HASHIMOTO
SOFTWARE
CONSULTING
INTERNATIONAL Inc.

STAMP/STPA の理論と実践

《入門編》

2016年12月14日

Takanari HASHIMOTO

アジェンダ

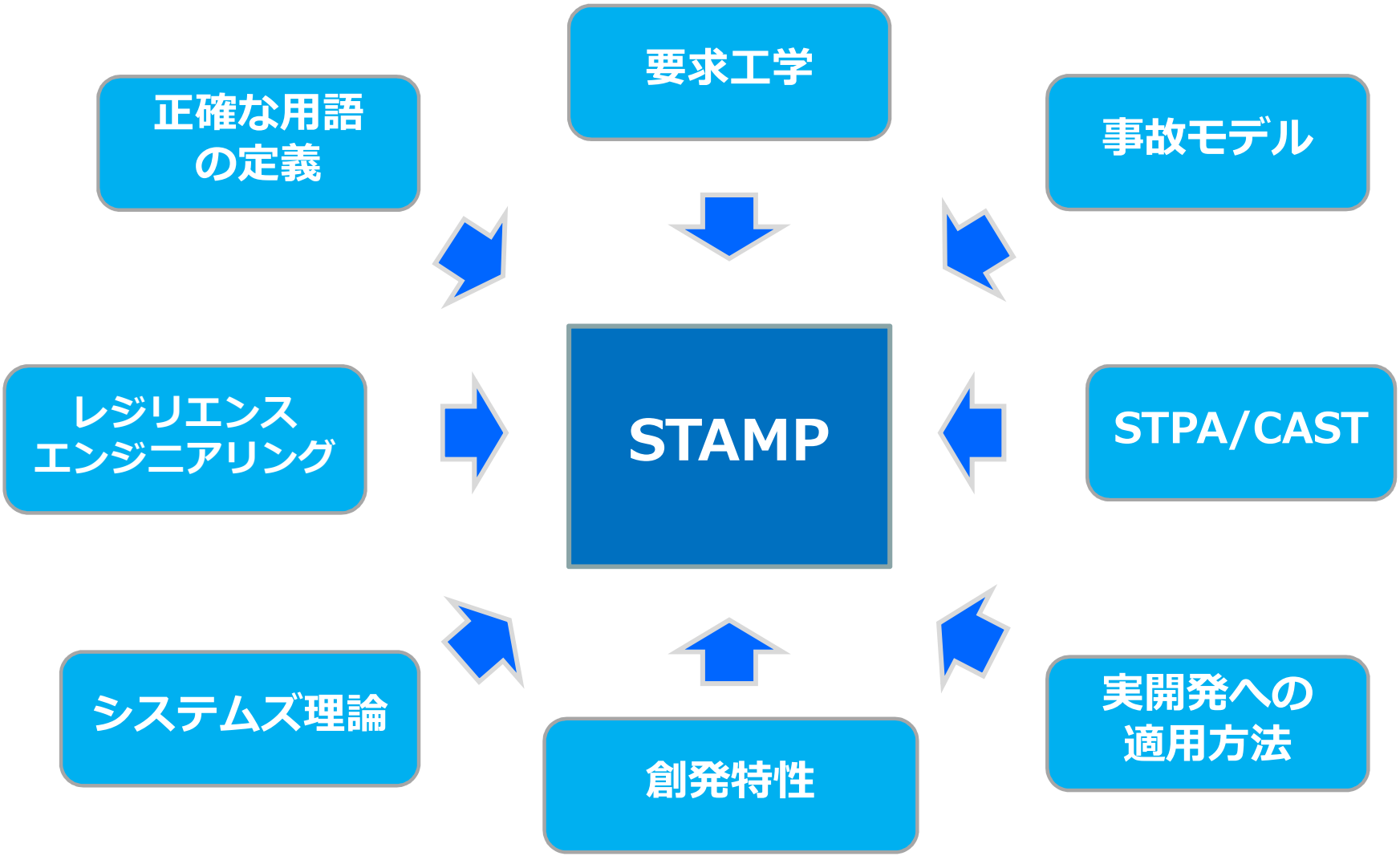
HASHIMOTO
SOFTWARE
CONSULTING
INTERNATIONAL Inc.

アジェンダ
• 《第1部》STAMP/STPAの背景と理論 • 《第2部》STPAの分析プロセス

《第1部》
STAMP/STPAの背景と理論

STAMPを理解するための前提知識と周辺知識

HASHIMOTO SOFTWARE CONSULTING INTERNATIONAL Inc.



事故モデル

HASHIMOTO
SOFTWARE
CONSULTING
INTERNATIONAL Inc.

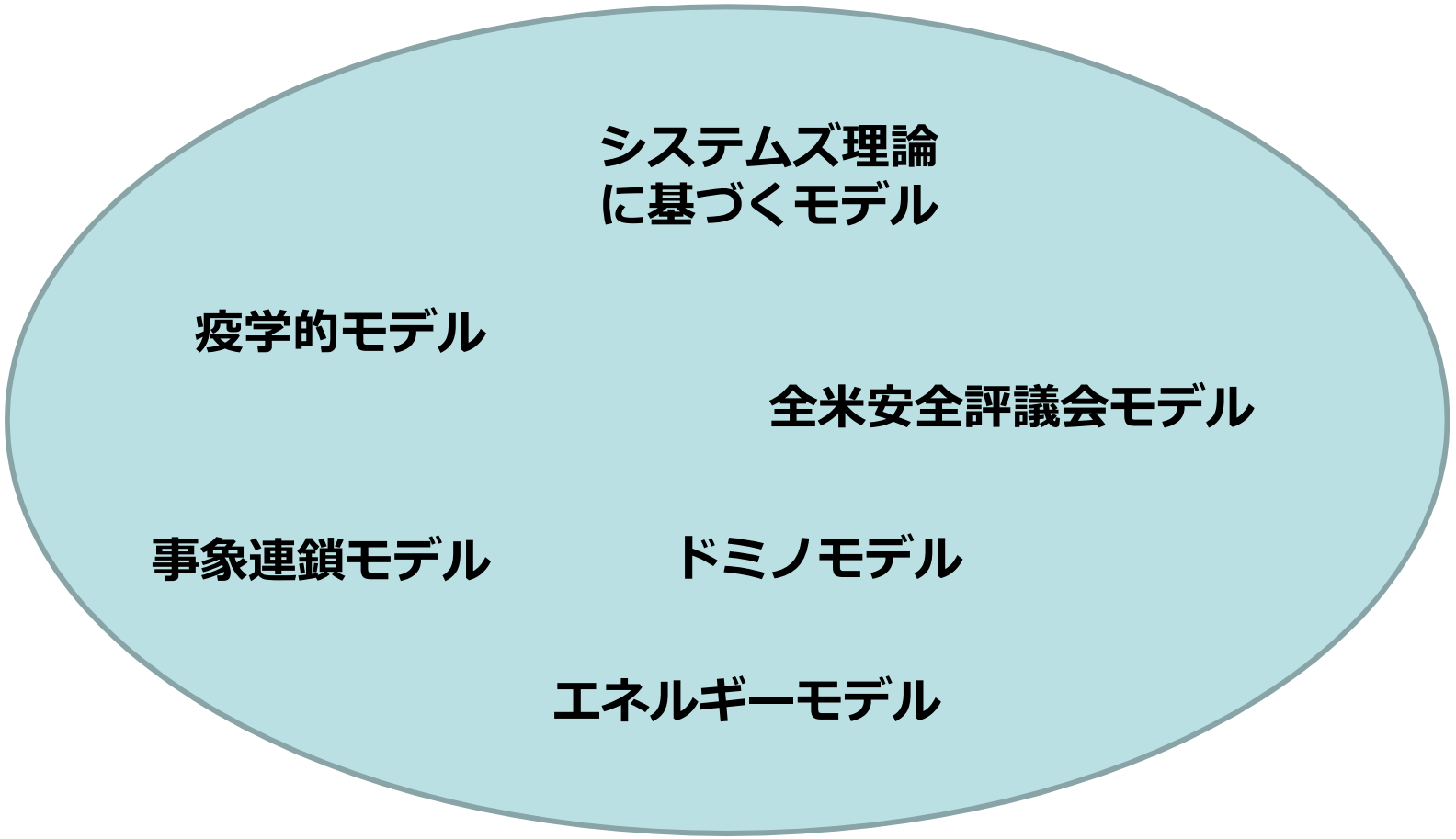
事故モデルと技法(手法)

- **事故モデル**の目的：
  - 事故の発生の原因とメカニズムを記述する
  - 過去の事故を分析する
  - 未来の事故を防止方法を検討する

事故モデルの種類

HASHIMOTO
SOFTWARE
CONSULTING
INTERNATIONAL Inc.

代表的な事故モデル



*他にも多数の事故モデルが存在する

事故モデルの種類

HASHIMOTO SOFTWARE CONSULTING INTERNATIONAL Inc.

事故モデル	内容
ドミノモデル	• **具体的&決定的な原因**が１つ存在し事故をもたらすと考えるモデル • 事故の解決＆防止には根本原因を突き止めれば良いという考え方 ➢ 電池の寿命を不確認⇒電池交換せず⇒電池切れ⇒時計停止 • **RCA(Root Cause Analysis:根本原因分析)型**の事故分析手法は,このモデルを基礎にしている
疫学的モデル	• **多因子的事故モデル** • 事故は**作用要因**,**環境**,**被害者**によって説明される • **安全意識が低く**,仕事への**モチベーションも低い**職場や社風に問題があると考えるモデル ➢ その点を改善しない限り事故の解決＆防止につながらない ➢ **疫学的**と言われるのは,事故を『体力が落ち病気が発病して,そして周囲に伝染・蔓延していく』の比喩と言われる • **『スイスチーズモデル』**が有名
エネルギー伝達モデル	• 事故は不制御の**エネルギーの放出**の結果と考えるモデル • エネルギーの**防護壁**を設けることや,エネルギーの**流れを制御**するメカニズムを使用することで事故を防ぐ考え方

＊他にも多数事故モデルがあるが存在する

事故モデルと技法

HASHIMOTO
SOFTWARE
CONSULTING
INTERNATIONAL Inc.

事故モデルと技法(手法)

- 解析技法や分析技法などは**理論的基盤となる事故モデル**が存在する

STAMPとシステムズ理論

HASHIMOTO SOFTWARE CONSULTING INTERNATIONAL Inc.

用語	特徴
STAMP (Systems-Theoretic Accident Model and Processes)	• MITのNancy G.Levesonが発表した ➢ レブソン教授が**ソフトウェアセーフティ**の研究で考案した • **『システム理論』**に基づく**『事故モデル(事故説明モデル)』** ➢ **ソフトウェア**／**ハードウェア**／**人**／**外部環境**を含む事故モデル ➢ **複雑な社会技術システム**に対応できる新しい事故モデル ➢ **ソフトウェア集約型システム**に適した事故モデルと言われる ➢ ヒューマンエラー／プロジェクト間の**連携ミス**などによる事故原因においても特定や説明が可能 • システムの**安全性**と**セキュリティ**は**『創発特性』**である • **安全性**は『不具合問題』ではなく**『制御問題』**として定義する

STAMP(事故モデル)

システムズ理論(科学理論)

システムズ理論

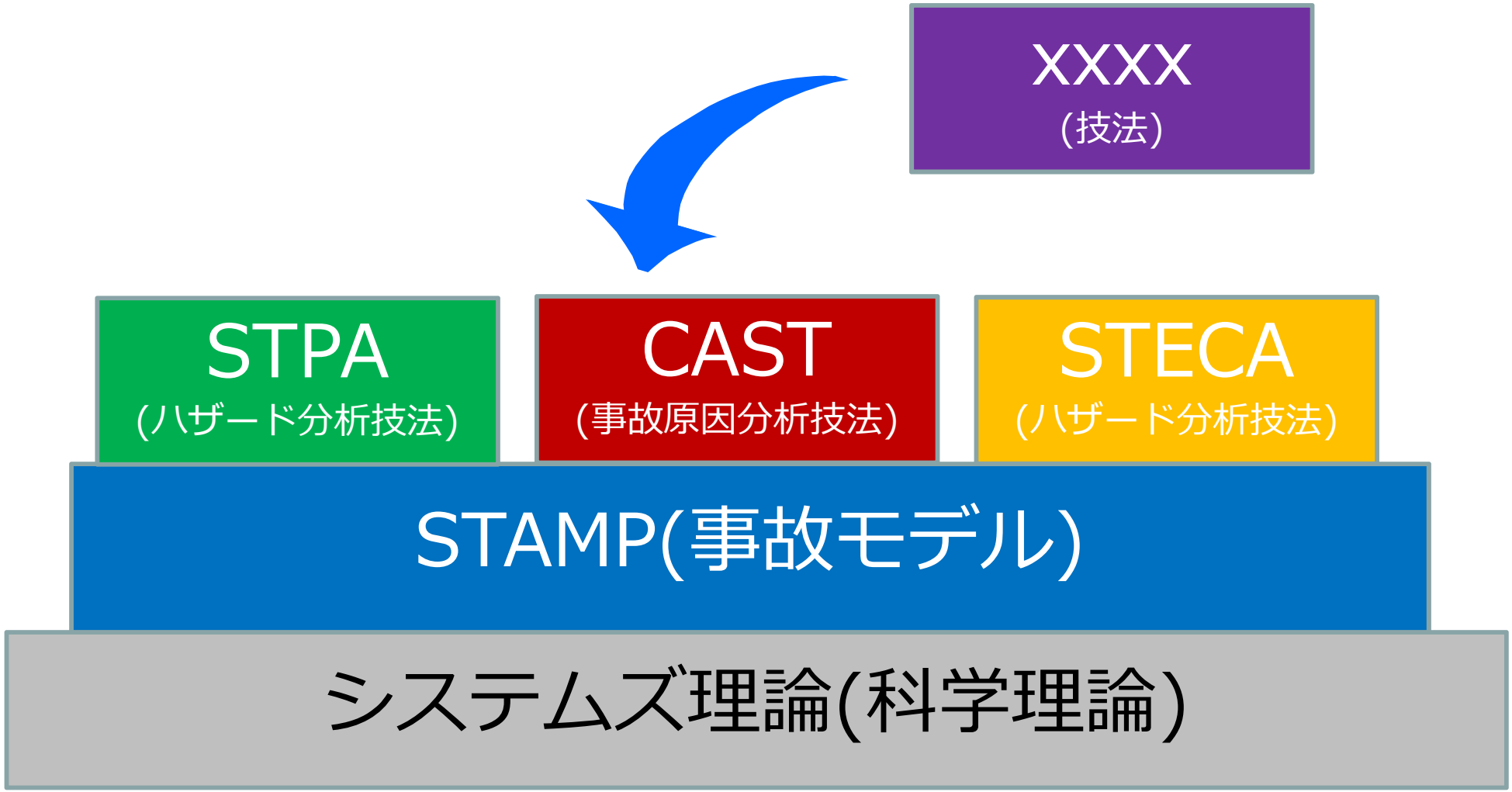
HASHIMOTO
SOFTWARE
CONSULTING
INTERNATIONAL Inc.

用語	内容
システムズ理論 (システムズ アプローチ)	• 『システムズ理論』は一般名称である ➢ システムズ理論の色々な方法論が提唱 • **古典的な『還元主義』**では**複雑なシステムの対処に限界**を主張している ➢ 生物系／社会システムなど複雑なシステムを対象 • **現象**をシステムと捉える • システムの構成要素や部分を個別に扱わず**システム全体**を扱う ➢ システム全体の振る舞いの分析や設計を重視 • **創発特性** ➢ 全ての局面と変数を考慮に入れ,社会的見地や技術的見地を関連させて全体を扱う時にのみ適切に扱えるシステムの**特性** ➢ **創発特性**はシステムを構成する部分の相互作用から出現 ➢ 『**部分の合計より全体が大きい**』

STAMPと技法の関係

HASHIMOTO
SOFTWARE
CONSULTING
INTERNATIONAL Inc.

STAMPに基づく新しい技法の作成が可能



STPA／CAST／STECA／STPA-Sec

HASHIMOTO SOFTWARE CONSULTING INTERNATIONAL Inc.

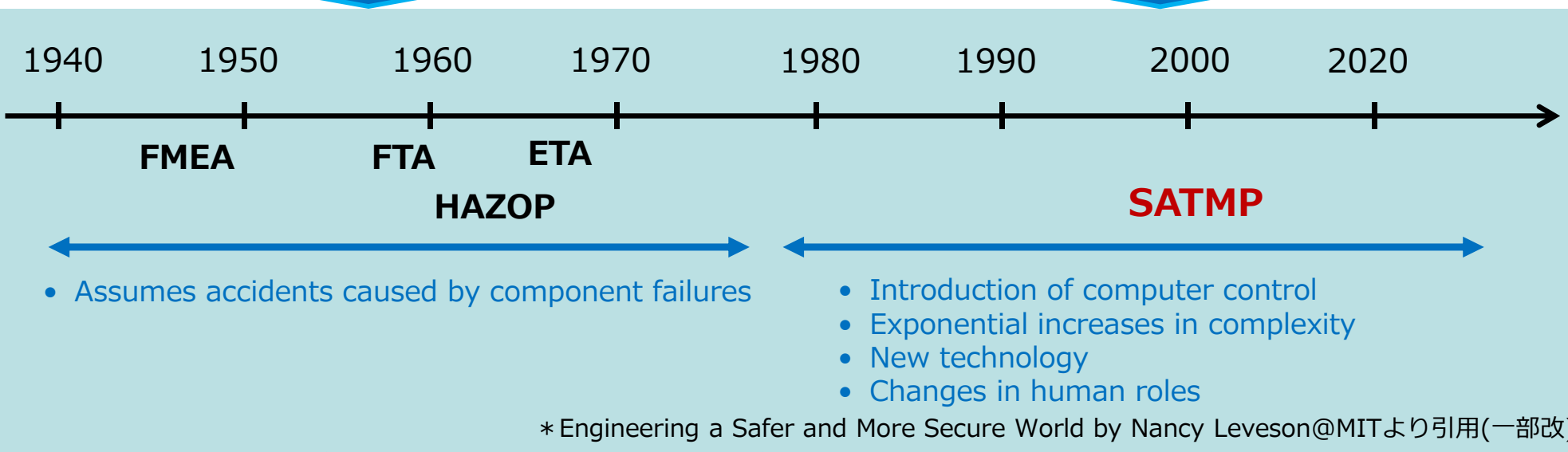
技法	内容
STPA (System-Theoretic Process Analysis)	• STAMPに基づく**ハザード分析手法** • 人や環境を含めた社会活動などのハザード分析が可能 • **トップダウンアプローチ** • **コントロールプロセス**にフォーカスして分析する ➢ FTAやHAZOP技法は**メカニズム**にフォーカスする
CAST (Causal Analysis based on STAMP)	• STAMPに基づく事故発生の**原因分析手法** • **ボトムアップアプローチ** • 人や環境を含む社会活動の原因分析が可能 ➢ 『ピーナッツバターによる集団食中毒 』の原因分析 ➢ 『職場での転倒事故の労働災害』の原因分析
STECA (System Theoretic Early Concept Analysis)	• システム開発の早期の段階から実施可能な**ハザード分析手法** • **トップダウンアプローチ**
STPA-Sec (STPA for Security)	• **サイバーセキュリティ**に特化した**ハザード分析手法** • **トップダウンアプローチ**

古い技法から新しい技法へ

HASHIMOTO SOFTWARE CONSULTING INTERNATIONAL Inc.

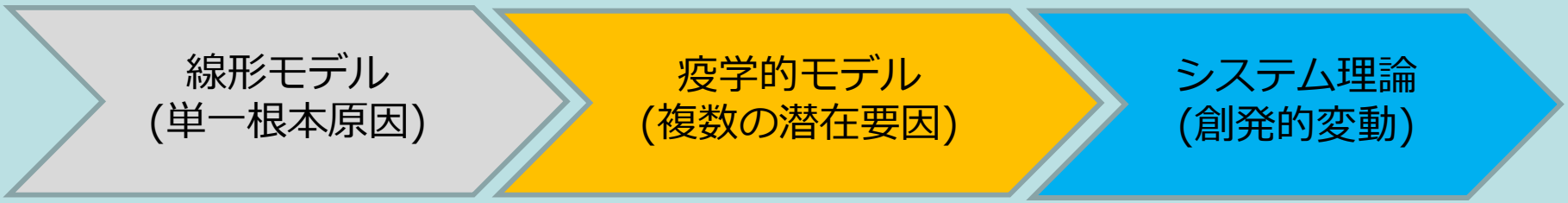
『信頼性工学』に依存した理論と技法
インターネット利用以前の理論と技法

IoT時代の複雑に関連し合うシステム
に適した理論と技法が必要な時代



＊Engineering a Safer and More Secure World by Nancy Leveson@MITより引用(一部改)

事故モデル

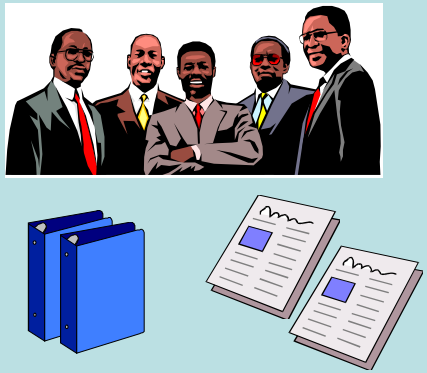


STPAと作業／成果物の関係

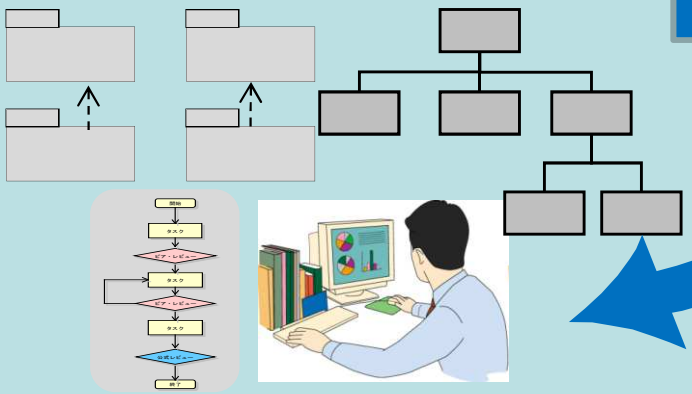
HASHIMOTO
SOFTWARE
CONSULTING
INTERNATIONAL Inc.

要件仕様に反映

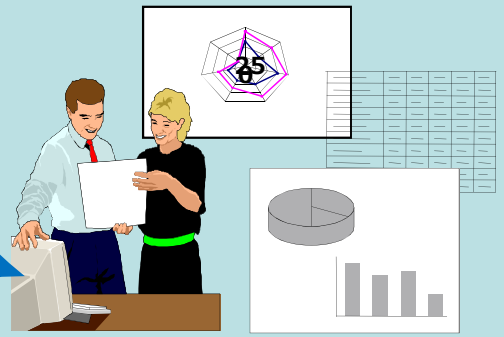
- ハザード分析結果
- 原因分析結果
- 安全解析結果



運用マニュアル／規約に反映



アーキテクチャ設計／メカニズム設計／実装に反映



テスト計画／テスト仕様に反映

ハザード分析・信頼性解析技法

HASHIMOTO SOFTWARE CONSULTING INTERNATIONAL Inc.

技法	内容
FTA (Fault Tree Analysis)	• 1961年にBell研究所で開発 • 「フォルトツリー解析」と呼ばれる • ハザード原因分析手法 • トップダウン・アプローチ
FMEA (Failure Modes and Effects Analysis)	• 「故障モード影響解析」と呼ばれる • ハザードやリスク分析技法よりも信頼性解析の技法と言われる • ボトムアップアプローチ
MORT解析 (Management Oversight and Risk Tree)	• Johnsonにより提唱された技法 • 「管理監督リスクツリー」と呼ばれる • 1970年代米国原子力規制機関のために開発 • 事故原因分析とハザード分析に利用できる
HAZOP (Hazards and OPerability Analysis)	• 1960年代初期に英国のImperial Chemical Industriesによって開発 • 後に改良されロンドンの化学工業協会により公開

＊ハザード分析・信頼性解析技法は他にも多数存在する

STAMPのシステム安全の用語

HASHIMOTO SOFTWARE CONSULTING INTERNATIONAL Inc.

従来のシステム安全の用語とSTAMPの用語定義	
STAMP以外	• システム安全に関する用語の定義に**一貫性が乏しい** • 国や産業毎に,用語定義や使用方法に違いがある ➢ エンジニアリング,コンピュータ科学,自然言語(日常生活)及びその他で**同じ用語が異なる定義・意味**で使われている
STAMP	• **エンジニアリング分野**の用語定義や使い方を基本にし,定義し直している ➢ 明確で厳密な用語の定義をしている ➢ 用語の中にはコンピュータ科学分野の定義とは異なる定義や使い方があるので注意が必要

STAMPの用語①

HASHIMOTO SOFTWARE CONSULTING INTERNATIONAL Inc.

用語	定義と説明
信頼度 (reliability)	• 装置やコンポーネントが定められた環境条件下で,規定の期間中に意図された機能を十分に遂行できることを**確率**で表現した特性のこと ➢ **信頼性工学**は主として故障と故障率の低減を扱う ➢ 信頼性工学の手法は事故原因としての故障に注目する ➢ 安全は故障より広い範囲を指す ➢ **信頼性分析**は故障に関連する事故の可能性だけを考える。 ➢ 信頼性分析は個々のコンポーネントが問題なく運用された場合の潜在的損害は調査しない
不信頼度 (unreliability)	• 不信頼度は**故障確率**のこと

STAMPの用語②

HASHIMOTO SOFTWARE CONSULTING INTERNATIONAL Inc.

用語	定義と説明
故障 (failure)	• システムまたはコンポーネントが定められた環境条件下で,定められた期間中に意図された**機能を遂行できない**,または**遂行不能**になること • 故障はある特定の瞬間に発生する
エラー (error)	• **設計上の欠陥**,あるいは**望ましい意図した状態からの逸脱**のこと • エラーは人間が介入を行い除去するまでその状態が続く • 抽象的概念,モデル,設計,ダイアグラム,プログラム等のような(状態はあるが)動作しないものはエラーはあるが故障はない
事故 (accident)	• 損失を引き起こし,望ましくない,かつ**計画外の事象**(必ずしも不測の事象ではない)のこと ➢ 事故は,生命,財産,あるいは環境に何らかの**損害あるいは損失を招く** ➢ 事故は事象のタイプに制限を付けずに**損失の事象**として定義する • 事故は望まれないので「計画外」であり「意図的」でもない ➢ 計画可能な事は「防止策」と「改善策」である ➢ システム開発では事前に**「何が事故」**を明確に定義する必要がある
安全 (safety)	• 事故や損失がないこと • 安全はコンポーネントが同時に作動中に,システムレベルで起こる**『創発特性』**である

STAMPの用語③

HASHIMOTO SOFTWARE CONSULTING INTERNATIONAL Inc.

用語	定義と説明
安全制約 (Safety Constraint)	• システムが安全に保たれるために必要なルールのこと
ニアミス(near miss) ／ インシデン(incident)	• **損失は伴わない**が(あるいは軽微な損失のみ),異なる状況の下では損失の可能性がある**事象**のこと • 被害が発生しない場合は**インシデント**(事故ではない)である
ハザード (hazard)	• システムの環境(または物体)において,他の条件(または物体)と共に,事故が発生する可能性のある**システムのある状態** • 事故の防止には事故の前兆を把握し,前兆をシステム設計者の制御下に置く要件が必要なので**システム安全設計**では**ハザード**という**概念**を用いる ➢ 事故はコンポーネントやシステムが存在する**環境**に関係がある ➢ ハザードはシステムまたはコンポーネントの**環境**に対して定義する ➢ ハザードの構成要素となるものは,**システムの境界をどこにするかに依存**する

STAMPの用語④

HASHIMOTO
SOFTWARE
CONSULTING
INTERNATIONAL Inc.

用語	定義と説明
ソフトウェアシステム安全	• ソフトウェアがハザードの原因にならずにシステム中で実行されること • 逆にソフトウエアがシステム安全に影響を与えるのは下記のとき: ➢ システムがハザードになることに影響を与える出力値やタイミングで作動する ➢ ハードウェア故障の認識や制御に失敗する
セーフティクリティカルなソフトウェア	• 直接または間接的にシステムのハザードな状態につながるあらゆるソフトウェアのこと
セーフティクリティカルな機能	• 正しい動作,誤った動作(誤ったタイミングでの正しい動作も含む)および動作不能によりシステムハザードになるシステム機能のこと
セーフティクリティカルなソフトウェア機能	• 他のシステムコンポーネントの振る舞いや環境条件などと共に,直接的または間接的にハザードをもたらすシステムの状態にするようなソフトウェア機能のこと

第1部のまとめ

HASHIMOTO
SOFTWARE
CONSULTING
INTERNATIONAL Inc.

第1部のWrap Up
• STAMPは**事故モデル**である • 解析技法／分析技法は**事故モデルに依存**する • STAMPは**システムズ理論**に基づいている • 複雑なシステムの**安全性**と**セキュリティ**は**『創発特性』**である • **安全性**を『不具合問題』としてではなく**『制御問題』**として定義する • **古い技法の『還元主義』』**では**複雑なシステムの対処に限界**にきている • STAMPはシステムの構成要素や部分を個別に扱わず**システム全体**を扱う • STAMPの用語の意味と使い方に注意する

《第2部》
STPAの分析プロセス

STPAのハザード分析

HASHIMOTO
SOFTWARE
CONSULTING
INTERNATIONAL Inc.

STPAのハザード分析の特徴

- トップダウンのハザード分析／安全性解析技法
  - 影響を解析し,早い段階での設計の決定に寄与
  - システム開発の初期からハザード分析／安全性解析を実施
  - システムの設計と並行で実施し,設計が進むにつれ解析／分析作業を累進的に反復して精緻化していく
- 各種制約／条件を特定する
  - アクシデント
  - ハザード
  - 安全制約／安全要件
  - Unsafe Control Action(UCA)
- 安全性志向のシステム開発を支援する

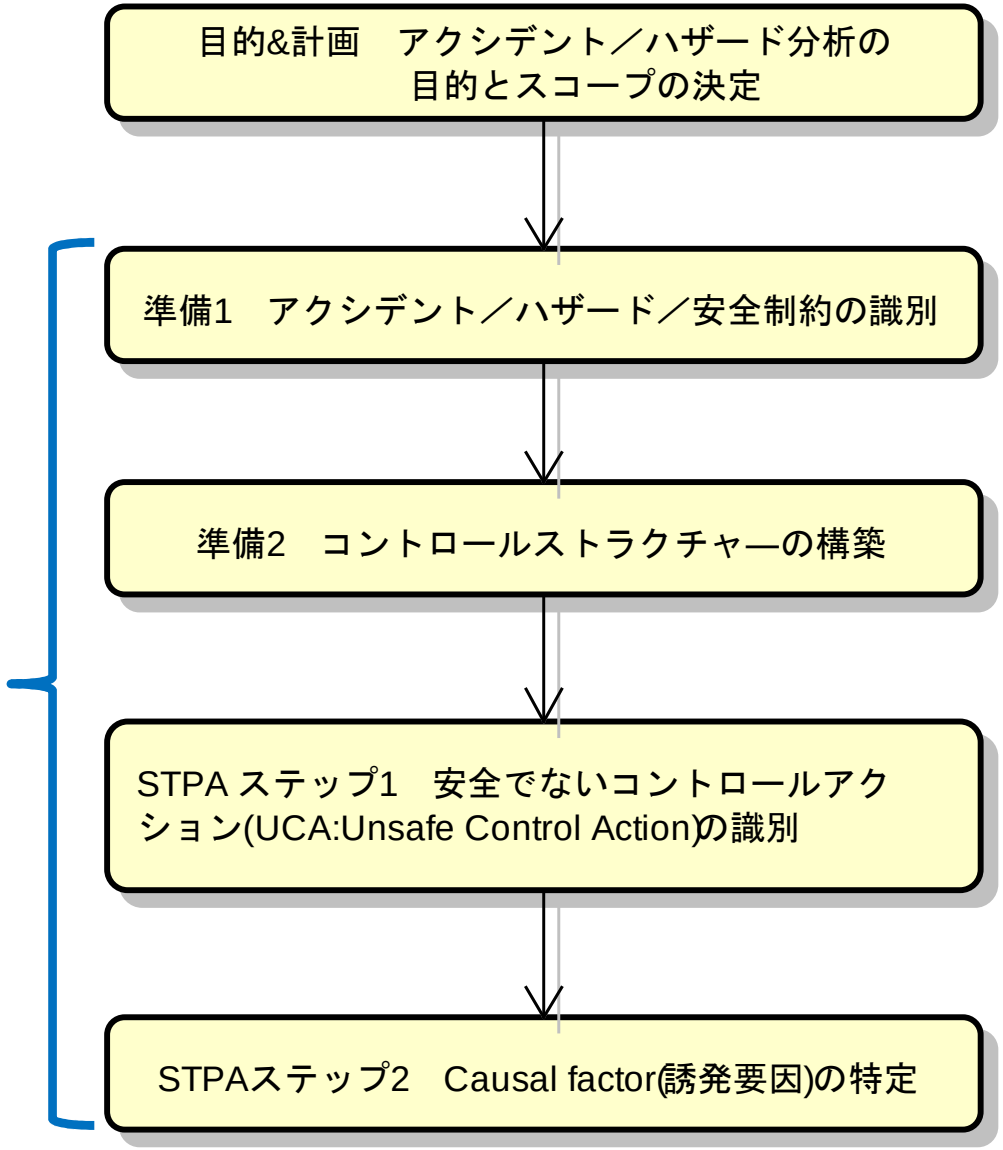
STPA

STPAの作業プロセス

HASHIMOTO SOFTWARE CONSULTING INTERNATIONAL Inc.

- 安全解析作業を行う上で重要

- 基本的にトップダウンに作業は進められるが,必要に応じて作業を反復的に繰り返しながら進める



目的＆計画

HASHIMOTO
SOFTWARE
CONSULTING
INTERNATIONAL Inc.

目的&計画　アクシデント／ハザード分析の
目的とスコープの決定

決定事項	説明
目的	• STPAで分析を行う目的
対象範囲(スコープ)	• どこを対象範囲とするか
計画	• スケジュール／担当者／使用ツール
アクシデント として扱う対象	• アクシデントとして扱う対象を具体的に決定する ➢ 人命／怪我／器物破損／環境破壊／ミッションの未達成／経済損失／ブランドイメージ失墜

準備1:アクシデント／ハザード／安全制約の識別

HASHIMOTO SOFTWARE CONSULTING INTERNATIONAL Inc.

準備1　アクシデント／ハザード／安全制約の識別

アクシデント／ハザード／安全制約の識別の進め方
• 初期のアクシデント／ハザード／安全制約／(安全要求)の識別を行う 　➢ 最初は思いつくアクシデント／ハザード／安全制約の識別ができれば良い 　➢ 後の作業でアクシデント／ハザード／安全制約／(安全要求)の追加／推敲する • アクシデント／ハザード／安全制約／(安全要求)の間の関係を明確にする 　➢ 表形式で整理すると良い 　➢ この表も作業を進めるに従い推敲されていく

アクシデント	ハザード	安全制約	(安全要求)

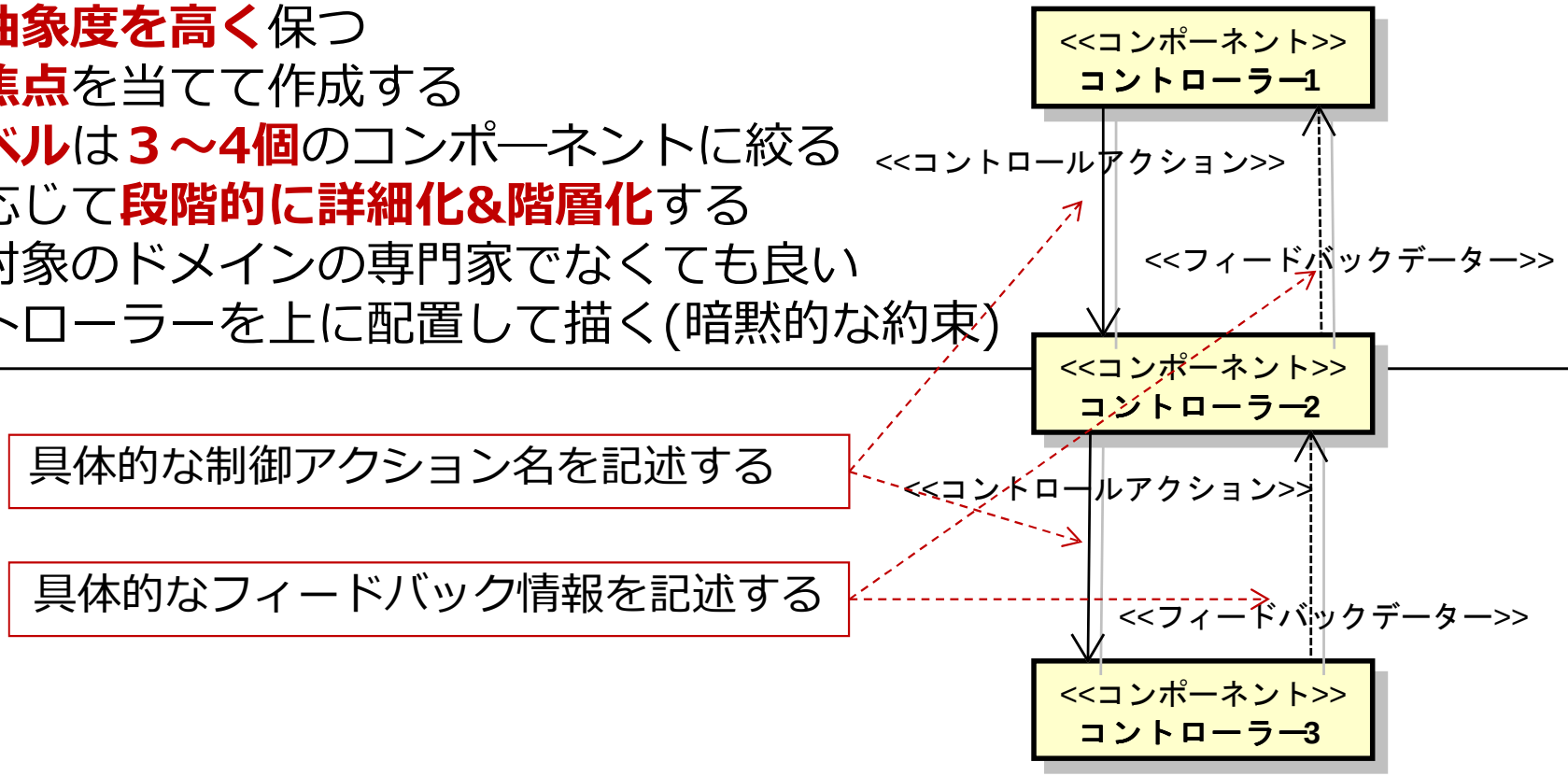
準備2:
コントロールストラクチャの構築

HASHIMOTO SOFTWARE CONSULTING INTERNATIONAL Inc.

準備2　コントロールストラクチャーの構築

コントロールストラクチャの作成の注意点

- コンポーネントになるもの：
  - 人／装置／組織／ソフトウエア／サブシステム／環境
- 可能な限り**抽象度を高く**保つ
- **安全制約に焦点**を当てて作成する
- **システムレベル**は**３～4個**のコンポーネントに絞る
  - 必要に応じて**段階的に詳細化&階層化**する
- 作成は分析対象のドメインの専門家でなくても良い
- 上位のコントローラーを上に配置して描く(暗黙的な約束)

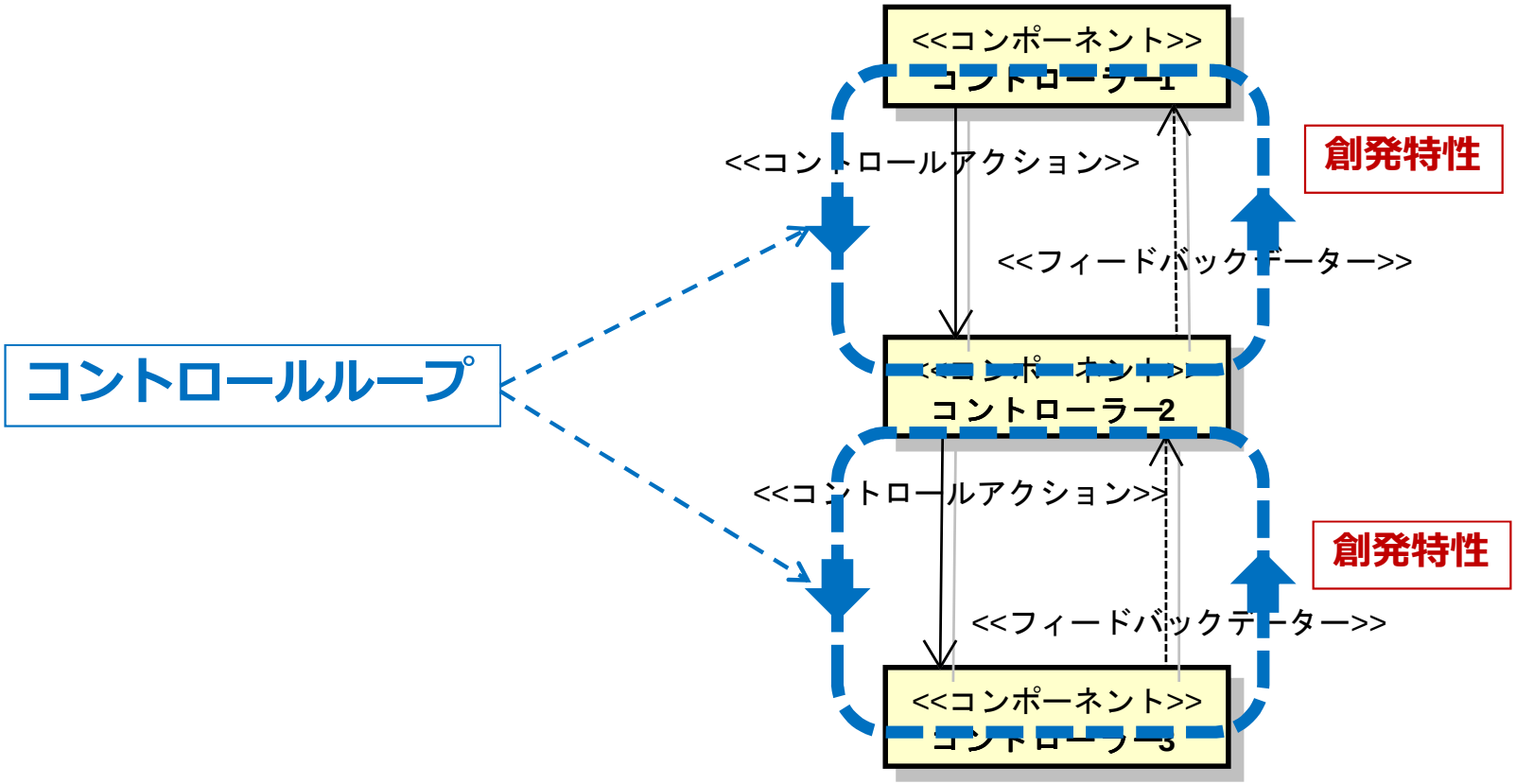


準備2:
コントロールストラクチャの構築

HASHIMOTO
SOFTWARE
CONSULTING
INTERNATIONAL Inc.

コントロールストラクチャの作成の注意点(つづき)

- **コントロールプロセス**にフォーカスして分析する
  - **メカニズム**にフォーカスしない
- コンポーネント間の相互作用である**コントロールループ**に注目する



UCA(Unsafe Control Action)の識別

HASHIMOTO SOFTWARE CONSULTING INTERNATIONAL Inc.

STPA ステップ1　安全でないコントロールアクション(UCA:Unsafe Control Action)の識別

UCA(Unsafe Control Action)

- **UCA(アンセーフ・コントロール・アクション)**はハザードにつながる(恐れのある)コントロールアクションのこと
- １つのハザードに至る複数のUCAがあり得る

UCA(Unsafe Control Action)の識別と4つのガイドワード

HASHIMOTO SOFTWARE CONSULTING INTERNATIONAL Inc.

STPA ステップ1　安全でないコントロールアクション(UCA:Unsafe Control Action)の識別

『4つのガイドワード』を使いUCAを識別する

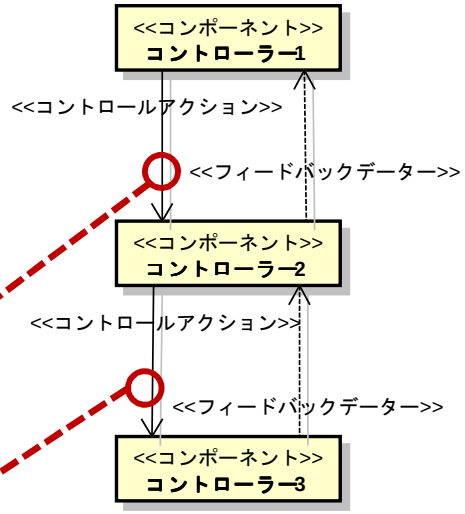
Control Actions	Not Providing Causes Hazard (与えられないとハザード)	Providing Causes Hazard (与えられるとハザード)	Incorrect Timing/Incorrect Order (早すぎ／遅すぎ／間違った順序でハザード)	Stopped Too Soon/Applied Too Long (早すぎる停止／長すぎる適用でハザード)
CA1:xxxxxxxxx				
CA2:xxxxxxxxx				
CA3:xxxxxxxxx				
CA4:xxxxxxxxx				

※STPAの事例報告では,この表の記載方法に幾つかのバリエーションがある

UCAとコントロールアクション

HASHIMOTO SOFTWARE CONSULTING INTERNATIONAL Inc.

STPA ステップ1　安全でないコントロールアクション(UCA:Unsafe Control Action)の識別



コントロールアクションには
ユニークなIDを付与する

Control Actions	Not Providing Causes Hazard (与えられないとハザード)	Providing Causes Hazard (与えられるとハザード)	Incorrect Timing/Incorrect Order (早すぎ／遅すぎ／間違った順序でハザード)	Stopped Too Soon/Applied Too Long (早すぎる停止／長すぎる適用でハザード)
CA1:				
CA2:				
CA3:				
CA4:				

コントロールストラクチャで識別した『コントロールアクション』を書き写す

UCAの識別

HASHIMOTO
SOFTWARE
CONSULTING
INTERNATIONAL Inc.

STPA ステップ1　安全でないコントロールアクション(UCA:Unsafe Control Action)の識別

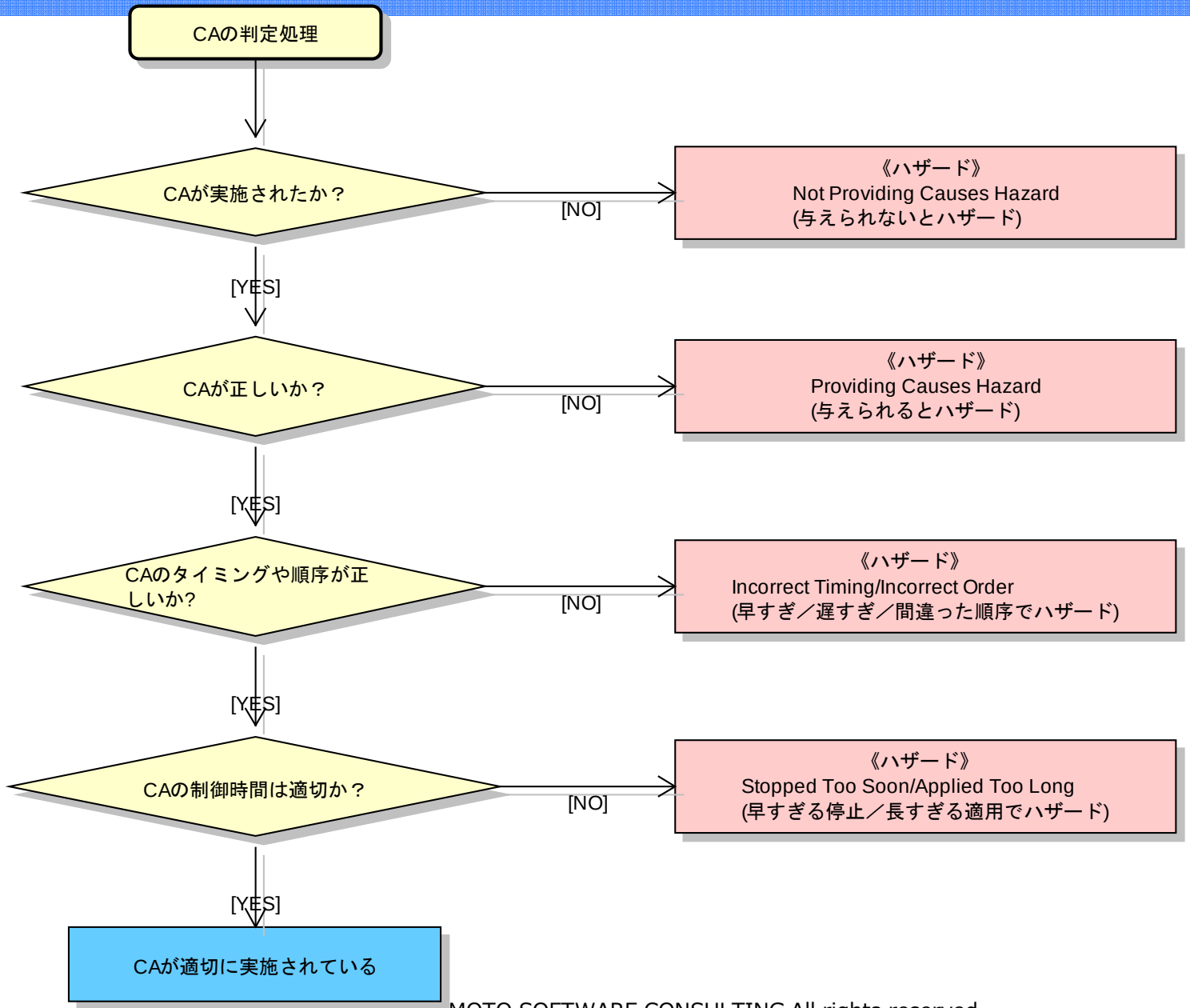
この順番で考えていく事が大切

Control Actions	Not Providing Causes Hazard (与えられないとハザード)	Providing Causes Hazard (与えられるとハザード)	Incorrect Timing/Incorrect Order (早すぎ／遅すぎ／間違った順序でハザード)	Stopped Too Soon/Applied Too Long (早すぎる停止／長すぎる適用でハザード)
CA1:				
CA2:				
CA3:				
CA4:				

- CA毎に４つのガイドワードを使い「**どのような状況でCAが実施(実施されない)されるか**」について記述する
- 記述した状況が「**UCAか?／UCAではないか?**」を検討する
- UCAと判断したらユニークなUCAのIDを付与する

『４つのガイドワード』を使った思考の流れ

HASHIMOTO SOFTWARE CONSULTING INTERNATIONAL Inc.



UCAの識別

HASHIMOTO SOFTWARE CONSULTING INTERNATIONAL Inc.

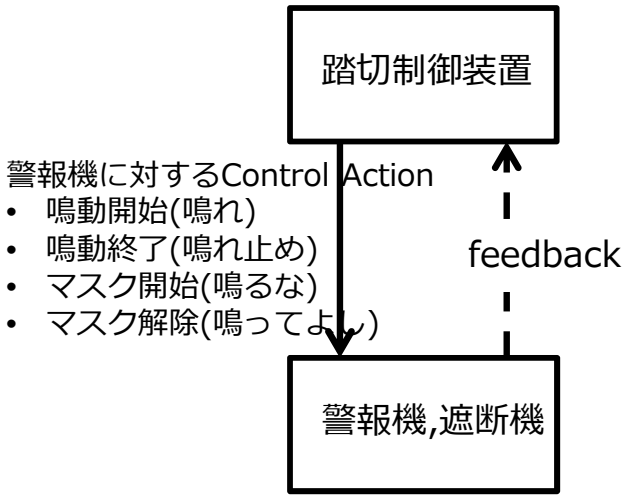
Control Actions	Not Providing Causes Hazard (与えられないとハザード)	Providing Causes Hazard (与えられるとハザード)	Incorrect Timing/Incorrect Order (早すぎ／遅すぎ／間違った順序でハザード)	Stopped Too Soon/Applied Too Long (早すぎる停止／長すぎる適用でハザード)
CA1: 鳴動終了(鳴り止め)	???	???	???	???
CA2:				
CA3:				
CA4:				

- 『４つのガイドワード』によるUCAの識別は難しい
  - 属人的な結果いなりやすい
  - UCAとなる状況の記述が不完全になりやすい

UCAの識別に,より効果的な体系的識別法を利用する(次のスライド)

UCAの体系的識別法

HASHIMOTO
SOFTWARE
CONSULTING
INTERNATIONAL Inc.



「はじめてのSTAMP・STPA」図 4.3-2より

UCAの体系的な識別法

- コントローラーとコントロールドプロセスとその間の各コントロールアクションに注目して**《コンテキスト》《タイプ》**の**起こり得る組み合わせ**を考えて文章にする
- **コンテキスト変数**と**コンテキスト値**を利用してUCAを精微化する
- コンテキスト変数と値は：
  - **「時間」「量」「距離」「割合」「数」etc**
- **コンテキストテーブル**の利用

《ソース・コントローラー》	《コンテキスト》	《コントロールアクション》	《タイプ》	《UCAか？》
踏切制御装置が	電車が踏切を10m通過後に	(警報機に)警報終了	指示する	OK
踏切制御装置が	電車が踏切を通過中に	(警報機に)警報終了	指示する	
踏切制御装置が	電車が踏切を10m通過後に	(警報機に)警報終了	指示しない	OK
踏切制御装置が	電車が踏切を通過中に	(警報機に)警報開始	指示する	?
踏切制御装置が	電車が踏切内に停車中に	(警報機に)警報終了	指示する	

UCAの体系的識別法

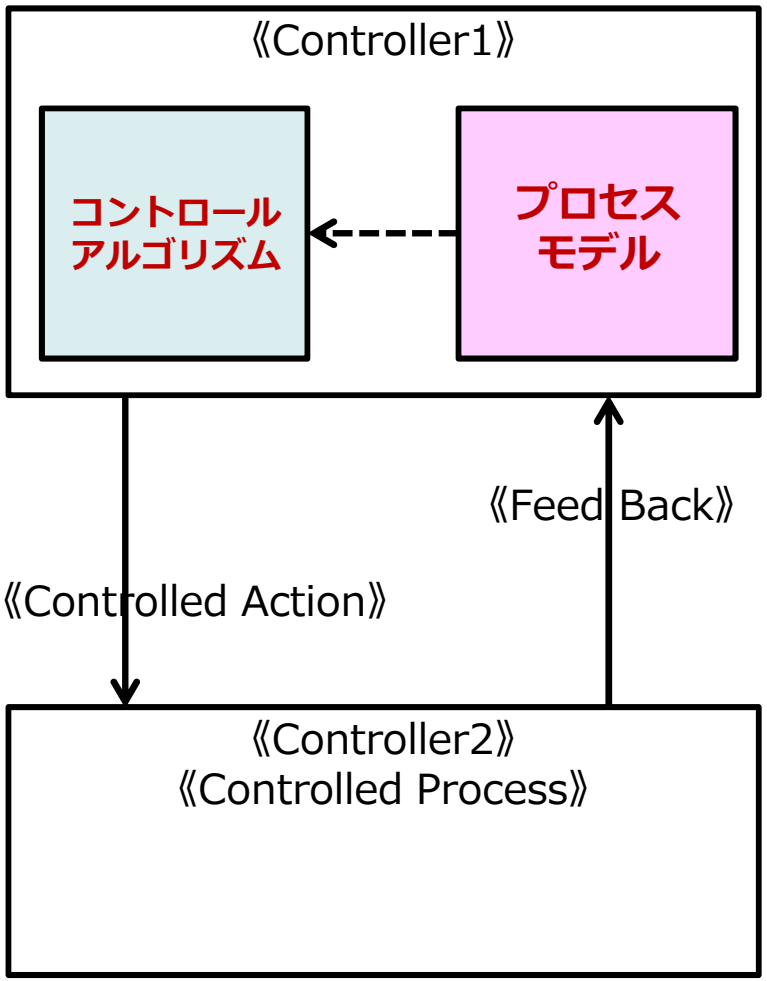
HASHIMOTO
SOFTWARE
CONSULTING
INTERNATIONAL Inc.

Control Actions	Not Providing Causes Hazard (与えられないとハザード)	Providing Causes Hazard (与えられるとハザード)	Incorrect Timing/Incorrect Order (早すぎ／遅すぎ／間違った順序でハザード)	Stopped Too Soon/Applied Too Long (早すぎる停止／長すぎる適用でハザード)
CA1: 鳴動終了(鳴り止め)	• 踏切制御装置が電車が踏切を10m通過後に(警報機に)鳴動終了指示を出さない(正常な状態ではないがUCAでは無い)	• 踏切制御装置が電車が踏切を10m通過後に(警報機に)鳴動終了指示する • 踏切制御装置が電車が踏切を通過中に(警報機に)鳴動終了指示する(UCA1) • 踏切制御装置が電車が踏切内に停車中に(警報機に)鳴動終了を指示する(UCA2)		
CA2: ・・・・・・・・	・・・・・・・・	・・・・・・・・	・・・・・・・・	・・・・・・・・

- UCAおよびUCAでない記述を明確かつ網羅的に記述することが可能
- 文章の構造を意識して記述する
  - **《ソース・コントローラー》**
  - **《コントロールアクション》**
  - **《コンテキスト》**
  - **《タイプ》**
- **コンテキスト変数**と**コンテキスト値**を利用して正確な記述ができる

コントローラーのプロセスモデルとコントロールアルゴリズム

HASHIMOTO
SOFTWARE
CONSULTING
INTERNATIONAL Inc.

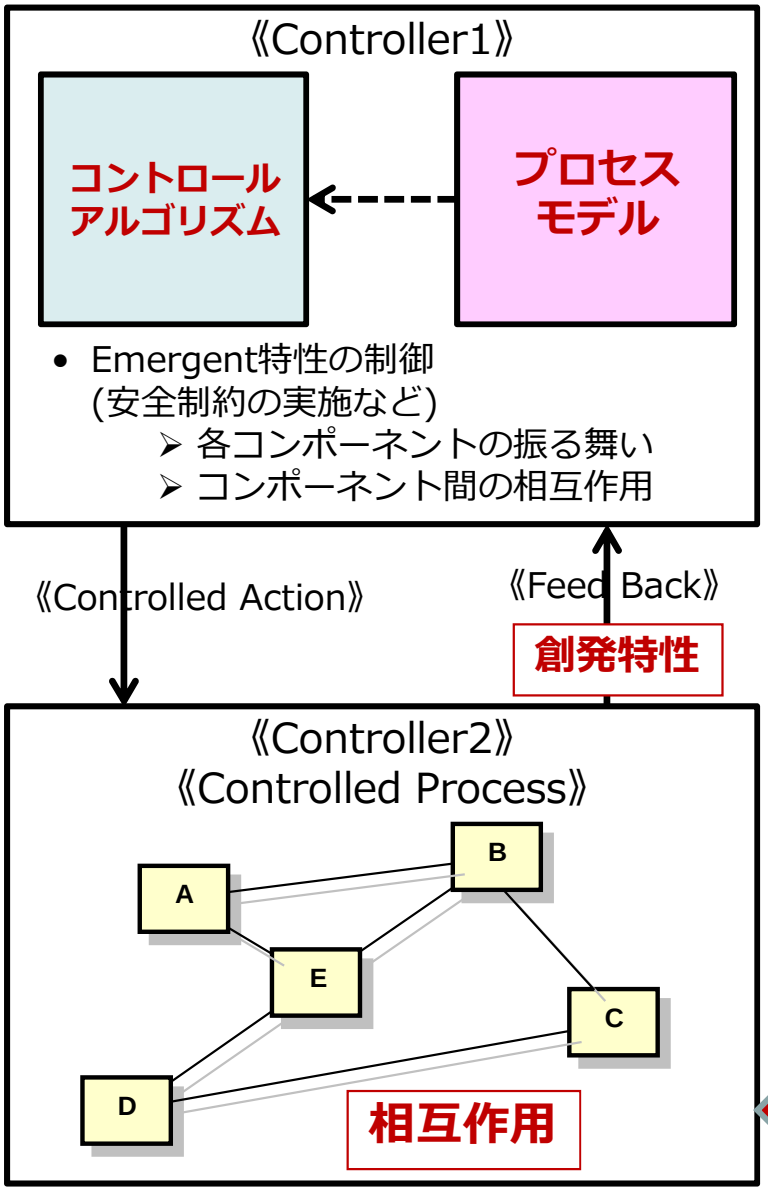


プロセスモデルとコントロールアルゴリズム

- コントローラーはコントロールアクションを決定する**プロセスモデル**を持っている
- **プロセスモデルに誤り(4つの誤り型)**が存在すると**予想外の動作／ハザード／アクシデントが発生**する
  - **Not Providing Causes Hazard**
    (与えられないとハザード)
  - **Providing Causes Hazard**
    (与えられるとハザード)
  - **Incorrect Timing/Incorrect Order**
    (早すぎ／遅すぎ／間違った順序でハザード)
  - **Stopped Too Soon/Applied Too Long**
    (早すぎる停止／長すぎる適用でハザード)

コントローラーのプロセスモデルとコントロールアルゴリズム

HASHIMOTO SOFTWARE CONSULTING INTERNATIONAL Inc.



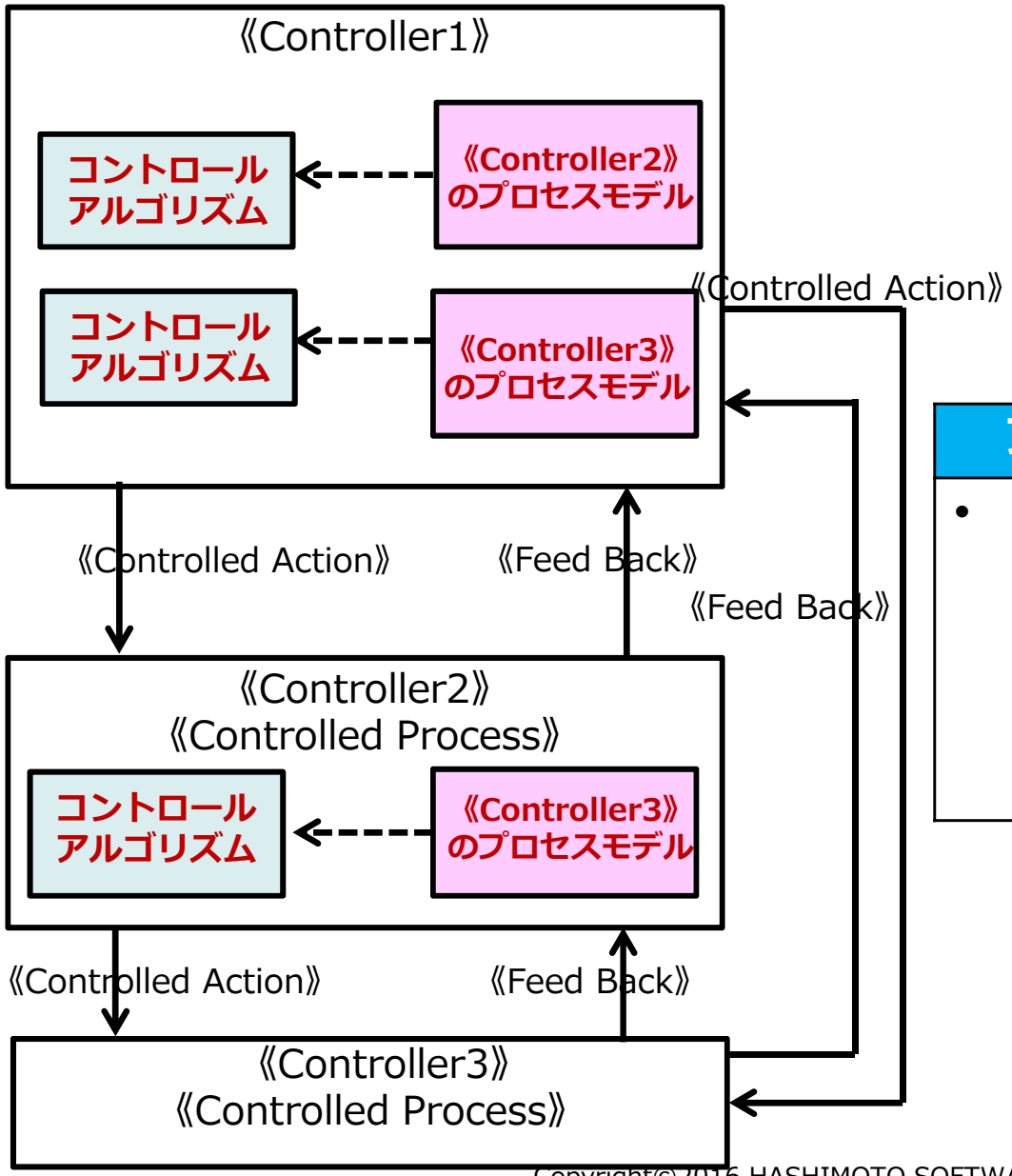
外部要因(環境)

Dynamic Control Problemの安全性

- **Safety as a Dynamic Control Problem**
  - ハザードはシステム設計やオペレーションでの**『安全性制約(safety constraints)』**に対する**不十分／不適切な対応**で発生する
- ゴールはコンポーネント及びシステム全体の**振る舞いを制御**し,システム内の操作で**安全性の制約が強制**すること
  - 不具合の防止や制御ではなく,システムの**振る舞いの安全性／セキュリティ制約**を実施する

コントローラーのプロセスモデルとコントロールアルゴリズム

HASHIMOTO SOFTWARE CONSULTING INTERNATIONAL Inc.



プロセスモデルとコントロールアルゴリズム

- コントローラーは複数のコントロールアクションを決定する**プロセスモデル**を持つ場合がある
  - 制御するコンポーネントの数だけプロセスモデルとコントロールアルゴリズムが存在する

UCAから安全制約を導く

HASHIMOTO
SOFTWARE
CONSULTING
INTERNATIONAL Inc.

識別したUCA	安全制約(Safety Constraint)
UCA1： 踏切制御装置が電車が踏切を通過中に(警報機に)鳴動終了の指示をする	どのコンポーネントにどのような対策が必要かを考えて安全制約の内容を更新／追加する
UCA2： 踏切制御装置が電車が踏切内に停車中に(警報機に)鳴動終了の指示をする	
・・・・・	・・・・・
・・・・・	・・・・・

Hazard Causal Factor識別

HASHIMOTO SOFTWARE CONSULTING INTERNATIONAL Inc.

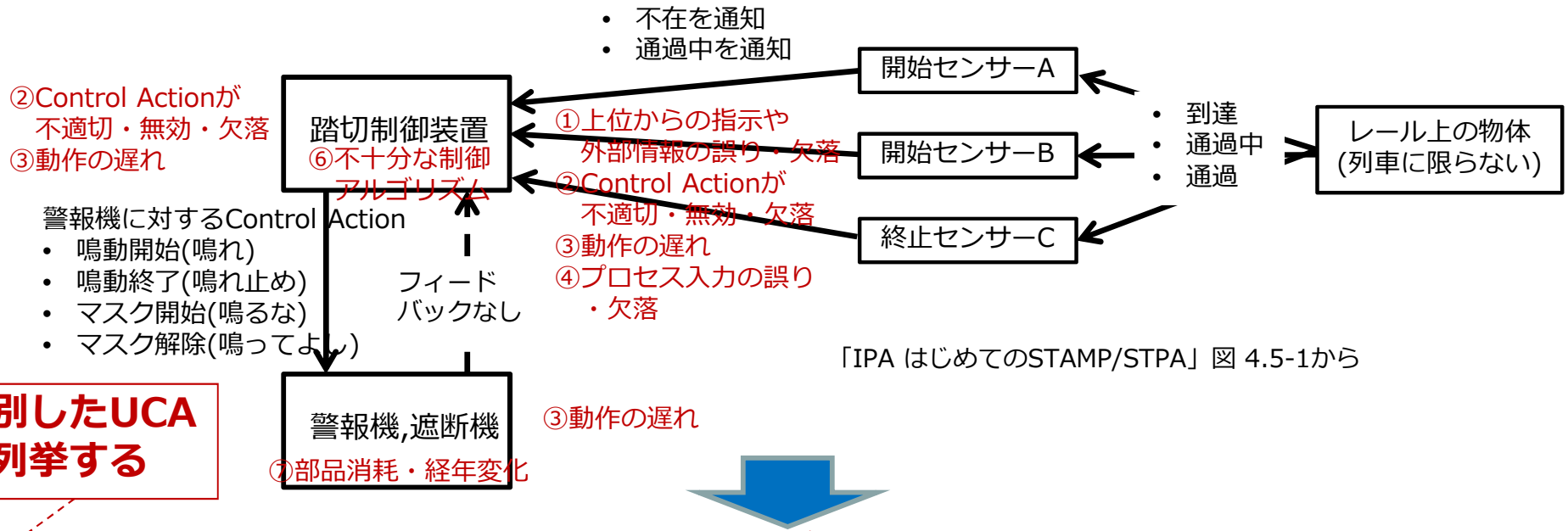
STPAステップ2　Causal factor(誘発要因)の特定

HCF(Hazard Causal Factor)

- **『ハザード誘発要因』**とも呼ばれる
- **HCF**は単に**CF**と書くこともある
- HCFはUCA(Unsafe Control Action)を誘発する原因
- UCAが発生する要因であるHCFを識別する
- HCFの識別には『11個のガイドワード』とコントロールストラクチャのシナリオを利用する

HCF識別表の利用

HASHIMOTO
SOFTWARE
CONSULTING
INTERNATIONAL Inc.



「IPA はじめてのSTAMP/STPA」図 4.5-1から

識別したUCAを列挙する

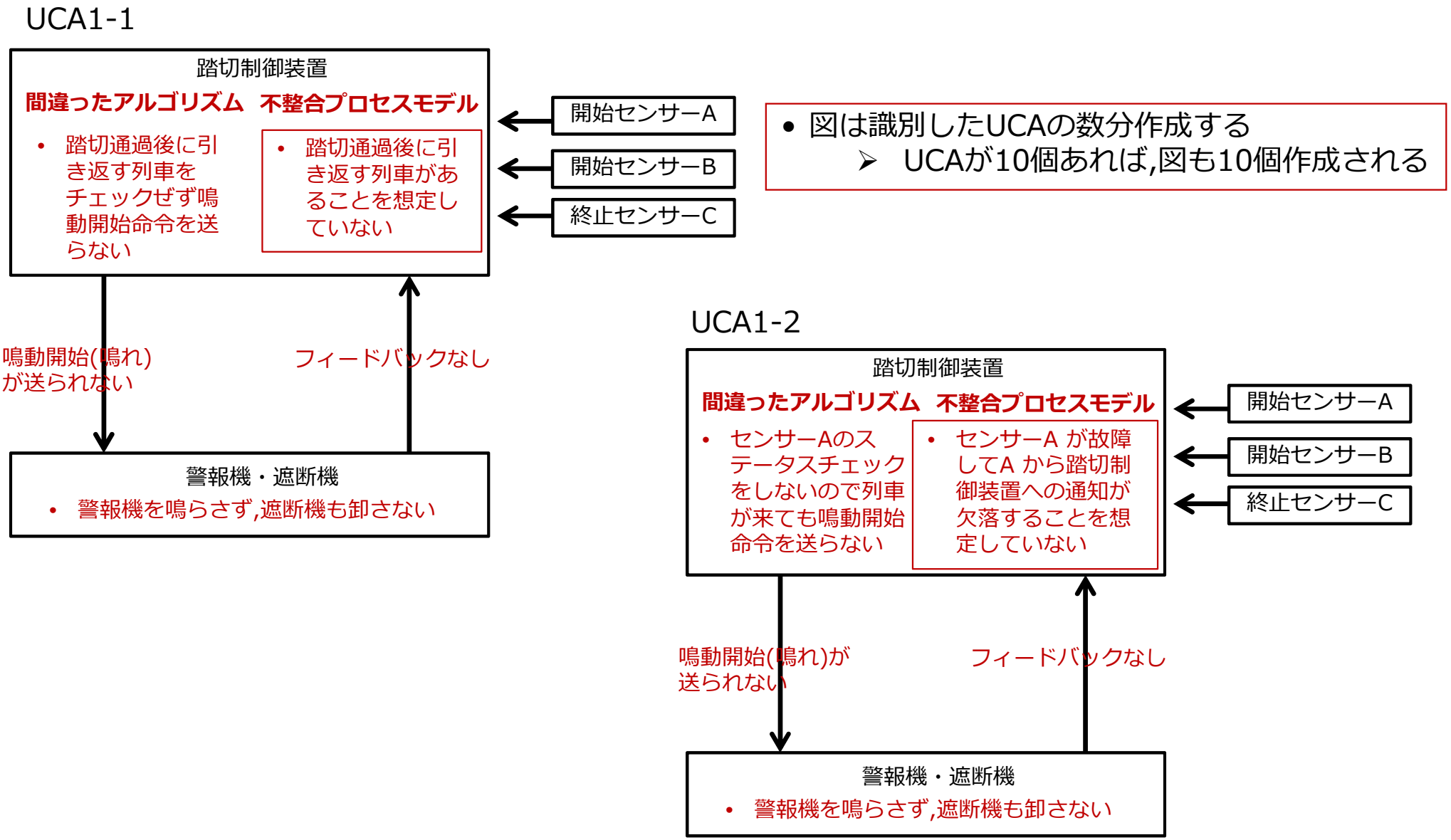
11個のガイドワードを記載する

UCA	①	②	③	④	・・・	・・・	・・・	・・・	・・・	⑩	⑪
UCA1-1											
UCA1-2											
UCA2-1											

- マス目に記載した内容がHCF
- 全てのマス目にHCFが記載されるとは限らない

各UCAに対してCSのシナリオ図を作成する

HASHIMOTO SOFTWARE CONSULTING INTERNATIONAL Inc.



第2部のまとめ

HASHIMOTO
SOFTWARE
CONSULTING
INTERNATIONAL Inc.

第2部のWrap Up

- STAMPおよびSTPAはFTAやHAZOPなどの信頼性工学に基づくハザード分析技法を**補完**する
  - 取って変わる技法ではない
  - 複数の技法を併用する必要がある
- **コントロールストラクチャの作成**が最初のハードル
  - コントロールストラクチャの作成は,ドメインエキスパートや開発者でなくても良い
  - **抽象度を高く**する
  - 必要に応じて段階的に詳細化／階層化する
- UCAを慎重に識別する必要がある
  - 4つのガイドワードの仕様
  - UCAの構造**「ソース・コントローラー」「コンテキスト」「タイプ」「コントロールアクション」**