セーフウェア

システム安全とコンピュータ

松原 友夫
松原コンサルティング

今や社会はシステムに依存している

- 今や社会はコンピュータで制御されたシステムに依存している
- システムの規模はコンピュータを利用するようになって急速に増大した
- 最初コンピュータの利用には事故を減らす狙いもあった
  - コンピュータは故障しない、という神話を信じて
- しかし、かえって重大な事故は増えている

システムの重大事故は増えている

- たしかに従来型のシステムでは対策が進んだ
  - 安全技術が確立し（例えば力学理論）
  - 安全技術を法律で規制するようになり（例えば、建築基準）
  - 社会がそれを安全技術を組み込むコストを容認するようになった
  - かくして、ボイラー、橋梁、建造物などの事故は著しく減った
- 技術が進んだのに事故は重大化している
  - 国境を越える規模
  - 自動化された機器の事故
  - 被害の規模が増大した
  - 事故のコスト、調査のコストが増大した

産業化社会における新たなリスク要因(1)

- 新たなハザードの登場
  - 過去に低減され除去されたハザードよりずっと影響範囲が大きく発見や除去が困難なものがある
  - まだ適切な対応策が確立していないものもある
  - 過去に学んだ教訓が生かせない
    - 例えばコンピュータの利用
- システムの複雑さの増大
  - コンポーネント事故からシステム事故へ
- 曝露の増大
  - より多くの人がハザードに曝されるようになった
    - 例えば、網（電力、通信、鉄道）の制御によって
- エネルギー量の増大
  - 高エネルギー源の発見と利用によって
    - 例えば原子力

産業化社会における新たなリスク要因(2)

- 増加する手動操作の自動化
  - オペレータエラーのリスクを減らそうとした結果、保守、設計、監視など、より高度な意思決定にリスクがシフト
  - オペレータに責任が押し付けられてきた
- 集中化と規模の増大
  - 自動化は規模の増大と集中化を促進した
    - 原子力発電所の規模は増大の一途をたどる
    - タンカーの巨大化は制御可能な規模を超えている
      - 例えば、船員が、海と戦う意識を養う環境が失われた
    - フィーリングで制御するのは困難になった
- 技術的変化の加速
  - 経験から学ぶ機会が減る

安全への取り組みの歴史

- 産業革命時代は事故のリスクはすべて労働者が負った
- 労働者の社会的運動が世論を喚起した
- 次第に法律により保護されるようになった
- 1914年に最初の安全規格が米国で発行された
- 1929年にハインリッヒ論文が書かれた
- ハインリッヒの仮説は労働者の不注意に注目を集めた
- ハインリッヒは最初にドミノ理論を提唱
- WWIIは訓練中の事故は戦闘時の２倍以上に及んだ
- WWII後、製品と開発プロセスに安全を組み込む安全プログラムが始まった
- 自動化が新たなハザードをもたらすようになった
  - ソフトウェアが安全に関与するようになった

事故から学ぶモデル

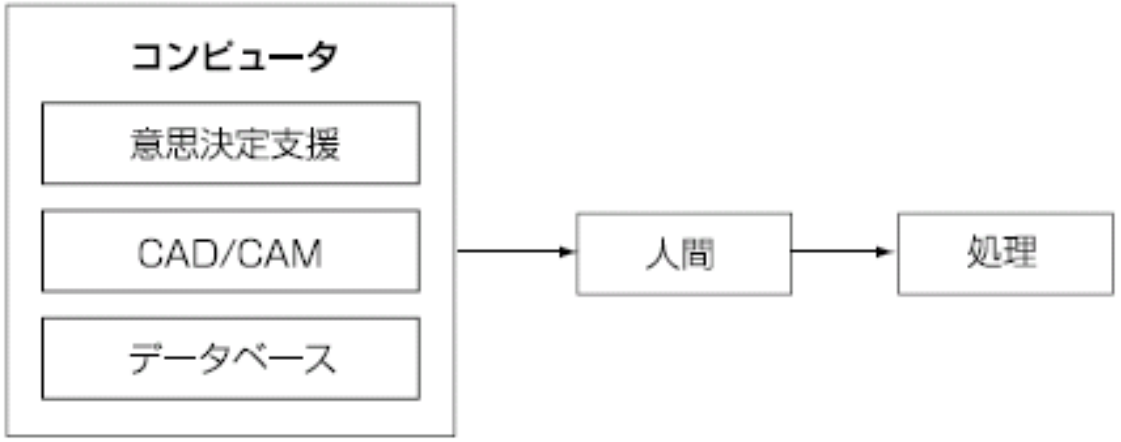
- 事故や事故に至らないニアミス（インシデント）は多くの教訓を含んでいる
- 従来の機械・電気的システムでは、事故から学び、それらを抑制する技術や手段を蓄積してきた
- それの適用が決して十分ではないうちに、情報技術のシステム制御への利用という新たな時代が到来した
- 大規模・複雑になったシステムでは、たった1度の事故が許容範囲を超えるようになった

いまや過去から学ぶだけでは十分でない

- 過去の事故から学ぶことが許されるか？
  - 特定の種類の事故が過去に起きなかったからといって、その種の事故が将来も起きないという保証はない
  - 事故から学ばねばならないのなら、1 つの事故が、受け入れ難いほどの悲劇的結果を招きかねないシステムに対して、われわれには何ができるのか
  - 起きてしまってからでは遅い
- リスク低減のためのシステム安全アプローチ
  - 事後の事故調査に頼るのではなく、事前のハザード分析で事故とその原因を予測することと、システムの寿命全体を通してできる限りハザードを除去又は制御することである

システムにおけるコンピュータの役割

- クリティカルシステムにおけるコンピュータの利用段階
  - オペレータに情報やアドバイスを提供する
  - データを解釈し制御上の決定を行う人にそれを表示する
  - 直接コマンドを出すが人間が動作を監視し入力を提供する
  - 制御ループから完全に人間を排除する
- 間接的制御もリスクを伴う



ソフトウェアが制御するシステムの事故 (1)

- 事故調査はより困難になりコストが増加する
  - 故障している電子部品が残骸の中から見つかることはない
  - 微妙なエラーは簡単には見つからない
    - 1つのソフトウェアエラーでF-14を失ったとき、それを調査するのに数百万ドルかかった
    - 私も銀行システムで類似の経験をした
- 新たなシステムでは過去に蓄積した技法が使えなくなった
  - 実績のあるエンジニアリング技法はソフトウェアを考慮していないし、それをソフトウェアに適合させるのは容易でない
  - ソフトウェア技術者と在来の技術者間のコミュニケーションもうまくいっていない

ソフトウェアが制御するシステムの事故 (2)

- 常に開発が先行し安全確保のための理論や手法は後になって確立するのが常である
  - 例えば、1883年にニューヨークのイーストリバーに架けられた鉄鋼製吊り橋、ブルックリン橋
- 自動制御への過度の依存という新種の事故が増えている
  - e.g. つい操縦桿を離してしまう
- 在来型のシステムに比べて事故調査が表面的になっている
  - 特に日本では調査結果が公表されないことが多い

システム設計者がおかす共通の誤り

- システム設計者の多くはソフトウェアの専門家ではない
  - ソフトウェアの理解が足りないシステム設計者の思いこみがシステム事故を招いている
- ソフトウェアをよく知らないシステム設計者がおかす共通の誤りをまとめたものが以下の「7つのソフトウェアの神話」である
- これらの思いこみが不適切な設計をもたらし事故を招く

ソフトウェアの神話 (1)

- 神話1：コンピュータのコストはアナログ機器や電気機械装置より安い
  - 表面的には正しいが、信頼性が高いソフトウェアを開発・保守するコストは膨大になることがある
- 神話2：ソフトウェアは簡単に変更出来る
  - 表面的には正しいが、エラーを組み込まずに変更するのは難しい
- 神話3：コンピュータは置き換えた電気機械装置よりも高い信頼性を提供してくれる
  - 「故障」はしないがバグのないソフトウェアを作るのは難しい
  - 長期間安定稼働していてもバグがないとは言えない

ソフトウェアの神話 (2)

- 神話4：ソフトウェアの信頼性が高まれば安全性も高まる
  - 仕様通りできていても安全ではない
  - コンポーネントが完全でも事故は起こる（事例は後述）
  - 安全性はシステムの特性であってコンポーネントの特性ではない
    - 欠陥のないブレーキが安全とは限らない
- 神話5：ソフトウェアを試験すること、または正しいことを証明することで、すべてのエラーが除去出来る
  - 安全性に関するソフトウェアエラーのほとんどはコーディングエラーではなく要件のエラーである

ソフトウェアの神話 (3)

- 神話6：ソフトウェアを再利用すれば安全性は高まる（次ページに実例）
  - 考慮されていない新たなシステム固有のハザードが事故を起こす
  - かえって安全性が低下することさえある
    - ソフトウェアの再利用が重大事故を招いた事例に事欠かない
- 神話7：コンピュータは機械式システムよりリスクが少ない
  - たしかにコンピュータは安全性を高める潜在能力があるが、現状ではそれを十分活かすに至っていない

神話6:再利用が安全性を損なった例

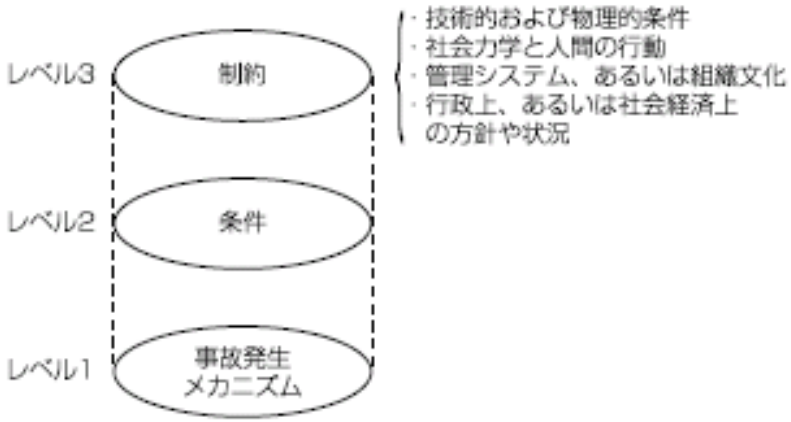
- Therac-20 の一部はTherac-25 に再利用されたが、Therac-25 によって少なくとも2 人が死亡した原因と同じエラーがTherac-20 にもあった。そのエラーは、Therac-20 においては時折ヒューズが飛んでしまうだけで、過剰照射という重大な結果を引き起こすことはなく、そのためにエラーが検出されることも修正されることもなかった。
- 米国の航空管制で長年にわたって順調に利用されていたソフトウェアが英国で再利用されたが、うまくはいかなかった。米国人開発者は経度0 度の扱いを考慮していなかったのである。その結果、そのソフトウェアはグリニッジ子午線に沿って英国を折りたたみ、ノリッチとバーミンガムが重なるようなことになった。
- 北半球向けに作成された航空用ソフトウェアは、南半球で使用された時に問題が起きることがよくある。さらに、米国のF-16 型機用のソフトウェアがイスラエル機で再利用され、高度が海抜高度よりも低い死海上空を飛行していた時に事故が起こっている。
- Ariane-5の打ち上げ事故では、Ariane-4から再利用されたソフトウェアの不要になった機能が災いしてブースターを止めたのが原因の一つ。

ソフトウェアを工学的に扱うのは難しい

- 複雑である
  - 「残念ながら、人間は複雑さにほとんど対処できない」 -- G.F. McCormick
- 離散状態システムである
  - 連続関数の扱いと比べて十分理解されているとは言えない
- コンポーネント間のインターフェースが見えにくい
- 柔軟性の呪い
  - 「柔軟性は後知恵の墓場」-- John Shore
  - 課せられるべき自然法則や物理的制約がないので、甚だしく複雑なシステムの構築を極めて容易にする
    - 「仕様はすぐに欲しいものリストになってしまう」-- .G.F. McCormick

因果関係の3階層モデル

- 事故原因の3階層モデル (Peter Lewyckyの提案)
  - レベル1は事故発生のメカニズム
    - e, g 運転手がブレーキを踏んだ、車が横滑りした
  - レベル2は事象が発生し得る条件または条件の欠如
    - e. g. 速度が速すぎる、路面が濡れている
  - レベル3は事故全般に影響を及ぼす根本原因
    - 識別し除去しなければ事故を繰り返す



事故の根本原因 (1)

- 安全文化の欠陥
  - 自信過剰と自己満足
    - スリーマイル、チェノブイリ、チャレンジャー、ボパールなど
  - リスクの過小評価
    - タイタニック号は、複数の事象の確率を掛け合わせた極端に低い確率で、世界で最も安全な船と自慢していたが、実際には事象が独立でなかった。これを「タイタニックの偶然の一致」と言う
  - 冗長性に頼りすぎること
    - e. g. チャレンジャーの１次、２次Oリング
  - 非現実的なリスクアセスメント
    - Therac-25のアセスメントではソフトウェアに10^{-4}の確率値を割り当て計測できない因子は無視された
  - 低確率で苛酷な事象の無視
  - ソフトウェア関連リスクの過小評価
  - 早期の警報や兆候の無視

事故の根本原因 (2)

- 安全に低い優先順位を割り当てる
  - 安全関連予算の割り当てが経営者からの安全へのメッセージ
  - チャレンジャー事故調査報告は、NASAの責任部署の20人のうち1人の25%だけが安全関連の仕事をしていたことを指摘した
  - JR西日本では安全よりも業績と体面を優先していたようだ
- 相反する目標への誤った解決
  - ほとんどの事故の背景には目先のビジネス目標の優先がある
- 効果的でない組織構造
  - 責任と権限の分散
    - 例えばチャレンジャー事故のOリングについての責任
  - 独立性の欠如と安全担当部門の低い位置
  - 限定された情報伝達経路と貧困な情報の流れ

事故の根本原因 (3)

- 効果的でない技術活動
  - 表面的な安全活動
    - 形式的な記録
    - 指摘してもフォローアップされない
    - 官僚的な業務
  - 有効でないリスク制御
    - 事故原因のほとんどは、事故を防ぐための知識がなかったからではなく、知識の使い方を知らないために起こった
    - ソフトウェアを安易に追加または変更することで複雑さを加えリスクを増やすこともある
  - 変更に対する評価の失敗
  - 情報の不足
    - 類似の悲惨な事故の防止に役立つ情報は、容易に入手できない
- こうした組織文化は容易には変わらない
  - 変える努力が外から見えない限り恐らくリスクは残る

原因識別の落とし穴

- 法的責任追及の観点から過度に単純化される
  - 最も関係のある要因が無視されるので再発防止には役立たない
- オペレータに責任を押し付ける傾向が強い
  - オペレータエラーの蔭には設計エラーや調査不十分が隠れている
  - 「オペレータエラー」からは建設的な改善策は生まれない
    - 鉄道の連結事故はヒューマンエラーとされていた間は改善されなかったが、政府の自動連結器導入の強制で、事故は大幅に減った
- 機械の故障や直近の事象のみを重視する傾向がある
  - これは最も重要な要因の軽視につながる
- 組織的要因を軽視する
  - 組織的要因は大規模システム事故のほとんどで指摘されているが、組織内調査では無視または軽視される
- 原因識別能力の低さ(とくにソフトウェア)も問題がある

ヒューマンエラーの位置づけ

- 大部分の事故の原因はオペレータにあるという一般的な仮説は正しくない
  - ハインリッヒは88%がオペレータにあると主張した
  - しかし、統計データは額面通りには受け取れない
    - データには偏りがある
    - プラスの行動は記録されない
    - オペレータはすべての緊急事態を克服出来るという前提に基づいて非難される
    - オペレータはしばしば極限状態で介入せねばならない
    - 後知恵なら何とでも言える
    - オペレータエラーと設計エラーを分離するのは不可能に近い
- 「ヒューマン(オペレータ)エラー」はシステムの改善には役に立たない用語であり「人間とタスクのミスマッチ」という言葉に置き換えるべきだ -- Rasmussen

ヒューマンエラーの意味

- ヒューマンエラーという用語は一般に次の2 つの意味で使われる。
  - 人間が悪いことと知っていたはずの何かをした
  - 人間が、その時点では悪いことだとは知らず、振り返って悪かったことが分かる何かをした
- だが、この区別は、事故を減らすことより、非難や罪を着せようとする場合に限り意味がある
- Trevor Kletz
  - 「ほとんどすべての事故はヒューマンエラーのせいにすることができるが、そうすることは事故の予防に役立たない」
  - 「この言葉は、今後の事故をどうやって防ぐかについての建設的な考えを妨げる」

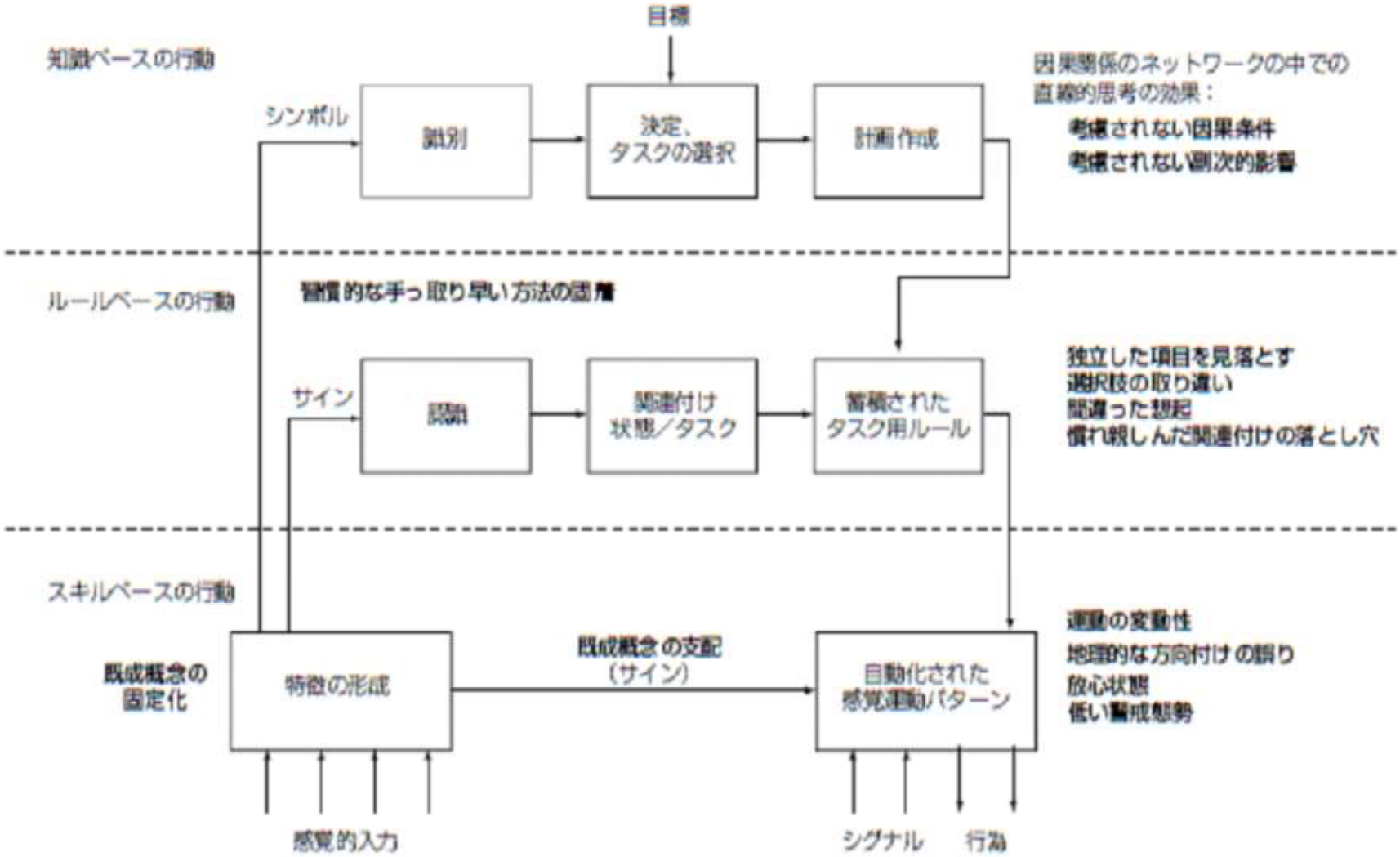
予想しない事態を克服するのは人の能力

- 人間が誤りをおかすという特性と同じ人間特性が、人間の計画プロセスの中では肯定的特徴として表現されることがある
  - 例えば、過去の規則性をモデル化してそれらを未来に適用するという人間の驚くべき能力、と言い直すことができる。人間は、この能力によって確実に未来を予測することができる
  - ある状況でヒューマンエラーをおかすものは、別の状況では人間の創意工夫の才であるかもしれない
- ヒューマンエラーは人間の適応能力と創造性の不可避的な副作用なのであり、それは技術的構成要素の故障とは全く異なる

人間の行動には3つのレベルがある

- Rasmussenの3レベル認知制御モデル（次ページに図）
  - スキルベースの行動
    - 行動はフィードフォワード制御に基づく
  - ルールベースの行動
    - よく分かっている問題の意識的解決
  - 知識ベースの行動
    - 制御者が制御のためのノウハウや以前の経験から制御に役立つ制御のためのルールが存在しない環境に直面した時、行動の制御は目標制御的で知識に基づく概念的レベルの高みに移行していく。この状況において、計画は、異なる計画が検討され、目標に対して効果が試験された上での選択によって、物理的な試行錯誤によって、または、環境の機能的特性の概念的理解と検討中の計画の効果予測によって、策定される。

Rasmussenの認知制御モデル



認知制御モデルの意味

- スキルベースの行動
  - 組織文化
  - スポーツ選手の行動
- ルールベースの行動
  - マニュアル文化
  - 応用動作を制約する
  - 規格に依存する
  - いわゆるマニュアル人間を生む
- スキルベースの行動
  - 前例のない事態への対処には必須

安全アプローチの重点の違い

- 欧州は概して安全規格に力を入れる傾向がある
  - 多くの国に跨るシステムが多いためか
  - 鉄道関連規格が充実している
  - 信頼性工学による安全アプローチ
- Nancy(≒米国)は安全規格の準拠や認証では安全を十分確保できないという立場
  - システム全体に対する安全アプローチ
- 日本では信頼性の延長線上に安全を置く
  - ハードウェア集約型システムへのアプローチが支配的
  - 唯一評価が高いのは新幹線

システムについての議論には注意が必要

- 同床異夢：人によってイメージが異なる
  - 会社の製品、専門分野、主観などによって
  - 国や地域でも異なる
- 我田引水
  - 自分の専門を中心にイメージする
  - 機械屋のシステム、電気屋のシステム、生態系システム、ソフトウェアシステムなど
- 開発中心
  - ほとんどの人が環境や社会の中で稼働する生きたシステムのことを考えない
- 範囲が異なる
  - 同じシステムでも人によってシステムに含む範囲が異なる

システム理論とは (1)

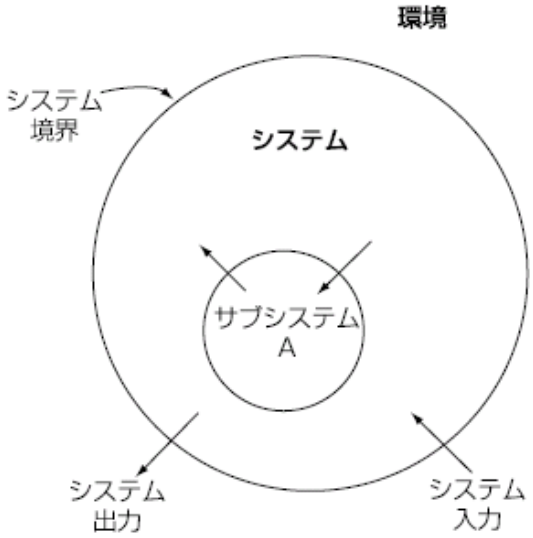
- なぜ生まれたか
  - システムの複雑性の増加には従来の科学では対処できない
  - システム理論は科学的還元主義の補完または反発として生まれた
- システムは3つのタイプがある
  - 体系的な単純さを提示するもの
    - システムは相互に作用し合わない部分に分解して分析できる
  - 体系的でない複雑さを提示するもの
    - 還元主義が有効な構造が欠けている
    - 集合として扱うことができ、挙動を統計的に扱えるほど規則的である
  - 体系的な複雑さを提示するもの
    - 完全に分析するには複雑すぎ、統計として扱うには体系的すぎる

システム理論とは (2)

- 主な適用領域
  - 第二次大戦後に設計された複雑なシステム
  - 生体システム、生態システム、社会システム
- 体系的複雑さを持つシステム
  - 還元主義では複雑過ぎて説明できない
- 創発特性(emergent property)
  - ある与えられたレベルの階層において、そのレベル固有の特性（創発特性）は下位レベルに還元できない
    - リンゴの形は下位レベル（細胞）での描写は意味をなさない。与えられた複雑性のレベルにおいて、そのレベル固有の特性は還元できない
  - 安全性はシステムの創発特性
- システム理論では個々の部分ではなくシステムを全体として扱う

システム理論とは (3)

- 用語定義:
  - システム:共通の目標または目的を達成するために、全体として一緒に行動する構成要素の集まり
  - システムの状態:その時点で関連する特性の集まり
  - システム環境:全体としての行動がシステム状態に影響を及ぼすような構成要素とそれらの性質の集まり
  - 入力/出力:システムと環境の境界を出入りするもの
    - Lehmanのプログラム進化論におけるE-Typeもシステムとして扱うことができる



計画されたシステムから生態的システムへ

- いままでのほとんどのシステムは意図され開発されたシステムだった
  - 開発の観点から安全を考える
  - 開発の規制に重点
  - 標準や規格で規制
- 社会にシステムが充満した現在、意図しないで異なるシステム同士が相互作用を起こすようになった
  - 想定外の事象を覚悟しなければならない
  - ハザードの分析とその緩和・回避が重点

システム安全の基本概念

- 完成した設計に安全を組み込むのではなく、設計に安全を組み込むことを重視する
- サブシステムや構成要素としてではなく、全体としてシステムを扱う
- ハザードを単なる故障より広い意味で捉える
  - 故障がないコンポーネントも安全解析の対象とする
- 過去の経験や規格よりも分析を重視する
- システム安全では、定量的手法よりも定性的手法を重視する
- システム安全では、システム設計におけるトレードオフと対立を重視する
- システム安全が扱う範囲はシステム工学より広い
  - 政治的社会的プロセス、管理者設計者オペレータの態度や動機、法律、許認可、世論など

ソフトウェアのシステム安全 (1)

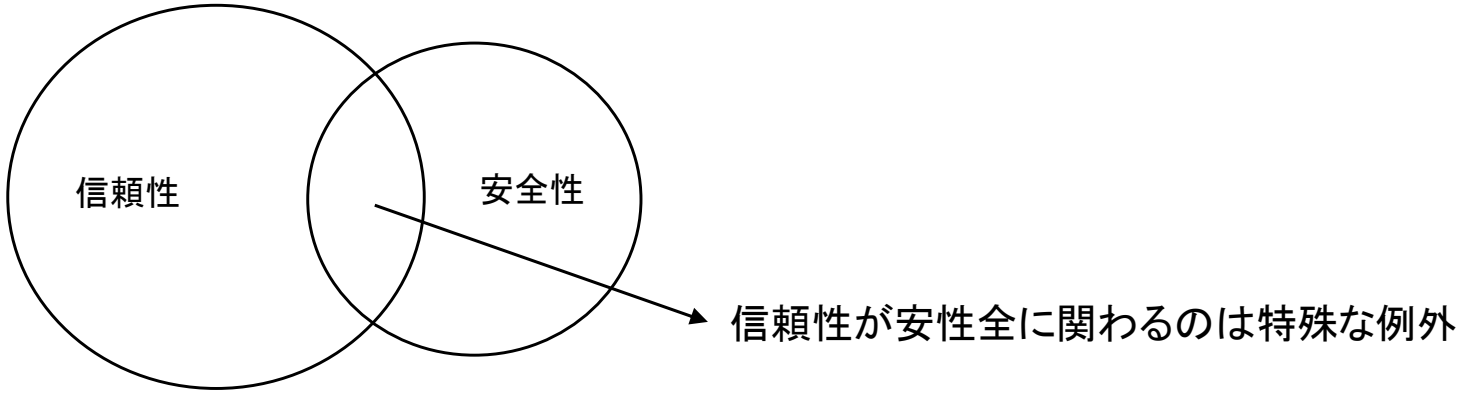
- ほとんどの事故はコンポーネント間のインターフェースと相互作用が原因にもかかわらず
  - つまり、ソフトウェアはシステム安全に直接的な役割を果たしているにもかかわらず
- システム安全技術者とソフトウェア技術者の間に深い溝やズレがある
  - システム安全技術者の一般的傾向
    - ソフトウェアを無視するかブラックボックスとして扱う
  - ソフトウェア技術者の一般的傾向
    - コンピュータを刺激⇔応答システムとして扱う
    - コンピュータの向こう側のことを思い描こうとしない
    - システムハザードへの影響を考慮に入れずに作る
  - 両者間のコミュニケーションは乏しい

ソフトウェアのシステム安全 (2)

- ソフトウェアの安全に関わる用語の定義
  - ソフトウェアシステム安全
    - ソフトウェアがハザードに寄与することなくシステム状況の中で実行されること
  - セーフティークリティカルなソフトウェア
    - 直接または間接にハザードをもたらすシステム状態につながるあらゆるソフトウェア
  - セーフティー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動作、または環境条件と共同して、直接または間接にハザードをもたらすシステム状態の存在につながるようなソフトウェア機能
- 安全問題の大部分は要件のエラー
  - 正しく要件を実装しているがそれがシステムの見地から安全でない動作を指定している
  - 要件がシステム安全に必要な特有の動作を指定していない
  - ソフトウェアが要件の指定を超えて意図されない危険な動作をする

信頼性と安全性は別のもの

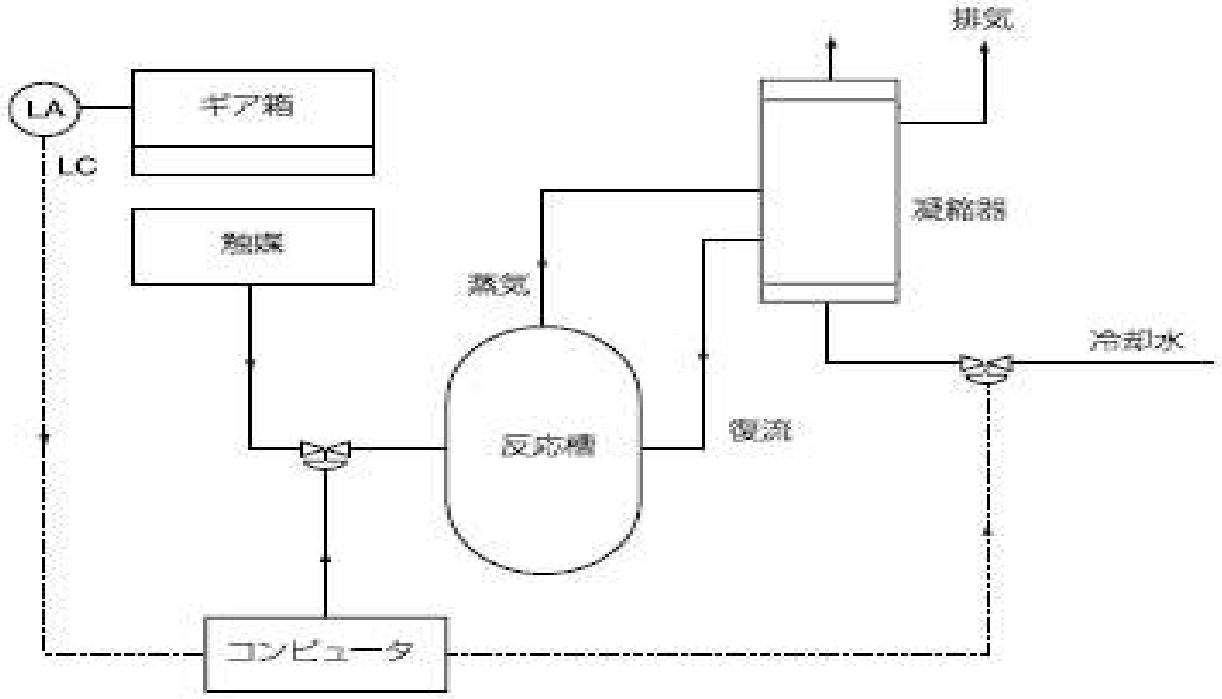
- しばしば信頼性と安全性は同じと仮定されるが、この仮定は特殊な例外でしか当てはまらない
- 信頼性の定義
  - あるコンポーネントが与えられた期間と指定された条件の下で求められた機能を実行する確率
- 事故はコンポーネントの故障の組み合わせによって起こるわけではない
  - コンポーネントに故障がなくても事故は起きる（次の頁に例）
- 信頼性が高くても安全でないことも、それと逆のこともある



コンポーネントに故障がなくても事故は起きる

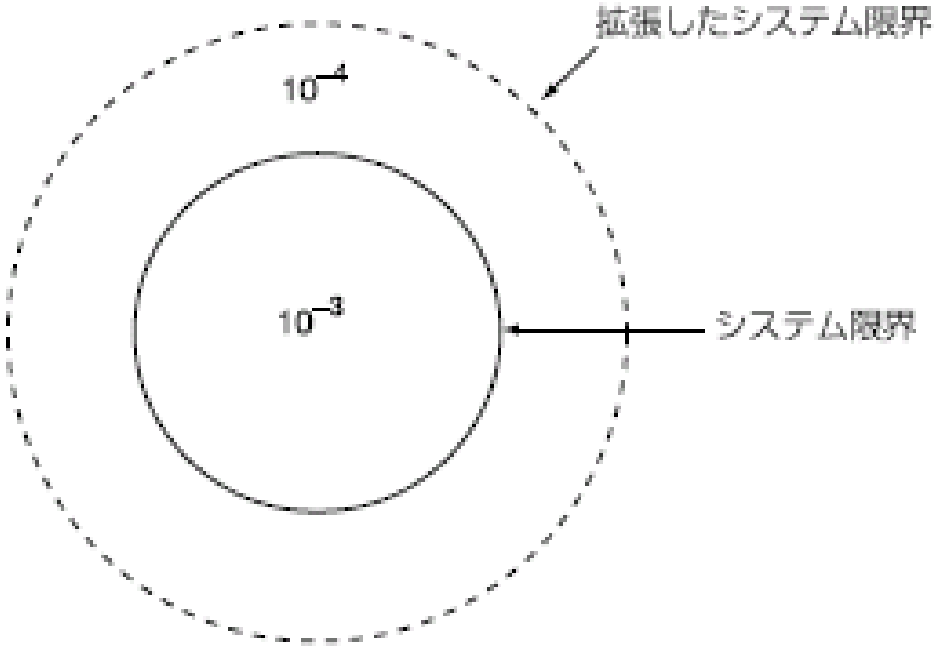
コンピュータは、反応槽への触媒の流入と、化学反応を冷やすための復流凝縮器への冷却水の流入を制御していた。さらに、コンピュータへのセンサー入力は、プラント内のどんな問題についても警告し、障害がプラント内で発生したら、すべての制御変数をそのままにして、警報を鳴らす仕様になっていた。

ある時コンピュータは、ギアボックスの潤滑油レベルが低くなっているという信号を受信した。コンピュータは、要件が規定した通りに反応した。コンピュータは警報を鳴らし、制御状態をそのまま固定した。偶然、触媒はちょうど反応槽に加えられており、コンピュータは復流凝縮器への冷却水の流入を増加させようとしていたところだった。そのため、流入率は低いまま保たれた。反応槽は過熱し、安全弁が開き、反応槽の内容物が大気に放出された。



信頼性評価には注意が必要 (1)

- 高信頼性を示す数値は安全を保証しないし、安全は高信頼性を必要としない
- 限定されたシステム境界（中央の円）内で冗長性なのでいくら信頼性を上げても、予想外の問題は防げない
  - 例えば、ヒューズの故障頻度は年間10^{-6}から10^{-7}だが、切れたときに面倒がって銅線で代用する確率は10^{-3}である



信頼性評価には注意が必要 (2)

- ばかげたリスクの見積もりの例
  - 最近まで、液化天然ガス研究における10^{-53}の事故確率というリスクアセスメントが、数字の記録としては最高記録だった。この記録は、核兵器の安全問題（通信回線上の特別な信号が、その意図なしに送られること）の確率が2.8×10^{-397}と計算されたため、破られた。墜落してくる飛行機に衝突される確率が10^{-7}から10^{-8}であるということを考えると、これらの計算値のばからしさは明らかだ。
- 多くの場合、事故を構成する複数の事象が独立だ、という仮定に誤りがある
- 安全では生起確率より影響を重視する
  - 10^{-7}~10^{-8}辺りから「稀に起きる」として同じ扱いにするのがよい

ハザード分析プロセス (1)

- 定量分析の落とし穴
  - 容易に、または合理的に定量化できる要因を重視し、設計エラー、建造上の欠陥、オペレータの誤操作、保守エラー、および管理上の欠陥など、予測や定量化が容易でない要因を無視または軽視する傾向があるが、これでは現実的なリスク見積はできない
    - 例えば、いくつかの確率論的アセスメントでは、装置の故障確率が 10^{-7} または 10^{-8} の範囲であることを重要視する一方で、同じ装置で 10^{-2} または 10^{-3} の範囲の確率で起こる据え付けミスや保守エラーを無視する
- むしろ動的なイメージを重視する
  - イメージする能力でハザード分析の深さは大きく変わる
  - かつて銀行システムを設計していたとき「みどりの窓口」システムの設計に関与した穂坂衛教授に、「大晦日に銀行に行き、いまシステムが止まったらなにが起こるかを想像せよ」という指導を受けた

ハザード分析プロセス (2)

- ハザードの識別
  - 基本的な手法
    - What-if分析
    - タイガーチーム攻撃
    - イメージを浮かべる
  - ハザードにレベルを付ける
    - 起こりやすさと過酷度によるマトリックス
    - これに基づいて優先順位を付ける
- ハザード分析能力の鍛え方
  - 実際に使われている現場、現物を見て考える習慣をつける
  - 原因分析のブレーンストーミング
  - 他の分野を含め広い範囲で好奇心を養う
  - 茎術など、趣味を持つのもいいかもしれない

安全のための設計

- ハザードを除去する設計
  - 置換
    - 可燃性物質から不燃性物質へ
    - 有毒物質から無毒物質へ
  - 単純化
    - 新たな技術と既存技術の新たな適用は、想像も予想もできない(unknown-unknown)事態をもたらすが、システムがより単純ならその機会は減る
    - Parnasは、ある原子力発電所の安全系のソフトウェアを6,000行のstraightforwardのコードで設計したが、英国の同等機能のソフトウェアは11万行であった
  - 分離(decoupling)
    - 延焼を抑える防火帯、高速道路の立体交差など
    - ソフトウェアでは、例えばモジュラー構造
  - 特定のヒューマンエラーの除去
    - ポインタ、制御移行、初期設定、グローバル変数などを避ける
  - ハザードをもたらす物質または条件の低減
    - 絶対に必要なコードだけを含める

ヒューマンマシンインタフェースの設計

- 一般的なプロセス
  - 使いやすさと安全性はしばしば両立しないので、一般的なHMI設計の指針をセーフティークリティカルシステムに適用する場合は要注意
  - クリティカルデータは多重入力を要求することになる
- 人間の特性への作業の適合
  - 「系統的なエラーと正しい動作はコインの裏表である」-- Reason
  - オペレータがエラーを犯したことを認識し、それを是正できるように設計する
- セーフティークリティカルなヒューマンエラーの低減
  - 安全を高める操作を簡単かつ確実にすることと、危険な行為を困難または不可能にすること
- 適切な情報とフィードバックの提供
- 安全なHMI設計のための指針
  - 人間の能力を置き換えるのではなく、それを高めるように設計する
  - オペレータを考慮することから設計プロセスを開始する
  - 設計の意思決定と安全解析にオペレータを参加させる

など、章末に全部で60項目の指針がある

システム安全への提言 ― 結びに代えて

- 開発だけを考えず稼働状況でのハザードを読むくせをつける
  - システムが稼働する現場に足を運んで観察する
- 原因をトレースできるように設計する
  - 意図的にトリガ事象を記録しないと消えてしまう
- 安全では計測可能な特性の「見える化」より「起こり得ることを見る」ことが重要
  - 安全技術者にとって「想定外」は恥
- システムを「プロダクト」ではなく行動する生き物として捕らえる
- 安全担当者の重要な特性は異分野への好奇心