21世紀のディペンダブルシステム

2008年11月21日

所　眞理雄
JST/CREST DEOS プロジェクト研究総括
（株）ソニーコンピュータサイエンス研究所

これまでの経緯

- 2007年12月にようやくDEOSプロジェクトにおけるディペンダビリティーの概念の方向が見えてきた
- 2007年12月12日に
  JST/CREST DEOS Project Symposium 2007において

ディペンダビリティーの基本概念
Fundamental Concept of Dependability

と題して以下の講演を行った

背景

情報システムの現状

現在、情報システムは多数のサーバーや組込みシステム、端末などがネットワークによって接続された複合システムとして稼動している。近年ではそれらの情報システム同士が相互に接続され、利用者により便利な利用形態を提供するようになってきている。このような情報システムは現在、必要不可欠な社会基盤となっており、その信頼性、安全性、使いやすさなどが情報システムの性能の必須の要件になっている。それにもかかわらず、これらの要求に対する対応は十分とはいえない状況である。

OSの役割

これらの要件はコンピュータや通信システムのハードウェアからアプリケーションに亘る全てのレイヤーに関わるものであり、それらの総合的な性能として評価されるものである。ソフトウェアの最下層を受け持つOSにおいては、上位レイヤーでの対応に対して基本的なツールを与える事が要求される。従って、そのための機能を十分に備えていることが必須である。これは組込みOSについても同様である。

DEOSプロジェクトの基本指針

我々は情報システムにおける信頼性、安全性、使いやすさなどの要件を「ディペンダビリティー」と言う観点から捉えなおし、本プロジェクトでは実利用を目指した組込みシステムのためのディペンダブルOS（開発環境などの関連基盤技術を含む）を開発することとする。

我々は組込みシステムを狭義には捉えず、ネットワークシステムに接続され、現在あるいは将来社会基盤システムの一員として使われるシステムを対象に含める。

ところで、ディペンダビリティーとは何？

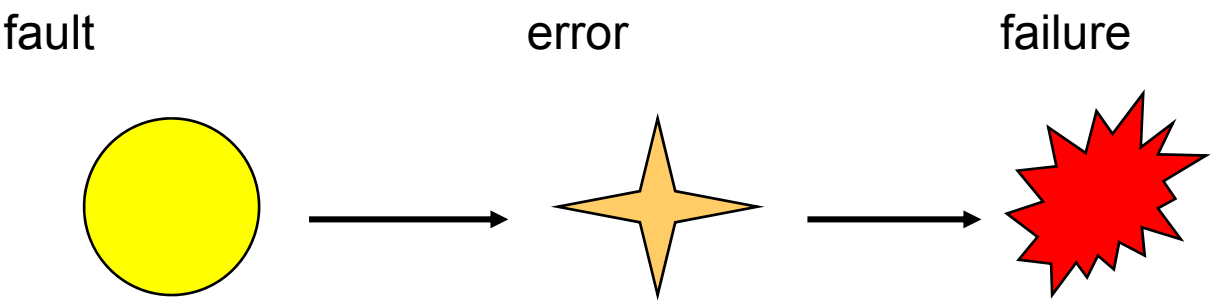
リライアビリティーやセキュリティーと何処が違うのか？

安全性と信頼性
Security & Reliability

- 安全性、信頼性とは、我々の日常生活、財産、生命を確保できるということ

- これらを脅かす脅威には以下がある
  - 物理的破壊
  - 経年変化
  - 悪意による攻撃
  - 設計・製造・運用時のエラー
  - 意図しなかった利用法
  - （人間の誤りやすさと貪欲さ)

信頼性に対する古典的アプローチ



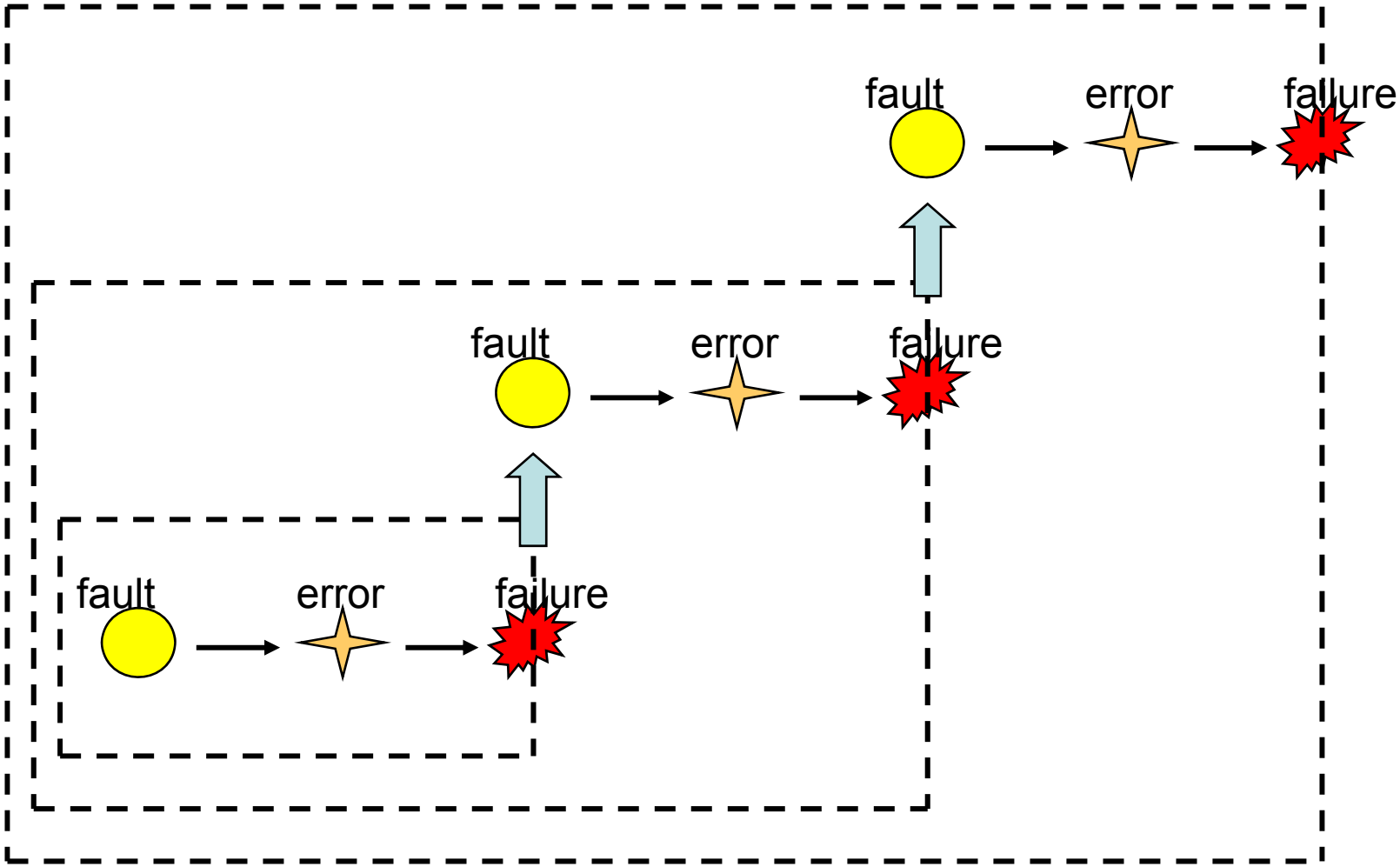
Fault Prevention, Detection, Tolerance, Removal, Forecasting, etc, by Redundancy

- in devices (dual, TMR,…)
- In location (distributed systems)
- In time (FEC, retry, reboot, …)

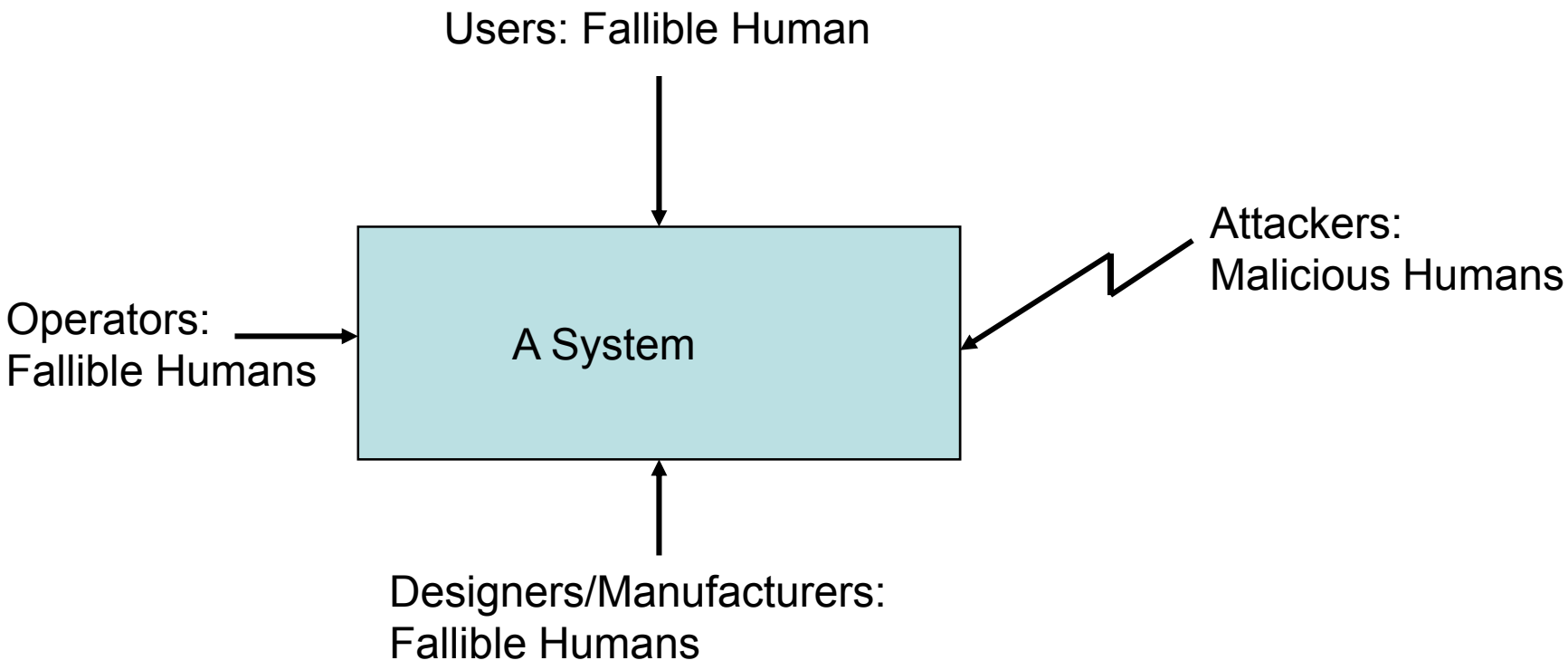
信頼性からRAS/RASISへ

- 背景はコンピュータの通信回線への接続とオンラインサービスの普及
- RAS(1960年代～）
  - 信頼性　Reliability　故障しにくいこと
  - 可用性　Availability　高い稼働率を維持できること
  - 保守性　Serviceability　障害が発生した場合に迅速に復旧できること
- RASIS（1970年代～）
  - RASに加えて保全性　Integrity　データが矛盾を起こさずに一貫性を保っていること
  - 安全性　Security　機密性が高く、不正アクセスがされにくいこと

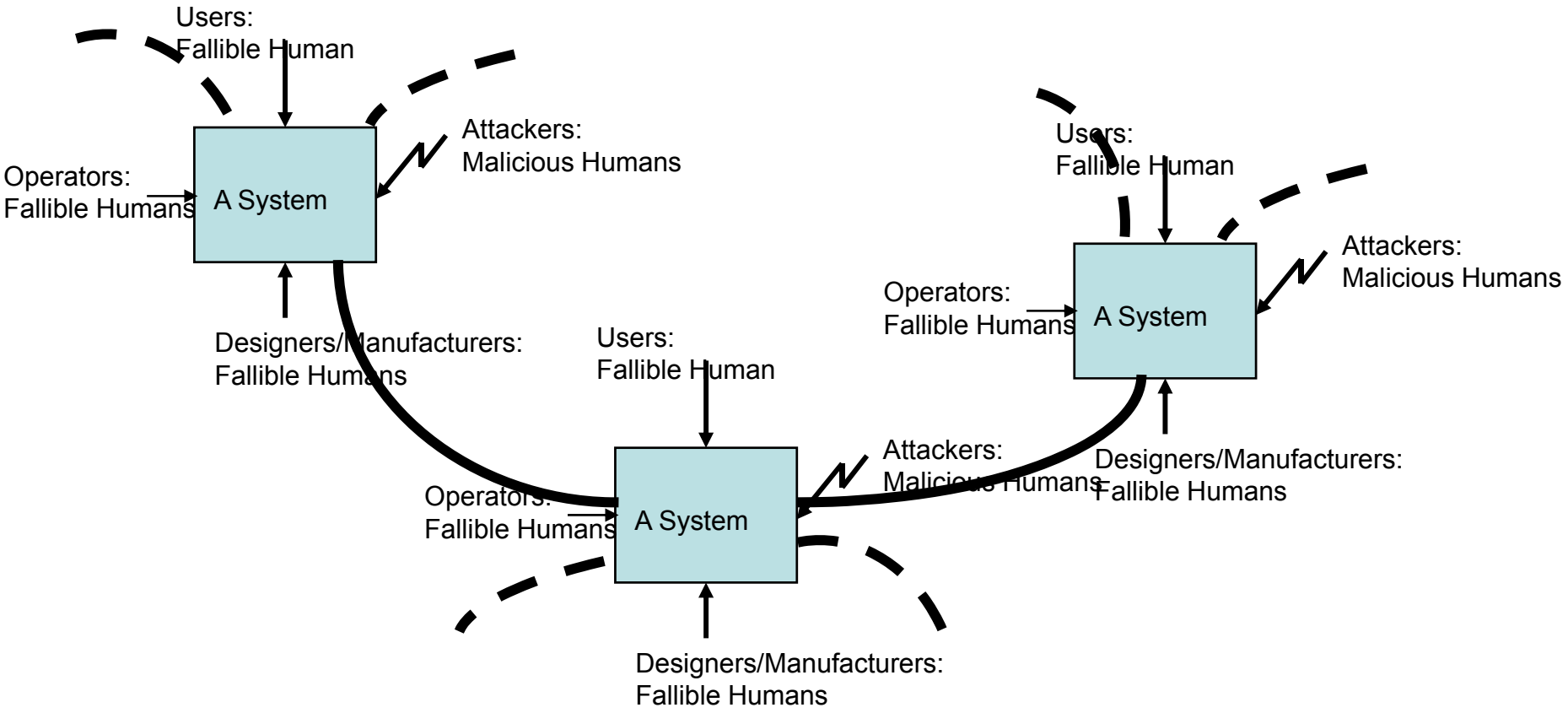
信頼性に対する階層的な見方



人間もシステムの一環として考える



ネットワークが普遍的に利用されるようになる



自律型計算（Autonomic Computing）

- 2000年～
- ネットワークで結合された複雑なシステムを想定し、以下の機能を備えた自律神経系を模したシステム構成により出来る限りインラインループでディペンダビリティーを確保しようとする考え方
  - Self Configuring
  - Self Optimizing
  - Self Healing
  - Self Protecting
  - By MAPE (Monitor, Analyze, Plan, and Execute) Loop

機能安全の考え方とSIL

- 国際安全規格ISO13849-1(EN954-1) や電気安全規格IEC60204-1は単純な部品や機器などに関するもので、ソフトウエアを含むシステムに対応していなかった。
- コンピュータ（ソフトウエアを含む）システムの安全規格の必要性から2000年に機能安全規格IEC61508が制定された。
- IEC61508では機器の故障をランダムハードウエア故障と系統的故障に分ける。
  - ランダムハードウエア故障は部品の劣化による故障→故障確率
  - 系統的故障はシステムの設計、製造、運用に起因する故障→安全ライフサイクルに基づいた手順と文書化 + Vモデルによるソフトウエア検証
- Safety Integrity Level (SIL): 低需要モードと高需要モードの2種類の運転モードにおける目標故障限度によりSIL4からSIL1として規定
- IEC61508をもとに、機械類関連のIEC62061、プロセス関連IEC61511、原子力関連IEC61513、鉄道関連61788、自動車関連ISO26262などが規定されている。

ディペンダブルシステムの構築の方法への疑問

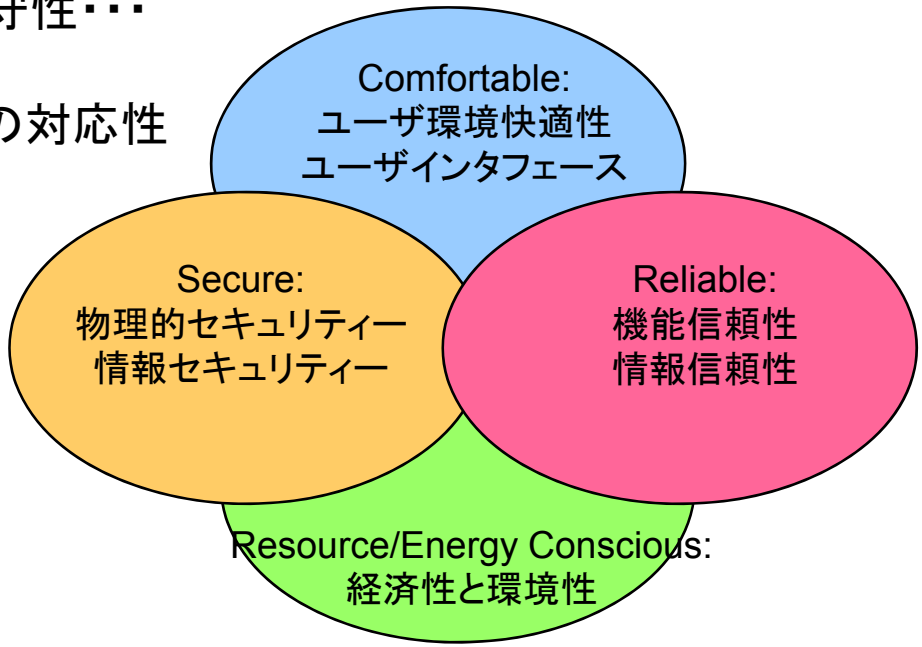
- よりSecureなシステムを作る？
- よりReliableなシステムを作る？
- そもそも、本当にディペンダブルなシステムは実現できるのか？
  - ありそうなこと、起こりそうなことを事前に想定してそれに対する対応策をシステム内に設計時に実装しておくことが出来るのか？
  - どれだけやったらディペンダブルなシステムになったと言えるのか？
- これまでの延長上にDependabilityに対する解はあるのか？　我々が求めているDependabilityは何なのか？

発想の転換の必要性

多分我々がDependabilityと言う言葉で表現したいことは
次のようなことなのではないか？

ディペンダビィリティーの定義

- システムが利用者に対し、利用者が期待する便益を出来る限り安全に且つ継続的に与えることが出来る性質
  - 安全に = 安全性 + 信頼性
  - 継続的に = 信頼性 + 可用性 + 保守性・・・
  - 出来る限り= リスクマネージメント
  - 利用者が期待する便益=ニーズへの対応性 + 使いやすさ + 少ない環境負荷



対象システム

- レイヤー化された複雑なシステム
- 人も含めたシステム
- ネットワークで繋がれたシステム
- それぞれのシステムやサブシステムの寿命が異なるものの集合体としてのシステム
- 要求の変化に対応し、進化し続けなければならないシステム

ディペンダビリティーの評価基準

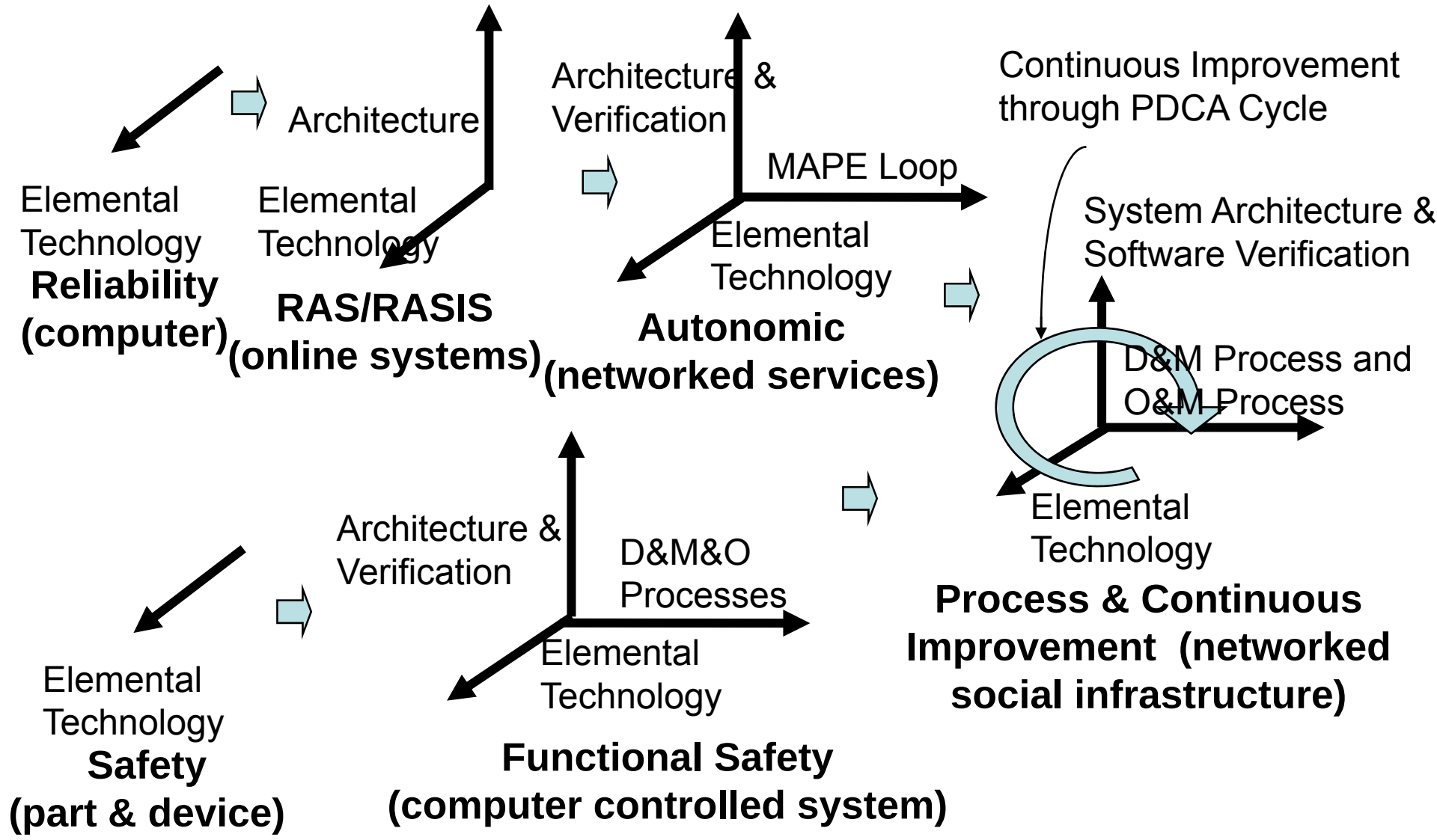
- Incidence(事故・故障)に対して、誰が説明し、どのように責任を取るか？ ⇒ どう作ったかよりも、ユーザーからどう見えるかが重要になる

- 説明責任を全うできるかどうかがディペンダビィディーの評価基準になる
- 説明責任は適切なリスクマネージメントをしてきたかどうかと言うこと

我々の方法

- 事故は必ずしも避けられない。ユーザーの要求の変化によるシステムの変更の必要性も避けられない。
- 我々は神様ではないので、完全なシステムを最初から作ることは出来ない。我々にできることは、地道で継続的な努力により、サービスを安全且つ継続的に利用者に提供すること。
- 即ち、事故の発生を最小限にし、被害を最小にし、短時間で復旧させ、同様な原因による事故が2度と起こらないようにする。
- システムの変更・改良が容易に行えるようなシステム構成とする。
- ディペンダビリティーを（1）関連する要素技術、(2)システム構成（アーキテクチャー）、（3）設計・製造ならびに運用・保守プロセス、（4）PDCAによる改良、の総合技術によって実現する。

ディペンダビリティーへの挑戦の歴史



大体方向はまとまってきたが、
もうひとつすっきりしない。。。。

なぜだろう、
何がすっきりしないのだろうか。。。

時代的な背景や、自分の立ち位置が明確でない。

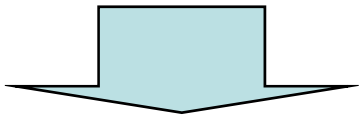
対象システムの定義がもうひとつすっきりしない。

科学的な方法論との対応がない。

Open System Science の考え方

ソニーCSL創立20周年記念シンポジウムで提案(2008.6.4)

- 21世紀初頭で人類が直面している最大の問題
  - 地球環境・エネルギー・資源の問題： 持続的社会の問題
    - 世界の気候、人口、食糧、生物多様性、格差、安全保障、
  - 健康・医療・生命の問題
  - 安全性の問題
    - 世界経済、食品
    - ネットワークで接続された巨大インフラ系
- これらは互いに関連する多数のシステムからなる統合システムの問題解決であり、Closed Systemに対して有効であったReductionismの手法のみでは解けない



- 新たなScienceの必要性： Open System Scienceの提案

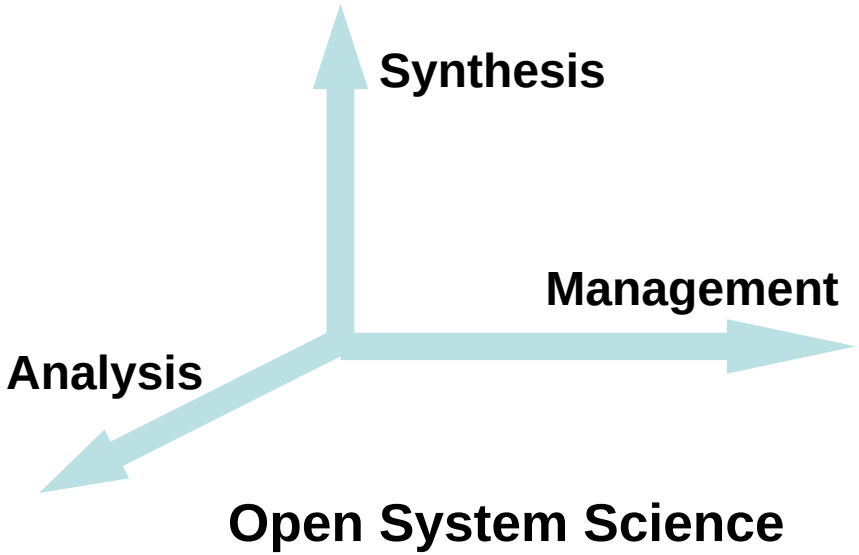
Open System Scienceとは（1）

- 複雑に相互に関連したサブシステムからなる
- 生きている形、あるいは実用として供している状況での問題解決
  - 境界領域や境界条件が動的に変化する
  - 定義や仕様が時間と共に変わる

- 還元主義（Reductionism）ならびに抽象化（Abstraction)だけでは問題解決ができない

Open System Scienceとは（2）

- 「真理探究の科学」から「問題解決の科学」へ
- 生かしたまま、あるいは稼働させたまま（止めずに）問題を解決するための科学方法論
- 「解析（Analysis)」と「合成（Synthesis」に加えて「マネージメント（Management)を統合・融合した新しい「科学」の形



これまでの科学方法論との比較

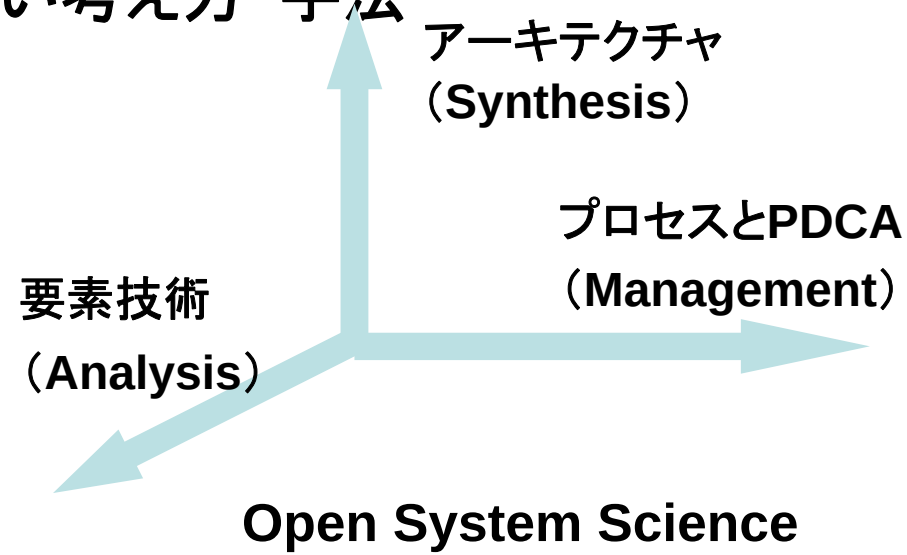
- これまではClosed System Hypothesisのもとでの Reductionism（領域分割と抽象化）による「基本原理の理解」
- これからはOpen Systemの継続性・持続性が最大かつ最重要課題
- Open System Scienceは「分析・合成・マネージメント」による「生きている/動いているシステムの問題解決」を目的とする

DependabilityとOpen System Science（1）

- ディペンダビリティー（システムが利用者に期待する便益をできる限り安全に且つ継続的に与えることができる性質）を要求するシステムはまさにオープンシステムである。
  - 複雑なシステム、
  - 人を含めたシステム、
  - ネットワークでつながれたシステム、
  - それぞれのサブシステムの寿命が異なるものの集合体としてのシステム、
  - 要求の変化に対応し、進化し続けなければならないシステム
- 還元主義と抽象化だけで問題を解くことは不可能
- オープンシステムサイエンスの手法が必須

DependabilityとOpen System Science（2）

- Dependabilityは「問題解決の科学」
- Dependabilityは生かしたまま、あるいは稼働させたまま（止めずに）問題を解決するための科学方法論
- Dependabilityは「要素技術（Analysis)」と「アーキテクチャ（Synthesis）」と「プロセスとPDCA（Management)」を統合・融合した新しい考え方・手法



DEOS プロジェクト

われわれは何をするか？

DEOSプロジェクトがやること

1. 要素技術とプロセス/PDCAサイクルをつなぐアーキテクチャの研究開発
   - 標準化・モジュール構造

2. デイペンダビリティ確保のためのプロセス/PDCAサイクルの研究開発とディペンダブルシステムの規格策定

結論

- ディペンダビリティーとはオープンシステムを対象とした概念であり、クローズドシステムに対する方法論では対応できない。
- ディペンダビリティーとはシステムが利用者に対し、利用者が期待する便益を安全に且つ継続的に与えることが出来る性質。
- ディペンダビリティーは（1）要素技術（解析的）、（2）アーキテクチャー（合成的）、（3）設計・製造・運用・保守プロセスとPDCAサイクル（マネージメント）により実現。
- 「要素技術」と「プロセスとPDCA」を結びつける「アーキテクチャ」と「ディペンダビリティー規格の策定」がプロジェクト成功のキー

引き続き議論すべき項目

- 標準化と多様性の減少の関係
  - 標準化により一つの設計・製品の改善に集中できる
  - 一方で、不具合が発生した場合には全てのシステムが被害を受け、大きな惨事に発展する恐れがある。
- オープンソース / オープンコラボレーション
  - 仕様策定、設計、正当性検証、改良をオープンに行える
  - PDCAサイクルによる改良を既に実行している
  - 多種多様な専門家やマネージメント経験者、思想家が共同で方向を決め、開発できる。
  - オープンソース / オープンコラボレーションの経済形体は将来どの用になってゆくか？　自己実現の経済はありうるか？