての作業が実施されたことを確認する
LOTOを管理するため、全ての航空機にGCを割り当てる
GCを割り当てる
GCを監督する
GCのシフトチェンジを行う
主要なLOTOに関する
問題を報告する
ワークパッケージを割り当てる
以下の報告
作業
ステータス
問題
グループコーディネーター (GC)
業務タスクがコンフリクトしないことと、ハザードを引き起こさないことを確認する
アクティブなLOTOのステータス
PAEの割り当て
LOTOの承認
PAEのシフト管理
チェックイン、
ワークパッケージの議論、
LOTO承認要請
LOTOステータスと問題の報告
サインイン
サインアウト
AEへのワークパッケージの割り当て
Primary Authorized Employee (PAE)
LOTO手続きのフォローと他メンバとのコミュニケーション
LOTO防護を全てのAEに対して提供する。またハザーダスエネルギーに対する防護を必要とする非従業員に対しても同様にサポートを行う
現在のLOTO タスクステータス
シャットダウン
蓄積エネルギーの解放
試運転の実行
システムの修復
HECPの選択
または固有DHCPの作成
作業ステータスと
問題点の報告
ログシートの作成
サインイン
サインアウト
認証済み労働者 (AE)
フライトラインの作業を
ワークパッケージに従って行う。
HEPCに従う 。
サインイン/
アウト
LOTOステータス
ログブック
アクティブなLOTO記録
の維持管理
ロックステータス
タグステータス
ロックアウトデバイスとタグ
セーフシステムステータスの管理、
ハザードの防止
GCにチェックインする、
ワークパッケージについて議論する
試運転の実施
エネルギー
ステータス
割り当てられた
タスクの実施
エネルギーステータス
エネルギー
遮断状態の維持
航空機回路, システム, または コンポーネント

下表に 2 つの UCA の因果関係シナリオの一部を示しています。PAE（承認一次従業員）は、オンになって動いているコンポーネントの電源を切りません。PAE は、作業が始まった後及び作業が完了する前に、オンになって動いているコンポーネントの電源を切って、作業員達はその場所から安全に出ます。一部分は修正を識別するのに十分な詳細が特定されています。そうでない場合は、詳細について、因果関係シナリオに追加する必要があります。

PAE が作業中のコンポーネントの電源を切らない	… HECP(High Enegy Control Procedure: 高エネルギー制御手続き)が詳細に記載されていなかったため、彼女は正しいコンポーネントの電源を切ったと思いこんでいた。 …トレーニングが不足していたため、彼女は正しいコンポーネントの電源を切ったと思い込んでいた … 高エネルギー制御手続き HECP が詳細に記載されていなかったため、彼女は正しいコンポーネントの電源を切ったと思いこんでいた。 …トレーニングが不足していたため、彼女はコンポーネントの電源を切ることが不要だと思い込んでいた。（訓練） …経験が不足していたため、彼女はコンポーネントの電源を切ることが不要だと思い込んでいた。 …誰かが既に切ったと思いこんでいたため、彼女はコンポーネントの電源を切ることが不要だと思い込んでいた。 …自分で既に切ったと思いこんでいたため、彼女はコンポーネントの電源を切ることが不要だと思い込んでいた。
PAE が作業開始後かつ作業完了前に電源を切る	…予定されていた作業ではなかったので、彼女は作業がまだ始まっていないと思い込んでいた …誰かが既に切ったと思いこんでいたため、彼女はコンポーネントの電源を切ることが不要だと思い込んでいた。 …自分で既に切ったと思いこんでいたため、彼女はコンポーネントの電源を切ることが不要だと思い込んでいた。（プロセスモデルの不整合）

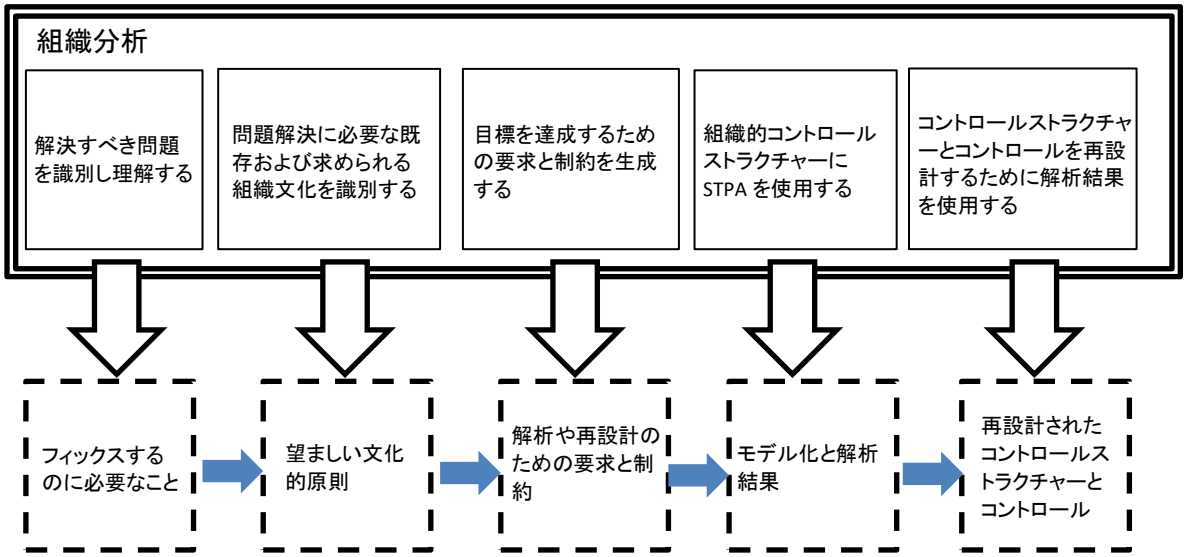
第 5 章：組織と社会分析

ナンシー・レブソン

組織や社会のシステムに STPA を適用することは、基本的にシステムエンジニアリングやリエンジニアリングの活動ですが、設計されたもの（製品）は、組織そのものです。システムエンジニアリングプロセスのこのタイプでは、目標は、システムがいまどこにあり、どこに向かっていく必要があり、そこへどのようにしてたどり着くことができるのか、決定することです。分析する製品及び二次的産物に含まれるもの：

1. 解決する必要のあるエンジニアリングおよびビジネスの問題、すなわち、分析の目的
2. 目標を達成するために必要な組織的な文化
3. 組織構造のための要求と制約
4. 組織構造の STPA 分析
5. 解析結果の使用対象：
    a) 組織構造の設計あるいは改善
    b) リスクおよびリスクの先行指標を識別すること
    c) リスク管理プロセスの設計（または再設計）

これらについては、ハンドブックの本章で説明しています。全体的なプロセスは、下のボックスにまとめています。



STPA の経営と社会システムへのアプリケーションは、設計の製品上での使用と似てはいるが、いくつかの重要な点で異なります。最も重要な違いは、解決すべき問題の最初の理解にあります。任意のシステムエンジニアリングやリエンジニアリングプロセスは、その問題を理解するから始まるべきですが、製品に使用されるよりも、組織に適用されたときは、この理解のプロセスはより複雑です。航空機、自動車、または原子力発電所のための目標と制約は、かなり理解されており、このようなシステムを分析するための最初のステップは、彼らが特定の製品に適用したいと思う具体的な目標と制約を選択するための利害関係者を得ることです。

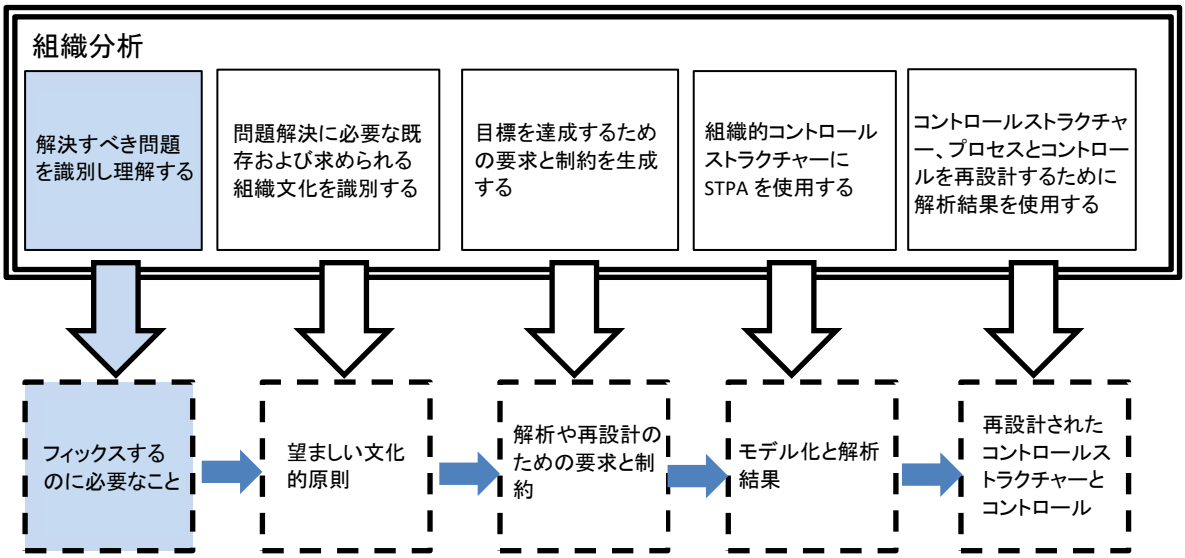
社会や経営システム（通常は既に存在している）の分析を行う際は対照的に、最初のステップは、その目標が現在達成されていないのはなぜか、どうすればより上手く達成することができたか、その理由を特定して理解することです。このハンドブックにある例示のほとんどは安全に焦点

を当てているので、ここでの例は、安全性やセキュリティとは異なる創発特性に STPA を使用する方法を説明するために、また異なった特性について検討します。

ここで使用する例では、システムエンジニアリングの効果、すなわち、予算の範囲内で、時間内にその製品を出し、顧客のニーズを満たすことです。その会社では、製品は複雑であるとみなしており、政府や営利団体の両方の外部顧客のために設計されている、と仮定します。

実存の企業にこのプロセスを実行したときに得られる独自の情報を漏洩することはできないので、プロセスの一般的な結果とその実際の企業での用途をここに示します。

解決を要するエンジニアリングとビジネスの問題の識別



常にならうことはありませんが、問題の解決に成功するための最初のステップは、解決すべきである問題点を理解することです。解決すべき問題を理解することに時間を費やす前に、ソリューションを特定することが成功につながる可能性はそれほど高くはありません。

利害関係者と鍵となる職員のインタビューは、単純に観察されているように、これらの問題について解釈されている原因と解決すべき問題点を特定するのに便利な方法です。得られた情報は、現在のシステムと潜在的な改善の分析のために使われます。そのようなインタビューを行う方法については社会科学の文献に含まれていますが、ここでは、最初に我々の目標を説明し、匿名性を約束して、簡単な質問で探りを入れて、彼らが望む方向へ回答を導くインタビュアの自由さを許し、シンプルで簡単なインタビュープロセスを使用することで、必要な情報を得ています。

この例示の目的のために、あるインタビュープロセスで、システムエンジニアリングは常に自分の会社では成功しないことがあるのはなぜか尋ねると仮定します。インタビューは、少なくとも管理、エンジニア、および可能な場合は顧客を含みます。システムエンジニアリングに焦点を当てれば、それらの結果は、次のように問題が解釈されたような説明や、問題が起こった原因を次のように上げるかもしれません：

- プロジェクトには、必要とされる時に常に専門知識とスキルがあるとは限らない。能力開発は、結局、必要とされているほど強力ではない。
- 提案書の執筆と契約交渉は、非現実的な予算やスケジュールを立てることにつながる。そのため、プロジェクトの取り掛かりが遅れて、これが製品開発の手抜きにつながり、さらに物事を悪化させる。

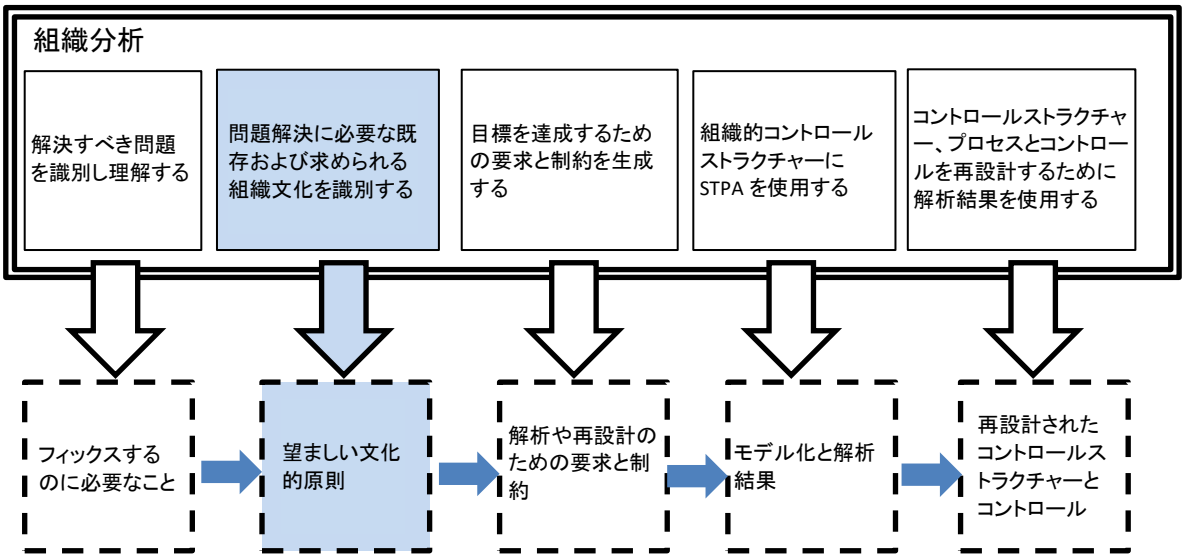
- ここに記述されているリエンジニアリングプロセスで解決すべき問題を決めることが重要であるのと同様、そのソリューションを作成する前にどの問題が解決されていくのかを決定することによって、顧客のニーズを満たし、システムエンジニアリングプロセス自体で作業をバックアップして再度行うことを回避する。しかし、この例にある会社では、事前の計画と開発の初期段階で、あまりにも少ししか労力を費やすことがない。これは、顧客のニーズと製品が使用される環境を理解するための時間があまりにも少ないこともある。エンジニアは、開発の初期段階で期待された製品に関するクリティカルな想定や要求を識別することから起こる問題ではなく、悪い計画から起こる問題を解決することに時間をかけている。性急な決定は、のちに時間とお金が必要になり、多くの場合、本来の意図ほど有用でない物をもたらす。
- 初期の計画の欠如は、多くの時間が会議と再計画に費やされ、先に進捗することから著しく後退し、問題の予算を立ててスケジュールすることになり、時間の浪費になって「スラッシング」製品を生み出すような遅延と再計画の連続につながる。また、慌てて変更が起こり始めることは、影響分析の変化や管理の変化が低下し始めて、問題を加速させる。
- プロジェクトのリスク分析と対策がおざなりであると、「ボックスをチェックする」活動になる。リスクを識別し軽減するために割り当てられたリソースはほとんどない。多くの場合、リスク分析は、他の活動よりも低い優先順位を与えられ、これをうまく行うのに十分な経験を持っていない新しいエンジニアに委任される。リスクの緩和は、多くの場合、財源が与えられておらず、それを達成するための手順はマスタースケジュールではない。
- システムエンジニアリングのゲートとマイルストーンは、プロセスの後半の段階の準備を確実にするために作成された。ほとんど誰もがゲートの必要性を理解しながら、スケジュールにプレッシャーが残ったままである。長期的なスケジュールと品質の問題は、事前に定義したスケジュールに対する短期的な焦点が、その一部であるとみなされる。プロジェクトマネージャは、多くの場合、彼らが次のフェーズで対応すると仮定して、マイルストーン（「ゲート」）を省く。プロジェクトは、決して追いつくことがなく、遅れて問題を修復することになり、もちろん常に大変高価である。後半のプロジェクトで変更が必要なときは、予算とスケジュールが粉々に吹き飛ばされる。さらに悪いことに、多くの場合、ほとんどではないが、重要な意思決定が開発の初期段階で行われて、莫大な、そして通常は非現実的な再設計をせずに元に戻すことはできない。その時点で、唯一の実行可能な解決策は、通常、不良品が発生することになるワークアラウンド（回避策）を作成することである。急いで行った決定と変更は、多くの場合、解決されているものよりもさらに悪い問題をもたらす。
- リワークに関連する多くの問題は、不十分な設計意図や根拠の記録が不十分である。そのような文書が欠如していると、再び提案したり試みたりするのは不適切であろうと早期に廃棄した解決策があるかもしれない。
- このすべてが、ヒーローやカウボーイのようなエンジニアリングが標準となることにつながり、それが非常に経営者によって高く評価されている。スケジュールとコストの目標を満たす責任があるが、開発のもっと初期の段階で欠陥のために要される変更はほとんどないことを意味して、最初の品質に対する報告義務や高い報酬がほとんどない。
- 継続的に改善するにはフィードバックが適切でなく成功と失敗の両方から学習が不十分になる。すべての問題に対しての解決策かのように、以下のような強健なプロセスに重点をおきすぎる。管理者は、正しくプロセスを制定しているならば人々が関与していることを忘れて、すべてのシステムエンジニアリングの問題が解決されると考える。経営者は、問題の解決策として、あるいは、開発プロセスの中の人間のコンポーネントをどのように開発し支援するのか、プロセスを使用する方法において、よりインテリジェントにする必要がある。この会社は「観測すること」（データを収集すること）から教訓をえることは得意であるが、「学習すること」から教訓をえることは得意ではない。

- 学習には、（厳密に特定のプロセスを強制することとは対照的に）技術革新、リスクテイク、および柔軟性が必要である。しかし、職場での革新を促進するようなファクターのために、十分な権限委譲やサポートはない。インタビューで述べられる共通的な意見としては、従業員がリスクを取ることや革新を導入することを恐れており、その理由は、これに対して少ししか報酬がなく、実際には罰則が与えられることもよくあるからである。そこで、彼らは常に行っていることや、やるように言われていることから外れない。失敗の恐れと、結果としての罰則が、革新を抑圧することになる。現在のエンジニアリング構造では、混乱をもたらすような革新的手法を扱うことができない、また、変更を処理する能力はほとんどないと説明している。
- 私たちのインタビューでよく出てくる、もう一つの革新性の欠如する理由は、技術革新やエネルギーの大部分が、より良い製品を作成するための努力にではなく、悪い慣行や欠損のリスク管理から起こるプログラム問題を解決することに費やされていることにある。
- 生産性とサポート（保守性）の設計における焦点が不適切であり、必然的に製品の進化のコストと品質の要因になる。
- ある完成した設計やアーキテクチャにおける安全性とセキュリティを確保することにあまりにも多くの重点が置かれており、その始まりから製品への安全性とセキュリティを構築することには重点が置かれていない。
- 自社製品の部品を下請けさせる会社では、役に立たないサプライヤーの見落としがあり、スケジュールの悪化や再作業が必要になると、よく耳にする。
- スケジュールが品質と競合していると、プログラムそして事業戦略上の考慮から、技術的な意思決定の独立性が欠如する。これは、オープンで正直なコミュニケーション、そして独自の経路の欠如に結びついて、技術的なリスクと問題を高める。
- エンジニアや管理者は、良い意思決定のために必要な情報やフィードバックを得ていない。時には、自分たちの持っている情報を無視している。フィードバックループは、存在していないか無効である。多くの場合、フィードバックが受信されたが無視されることには不満がある。よく知られている確証バイアスと呼ばれる現象があり、これでは、人々は、自分の仮説を支持する情報を信じる傾向にあり、そうではない情報を無視する傾向がある。確証バイアスは、人が聞きたくない情報は、信じなかったり無視する時に起こる、防御的回避に関連している。
- 特に製造業や顧客の人々によって、よく表出される問題は、設計者や製品の組立者の間の関係や、設計者と設計したシステムのオペレータとの間の関係に関したものである。一般的な問題は、作業指示の不十分さ、エンジニアを設計するメカニックからのフィードバックの欠如に関連している。

上記のような多くの問題は、意外に思えるかもしれないが、実際には産業システムエンジニアリング組織ではかなり一般的です。

私たちは、インタビューの間に最近の組織文化について学ぶようにしています。

目標を達成するために必要な組織文化



組織文化を議論するために、私たちは実用的な定義から開始する必要があります。ここではエドガー・シャインの定義を使用します。

<u>組織文化</u>とは、意思決定の基礎をなす、あるグループや組織で共有された価値および深い文化的前提である。

最も効果的に組織内で重大な変更を行うには、それらの変更を促進するような文化を変更する必要があります。

インタビューの間に、私達は、組織文化とするものの意味について定義し、彼らの組織で必要とされていると思われるのは、どの文化的な変化であるのか、インタビューされる人に尋ねます。彼らの答えは、その文化が推進すべきだと考える、望みの行動を反映します。インタビューで得たフィードバックの中で、システムエンジニアリングの有効性と、改善するために必要な組織文化に関するものには以下の例があります。

- 意思決定の主な要因として、スケジュールへの関心が低いこと
- 顧客や会社内で「ノー」と言える能力。エンジニアリングの回答ではなく、マーケティングとセールスの回答が提供される。
- 即時的なプログラムと顧客を超え、プログラム全体を超えたところで、思考すること
- より効果的で明確なコミュニケーション
- 失敗と成功の両方から、より多くを学ぶこと
- 後に時間を節約するために、最前線で多くの時間を費やす
- もっとコラボレーションすること
- エンジニアを大切にし、尊敬して待遇すること
- 顧客が製品を使用して満足できるまで、その仕事は終わっていないと信じること
- 助けを求めることを恐れないこと
- 製品に関する開放性と誠実さ。メッセンジャーを攻撃しない
- 局所的ではなく、大きく会社の目標を考えること
- 「苦痛を延期」しないことと、長期より短期の優先順位を与えること

- リスクを軽減する計画は、ただ意思疎通するだけではなく、適切な財産であると確認すること

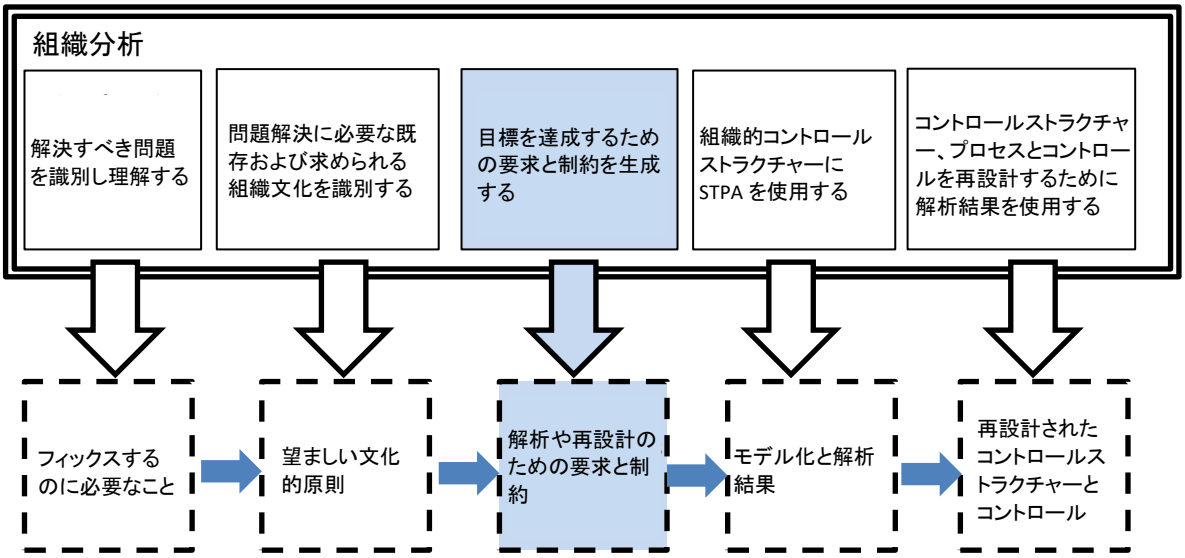
明らかにこれらはインタビューで明確になった問題に関連しています。私たちの通常のプロセスにおける次のステップは、インタビューで明るみになり、前章に記載したような懸念と共に、組織の希望するような種類のものを与える文化的な原則（共有組織価値システム）として再度示すことです。その結果は、広く行き渡っている一連の信条あるいは価値として、また組織設計に実現されている場合、明確にされた問題を解くことは長い道となるであろうことが記されます。ここでは、これまでに提示されたインタビュー結果の例から作成されるかもしれない文化的な一連の原則の例は以下のとおりです。

- 品質と安全性：品質と安全性が増すことは、コストとスケジュールが減ることであり、長期的には増益につながる。安全性は、生産性と収益性の重要な決定要因である。
- 計画：前もって優れたシステムエンジニアリングを行い、後にコストのかかる作業のやり直しを削減する。優れたエンジニアリングは、過度の努力とリワークを必要とすべきではない。
- 労働力と能力開発：効果的な労働力とスキルの開発と活用は、その士気を高め、コストを低減し、将来のために才能の基盤を開発することに役立つ。
- 技術的な独立性：プログラム実行の意思決定とは無関係に行われた場合の技術的な意思決定が良くなる。当面のプログラムを超えて考えることは、全体として企業のための長期的な利益につながる。
- コミュニケーション：効果的で明確な一貫性のあるタイムリーなコミュニケーションと情報の共有は、エンジニアリングの目標を達成するために不可欠である。
- 学習と改善：効果的な技術は、成功と失敗の両方からの学びを含んだ、継続的な学習と改善が必要である。知識の構築は、製品を構築することと同様に重要である。
- イノベーション：新たな知識と技術（革新）への投資は、成長と生産性を駆り立てる。
- コラボレーション：適用可能なすべてのエンジニアリングを含めて、うまく成功するエンジニアリングには、分野や構造的な境界を越えて作業することが必要になる。
- リスクマネジメント：リスク評価と管理は、パフォーマンスの目標を達成するための鍵である。

所望する文化的原理（価値および信条）のリストを生成することは、単なる出発点であり、レビュー、編集、（完成時には）リーダーシップチームとエンジニアからの賛同（バイイン）が必要になります。その目標のためには、議論を始めて、企業文化を変更する方法のための模索を開始しなければならないでしょう。

一度コンセンサスがエンジニアリングの文化の形成でなされたなら、目標を達成するのに役立ちます。そして、その文化を実現するための必要条件の面で、前に進む道筋は、計画、実現しやすくなります。

効果的な組織構造を実現するための要求と制約の生成



重要なステップは、任意のシステムエンジニアリング作業のための目標を明確にした後、要求を作成することです。これらは、問題を特定するプロセスの結果、所望のエンジニアリング文化の誘発から生まれます。ハンドブックのこのセクションでは、これまでのサンプルの分析結果を用いて生成される要求の例の一覧を示します。それらは、合意された文化的な目標のそれぞれを言い換えて、グループ化されています。

品質と安全性：品質と安全性が増すことは、コストとスケジュールが減ることであり、長期的には増益につながる。安全性は、生産性と収益性の重要な決定要因である。

- 各プロジェクトは、成功裏にプロジェクトの目標を達成するために必要となる、適切なスキルと専門知識を持つ人々を保有すべきである。
- 安全性とセキュリティ分析は概念分析の段階から開始し、これらが行われている間、設計上の決定を通知するために、安全性とセキュリティの分析を用いて開発プロセス全体にわたって継続するものとする。
- ゲート（マイルストーン）コントロールには、満足したと考えられる前に、品質と安全性の分析が入っていなければならない。
- プロジェクトマネージャは、最初の品質[18]に対して報われなければならない。
- すべての製品開発は、生産性とサポート性を重要視するべきである。
- 検証は、開発の各段階で重要視されるべきである。検証ステップを避けて通る、あるいは減少させるような試みは防止しなければならない。
- 会社は、下請け業者の製品の品質と安全性を監督しなければならない。

[18] 「正しい」行動を促すことの重要性は、「安全管理システム」の章で説明します。例えば、タイムラインを満たすためにマネージャーにボーナスを与えると、品質が低下する可能性があります。

<u>計画</u>：前もって優れたシステムエンジニアリングを行い、後にコストのかかる作業のやり直しを削減する。優れたエンジニアリングは、過度の努力とリワークを必要とすべきではない。

- 提案の中で将来の業績を予測することは、少なくとも部分的には、顧客の目標のように単に外部からの期待に基づいてというよりも、実行可能性とエンジニアリングの実現可能性に基づいていなければならない。
- 詳細なプロジェクトの計画とシステムの分析は、詳細なリスク管理と緊急時対応計画、顧客のニーズ、製品によって解決すべき問題、および製品が使用される環境の十分な理解、そしてシミュレーション、プロトタイピングなどのリスクを理解して軽減するための初期の活動を含め、プロジェクトの最初に行わなければならない。
- 変更が必要な場合は、厳格な変更管理を実施しなければならない。
- 下請けの決定は、適切なエンジニアリングの入力とともに行わなければならない。
- エンジニアリング計画には、避けられない「技術的に未知数のもの」の取り扱いを可能にするためにスケジュールと費用マージンあるいはリソースの柔軟な再配分のための行程を入れなければならない。
- 計画は、顧客のニーズの変化に直面しても適応しなければならない。

<u>労働力と能力開発</u>：効果的な労働力とスキルの開発と活用は、その士気を高め、コストを低減し、将来のために才能の基盤を開発することに役立つ。

- 適切な訓練、指導、および成長の経験を提供することにつながる、スキルのニーズを予測しなければならない。特に、システムエンジニアリングについては、訓練を受け各プロジェクトの中核をなすべきである。
- 可能であれば、指導者は、分野固有の経験に基づいて選ばれるべきである。この経験は、将来のリーダーのために開発されなければならない。
- マネジメントは、その労働力が適切に報われて、彼らの貢献は価値あるものと確証するための措置を講じるべきである。

<u>技術的な独立性</u>：プログラム実行の意思決定とは無関係に行われた場合の技術的な意思決定が良くなる。

- 独立した機関（プロジェクトマネージャーではなく）は、あるゲートが通過されているかどうかを判断する責任を持つ。
- スケジュールは品質と競合しているとき、技術的な決定は、プログラムや事業戦略の検討事項から独立していなければならない。ある独立した技術機関は、この独立性を確保するための責任を持たなければならない。
- プロジェクトマネージャよりも高い権限は、プロジェクトマネージャと独立した技術的な権限との間の競合を解決する責任を負わなければならない。
- 技術的なリスクや他のエンジニアリングの問題を高めるための、独立した経路が存在していなければならない。

<u>コミュニケーション</u>：効果的で明確な一貫性のあるタイムリーなコミュニケーションと情報の共有は、エンジニアリングの目標を達成するために不可欠である。

- エンジニアリングと事業買収との間のより良いコミュニケーションのためには、準備している提案の実行可能性についてのフィードバック、完了（またはキャンセル）した契約上の成功または深刻な問題を含めて、制定されなければならない。

- 意思決定者が、適切な意思決定を行うために必要な情報を持っていることを確認するために、フィードバックと情報チャンネルを作成しなければならない。

学習と改善：効果的な技術は、成功と失敗の両方からの学びを含んだ、継続的な学習と改善が必要である。知識の構築は、製品を構築することと同様に重要である

- 事後検討会[19]は成功と失敗の両方から学ぶために、プロジェクトマネージャによって行われなければならない。プロジェクトマネージャは、これらの事後検討会における誠実さと徹底のために報われなければならない。
- プロジェクト管理情報システムは、過去のプロジェクトからのデータからだけではなく、成功と失敗の両方から学んだ教訓も含まれていることが確立されなければならない。

イノベーション：新たな知識と技術（革新）への投資は、成長と生産性を駆り立てる。

- 革新性、柔軟性、および適切なリスクテイクを奨励し、報われなければならない。
- 組織は、新しい技術の課題と機会や、ほとんどのプログラムに発生する避けられない技術的な問題に対して、革新的なソリューションを生成するためのリソースを提供しなければなりらない。
- エンジニアが革新的なリスクを取ることを妨げるような文化的な壁を識別して、これを除去しなければならない。

コラボレーション：適用可能なすべてのエンジニアリングを含めて、うまく成功するエンジニアリングには、分野と構造的な境界を越えて作業することが必要になる。

- コラボレーションを促進するために、分野と構造的境界を越えて働くことが促進され、最適化されなければならない。
- 作業グループには、該当するすべての分野を含んで、分野や構造上の境界を越えて確立されなければならない。

リスク管理：リスク評価と管理は、パフォーマンスの目標を達成するための鍵である。

- リスクの識別と評価は、その主題の最も知識豊富な専門家が識別するべきであり、きちんと明確に影響を受ける担当者のすべてに伝達するべきであり、そしてこれはプログラムの目的と顧客のニーズに合わせて定められる計画や資源を与えられ、明確に定義された一連の緩和策につながる。
- リスク軽減は、初期のリスク評価と経験にともなったあらゆる更新に基づいて、プロジェクトの開始時から計画されるべきである。リスク軽減措置には、資金を供給し、マスタースケジュールに組み込まれなければならない。
- リスクは、明らかに影響を受けるすべての関係者に伝達されなければならない。

---

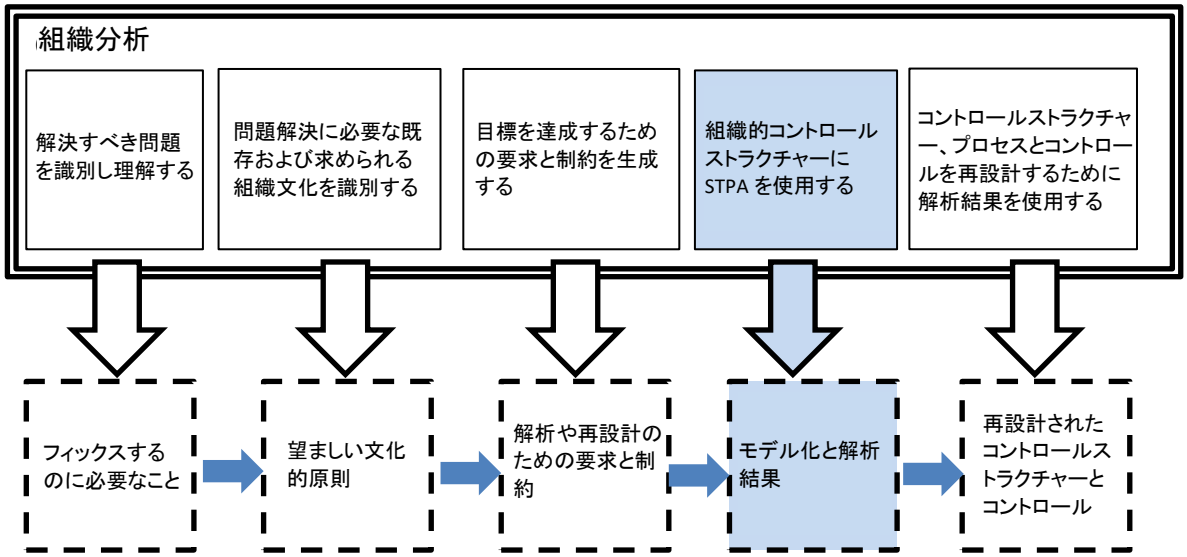
[19] Microsoft が成功した理由の１つとして、容赦のない正直なプロジェクトの事後検討会が取り挙げられています。プロジェクト結果の詳細な分析のための時間が与えられ、リーダーは成功と失敗の理由について正直であると予想されます。軍隊はアフター・アクション・レビューと呼ばれる同様の概念を持っています。自分自身の経験（そして他人の経験）から学ぶことは、将来の成功のための要件です。正直に自分のパフォーマンスを評価して、学んだ教訓を共有することで人々に報いないことは、同じ過ちをしてしまうかもしれない新しい人間を促進することになり、新しい経験から学ぶのを妨げるかもしれません。

- リスク管理計画には、「まだ不明の未知数のもの」に対して十分なコストとスケジュールの余白がなければならない。
- リスク管理には、管理者が現実的なリスクを無視できないような、適切なコントロールがなければならない。
- リスク管理には、適切に資源を与えられなければならない。

これらは、単に、組織の基盤として機能する文化的な原則の合意から生成される要求の例です。その他も起こりうるでしょう。

これらの要求が合意された後に、実際のモデリングと解析を開始することができます。

組織構造の STPA 解析



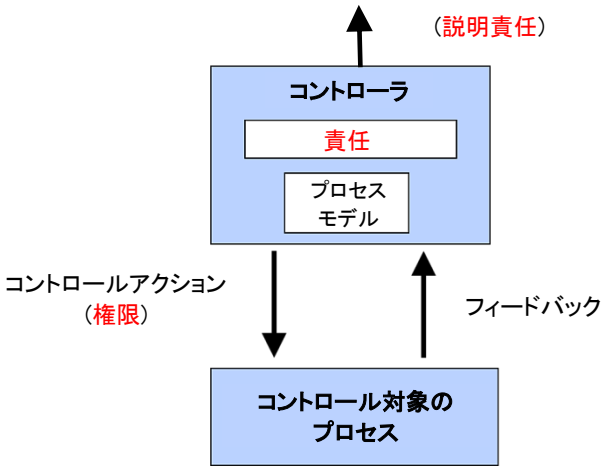
プロセスの次のステップは、現在のシステムエンジニアリングのコントロールストラクチャーをモデル化するためにあり、識別された要求が（もしも少しでも）現在履行されているところ、あるいは少なくともこれらの活動のために責任が割当てられているところを識別するためであり、さらに、識別された（フェーズ 1 で）懸念事項は何故適切に取り組まれておらず、要求が履行されたのか、それらの理由を決定するためです。これらの結果は、組織設計に、より良い要求を実現するであろう変更を提案するために使われます。

モデリングが完了するか、または少なくとも一般的なコントロールストラクチャーが理解されている地点で、構造上のハザード解析を開始することができます。ハザード解析では、組織や環境要因が時間とともに変化していて、真実ではないか、もはや真実ではないかもしれない組織内のシステムエンジニアリングに関する仮定の早期発見につながる先行指標を生成します。これらの先行指標も、懸念や目標がどれだけうまく対処されているかを識別するための方法を提供する必要があります。

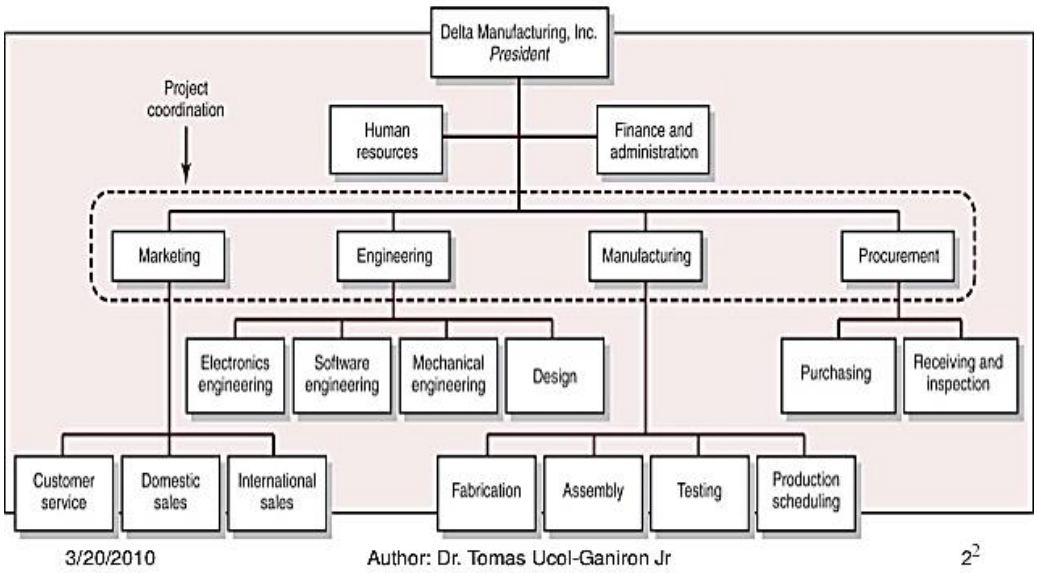
また、モデリングと解析のプロセスの過程で、人々は問題の解決策を大抵よく知っているのですが、日々のスケジュールや問題に圧倒されて、それらを実現する時間がないということを発見しました。また、階層コントロールストラクチャーモデルを使うことは、解決策生成グループ内のコミュニケーションに対する、また潜在的な再設計ソリューションの生成および評価に対して、支援を

することに計り知れない価値のあることを見出しました。議論や問題解決を導くための共通モデルを持つことは、それ自体、問題の解決策を見つけるのに非常に便利です。例えば、時には、任意の高度な分析もなく、何らかの実体が自分のコントロール（管理）アクションを導くために任意のフィードバックなしに別のものをコントロールしようとしていることを観察することができます。効果的なフィードバックを追加することで、目下の問題解決に向かう長い道のりを行くことになるかもしれません。

以下に、このハンドブックで以前に示した一般的なループ構造を示しますが、責任、権限、および説明責任の標準管理要求を補強しました。



以下にモデリング作業に起因するかもしれないシステムエンジニアリング構造の例を示します。



プロジェクトの調整は、任意の個人に割り当てられた責任のないまま、この例にある 4 個の同じレベルのコントローラ間で起こると単純に想定されていることに注意してください。会社の社長に適した責任はありそうにもないでしょう。

このモデルは、当然のことながら、責任の仕様、権限、および構造の各機能についての説明責任が含まれるでしょう。例えば、製品の開発会社での一般的なコントローラの割り当てられた責任には、次のものがあります。

人事：

トレーニング、雇用、労働力、能力開発

プロジェクト管理

プロジェクトの計画とコンセプト開発

技術的な意思決定

エンジニアリングの取り組み、プロジェクト内のコミュニケーションの統合

リスク管理と緊急時管理

マイルストーンとスケジュール管理

予算/リソース管理

下請けの管理

チームダイナミクス/通信

システムエンジニアリング

技術的な意思決定の実現

専門のエンジニア

コンポーネントデザインの技術的な決定、プロジェクトの要求の実現

事業開発

新しいビジネスへの導入

プロジェクトに入札

プログラム管理

プログラム/プロジェクトのリスクマネジメント

プロジェクトの進捗や成功、ワークフォース管理の監督

下請け関係、サプライチェーン管理

経営管理

テクノロジー/プロセス改善の実現

ビジネスリスクマネジメント

将来のスキルとエンジニアリングのニーズを予測

現在および将来のスキルや労働力の必要性を満たすことの確証

労働力「管理」

リサーチとエンジニアリングの改善

イノベーション、新技術/プロセスの改善

将来の事業動向や技術動向、スキルのニーズを明確化

品質保証とシステムの安全性

製品の品質を監督

製品の安全性を評価

次のステップとして適切なことは、（1）弱点を識別するとともに、コントロールストラクチャーの正確さを確保するための非公式のレビューセッション、（2）要求からコントロールストラクチャー責任までのトレーサビリティを用いたコントロールストラクチャーモデルの分析、及び（3）役立つところでの正式な STPA 分析のアプリケーションです。この目標は、どこで、どのように、（以前に識別した）要求それぞれが組織コントロールストラクチャーの中で実現されているのか、また、これらの要求に対するコントロールストラクチャー内の任意の弱点をすべて識別することです。以下に例を示します。

最初のステップは、現在の組織設計の専門家による審査を経てモデルを検証することです。査読者が、その構造の一部のみに精通しているかもしれないので、複数のレビューが必要になることがあります。

全体的なコントロールストラクチャーが確認されたら、次のステップは、インタビューで識別された特定の問題に対するコントロールストラクチャーを調べるだけでなく、意思決定を改善するために必要とされるかもしれない一般的な変更を識別するため、コントロールストラクチャー設計を分析することです。多くの場合、すぐに目立つ事項があります。例えば、ストラクチャーの複雑さ、あるいは複数の人が同じプロセスをコントロールする、同じ重複した責任を持つ、意思決定の適切または独立した監督の欠如、またはフィードバック経路の欠落（すなわち必要とされる情報の全てはないか不正確な情報で人々が意思決定をさせられる可能性があること）です。

さらに弱点を見つけるために、要求からコントロールストラクチャーへのトレースを行うことができます。以下の表で、上記に含まれる要求の結果になる可能性のある追跡結果の例を示します。

要求	コントロールストラクチャーコンポーネント	識別された弱み
リスク管理	プロジェクト管理 プログラム管理	• コストとスケジュールが不測の事態への対応には不十分なことが多い • リスク識別が不完全なことがたまにあり、重要なリスクが無視される
技術的独立性	プロジェクト管理 プログラム管理	• 技術的な判断がプログラムやビジネス上の戦略から独立していない、またはプロジェクトマネージャから独立していない
品質と安全	品質保証 プロジェクト管理 プログラム管理 エグゼクティブ管理	• 安全は保証中心であり、開発チームとの緊密な連携や彼らにより実施されない • 品質保証の多くは、”チェックリスト”によるもので、形式的である • ゲート(マイルストーン)のコントロールに品質と安全解析が含まれていない • プロジェクトマネージャは、品質と安全よりも、スケジュールや予算実績に、より高い価値を感じている • 生産性や支持性は重要視されない • 品質と安全に関しては、請負業者の監視は最小限である

識別された弱点は、要求の満足度を向上させるための再設計活動の手引きになるでしょう。何らかの改善事項には、何らかの構造的な変化が必要になります。例えば、明らかに、品質と安全要求に関して請負業者の不十分な監督に対しては、請負業者のマネージャへの単純な責任の割り当て、おそらく品質保証と安全性からのアシスタンスが必要です。その他の修正には、より広範にわたるストラクチャーの変化が必要な場合があります。例えば、このような請負業者の監督はどのように行われるべきかについてのポリシーが不十分または欠落している場合には、そのポリシーの作成と変更に努力を要します。

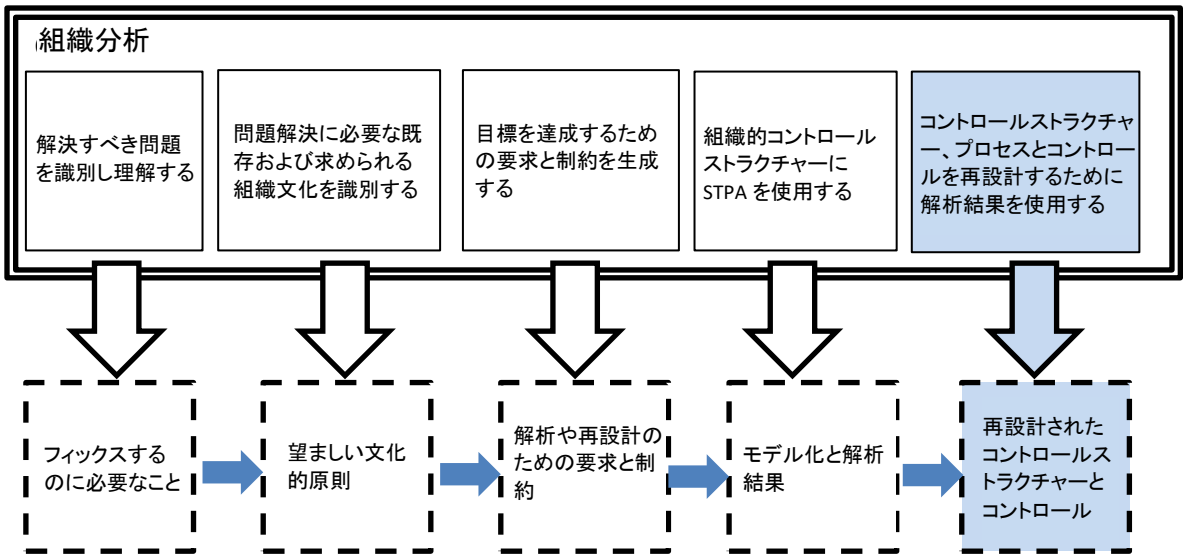
より複雑な変化の例として、技術的な独立性には、新たなコミュニケーションリンクと競合を解決するための意思決定プロセスだけではなく、独立した技術的権限[20]を設定するような新しい管理アプローチを要します。独立性は、時間の経過とともに弱くなる傾向があり、そういうわけで先行指標も、ここに適しているでしょう（先行指標についての次章を参照）。

最後の例は、エンジニアリングの意思決定から安全性を分離することは、設計の取り組みにほとんど影響を与えずに、それを事後保証やコンプライアンスの取り組みにさせる傾向があるということです。安全性や品質の問題を後になって見つけることは、修正に膨大な費用がかかり、基本エンジニアリングから安全性の責任を分離することは、かなり一般的に起こるが間違いです。相互作用を最適化するための最善の戦略を決定することは、責任、コミュニケーション、およびストラクチャーの大規模な変更が必要になります。例えば、プロジェクトエンジニアリングにおける安全工学コンポーネント、および品質保証における別のコンポーネントがあり、それぞれには非常に違った責任があり別々の活動を行なっています。

場合によっては、コントロールストラクチャー上またはその一部に対して、完全な STPA 因果関係解析を適用することが適切かつ有用であるかもしれません。例として、リスク管理が適切ではない明確な理由は明らかではないかもしれず、それはほとんどの企業でこの重要なタスクを実行することに力を入れているからです。実際には何がリスクなのか（欠点のあるメンタルモデル）についての情報の欠如、明瞭さが重視され、潜在的なリスクを重視しない文化、リスクアセスメントの不十分な訓練、不十分な手続き、重要なプロジェクトのリスクを軽視する経営からの圧力など、すべての潜在的な原因を識別することが役に立つかもしれません。すべての潜在的な原因を生成した後で、この組織で最も重要な原因や、それらを排除または制御するのに便利であるような変更は何かを調べることから研究を始めることができるかもしれません。いくつかの仮説が場当たり的に定式化できますが、STPA の段階的なプロセスを使用することは、完全性を改善し、重要な因果関係要因が失われる可能性を低減します。組織内にあるリスク管理コントロールアクションの各タイプの潜在的な不備には様々な理由があるかもしれません。例えば、ある種類のリスクを識別することに適切な重きを置かないということは、十分な不測事態プランを提供していないことより、別個の因果関係要因を含んでいるかもしれません。私たちは、リスクを識別するために STPA を使用するとは、経験豊富な従業員が行う通常のブレーンストーミングの過程より、はるかに効果的であることを見出しました。

[20]米国海軍の SUBSAFE プログラムの独立技術的権威は、過去 50 年以上にわたり、そのプログラムで扱われている事故を取り除くために大変有効でした。ナンシー・レブソンの「Engineering a Safer World 2012」の第 14 章、および安全管理システムの章でさらに詳細を読むことができます。

組織再設計プロセスの結果の使用



プロセスの最終ステップは、全体として、コントロールストラクチャーのための再設計およびリスク管理プロセスにおける解析結果を使用することです。コントロールの設計におけるリスク管理の一部分として、先行指標計画(program)が必要です。コントロールストラクチャーまたはその一部に STPA を適用すると、次章で説明するように、増加するリスクの先行指標を識別するのに役立てることができます。

第 6 章：STPA を用いた先行指標の識別[21]

ナンシー・レブソン

リスク管理は、すべてのプログラムの重要な一部であり、計画や組織のリスクは慎重に識別して管理する必要があります。通常、リスクアセスメントは、頻繁に任意の専門家のグループとともにプログラム内のリスクとして現れるものをブレーンストーミングし、リスクレベルをアサインするというアドホックな方法で行なわれます。STPA はリスク管理プログラムの基礎となる構造化されたプロセスを提供します。

先行指標がリスク管理のために使用される場合、先行指標は通常アドホックな方法で作成されます。このハンドブックの前半で説明したように、製品の安全性やその他の創発特性を考慮すると、より正式で構造化されたシステム思考プロセスを使用する方が効果的です。STPA を使用して先行指標を作成することは、得られた指標が直接的に危険と事故に追跡可能であるという利点にもなります。この章では、STPA の結果とシステム全体のエンジニアリングプロセスの結果を使用することを含む先行指標を特定するための構造化アプローチについて説明します。トピックは以下の通りです：

- 先行指標とは何か？
- 人は先行指標のために現在何を使うのか？
- 前提条件に基づく先行指標とはなにか？
- 方法
  - ✓ STAMP と STPA を用いて適切かつ効果的な先行指標を特定する方法
  - ✓ 先行指標の基礎となる前提条件を生成する方法
  - ✓ 前提条件に基づく先行指標計画を作成するために、前提条件を使う方法
  - ✓ リスク管理プログラムに先行指標を統合する方法
- 実現可能性と最終的な考察

先行指標とは何か？

先行指標は、事故を防ぐための措置がとれるように、事故が起きる前に可能性を識別するために使用されます。先行指標は、 ランダム、近位(proximal)イベントの一意の組合せによるものではないという前提条件に基づきます。代わりに、事故は以下のことから起こります。

セーフガードとコントロールが緩んでいくことが繰り返され、
組織はリスクが増大する状態へ変化していく
これは、複数の目標間やトレードオフ間の競合およびリスクの認識の低下によって生じる
その結果、更に危険な挙動を引き起こすことになる

[21] この章は、ハンドブックにある他の章のように、導入的な説明情報を入れた実用的な手引き書として執筆しています。リファレンスや技術的扱いなどの詳しい情報はナンシーレブソンの以下の書物を参考にしてください。A systems approach to risk management through leading safety indicators, *Reliability Engineering and System Safety*, Elsevier Publishers, 2014.

この前提条件は、重大な事故に向かう進行は時間をかけて起こり、それゆえ、この危険な経路を検出して介入する可能性があることを意味します。先行指標は、介入が必要であるというシグナルです。

組織では、先行指標は、組織の目標に対する長期的な影響が見えるようになる前に、製品、サービス、または組織行動の様相が軌道から外れ始める時期について、早期に兆候を提供することができます。

先行指標として何を使うのか？

石油化学業界では特に、一般的な先行指標の特定に多くの努力がなされてきました。この試みには、どの企業や業界でも旗を揚げることができるいくつかの一般的に適用可能な測定基準を見つけることが含まれます。メンテナンスのバックログ、マイナーインシデント、装置の故障率、従業員文化（信念）に関する調査結果などがその例です。後者は、すべてまたはほとんどの事故がオペレータによって引き起こされると想定されます。先行指標を特定しようとするこのような試みは、特に有効であるとは考えられていません。また、ほとんどがシステムの安全ではなく職場の安全に向けられています。

先行指標を識別するための他の試みは、インシデントやインシデント報告書を使用しています。1 つの欠点は、以前に起こったことだけが考慮され得ることです。これらの従来の要因の多くは、すでに排除または制御されています。別のアプローチは、ハザード解析を使用して将来のシナリオを予測することです。ここでの問題は、ハードウェア障害にほとんどすべての注意を集中させる、従来のハザード解析技術によって生成されるシナリオの限られた性質です。

事故の原因となる社会的、組織的、管理的要因は先行指標として有用であると仮定されてきましたが、そのような一般指標がどれほど有効であるかについての実際的なデータはごく少数（5 または 6）です。問題は、これらの仮説が、どのように、なぜ事故が発生するのかについてのモデルまたはフレームワークに基づいておらず、非常に不完全であるということです。さらに、これらの先行指標には、多数の組織で共有されている最も一般的な要素のみが含まれているため、特定の組織にとって最も重要な要素が省略される可能性があります。

この章で説明する代替策は、すべての事故の一般的な先行指標を特定するのではなく、特定の会社または特定の製品またはサービス内の安全管理の設計に基づいて先行指標を特定します。これらを前提に基づく先行指標と呼びます。

前提に基づく先行指標(Assumption-Based Leading Indicators)とは何か？

先行指標を特定するための基礎となる前提の考え方は、もともと、エンジニアリング外のリスク管理プログラムに提案され、使用されていました。RAND は、米軍のクライアントの中長期防衛計画を支援し、陸軍ミッションにおける不確実性を減らし、リスクを管理するために、1980 年代後半に前提に基づくプランニング assumption-based planning（ABP）の方法論を開発しました。

私たちはこの基本的な考え方をとり、エンジニアリングにおけるリスク管理のアプローチを作成しました。このアプローチは、特定の組織、製品、またはオペレーションの安全設計プロセスの基礎となる前提に基づいて、リスクの増加を示す有益な先行指標を特定できるという根拠に基づいています。すべてのエンジニアリングは、たとえそのシステムが組織的または管理的な構造であっても、運用システムの動作についての前提を伴います。そのような前提には、例えば、時間経過によるハードウェアコンポーネントの故障率、どのように製品が使用されるか、製品が使用される環境、またはサービスが提供される環境、使用するように指示されたツールに関する人々への基本的な訓練、意思決定のための情報ニーズと情報チャネルがいかに効率的に機能するかについての前提

です。市場全体が変化するにつれて変化する可能性のある顧客が望むと考えられるものから、外部的または環境的な前提が生じる可能性があります。

したがって、前提に基づく指標の正式な定義は、以下の通りです：

> *前提に基づく先行指標は、前提が壊れているか、または危険に弱いかを検出するためにプロセスを監視する際、または前提の正当性が変化しているときに使用できる警告標識として定義されている。*

先行指標プログラムの目標は、設計されたシステムの設計者が行った社会的、管理的、製品/サービスの前提を監視することであり、人や他のシステムコンポーネントの考え方、特定のコンテキストで動作するだけでなく、もともとは正しいものの、人々がローカルの目標を最適化したり、環境が変化したりするにつれて時間が経つにつれて不正確になることがあります。すべての組織やシステムに適用する先行指標を見つけようとするかわりに、先行指標は、その特定の設計に関する安全関連の前提に基づいて、特定の組織やシステム設計のために生成されます。

組織構造が設計された通りに機能しない理由は数多くあるかもしれないが、1 つの重要な原因は、組織の文化、すなわち組織の目標とその中の目標は、プログラムの目標を達成するために必要なものに合致しないことです[22]。

前提に基づく先行指標はどこから来るのでしょうか？最も基本的な意味で、システム目標の達成方法に関する前提を表しています。これは一般的に 3 つのタイプの前提が含まれます：

- 設計されたシステム（組織構造を含む）の初期設計で使用されたモデルおよび前提が正しかったということ。
- システムは、設計者が想定した方法で構築され、操作されること。
- 元のモデルや前提が、おそらくプロセスを改善または最適化する試みや環境の変化による、時間の経過とともに侵害されないこと。

この種の前提は、前提に基づく先行指標プログラムにおいて、前提がもはや真ではない(もしくは、真ではなかった)と判断された場合に、何をチェックすべきか、どのように、いつ、どのような行動をとるべきかを決定するために用いられます。そのようなプログラムは３つの側面を持ちます：

1. 適切かつ効果的な先行指標を特定する。
2. それらを使用する先行指標の監視プログラムを作成し、
3. 慎重に設計されたリスク管理プログラムの中に、この監視システムを組み込む。最初の 2 つの側面だけでは、検出だけでは十分ではない。先行指標が行動が必要であることが示されたときに行動する管理プロセスがなければならない。

これらの３つの側面のそれぞれについて説明します。

STAMP と STPA を使用して適切で効果的な先行指標を特定する方法

事故は、防ぐための多くの努力にもかかわらず、まだ起きます。理論的には、安全なシステムを設計する、すなわち、すべての危険を排除するか、適切に制御または緩和し、何も変わらない場合、事故を起こすべきではありません。問題は、実際には、以下の条件のいずれも通常は正しくありません。エンジニアリングプロセスは完璧でも人間の行動ではそうではありません。さらに、すべてのシステムとその環境は、時間の経過と共に変化する可能性がありません。より効果的な先行指標を求める際の出発点は、事故の一般的な原因を考慮することです。

[22] 安全マネージメントシステムの設計の章には、組織文化についての詳しい定義や議論があります。

事故原因の一般的な分類

事故の原因は、技術システムの開発、運用、および管理において発生する可能性があります。通常、これらのタイプの原因のいくつかまたはすべては、事故シナリオで見つけることができます。以下のリストは、これらの 3 つの領域のそれぞれに事故原因が発生する理由を示しています：

<u>設計と製造</u>

- 不適切なハザード解析：システムのハザードやそれを特定するためのプロセスについての前提が成り立たない。
    - HA は行われていないか、または完了していない。
    - いくつかのハザードは、ハザード解析プロセスやどのように行われるかが不十分であることにより識別されない。重要な原因は除外され、したがって処理されない。
    - ハザードは識別されているが、「十分に起こりそうもない」と考えられるため、取り扱われない。
- おそらく不適切なエンジニアリング知識や操作に関する不適切な前提のために、識別されたハザードのコントロールおよび緩和策の設計が不適切である。
- コントロールと緩和策の不十分な構成

<u>オペレーション</u>

- 設計者が操作中に存在すると想定していたコントロールは、実際に存在しないか、使用されていないか、もしくは有効ではないことが分かる。
- コントロールは存在し、使用されていてもともと有効だったが、時間の経過とともに変化すると、コントロールの元の設計の根底にある前提に反する。
    - 変化する状況に伴って発生する新しいハザードは、開発中には予想されなかったか、発生しそうにないとして却下された
    - 物理的コントロールと緩和策は、分析と設計プロセスでは考慮されていない方法で、時間の経過とともに低下する
    - コンポーネント（人間を含む）は、時間の経過とともに異なる挙動を示す（設計および解析中に行われた前提に違反する）
    - システム環境は時間の経過とともに変化する。（設計および解析中に行われた前提に違反する）

<u>マネジメント</u>

- 安全管理システムの設計に欠陥がある（このハンドブックでは、効果的な安全管理システムを設計する方法の章を参照）。
- 安全管理システムは効果的であるかもしれないが、それが設計された（そして仮定された）ように動作しない。誤動作には多くの理由があるかもしれないが、一つの一般的な原因は、安全文化、すなわち安全に関する組織の目標と価値は、時間の経過とともに低下し、または最初から無効であったということである。さらに、安全関連の決定を下す人々の行動は、競争的、財政的または他の圧力によって影響を受ける可能性がある。

事故を防ぐためには、これらの原因の排除または削減が必要です。 事故が発生する前に、先行指標プログラムを使用してそれらを検出することができます。 これらの 3 つのカテゴリの事故原因をそれぞれ見てみましょう。

<u>設計や製造</u>時に、使用されるハザード解析プロセスが不十分であるか、またはスキップされる可能性があります。 場合によっては、スケジュールの圧力がハザード解析プロセスをスキップするか、表面的にしか実行しないことがあります。今日の複雑なソフトウエア集約型システムでは、ハ

ザード解析技術が新しい原因を特定できないため、事故は頻繁に発生します。システムの開発と実装中に発生する事故原因に関連するこれらの要因の排除は、表面的には比較的単純に見えるかもしれないが、一部の組織内では事実上、政治的にはかなり困難な場合があります。このハンドブックの章には、STPA を組織に統合するための提案があるが、このハンドブックの現在の範囲を超える広範な議論があります。

非常に一般的な事故の原因は、技術的要因が開発中に特定されましたが、無視される可能性は十分に低いと計算されたため、除外されました。これに対する最善の解決策は、いくつかの危険またはその原因シナリオを無視することの正当性として、確率的リスクアセスメントまたは informal likelihood assessments を使用することではありません。この解決策は政治的に可能ではないかもしれないが、不可能である、または無視するほど無関心にされていると不正確な判断をされたイベントがいつ発生するのかを識別する先行指標を作成することができます。私が分析していた最近の化学プラントの爆発と火災では、1970 年代に触媒が不活性であると判断され、事故に寄与する可能性はゼロと判断されました。しかしながら、長年にわたり、触媒の組成および製造プロセスが変更されました。 事実、この同じ会社が所有する化学プラントでこの触媒に関連する 2 つの爆発がありました。 しかし、この工場のエンジニアが新しいプラントを設計したとき（2 回の爆発の後）、設計者は触媒が不活性であると仮定し、リスクアセスメントを無視しました。このプラントは後で、同じ触媒を使用することに関連する重大な事故を起こしました。 当初の前提は明らかに間違っていました。主要する 2 つの爆発のうちの最初の爆発の後に、先行指標は作動していたはずであり、爆発の発生は設計前提の変更につながったはずです。

別のケースとして、ハザード解析で原因を特定することができますが、コントロールおよび緩和策が適切に設計および実装されていない可能性があります。 一例は、物理的コントロールストラクチャーの設計が不十分であることです。 単純な計算や知識のエラーが含まれる可能性があるが、間違った前提も重要な役割を果たします。 冗長性を使用して故障から保護する場合の独立性に関する一般的なエンジニアリング設計の前提がその例です。Macondo（Deep Water Horizon）のブローアウト防止装置を考えてみます。潜在的なブローアウトを制御する手段には冗長性がありましたが、冗長ユニットには共通原因障害モードが含まれていました。効果のないブローアウト防止装置が過去にいくつかの重大な事故に寄与していたにもかかわらず、ブローアウト防止装置が失敗したことはないという信念の受容は、この業界で広まっていました。前提が間違っていたことを示す明確な先行指標は存在したが無視されました。チャレンジャーは、共通原因故障の失敗のもう一つの例です。今回は、致命的なチャレンジャーの飛行の何年も前に、O リングの独立性についての仮説がチェックされて科学的に無効とされました。しかし、この変更は、打ち上げの決定が下された Marshall Space Center データベースでは決して記録されませんでした。共通モード/原因故障を伴う非常に多数の事故があると、多くの事故に寄与した他のものと同様に再考することが重要なエンジニアリング設計上の前提であるように思われます。この場合、欠陥のある安全情報システムも関与しています。

オペレーション中に使用されることが想定されるハザードコントロールは、おそらく他のコントロールや誘因との競合が識別されていないために、設計が不適切で効果がない場合があります。また、適切に実装または使用されていない可能性もあります。前述したように、システムの安全性を支える運用上の仮定に違反する可能性は、一連の先行指標に反映させる必要があります。たとえば、運用者は、より効率的に動作するために、ショートカットを取ったり、安全装置をオフにしたりすることがあります。 または航空交通管制システムにおいて、空域は、システム設計中に最初に考慮されたものよりも混雑する可能性があります。 ここでチェックする必要のある前提は、文書化して運用者に渡す必要があります。

最後に、安全管理システム（第 6 章を参照）の設計と運用に関わる前提が真ではない、または真にならない可能性があります。産業や組織に属する人々の目標、すなわち安全文化は、誤ったことが事故の主要要因となり、先行指標のセットに反映されなければならないという重要な前提です。

例えば、安全方針は、あらゆる企業や組織が、個人の期待する安全文化と行動を伝達するための基本的な要求です。そのポリシーがどの程度うまく守られているか、遵守率が時間の経過とともに低下しているかを測定する方法が必要です。 事故が発生した後も、経営行動や意思決定の仮定が破られていることが一般的に判明しており、これらの前提を監視する必要があります。

先行指標の基礎となる前提を生成する方法

STPA を使用して、適切かつ効果的な先行指標は少なくとも部分的に特定することができます。目標は、階層コントロールストラクチャーモデルと組織に組み込まれたコントロールを使用して、創発特性に関する制約の基礎となる前提を特定し、システム目標を達成するように設計されたコントロールの効果を弱める方法を明らかにすることです。このセクションで説明するように、システム設計プロセス中に追加の仮定を特定することができます。

前提がどこから来るかのリストは下のボックスに表示されています。

前提は以下から生成される:

- コンセプト開発中に生成される、特に量を含む上位レベルのシステム目標
- システム目標から生成されたシステムレベルの要求
- システムが動作する外部環境に関する前提
- 安全関連の環境要求と制約（システムの使用上の制約を含む）によって課されるシステムの動作要求
- STPA 生成のハザード、階層コントロールストラクチャー、非安全なコントロールアクション、および因果関係シナリオ
- 因果関係シナリオを管理するために工夫された設計機能
- システム設計機能によって完全には処理できない因果関係シナリオを管理するために作成された操作要求
- 運用コントロールを含む安全関連コントロールの設計における制限事項
    - 機能要求の達成に関連する制限事項
    - 環境の前提条件に関連する制限事項
    - 設計または運用手順において完全に排除または制御できなかったハザードおよび原因要素に関連する制限事項
    - システム設計時のトレードオフに起因する制限事項

TCAS（Traffic Alert and Collision Avoidance System）と呼ばれる航空機衝突システムの例を用いて、先行指標がベースとなる致命的な前提の生成を説明します。私は学生と共に 1990 年代初頭に TCAS の公式認定を支援し、後に TCAS intent specification[23]を作成しました。

Intent specification の重要な部分は、システムの安全性の基礎となる仮定の文書化です。ここでの例はその仕様に由来し、ほとんどすべてが TCAS 認証に使用される定性的ハザード解析[24]によって生

[23] *Intent specification* は、システム論、システムエンジニアリングの原則、人間の問題解決とその強化についての心理学的研究に基づいています。目的は、複雑なシステムの複雑さを明らかにする人の処理を助けることです。*Engineering a Safer World* の第 10 章に詳しい情報があります。

[24] 人間とソフトウエアの動作を詳細に入れた、大規模なフォールトツリー解析が TCAS 開発と認証で使われました。目的は量より完成であり、数量化するフォールトツリーボックスを制限しませんでした。結果のフォールトツリーは安全エンジニアリングで 35 年見た中で最も完全なものでした。STPA で見つけたシナリオ全てを入れませんでしたが、これはまた、目的がシナリオの確率を作ることにある時に起こる数量化は可能でないシナリオを除外しませんでした。

成されたシナリオに関連しています。前提のいくつかは、ハザード解析のシナリオからではなく、TCAS が動作すると予想される設計目標および環境からきています。STPA は、TCAS が設計され認定された時点では存在しなかったが、STPA は、使用されたフォルトツリーによって生成されたすべての危険なシナリオと、フォルトツリー解析から省略された多くのシナリオを生成することが示されています。従って以下のすべての前提例は STPA 結果から生成することができます。

前提の生成は、早期概念開発中に開始されるべきです。最高水準のシステムの目標と要求に数値が含まれている場合、手がかりが見つかることがあるが、それは必須ではありません。

TCAS では、TCAS のシステム目標は以下の 2 つです：

***G1**: 広範囲の National Airspace System（NAS）ユーザーに、手ごろで互換性のある衝突回避システムオプションを提供する。*

***G2**: ATC の第 1 次レーダ・システムまたは第 2 次レーダ・システムによってカバーされていない空域を含む航行可能な空域全体、および地上設備がない場合等のすべての気象条件で他の航空機との潜在的な空中衝突を検出する。*

システムの目標は、システム要求を作成するために使われます。 これらの要求の根底にある前提は、同時に生成することができます。たとえば、

***1.18**: TCAS は、任意の 2 つの航空機が水平方向に 1200 ノット、垂直方向が 10,000 フィート/分までの間、衝突回避保護を提供しなければならない。[G1]*

<u>前提</u>：この要求は、商業用航空機が垂直上りまたは制御された降下中に 600 ノットおよび 5000 フィート/分で動作することができ、したがって、2 つの飛行機が水平方向に 1200 ノットまで、垂直方向が 10,000 fpm まで接近することができるという前提に由来する。

この前提は、技術的な変更がそれに反していないことを確実にするためにチェックする必要があるものの一例であり、それ(意思または他の仕様におけるトレーサビリティポインタによって識別される)に基づいた技術設計の決定を脆弱にします。

別の例のシステム要求は：

***1.19.1**: 1.19.1:TCAS は、往路と終着区間で、平方海域あたり最大 0.3 機の交通密度で運航しなければならない（すなわち、5nmi 内 24 航空機）[G2]。*

<u>前提</u>：交通密度は 1990 年までにこのレベルまで上昇する可能性があり、これは今後 20 年間で最大の密度になる。

ここでもまた、将来の航空機の性能限界が変化するか、垂直方向の離隔距離の減少または航空交通管制の非常に異なる方法など、空域管理に大きな変化が生じる可能性があります。TCAS の多くの計算は要求 1.19.1 の前提に基づいており、変更された場合には安全パラメータの再評価を作動するために監視する必要があります。

意思決定を説明するため、または設計の基礎となる基本情報を記録するために、別のタイプの前提を特定（および指定）することができます。例えば、設計は、システムが動作する環境に関する前提に基づくことができます。 TCAS の例には：

***EA1**: 航空機間に高い整合性の通信が存在する*

***EA2**: TCAS 装備航空機には、モード-S 航空交通管制トランスポンダが搭載されている[25]*

***EA3**: すべての航空機がトランスポンダを作動させている*

***EA4**: すべての航空機は、法的識別番号を持っている*

***EA5**: 最低 100 フィートの精度で侵入対象から高度情報を入手できる*

[25] 航空機のトランスポンダは管制間隔を維持する航空管制を支援する情報を送ります。

***EA6**: 航空機の気圧高度をTCAS 装置に提供する高度計システムは、RTCA 標準の要求を満たすであろう…*

***EA7**: 脅威となる航空機は、TCAS のエスケープ操作を妨害する突然の操縦をしない。*

制御された空域に統合された新しい技術と新しいタイプの航空機は、これらの前提に違反する可能性があります。 EA4 は、非技術的な前提の例です。 識別番号は、通常、各国の航空機関によって提供されます。 その前提は、国際協定によって保証され、国際機関によって監視される必要があります。

航空機が運行中のトランスポンダを持っているという前提例（EA3）は、特定の国の空域規則によって強制され、あるグループによって保証されなければなりません。 この前提の真実は、TCAS が動作中のトランスポンダなしで航空機を表示せず、解決勧告（RA）[26]を提供しないために重要です。EA7 は、パイロットと航空管制システムの動作に関する前提の一例であり、無人またはその他の種類の新しい航空機を空域に導入することによっても違反される可能性があります。

いくつかの仮定は、環境要求と制約によってシステムに課されることがあります。これらの仮定は、新しいシステムの使用に制限をもたらすかもしれない(これは前提チェックを必要すると)し、システムの安全性や他の分析が、安全性を確保するために作成されたシステムまたはより包括的なシステムに課されなければならない制約を決定する必要性を示しているかもしれません。TCAS の例は：

***E1**: 非TCAS 装置とTCAS の動作または相互作用は、TCAS 装置の性能またはTCAS が相互作用する装置の性能を低下させてはならない*

***E2**: 航空機環境警報の中で、優先度は以下の通りでなければいけない：ウィンドシアが第1 優先であり、次にGround Proximity Warning System（GPWS）、そしてTCAS*

***E3**: TCAS アラートおよび勧告は、マスタ警告および警告システムを使用するアラートおよびアドバイザリから独立していなければならない。*

これらの仮定は、航空機に大きな設計変更が導入された場合にのみ確認する必要があります。

STPA は、システムハザードの仕様から始めて、モデリングと分析プロセス全体を通して進んで、仮定と先行指標プロセスに重要なインプットを提供します。TCAS の例として、システムのハザードは次のように識別されました：

***H1**: TCAS は、最小分離基準に違反する制御された航空機のペアとして定義される、近距離衝突（NMAC）を引き起こすか、またはこれに寄与する。*

***H2**: TCAS は、固定された構造物または自然の地形に近付きすぎる航空機の原因となるか、それに寄与する。*

***H3**: TCAS は、パイロットが航空機に対して制御を失う原因となるか、それに寄与する。*

***H4**: TCAS は他の安全関連航空機システム（例えば、地上近接警報）と干渉する*

***H5**: TCAS は、地上ベースの航空交通管制システム（例えば、地上またはレーダーまたは無線サービスへのトランスポンダ伝送）と干渉する。*

***H6**: TCAS は安全関連のATC 勧告（例えば、限られた地域や悪天候を避ける）に干渉する*

セーフティクリティカル仮定の最初の基本セットは、適切に設計され、運用されているシステムではハザードが発生しないということです。これらのハザードのいずれかが発生した場合（たとえ事故が発生しなくても）、安全技術プロセス（この場合は TCAS ハザードを排除または緩和するために使用されるプロセス）の完全なレビューが開始されるはずです。しかし、ハザードが発生した後に前提を確認することは遅すぎる可能性がありますが、ハザードの識別は、STPA を使用してハザー

[26] 解決勧告は、基本的に侵入者を避けようとする作戦行動です。

ドにつながる可能性のあるシナリオを特定することによって、より早期のチェックを導き出す出発点となります。

このような高いレベルであっても、（将来的には変化する可能性のある）地上ベースの航空交通管制システムが存在し、TCAS がその運用に干渉しないなど、追加の前提が導き出される可能性があります。ハザードはほとんど変化しないが、システムに変更が加えられたときに新しいものが導入され、それらを処理するプロセスが損なわれる可能性があります。

ハザードの発生と非安全なコントロール行為のチェックは、ハザード解析プロセス自体の妥当性に関する重要な情報も提供します。 ハザード解析と安全工学の目的は、ハザードを特定し、そのハザードを排除または防止することです。 防止できない場合は、緩和する必要があります。エンジニアが考えたハザードは除去されるか、防止され、当然のことながら、発生することはありません。彼らが行う、このイベントは、工学的プロセスの欠陥、またはパイロットや航空交通管制官の行動に関する前提など、おそらく運用システムに関する前提の欠陥を示すものです。技術的なプロセスを修正するだけでは不十分です。危険な動作を起こすことができる開発プロセスの穴も修正する必要があります。

理想的には、実際のハザードの状態が発生する前に、エンジニアリングプラクティスまたは動作上の動作の仮定の欠陥が先行指標によって特定されます。 この目標は、STPA で特定されたハザードのシナリオの基礎となる前提を特定することで達成できます。階層的なコントロール構造の設計、UCA、および因果関係シナリオは有用です。

コントロールストラクチャー 自体には、先行指標を作成するために使用できる前提が含まれています。2002 年に Uberlingen Germany を中心とした空中衝突により、コントロール構造の安全性に関する操作の仮定が明らかになりました。潜在的な NMAC へのパイロットの対応に対して潜在的な責任を負う 3 つのグループ、TCAS、地上航空管制官、航空運行センターが参加しました。後者は、TCAS アラートに応答し、パイロットを訓練するための航空会社手続きを提供します。他の 2 つのグループの関連性は明らかです。 明らかに、航空管制システムの全体的な設計と運用において、これら 3 つのコントローラ間の潜在的な競合や調整の問題を解決する必要があります。 TCAS の場合、TCAS によって提供された RA は、矛盾する勧告の場合には必ず従わなければならないと想定されていました。 設計者は、現時点では、実用的な方法がなかったため、乗組員に発行された可能性のある TCAS 勧告についての情報を地上コントローラにダウンリンクするため、パイロットが TCAS 勧告を直ちに実施すること、副操縦士はラジオによる TCAS アラート情報を地上 ATC に送信することを決定し、よって地上航空管制官が空域の状態および勧告を受けることを知ることができます。航空会社は、このプロトコルを実施するための適切な手順と訓練を提供します。

競合する勧告がどのように扱われるかについてのいくつかの重要な前提が Uberlinger の空中衝突で違反されました。例えば、地上 ATC 塔には 2 つの航空管制官が設置されていたはずであり、地上 ATC システムと TCAS が提供する勧告との間に矛盾がある場合、パイロットは TCAS 操縦に従うことになっており、そのような矛盾する状況では訓練を受けたパイロットは TCAS の警告に従うよう航空会社は想定されています。これらの前提のうち最初のものは、悲劇の時に 2 機の航空機を取り扱っていたスイスの航空管制センターで、しばらくの間、違反されていました。その情報が決してチェックされなかったので、第 2 のものが以前から違反していたかどうかは不明です。第 3 の前提、すなわち、関係する航空会社が、矛盾する勧告が提示されたときに常に TCAS に従うように操縦士を訓練していたことは、長い間真実ではなかったが、そのような訓練が行われていることを保証する責任を誰にも与えられなかったか、またはその責任を行使していませんでした。チェックされていれば、コントロール構造の操作に関するこれらの誤った前提は、設計されたコントロール構造が劣化していることを示す先行指標として役立つ可能性があります。

非安全なコントロールアクション、安全上の制約、および因果関係シナリオは、致命的な前提についての情報も提供します。因果関係シナリオはコントロールを設計するために使用され、コント

ロールが作成されるという前提を形成します。たとえば、H5 では、次のシステム安全性の制約が生じます：

***SC.2**: TCAS は、地上 ATC システムまたは地上 ATC システムへの他の航空機の送信を妨害してはならない（H5）。*

STPA は、SC.2 の違反の原因となるシナリオを特定するために使用できます。この情報は、その後、より詳細な安全制約 SC2.1 に洗練されます：

***SC2.1** システム設計は、地上ベースの二次監視レーダー、測距装置チャネル、および 1030/1090 MHz 周波数帯で動作する他の無線サービスと干渉してはならない（2.5.1）。*

TCAS の安全な設計の根底にある前提は、そのような干渉が決して起こらないということです。 もし起これば、これはシステムの設計または動作に欠陥があることを示す先行指標です。

人間は時間の経過とともに動作を変え、設計者が意図した方法とは異なる方法でオートメーションを使用する傾向があります。したがって、オペレータの行動に関する前提は、先行指標を特定するための別の重要な情報源を提供します。 例えば、H3 は、TCAS は、パイロットが航空機に対して制御を失う原因となるか、それに寄与するということです。H3 の STPA によって導出された安全制約 SC.6 は次のように述べています：

***SC.6**: TCAS は、飛行の重要な段階でパイロットおよび ATC の操作を中断してはならないし、航空機の操作を中断してはならない（H3,2.2.3,2.19,2.224.2）*

この安全性制約が導出された関連ハザード（この場合は H3）を特定することに加えて、ここでの仕様は、そのハザードを制御するために、すなわち SC.6 を強化するために使用される設計の特徴（TCAS の意図仕様の 2.2.3,2.19、および 2.24.2）を指しています。これらのコントロールには、チェックする必要がある重要な前提も含まれています。最も基本的な前提は、これらのコントロールが危険なシナリオを防止する上で有効であり、それらが正しく実装されているということです。例えば、STPA 分析では、SC.6 の違反につながる可能性のあるシナリオの 1 つは、パイロットが地上または離陸中に TCAS は、気が散る解決策勧告を提供することです。操縦士が離着陸の重要なフェーズで潜在的に注意散漫になるような解決策勧告の生成を禁止することを可能にするこのシナリオを防止するためのコントロールが設計されました：

***SC6.1** TCAS 搭載航空機のパイロットは、解決策勧告の表示は禁止されているが交通勧告が表示される Traffic-Advisory-Only モードのみに切り替えるオプションが必要である (2.2.3)。*

<u>前提</u>：この機能は、離陸中または平行滑走路への最終的なアプローチでのみ使用される、これは 2 つの航空機が互いに近づくと予測され、TCAS は回避操縦を要求するときである。

コントロールの追加、すなわちパイロットが Traffic-Advisory-Only モードに切り替えることによって TCAS 解決策勧告を禁止する能力は、パイロット手順、訓練等によってコントロールされなければならない別の危険なシナリオを作り出し、パイロットが SC6.1 に関連する前提に違反していないことを確認するために、システムの運用中にチェックすべき別の前提が導かれます。

危険なシナリオを排除またはコントロールするために作成された運用要求のその他の例は：

***OP4**: 脅威が解決された後、パイロットは以前に割り当てられた飛行経路に迅速かつスムーズに復帰しなければならない。*

***OP9**: パイロットは、Traffic Advisory のみに基づいて操縦してはならない*

これらの手順は、ハザードにつながると識別された特定のシナリオに対処するために作成されたものであるため、事故につながる可能性のある危険な動作を特定するためにチェックする必要があ

ります。前提を侵害するこの情報は、FOQA[27]システムを使用して自動的に収集される可能性があることに注意が必要です。

別の例として、Uberlingen の事故では、以前に言及されていない追加の因果要因がありました。1 つは、衝突時に ATC 装置のメンテナンスが実施されていたことで、航空管制官の聴覚衝突警報が機能していないことを知らずに無効にしました。航空交通管制官は警報が無効であることを知っていたら、彼は行動を調整した可能性があります。この種の因果要因は、操作手順、この場合、ATC タワーが依然として動作している間にメンテナンスを実行する手順でコントロールすることができます。もちろん、追加の重要な前提は、そのような手順が守られていることであり、この前提もチェックする必要があります。

先行指標を特定するために使用できる仮定の最終的な情報源は、安全関連コントロールの設計上の制限です。これらの制限は、システムを導入するかどうか、どのように導入すべきかについての決定において重要な情報を表すため、文書化する必要があります。いくつかの TCAS の制限事項は、基本的な機能要求に関連しています。例えば：

> ***L2**: TCAS は現在、水平脱出操縦を指示しないため、水平方向の分離を増加させない（そして意図していない）*

その他の制限事項は、環境の仮定に関連しており、例えば：

> ***L1**. TCAS は、トランスポンダを持たない航空機または非運用のトランスポンダに対する保護を提供しない **(EA3).***
>
> ***L6**: 航空機の性能制限は、フライトクルーが解決勧告に応答して安全に実行することができる脱出操縦の大きさを制限する。これらの制限により、衝突の解決策が妨げられる可能性がある **(H3, 2.38, 2.39)***
>
> ***L4**: TCAS は、脅迫されている航空機の報告された高度の精度に依存する。分離保証は、侵入した航空機のトランスポンダによって報告される侵入者による圧力高度の誤差によって低下する可能性がある **(EA5)***
>
> *<u>仮定</u>：この制限は、多くの航空機が GPS ではなく圧力高度計を使用する最初の TCAS 配備時に存在していた空域に適用される。より多くの航空機が現在の圧力高度計よりも高い精度で GPS システムを設置するにつれて、この制限は減少または排除される。*

L1 に関連する前提の例は、トランスポンダを持たない航空機の操作が、運用において排除されることです。

制限は、設計において完全に排除またはコントロールすることができないハザードまたはハザード要因に関連している可能性があります：したがって、それらは容認されたリスクを表します。

> ***L3**: TCAS は、電源投入時またはコンフリクト中に解決策勧告を出すことができる場合には、勧告を出すことはない*[28]

ここでの暗黙の前提は、例外的な状況を除いて、パイロットが離陸前に TCAS をオンにすることで、性能監査でチェックすることができるということです。

最後に、制限はシステム設計中に遭遇する問題やトレードオフに関連している可能性があります。たとえば、TCAS には、TCAS が正しく機能しているかどうかを判断するために、システム設計

---

[27] FOQA とは運行業務品質保証（Flight Operations Quality Assurance）の意味で、フライトデータモニタリング (FDM)とも呼ばれています。フライトを安全かつ有効に改善するために機上でデータを収集しモニタするプログラムを表します。名前は違いますが他の業界でも同様なモニタプログラムがあります。

[28] この制限の背景にある論拠は、ハンドブックの範囲を超えますが、ハザード解析により生成されたシナリオに起因します。

にセルフテスト機能を組み込むための高水準のパフォーマンス監視要求があります。この自己テスト機能に関連するシステム制限は次のとおりです：

> ***L9**: 飛行中にパイロットが自己試験機能を使用すると、追跡対象の数に応じて最大 20 秒間 TCAS の動作が抑止される。ATC トランスポンダは、自己テストシーケンスの一部の間は機能しない。*

安全性に関連した前提は、この動作はまれであり、したがって頻繁な TCAS の非作動の結果ではなく、したがって NMAC のリスクが増加するでしょう。

前提に基づく先行指標プログラムを作成するための前提の使用

前のセクションでは、前提に基づく先行指標を生成する方法について説明しました。システム設計中の前提に対するコントロールを実施し、システムの動作中に先行指標を検査することによって、前提を維持しなければなりません。どのようにして安全性の基礎となる特定された前提が強制されるのでしょうか？いくつかの新しい用語は、さまざまなケースの差別化に役立ちます。

第 1 に、元のシステム設計では、仮定の違反を防ぐために、シェイピング*(shaping: 適合)アクション*と呼ばれる前提を維持するためのアクションが取られる可能性があります。具体的には、前提を維持し、危険を防ぎ、より高いリスクの状態への移行をコントロールするために使用されます。シェイピング・アクションは、創発特性に対してフィードフォワード制御[29]を提供し、階層コントロールストラクチャーに組み込まれています。

例には、危険な状態に入るのを防ぐための物理的インターロックが含まれます。腐食を防ぐための乾燥剤の使用、人間の操作を簡単にして省略できないように設計すること、 安全クリティカルな意思決定のための独立した技術機関のような特別な管理体制の構築などです。

ヘッジング（hedging: *損失防止） アクション*は、前提が機能せず、それに応じて行動する可能性を準備するために使用されます。STAMP の用語では、ヘッジングアクションはフィードバックを使用してシステムの創発特性をコントロールします。本質的には、システムの動作を監視し、システムコントロールの基礎となる前提で特定された特定の変更に対応するための計画を含みます。 そのような監視には、シェイピングアクションの有効性の監視が含まれます。一例は、場合によっては前提の違反を解消することができないようにシェイピングアクションの可能性を予期するフェールセーフ設計（例えば、保護およびシャットダウンシステム）です。

シェイピング(shaping)とヘッジング(hedging)アクションを生成 するための情報は STPA の因果的シナリオおよびこれらのシナリオから生成された前提から得られます。

前提を確認することは、継続して行う必要はありません。いくつかの前提は、システムまたは環境における特定の変更がある場合にのみ、違反する可能性があります。*標識*は、現在のコントロールの変更（適合と損失防止アクション）が必要または推奨される、将来が見えてくるポイントです。標識は、コントロールの継続的な有効性に関する前提を含む、仮定のチェックを引き起こす時点または特定の将来の事象または変更点です。変更ポリシーの管理には、通常、さまざまなタイプの標識が含まれます。一例として、分離の減少または交通密度の増加などの空域の変化は、TCAS の標識として特定される可能性があります。

シェイピングとヘッジングアクションは、技術的および組織的な両方の安全コントロールストラクチャーに設計されています。標識は、システム設計および開発プロセス中に識別され、特定の応答が作成され、指定されます。システム動作中のより一般的な*前提チェック*は、システム設計の根底にある前提が依然として有効かどうかを判断するための特定のチェックを含みます。前提チェックでは、リスク管理者とコントローラは、運用中にシステムを監視して、元の前提が依然として有

[29] 初歩的なエンジニアリングの用語にあまり馴染みがなくエンジニアとしての訓練をしていない読者は、付録 F での定義を見てください。

効かどうかを判断します。そのようなモニタリングには、シェイピングとヘッジングアクションによって適切に、あるいは完全には処理されていないとされているもしくは、シェイピングとヘッジングアクションによってまったく実施されていない標識や前提の変更や失敗のモニタリングが含まれます。チェックには、パフォーマンス監査、調査、FOQA やその他のデータの自動収集が含まれます。

前提に基づく先行指標を実施するためのいくつかの方法

- 前提の違反を防ぐためのシェイピング(Shaping)アクション
- 前提の不履行に備えるためのヘッジング(Hedging)アクション
- 運用中の前提チェック
  - 特定のチェックを開始する標識
  - システム稼働中のチェック（定期または継続）
    - パフォーマンス監査
    - 調査
    - 自動的に収集されたデータ

前章の組織分析の章では、この例の分析の結論の 1 つは、技術の自立が組織の望ましい目標であったということでした。このような独立した技術的権限を設定するには、新しいコントロールのフィードバックチャネルだけでなく、シェイピングアクションも設計する必要があります。さらに、プロジェクトマネージャーが独立した権限を無視し、賢明でない決断を下す場合に備えて、ヘッジングアクションが確立される可能性があります。 第 3 の可能性、そして共通しているのは、独立が最初にうまくいくということですが、一連の小さな決定や経時的な状況の変化によってゆっくりと崩壊します。この後者の劣化のケースは、コロンビアのスペースシャトル・喪失の後に設立された NASA 独立技術局の GAO によって最近文書化されています。ニアミスまたは更に悪いことが生じるまで待たずに、 これらの問題を早期に発見するために、実現可能であれば継続的に、あるいはそうでない場合には定期的に、独立した意思決定構造の悪化によりシステムがより高いリスク状態に漂っている時期を特定するために独立した機関（この場合）の設立の根底にある前提は文書化され、チェックされる必要があります。インシデントや事故の調査は、もちろん失敗した前提が特定できるもう一つの場ですが、失われた後にそうするのでは遅いのです。

STPA および他の手段によって特定された各先行指標は以下によって文書化されるべきである：

- 関連する前提
- どのようにチェックするか
- いつチェックされるか
- 先行指標が真（想定違反）の場合に取るべき行動

この文書の実際の構造は、組織とそのリスク管理システムによって異なります。

リスク管理プログラムへの先行指標の統合

先行指標を特定するには、先行指標で行動が必要であることを示すときに行動することができるリスク管理構造の設計が伴わなければならなりません。先行指標はリスク管理の基礎を提供します。 全体的なリスク管理プロセスには、新しいシステム設計でリスクが最初に処理されるだけでなく、1 つまたは複数の前提および制約の先行指標が特定され、アクションが実行されることが保証されます。

先行指標を用いたリスク管理システムの設計には、一般的な管理原則が適用されるが、いくつかの具体的な機能が特に重要です。 たとえ情報が明確で、システムのリスク上昇について利用可能で

あっても、多くの組織は依然としてこれらの信号を処理して応答するのが遅いのです。先行指標の作成とモニタリングは、その出現が適切な措置を講じる結果とならない場合には役立ちません。あまりにも頻繁にヒューリスティックな偏見が介入し、特に防御的回避[30]と明確な先行指標はあまりにも遅くまで無視されて、重大な悪影響を避けることができなくなります。心理的偏見に加えて、組織文化や政治は、先行指標の設計や反応に問題を引き起こす可能性があります。

先行指標の作成と使用が成功するには、これらの心理的および文化的偏見をコントロールする方法が必要となります。例えば、効果的な行動を促すためには、先行指標を総合的なリスク管理プログラムに慎重に組み込む必要があります。 彼らは適切な意思決定者に伝達されるだけでなく、重要なシナリオの詳細な行動計画が策定され、それらの行動計画を実施するためのトリガーが特定されなければなりません。 先行指標の存在を確認し、問題が発見された場合には、最後までやり抜くように、その責任を割り当てる必要があります。

前のセクションで示したように、すべての先行指標はいつ、どのようにチェックされるかを示し、関連するアクションを持っていなければなりません。拒否と予防動作を減らし、組織や文化の盲目者に打ち勝つために前提が無効であることが判明する前に、必要な行動計画を作成する必要があります。モニタリングと行動の責任は、独立した組織に割り当てる必要があるかもしれません。プログラムやプロジェクト管理者、そして矛盾する圧力の下に置かないようにする必要があります。定期的に先行指標のリストを再考し、必要に応じて更新する必要があります。経験に照らして現在の指標を時間の経過とともに再評価し、特定された有効性の欠如を診断する継続的な改善プロセスを作成すべきです。例えば、先行指標によって示されなかった否定的結果が生じた場合、先行指標が問題を特定できなかった理由を分析し、もし特定できていたなら、なぜ効果的な行動が取られなかったのかを分析すべきです。その後、この情報を使用して、先行指標プログラム全体を評価し改善する必要があります。先行指標は特定されたが、チェックされなかったのか？ あるいは、適切なものが特定されなかったのか？ これらの質問に対する答えは、先行指標自体を改善するために使用されるだけでなく、省略または不適当な検査の理由が決定されるべきです。その理由は、プロセスの欠陥を伴い、チェックされていない他の重要なプログラム上の前提を特定することにつながる可能性があります。

実現可能性と最終的な考察

このすべてがうまくいくと仮定して、先行指標の作成と検証の実行可能性の問題は依然として存在します。STPA プロセスに起因する潜在的な先行指標は非常に多数ありますが、これらのすべてをチェックしたり、頻繁にチェックする必要はありません。識別された因果要因の大部分をコントロールするために、通常、ヘッジング(hedging)および不測事態(contingency)のアクションが安全コントロールストラクチャーに組み込まれています。多くの前提は適切な設計によって実施することができ、ハザード解析の結果を処理する上で最優先事項であり、STPA をシステムエンジニアリングに統合する章で説明されているように、初期のコンセプトおよび設計段階でハザード解析が行われている場合に最も実現可能です。

標識は、特定のトリガイベントの発生にチェックを制限します。重要な前提の文書化は、システム開発および仕様活動の一部であるべきです。前提条件を文書化することは、先行指標を特定して安全性を向上させることだけでなく、システムの通常の維持および進化にとっても重要です。最初の開発中であっても、文書化されていない限り、なぜ決定を下したのかを忘れてしまいます。TCAS

[30] 心理学者は、リスクの評価時に人が示すバイアスについて広範囲にわたる論文を書いています。これらのバイアスは先行指標を設計する方法や出来事への反応の仕方に影響を与えます。すべてのバイアスは重要ですが、先行指標の対応する最も重要な用語は「防御的回避」と呼ばれます。このバイアスは、先行指標を正確に受け取ることを軽視することや、真剣に受け止めてリスクが増大していると受け入れる能力に影響します。防御的回避は、ストレスがかかったり急を要する目的と競合するテーマを考えることを合理化し回避する、よくある心理的傾向に基づいています。

のような実際の問題に関する先行指標にこのアプローチを使用した場合、開発とテストの時間を短縮し、プロセスにハザードを導入することなくシステムを維持するためには、重要な前提の文書化が重要であることが分かりました。私は、数ヶ月の間に自分の「余暇時間」に、安全上重要な（およびその他の）前提だけを含む TCAS の Intent specification を作成したということを言及します。

非常に多くの前提が存在するのでしょうか？もちろんそうです。安全性において多数の因果要因を扱うための通常のアプローチは、確率を使用してそれらを減らそうとすることです。問題は、将来の事象[31]の確率を正確に予測する方法がないことです。多くの場合、数字と比較して、特にその数字がコンピュータによって生成される場合には、人間（エンジニアリングとマネジメントの両方）の専門知識と知識の使用は過小評価されます。しかし、数字はより正確に見えるかもしれませんが、ほとんどの場合、実際の少しのデータに基づいた粗暴の推測に過ぎません。心理学者は、実際には、人間は将来の出来事の確率を見積もることに非常に悪いことを発見しました。確かに、先行指標に確率や可能性を割り当てることは自由ですが、エンジニアリング、マネジメント、心理学、社会学的知識に基づくリスクレベルがより良い結果をもたらすことがわかりました。 さらに、非数値的リスクアセスメントの独立したレビューは、数値的リスクアセスメントよりも「深い」可能性が高いと考えられます。人は、数字がうまく確立されており、数字が大きく外れていない限り（そして時にはそれでも時々）、数字に挑戦する可能性は低いと信じる傾向があります。

[31] ヒューリスティックバイアスについては Tversky and Kahneman を参照してください。

第 7 章：安全マネージメントシステム

ナンシー・レブソン

前の章では、社会技術システムを解析するために STAMP（システム思考の原則）と STPA を使用する方法について説明しています。この章では、安全管理システムや SMS といった社会技術システムの特定のタイプの設計と解析に着目します。セーフティクリティカルなシステムを作成したり、これを運用するすべての組織では、SMS や安全文化が社会や組織の安全目標を達成するために多かれ少なかれ（時には有害と）役立つかもしれないが、すべて安全文化を持っているのと同様に、SMS を持っています。

この章では、システム思考（STAMP）と STPA を、新しい SMS の作成中または既存のものを分析し改善する際に使用する方法について説明します。 STPA は主に間接的にここで扱われている間、それは支援管理構造内（supportive management structure）で使用されている場合、STPA に基づく安全工学プログラムにのみ効果的です。この章の目標は、このような構造を設計することを支援することです。

SMS とは?

SMS の目標は、積極的に組織のあらゆる面における安全性を制御（管理）することです。製品を生産する組織では、SMS は、製品開発プロセスで、製品自体の安全性を管理します。サービス業では、安全性を製品内に設計し、職場で管理し、サービスが提供されなければなりません。社会技術システムは、最初から完全であることはめったになく、世界が時間の経過とともに変化するため、継続的な経験から学び、職場や製品やサービスだけでなく、管理システム自体を改善している場所で効果的な学習プロセスがなければなりません。

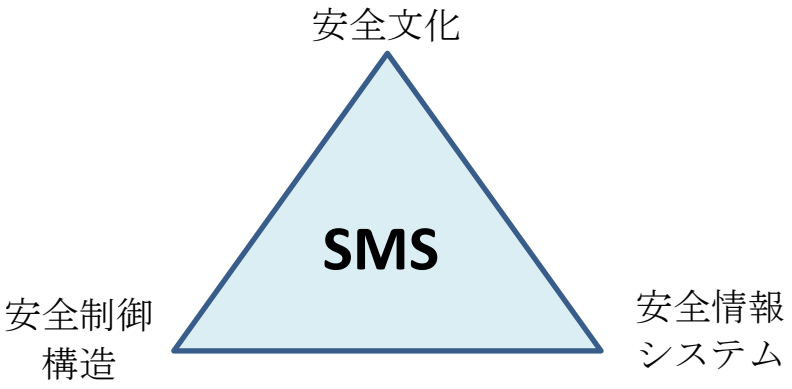
効果的な安全管理システムの決まったデザインはありません。設計の目標は、組織（製品開発やサービスプロバイダ）、組織が置かれている社会文化や組織構造や文化、製品やサービス固有の安全性、環境要因、組織が達成したい他の目標や規制環境などの要因に依存しています。FAA のようないくつかのグループは、リスク管理への一つのアプローチを指定しているが、それは皆のための適切な解決策ではないかもしれなく、リスク管理は、安全管理システムに必要とされるもののサブセットのみです。また、特定の安全管理システム規則は、高い安全レベルを達成するための特定のグループや組織のために最高のものではないかもしれない根本的な文化の価値を暗示しています。

しかし、成功した SMS には、いくつかの一般的な特性があります。この章で説明する SMS を設計するシステム思考アプローチは、事故報告書を読んだり作成して、関与する因子を特定し、事故を防止するために組織を支援してきた、システム安全工学における著者の数十年の経験の結果です。業界や組織により、多くの事故があるところもあれば、少ないところ、全く経験していないところもあります。（例えば、SUBSAFE、米国海軍の原子力潜水艦の安全プログラム[32]）。特定の要因は、組織あるいは産業が陥る三つのカテゴリーが目立ちます。安全性を制御するための最も効果的な方法の中には、皆のために実用的なものではないものがあるかもしれません。 SMS を設計することは、システムエンジニアリングの問題です。目標は、組織内で実装することが可能であるほど効果的である SMS を設計し、運営させるべきです。

文化、安全コントロールストラクチャー（管理）、および安全情報システムの安全性を管理する上で最も重要な組織のいくつかのコンポーネントがあります。しかし、これらの 3 つのコンポーネントを、分けて考えることはできません。システムアプローチは、これらがすべて最も効果的であ

[32] SUBSAFE は、２つの安全ゴールに焦点を当てている。: (1) 潜水艦の船体の防水性　と　(2) 洪水の緊急事態を制御し回復するための重要なシステムの操作性と完全性

るために、コヒーレントと一貫性のある全体の一部でなければならないことを示唆しています。文化が望ましいと許容する行動は何であるのか、どのような決定がなされるべきかを定義します。管理またはコントロールストラクチャーは、文化が、組織内で実施される方法を決定します。最後に、安全情報システムは、望みの安全文化を達成するための管理体制を成功させるために必要な情報を提供しています。それらを実行するための適切な情報がなければ、最高の意図でさえ十分ではありません。



安全文化

人々はさまざまな方法で安全文化を定義します。私は、安全文化のエドガー・シャインの定義が好きです:

安全文化とは、安全性に関する意思決定が行われる際の価値観や前提条件である。

図 6.1 は、組織文化の Shein の三つのレベルを示しています。一番下のレベルは、文化の本質です。価値観と深い文化的前提は、組織のルール、方針、慣行を作成するための基礎を形成します。また、これらの文書は、表面レベルの文化的成果物（例えば、ハザード解析、事故調査報告書など）がどのように生成されるかを記述します。

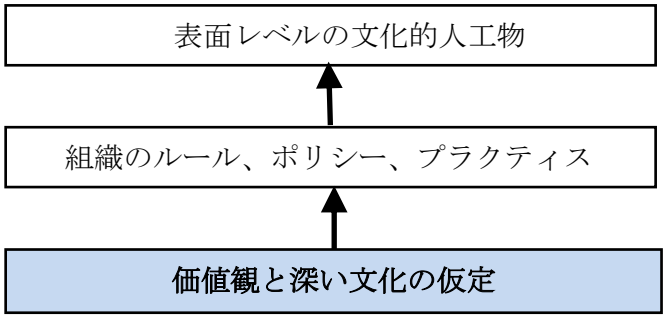


図 6.1：組織文化の Shein の 3 つのレベル

中間およびトップレベルは、文化ではありません。それらは単に組織の文化を反映しています。上位 2 つのレベルのみを変更しようとすると、一時的に行動が変化し、短期的にリスクが低下する可能性があります。下位レベルの共有価値や社会的規範に対応していないこれらのレベルの表面的な修正は、時間が経つと無効になります。同時に、それが埋め込まれている環境を変えずに文化を変えようとすると、失敗することもあります。

安全文化を「創造」することについての話は時々ありますが、既存のものは安全を促進する上であまり有用ではないかもしれないが、何らかの安全文化が常に存在します。無文化の組織、産業、社会といったものはありません。

多くの場合、歴史的要因と環境要因は、既存の安全文化を作成するための鍵となります。例えば、商用航空産業を構築するときに、ウィリアム・ボーイング社は、安全性の向上は、航空機および航空旅行を販売するために必要とされるであろうという事実に直面しました。 1955 年には、米国市民のわずか 20%しか飛行機を利用していませんでした。その時代に、航空機のクラッシュは、今日に比べて比較的一般的に発生していました。対照的に、原子力産業は、1957 年に Price-Anderson 法が成立したことにより開始され、事故の場合には責任を負うことと引き換えに政府によって規制されることに合意しました。いくつかの産業では、安全文化は原子力海軍のハイマン・リックオーバー司令官のような強力な個人のリーダーシップの結果でした。これらの産業のそれぞれにおいて、安全性の取り扱い方法の違いが存在します。

「良い」安全文化と「貧しい」安全文化を定義することは困難ですが、他のものよりも多くの事故や大きな損失を伴う産業や組織があります。事故率が比較的高い組織の安全文化は、以下の特徴の 1 つ以上を持つ傾向があります：

- リスク受容の文化：この文化は、事故が避けられず、誰もが注意するように勧めることを避けるためにはそれほど多くのことができないという前提に基づいている。事故は生産性の価格とみなされる。多くの場合、この仮定には、誰もが自分自身や他人の安全に責任を負うべきであり、その事故は個人の責任ある行動の欠如に起因するとの信念が伴う。誰もが責任感を持って安全に行動することができれば、事故は減少するだろうという信念が広まっている。
- 拒否の文化：拒否の文化では、リスクアセスメントはしばしば非現実的に低く、適切な調査がなければ信頼できるリスクと警告が却下される。経営陣は良いニュースを聞きたいだけである。焦点は、システムが許容可能で安全であることを示すことにあり、危険でない方法を特定することではない。
- コンプライアンスの文化：ここで焦点は、政府の規制を遵守することである。基本的な文化的信念は、規制を遵守すれば受け入れられる結果につながることである。規制当局は、事実の保証が強調された後、多くの場合、実際の製品やプロセスの安全性にほとんどまたはまったく影響を及ぼさずに、広範な「安全性ケース」の議論が行われた後、製品やサービスが安全であることを認証することに重点を置く傾向がある。
- ペーパーワークの文化：この文化は、多くのドキュメンテーションと分析書類を作成することで、安全な製品とサービスにつながるという信念に頼っている。たくさんの紙の山による解析が積み重ねられるが、設計や操作にはほとんど影響しない。安全関連の書類の大部分は、製品を設計および操作しているプロセス、プロセスを実装するプロセス、またはサービスを提供しているプロセスとは独立しており、相互作用がほとんどないグループによって作成される場合がある。
- 「SWAGGER」の文化：安全は弱い者のためのモノである。本当は、リスクに強い。

良い安全文化の特徴は特定するのが難しいですが、次のことが含まれます。安全性と安全性の目標についての開放性、悪いニュースを聞く意欲、そして、政府の規制に応じて多くのペーパーワークを作り出すことよりも安全を高めるために必要なことは何かを重視すること、組織の目標を達成する上で安全性が重要であると信じていることです。効果的な安全文化において、従業員は安全に関する懸念を聞き、適切な措置を講じることができ、従業員は耳を傾ける価値があり、尊重する価値があると考え、従業員は安心して懸念を報告し、その声には価値があると感じています。安全は、従業員がソリューションの一部であり、単に問題とみなされているのではないところで、共有

されている責任です。同時に、自分自身や他人を安全に保つために、単に全従業員に責任を置くだけではありません。

組織の安全文化はどのように確立され、変わりましたか？どんな組織における安全文化（意思決定に使用されるもの）も、トップマネジメントによって設定されます。安全管理に対する真摯なコミットメントは、それを達成するための最も重要な要素としてよく引用されています。従業員は、作業の安全性について合理的な懸念を示し、スケジュールやコストなどの他の目標よりも安全性を優先させる場合、サポートされていると感じる必要があります。

社員との日常的なやり取りにおける安全に対するマネージャの懸念事項は、安全問題に対する大きな影響を与える可能性があります。Paul O'Neill が 1989 年に Alcoa の CEO として社外に雇われたときです。彼はすぐに彼の主な目標は Alcoa をアメリカで最も安全な会社にし、怪我人がゼロであることだと発表しました。この発表された目標への典型的な反応は、Alcoa の取締役会が「狂ったヒッピー」でしたが、彼は会社を破壊しようとしていました。実際、Alcoa の利益は過去最高を記録し、2000 年に引退するまで継続しました。同時に、Alcoa は世界で最も安全な企業の 1 つになりました。 O'Neill は、安全性と生産性が矛盾しておらず、実際にお互いをサポートしていることを明確に理解していました。

あらゆるシステムの設計や工学では、目標を設定し、その目標を達成するための要求を確立することは、成功した取り組みの第一歩です。あなたがどこに行くのか分からなければ、そこに着くことはまずありません。設計プロセスを成功に導くためには、目標が必要です。経営陣は、組織内で意思決定を行う価値体系を確立します。したがって、安全文化の設計または改善の第一歩は、経営陣が安全関連の意思決定および行動の方法で期待することを確立し、伝達することです。

トップマネジメントが希望する価値を決定した後、次のステップは、短期間の書面による安全哲学とより詳細な安全方針を通じ、リーダーの基本を伝え、一般哲学を実施することです。トップマネジメントは、哲学と方針を明記することがマネージャと従業員から幅広く受けられるようにする責任があります。

安全文化の向上、つまり安全性の向上は、確立したい安全文化の種類を指導者が伝えることから始まります。ほとんどの企業には、安全性の取り扱い方法を正確に記述した広範な安全性ポリシー文書があります。これらが重要ではあるが、哲学的原則のより短い記述は、組織の期待される文化的原則を伝えるよい方法です。この哲学的声明は、安全と他の組織目標との関係を定義すべきです。

一部の一般的な文化原則はすべての組織に適用されますが、他のものの適用性は、組織が安全上重要なシステムの開発、運用、安全関連サービスの提供に依存します。安全哲学声明の一部であるかもしれないいくつかの一般原則：

1. すべてのけがや事故が回避可能である。
2. 品質と安全性の向上はコストとスケジュールの削減につながり、長期的には利益の増加につながる。事故防止は良いビジネスである。
3. 安全性と生産性は両立する。安全管理を改善することは、他の品質および性能要因を改善することにつながる。最大の業績には安全が求められる。
4. 製品やサービスの設計に安全性を組み込む必要がある。後でそれを追加することは効果が低く、より高価になる。事後保証は、安全性がまだ存在しない安全な設計を保証するものではない。それは事実の後にそれを保証しようとするよりも安全を築く方が良い。
5. 事故/事故因果関係分析の目的は、損失（またはほぼ損失）が発生した理由を判断して、誰かまたは何かを非難するのではなく適切な変更を行うことができるようにすることである。
6. インシデントや事故は、安全に動作していないシステムの重要な情報であり、総合的な因果関係の分析と改善活動をすべきである。

7. 安全に関する情報は、恐れなく表に出す必要がある。安全解析は責任を負うことなく行われる。
8. 安全性コミットメント、公開性、正直性が評価され、組織内で報われる。
9. 効果的なコミュニケーションと情報の共有は、損失を防ぐために不可欠である。
10. 各職員は、自分の業績と安全活動への貢献について評価される。

4 番目は、今日のどのぐらいの組織や業界で安全性が確保されているか、違反しているか、何らかの説明が必要です。最初から製品や作業スペースに安全を築くのではなく、事実の後に安全性やリスクを評価または評価することに重点を置くことが一般的です。デザインは安全であるか、そうでないかのどちらかであり、議論と評価がすべてその事実を変えることはありません。さらに、実際に安全性を設計するよりも高価であるだけでなく、設計が完了した後に可能性のある改善が通常は非常に限られているため、それを評価または確認することはあまり効果的ではありません。

実際の評価は、安全性が既に存在するという自信や、そうではないという証拠を提供するだけです。したがって事後評価を重視すると、肯定的な要素のみを強調したり、正の数を生み出す要因（確認バイアスと呼ばれる）を測定することができ、他の要因（管理や設計上の欠陥など）を無視しながら、数値を作り上げたり数値的な目標を達成するように操作したりします。これは、測定と評価が重要ではないことを意味するものではなく、安全なシステムを作成する努力が成功したかどうかだけを保証することができるが、すでに非安全な場合はシステムを安全にすることはできません。

事後評価の代わりに、安全であるという主張を提供するのではなく、設計が安全でないという証拠を提供するための考え方で評価プロセスに入ることがあります。このアプローチは、より効果的である可能性が高いです。ただし、この評価が行われるまでには、通常は見つかった問題を修正するには費用がかかりすぎるか、初めから安全性を設計に組み込むよりも費用がかかり、効果が低下します。設計プロセスの開始時と終了時に安全解析プロセスを開始すると、最良の結果が得られます（STPA をシステムエンジニアリングプロセスに統合する際のこのハンドブックの章を参照）。すなわち、安全性を証明する代わりに、危険を特定し、排除し、または制御することが重視されるべきです。安全性の事後評価を重視する者は、高い安全性を達成する可能性は低いでしょう。

さらに、評価の可能性や可能性を重視すると、過信や非現実的な評価、確率的でないか、確率を得ることができない重要な要素の省略につながる可能性があります。成功した SUBSAFE プログラムでは、安全性のためにシステムを認証する際に使用されるすべての証拠は、客観的品質証拠（Objective Quality Evidence：OQE）でなければなりません。OQE は、「検証可能な観測、測定、またはテストに基づいて製品またはサービスの品質に関する量的または質的な事実の記述」と定義されています。確率的リスクアセスメント（または事故に至る危険性の可能性を特定することを含む定性的リスクアセスメントを含む）は、未来を予測する試みであり、予測が有効であるかどうかを長年待たずに確認することはできません。効果的な将来を見通す方法はほとんどありません。確からしさを評価することは、過去と過去が同一で過去の経験が適用されたり、過去からの推測が行えるほど近い場合にのみ有効です。しかし、過去の製品を何らかの形で改良して、新製品を作ります。複製することで作るのではありません。また、時間の経過とともに変化しないシステム（およびその環境）はまれです。たとえ製品自体が変化したり劣化したりしなくても、システムを操作または使用しているユーザーは行動を変え始め、システムの使用が進化し、環境が変化します。また、時間の経過とともに変化しないシステム（およびその環境）はまれです。たとえ製品自体が変化したり劣化したりしなくても、システムを操作または使用しているユーザーは行動を変え始め、システムの使用が進化し、環境が変化します。

安全哲学とより詳細な方針の文書による方針がスタートであるが、十分ではありません。書面によるポリシーが実際の管理者の行動と異なる場合、職員はすばやく識別します。成功するためには、ただ単に驚くばかりでなく、その動きを通るだけでなく、トップにいる人々による真のコミットメントが必要です。

約束はどのように実証されるのでしょうか。優先順位を設定し、それをフォローして表示されます。個人的関与（例えば、安全上の意思決定が行われるトップマネジメントチェアーグループ）、適切な組織構造を設定することによって、指定された高級指導者を安全関連の責任に任命し、それらが効果的になるための十分な資源を提供することによって、最高の従業員を安全関連活動に配属し、努力のために報酬を与えることによって、他者の取り組みに対応することによって、それは責任を最小限にすることによっても伝えられます。指導者は、損失につながる系統要因（systemic factors）を修正し、責任を負う人（通常は組織の最下位レベルの人物）を見つけて前に進むことだけが最優先であることを実証する必要があります。最後に、リーダーは、必要な行動を促進するために、組織内のインセンティブ構造を設計する必要があります。

このセクションの主要なアイディアは、次のボックスにまとめられています:

管理者の安全文化向上のためのヒント

- 意思決定に使用される目標と価値を設定する：安全関連の意思決定と行動において期待されることを確立し、伝達する。
- 仕事における安全性について合理的な懸念を示す従業員をサポートする
- 短く書かれた安全哲学とより詳細な安全方針を作成する
- 安全理念と方針が管理者と従業員から幅広い賛同を得られるようにする
- 意思決定の安全哲学に従い、誰もが同じことを期待する
- 保証や事後評価ではなく、建物の安全を強調する
- 設計が安全ではないという証拠を提供し、安全であるという主張を提供しないという目的で評価を実施する
- 認証または保証の客観的品質証拠を要求する
- 安全への取組みを示す
  - 個人的な関与
  - 優先順位を設定し、それに従う
  - 適切な組織構造の設定
  - 高位で尊敬される指導者を安全関連の役割と責任に任命する
  - 安全のために十分な資源を提供すること
  - 不必要な人や消耗しない人だけでなく、最高の従業員を安全関連の活動に割り当てる
  - 従業員の安全対策への報酬
  - 他者の取り組みへの対応
- 責任を最小限に抑える。「人」ではなく「なぜ」に焦点をあてる
- 望ましい安全関連の行動を促すためのインセンティブ構造を設計する

安全コントロールストラクチャー

前の 2 つの章で説明したように、安全コントロールストラクチャーの設計は、システムエンジニアリングのもう 1 つの形式です。成果の目的の宣言で始まります。 SMS は次のとおりです:

何：組織内の安全の確保と維持を支援するコントロールストラクチャー

なぜ：危険を排除するために、可能でない場合、制御された（緩和された）、効果的な安全文化を促進するために

どうやって：ハザードを管理する責任能力（RAA）や包括的で使用可能な安全情報システムや管理コントロールを確立することにより

全体的な目標は、損失を除去または低減するコントロールストラクチャーを設計することです。この目標を達成するには、コントロールストラクチャーのすべてのレベルで安全タスクの期待、責任、権限、説明責任を明確に定義する必要があります。 さらに、効果的に動作させるためには、構造体に適切なフィードバックと調整が必要です。また、内部または外部の変化のためにコントロールが無効になったときに、合図をする先行指標も入れるべきです。 全体のコントロールストラクチャーは、物理的な設計、定義されたプロセスと手順、社会的な相互作用と文化を通じて、システムの行動に関する安全上の制約を強制しなければなりません。

おそらく、開発のための安全管理構造と運用のための安全コントロールストラクチャーとの間には、大きな違いがあります。 あなたが規制産業であれば、政府の規制機関をあなたの安全管理構造モデルに含める必要があります。 その他の外部グループは、裁判所、保険会社、ユーザーグループなどに含めることが重要です。

一般的な安全管理体制の設計上の考慮事項

安全管理構造の評価を設計する際の基本的な考慮事項のいくつかは、安全に関する責任の割り当て方法、安全関連活動のための組織内の適切な場所、情報の伝達と活動の調整、変更の管理と制御、設計と管理 フィードバックを促し、使用するリスク管理手続きの設計と役割を決定します。 さらに、必要とされる教育訓練の努力と、コントロールストラクチャーそのものの学習と継続的な改善がどのように保証されるのかを検討しなければなりません。 さらに付録 D「SMS の設計と評価のガイドライン」を参照してください。

<u>責任の割り当て</u>:

リーダーシップは、高いレベルの安全性を達成するための鍵です。 つまり、組織の安全機能のリーダーシップは、将来の指導者やマネージャーを養成するためのローテーションの割り当てであってはなりません。 損失の少ない組織では、安全に熱心な安全機能のリーダーと、損失を防ぐために果たす役割を任命します。 組織内には、安全に関するコントロールストラクチャで損失を防止することを約束している人たちが登場することを可能にするキャリアパスが必要です。

効果的な管理システムと同様に、責任、権限、説明責任が割り当てられていなければなりません。 「誰もが自分自身や他人の安全と製品の安全性に責任がある」という考えは、過度の事故を招く。 誰もが安全の責任を負うなら、誰も責任を負いません。

安全の責任は組織のあらゆるレベルにありますが、すべての責任において適切な責任が異なります。通常は、いくつかの事故の多い組織では、安全の責任がコントロールストラクチャーに表現されるべきであるという考え（信念）が一般的です。システムの一部が実際どのように動くのかについての詳しい情報は、より下位のレベルで持っているということが議論されています。この議論の問題は、下位レベルにも視点がないことです。彼らはシステムの特定の部分に関する詳細な情報を持っていますが、システムの他の部分に関する情報を持っていないため、システム全体のコンポーネント間の非安全な相互作用を予期したり、防ぐことはできません。下位レベルは長期的ではなく短期的な視点に集中する傾向がありますが、上位レベルの組織レベルでは、システムの下位コンポーネントの目標とは対照的にシステム目標に焦点を当てています。より高度な組織レベルでは、より低いレベルのコンポーネント間のやりとりをコントロールし、その下にあるコンポーネントで、安全性の制約が強制されるようにする必要があります。

各レベルは、適切な手順を使用し、正しく実行し、効果的であることを確認することによって、下のレベルの監視を提供する必要があります。 また、通常、安全管理システムが設計され、管理システムの各レベルで適切に機能することを確実にする責任を持つ管理上の焦点（またはポイント）が必要です。

責任を割り当てるためのヒント

- 安全に熱心な指導者を任命する
- 損失を防ぐことに取り組んでいる人のキャリアパスを作成する
- 安全管理構造のすべてのレベルでの期待、責任、権限、説明責任の明確な定義を提供する
- 誰もが安全の責任を負わないようにする
- コントロールストラクチャー全体のすべてのレベルで安全責任を割り当てる
- SMS が設計され、正しく機能することを保証する責任を誰かに割り当てる

組織の配置:

組織のシステム安全活動をどこに配置するか、フィードバックの設計と管理、変更の管理方法を決定する際に考慮する必要がある基本原則があります。

第一に、必要な活動が確実に行われ、効果的であることを保証するなど、企業レベルの責任を持つハイレベルなグループが必要です。このグループはまた、リーダーシップと調整を提供します。安全活動は大企業の開発と運営のあらゆる部分に浸透するが、共通の方法論とアプローチが個々の活動を強化します。このグループのリーダーは、トップマネジメントと直接コミュニケーションを取って、あらゆるタイプの経営意思決定に情報を提供する能力を持つ必要があります。これは、この職位の人が影響力のある人に報告し、上級管理職のサポートを受けていることを意味します。

このトップレベルの安全管理の下で、組織のすべてのレベルで活動が行われることがありますが、上位レベルはより広い責任を持ち、後続レベルはそれぞれ、開発組織の事業単位、プログラムおよびプロジェクトレベルで適切な責任を果たします。

よくある間違いは、安全を ライン操作にはほとんど影響しない職員の機能にすることです。

安全コントロールストラクチャーには、決まった正しい設計が存在しません。効果的なデザインは、業界、組織の管理スタイルなどに依存します。このハンドブックの付録には、開発および運用の安全管理構造に含めるべき責任の一部を記載しています。

2 つ目に、このハンドブックで前述したように、システムエンジニアリングからシステム・セーフティ・エンジニアリングを分離することは、製品開発組織の間違いです。安全性の問題は開発プロセスの早い段階から取り組まなければならないため、安全目標を達成するためのコストが高くなる可能性があります。安全工学の取り組みは事実上の保証に集中すべきではありません。サービス組織には異なる懸念があるが、安全に関する意思決定が行われているすべての部門またはグループには、安全工学のコンポーネントが必要です。同時に、第一次意思決定グループ外の意思決定機関で独立した監督を受ける理由もあります。すべてのタイプの組織にとって、安全は職場に配慮されなければならず、組織におけるこの責任の適切な場所についての決定が必要になります。

組織の構造に安全責任を割り当てるための一般的な原則は次のとおりです:

- 意思決定者は、安全情報を提供できる人と直接リンクする必要がある。重要な情報が一連のコマンドを作り出す必要がある場合、スケジュールや予算のために故意に、または不注意に、紛失または修正される可能性がある。直接通信チャネルは、潜在的に競合する利益を有するグループを介してフィルタリングされることによって、変更されることなく、情報がタイムリーに受信され、より多くの機会を提供する。意思決定者によっては、情報への迅速なアクセスが必要な場合もある。
- 組織のほとんどの部分への直接通信チャネルが必要である。 ほとんどすべての組織活動に安全性が関与している可能性がある。

- 安全は意思決定に影響を及ぼすものでなければならず、意思決定者は安全に関する意思決定が必要な時に必要な安全情報にアクセスできることを意味する。

米国原子力潜水艦の SUBSAFE プログラムは、これらの要求の多くを満たす独自の設計を採用しています。彼らは、彼らの構造を権力の分離または「3 足歩行」と表現しています（図 6.2）。マネージャは、Independent Technical Authority（ITA）によって導出された一連の許容可能なオプション（安全性に関して）からのみ選択できます。テクニカルオーソリティとは、技術標準やポリシーへの準拠を確立し保証するプロセスです。ITA は、技術的に許容可能な代替案を提供し、リスク[33]評価と価値評価を提供しています。責任（およびアカウンタビリティ）には以下が含まれます:

- 技術基準の設定実施
- 主題専門知識の維持
- 安全で信頼できる運用の保証
- 効果的かつ効率的なシステムエンジニアリングの保証
- 不偏の独立した技術的意思決定
- テクニカルおよびエンジニアリング機能のスチュワードシップの提供

第 3 の組織は、コンプライアンス検証機関です。プログラム管理者と ITA と同等の権限を持ちます。

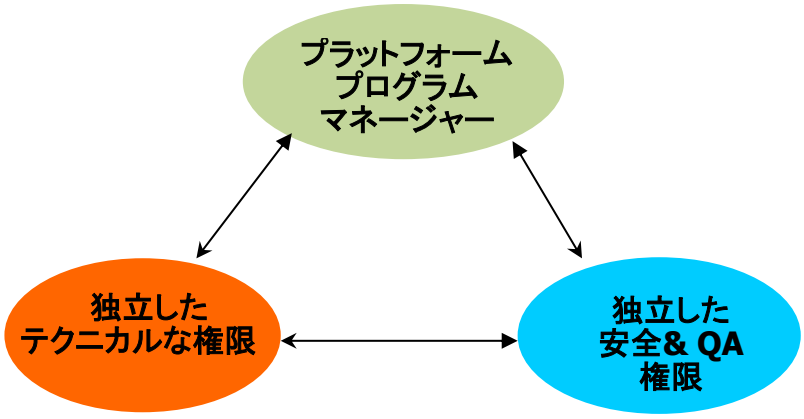


図 6.2: SUBSAFE 3 足歩行コンセプト

[33] リスクは定量的または定性的に指定することができるが、本章の他の部分で説明したように、SUBSAFE では客観的な品質証拠を使用する必要があるため、検証または検証できない確率的リスク評価または尤度推定は認められない。

コントロールストラクチャーの中の活動の場所を見つけるためのヒント

- 必要な活動が確実に行われ、効果的であり、リーダーシップと調整を提供するなど、企業レベルの責任を持ったハイレベルなグループを作成する。
  - このグループは共通の方法論を保証し、誰もがアプローチを使用しています
  - トップマネジメントへの直接的なパスを持ち、経営意思決定へのインプットを提供しなければならない
  - 影響を及ぼす人物に報告し、上級管理職の支援を受けなければならない
- 安全を職員の機能だけでなく、ラインの運用に直接的な影響を及ぼします。
- 製品開発におけるシステムの安全性とシステムエンジニアリングを分離しない組織。安全機能をエンジニアリング機能に統合する。
- 安全関連の意思決定が行われるすべての部門またはグループに安全エンジニアリングコンポーネントを含める。
- 統合された機能に加えて独立した監視を提供する
- 意思決定者が安全解析と情報を提供できる人に直接リンクするようにする。
- 安全関連の活動があり、安全意思決定が必要な組織のあらゆる部分の間に直接のコミュニケーションチャネルを作成する。
- 権限構造の SUBSAFE 分離を実装することを検討する。少なくとも、プログラム上の意思決定とは独立した技術的決定が行われるようする。

<u>コミュニケーションとコーディネーション：</u>:

安全管理の役割を果たすために情報を必要とする人に情報をフィードバックし、伝達することは、安全管理構造の設計において考慮すべき重要な要素です。たくさんの（時には莫大な量の）情報を収集することだけではありません。実際、情報の収集が多すぎると、効果的な安全意思決定を行うために各自が実際に必要とする情報を識別する能力を圧倒すると、情報システムが低下する可能性があります。さらに、安全関連の意思決定を改善するために必要な情報は、必要なときに利用可能でなければなりません。損失を減らすことが非常に重要であるため、安全情報システムについてはこの章の後半で別途説明します。

コミュニケーションは、活動とイベントへの対応の調整においても重要です。重複する責任を持つ人々は、通信制約と安全制約が確実に適用されるように活動を調整することが必要です。たとえば、1 つのサブシステムにおける安全性にかかわる個所の変更は、別のサブシステムおよびシステム全体に影響する可能性があります。安全活動は断片化して調整してはいけません。 インタラクションは、階層コンポーネント間だけでなく、開発と製造の間、開発と運用の間など、同じレベルの異なるパーツやシステム間でも定義する必要があります。図 6.3 は、システム開発者とシステムオペレータの間で伝達される必要のある安全関連情報のタイプの例を示しています。

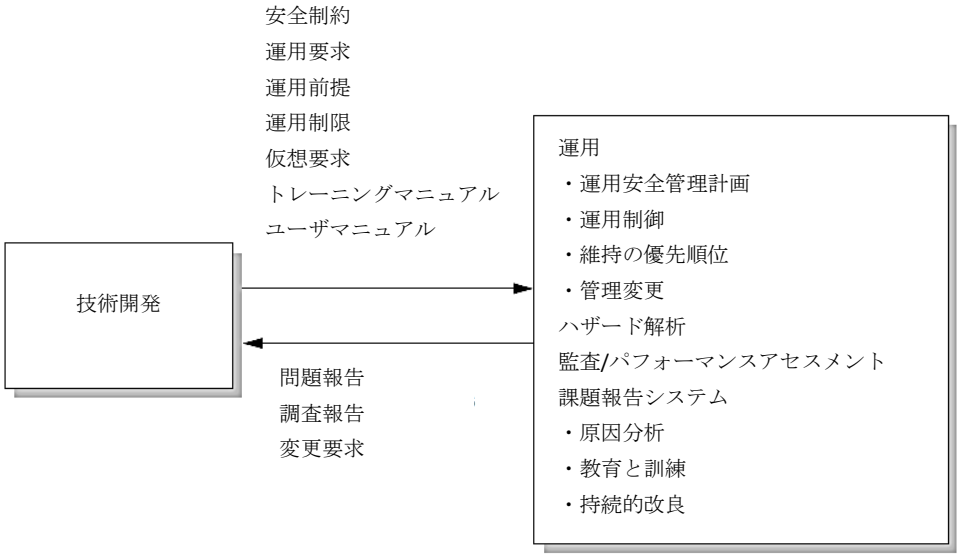


図 6.3. 両者の安全活動を支援するために、開発と運用の間に必要な情報チャンネル

安全コントロールストラクチャーのすべての構成要素間で伝達される情報と同様の記述を特定し、文書化する必要があります。

防衛省が策定したコミュニケーションと調整のための非常に有効な手段の 1 つがワーキンググループです。防衛部門のプロジェクトは長年にわたり、非常に大きくて複雑なものであり、多くの場合、地理的にも組織的にも分散している多数の参加者が関わっています。このようなプロジェクトでは、調整とコミュニケーションが大きな問題になる可能性があります。ソリューションの一部は、プロジェクトの高度な管理が国防総省にある階層構造によって提供されますが、下請け業者間でシステムエンジニアリングと調整を提供するプライム請負業者があります。このような構造が機能するには、通信が重要になります。ワーキンググループは、こうした調整とコミュニケーションの取り組みに成功し、それほど複雑ではないプロジェクトにも適応することができます。

作業グループは、安全コントロールストラクチャーの 2 つの階層構成要素間、または同じレベルの 2 つ以上の構成要素間のインタフェースを提供します。これらのグループのメンバーは、努力の調整、未解決の安全問題のステータスの報告、独立した安全活動に関する情報を共有しています。コントロールストラクチャーの各レベルには、その活動レベルに応じてメンバーと責任があるワーキンググループが存在する場合があります。企業のワーキンググループは、さまざまな部門やプログラムの安全管理者で構成され、下位レベルのワーキンググループは、製品のさまざまな部分を設計するグループの代表から構成されます。新しい解決策を必要とする特別な種類の安全上の問題がある場合、ソフトウエア安全作業部会などの経験とアプローチを共有するために安全グループを作ることができます。政府機関は、代理店のオフィスとプライムコントラクターから成る大規模なプロジェクトのための安全ワーキンググループを作成するかもしれません。プライム請負業者は、すべての下請け業者の安全管理者で構成される安全ワーキンググループを作成することができます。

コミュニケーションとコーディネートの設計のためのヒント

- 意思決定者が安全に関する意思決定に必要な情報を入手できることを確認する。
- 通信チャネルと重複する安全責任を持つ人々の間の活動を調整する方法を提供する
- 安全活動が細分化されていないこと、調整されていないことを確認する。
- 必要なインタラクションを定義し、必要な情報フローを定義して使用することを保証する
- 安全コントロールストラクチャーのすべてのコンポーネントの中で、必要な安全関連の通信チャネルを特定し、文書化する。
- フィードバックと調整チャネルが存在し、動作していることを確認する
- 安全作業部会の設置を検討する

変更管理と制御

ほとんどの事故は、ある種の変化の後に起こります[34]。この事実は継続しているので意外ではありません。

理論的には、行動や環境の変化なしに行ったことは、同じ結果につながるはずです。適応と変化は、あらゆるシステムの本質的な部分であり、組織が繁栄するために必要です。問題は変わっていないが、非安全な状態に変化します。したがって、SMS は、非安全な変更を防止するために慎重に設計されたコントロールを備えている必要があります。非安全な状態を防ぐための努力にもかかわらず、それらが発生した場合、それらを検出します。目標は、安全上の制約に違反しない限り変更を許可することです。計画変更と計画外変更の 2 種類の変更を考慮する必要があります。

予定変更：SMS は、組織の変更、人間の行動やプロセス、運用（メンテナンスフェーズへの変更、運用フェーズへの変更を含む）、または環境の変更を含む計画的な変更に直面しても引き続き有効でなければなりません。ほとんどの企業では、変更のすべての計画変更が安全性への影響に関して評価されることを要求する変更管理（MOC）方針を持っていなければなりません。そのような評価の費用は、変更の範囲、文書の質、および元のハザード解析の実施方法に依存します。STPA は、識別されたハザードからその原因となるシナリオおよびそれらを防止するために使用される設計機能に完全なトレーサビリティを組み込んでいます。そのようなトレーサビリティがなければ、変更後の再分析の費用は実用的でない可能性があり、したがってスキップされる可能性があります。

実際には、MOC 手順は実際にはスキップされることが多く、決して発生する必要のない損失が生じます。安全管理構造は、MOC 手続およびフィードバックチャネルの実施に責任を負わなければなりません。それらをスキップする原因が MOC 変更手続きそのもの（あまりにも面倒、困難すぎ、時間がかかり過ぎるなど）である場合、MOC 手続きを変更する必要があるかもしれません。

MOC 手順の実施コストは、システムの文書化の質と元のハザード解析がどのように行われたかに依存します。STPA の設計目標は、変更を評価する際のコストを最小限に抑えることです。

計画外の変更：計画外の変更にはより困難な課題があります。（1）安全でない可能性のある変更を特定する方法、（2）これらの変更に対応する方法が必要です。前章で説明した先行指標は、システム（SMS を含む）の設計および安全解析プロセスで行われた前提から作成する必要があります。先行指標指標を特定するには、組織構造、SMS、製品および職場の設計および開発中に前提条件を記録する必要があります。STPA のコントロールストラクチャー、UCA、および因果関係は、効果的な先行指標指標を作成するために必要な情報を提供します。因果関係のシナリオを排除できない場合は、STPA 分析の情報を使用して、リスクが起きないために対策を行うことができ、また、監

[34] Nancy G. Leveson, *Safeware*, Addison-Wesley Publishers を参照

査とパフォーマンスアセスメントをチェックするための指標や前提条件を特定することができます。

危険な変更が発生する一般的な理由は、リスクの再評価です。損失のない期間が経過した後、人々はリスクの見直しを下方に開始します。変更されていない可能性があるので、リスクは、本当は減少していないが、人々はリスクが下がっていると信じるかもしれません。彼らは自分のルールに違反し始め、安全が影響を受けないと主張して正当化します。安全管理構造のプロセスは、安全マージンが侵食される前に、このリスクの再評価プロセスを中断する必要があります。1 つの要求は、実際のリスクレベルに関する適切な情報の伝達です。経営陣は、管理しているプロセスにおけるリスクの状態を認識する必要があります。それには、設計された安全コントロールの状態に関する適切なフィードバックが必要です。

さらに、行動が真のリスクレベルに反している場合、責任を持つ人に警告機能が必要です。

この現象を防止するための鍵は、安全目標の達成方法に柔軟性を持たせ、意思決定者による正確なリスクアセスメントを可能にする情報を提供することです。"

効果的な SMS は、時間の経過とともにシステムが劣化するのを防ぐために、継続的な確認を行います。SUBSAFE プログラムは、一例として、3 つの最も困難な課題であることを説明するために多くの努力を払っています：

1. 無知：知らない。
2. 傲慢：自尊心、思いやり、または知的優越性の仮定と事実に裏付けられていない知識の推定。
3. 自己満足：実際の危険または認識不足の欠如に伴う業績への満足

SUBSAFE プログラムの設計と焦点の多くは、これらの 3 つの課題に対処することを目的としています。SMS の設計には、それらに対する警戒を提供するための手順が含まれている必要があります。

変更管理および制御するためのヒント

- 安全でない変更を防ぐためのコントロールと MOC ポリシーの設計と、それが発生するかどうかの検出
- 潜在的な安全への影響を考慮して、一時的な変更を含むすべての計画された変更を評価する
- MOC 手続が実施され、遵守されていることを保証する責任を負う。 そうでない場合は、理由を見つけて問題を解決する。
- MOC 手順の実行コストを最小限に抑える文書と手順を作成する。
- 不安定なものである可能性のある予定されていない変更を特定し、これらの変更に対応する方法を作成する。
    - 想定に基づく先行指標を作成し、潜在的に安全でない変化が特定された場合に効果的に監視し、対応するリスク管理プログラムを作成する。
    - 組織構造の設計と開発、SMS、製品及び職場の前提を記録する。
    - 監査やパフォーマンスチェック、標識などに含まれている、状態やリスク回避活動や主要なチェックプログラムを作成する。
    - コントロールが無効になったときに先行するインジケータに信号を送る
- 意思決定者は、設計された安全管理の現在のレベルおよび状態に関する情報を確実に入手する
- 振舞いが実際のリスクレベルと一致しないことの応答が出ている時に、応答する責任をわりてることが必要
- 時間の経過とともに安全管理体制や安全文化の低下や自己満足の増加について警戒が必要。

フィードバックの設計と奨励

正確なリスクアセスメントには、情報フローと適切に機能するフィードバックチャネルが必要です。文化的な問題は、フィードバックの問題の一般的な理由です。フィードバックチャネルを実装するには、監査とパフォーマンスアセスメント、事故/インシデントの原因分析、レポートシステムの 3 つの一般的な方法があります。

監査とパフォーマンス評価: 多くの種類の監査と性能評価がありますが、安全性の状態を評価することを目標とするものは、安全性の制約と安全コントロールの設計における前提から始まります。 監査には、製品やプロセスだけでなく、安全管理システムそのものや、損失の防止を確実にするために設計されたコントロールの有効性が含まれるべきです。 安全監査と性能評価の少なくとも一部は、設計され、動作すると想定される安全コントロールストラクチャーの動作に焦点を当てるべきです。

安全コントロールストラクチャー全体は、より低いレベルだけでなく、監査されなければなりません。SUBSAFE プログラムでは、上級 Admirals でさえ SUBSAFE 監査の対象となります。これにより、プログラムが想定どおりに運用されているだけでなく、すべての従業員が SUBSAFE ルールに従うことが期待され、トップマネジメントが受け入れて解決する意思があることが、監査にプラスの文化的要素を提供する SUBSAFE コミュニティの他のメンバーと同様に調査結果を監査します。SUBSAFE 安全管理体系の低レベルの人々は、それらの者の業績評価に参加し、プログラム全体の安全へのコミットメントの目に見える兆候を示し、損失を防止する上での全員の重要性を示します。監査の受け入れと結果としての改善の実施、つまり例としてのリーダーシップは、安全とその改善へのコミットメントを伝える強力な方法です。

参加型および非懲罰型監査は非常に効果的です。 そのような監査の目的は、それが建設的な学習経験であり、判断プロセスではないということです。 従業員を評価する方法ではなく、安全性を向上させるチャンスとみなされるべきです。 外部の監査会社による通常の監視のみの監査プロセスの代わりに、直接監査されない組織の他の部門の専門家が監査チームを構成する必要があります。さまざまなステークホルダー（利害関係者）が監査で役割を果たし、監査対象のグループの個人も監査チームに参加する可能性があります。 目標は、これが私たちのプラクティスを改善し、監査人を含む関係者全員に学習活動を提供する機会であるという姿勢を作り出すことです。

監査自体には、特定された問題や潜在的な解決策を完全に理解するために、監査対象者との継続的なコミュニケーションが必要です。 外部監査の通常のルールとは異なり、参加型監査では、即時のフィードバックが提供され、解決策が議論されるべきです。 これを行うことで、安全性を向上させ、関係者を罰するか評価しないことが目標であるという理解が強化されます。また、実際に監査が行われてから数ヶ月後に通常の書面による報告が行われた後ではなく、知識のあるチームが現場にいるときに、問題を解決する機会も提供します

安全監査とパフォーマンスアセスメントの重要な目標は、実際に存在する安全知識と訓練のレベルを測定することであり、管理者が何を考えているのか、訓練プログラムとユーザマニュアルに存在するものではありません。 監査は、トレーニングおよび教育活動の潜在的な改善についての重要なフィードバックを提供することができます。 非懲罰的監査哲学に沿って、知識評価は、査定対象者による罰として扱われる否定的な方法で使用すべきではありません。 代わりに、目標は訓練と教育努力を改善することであるべきです。

事件と事故調査: インシデントと事故の調査は、SMS が設計通りに動作していないか、または予想通りに動作していないという明確な証拠を提供します。調査手続きは、結果を有効利用できる組織構造に埋め込まなければなりません。 関係するすべての系統要因は、症状または技術的要因だけでなく、特定されなければなりません。 責任を割り当てることが目標であってはなりません。 むしろ、安全コントロールストラクチャーが損失または損失に近いことを防止するのに効果的でなかっ

た理由を見出すことが目的でなければなりません。これは、有害事象に対する各コンポーネントの潜在的な貢献を特定するために、安全コントロールストラクチャー全体を調査しなければならないことを意味します。

マネージャーは、一連の命令で発生するインシデントの調査に責任を負うべきではありません。調査官と因果アナリストは、管理上、財務上独立していなければなりません。 独立した予算を持ち、高水準かつ独立した管理を持つ訓練されたチームを使用することが検討されるべきです。

調査には単にレポートを書くだけではありません。事故に寄与した安全管理体制の側面を強化するための適切な措置を取る責任を負わなければなりません。その後、修正が有効であることを確認するためのフォローアップが必要です。あまりにもしばしば修正が加えられましたが、その修正が安全管理の改善に成功したかどうかを判断する試みはありません。最後に、この調査結果は、将来の監査と業績評価へのインプットとして使用すべきです。過去に事故や事故に至ったのと同じ要因が再発した場合、なぜそれらの要因が是正されなかったのか、あるいはしばらく取り除かれたにもかかわらず再発した理由を調査する必要があります。インシデントの原因を取り除くための修正が効果的でない場合、推奨策を作成し、それに対応するプロセスの調査は、組織におけるこれらのプロセスの弱点を特定し、改善するために必要です。つまり、インシデントに関わる要因を修正するだけでなく、以前のインシデントや事故の後に修正が不十分になったプロセスを修正する必要があります。

システム報告: 報告システムは非常に重要です。 多くの場合、事故の後、同じ事象が過去に複数回発生したが、報告されなかったか、報告された場合には決して訂正されなかったことが分かりました。これらの出来事はニアミスを含んでいるかもしれません。 たとえば、いくつかの航空機はアプローチを正しく飛行していないかもしれませんが、事故には至らず、報告システムが存在していても報告できない場合があります。 事故が起きてはじめて、措置が講じられます。

報告システムに掲載されている事故報告には、共通事象が記載されているが使用されていません。 これらのレポートには、通常、報告システムの使用方法を習得し、報告システムの使用を要求するための推奨事項が含まれています。 この勧告では、報告システム自体の設計ではなく、潜在的な報告者に問題があると想定しています。

報告システムそのものを調べると、使用するのが難しくて扱いにくく、報告された情報がブラックホールに入っているように見えるかもしれません。 組織は報告された要因について何もしていないため、公式の報告システムを通過することに意味がないと信じているかもしれません。 多くの場合、発見した問題は、警報システムを迂回して、その問題を解決できると信じている人やグループへとただ単に直接向かいます。これは短期間で効果的かつ効率的であるかもしれないが、根本原因が排除されないため、将来同じ問題が発生する可能性があります。 場合によっては、報告された情報が記者に対して使用される恐れがあるため、報告は禁止されます。 匿名の報告システムはここで役立ちます。

考慮すべきもう 1 つの要因は、リスクの認識を作成する前に問題を特定したか、またはエラーとして認識している可能性があるため、イベントがフロントラインオペレータによって報告されないことが多いということです。 ほとんどの人は、通常の雇用プロセスの一部としてリスクが発生した場合、そのリスクを認識するように訓練されていません。 人々はデザインの欠陥を「正常」と受け入れ、おそらくすでに知られているかもしれないと報告しています。 関連する要因は、人々が一般的に危険な出来事を報告する必要がないということです。 ニアミスは、後者のカテゴリに入る可能性があります。

一般的に、報告は奨励される必要があります。 これは、アクセシビリティを最大化し、不安を最小限に抑え、得られた情報に基づいて行動することによって達成することができます。報告書やチャンネルは、簡単かつ遍在的に利用可能であり、記入や使用が煩雑ではありません。不安を最小限にするために、報告プロセスが何であるかに関する書面による方針があるべきです。報告の結果報告書とその報告書に従っている者の権利、特権、保護、義務を含みます。書面による方針がなけ

れば、あいまいさが存在し、人々はより少ない情報しか開示しません。 あるいは、誰がフォローアップの責任を負っているかについてのあいまいさは、誰か他の人が対処するだろうと皆が思い込むことにつながるかもしれません。

最後に、報告を促すことはフィードバックを提供することを含みます。 レポートが作成された直後に、情報を提供した人は、レポートが受信されたことを通知され、調査されることを保証され、入力に感謝されます。 第 2 の重要な要素は、調査の結果についてのフィードバックと、再発を防止するための結果として得られたすべてのステップです。 記者は、彼らの懸念が無視されているように感じるべきではありません。

フィードバックの設計と奨励のヒント

- 監査、業績評価、報告システムの継続的な効果を設計し、保証する。
- 安全管理構造そのもの（すべてのレベルを含む）と設計されたコントロールの有効性を監査する。
- 判断プロセスではなく、構成的学習経験であるように監査を設計する。
  - 参加者として、監査対象のグループのメンバーを含める
  - 安全を向上させる方法として、また学習活動として監査を使用し、従業員を評価しない。
- 監査を活用して、訓練活動の効果を評価し、訓練活動を改善するためのフィードバック（知識評価）を提供するために使用する
- 理由にフォーカスし、人にはフォーカスせず、効果的なシステムレベルの事故/事故原因分析手順を作成する。
- より深刻な問題の症状だけでなく、システミックファクタを特定し、STAMP とシステム思考（CAST など）に基づいてインシデントと調査の手順を作成する
- 結果を活用できるように組織構造に調査手順を埋め込む。任を割り当てること、または「根本的な原因」を見つけることは、目標にするべきではない。
- 事故調査は、直面した管理組織でから管理上、財政的に行うべきである。
- 独立した予算と高度な管理を備えた高度に訓練されたチームの使用を検討する。 彼らが効果的であったかどうか、もしそうでなければなぜかを判断するために推奨事項をフォローアップする。
- 報告システムを使いやすく、利用できるようにし、匿名の報告チャネルが存在するようにする。
  - 報告を奨励し、いつ使用すべきかを人々に教える。
  - 書面によるポリシーを提供する
  - アクセシビリティを最大化し、不安を最小限に抑え、得られた情報に基づいて行動する。
  - 報告チャネルを使用している人にフィードバックを提供する。記者は、懸念事項が無視されていないことを感じる必要がある。

<u>リスクマネジメント</u>

リスク管理システムを「安全管理システム」と呼ぶこともありますが、同じではありません。SMS は、より大きなコンセプトであり、リスク管理活動を伴うでしょう（通常はそうです）。 リスク管理は、ハザードを特定し、分析し、この情報を使用して製品、プロセス、サービス、および職場の設計を通じて損失を減らすことに関連する一連の活動です。 これらの活動は、安全管理構造に組み込まれる予定です。 リスク管理は、危険を防止するための技術活動で構成されていますが、SMS には組織的、管理的、文化的側面の安全性も含まれています。

STAMP は、より広範な、または少なくとも異なる定義のリスクを意味します。 リスクは伝統的に、発生する危険性や事故の重大性と可能性として定義されてきました。 一方、STAMP の中では、

リスクは、安全なシステムの動作を強制するために使用されるコントロールの有効性の観点から定義されます。つまり、安全コントロールストラクチャーの設計と操作です。

この定義は事象の発生の可能性の決定を必要とせず、むしろそれらを防止するために使用されるコントロールの有効性の評価を必要とすることに留意します。

リスク管理システムを構築するには、技術リスク管理活動を実施する際に使用される手順を設計し、これらの手順を実施する責任を安全管理のコンポーネントに割り当てる必要があります。 リスク管理には、リスクが増加している時期を特定するための先行指標を作成することも含まれます。

実際、リスク管理システムの有効性の重要な要素は、重大な損失が発生する前にリスクが増加している時期を特定できることです。 すべての安全対策には、システムの構成要素がどのように動作するか、およびそれらが動作する環境についての前提が関係します。 これらの仮定を逸脱すると、当初のリスク識別および管理上の前提が損なわれます。 実質的にすべての事故には、認識されなかった、またはその応答が存在しないか効果がなかった前兆があります。 事故の前兆は、損失が発生しないが、他の状況下で起こる可能性がある事象または状況です。 先行指標は、安全管理システムが想定されていた設計とおりに動作していないことを示す組織または組織の動作の特性です。

これを見るもう 1 つの方法は、先行指標は、安全性が保証された前提条件に欠陥があるか、もはや真実でなく、リスクが増加する可能性がある損失が生じる前の証拠にすぎないということです。 例として、オランダの Shell Moerdijk 化学プラント爆薬の前に、同じ条件が他の Shell プラントで爆発を引き起こした 2 つの事例がありました。これらの爆発は決して徹底的に調査されなかったが、少なくとも解析結果は、条件が不可能であり爆発に至らないと仮定された Shell Moerdijk プラントの設計など、後の設計活動で使用されませんでした。驚くべきことに、確率論的リスクアセスメントは、分析の前提が真実ではない具体的な証拠がシステムの実際の使用から集められた後でさえ、しばしば再評価されません。明らかに、計算された数値の威力は、その数値が真実ではないという証拠よりもはるかに優れています。リスクアセスメントの真実とその根底にある仮定についての証拠を生成することは、あらゆる安全管理システムの重要な部分であるべきです。

事前に何が原因でおこるかを認識することは現実的ではないが、事後でも可能な限り調査されるべきであり、すべきでした。効果的な SMS は、時間の経過と共に安全コントロールストラクチャーの劣化に対する継続的な警戒を特徴とします。

共通のパラドックスは、組織の事故が少なくなればなるほど、将来的にはより多くの事故が発生する可能性が高くなるということです。 安全管理システムが有効であれば、損失はほとんどありません。 残念ながら、人々は、現在の事故率に関するリスクの本質的な量を判断する傾向があります。 事故が少ない、すなわち安全管理システムが非常に有効である場合、時間の経過とともに固有のリスクは低いと判断され、安全管理は低下し始め、効果が低下します。 意思決定者のプロセスモデルをいつでも実際のリスクレベルで最新の状態に保つためのフィードバックと情報が必要です。

リスク管理のヒント

- ハザードを識別し、分析し、その結果得られた情報を製品開発および運用に使用する手順を作成する。これが確実に行われるように責任を割り当てる。
- 定量的な評価だけでなく、改善に必要な情報を提供する定性的手順を重視する。 知られていないようなものについては、定量的な評価をしない。
- データが取得され、変更が時間とともに発生すると、リスク評価結果を再評価する。
- 当初想定されていたリスクよりもリスクが増加している時期を特定する方法を提供する。
- 手続きが遵守されていないときは、強制しようとしない。代わりに、追跡されていない理由を評価し、再設計する。指定されたものと実行されたものとの間のギャップが発生している理由を理解しよう。

教育とトレーニング

物理システムの下位レベルのコントローラだけでなく、安全コントロールストラクチャーのすべての人は、安全に関する役割と責任、およびシステムが安全コントロールストラクチャーの組織的側面を含む理由を理解していなければなりません。 従業員が SMS の意図を理解してそれにコミットした場合、従業員は都合の良いときに単にルールに従うのではなく、その意図に従う可能性が高くなります。 トレーニングでは不十分です。 教育が必要です。

教育には、統制によって強制される危険性や安全性の制約に関する情報だけでなく、優先順位や意思決定の方法についても含める必要があります。 前述の安全哲学のステートメントは、意思決定に使用される安全値に関する情報を提供します。 さらに、誰もが決定を下していくリスクを知る必要があります。 しばしば、意思決定の貧弱さは、リスクの誤った評価が想定されることから生じます。 上記で提供されたリスクの非確率的定義を使用することは、意思決定者がその決定が安全コントロールストラクチャーの設計されたコントロールにどのように影響するかを知る必要があることを意味します。

マネージャと従業員に「微弱なシグナルに注意を払う」ように指示すること（高信頼性組織や HRO 文献では共通の提案）は、たんに、ある損失イベントが発生した後で非難の口実を作り出すことになり、後知恵は振り返ったときに弱いシグナル（ノイズ）を強いシグナルへと変える明瞭さを与えます。代わりに、誰もが、システムの操作に関連する危険性について訓練を受けていなければなりません。 人々は、未定義のものを探すように言われるだけでなく、何を探すべきかを知る必要があります。

トレーニングには、「なぜ」と「何か」も含める必要があります。従うように求められている安全規則の背後にある理論的根拠を理解することは、独りよがり、無謀な行動（しかしその人にとっては理にかなっていること）に見えること、ハザードにつながる意図しない変更を減らす手助けをします。その根拠には、以前の事故がなぜ発生したか、そして再発を防ぐためにどのような変更が加えられたのかを理解することが含まれます。ますます自動化され、人々と相互作用しながら、制御された危険を理解することは、危険性がより明白で直感的だった過去に必要だったより多くの訓練を必要とするかもしれません。複雑なシステムと対話する人々は、従うべき手順だけでなく、コントロールされた物理的プロセスや、監督することができる自動化コントローラ（ソフトウエア）で使用されるロジックを深く理解し、相互作用している可能性があります。安全コントロールストラクチャーのすべてのレベルでどのコントローラを知る必要があるかのリストは、より安全な世界の設計に含まれており、ここで繰り返されています。

- システムの危険性および安全上重要な手順および運用ルールの背後にある理由。

- コントロールを削除またはオーバーライドし、規定の手順を変更した場合の潜在的な結果。安全に不可欠な機能や操作に不注意を感じることがある。：過去の事故とその原因は、見直され、理解されるべきである。
- フィードバックの解釈方法：トレーニングには、単一のイベントだけでなく、アラートと一連のイベントのさまざまな組み合わせを含める必要がある。
- 問題を解決する際に柔軟に考えるには：コントローラは、安全に関する問題解決を実践する機会を提供する必要がある。
- 特定の反応ではなく一般的な戦略：コントローラは、予期せぬ出来事に対処するスキルを開発する必要がある。
- 適切な方法で仮説をテストする方法：メンタルモデルを更新するために、ヒューマンコントローラは、現在のシステム状態を理解し、プロセスモデルを更新するために仮説テストを使用することがよくある。文書化は通常あまりにも貧弱で使いにくいので、自動化の設計や動作を理解するための唯一の方法であるようなコンピュータや自動化されたシステムでは、このような仮説テストは一般的である。 しかしながら、仮説テストは損失につながる可能性がある。 設計者は、仮説を安全にテストする能力をオペレータに提供する必要があり、コントローラはその仮説をどのように行うべきかについて教育を受けなければならない。
- 緊急時の処置は無視され、継続的に練習されなければならない。 コントローラは、動作限界値とそれを超える場合に取る特定の動作について教える必要がある。 オペレータがストレス下で十分な情報なしで良い決定を下すことを要求するだけでは、損失が発生した後に責任を負うことが保証されているようなものです。

研修は、従業員にとって一回限りのイベントではなく、責任とシステムハザードを念頭に置いている場合に限り、雇用を通じて継続して行われるべきです。 最近の出来事や傾向について学ぶことは、この訓練の一部でなければなりません。 おそらく、定期的な監査とパフォーマンスアセスメントを通じて、トレーニングの効果を評価することは、効果的な改善と学習プロセスを実施するのに役立ちます。 インシデントと事故調査の結果は、トレーニングの有効性に関する重要な情報源です。

一部の企業では、マネージャが安全トレーニングを提供するプラクティスを作成しています。 トレーニングの専門家は、グループのダイナミクスやカリキュラムの開発を管理するのに役立ちますが、トレーニングはプロジェクトリーダーやグループリーダーによって提供されます。 材料を教えることを学ぶことによって、監督者と管理者は、より重要な原則を吸収し実践する可能性が高くなります。 さらに、社員がトレーナーや安全エキスパートよりも上司からのメッセージにもっと注意を払うことがわかっています。

教育訓練のヒント

- 教育であって、単なる訓練ではない
- 誰もが自分の役割と責任、システムがそのように設計された理由、コントロールによって強制される危険と安全性の制約に関する情報、およびリスクに対する意思決定のリスクを理解する必要がある
- トレーニングには「なぜ」だけでなく「何か」を含める必要がある。
- 誰もが以下のことを学ぶべきである:
  - システムのハザードと、安全に不可欠な手順とその職務に関連する運用規則の背後にある理由
  - コントロールの削除または上書き、規定された手順の変更、安全上重要な機能や操作への不注意の潜在的な結果
- 過去の事故の原因がよく伝播され、理解されていることを確認する
- 安全にかかわる問題解決を練習する機会を人々に与える
- コントローラが予期せぬ出来事に対処するスキルを開発できるように、具体的な回答ではなく一般的な戦略を教える
- 仮説を適切な方法でテストする方法をオペレータに教え、このようにして学習する方法を提供する
- 緊急時の処置は、過度に認識され、継続的に実施されるべきである
- 緊急時の手順は、安全に不可欠な活動の前に見直されるべきである
- 訓練は、雇用されたときの 1 回限りのイベントではなく、継続的なものでなければならない
- 最近の出来事や傾向について教える
- 管理者に安全訓練を提供させることを検討する

学習と継続的改善:

SMS の設計は当初から完全ではなく、世界は時間の経過と共に変化するため、組織が安全インシデントから継続的に学習し、SMS 自体や職場、製品、およびサービスを改善するための効果的なプロセスが必要です。

実験は学習プロセスの重要な部分であり、新しいアイディアや安全性の向上への取り組みを奨励すべきですが、改善が実際に行われるように慎重に評価する必要があります。

安全情報システム (SIS)

包括的で使用可能な安全情報システムは、SMS を成功させるための鍵です。 これは、コントロールストラクチャーの有効性に関する情報源を提供します。 STAMP の用語では、これがリスクの定義です。

コントローラと意思決定者のプロセスモデルは、正確で、かつ調整されていなければなりません。 本質的に、SIS は共有プロセスモデルと個々のプロセスモデルを更新するソースとして機能します。 したがって、正確でタイムリーなフィードバックとデータが重要です。SIS は、事故の傾向、変化および他の前兆を検出するために、モデルとリスクアセスメントを実際の行動と比較するために、イベントから学び SMS を向上させるために、必要な情報を提供することができます。重大な事故の後、損失を防止するための情報は存在していたが使用されていないか、関係者は利用でき

なかったことがしばしば分かっています。 一般的には、政府の報告書に必要とされるから多くの情報が収集されているだけであり、必ずしも効果的な SMS の運用に必要であるからではありません。

SIS に何が含まれるべきかを判断するために、責任を果たす必要がある安全管理構造およびフィードバックの責任者は、適切な決定を下すためにプロセス/メンタルモデルに必要な情報を使用することができます。一般的に、最小限で、SIS には、以下が含まれます:

- 安全管理計画（開発と運用の両方）、
- システムに関するすべての安全関連活動のステータス
- 設計の基礎となる安全上の制約と前提条件（運用上の制限を含む）
- ハザード解析の結果（ハザードログ）と、既知のすべてのハザード、
- パフォーマンス監査と評価の結果
- インシデントおよび事故調査の報告および取られた是正措置
- 教訓と歴史的情報
- 傾向分析の結果

安全コントロールストラクチャー設計を使用して、安全な意思決定のために各人が必要とするものを特定することは、各人が必要な情報を見つけることができる有用な SIS を設計するのに役立ちます。

情報は、企業または産業によって収集されることがあります。どのように情報が収集されても、その限界を理解することは重要です。最も有用であるためには、情報は正確でタイムリーでなければならず、有用な形で適切な人々に伝えなければなりません。収集、分析、および普及という 3 つの要因が関与します。

収集: データが収集される方法によって歪むことがあります。一般的な問題は、系統的なフィルタリング、抑制、および信頼性の低下です。 事故報告のために収集されたデータは、近位の事象および当事者に集中する傾向があるが、管理上の問題または組織的な欠陥などの系統要因には関与していません。 研究[35]では、オペレータによるニアミス報告のデータがフィルタリングされ、主にオペレータのエラーをイベントの原因として指摘しています。 管理者によるレポートも同様にフィルタリングされ、原因としてオペレータエラーが示されます。 一般的な安全検査でさえ、特にチェックリストが使用されている場合には、限定された条件カテゴリを特定する傾向があります。

過去に発生した事故に類似した事故では、過去のハザードや因果関係の経験が限定されている新しいタイプのシステムよりも、データ収集の信頼性が向上する傾向があります

ソフトウエアのエラーとコンピュータの問題は、知識の欠如、そのようなエラーの受け入れられた一貫した分類の欠如、または単に真剣に因果関係の要素として考慮していないため、しばしば省略されるか、または不十分に記述されます。

経験によれば、事故の長期にわたる信頼性の高い報告を維持することは困難であることが示されているため、この種のデータは、事象の確率またはその潜在的結果を推定するのにあまり有用ではありません。一部の業種では、最大 4 分の 3 の事故が報告されない可能性があるというデータがあります。時には、会社が秘密を保つことを望んでいる、または報告書を作成している人が仕事の評価においてそれらに対して使用されることを心配するかもしれない情報が含まれることがあります。潜在的な法的責任も関係する可能性があります。

発生した操業上の問題についての議論は、SIS に文書化し、ハザードに至る可能性について分析すべきです。 これまで述べたように、SIS の文書を迂回して対処できる人に直接関わる懸念につい

[35] Many scientific references for this section can be found in Nancy G. Leveson, *Safeware*, Addison Wesley Publishers, 1995, Chapter 11. 私たちは、このハンドブックを学術論文のようには読ませないようにした。

ては、これらのディスカッションや電子メールには参加者が認識していない安全上の問題があることがよくあります。 さらに、これらの議論を文書化することは、人が職場を離れたり、電子メール保持基準のため退職する際に失われる貴重な企業知識を含んでいるので有益です。

これらの通信を保持し文書化するための要求を実装するか、またはそのような情報を自動的に保持する企業の通信チャネルを開発する必要があります。データコレクタの特別なトレーニング、データコレクタへの結果のフィードバック、固定ルーチン、および匿名のレポート作成など、データ収集の包括性と信頼性を向上させるためにいくつかの手段を使用できます。 CAST のような技術の訓練は、しばしば省略される情報の包含を促すことができます。 現在は自動化された監視システムが普及していますが、多くの情報を収集し、通常は結果を分析して問題を検出することが困難であるという問題があります。 STPA は、どの種類の情報を収集する必要があるかを判断し、収集された大量のデータの先行指標を特定するのに役立ちます。

分析: 大量のデータを体系化して学習に役立つ形にまとめることは困難です。 統計的有意性が含まれていない場合は、生の定量データが誤解を招く可能性があります。データは情報と同じではありません。 繰り返しになるが、STPA の結果は、収集する必要があるデータを特定するだけでなく、発生しているイベントの重要性に関するガイダンスを提供するためにも使用できます。 さらに、STPA の因果関係分析結果を検証し、排除または緩和されたと考えられる要因を特定するためのデータは、SIS の収集および分析プロセスの一部である必要があります。

分析の最大の問題は、データ収集チャネルを設計するのは簡単ですが、結果として得られるすべてのデータを分析する時間と人手を見つけることは困難または非実用的である可能性があることです。 STPA のようなツールは、収集する最も重要なデータと最も有用な分析プロセスを特定するのに役立ちます。

普及: 有用な形で情報（データではない）を普及させることは、SIS の最も難しい部分である可能性があります。 情報は、概念設計段階だけでなく、人々が学び、毎日の仕事に応え、プロジェクトのライフサイクルを通じて使用できる形で提示する必要があります。 また、タイムリーに更新する必要があります。事故は、操作中の変更やエンジニアリングの変更が行われたときのハザード解析の更新不足のために発生しています。 STPA は、変更プロセスを支援するための根拠とインテントとトレーサビリティを含む使用可能なドキュメントを強調しています。 また、安全管理構造内の個人およびグループへの安全関連の責任は、それを受け取った人々のニーズに合わせて情報を調整するのに必要な情報を提供します。 個人にあまりに多くの情報を提供することは、あまりにも少ない情報を提供することと同じくらい危険です。

情報を提示する方法は、ユーザーの認知スタイルに適応可能であり、安全関連の決定が行われる環境に統合されるべきです。 ここでも、安全コントロールストラクチャー設計プロセスは、必要な情報、必要なとき、およびコントロールストラクチャー内の個々のコントローラによってどのように使用されるかについての有用な情報を提供します。

安全情報システムの設計のヒント

- 正確でタイムリーなフィードバックとデータが重要である
- SIS は、事故の傾向、変化およびその他の前兆を検出するために、安全管理の有効性を評価するために、モデルとリスクアセスメントを実際の行動と比較するために、イベントから学び SMS を向上させるために、必要な情報を提供する必要がある。
- 安全管理構造の定義された責任を使用して、適切な意思決定のためにプロセスモデルを正確に保つために必要な情報を特定する
- 収集したデータの限界を理解する
- 最も有用であるためには、情報は正確でタイムリーでなければならず、有用な形で適切な人々に伝えなければならない
- データの歪みを最小限に抑えるデータの収集方法（フィルタリング、抑制、および信頼性の低いもの）を見つけます。
- 実際の安全関連の事故に関する詳細情報を保持し、情報が必要な人々に確実に届くようにする
- データ収集の理解と信頼性を改善する方法を作成する
- 可能であれば、数値データに統計的有意性を含めるようにする
- STPA を使用して、収集する必要があるデータを特定し、発生しているイベントの重要性に関するガイダンスを提供する
- STPA の因果関係の分析結果を検証し、排除または緩和されたと考えられる要因を特定するためにデータを収集する
- データが分析され、収集されるだけではないことを確認する
- 人々がそれから学び、毎日の仕事にそれを適用できるような方法でデータを提示する
- データを最新の状態に保つ
- 文書設計の根拠と意図と変更プロセスを支援するためのトレーサビリティを提供する
- 提供された情報とそれを受け取った人のニーズに合わせて提示形式を調整する
- あまりにも多くのデータ、特に生データを提供することは、あまりにも少なくデータを提供することと同じくらい危険である
- ユーザーの認知スタイルに情報の提示を適応させ、それを安全関連の決定が行われる環境に統合する
- 安全制御設計プロセスを使用して、必要な情報、必要な時期、および使用方法を決定する

要旨

一般的に、効果的な安全管理は:

- すべてのレベルのおけるコミットメントとリーダーシップ
- 強力な企業安全文化
- ステークホルダーの間で共有されている、はっきりと明解な安全ビジョン、価値観、手順
- 責任、権限、説明責任の適切な割り当てによる安全管理構造
- 安全コントロールストラクチャーのすべてのレベルでの安全状態の正確なビューを提供するフィードバックチャネル
- 開発とラインオペレーションへの安全の統合（単に個別の独立したグループまたは別個のサブカルチャーではない）

- 適切な知識、スキル、能力を持つ個人
- パートナーシップの役割と責任を持つステークホルダー
- 安全の優先順位と他の優先順位との間の敵対意識を解決するための指定されたプロセス
- 安全情報の普及のためのリスク意識とコミュニケーション・チャネル
- リスクの高い方へのシステム移行のコントロール
- 効果的かつ有用な安全情報システム
- 継続的な改善と学習
- 教育、訓練、能力開発

詳細については、”Engineering a Safer World”の第 13 章と”Safeware”の第 11 章と第 12 章を参照してください。 本ハンドブックの付録 D には、効果的な安全管理構造に含める必要がある責任のリストも記載されています。

第 8 章：あなたの組織への STPA の導入

ジョン・トーマス
ナンシー・レブソン

比較的小さな組織において、STPA を導入するには、STPA を用いることによる実用性と利点に価値があることを確証するために、愚直にトレーニングとおそらく実証プロジェクトを取り組むとよいでしょう。より大きな組織には、文化を変える、使用される標準プロセスを変える、従来とは異なる新しいことをするための大人数のトレーニングを行う等の、他の課題があるかもしれません。この章では、組織の中で広く使用されるために、組織が STPA を評価または適用することを決めた際に生じる課題を検討します。

トレーニング

STPA を習得するためには、いくつかの練習と「実践で習う」アプローチが必要です。正式には、対面トレーニングは非常に有効で、たった数日で専門家が育成され、STPA を開始する準備ができることがわかりました。最も効果的な STPA トレーニングは、プロセスを強化するための実践的な演習による対話式です。驚くべきことに、従来のハザード解析手法を用いている人は STPA を習得することに困難を感じることが多く、ある一定量の知識の捨て去りと、追加のトレーニングが必要な場合があるようです。我々の経験から、システムエンジニアは最も早く習得する傾向があります。

トレーニングクラスの規模に依存したトレードオフがあります。小規模なクラス（１０名以下）では STPA の深くて詳細な理解を提供でき、STPA の評価のために小さなパイロットスタディを初めたい、小さな組織にとっての良いオプションとなります。大規模なクラス（５０名以上）は時間をより要し、可能な対話は限られますが、大きなグループに STPA の基本的な認識を提供する必要がある組織にとっては、良いオプションとなります。中規模なクラス（２０から３０名程度）はリーズナブルな妥協点を提供する傾向があります。

何百あるいは何千者もの人をトレーニングする必要があるとても大きな組織では、「トレーナーをトレーニングする」アプローチが望ましいことがあります。将来のトレーナーやファシリテータ（下記を参照）を担える STPA の専門家を作る最も効果的な方法は、STPA が活発に使われている実プロジェクトに候補者を投入することです。短いトレーニングクラスに参加するだけでは STPA の専門家を作るには足りないが、大きな STPA プロジェクト数件に投入された人は、将来のトレーナーやファシリテータとして良い候補者となります。上手くいったことがある一つのアプローチは、１人または複数のファシリテータを、実プロジェクトで働いている STPA ファシリテータに同行させてトレーニングすることです。トレーニングでは、ファシリテータは、異なるプロジェクトで出た課題や、上がった疑問点を直に見ることで多くを学びます。彼らが効果的なガイダンスを提供し、課題を解決し、他の参加者からの詳細な STPA の質問に回答する能力を実演することにより、自分達（のプロジェクト）でファシリテートする準備ができるかもしれません。

このハンドブックは、今までなかった時には提供できなかった、より多くのトレーニングオプションを提供できるはずです。加えて、このハンドブックと連動する動画を作成しています。

ファシリテータ

STPA が最も有用なのは、別のグループにより独立に行われる場合でなく、設計作業に加えられたときです。しかし、誰もが、STPA の専門家として教育を希望したり、必要とされたりするものではありません。HAZOP で利用されるファシリテータというアイディアについては、システム設計の専門家は STPA の参加者として最適ですが、彼らが（STPA に）慣れていなければ技術を適用するためのガイダンスが必要な場合があるという同様の理由で、私達も STPA において推奨します。

STPA ファシリテータの仕事は、STPA の専門知識やガイダンスをプロセスに沿って提供することと、発生し得る共通的な落とし穴を避けるのを助けることです。ファシリテータは、プロジェクトの選択とスコーピングの最中にガイダンスを提供することができます。（例えば）初期のチームメンバーが必要となる専門知識を識別するのを助け、過去の類似のプロジェクトに基づいて、損失、ハザードや潜在的な最初のコントロールストラクチャーなどの初期の一式を作成でき、プロセスに沿って全体的な調整を提供できます。大規模なプロジェクトでは、ファシリテータは個人や小さなグループの専門家でうまく実行できる小さな個別部品にプロセスを分割します。ファシリテータは、異なるグループで並行して実施できるタスクを識別し、スケジューリングを行い、最終的にファシリテータはこの個別部品を全体的な解析の中にまとめます。また、ファシリテータは参加者による会議をリードし、チームメンバーのやり取りを管理します。

ファシリテータは、プロジェクト開始時に初期の STPA トレーニングを提供することもできますが、プロジェクト進行中に発生した STPA に関連するいかなる質問にも答える必要があります。ファシリテータは、一般的には、プロセス中に結果をレビューし、プロセスの最後には手法が正しく行われたか、ずれが見落とされていないかを確かめます。プロジェクトの終結時に、ファシリテータは結果を最終レポートにまとめ、識別された問題が適切な人に提供されたことを確かめることになります。

チーム構成

STPA は異なるタイプの専門家を含む小さなチームで実行されることが最も良いです。最初のコントロールストラクチャは、必要となる専門家の領域を特定するのに役立ちます。学際的なチームは、ソフトウエアの専門家、運用の専門家、保守の専門家等、異なる領域の専門家を典型的に含みます。いくつかのケースでは、全ての関連する専門家を STPA のチームに含めることができないことがあります（例えば、専門家がひっぱりだこの時）。この場合、チームは少なくとも正しい技術的知識を持つ人にアクセスできること、分析中に生じる質問に対する回答を得ることができることを確実にすることをお勧めします。

パーソナリティも重要です。エンジニアやシステム設計者は最も専門知識を持っているかも知れないが、彼らは時々保守的になり、STPA の作業は彼らの仕事の品質に対して挑むものと考えることがあります。彼らは、設計でもうすでに除去されていると思っているか、または、識別された UCAs は現実的ではなく、除去されるべきで分析対象とならないと主張して、潜在的な UCA を提供したくないかもしれません。一方、この特定の設計に関係していない部外者や他の設計者は、システムに対する知識は低いかも知れないが、そこまで保守的ではなく、潜在的な欠陥を喜んで提案するかもしれません。エンジニアや設計者には、STPA はワーストケースを分析する手法であって、手戻りを避けることで仕事を厳しくせずにより容易にするように手助けできると説明することが役に立つかもしれません。設計が完了したり意思決定がなされる前に、STPA の適用をなるべく早く開始することも有益です。最も強みが出るのは、意思決定を促し、情報に基づいた選択をするために、プロジェクトのエンジニア自身で STPA を適用する時です。どんなケースでも、パーソナリティの問題はあり、新しいアプローチに対して、オープンな知識豊富な専門家から、過去に見落としてきた問題を識別することに躊躇しない人をみつけることが理想です。

プロジェクトの設計に深く関与することによって、彼らが以前に識別できていなかった UCA やシナリオを発見すると、初期の懐疑的態度を急速に克服していることに、よく気づかされます。そして、彼らはすぐに STPA プロモーターになります。

テストのバックグラウンドや経験を持つ人は、STPA のチーム員の一人として優れた選択となりえます。テスト関係者は、どのようなことが悪い方向へ行くのかの知識が極めて豊富であり、見落としがちな問題を見つけることに対して、望むだけではなく、熱望しています。テスト関係者は、要求、仕様書または設計に対する意思決定が正しいと想定することは滅多になく、彼らの仕事は作

られた想定や主張を継続的に疑問に思うことです。これらは STPA のチーム員としての素晴らしい特徴です。

コストと投資対効果

STPA は非常に効率的です。*図 8.1* は近年の産業界のプロジェクトにおいて、全ての STPA のチーム員が費やすた時間の相対量を示しています。多くの産業界の STPA プロジェクトでは、その時間のほとんどは、STPA を習得したり、STPA を実行するのではなく、分析するシステムがどのように動くのか理解すること（ハザードの原因を識別するためにどのような手法でも必要なこと）に費やされています。およそ 1/4 の時間が、過去に考慮されていこなかった、システムについての「もしこうなったら」という技術的な疑問に対する回答を見つけることに費やされます。比較的少ない時間が、典型的に STPA の習得やプロセスの実行に費やされます。

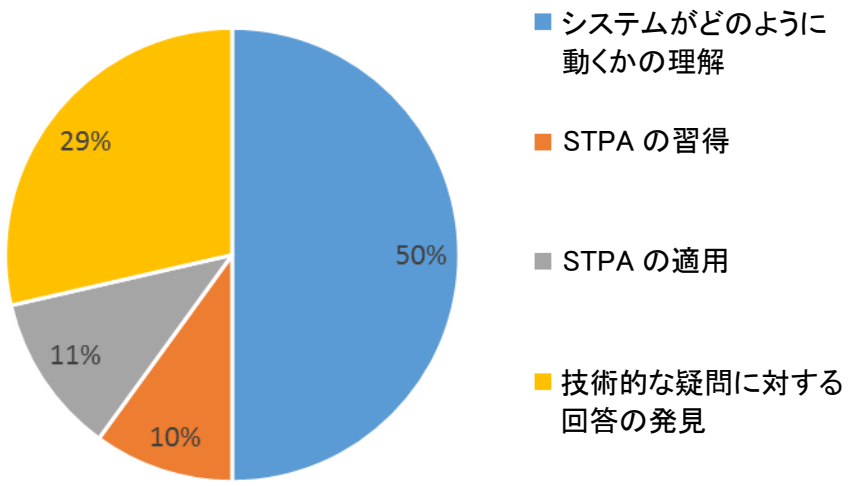


図 8.1 近年の STPA プロジェクトにおける、異なるタスクに費やされる時間の相対量

STPA は論文を書くための練習問題ではありません。多くの時間は、どのように改善すれば良いのかといった洞察の提供や、意思決定のために直ちに使うことができるような、システムの実懸念を探すことに費やされます。

STPA に必要な時間について比較するような計画的に収集したデータを多く有しているわけではありませんが、人々は、STPA が従来のハザード解析手法と比較して必要となる時間とリソースは、2～3 桁の規模で少なく、より完全な結果が得られると報告しています。一般的に、習得コストのために初回はコストが高くつきますが、プロジェクトを繰り返すことでコストはすぐに下がってきます。

大規模な組織での STPA の導入例

第４章では、ある大規模な組織で、複雑な高度に自動化されたシステムで STPA での使用について説明しています。ここではそれを繰り返します。このケースでは、STPA は、インテグレーション、エンジニアリング、運用と保守からのメンバーで構成され、８人のメンバーの横断的機能を持つチームにより適用されました。参加した全ての人々は、システムに対して、彼らの責任範囲に合わせて、リーダーや内容に関する専門家とみなされました。ファシリテータは、STPA について訓練されていたが、工場に導入される新しい全自動の製造プロセスについての知識はほとんどなかった MIT の修士学生でした。他のチームメンバーはプロジェクトの初めには STPA についての知識はありませんでした。

この分析は３週間のコースで完了しました。グループでの評価は、プロセス中に２回だけ、それぞれ３時間以下で行われました。残りの評価は、ファシリテータと議論対象のシステムの部分に対して適切な専門家との１対１のミーティングで完了しました。

STAMP と STPA は、企業にとって比較的新しく、チームにとっては全く新しかったので、最初のグループミーティングは、使用する手法の概要を説明するため、および、チームが防止したい事故（または損失）、ハザードとシステム境界、このアプリケーションのためのセーフティコントロールストラクチャの初期草案を定義するためのインストラクション期間として使用されました。

初めのコントロールストラクチャは、主要コンポーネントが全て捉えられていることを確認するために、グループ全員で作成しました。コントロールストラクチャの初期草案を作成した後、詳細を付け加えたいシステムの異なる部分に対する適切な技術専門家との個人や小グループでのフォローアップミーティングのスケジュールを設定しました。

適切なレベルでの詳細なコントロールストラクチャが作成されると、１対１や小グループでの議論が非安全なコントロールアクションとシナリオの識別のために続きました。この個別部分での解析結果は他のグループにおいて議論され、グループの個別の気づき事項を確かめ、システムのための潜在的な緩和策を議論しました。このプロセスは３週間のコースで行われ、合計約 300 人時がグループ、１対１のミーティング、ファシリテータ個人の作業として費やされました。

一般的に、識別された因果関係シナリオは、ハードウエア故障、誤ったプロセスモデル（欠落したまたは誤ったフィードバックにより発生）、システムコンポーネントの相互作用、そして工場のスケジュールプレッシャーから影響を受ける安全に関する管理と手順の役割が含まれます。チームが STPA のシナリオを識別した後、結果を利用して、ハザーダスな状態に入ることの除去または回避する目的でシステムに新しいコントロールを構築しました。例はこのハンドブックの第４章に示しています。

STPA はコントロールストラクチャーのモデルを用いるため、その評価は完全なモデルが構築された後に、特定のコントロールループに焦点を当てることができます。このモデリングと分析プロセスを分割できることにより、ミーティング時間のスケジューリングを容易くし、グループ全員の可能な時間の中での数多くのミーティングを設けることができました。これらのミーティング中に、コントロールアクション、プロセスモデル、フィードバック等の詳細が加えられ、影響を受ける関係者で、競合や課題を解決することができました。

その他のアドバイス

投資対効果(ROI)のデータも含み、STPA を大きな組織に適用する際の情報をもっと準備する予定です。数多くの組織が STPA をいくつかのプロジェクトで試し、既存の方法と比較した後に、STPA を採用しているという事実には、自信を与えられます。

付録 A：ハザードの例

ハザードは、業界内では非常によく似ているものとなります。自分の業界、製品、あるいはサービスに特有のハザードを識別したら、おそらく小さな変更を加えて、そのリストを再利用することができます。この付録には、実際のプロジェクトで使用されているハザードのリスト例を添付しています。もちろん、利害関係者（ステークホルダー）は、どのような種類の損失やハザードを考慮すべきかについての最終的な意思決定者です。特定のプロジェクトの利害関係者にとって重要だと考えられる損失は、明らかに、識別されるハザードに影響を与えます。

原子力発電所

損失：

L1: 人間の負傷あるいは死亡

L2: 環境が汚染される

L3: 機器の損害（経済的損失）

L4: 電力損失

ハザード

H1: 放射性物質の放出 [L1, L2, L3, L4]

H2: 原子炉の温度が高すぎる [L1, L2, L3, L4]

H3: 限界を超えて運転している機器 [L3, L4]

H4: 原子炉の停止 [L4]

航空機

損失：

L1. 生命の喪失または重傷者

L2. 航空機または航空機外部の物体への損害

ハザード

H-1: 航空機が飛行中の最小間隔基準に違反する [L1, L2]

H-2: (軍事的)地域への航空機のコントロールされたフライト [L1, L2]

H-3: 航空機コントロールの喪失 [L1, L2]

H-4: 航空機の機体の完全性が失われる [L1, L2]

H-5: 航空機の環境が人の健康に有害である [L1, L2]

H-6: 航空機が、地上で指定された誘導路、滑走路、またはエプロンを逸脱する [L1, L2]

H-7: 航空機が地上の他の物体に接近しすぎている [L1, L2]

放射線療法

損失：

L1: 患者が過度の曝露または不適切な治療によって傷害または死亡

L2: 非患者が放射線によって傷つく、あるいは殺される

L3: 機器の障害または喪失

L4: 治療中の患者または非患者への物理的損傷

ハザード：

H1: 間違った照射：患者に施される放射線照射で、その量、位置、タイミングを間違える[L1]

H1a: 正しい患者、正しい照射、間違った位置

H1b: 正しい患者、間違った照射、正しい場所

H1c: 正しい患者、間違った照射、間違った位置

H1d: 間違った患者

H2: 非患者が不必要に放射線に曝される [L2]

H3: 機器が不必要なストレスを受ける [L3].

H4: 人が非放射線による傷害を受ける [L4].

軍用航空機

事故

M-1: 航空機または航空機の機器の喪失または障害

M-2: 重傷の人身事故または死亡事故

M-3: ミッションを完了できない

ハザード

H-1: 固定物体または移動物体からの最小分離基準違反 [M-1, M-2, M-3]

H-2: 航空機のコントロール不能 [M-1, M-2, M-3]

H-3: 機体完全性の喪失 [M-1, M-2, M-3]

航空機の任務に応じては、追加のハザードが適しているかもしれません。例として、 航空機の武器システムの場合は以下の通り：

H-4: 指令されていない爆発 [M-1, M-2, M-3]

H-5: 指令されていない発射 [M-1, M-2, M-3]

H-6:巻き添え被害または味方からの誤射 [M-1, M-2, M-3]

H-7: 命令された時の布告の非展開（爆発および/または発射） [M-3]

自動車

損失

L1: 生命の喪失または重傷者

L2: 車両または車両外の物への損害

ハザード

H1: 車両が近くの対象物から安全な距離を維持しない [L1, L2]

H2: 車両が危険領域/地域に入る [L1, L2]

H3: 車両が環境（速度、横方向/縦方向の力）に対して安全な動作エンベロープを超えている [L1, L2]

H4: 有害な影響および/または健康ハザードに曝された同乗者 [L1, L2]
（例えば、火災、過度の温度、逃げることができない、乗客のドアを閉じるなど）

付録 B：機能的コントロールストラクチャーの例

この付録では、私たちのプロジェクトで使っているコントロールストラクチャーの例を示します。

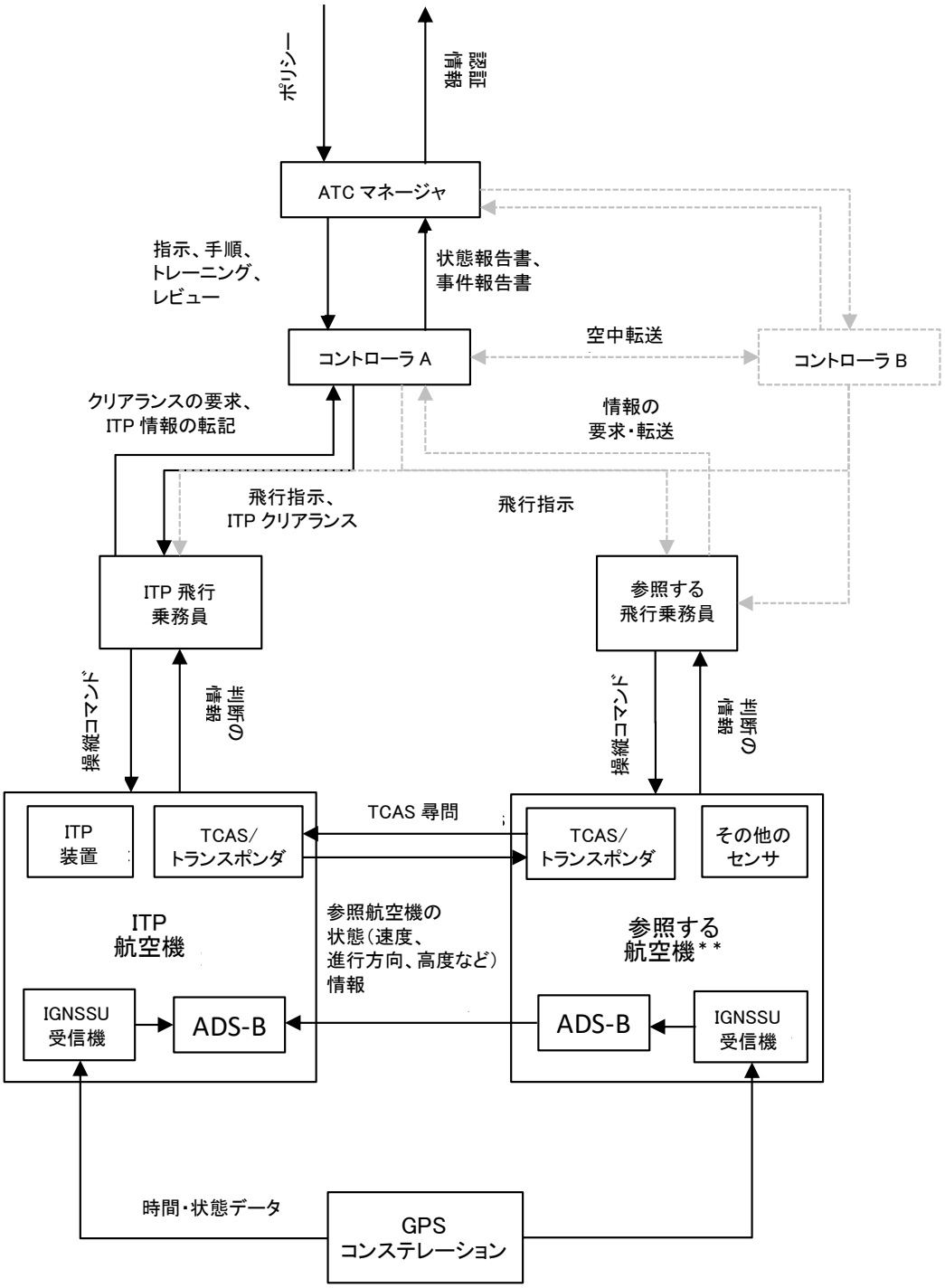


図 B.1: 新装置（航空）での NextGen イントレール手順のコントロールストラクチャー

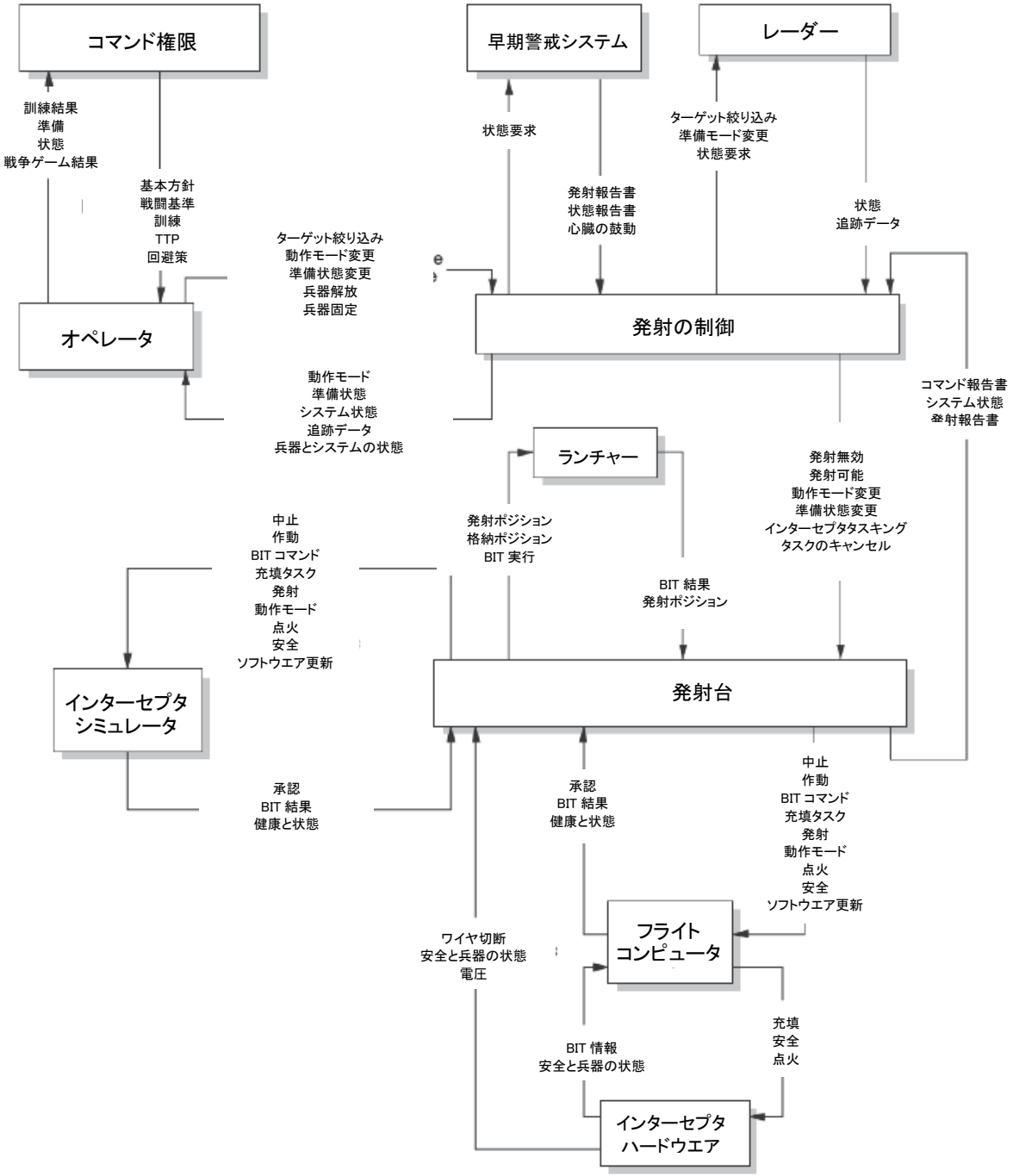


図 B.2: 架空のミサイル迎撃システムのコントロールストラクチャー

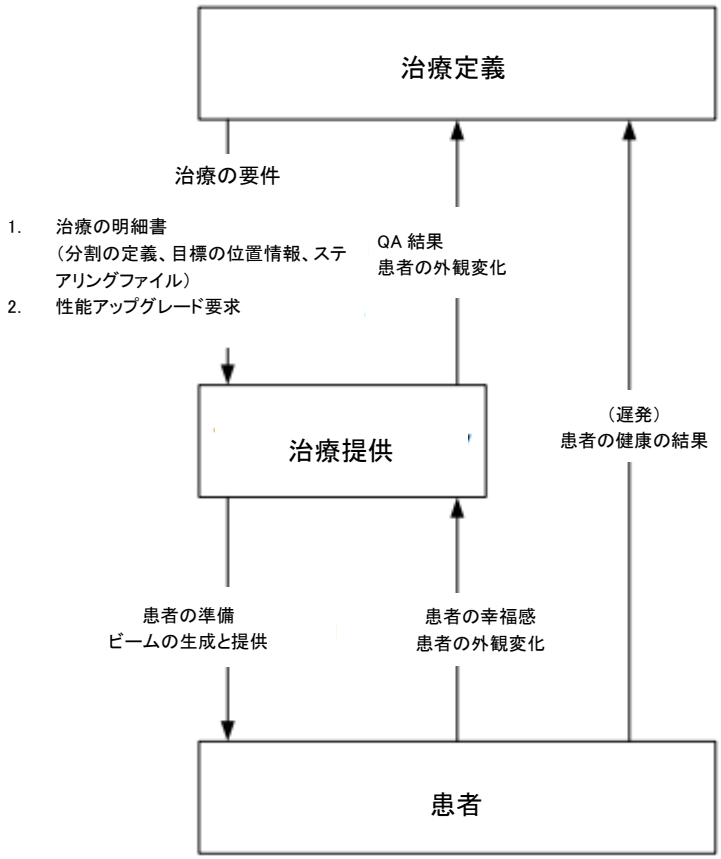


図 B.3: プロトン治療器の高レベル（抽象化）コントロールストラクチャー

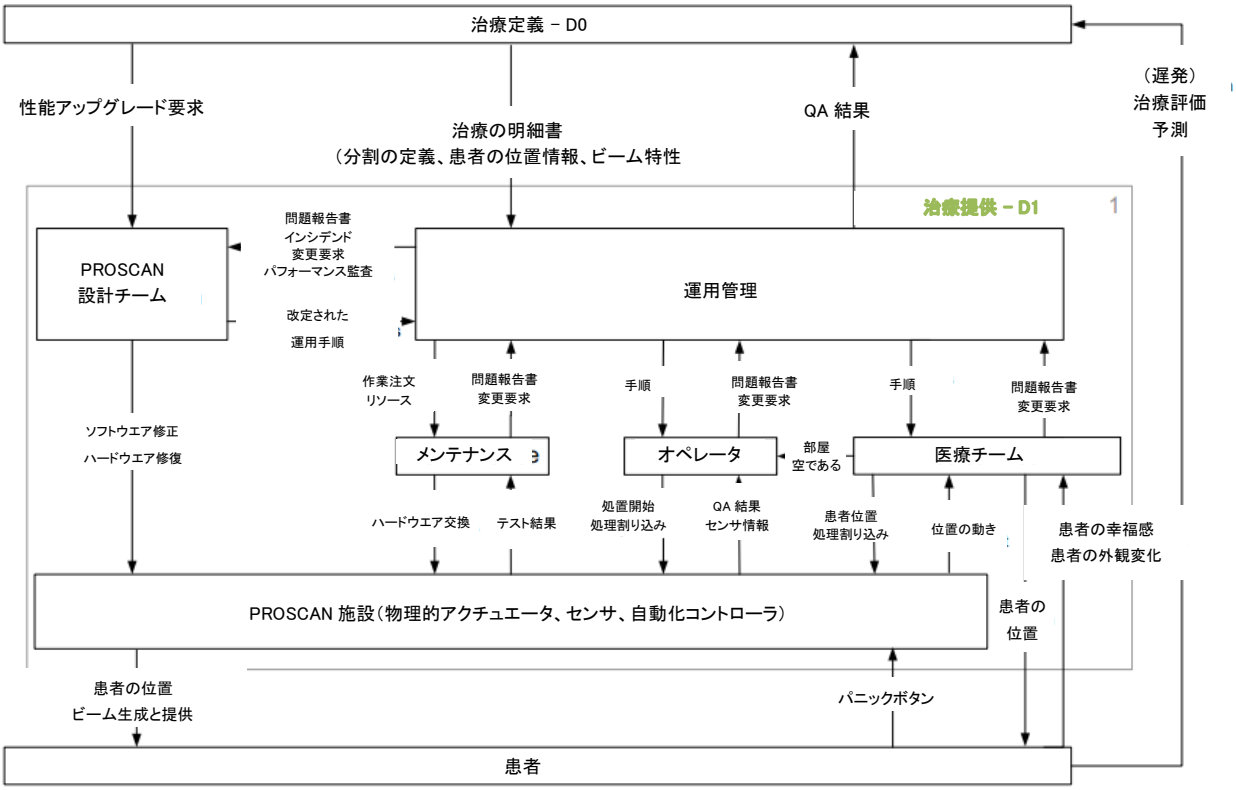


図 B.4: コンテキストを与えるために表現した治療定義と患者のコンポーネントを持つ図 B.3 の治療提供部分を拡大した図

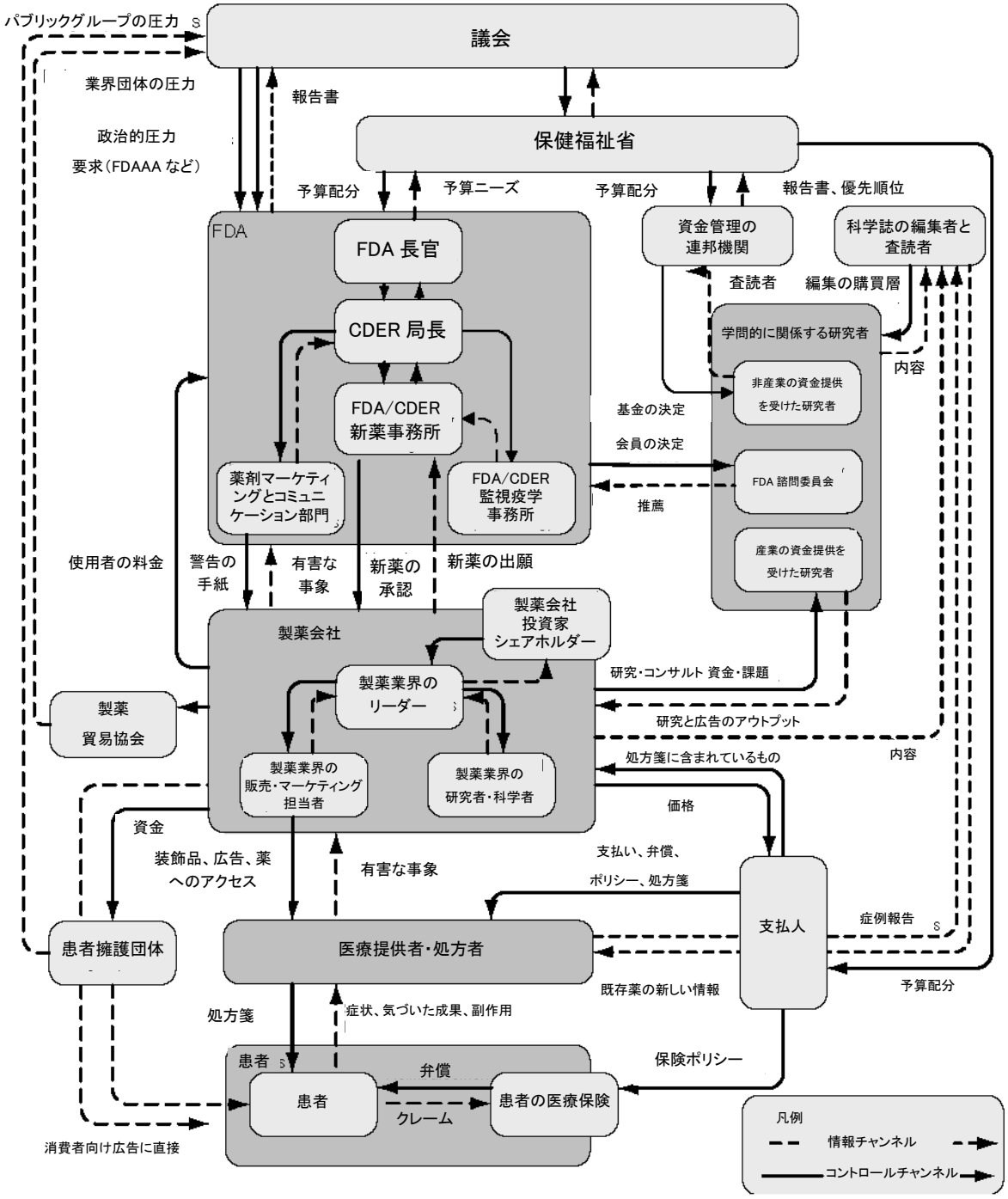


図 B.5: アメリカ製薬承認安全性コントロールストラクチャー

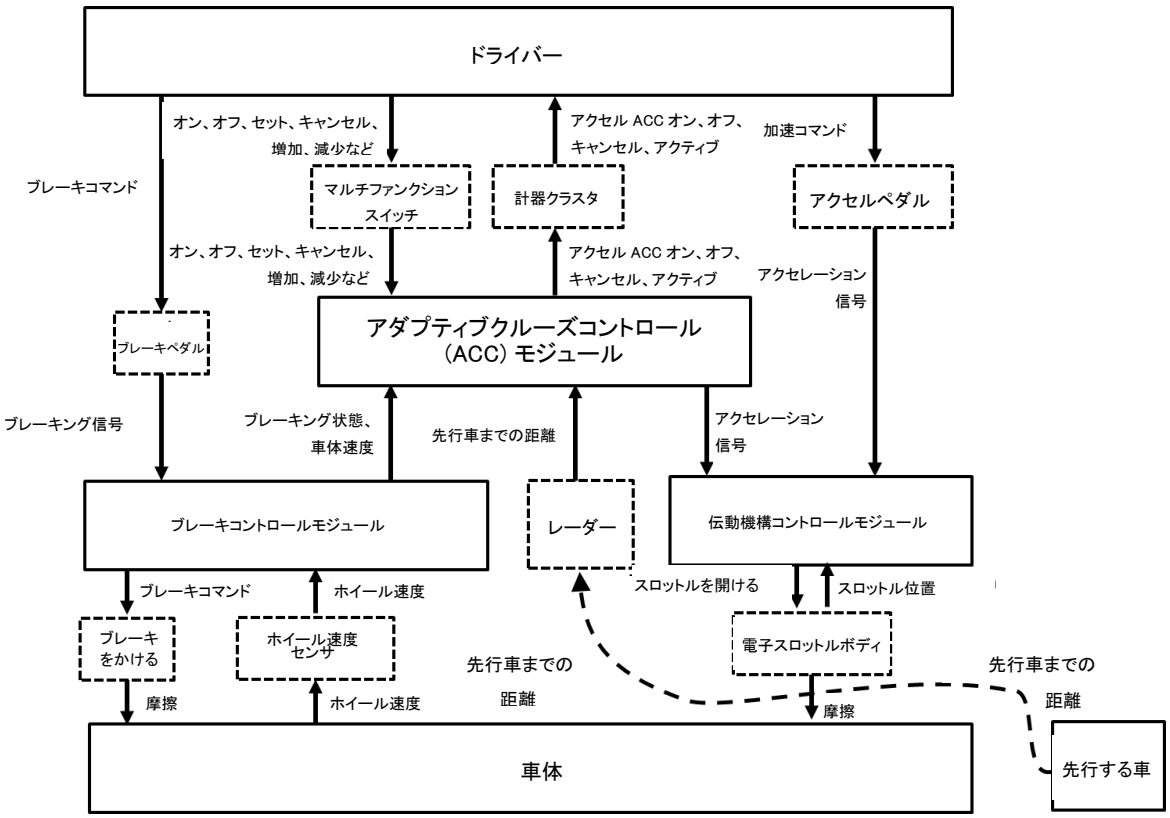


図 B.6: 自動車のアダプティブクルーズコントロール（定速走行・車間距離制御装置）システム

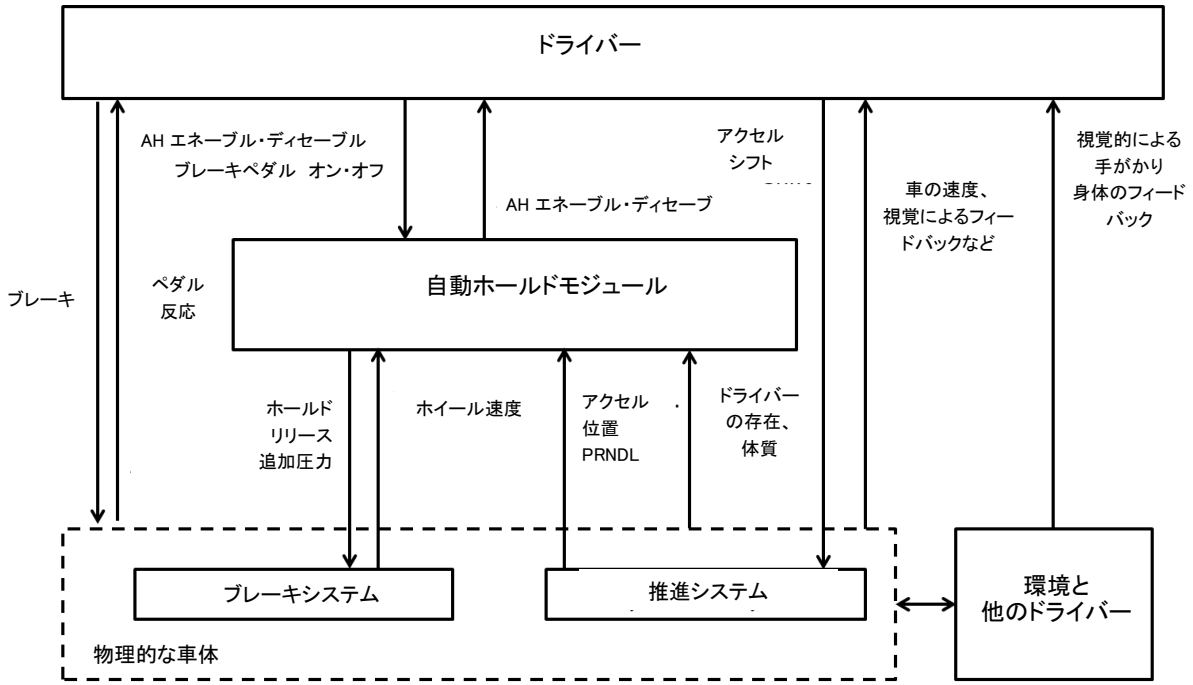


図 B.7: 自動車の自動ホールドシステムのためのコントロールストラクチャー

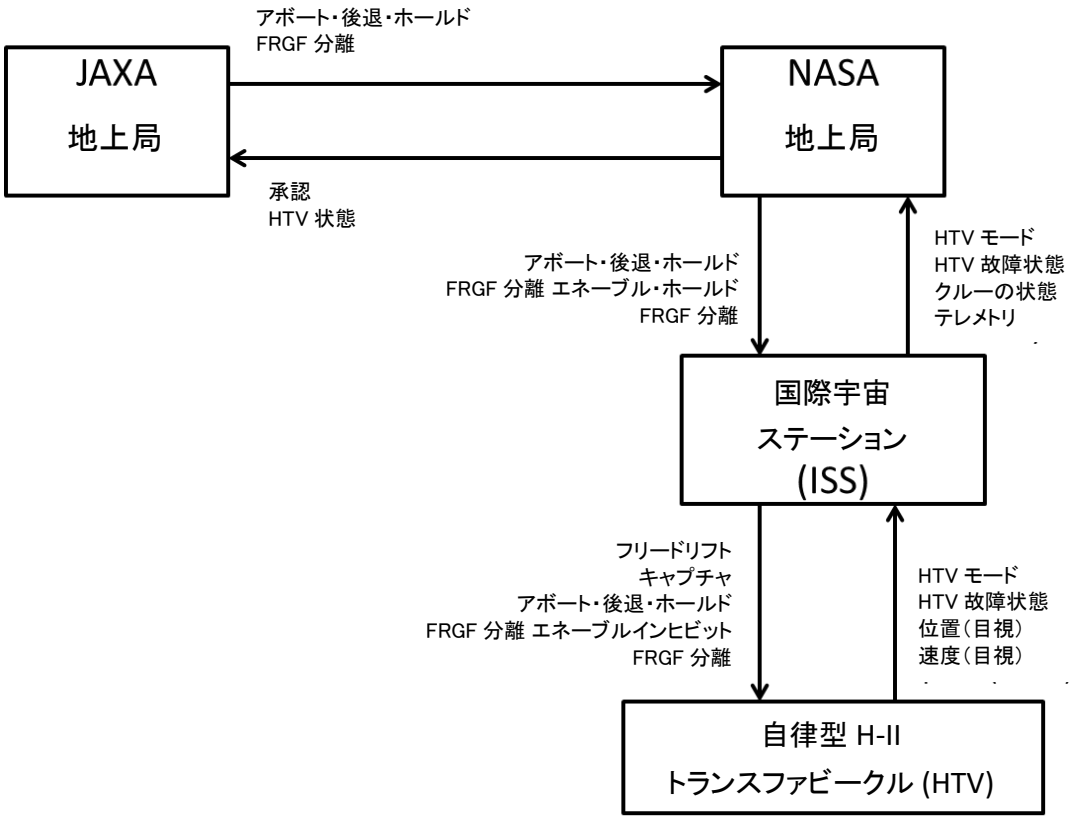


図 B.8: 自律型宇宙機オペレーションのためのコントロールストラクチャー例

付録 C：UCA 表の例

図 C.1: 自動車オートホールドシステム AHS の UCA

コントロールアクション(オートホールドによる)	与えられないとハザード	与えられるとハザード	誤ったタイミングや順序	早過ぎる停止、長過ぎる適用でハザード
ホールド Hold コマンド	UCA-AH-1: 車体が止まってブレーキペダルが離された時に AH が HOLD を出す [H-1,2]	UCA-AH-2: ドライバーがアクセレータを適用している時に AH が HOLD を出す [H-1] UCA-AH-3: AH がディセーブル DISABLED の時に AH が HOLD を出す[H-1] UCA-AH-4: 車体が動いている時に AH がホールド HOLD を出す [H-1] UCA-AH-5: ドライバーがブレーキを出していない時に AH が HOLD を出す [H-1,2]	UCA-AH-6: 静止に必要とされる時間よりも前、あまりにも早く AH が HOLD を出す[H-1] UCA-AH-7: 車体が止まった後に AH が HOLD をあまりにも遅く出して、車体が転がり始める [H-1]	N/A
リリース Release コマンド	UCA-AH-10: ドライバーがアクセレータペダルを適用する時に AH が RELEASE を出さない [H-1,2]	UCA-AH-12: ドライバーがアクセレータペダルを適用していない時に AH が RELEASE を出す[H-1,2]	UCA-AH-13: 十分なホイールトルクになる前に AH があまりにも早く RELEASE を出す[H-1,2] UCA-AH-13: アクセレータが適用された後に AH があまりにも遅く RELEASE を出してエンジントルクがホイールトルクを超える[H-1,2]	N/A
追加圧力 Additional Pressure コマンド	UCA-AH-14: 車体が滑って AH がアクティブである時 AH が ADDITIONALPRESSURE を出さない [H-1,2]	UCA-AH-15: AH がアクティブでない時 AH が ADDITIONALPRESSURE を出す [H-1,2] UCA-AH-16: ブレーキシステム仕様を超えている時に AH が ADDITIONALPRESSURE を出す [H-1,2]	UCA-AH-16: 車体が滑っている後で AH が ADDITIONAL-PRESSURE を出すのが遅すぎる[H-1,2]	N/A

表 C.2: 自律型 H-II トランスファビークル (HTV) のオペレーションのための UCA

コントロールアクション (ISS クルーによる)	与えられないとハザード	与えられるとハザード	早すぎる・遅すぎる・誤順序	早すぎる停止・長すぎる適用
アボート Abort	緊急条件が存在する時に ISS クルーがアボート Abort コマンドを与えない[H-1]	HTV がキャプチャされた時に ISS クルーがアボート Abort コマンドを出す[H-1] ISS が Abort コマンドパスにある時 ISS クルーが Abort コマンドを出す[H-1]	ISS クルーが Abort コマンドをあまりにも遅く出しすぎて衝突を回避できない [H-1] キャプチャがリリースされる前に ISS クルーが Abort コマンドをあまりにも早く出す [H-1]	N/A
フリードリフト Free Drift	HTV がキャプチャボックス内に停止している時に ISS クルーがフリードリフト Free Drift コマンドを出さない [H-1]	HTV が ISS に接近しているときに ISS クルーがフリードリフト Free Drift コマンドを出す [H-1]	HTV の停止から X 分以上後にあまりにも遅く ISS クルーがフリードリフト Free Drift コマンドを出す [H-1] HTV の停止前あまりにも早く ISS クルーがフリードリフト Free Drift コマンドを出す[H-1]	N/A
キャプチャ Capture	HTV がフリードリフトでキャプチャボックスにある時 ISS クルーがキャプチャ Capture を実行しない [H-1]	HTV がフリードリフト中でない時に ISS クルーがキャプチャ Capture を実行する[H-1] HTV がアボート時に ISS クルーがキャプチャ Capture を実行する[H-1] 過度・過少の動きで ISS クルーがキャプチャ Capture を実行する（HTV にインパクトを与え、衝突コースの原因になる） [H-1]	HTV のディアクティベート後 X 分以上後にあまりにも遅く ISS クルーがキャプチャ Capture を実行する [H-1] HTV のディアクティベート前あまりにも早く ISS クルーがキャプチャ Capture を実行する[H-1]	緊急状態が存在している後あまりにも長く ISS クルーがキャプチャ Capture を実行し続ける [H-1]

表 C.3: ホイールブレーキングシステムに関連する航空機フライトクルーのための UCA

コントロールアクション（フライトクルーによる）	与えられないとハザード	与えられるとハザード	早すぎる・遅すぎる・誤順序	早すぎる停止・長すぎる適用
CREW.1 ブレーキペダルによるマニュアルのブレーキ	CREW.1a1: 自動ブレーキがブレーキを出さないか、不十分なブレーキである場合、着陸時、RTO、タクシー中にクルーがマニュアルブレーキを出さない [H4.1]	CREW.1b1: クルーがマニュアルブレーキを与えたが、ペダル圧力が不十分[H4.1]	CREW.1c1: タッチダウン前にクルーがマニュアルブレーキを与える（ホイールのロックアップ、制御不能、タイヤのバーストの原因）[H4.1]	CREW.1d1: 安全なタクシー速度（TBD）に達する前にクルーがマニュアルブレーキコマンドを出すことをやめる [H4.1, H4.4]
		CREW.1b2: クルーが過度のペダル圧力でマニュアルブレーキをかける（結果として、制御喪失、乗客・クルーの傷害、ブレーキの過熱、ブレーキのシフェード、着陸時タイヤの破裂） [H4-1, H4-5]	CREW.1.c2: クルーがマニュアルブレーキをかけるのが遅すぎ（TBD）他の物体との衝突やコンフリクトを避けることができない（与えられた航空機の重量、速度、対象物までの距離（コンフリクト）、タールマック条件でブレーキ性能をこえる） [H4-1, H4-5]	CREW.1d2: クルーがマニュアルブレーキを長くかけすぎる（その結果、滑走路あるいはアクティブなタクシー道路上で止まる） [H4-1]
		CREW.1b3: 通常の離陸時にクルーがマニュアルのブレーキをかける[H4-2, H4-5]		
CREW.2 自動ブレーキを作動する	CREW.2a1: 着陸前にクルーが自動ブレーキを作動させない（スポイラ展開時、自動ブレーキ作動の喪失の原因になる。クルーの反応時間はオーバーシュートにつながる）[H4-1, H4-5]	CREW.2b1: 離陸時にクルーが最大レベルに達するまで自動ブレーキを作動させない（離陸中止には最大ブレーキ力が必要と推測）[H4-2]	CREW.2c1: クルーが作動コマンドを出すのが遅すぎる（TBD）（ブレーキを出すための BSCU に十分な時間がない）[H4-1, H4-5]	
	Crew.2a2: 離陸に先立ってクルーが自動ブレーキを作動させない（離陸中止の間にブレーキが不十分になり、 クルーがスロットルを下げた後の RTO 中に自動ブレーキはブレーキに責任があると推測している）[H4-2]	CREW.2b2: 滑走路条件の減速率よりも高すぎたオードブレーキをクルーが作動させる（制御不能になり乗客やクルーの傷害になる）[H4-1, H4-5]		

CREW.3 **自動ブレーキの作動解除**	CREW.3a1: TOGA 中にクルーが自動ブレーキの作動解除をしない（（再）離陸時の加速喪失になる）[H4-1, H4-2, H4-5]	CREW.3b1: 着陸あるいは TOGA 中にクルーが自動ブレーキの作動解除をする（スポイラ展開時の自動ブレーキ操作の喪失となる。クルーの反応時間はオーバーシュートにつながる） [H4-1, H4-5]	CREW.3c1: クルーが、(a) 航空機の降下が TBD fps を超えた後、(b)視界が TBD ft 以下、(c)などの TBD 秒以上後に、自動ブレーキの作動解除をする（航空機の制御不能または（再）離陸時の加速喪失になる） [H4-1, H4-2, H4-5]	
CREW.4 **BSCU 電源オフ**	CREW.4a1: 異常な WBS 動作の場合にクルーが BSCU の電源をオフにしない（代替のブレーキモードを可能にするために必要とされる）[H4-1, H4-2, H4-5]	CREW.4b1: 自動ブレーキが必要とされていて WBS が正常に動作している時に、クルーが BSCU の電源をオフにする [H4-1, H4-5]	CREW.4c1: 異常な WBS 動作の場合にクルーが BSCU の電源をオフにするのが遅すぎて(TBD)代替のブレーキモードが可能にできない[H4-1, H4-5]	N/A
		CREW.4b2: 自動ブレーキが必要とされて（または使用されようとして）いて WBS がもしも正常に動作している場合、クルーが BSCU の電源をオフにする [H4-1, H4-5]	CREW.4c2: 必要とされる時に自動ブレーキやアンチスキッド動作が完了する前に、クルーが BSCU の電源をオフにするのが早すぎる[H4-1, H4-5]	
		CREW.4b3: アンチスキッド機能が必要とされていて（または必要とされるであろう時）WBS が正常に動作している場合、クルーが BSCU の電源をオフにする [H4-1, H4-5]		
CREW.5 **BSCU 電源オン**	CREW.5a1: 通常のブレーキモード、自動ブレーキ、アンチスキッドが使われようとして WBS が通常に機能している時、クルーが BSCU の電源をオンにしない[H4-1, H4-5]		CREW.5c1: 通常のブレーキモード後、自動ブレーキ、アンチスキッド Anti-Skid が必要とされているのに、クルーが BSCU の電源をオンにするのが遅すぎる [H4-1, H4-5]	N/A

図 C.4: 航空機ブレーキシステムコントロールユニット（BSCU）自動ブレーキの UCA

コントロールアクション(BSCU による)	与えられないとハザード	与えられるとハザード	早すぎる・遅すぎる・誤順序	早すぎる停止・長すぎる適用
BSCU.1 自動ブレーキコマンド	BSCU.1a1: 自動ブレーキがブレーキコマンドを RTO 中 V1 まで出さない（滑走路長内に停止できない）[H4-1, H4-5]	BSCU.1b1: 自動ブレーキが着陸滑走中に過度なブレーキコマンドを出す [H4-1, H4-5]	BSCU.1c1 自動ブレーキがタッチダウン前にブレーキコマンドを出す（バースト、制御不能、傷害、他の損傷になる） [H4-1, H4-5]	BSCU.1d1 着陸滑走中タクシーTBD 速度達成前に自動ブレーキがブレーキを止める（減速の低下につながる） [H4-1, H4-5]
	BSCU.1a2 着陸滑走中 BSCU が作動した時自動ブレーキがブレーキコマンドを出さない（不十分な減速とオーバーシュートになる）[H4-1, H4-5]	BSCU.1b2 離陸中自動ブレーキがブレーキコマンドを不適切に出す（不十分な加速になる）[H4-1, H4-2, H4-5]	BSCU.1c2 タッチダウン後 TBD 秒以上自動ブレーキがブレーキコマンドを出す（不十分な減速と潜在的な制御不能、オーバーシュートになる） [H4-1, H4-5]	BSCU.1d2 着陸滑走中自動ブレーキがブレーキコマンドを長く（TBD 秒以上）出しすぎる（滑走路上での停止になる） [H4-1]
		BSCU1b3 自動ブレーキがブレーキコマンドを出すが不十分（着陸滑走中の減速が不十分になる）[H4-1, H4-5]	BSCU1c3 車輪が地上を離れて RTO が要求されていない時のいつでも自動ブレーキがブレーキコマンドを出す（ギアの取り外し前に車輪を停止させるブレーキをかけている）[H4-1, H4-2, H4-5]	BSCU.1d3 アプローチ中タッチダウン TBD 秒未満前まで自動ブレーキがブレーキコマンドを出す（制御不能、装置の損傷になる） [H4-1, H4-5]
	BCSU.1a4 離陸後自動ブレーキがブレーキコマンドを出さない（車輪のロックに必要で、着陸ギア取り外しや飛行中車輪回転で潜在的に装置の損傷になる）[H4-6]		BSCU.1c4 離陸中止時 V1 後 TBD 秒以上自動ブレーキがブレーキコマンドを出す（自動ブレーキは RTO 中クルーがスロットルを下げた後の責任があると推定する）[H4-2]	

付録 D：安全に関するコントロールストラクチャーに係る責任

下記のリスト（「Engineering a Safer World」、第 13 章より）は、新しい安全に関するコントロールストラクチャーを作る際に、既存の安全に関するコントロールストラクチャーのモデルを作成したり改善したりするための活動を開始する人々のためのチェックリストとして使用できます。また、インシデントとアクシデントの分析を行う際に不適切なコントロールやコントロールストラクチャーを特定する際にも使用できます。このリストは網羅的なものではなく、個別の産業や安全プログラムに合わせて補完が必要です。

このリストには一般的な責任だけが含まれており、割り当て方法は示していません。組織内の特定の人や場所への、責任の適切な割り当ては、各組織の管理構造に依存します。それぞれの一般的な責任は複数の個別の責任に分割され、1 つのグループが責任を実際に果たし、その上に位置する他のグループがその活動を監督、指導、指揮、監視することで、安全に関するコントロールストラクチャー全体に割り当てます。もちろん、各責任は、関連する権限と説明責任の必要性、責任を実施するために必要なコントロール、フィードバック、コミュニケーション・チャネルを前提としています。

総合管理

- 組織のあらゆるレベルで、リーダーシップ、監督、安全管理を提供します。
- 企業または組織の安全指針を作成します。安全に関する重要な決定の評価と安全管理の実施を行うための基準を確立します。指針を浸透させる経路を確立します。フィードバック・チャネルを確立して、従業員がそれを理解しているかどうか、従っているかどうか、そしてそれが効果的かどうかを判断します。指針は必要に応じて更新します。
- 企業または組織の安全基準を定め、それを実装、更新、実施します。開発およびオペレーションにおける安全工学の最低要件を設定し、請負業者の活動を含め、それらの要件の実施を監視します。危険な操作のための最小限の物理的および操作上の基準を設定します。
- インシデントとアクシデントの調査基準を確立し、勧告が実施され有効であることを確実にする。フィードバックを使用して標準を改善します。
- 安全に関するコントロールストラクチャーの変更を含む、安全への影響に関するすべての変更を評価するための変更要求の管理を確立します。計画外の変更やより危険度の高い状態への移行については、安全に関するコントロールストラクチャーの監査を行います。
- 組織的な安全に関するコントロールストラクチャーを作り、監視します。安全のための責任、権限、説明責任を割り当てます。
- 作業部会を設置します。
- 強固で信頼性の高いコミュニケーション・チャネルを確立し、開発システムの設計と、オペレーション・プロセスの状態を正確に管理するリスク認識を確実にします。このチャネルには、請負業者の活動を含めるべきです。
- 安全に関する活動のために物理的資源および人的資源を提供します。安全上重要な活動を行っている人が、適切なスキル、知識、および物理的なリソースを有していることを確実にします。
- 使いやすい問題報告システムを構築し、必要な変更と改善を加えるために監視します。
- 全従業員を対象とした安全教育と訓練を確立し、その安全教育、訓練および継続的改善のプロセスが有効かどうかを判断するためのフィードバックチャネルを確立します。教育には、過去の事故や原因の振返り、教訓からのインプット、トラブル報告などを含まなければなりません。有効性の評価には、監査中の知識評価から得た情報を含めることが可能です。

- 安全性に関連した技術的意思決定が、費用やスケジュールなどのプログラム上の考慮事項とは無関係であることを確実にするために、組織体制および管理体制を確立します。
- 安全関連の技術的決定とプログラム上の考慮事項との間の対立について、明確で透明かつ明示的な解決手順を確立します。対立の解決手順を使用し、確実に効果があげます。
- 安全に関する意思決定を行うマネージャが十分に情報を得て熟練していることを確実にします。すべての従業員（現場作業員を含む）および請負業者が安全関連の意思決定に貢献できるようにし、奨励する仕組みを確立します。
- 安全に関する意思決定のための評価と改善プロセスを確立します。
- 組織の安全情報システムを構築および更新します。
- 安全管理計画を作成し、更新します。
- 従業員および請負業者が適切に機能していないシステムまたは安全に関するコントロールストラクチャーの一部について、安全性に関する苦情や懸念を表明できるよう、コミュニケーション・チャネル、解決プロセス、裁定手続を確立します。懸念を報告する際の匿名性の要求を評価します。

開発

- 開発者や開発マネージャー向けに、安全に配慮した設計やその他の必要なスキルに関する特別なトレーニングを実施します。イベントが発生したときにこのトレーニングを更新し、さらに経験から学びます。トレーニングのフィードバック、評価、改善プロセスを整備します。
- ハザード・ログを作成し、維持します。ハザードとその状態の文書化と追跡を確立し、維持します。
- 作業部会を設置します。
- システムのハザードと安全性の制約を使用して、システムの安全を設計します。設計プロセスが進むにつれて、設計と安全制約を繰り返して洗練します。システム設計では、非安全なオペレータの行動を引き起こすか、それに寄与し、さらにシステムのハザードにつながる文脈上の要因を排除または低減する方法の検討を確実にします。注意散漫、疲労などは、そうした状況で通常の人間と同様に行動するのではなく、設計者が想像したように人間が振る舞う設計となっていることから生じるリスク要因となります。
- オペレーション上の想定事項、安全制約、安全性に関する設計上の特徴、事業上の想定事項、安全性に関するオペレーション上の制約、訓練と操作に関する指示、監査と性能評価の要求、オペレーション手順、安全性の検証と分析の結果を文書化します。安全制約と設計上の特徴の間におけるトレースを含め、「何を」と「なぜ」を、それらを実施するために文書化します。
- 安全に関する意思決定が必要な場合に利用可能な高品質かつ包括的なハザード解析を実施します。ハザード解析は早期の意思決定から始め、システムのライフサイクルを通して継続して行います。ハザード解析の結果は必要な人に適時に伝達します。コミュニケーションが下向き、上向き、横向きに（すなわち、サブシステムの間で）なされることを可能にするコミュニケーション構造を確立します。 設計が進展し、検証の経験を積むにつれて、ハザード解析を更新します。
- エンジニアおよびマネージャが、意思決定においてハザード解析の結果を使用するように指導します。
- 経験を積む中で、ハザード・ログとハザード解析を維持し、使用します。開発に関わるすべての人に安全関連の要求と制約を確実に伝えます。
- オペレーションの教訓（アクシデントとインシデントの報告を含む）を収集し、開発プロセスの改善に使用します。オペレーション経験を使用して、開発の安全管理の欠陥を特定し、改善を実施します。

オペレーション

- オペレーションの安全管理計画を作成します。
- オペレーション担当者やオペレーション管理者向けに特別なトレーニングを開発し、必要なスキルを教育し、イベントが発生したときにこのトレーニングを更新し、さらに経験から学びます。このトレーニングのフィードバック、評価、および改善プロセスを整備します。従業員が安全に仕事を遂行し、安全装備の適切な使用を理解し、緊急時に適切に対応できるように指導します。
- 作業部会を設置します。
- 経験を積む中で、ハザード・ログとハザード解析を維持し、使用します。
- 危険な作業中は、すべての緊急用装置および安全装置が常に作動可能であることを確実にします。安全上重要で、非定常的で、潜在的に危険を伴うオペレーションが開始する前に、警報装置のテストを含め、すべての安全装置を点検し、動作可能であることを確実にします。
- 危険な状態（水に反応する化学物質を含むタンク内の水など）やイベントを含む、あらゆる動作異常の詳細な調査を実施します。潜在的な危険を伴うオペレーションが開始または再開する前に、そうした動作異常がなぜ発生したのかを特定します。この種の調査を実施するのに必要なトレーニングと、管理者への適切なフィードバック・チャネルを提供します。
- 変更手順の管理を定め、変更手順が遵守していることを確実にします。これらの手順には、提案されたすべての変更についてのハザード解析と、安全上重要なオペレーションに関連するすべての変更の承認を含む必要があります。安全上重要な機器を使用できないようにする指針を作成し、実施します。
- ハザード解析の結果をオペレーションと保守の前提条件として使用し、安全監査、性能評価、および検査を実施します。安全に関する方針と手順が守られていること、そして安全に関する教育訓練が有効であることを確実にするためにデータを収集します。増加するリスクの先行指標のフィードバック・チャネルを確立します。
- 開発中に作成され、よりリスクの高い状態に移行する先行指標を特定するためにオペレーションに引き渡されたハザード解析と文書を使用します。先行指標を検出し適切に対応するためにフィードバック・チャネルを確立します。
- オペレーション経験についての情報を戻すために、オペレーションから開発へのコミュニケーション・チャネルを確立します。
- すべてのシステミックな要因を含め、インシデントやアクシデントの詳細な調査を実施します。すべての勧告を実施するための責任を割り当てます。勧告が完全に実施され有効であったかどうかを判断するために追跡調査を行います。
- 安全上重要な活動が適切に実施されたかどうかを独立して確認します。
- 安全上重要であると特定された項目の保守を優先させます。保守をスケジュールに沿って実施します。
- 安全上重要な装置を使えないようにし、物理システムを変更するための指針を作成し、実施します。
- 以前にシャットダウンしたユニットの操作、あるいは保守作業後の操作を開始するための特別な手順を作成し、実行します。
- 擬似警報の頻度を調査し、削減します。
- 誤作動している警報と計器にはっきりと印を付けます。一般に、現在誤作動しているすべての機器に関する情報をオペレータに伝達する手順を確立し、その手順を確実に順守します。誤作動した機器の報告に対するあらゆる障害を取り除きます。
- 安全上重要なすべての機器および警報手順について、安全オペレーション範囲を定義し、伝えます。オペレータがこの安全オペレーション範囲を認識していることを確実にします。緊急事態が発生していなくても、オペレータは動作範囲と緊急時の手順に従うことで称えられるよう

にします。必要に応じて、時間の経過とともにオペレーション範囲と警報手順をチューニングします。

- 安全上重要な物品のスペアの在庫を保有しており、迅速に入手できるようにします。
- 安全関連のすべてのイベントや活動について、工場管理者に連絡するチャネルを確立します。工場管理者がオペレーションに関する安全な決定を下すために必要な情報とリスク意識を有していることを確実にします。
- 怪我をした作業員を処置するために緊急装置と対応が備わっており、操作可能であることを確実にします。
- 地域社会へのコミュニケーション・チャネルを確立し、ハザードに関する情報および必要なコンティンジェンシー対応と緊急対応を提供します。

付録 E：ソフトウエア集約型システムの分析的分解における限界（アビオニクスの事例）

システム内のプロパティを保証する従来のアプローチでは、分析的分解を適用します。つまり、個々のコンポーネントを分析し、結果を統合することで、プロパティがシステム全体に存在することを保証します。このアプローチは、コンポーネント間の結合および相互作用が比較的単純であるシステム、すなわち主に電気機械システム[36]を前提とした場合に適用できます。ソフトウエア集約型システムでは、このアプローチの基礎となる独立性と還元性（例えば、コンポーネント間に間接的または意図しない影響がないなど）の前提が成立しません。航空業界はこの事実を発見し始めています。Bartley と Lingberg の論文がこの問題について述べ、飛行中の飛行機の事例を示しています。彼らは、さまざまなメーカーによって作られ、個別に更新される一連の航空機アビオニクス機能である Integrated Modular Avionics (IMA)システムの文脈でこの問題を議論しています。

IMA システムでの情報共有を可能とするため、電気的相互作用、機械的または物理的相互作用、およびソフトウエアモジュールの相互作用について、インタフェースをインタフェース管理文書 (ICD)で指定することが多い。典型的なソフトウエア・インタフェース仕様にはデジタル信号特性、並びにデータの伝送フォーマット、コーディング、タイミング、更新要求、並びにデータとデータ要素の定義、メッセージの構造と流れ、イベントの操作シーケンス、並びにエラー検出と回復の手順などを含みます。ICD は、インタフェース上に戻されるべき変数を実際に生成する基本的なロジックまたは振舞いを明示的に規定しています。

ICD を使用する基礎となる前提は、コンポーネント間のデータと制御の結合が最小限に抑えられ、ある機能への変更が ICD への変更を必要としない場合、それらのパラメータを使用している機能はその変更によって影響を受けないということです。この前提は、Bartley および Lingberg[37]が示しているフラップ制御に関する例に示されているように、誤りです。

図 E.1 は「フラップ拡張」の離散変数が、一般には独立していると考えられているアビオニクス機能間を通過する、フラップシステム・コントローラ間のインタフェースを示しています。拡張されたフラップの正確な技術的意味をさらに定義することなく、フラップ拡張された離散変数の真の状態は、次のいずれかを意味する可能性があります。

- 1 つ以上のフラップ止め[38]で左右両方の後縁（TE）フラップ面が検出される。
- 左右の後縁フラップ面が、フラップ止め”Up”で検出されない。
- フラップ・レバーハンドルが 1 つ以上のフラップハンドル止めで検出される。
- フラップ・レバーハンドルが、フラップハンドル止め”Up”で検出されない。

これら 4 つの可能性のそれぞれは、互いに異なる特性を有し、異なる受信機能は、フラップ拡張離散を異なる方法で使用することができます。例えば、後縁フラップが移動し始めると、「”Up”位置にないフラップ」を決定するロジックが、ほぼ即座に条件を満します。対照的に、フラップは、「1 つの止めで検出されたフラップ」の位置に完全に到達するまでに 5〜10 秒かかることがありま

[36]統合したコンポーネント分析からの結果は、コンポーネント間の微妙で非直接的な相互作用が統合コンポーネントの相互作用を歪める場合、これらのシステ ムであっても正しくないことがあります。

[37] Gregg Bartley and Barbara Lingberg, Certification Concerns of Integrated Modular Avionics (IMA) Systems, Federal Aviation Administration, Washington D.C.

[38]戻り止め detent とは、ホイール、車軸、主軸の回転に対して抵抗したり阻止したりする装置です。

す。さらに、油圧システムの故障のためにフラップ面が有効なコマンドに応答しない場合、レバーが止め"Up"から移動すると、フラップレバーの位置はもはやフラップ面の真の位置を反映しなくなります。明らかに、信号がどのように計算されるかは、その特定の信号を使用する機能の設計者が理解する必要があります。説明されているような違いは、システムを使用する上で非常に重要です。

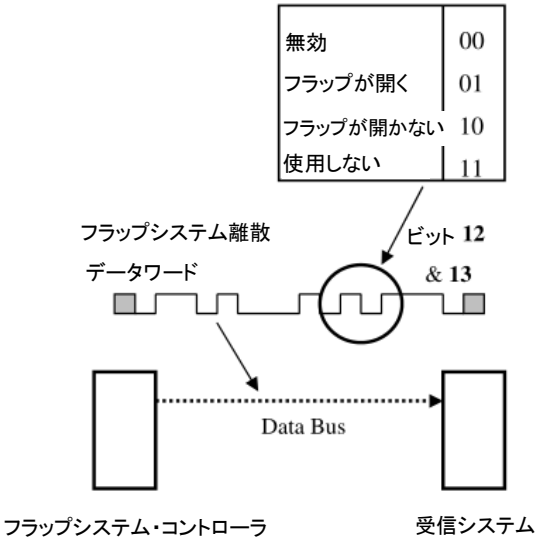


図 E.1: フラップ拡張(Ext)離散データ伝送 [3].

図 E.1 の情報のみがデータバスを介して通信される場合、フラップシステム・コントローラのロジックの変更は、フラップ拡張された変数を使用する機能の動作に悪影響を与える可能性があります。したがって、受信システムの開発者は変更に関与することはなく、システムレベルの影響を知ることありません。例えば、個々のアプリケーションの設計が「ファイナライズ」された後、開発サイクルの後半で発見された問題は、フラップシステム・コントローラのロジックの変更につながる可能性があります。

フラップアラート・ロジックでこの問題に対処しうる方法の 1 つは、実際のフラップ面の位置の代わりにフラップレバー位置を使用してフラップ拡張変数を計算することです。フラップ拡張変数の物理的な意味は変化してきています。

Bartley と Lingberg は以下のように述べている：

> 図のポイントは、更新される機能（この例ではフラップシステム）の設計者とシステム専門家は、その変更が川下のユーザーに与える影響を判断できないということである。変更の影響を評価する必要のある専門家は、ソースシステムではなく、使用システムの設計者とアナリストだ。ソース機能の ICD が変更されていなくても、この例では、機能境界をまたぐ変更影響分析が明らかに必要だ。さらにこの例は、1 つのパーティションに存在する機能を、強固なパーティショニングが異なるパーティション内の機能の変更からなぜ完全に遮断しないのかを示している。

FMEA や故障木分析(FTA)などの伝統的なハザード解析手法は、個々のコンポーネントの故障または故障モードに焦点を当てています。コンポーネントの信頼性に重点を置いているのは、強固なパーティショニングとインタフェース制御が有効であり、コンポーネントが直接的または間接的に相互作用しないことを前提としています。しかしながら、例が示すように、これらの前提は通常、複雑なシステムでは成り立ちません。

この問題は、STPA を使用して理論的に説明し解決することができます。

付録 F：非エンジニアのための工学とシステムエンジニアリングの基本的概念

このハンドブックの初期の草稿をレビューした際、私たちはすべての読者が持っているとは限らない工学のバックグラウンドを前提としていたことに気づきました。STPA のユーザーは必ずしも工学の訓練を受けているとは限りません。この付録では、このハンドブックの一部のユーザーに馴染みのない工学の基本的概念を紹介します。

私は、読者が興味のあるセクションを選ぶことができるよう、このセクションを付録の中で独立させようとしました。以下のようなトピックについて述べる上で、特定の教育バックグラウンドは想定していません：

- 「システム」とは何か？
- システムエンジニアリングの基本的概念と用語
- 工学における「コントロール」の概念
- システム理論 vs.複雑性理論

「システム」とは何か？

このハンドブックのすべての根底にある最も基本的な概念は、*システム*の概念です。

> *システム：共通の目標や目的を達成するために全体として作用する一連のもの（システムコンポーネントと呼ばれる）。*

システムの定義の中には、「目標」または「目的」を省略し、本質的にシステムと統合された全体を形成する、接続された一連の物または部品であると述べているものもあります。この定義は、「システム」という用語の通常の使用方法を踏まえていません。靴、星、機関車はシステムの一連の物あるいは一連の潜在的部品ですが、ほとんどの人はこれをシステムとは通常考えていません。これらの個々の物を一緒に考えるために「目的」も考慮するならば、それはシステムとして考えることができます。「目的」はシステムという概念の基本であります。

他の定義では、システムのコンポーネントが相互依存関係にあるか、接続されているか、相互作用していなければならないと述べられているが、そうした条件はシステムの成立に特に不要です。これらの定義はシステムとして通常考えられている物を除外する形で定義を制限しています。システム構成要素は、直接的または間接的に相互に接続しており、後者の場合は、システム目的のみの接続、あるいは様々なタイプの非線形相互依存性の接続を含みます。

この定義の結論は、「目標」や「目的」はシステムにとって基本的なものであるということです。しかし、システムを定義し捉えるには異なる「目的」があるかもしれません。このセクションではさまざまな点を説明するために空港を取り上げます。旅行者にとっては、空港の「目的」は、他の場所への航空輸送を提供することでしょう。地方自治体または州政府にとっては、空港は政府の歳入および空港地域における経済活動を増やすための手段となるでしょう。航空会社にとっては、空港の「目的」は乗客と貨物の乗降となるかもしれません。空港事業にとっては、空港の「目的」は製品やサービス販売相手として顧客を引き付けることに違いありません。したがって、システムについて話すときは、考慮するシステムの「目的」を必ず特定する必要があります。

システムとは抽象化されたもの、すなわちそのシステムを見る人が捉えたモデルである。

システムを見る人は、設計者とは異なるシステム目的を捉えたり、さまざまな関連特性に焦点を当てたりすることがあります。システムの「目的」を含む仕様は、システムエンジニアリングで非常に重要です。仕様は、システムを設計、使用し、あるいは見る人々の間でそのメンタルモデルの一貫性を保証し、コミュニケーションを強化します。1 つの「空港」システムには空港のさまざまな

構成要素が含まれていることを考えてみてください。例えば航空会社の視点から空港を見ると、乗客のチェックインカウンターや飛行機のタラップ、誘導路があります。一方、商業的に見ると店舗や顧客があります。これらは、人間の心によって実際の物理世界に置かれたモデルまたは抽象化されたものであることに再度注意してください。あらゆる「空港システム」あるいはサブシステムにおいて考慮されるコンポーネントと、システム全体の中でコンポーネントが果たす役割は、空港システムまたは空港サブシステムの各コンセプト（モデル）ごとに異なる場合があります。このトピックについては、システムエンジニアリングにおける仕様の重要性に関する次のセクションで詳しく述べます。ここでの基本的な考え方は、あるシステムのモデリングと分析を行う人々は、考慮するシステムの「目的」または「目標」を特定し合意しなければならないこと、およびシステムのこれらの側面は、システムを見る人々が現実世界の事物に投影する抽象物またはモデルであるということです。

設計された（人工の）システムの場合、システムを見る人々はシステムの「目的」を設計者が意図したものと異なる形で解釈することこともあるが、システムの「目的」は通常、システムが作られる前に特定されています。自然のシステムでは、動作はシステムを見る人によって意図的に解釈されることがあります。

システムの概念の根底にはいくつかの基本的な前提があります：

- システムの「目標」は定義することができる。
- システムは原子的である。つまり、相互作用が直接的か間接的かにかかわらず、相互的に動作するか、あるいはコンポーネント間で関係性を持つコンポーネントに分けることができる。

システムそのものは、より大きなシステムの一部であっても、サブシステム（全体システムのコンポーネントと見なされる）に分割されてもよいです。ここでの定義は、数学とコンピュータサイエンスにおいて「再帰的」と呼ばれるものであることに注意してください。つまり、定義はそれ自体でなされます。システムは通常、より大きなシステムの一部であり、サブシステムに分割することができます。図 F.1 は、A と名付けられたシステム（例えば、「空港」システム）および 3 つのサブシステム（各サブシステム自体も、システムとして考えられる）について、典型的な抽象階層を示しています。

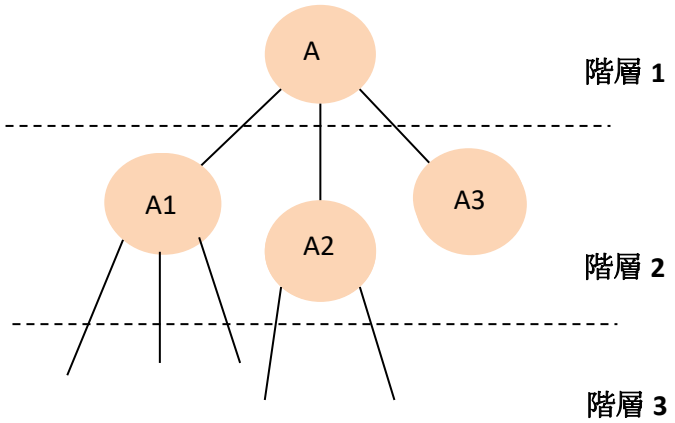


図 F.1：抽象システム A は、3 つのサブシステムで構成されているとみなされる。各サブシステム自体がシステムである。

図 F.1 は接続図ではありません。階層的に抽象化したものであり、異なるレベルで抽象化したものあるいはモデリングにおける、システムの異なるビューを示しています。*図 F.2* は、A 自体がより大きなシステム AB の一部である場合に抽象化したものを示しています。

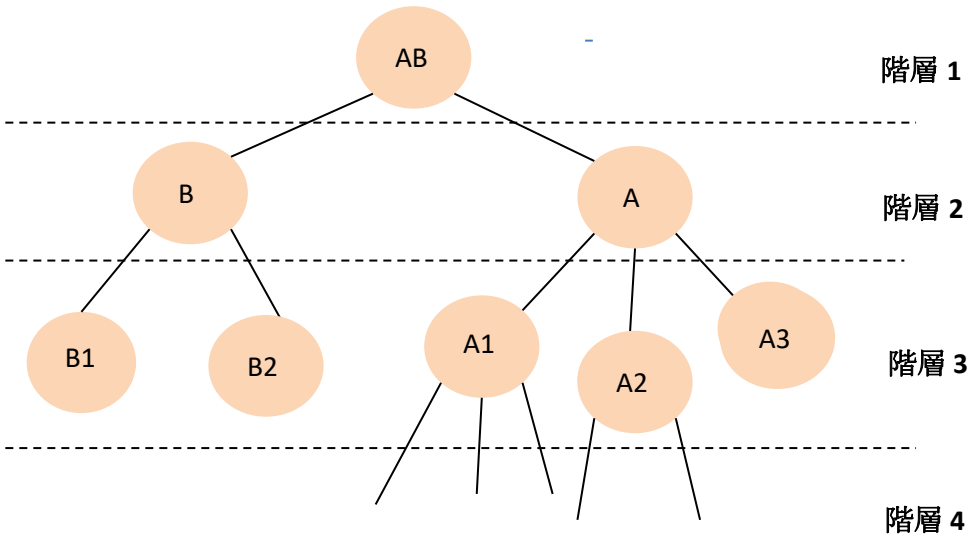
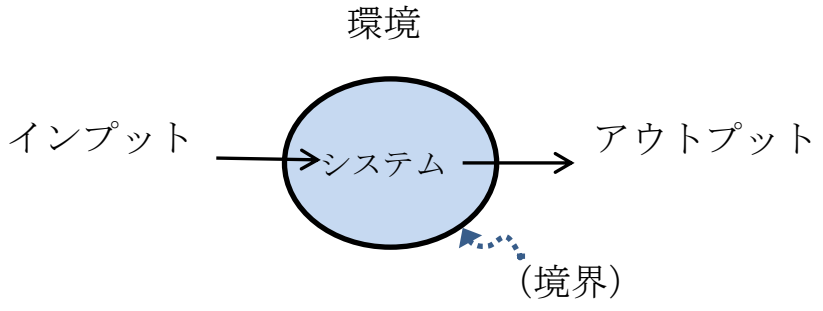


図 F.2: システム A はより大きなシステム AB のコンポーネント（サブシステム）とみなされる

システムの定義の再帰的な性質は重要です。なぜなら、多くの人々が「システムオブシステムズ」はシステムとは異なる扱いをしなければならないと述べているためです。実は、一般的なシステムエンジニアリングと分析の手法および技術は、すべてのシステムに適用可能です。「システムオブシステムズ」は、システム理論で定義されているシステムの定義を使用すると「システム」にほかならず、工学においては概念として使われます。

システムには状態があります。状態とは、任意の時点におけるシステムを記述する関連特性の集合です。航空輸送システムと見なされる空港の場合、状態の特性とは、航空機が停まっている特定のゲートにおける乗客の数であり、彼らが何をしているか（乗客の搭乗、タキシング、離陸または着陸）であると言えます。関連する状態のコンポ―ネントは、システムとその環境との境界がどのように引かれるかに依存します。

環境は通常、システムの一部ではなく、その動作がシステムの状態に影響を与える可能性のある一連のコンポーネント（およびその特性）として定義します。したがって、システムは特定の時点で状態を有し、環境も状態を有します。 環境という概念は、システムとその環境との間に境界が存在することを意味します。ここでも、この概念はシステムを見る人によって生み出された抽象的なものであり、物理的境界である必要はありません。何がシステムの一部あるいは環境の一部であるかは、あるシステムとそれがその時点でどのように使用されるかに依存します。



システムの入力と出力はシステムの境界を横断します。このモデルは通常、*オープン・システム*と呼ばれます。*クローズド・システム*（入力と出力がない）という概念もあるが、この概念は、私たちがシステム安全で最も関心のある設計されたシステムとはあまり関連性がありません。

システムエンジニアリングとは何か？

システムエンジニアリングとは、設計作業の結果を向上させるために設計に構造を与え、システムを構築する試みです。比較的シンプルなシステムであれば、システムエンジニアリングは必要ないかもしれません。システムエンジニアリングは第 2 次世界大戦後に生まれました。より複雑なシステム、特にミサイルやその他の防衛システムの構築が始まった頃です。またその頃人々は、非公式な設計や構築プロセスでは、システムに想定された「目標」を満たすようなシステムは構築できず、潜在的な行動に対する制約、すなわち目標達成の制約になることに気づきました。構造化されたプロセスの欠如は、プロジェクトの遅れと予算の超過にもつながりました。

システムエンジニアリングは大きなテーマですが、このハンドブックを理解するのに必要なのは 2 つの概念のみです。1 つ目は、システムエンジニアリングのフェーズとプロセスの概念です。これらのフェーズの最も基本的なモデルを*図 F.3* に示します。

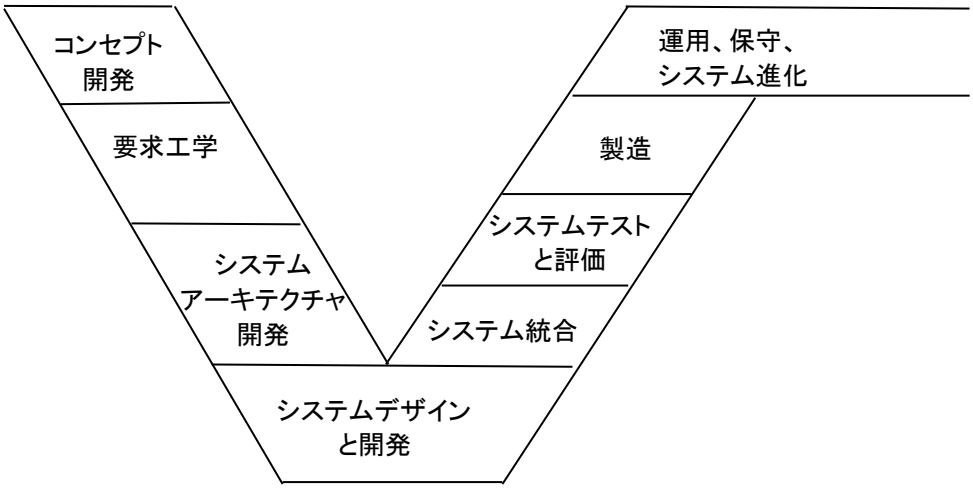


図 F.3: 基本的なシステムエンジニアリングの「V 字モデル」

このモデルは V 字モデルと呼ばれ、さまざまなコンポーネントに異なる名前を付けることができます。場合によってはフェーズが直線で描かれることもありますが（直線が下に傾いている場合は「ウォーターフォール」モデルと呼ばれる）、記述方法を除いて違いはありません。基本的な考え方は、構造化されたプロセスが使用される場合に、最良の結果が得られるというものです。そこではシステムコンセプトが最初に開発され、システムの基本的な目標と制約が定義されています。開発の初期段階では、手戻りを防ぐためにさまざまな実現可能性やその他の分析が行われます。要求工学が、通常、プロセスの次のステップであり、事前に合意された基本的な目標と制約を考慮して、システムの詳細な要求が開発されます。システムアーキテクチャ開発とは、詳細な設計と開発を行う前に、システムの基本アーキテクチャまたは上位レベルを設計することを意味します。「V」の左側は、基本的な設計プロセスを表します。

V の右側は、保証と製造が行われるとき開発の後半段階を示しています。システムの運用も含まれる場合があります（ライフサイクル・モデルと呼ばれる）。なぜならば安全分析はシステムのライフサイクル全体を通して続ける必要があるためであり、このハンドブックの図 F.3 および第 3 章で

も含めています。V 字モデルプロセスの右側では、設計および構築されたさまざまなコンポーネントが統合され、テストおよび評価フェーズに進みます。その後製造され、運用されます。

上記の図は、乱雑さを減らすために意図的に簡略化しており、全体のプロセスコンポーネントのみを反映しています。フェーズ間では常に多くの行き来があり、1 つの直線プロセスではありません。たとえば、設計プロセス自体が、解決する問題をより深く理解し、元の要求と制約を追加または変更する結果となるため、要求は当初、完全には特定されません。開発中に深刻な問題が発生すると、プロジェクトが完全に前の段階に戻る必要が生じることがあります。しかし一般的に、この段階的なプロセスに則って進められます。

さらに、このモデルは文字通り理解されるべきではありません。フェーズの名前付けはフェーズが並行して進行するのが不可能であることを意味するものではなく、実際には一般的に並行して進行します。例えば、すべてのテストが行われるのを最後まで待っているのは、プロジェクトにとって非常に大きなリスクです。様々な種類のテストは、最も初期の概念フェーズから（おそらく、シミュレーションやプロトタイプの形で）始まり、通常は各フェーズで継続して行われます。また、設計中のシステムのタイプに応じて、様々な名前が各フェーズに付けられます。目標とは単に、特定のプロジェクトのさまざまなニーズに合わせて調整される構造化プロセスを定義することです。第 3 章では、STPA が複雑なシステムの設計に関わるさまざまなフェーズや活動で一般にどのように使用されるのかを示します。

V 字モデルまたはウォーターフォール・モデルのもう 1 つの側面は、ドキュメント駆動であることです。私の経験では、複雑なシステムをうまく設計するための鍵は仕様にあります。通常、そのようなシステムには数多くの人（時には数千人）が関わっており、非常に長い時間をかけて開発や運用が行われます。結果がユーザーのニーズを満たし、意思決定がコミュニケーション上の問題が原因で混乱に陥っていないことを保証するためには、仕様が必要です。

仕様は開発の最も初期の段階から始めます。この付録の前のセクションでは、システムを、抽象化することとして定義し、その目的はシステムを見る人によって設定されるものであるとしました。構築するシステムの基本的な目的に誰もが確実に同意するためには、仕様は少なくともシステムの目的（目標）、および対象のあるいは必要とされている関連システムの特性、目標の達成における制約を特定していなければなりません。加えて、システム境界を特定しなければなりません（すなわち、何が設計者の境界内およびコントロール下にあるか、および何が設計者によって想定されておりコントロールされていない環境の一部であるか）。初期仕様のその他の側面としては、入力と出力、物理または論理コンポーネント、構造、コンポーネント間の関連する相互作用、およびそれらの動作がシステム全体の状態に与える影響が含まれます。開発が進むにつれて、最初の仕様は洗練され、詳細化されます。

トレーサビリティは、システムエンジニアリングのもう一つの重要な概念です。非常に大規模で複雑なシステムは完成するまでに、何千ページもの詳細な仕様が記述される場合もあります。システムの構築と運用においては、システムの各部分と、ある決定が依拠した開発プロセスの前提やその前の結果の関係を示すことが求められます。たとえば STPA では、非安全なコントロールアクションはシステムレベルのハザードまでトレースされ、シナリオは引き起こされる非安全なコントロールアクションまでトレースされます。変更が必要な場合は、最初からやり直すことなく何を変更する必要があるかを判断することができます。トレーサビリティは、意思決定の根底にあるいくつかの種類の根拠を記録する方法でもあります。

工学における「コントロール」の概念

コントロールの概念は、システムの構想、設計、および運用において重要です。システムはその動作を何らコントロールすることなく動作すると考えられますが、望ましくない悪影響が生じないように目標達成の方法も制限しながら、同時にシステムがその目標を満たしていることを保証するのは困難です。

コントロールは工学、特に複雑なシステムの基本です。エンジニアは生き物ではなく、何らかの目標を達成するために作られた人工物を扱っています。例えば自動車、ダム、航空機、化学プラント、宇宙船、洗濯機、ロボットなどです。それらが自力で動かない（橋のような）場合、それらの動作、すなわちそれらの状態を操作またはコントロールする必要があります。

非エンジニアは、「コントロール」という言葉を工学で使用されるのとは違う方法で解釈することがあります。コントロールとは、個々人が特定の役割と手続きを遵守するようある種の軍事的スタイルの権限によって強制するものではありません。むしろコントロールはここでは非常に広い意味で使用されており、インターロックやフェイルセーフ設計などの設計管理、保守や製造手順などのプロセスコントロール、あるいは文化、インセンティブ構造、個人の自己利益といった社会構成物を通じてなされます。何らかの形のコントロールを課さなければ、システムの目標と制約を確実に達成することはできません。

コントロールとして機能するシンプルなデザイン機能は、複雑さとコストを増加させることなく安全性を向上させることができるが、通常は設計プロセスの早い段階で安全性を検討する必要があります。デザインを通して安全性を高める最も効果的な方法は、危険な状態を排除する設計を用いることです。次善策は、危険が稀で、発生した場合も取り扱いが容易であるように設計することです。結果として生じる被害を最小限に抑えることによって危険の発生に反応する設計は、ハザード制御の最も望ましくない形態であることは明らかですが、そのようなコントロールを含める必要はしばしばあります。

従って、コントロールループはコントロールの 1 つの形態に過ぎません。「Engineering a Safer World」はそのことを強調しています。なぜならば、私の前著「Safeware」で、安全のための基本的な設計のその他の側面を広範にカバーしていたからです。さらに、システム理論はコントロールループまたは能動的コントロールの概念を強調しています。本付録では安全設計技術を教えることはできないが、受動的コントロールと能動的コントロールの区別のようないくつかの基本的な概念は、一般的な安全設計技術の理解に役立つでしょう。

安全を「コントロールする」には、3 種類の設計手法があります。1 つ目は、本質的に安全な設計を採用することです。本質的に安全な設計は、懸念されるハザードを生じさせない設計です。例えば、爆発を引き起こすのに十分なエネルギーを有さないようなシステム設計とする方法が考えられます。潜在的に有害な物質の使用に懸念がある場合、毒性のない代替物質を使う方法が考えられます。通信ネットワークを介して情報が失われる恐れがある場合、可能であればより直接的な通信を使用する方法が考えられます。

本質的に安全な設計が不可能な場合、受動的なコントロール機構を使う方法があります。受動的コントロールは能動的アクションを必要とせず、その代わりに重力のような基本的な物理的原理を用います。明白なコントロールアクション無しにシステムが危険な状態に入ることを防止する物理的なインターロックは、受動的なコントロール機構です。以下のような例があります：

- 誰かが近づいた場合にロボットや危険な機械の電源を遮断する感圧マットやライトカーテン
- バルブのターンオフとターンオンの正しいシーケンシングを保証する他、2 つのバルブを同時にオンまたはオフにしないようにする物理デバイス
- ブロック内の水が凍った場合、シリンダに亀裂を生じさせず、拡張してプラグを押し出す自動車のエンジン冷却システムの凍結プラグ
- 過度の熱が発生し、水が所定のレベル以下になると露出するボイラーの可溶性プラグ。その場合、プラグが溶け、開口部が蒸気を逃がし、ボイラー内の圧力を低下させ、爆発を防止する。
- 一定限度以上の速度超過を防ぐ調速機や、圧力を危険レベル以下に保つリリーフ弁

受動的コントロールは、システムが本質的に安全な状態になるように設計されることが多いです。事実、航空機の安全に対する基本的なアプローチは、システムをフェイルセーフになるよう設計することです。フェイルセーフ設計は以下のような例があります：

- セマフォ（信号機）をコントロールするケーブルが破損した場合、アームが自動的に停止位置に落ちるようにウェイトを使用した旧型鉄道セマフォ
- 空気圧によってオフ位置に保持されるエアブレーキ。 ブレーキラインが破損すると、空気圧が失われ、自動的にブレーキがかかる。
- デッドマンスイッチ
- バラストがマグネットによって適所に保持されているバチスカーフ（深海探査艇）。電力が失われた場合、バラストは解放され、バチスカーフは表面に上昇する。
- 障害の影響（航空機エンジンが故障したときにファンブレードが排出されるなど）が隣接するコンポーネントに影響を与えないように設計されている設計

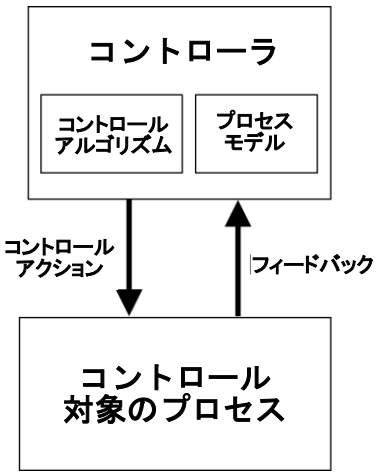
対照的に、*能動的コントロール*は、危険な状態を検出して修正することを必要とします。今日の能動的コントロールにおいては、このハンドブック全体で記述され使用されているコンピュータと基本的なコントロールループが活用されます。たとえば、障害や危険な状態が検出されないか、検出されても修正されないか、または損失を防ぐために時間内に修正されないなど、能動的コントロールには誤りがある可能性があることに留意ください。受動的コントロールは、通常、物理的な原理やよりシンプルな設計に依拠するため、能動的コントロールよりも信頼性が高い。能動的コントロールは検出および回復機構の信頼性が低く、受動的コントロールよりも通常ははるかに複雑な設計を用いています。しかし、能動的コントロールは設計自由度の点でより制約を受ける傾向があり、複雑なシステムでは常に実行可能とは限りません。

能動的コントロールにおいては通常、コントロールループが存在します。

コントロールループとは何か？

コントロールシステムは、コントロールループを使用して他の機器またはプロセスの動作を指令、指示、または規制します。このようなコントロールシステムは、ボイラーを制御するサーモスタットを使用する単一の家庭用加熱コントローラから、複雑なプロセスおよび機械をコントロールするために使用される大型の産業用制御システムにまで及びます。

非常に基本的なコントロールループを以下（およびこのハンドブックの他の箇所）に示します：



プロセスをコントロールするためには、コントローラは、コントロールされたプロセスの挙動に対する制約を維持することを含む、1 つまたは複数の目標を持たなければなりません。さらに、コ

ントローラーは、コントロールされたプロセスの挙動、すなわちコントロールされたプロセスまたはシステムの状態に影響を及ぼす何らかの方法を持たなければなりません。これらは図中で、コントロールアクションとして示しています。工学では、コントロールアクションはアクチュエーターによって実施されます。どんなコントロールアクションが必要であるかを知るために、コントローラーはシステムの現在の状態を判定できなければなりません。工学用語では、システムの状態に関する情報はセンサによって提供され、フィードバックと呼ばれます。最後に、ハンドブックの他の部分で説明しているように、コントローラーにはコントロールされたプロセスの状態のモデルを含まなければなりません。

一例として、単純なサーモスタットは、温度を特定の範囲内に維持するという目標を有しています。フィードバックは、コントロールプロセスの現在の状態、この場合は室内の空気の温度に関する情報をコントローラーに提供します。フィードバックは、コントローラーのプロセスモデルを更新するために使用します。他の情報は、環境情報（外気温度）や自然温度の変動に関する基本情報などのプロセスモデルの一部であってもよいです。コントローラは、プロセスモデルを使用して、どのようなコントロールアクションを提供するかを決定します。例えば室温（コントロールプロセス）の状態を調整し必要な範囲内に維持するために、熱風または冷気の送風を決定します。

コントロールループには一般的に、フィードバック制御とフィードフォワード制御の 2 つの動作モードがあります。 フィードバック制御は、上記の段落で説明したものです。 運転するとき、ドライバーは速度計（フィードバック）を読み取り、自動車の速度を自分が望む水準に保つためにブレーキあるいはアクセルを踏むことを決定します。

フィードフォワード制御の一例は、運転者が丘に近づくときに発生します。 ドライバーは、車が減速を開始するまで待ってアクセルを踏むことができます。しかし熟練したドライバーは、坂を上る際に一定の速度を維持するためには加速が必要であると予測し、車が実際に減速する前にアクセルを踏みます。したがって、フィードフォワード制御では、コントローラはプロセスの現在の状態（この場合は車の速度）のモデルと将来（傾斜上で動作する）のモデルを使用し、必要性を特定する特定のフィードバックなしでコントロールアクションを提供します。

多くの場合、フィードバックとフィードフォワード制御は並行して使用します。この例では、ドライバーはフィードバックを受け取る前にアクセルを踏む必要があると、フィードフォワード制御を使用して予測します。しかしフィードフォワード計算が完璧ではない場合、フィードフォワード・コントロールアクションを調整して傾斜上で一定速度を保つためにフィードバックも使用されます。

このハンドブックの先行指標に関する章では、シェイピング・アクションは、仮定を維持し、ハザードを防止し（コントロールプロセスにおける非安全な状態）、より高いリスクの状態へのプロセスの移行をコントロールするように制御設計することによって安全な状態を維持しようとする試みとして説明します。シェイピング・アクションは、フィードフォワード制御の一種を提供します。使用されるコントロールのタイプは、受動的であっても能動的であってもよいです。ヘッジング・アクションやコンティンジェンシー・アクションは、リスクを増加させる兆候とそれに対する対応のためにシステムを監視することを含む、一種のフィードバック制御を提供します。シェイピング・アクションの有効性を監視することを含むヘッジング・アクションは、フィードフォワード制御とフィードバック制御を組み合わせて提供します。

システム理論に基づくコントロールの工学的概念は、社会システムや組織システムにも適用することができ、経営理論や社会科学で広く使われています。

システム理論 vs.複雑性理論

STAMP は複雑性理論ではなくシステム理論に基づいています。システム理論に貢献してきた人々は多いが、オーストリアの生物学者であるルートヴィヒ・フォン・ ベルタランフィ(Ludwig von Bertalanffy)は、彼が一般システム理論と呼んだものの創始者の 1 人として認められています。ノー

バート・ウィーナー（Norbert Weiner）は、数学と工学への応用を探りました。ウィーナーは自分の理論をサイバネティクスと呼んだが、この用語は時間が経つにつれ消えてゆき、今日ではベルタランフィの用語が一般的に使われています。

このハンドブックなどで述べているシステム理論は、自然システムあるいは人口システムを問わず、複雑なシステムの動作を理解するために活用できる一連の原則です。*システム思考*は、システム理論の原則を適用するときに人々が何をしているかを記述するためによく使われる用語です。これらの原則は、このハンドブックの第 1 章で簡単に説明しています。

複雑性理論は 1960 年代にシステム理論やその他の概念から生まれ、サンタフェ研究所や同研究所の研究者との関連で通常語られています。システム理論と複雑性理論には共通点があります。それは出現（emergence）のような用語を含んでおり、コンポーネントの削減や分解ではなく複雑なシステムの動作全体に焦点を当てている点です。システム理論の基本的な要素は、出現、階層、通信、コントロールであり、これらは複雑性理論にも含まれています。またシステム理論と複雑性理論はともに、フィードバック制御とフィードフォワード制御、適応性、非線形相互作用、制約の概念も含んでいます。どちらもシステムの動作を理解するための原則として、還元主義または分解を拒んでいます。

しかし、大きな違いがあります。複雑性理論は自然システムを記述するために生み出されたものです。ここでは一見独立したエージェントが、我々がまだ完全に理解していない自然法則を使用して自分自身を自発的に順序付けし、また一貫したシステムに順序変更します。対照的に、システム理論は人工的に構築・設計されたシステムにより適しています。ここでは人間が何らかの工学的プロセスまたは設計プロセスを使用して意図的にシステムを構築します。基本的な設計は既知であり、変更はコントロールされています。

もう一つの大きな違いは、システム理論はすべてのシステムを創発的な動作を表すものとして捉えている一方、複雑性理論はシステムを単純（Simple）、込み入った（Complicated）、複雑(Complex)、カオス (Chaotic)の 4 つのタイプに、出現の度合いやタイプによって分けていることです。

このように、システム理論は設計されたシステムに最も適しているようです。一方、複雑性理論は、天気などの設計が不明な自然システムや、設計がなされておらず秩序が欠如しており創発的な動作を予測することが困難あるいは不可能な、コミュニティなどの社会システムに最も適しています。複雑性理論は、設計はできないが、しかし発生したときに経験と学習を行わなければならないような行動を扱います。

少なくとも私にとっては、システム理論の方が複雑性理論よりも理解し使用するのが容易です。複雑性理論の枠組みは、適用および理解が困難で混乱する可能性があり、安全性およびその他の創発特性を向上させるという工学目標には過剰である可能性があります。したがって、STAMP の基礎としてはシステム理論が選択されました。

付録 G：因果関係シナリオ生成を支援するコントロールモデル

STPA の目標は、ハザーダスな状態を引き起こしうる因果関係シナリオを明確にすることです。ハザーダスな状態が生成される一般的理由を示す「Engineering a Safer World」に挙げたモデルを示します（下図 G.1 参照）。しかし、一つ問題となるのは、このモデルは非常に抽象的であり、シナリオ生成に有用な詳細が省かれているということです。また、自動化されたものと人間によるコントローラを一つの「コントローラ」とラベル化した一般的なボックスにまとめており、抽象化したものでは重要な違いが欠落しています。のちに、第 9 章では、設計技術を説明するために、他のモデルを作りました。このモデルでも、極端に一般化されています。この本が出版されてから、この他にもっと詳細なヒューマンコントローラモデルも作られています。その付録には、このような因果関係シナリオを識別するために、もっと完璧で、もっと役に立つと期待されるモデルについて説明します。

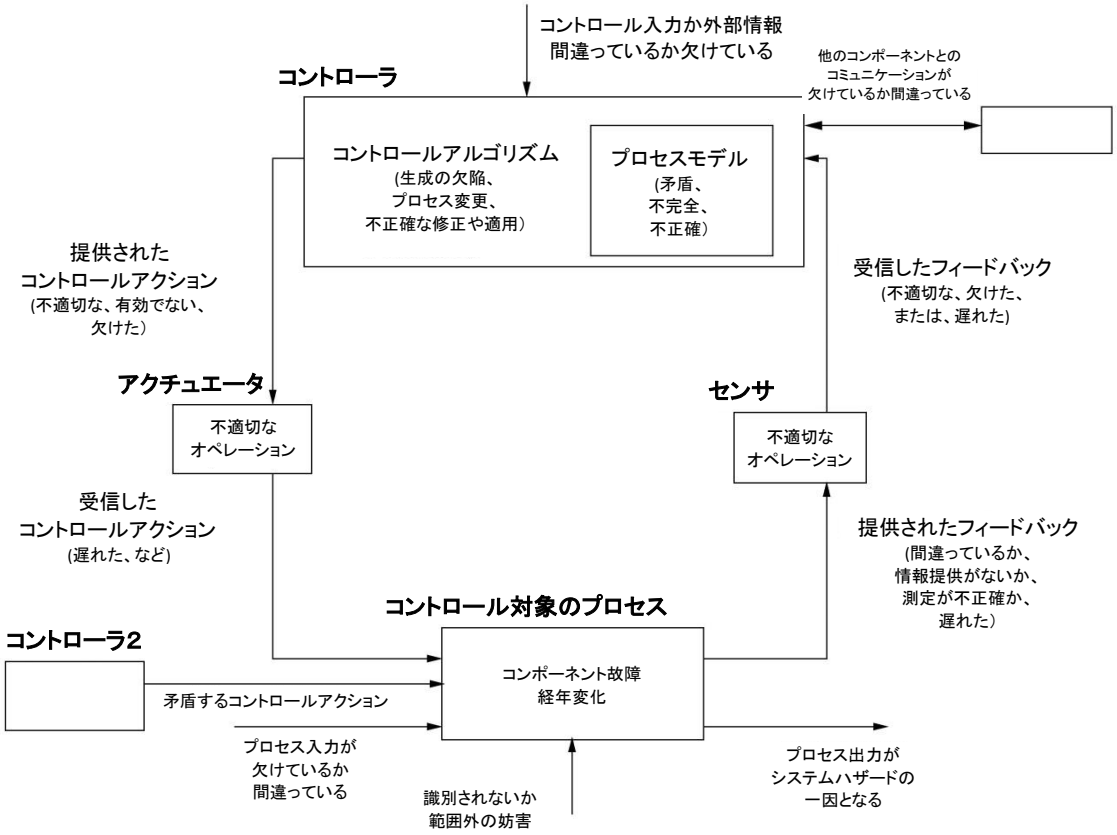


図 G.1: 因果関係シナリオ生成を支援する以前の一般的モデル

因果関係シナリオ生成の抽象モデル

新しいモデルを図 G.2 に示します。そしてモデルの各部分についてシナリオの生成に使われる方法に従って説明します。潜在的な因果関係要因(potential causal factors)があまりにも多く、一つの一般的モデルに全てを入れることはできないため、個別に説明します。因果関係シナリオの生成で、このモデルをガイドとして使用するなら、アプリケーション（システム）に特有なコントロールモデルから始めます。以下の一般的なコンポーネントは、あなたのモデルのそれぞれの部分に対応します。一般的なガイダンスとして、相応するシステムに特有の因果関係シナリオを作るのに役立つでしょう。

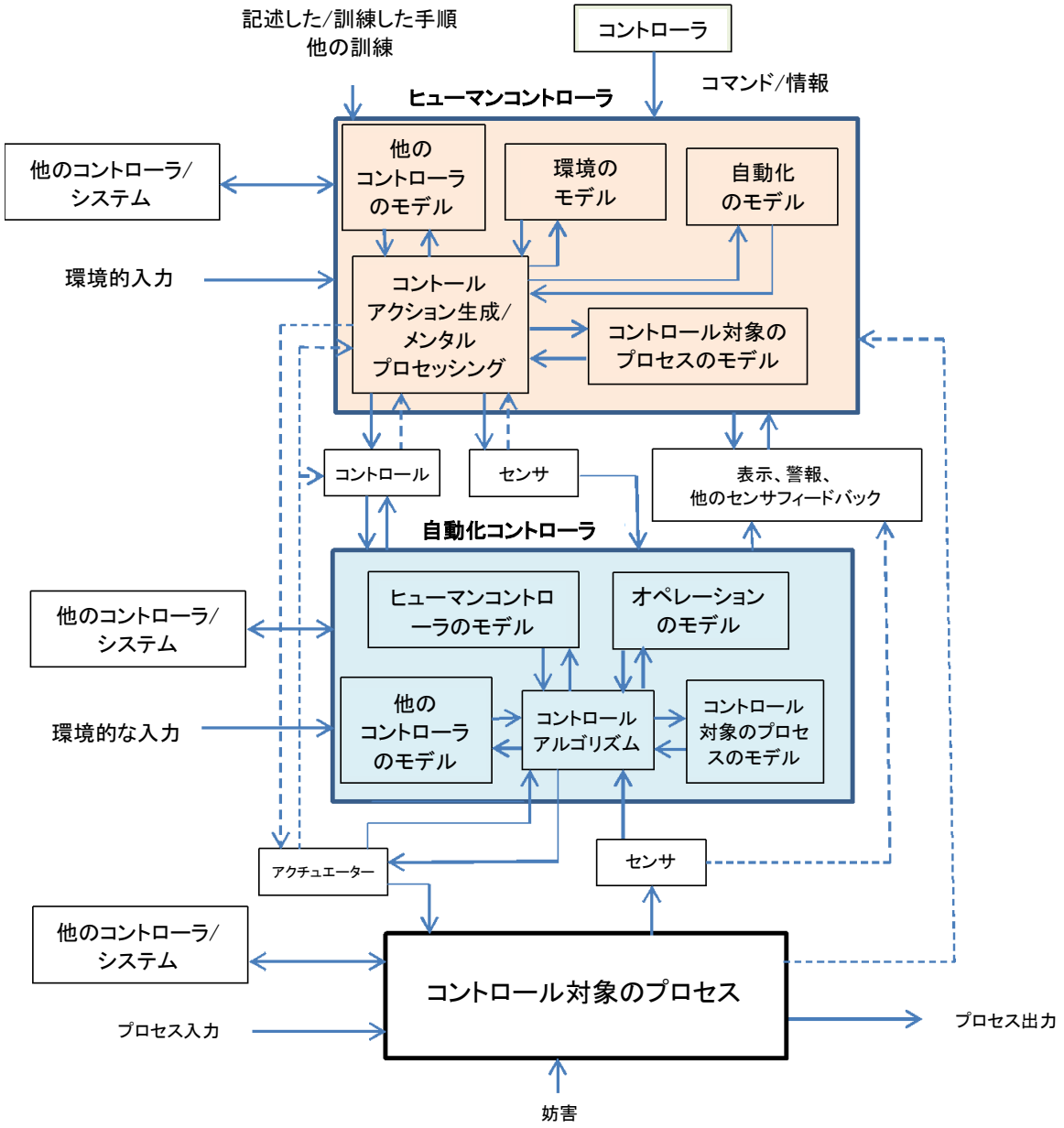


図 G.2：新しいSTPA 因果関係分析シナリオ生成のための一般的な因果関係コントロールモデル

コントロール対象のプロセス

例えば、航空機の姿勢、自動車の速度、ロボットの位置のように、図の下部にある「コントロール対象のプロセス」の状態の観点から、ハザードを定義します。これらの状態はコンポーネントあるいは変数で成り立っています。STPA の目標は、コントロール対象のプロセスが、どうやってハザード状態に陥るのかを明確にすることであり、そのため、コントロール対象のプロセスの状態が変わっていく様子に注目する必要があります。この状態を変化させるものは何でも潜在的にハザードの因果関係シナリオの一部分です。

コントロール対象のプロセスコンポーネントの経年による故障あるいは劣化

コントロール対象のプロセスハードウエアの故障は、ハザーダス状態になる可能性があります。腐食といったハードウエアの時間経過による変化により、設計されたものとは違った動作をすることがあります。ハードウエアのインターロックのようなコントロールも、故障(fail)あるいはデグレード(degrade)する可能性があります。それによりハザーダス状態を引き起こすことになります。覚えておきたいのは、目標は FMEA を行うことではなく、起こりうるすべての故障とその結果を意識して見ることにあります。目標は、ハザードから始めて、コントロール対象のプロセスのどの条件がハザーダス状態に至る一因となるのかを判断することです。例として、因果関係分析に関わるハザードとして、航空機のピッチ制御の喪失を想定してみましょう。スラット、フラップ、補助翼、昇降舵といった航空機のピッチ制御装置の故障は、ハザードに至る可能性があります。ハザードが効果的なブレーキ（減速）の不足である場合には、これは油圧が不十分（ポンプの故障、油圧リークなど）であることによるのかもしれません。

外乱

落雷のような航空機の電気システムに影響をあたえるプロセスへの外乱は、ハザーダス状態に至る可能性があります。エンジンに吸い込まれた鳥などは、航空機のパワー（推力）に影響を与えます。現在その空域にある航空機それぞれに対して、ハザーダス状態を増加させるような天候によって、領空は混乱するかもしれません。滑走路が雨で濡れていてブレーキが効かない（例えばホイールのハイドロプレーン現象）場合のように、ある環境が、コントロール対象のプロセスを適切に実行することを妨げることがあります。

コントロール対象のプロセスへの直接入力

プロセスへの直接入力は、その状態に影響を与えるかもしれません。 航空機では、乗客や貨物の積み下ろしで、その重量が変化して、航空機の制御性に影響を与える可能性があります。空域では、入力とは、この空域に入ってくる航空機であり、出力とは、空域から出ていく航空機のことです。 コントロール対象のプロセスのハザーダス状態にあるコンポーネントが、どんな入力（または出力）にも影響を受ける場合、それらのプロセス入力と出力を因果関係シナリオの生成において考慮する必要があります。

コントロール対象のプロセスのコントローラ

明らかに、ハザーダスプロセスのコントローラは、潜在的にハザーダス状態の一因となります。このモデルでは、青色と黄色のボックスで主要なコントローラを示しています。 航空機では、これらは人間のパイロットと自動フライトコントロールシステムであると言えます。 モデルの中では、パイロットとは物理的に航空機に搭乗しているものを意味するのではないことに注意してください。図 G.2 のモデルでは、別のボックス（コントロール対象のプロセスの左側）を示しています。このボックスは、コントロール対象のプロセスあるいはそのコンポーネントの状態に影響を与えるような他のコントローラまたはシステムを表しています。たとえば、メンテナンス活動やその活動が足りないことは、物理的システムでのハザードにつながることがあります。他の自動化コントローラとヒューマンコントローラの影響については、以下で個別に説明します。

自動化コントローラ

この図では、自動化コントローラを一つだけ示しています（青いボックス）が、これは複数個あるかもしれないし、階層構造で設計されている可能性があります。自動化ントローラとヒューマンコントローラのモデルは、常に同じコンポーネントが含まれているので、ここでは一つのみ示します。

非常に単純化したシステムを除いて、自動化コントローラは、コントロール対象のプロセス状態を変更するアクチュエータに対してコントロールアクションを与えます。センサは、現在のプロセス状態に関するフィードバックを与えます。

自動化コントローラからコントロールプロセスへのアクチュエータを介したコントロール経路

このハンドブックの第 2 章で述べたように、このコントロール経路は自動化コントローラからコントロール対象のプロセスへとコントロールを伝送します。コントロール経路は、単純なアクチュエータから構成されていることもあれば、一連の複数のアクチュエータが含まれているかもしれません。あるいは、スイッチ、ルータ、衛星、またその他の装置を有する複雑なネットワークを介して、コントロールアクションを伝送しているかもしれません。どのように実装されていたとしても、ハザードを阻止するために必要なコントロールアクションが適切に実行されない、あるいは不適切に実行される、ある一部の条件下ではハザードになるまでそのアクションが遅延している場合、このコントロール経路に伴う問題が、ハザーダスな制御システムの状態に至るかもしれません。これらの例としては、アクチュエータの故障、コントロール経路に沿った伝達上の問題、コントロールアクションの送信や実行の遅れなどがあります。セキュリティ上の不備に関するハザードとして、攻撃者がハザードを防止するために必要なコントロールアクションを妨害することや、コントロール対象のプロセスの中でハザードにつながるコントロールアクションを入れ込む可能性もあります。

コントロール対象のプロセスからセンサを介する自動化コントローラへのフィードバック経路

アクチュエータのコントロール経路と同様に、フィードバック経路の欠陥は、遅延、測定の不備および不正確さ、センサ故障、またはセンサの設計や動作に関連のある欠陥などを含めた、伝送の問題に起因する可能性があります。 先にも触れましたが、攻撃者がフィードバック経路に悪影響を及ぼしているのかもしれません。自動化コントローラの欠陥のあるプロセスモデルが、非安全なコントロールアクションを引き起こすなら、このフィードバック経路は、因果関係シナリオの生成において考慮されるべきです。例えば、もはやデータが現在の状態を反映しなくなり、コントローラが不正確な情報に作用するほどの地点まで遅れるかもしれません。因果関係解析の生成において、「欠陥のあるプロセスモデル」で止めないでください。それでは、非安全なコントロールアクションに対するセーフガードをもたらすために十分な情報を提供しません。その代わりに、このプロセスモデルに欠陥がある理由を明らかにする必要があります。

自動化コントロールアルゴリズム

自動化コントロールアルゴリズムは、（1）コントロールアクションを生成し、（2）コントロールアクションの生成に影響を与えるようなコントロール対象のプロセスおよび外部システム構成要素および環境の状態に関する正確な情報（モデル）を維持する、といった二つの主な機能があります。これらの機能に欠陥があれば、UCA につながる可能性があります。

1. <u>コントロールアクションの生成</u>

コントロールアルゴリズムそのものにある欠陥、外部から自動化コントローラへの非安全なコマンド、コントロール対象のプロセスおよびその環境における自動化コントローラがあるモデルの中に欠陥があることで、自動化コントロールアルゴリズムは、非安全なコントロールアクションを生成することがあります。 後者の原因については、次のセクションで説明します。

STPA は、自動化コントロールアルゴリズムの安全要求と制約を生成します。もちろん、これらに誤りがないと検証しなければなりません。ここでの因果関係要因は、不十分なコミュニケーションと、コントロールアルゴリズムの運用上の安全要求と制約の検証です。

2. UCA に関連するコンテキストファクターの内部モデルの維持

実際の自動化アルゴリズムが安全要求と制約を満たしていても、実行するコントロールアクションに関する意思決定の安全性は、自動化コントローラがこれらの決定を行うときに使う情報によっても影響を受けることがあります。自動化コントローラには、（1）コントロール対象のプロセスの状態、（2）コントロール対象のプロセスにおける他のコントローラの状態、（3）自動化のオペレーションモード、（4）（洗練された新しいシステムの中には）ヒューマンコントローラの状態、の４種類の情報が用いられます。これらのモデルにあるさまざまなタイプの欠陥は、自動化コントローラによる非安全なコントロールアクションにつながります。 UCA のコンテキストには、モデル内のどのような欠陥が UCA につながる可能性があるのか、といった情報が含まれます。例えば、BSCU 自動ブレーキは、通常の離陸中にブレーキコントロールアクションを出します。この UCA は、コントロール対象のプロセスの状態に対して BSCU が混乱している時に、すなわち、ブレーキ動作コマンドが与えられたときに航空機が通常の離陸状態にあることを検知していないときに発生することがあります。因果関係のシナリオには、BSCU 自動ブレーキが航空機の状態について混乱している理由と、これらの条件下でブレーキ指令が与えられているかもしれない理由が入っています。

コントロール対象のプロセスのモデル

コントロール対象のプロセスのモデルとは、そのコントロール対象のプロセスが存在すると、この自動化コントロールされているものが考えている状態です。このモデルには、コントロール対象のプロセスが動作している環境についての情報があるかもしれません。このモデルはコントローラによって更新されて、それはコントロールアルゴリズムの一部として定義します（これが別個のものである場合は問題ありません[39]）。自動化コントロールアルゴリズムは、コントロール対象のプロセス変数を計測するように設計されたセンサからの状態に関して、直接的に情報を得ます。コントロールアルゴリズムを経ずにプロセスモデルが自動的に更新されるような設計もあります。一例として、スペースシャトルの打ち上げ直前に、特定の飛行データを読み込むことです。ミサイルシステムでもこのようなことを行います。「Engineering a Safer World」の付録 B では、ソフトウエアフライトデータの読み込み時にエラー（typo）から発生した非常に損失の大きい事故について詳しく説明しています。

通常のブレーキおよび自動ブレーキの処理のように、コントロールアルゴリズムがいくつかの部分に分かれている場合、コントロール対象のプロセスモデルの更新では、様々な方法で未知または考慮されていない相互作用による間違いが起きます。UCA につながるシナリオを識別する時は、これらのケースを考慮しなければなりません。

コントロール対象のプロセスモデルを更新する際に、そのコントロールアルゴリズムは、コントロール対象プロセスの状態についての間接的な情報を受け取るか、それに関しての想定をします。一般的な想定の一つとして、コントロールアルゴリズムがコントロールアクションを出すときに、そのアクションが実行されて、プロセスモデルを更新する、と想定することがあります。例えば、航空機上の電子フライトコントロールシステムは、昇降舵を一定の角度だけ動かすアクチュエータのコントロールコマンドを出します。次に、昇降舵が動いたことと、指定された分だけ動いたこととの両方を仮定するでしょう。命令された動きが起こらない理由として、アクチュエータへの通信

[39]自動化コントローラソフトウエアのさまざまな機能を分離させることは有効であるかもしれませんが、ここでは組み合わせています。それらを分離させる理由の一つは、これが論理的に分離されていると、ソフトウエアの設計が容易に実装あるいは変更できるからです。また、ソフトウエアが正しいことを検証することも簡単であるかもしれません。

リンクの欠陥、不適切なアクチュエータのオペレーション、コントロール対象のプロセス（昇降舵自体）の故障または不具合が挙げられます。コマンドの非実行に関するフィードバックがシステム設計の中には含まれていないのかもしれません。または設計に含まれていても自動化コントローラによって受信されなかった、または処理されていないのかもしれません。遅延（例えば、入力が受信されてから処理されるまでの時間、割り込みやポーリングの使用などのソフトウエアキテクチャ設計の決定による影響を受ける）など、さまざまな種類の技術的な設計上の問題もありえます。データの老朽化の問題もハザードにつながります。つまり、使用されたデータが有効ではなくなっている、あるいは出力コマンドが実行中にハザードとなるほどの時点まで遅れてしまっています。後者は、作動経路の遅延、またはコントロール対象のプロセスの条件によって起こります。例えば、ある 1 機の軍用機があり、メンテナンスの作業員が爆弾のベイドアで作業していて、作業の経路にいるときにドアを閉鎖することを防止するための機械的インターロックを作動させてしまいました。同時に、コックピットで作業する他の人間が、爆弾のベイドアを閉める命令を出しました。メンテナンスの作業者が機械的インターロックを解除した数時間後になるまで、このコマンドは実行されませんでした（機械的インターロックのため）。そしてこの作業員は死に至りました。

プロセスモデルと実際のプロセス状態（さらに様々なコントローラのモデル間）との不一致の原因は、他にもたくさん存在します。 理由の一つとして、プロセスモデルの更新が遅れて、これが不一致や、非安全なコントロールアクションへとつながることがあります。最初の起動時には、場合によって正しくないデフォルト値を設計者が想定しているのかもしれません。例えば、飛行機の自動化では、離陸する前に電源を入れて初期化することになっています。その場合、デフォルト値が与えられることがよくあります。しかし、デバイスが離陸より遅く起動されると、デフォルト値が保持されないことがあります。例えば、TCAS は、航空機が地上にある間に起動され、地上にあることに対応した値でシステムを初期化すると仮定します。メンテナンスのためにシャットダウンされ、そのコントロール対象のプロセス状態は、そのシステムがメンテナンス手順のためにシャットダウンされたときと同じであると想定して、再起動されるようなデバイスのプロセスモデルについても同様です。他の例としては、ヒューマンコントローラが、正常に動作していないと思われるデバイスを再起動しようと決めることがあるかもしれません。例えば、TCAS では、パイロットが空中で再起動させることが可能です。ただし、フライトのある時点で正しくないかもしれない初期化値でデバイスが再起動されます。プロセスモデルを更新する入力は、デバイスの電源がオンになる前、シャットダウン後、あるいはデバイスが切断（オフライン）されている間にも入ってくることがあり、したがって無視されるかもしれません。

オペレーションモードのモデル

運用モードのモデルも、UCA で定義されたコンテキストに関連することがあります。 （1）コントロール対象のプロセスモード、（2）自動化モード、（3）監視モード、（4）表示モードの 4 種類のモードを考慮しなければならないでしょう。

1. コントロール対象のプロセスモード

コントロール対象のプロセスモードでは、そのコントロール対象のプロセスの運用モードを識別します。例えば、ある航空機（コントロール対象のプロセス）では、離陸モード、着陸モード、あるいはクルーズモードです。モデリングに必要なモードは、UCA のために識別されたコンテキストによって決まります。例えば、この航空機が（着陸モードではなく）クルーズモードにある場合、逆噴射の動作は、非安全であるかもしれません。このコンテキストは、UCA コンテキストテーブル（すなわち、航空機が着陸モードにないときの逆噴射の動作）で識別される必要があります。 UCA の根拠（因果関係シナリオ）は、コントロール対象のプロセスの現在のモードについて、そのオートメーションが混乱していることが少なくとも部分的に関連しているでしょう。

2. 基本的な自動化オペレーティングモード

コントロール対象のプロセスに加え、自動化自体（図 G.2 で自動化コントロールされている）に運用モードがあります。それらのモードは、しばしば、コントロールアルゴリズムのロジックに使われており、その出力に影響を与えます。例えば、ノミナル（正常）動作モード、シャットダウンモード、欠陥(fault)処理モードです。一例として、自動化は、それ自体で、またはコントロール対象のプロセスで欠陥を検出した後に、部分的シャットダウンモードになるかもしれません。Air France 447 の損失(loss)では、航空機の外側にある圧力プローブ（ピトー管）が氷結し、オートパイロットが航空機はどのくらい速く動いているのかを判別できなくなったとき、オートパイロットが自動的にシャットダウンしました（シャットダウンモードに入った）。フライバイワイヤシステムは、これと同時に運用を続けましたが、もはや空気力学的失速に対する保護を与えないモードへ切り替えられました。最新のエアバス航空機では、電子フライトコントロールシステムの運用モードには、通常、代替、直接、および機械的法則があり、その各々が自動フライトコントロールシステムの動作を変更します。

自動化コントローラの部分的なシャットダウンモードによって、ヒューマンコントローラは非常に混乱することがあります。これは後にヒューマンコントローラが間違いをもたらすようなことを覆い隠してしまうでしょう。モードの混乱は、コントロール対象のシステムのモードと、このコントローラの考えることが異なっている場合に発生します。モードの混乱は、多くの事故に寄与し、これは因果関係シナリオを生成する際に考慮しなければなりません。

欠陥(fault)処理モードでは、自動化コントローラは、コントロール対象のプロセスまたはそれ自体に欠陥があり、安全なコントロールを提供するためには異なった振る舞いが必要であると信じて、振る舞いを変更することがあります。

3. 監督(supervisory)モード

このモードでは、自動化コントローラに対して複数の監督者がコントロールしている（コントロールコマンドを出している）可能性があるとき、誰あるいは何が、自動化をコントロールしているのかを識別します。基本的に、監督モードでは、複数の監督者の間で、コントロールアクションの実施を調整することができます。例えば、航空機内の飛行誘導システムは、人間のパイロットによって直接的な指令を出すことができますし、あるいは人間または自動化のいずれかによる他のシステム構成要素からのコマンドを受信することもできます。図 G.2 は、一般性のために、自動化コントローラ全てを 1 個のボックスに抽象化していますが、普通は、自動化コントローラは複数存在しています。ある一個のコントローラは、複数の他のコントローラを階層的な構成でコントロールしていることもあり、また複数のコントローラを、並列に、またはその両方で運用させても良いかもしれません。もちろん、これは非常に複雑になって、複数のコントローラが、それぞれに別のコントローラなどによって監督されているかもしれません。

4. ディスプレイモード

このモデルは、自動コントローラのヒューマンコントローラに対して表示される情報のタイプを識別します。現在のディスプレイモードは、ユーザーの解釈の仕方と同様に、与えられる情報に影響を与えます。図 G.2 におけるヒューマンコントローラのモデルのコンポーネントの 1 つは、人が現在見ているディスプレイモードです。これらのディスプレイモード（人間およびオートメーションにおける）が矛盾する場合は、深刻な問題が起こる可能性があります。すなわち、一方または両方の振る舞いの一部分がシステムのハザードにつながる可能性があります。このようなモードの混乱は、「インタフェース解釈モードの混乱」と呼ばれており、「Engineering a Safer World」の第 9 章で詳しく説明します。

ヒューマンコントローラのモデル

最近では、眠気や不注意などのヒューマンコントローラの状態のモデルが、その振る舞いを決めるために自動化コントロールアルゴリズムによって使われています。ヒューマンコントローラの状態を測定するため、数多くのセンサが使用されています。例えば、自動車は自動化コントローラへ入力する様々なタイプのセンサによって、運転手が酔っ払っているのか、不注意であるのかを検出しようとします。図 G.2 では、センサからヒューマンコントローラへ伸びている点線を描いていますが、これは、人間が、センサが作動しているかどうか、または感知された変数の状態（例えば、運転者の酩酊レベル）がどうかについてフィードバックを得る設計であることを示しています。

他のコントローラのモデル

自動コントロールに複数の潜在的なコントローラが存在している場合は、それらの複数のコントローラにとって重要な状態変数についてのモデルが必要かもしれません。

環境的入力

ヒューマンコントローラではなく、自動化コントローラに直接環境からの入力があるかもしれません。たとえば、今日のオートメーションでは、位置に関する情報が GPS から直接的に得られます。GPS 位置情報は、その環境での信号から（例えば、アンテナを介して）計算されて、ヒューマンコントローラを介さずに自動化コントローラへと向かいます。最新の航空機には、ADS-B（Automatic Dependence Surveillance）と呼ばれるものがあり、これはヒューマンコントローラを経由せずに、異なる航空機間での情報のやりとりに使用することが可能です。これには、コントロール対象のプロセスの中またはプロセス上のセンサを介して送信され、コントロール対象のプロセスのモデルの一部である、航空機自体の状態に関する情報（例えば、ピトー管からの入力）に関する情報は含まれていないことに注意してください。

他のコントローラ/システムのモデル

現在のいくつかの複雑なシステムや将来のシステムでは更に複数の自動化コントローラが存在するでしょう。例えば、航空機の自動化での人間による通常のコントロールが、その航空機自体の外にある航空機自動化のコントロール（通常は地上にあるが、自動化された航空機が繋がれているような、もう一つ別の航空機にもあるかもしれない）と共にある可能性もあります。これは人間や自動化である可能性があります。これらのコントロールの責任は、別々のコントローラによって共有されているかもしれません。図 G.2 は、1 台のコントローラ（黄色内）と、各コントローラが同じ内部の詳細な記述を持つために、オートメーションに接続された一つのボックス内に他のコントローラが存在している可能性を表しています。

動作モードの説明で述べたように、自動化では、いつ、どのコントローラがそれをコントロールしているかを知る必要があるでしょう。共有コントロールを使用すると、競合するコマンドが発生する可能性があります。このような混乱は、オートメーションのモデルとそのコントローラのモデルとの間の矛盾から生じ、UCA につながるかもしれません。起こりうる矛盾と混乱の種類は、因果関係シナリオの生成の重要な要因となります。

オートメーションとその監督者の間の情報の伝達

ヒューマンコントローラは、様々な種類のコントロールを介してコントロールコマンドを送信し、ディスプレイを介して自動コントローラからのフィードバックを得ます。繰り返しますが、ある自動化コントローラとそのコントローラとの間の情報の間違った伝達は、因果関係シナリオ生成プロセスの重要な（単純ではありませんが）部分です。

コントロールアルゴリズムを経由しない自動コントローラへの直接的な変更

現在のシステムの中には、また将来のシステムではさらに一層多く、コントロールアルゴリズムや他のソフトウエア、ならびにオートメーションのコントロールアルゴリズムで使用されるすべての内部モデルは、ヒューマンコントローラの知らないうちに外部から直接変更できる可能性があります。たとえば、一部の航空機では、パイロットや航空会社のメンテナンス担当者が関与していなくても、新しいソフトウエアがインターネット上の航空機で更新される可能性があります（たとえば、OEM から生じる変更など）。この変更は、自動コントローラのマップやその他の情報のように、またヒューマンコントローラによって間接的に（ディスプレイを介して）、コントロールアルゴリズムそのものや、そのアルゴリズムによって使用されるモデルなどの両方に影響する場合があります。ここには明らかに、潜在的な安全性、特にセキュリティ問題があります。そのような変更は、UCA のコンテキスト要因を生成する恐れがあります。I

ヒューマンコントローラ

ヒューマンコントローラは、*図 G.2* のモデルの中でも、最も複雑なコンポーネントです（それなのにシステム設計者は自動化コントローラを更に複雑にしています）。したがって、これが因果関係シナリオ生成プロセスの最も重要な部分です。そのコンポーネントの機能に関連する、コントロールアクションの生成、メンタルプロセッシング、およびメンタルモデルを含めて、このモデルには多くの関連コンポーネントがあります。

コントロールアクションの生成／メンタルプロセッシング

これらの 2 つの機能は、異なるコンポーネントとしてモデル化することができますが、それらを分離することで重要な相互作用が失われる可能性があります。したがってここでは、一緒にまとめておきます。このコンポーネントには、コントロールアクションの生成とメンタルモデルの更新の 2 つの基本機能があります。

<u>コントロールアクションを生成する</u>：

トレーニング、手順書、および経験はもちろん、*図 G.2*（環境、自動化、およびコントロール対象のプロセス）で示したヒューマンコントローラにあるすべてのモデルからの情報も使って、コントロールアクションを生成します。コントロールアクションの生成は、人間のオペレータが対話する他のコントローラからのコマンドによって影響を受けます。例えば、パイロットは、航空会社のオペレーションセンターから受信した、または航空交通管制や TCAS（Traffic Alert and Collision Avoidance System）[40]などの搭載機器やシステムのような他のコントローラから受信したコントロールアクションや情報によって部分的にコントロールされます。コントロールの責任についての記述が明確でない場合、非安全なコントロールアクションの因果関係シナリオと同様に、調整すべき難題が多くなります。例えば、FAA は、航空管制に関するいくつか特別の決定に関して、パイロットと航空会社のオペレーションセンターとの関係を 50/50 と定義しています。責任についての慎重な説明（最終的に航空会社の運営管理者の責任になりうる）がないと、誰が決定を下すのかについての混乱は、ハザードや事故につながる可能性があります （これは実際に起こっています）。

もちろん、コントロールアクション生成を通じて、非常に多くの誤った振る舞いがみつかる可能性があります。現行のコンテキストを識別するためにヒューマンコントローラが使用するプロセスとともに、コントロールアクションが非安全なコンテキストを識別するための UCA 生成プロセス

[40] TCAS は（図 G.2 の階層的コントロール構造の中間にある）自動コントローラとみなすことができるが、パイロットは TCAS 回避指示（コントロールコマンド）に従う必要があり、ある面では TCAS がパイロットをコントロールしています。ただし、パイロットは、TCAS の設定を入力して、どのように回避指示を生成するかをコントロールできます。航空機や他のシステムにおいて、さらに多くの機能が自動化されるにつれて、人間とオートメーションとの間の様々なタイプの「パートナーシップ」が必要となり、さらに相互に連携していくことが必要でしょう。

（その結果は UCA 表に文書化されています）は、生成された因果関係シナリオの数を制限するように働きます。シナリオ生成のプロセス中に、アナリストは、UCA に関連付けられた特定のコンテキストがどのように発生するのか、なぜコントローラが現在のコンテキストについて混乱しているのか判断する必要があります。

ヒューマンコントローラが、あるシステムでは、オートメーションを介さずに、コントロール対象のプロセス（ヒューマンコントローラとアクチュエータとセンサとコントロール対象のプロセスとの間に点線で示す）と直接やり取りできます。

メンタルモデルを更新する：

ヒューマンコントローラの主な責務は、コントロールアクションを生成することに加えて、そのメンタルモデルを更新することです。メンタルモデルは、人間の思考の処理によって常に更新されていますが、この処理の一部は無意識であるかもしれません。したがって、モデルには双方向の矢印があります（人間はメンタルモデルの情報を更新して使用もしますが、物事は両機能で間違っていることがあります）。

この更新中に人間が使用する情報は、一部はディスプレイからであり、また一部分は、そのコントローラや、これに互いにやり取りする他のコントローラからの入力（例えば、航空交通管制や航空会社のオペレーションセンターからの情報）から得られており、また部分的には直接的な感覚入力（例えば、窓から外を見て、竜巻が航空機に接近するのを見たり、乱気流を感じること）からも情報が得られます。例えば、パイロットは、航空会社のオペレーションセンターまたは航空交通管制のような他のコントローラから送られた情報（NOTAM のような）を受信します。いくつかのシステムでは、アクチュエータからヒューマンコントローラまでの点線で図 G.2 に描かれているように、人間のオペレータは、作動しているアクチュエータを直接観察する能力を持っています。例えば、あるオペレータがプラントの中でアクチュエータおよびバルブまで歩いていき、その開閉を監視して、適切なオペレーションを確認することができます。別の例として、パイロットは通常、離陸前、あるいは、飛行前の検査のあいだに、歩きまわって、コントロール装置を動かし、飛行機のエルロンを直接目で見て、正しい方向に動いているかを確認しています。ヒューマンコントローラはまた、コントロール装置の状態から入力を受けとっています。例えば、自動車のドライバーは、自分でハンドルの動きを見て、自動化がステアリングに関するコントロールコマンドを出したのだろうと推測しているかもしれません。

人間のオペレータ（コントローラ）が受ける訓練の影響を受けて、モデルが更新されることもあります。ある微妙なタイプの入力は、パイロットがコントロールを操作したときのコントロールの反応から来ることがあります（図 G.2 の点線で示されています）。オペレータは、自動化コントローラおよびコントロール対象のプロセスの状態について、航空機の操縦桿の動きまたは自動化されている自動車上のステアリングホイールのようなコントロールの動きに基づいて推測をおこなうかもしれません。

モデルの更新とコントロールアクションの生成との相互作用：

自動化コントローラとそのプロセスモデルの更新のように、自動化コントローラの説明と同様、コントロール生成プロセスとヒューマンコントローラのメンタルモデルとの間に重要な相互作用が存在します。例えば、あるヒューマンコントローラが「DO X」とコマンドを出したとしたら、その他のヒューマンコントローラが、この「DO X」コマンドは自動化コントローラとコントロール対象のプロセスのアクチュエータによって実行されている、と仮定し、無意識に、オートメーションおよび/またはコントロール対象のプロセスの状態を表す自分のメンタルモデルを更新する可能性があります。実際には、「Do X」コマンドが実行されない場合、ヒューマンコントローラのプロセスモデルは、オートメーションおよび/またはコントロールされたプロセスの実際の状態と矛盾します。コマンドが実行されない理由はたくさんあるでしょう。例えば、オートメーションによってコマンドが非安全である、あるいは、その時点で間に合わせるのは不可能であると判断された場合や、コントロール対象のプロセスで欠陥や故障が発生するかもしれない場合などです。

プロセスモデルの同種の誤りのある更新については、自動化コントロールアルゴリズムのセクションで説明しました。しかし、自動化の場合、このような問題が発生しないことを確認するためにアルゴリズムをチェックすることができます。人間のメンタルプロセッシングからこれを排除するのは、より難しくなります。

またコマンドの非実行に関するフィードバックがヒューマンコントローラに与えられるかもしれませんが、注意散漫などの理由で人間は気づかないかもしれません。

ヒューマンコントローラが使用するメンタルモデル

図 G.2 には、ヒューマンコントローラとして、4 つのタイプのメンタルモデルを示しています。環境のモデル、自動化の状態のモデル、コントロール対象のプロセスのモデル、および他のコントローラのモデルです。これらのモデルのいずれかが実際の状態と矛盾し、非安全なコントロールアクションに至るかもしれません。これらのモデルには、似たような内容があり、自動化コントローラのモデルと同様の用途がありますが、欠陥になる理由については、かなり異なっているかもしれません。その因果関係シナリオは、UCA のコンテキスト条件を引き起こしうる、危険な矛盾を生み出す潜在的な理由（その原因）を含まなければなりません。

ヒューマンコントローラは、通常、自動化（直接的に）とコントロール対象のプロセス（間接的であるが場合によっては直接的に）の両方をコントロールするため、人間には両方の状態のモデルが必要です。

オートメーションを監督するために、人間は、どのように自動化が動くかという基本的な理解が必要です。これについて混乱すれば、非安全なコントロールと損失につながります。

人間のメンタルモデルが正しくないかもしれない理由はたくさんあります。自動化コントローラでモデルが正しくないとなる理由は、これまでものが当てはまるかもしれません。しかし、人間のプロセッシングは他にも多くの機会を示しています。ヒューマンファクターの専門家達を、ヒューマンコントローラを含めた、因果関係シナリオ生成の一部とすることを強く推奨します。考慮すべきいくつかの要素（ただし、ほんの一部）を以下に示します：

- モードの混乱とは、人間が自動化のモードで混乱したり、コントロール対象のプロセスの現在のモードについてオートメーションが混乱するといったような、モードがたくさんあるシステムでの事故の一般的な原因です。モードの混乱は、プロセスモデルの更新の誤りによって、あるいは入力が正しくないか遅延していることによって、また更新が遅れたり、ヒューマンコントローラにモード変更を通知しても、これに気づかなかったり、その情報を処理しないことによって、引き起こされることがあります。ヒューマンコントローラが指示することなく、コントロール対象のプロセスのモードをオートメーションが変更できる場合、モードの混乱と潜在的に非安全なコントロールアクションが生じる可能性があります。
- 「状況認識」は、一般的に、その用語の意味を慎重に定義することもなく、事故の原因としてあげられています。非常に多数のエラーが、このカテゴリに該当します。 STPA で使用されるモデリングは、ヒューマンコントローラのメンタルモデルが実際の状態と一致しないことを単に意味するため、状況認識エラーを識別することができます。
- 非決定論的であるとか、ほとんどの場合に一つの方法で動作するのに特殊なケースでは違った動作をとるようなオートメーションによって、人間は簡単に混乱することがあります。このようなオートメーションではまた、訓練の困難さが大幅に増加します。
- 一部のオートメーションでは、そのロジックが、ヒューマンコントローラが意図していない、または知らない副作用を引き起こすようにプログラムされています。
- システムの設計に、人間に対するフィードバックが含まれている場合がありますが、ほとんど変化しなかったり、間違っていることがほとんどないものを監視する際には、注意散漫や注意力低下などのように、フィードバックに気づかない理由はたくさんあります。

- 今日の自動化されたシステムでは、オートメーションに対して、人間による不満や過度の依存性がますます問題になっています。
- オートメーションがあまりにも少しずつ正常に機能しなくなっていくため、ヒューマンコントローラがプロセスの後半になるまで、問題に気付かない場合があります。人間はまた、オートメーションが停止しているのか、故障していないのに故障していると考えているかもしれません。この種の問題は、ロボットと人間が同じ分野で働かなければならない場合に発生しています。エネルギーをオフにしていると人間が考えているが、実際にはオンであるといった、ログアウト/タグアウトの問題は、職場での多数の事故につながります。

このリストは、人間がシステムの一部であるときに、考慮すべき多くの原因があることを示すためのものです。それは網羅的ではありません。もし、排除または制御するために、因果関係シナリオをこのような詳細なレベルにする必要がある場合は、ヒューマンファクターの専門家と協力して作業することをお勧めします。

最終的な考察

この付録で扱っているモデルやガイダンスは、因果関係シナリオを生成する際のチェックリストとして使用してはいけません。特に、ユニークなシステムに対しては、ほぼ確実に不完全です。システムがどのように動作するのかは、自分で慎重に考える必要があります。我々は、システムの設計と運用に関する専門家を、因果関係シナリオの作成に参加させることが非常に役立つことに気がつきました。しかし、この付録のモデルは、初歩的な見落としを防ぎ、より完全なシナリオを識別するのに役立ちます。

また、システムの潜在的な欠陥をすべて取り上げて、UCA（基本的に FMEA で使用されているプロセス）につながるかどうかを確認しようとすることは、複雑なシステムでは絶望的です。常に、ハザード、そして UCA とそのコンテキスト上の要因から始めて、それから後方に向かって潜在的な原因を識別します。依然として非常に多いかもしれませんが、誤りを起こす可能性のあるすべてのものを識別し、それが UCA につながるかどうかを調べる努力よりは桁違いに小さくなるでしょう。