ソフトウエア品質とシステム信頼性を考える

障害事例に学ぶ

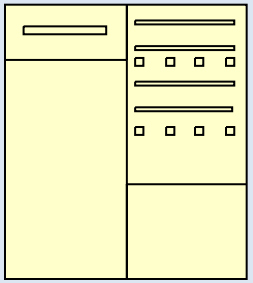
Part-1　システムの基礎知識
Part-2　システム障害の事例
Part-3　ソフトウェア品質の向上策

2012年10月27日（土）

航空運航システム研究会（TFOS.SG)
航空システム部会　松田　宏

問題認識

- 情報技術の急速な進歩（性能、価格）
  ⇒あらゆる分野でコンピュータを高度に利用



- ソフトウェアの品質がシステムの信頼性を左右
  ⇒重要性が認識されていない。法的にも不備？

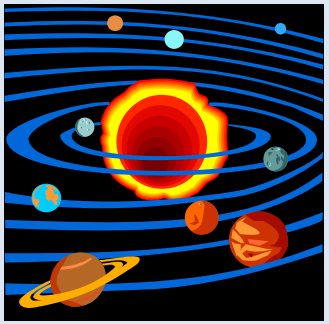
- 巨大で複雑なソフトウェアがブラックボックス化
  ⇒品質の確保や検証が非常に困難

- 同じようなシステム障害が昔から繰返されている
  ⇒他分野の障害事例から学ぶことが多い

Part-1　システムの基礎知識
①システムとは何か（1）

- 個々の要素が相互に影響しあいながら、全体として機能するまとまりや仕組み

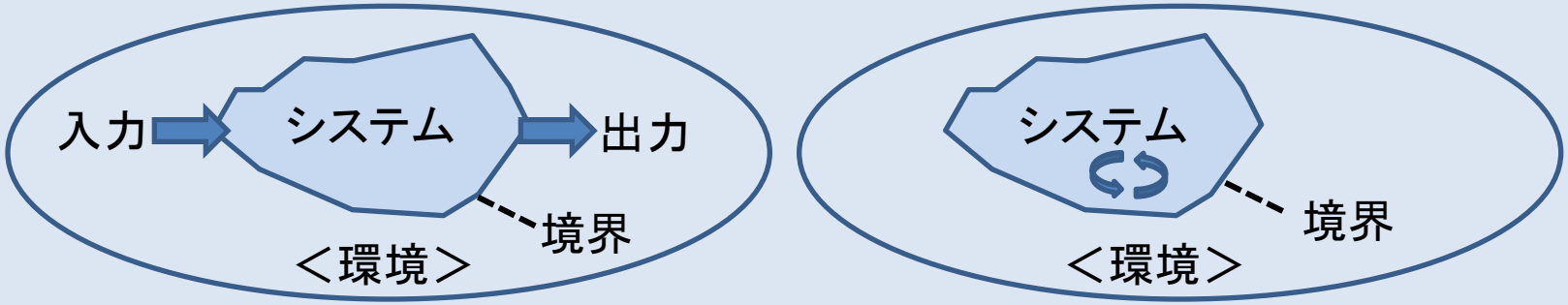
（系、体系、制度、方式、機構、組織など）



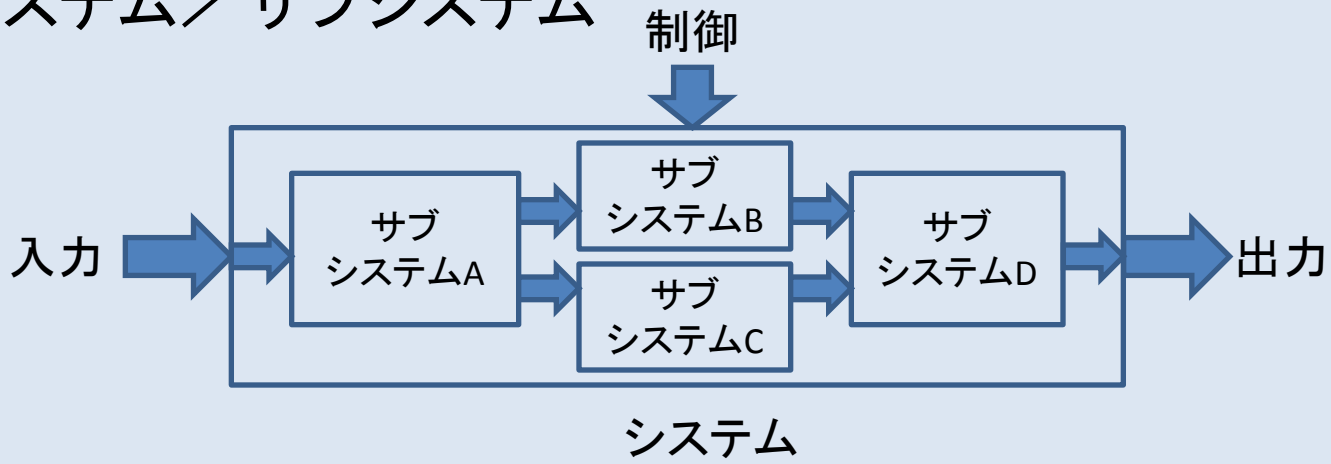
（例）
太陽系（Solar System）、生体系（Biological System）、
循環器系（Circulatory System）、呼吸器系（Respiratory System）、
力学系（Dynamical System)、代数系（Algebraic System)、
社会構造（Social System）、年金制度（Pension System）、
法体系（Legal System）、メートル法（Metric System）、
輸送システム（Delivery System)、鉄道システム（Railway System)、
計測システム（Instrumentation System)、制御システム（Control System）、
データ処理システム（Data Processing System)、
情報システム（Information System）、・・・

Part-1　基礎知識
①システムとは何か（2）

- 開かれたシステム（オープンなシステム）
  閉じたシステム（クローズドなシステム）



- システム／サブシステム



Part-1　システムの基礎知識
②システムの信頼性（1）

- 平均故障間隔
  （Mean Time Between Failure: MTBF)
- 平均修復時間
  （Mean Time To Repair：MTTR)
- 稼働率（Availability)

（例）99．9%　　1年間で8時間46分間停止
　　99．99%　　1年間で　　53分間停止
　　99．999%　　1年間で　　5分36秒間停止
　　99．9999%　1年間で　　32秒間停止
　　99．99999%　1年間で　　3秒間停止

（注）1年間＝3,600秒／時間×24時間／日×365．25日／年
　　＝31,557,600秒

Part-1　システムの基礎知識
②システムの信頼性（2）

- フェイルセーフ（Fail Safe)

  故障などにより障害が発生しても、常に安全側に制御する設計手法

  （例）鉄道車両はブレーキ故障時に非常ブレーキがかかる
  　　鉄道信号は異常時や停電時は必ず「赤」となる

- フェイルソフト（Fail Soft)

  故障個所を切り離して被害を最小限に抑え、機能は低下してもシステムを停止させず、主要機能を維持する設計方式

- フォールトトレラント（Fault Tolerant)

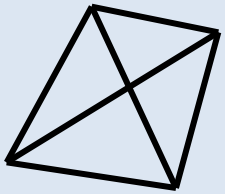
  一部に問題が生じても、全体が機能停止せず動作し続けるような設計手法

  （例）フォールトトレラント・コンピュータ（金融機関等）

Part-1　システムの基礎知識
③構成要素の多重化

- ハードウェアの多重化

（例）航空機のエンジン／航法計器／通信機器
　　航空路監視レーダー／対空通信設備
　　金融機関など重要な情報システム機器



- ネットワークの多重化

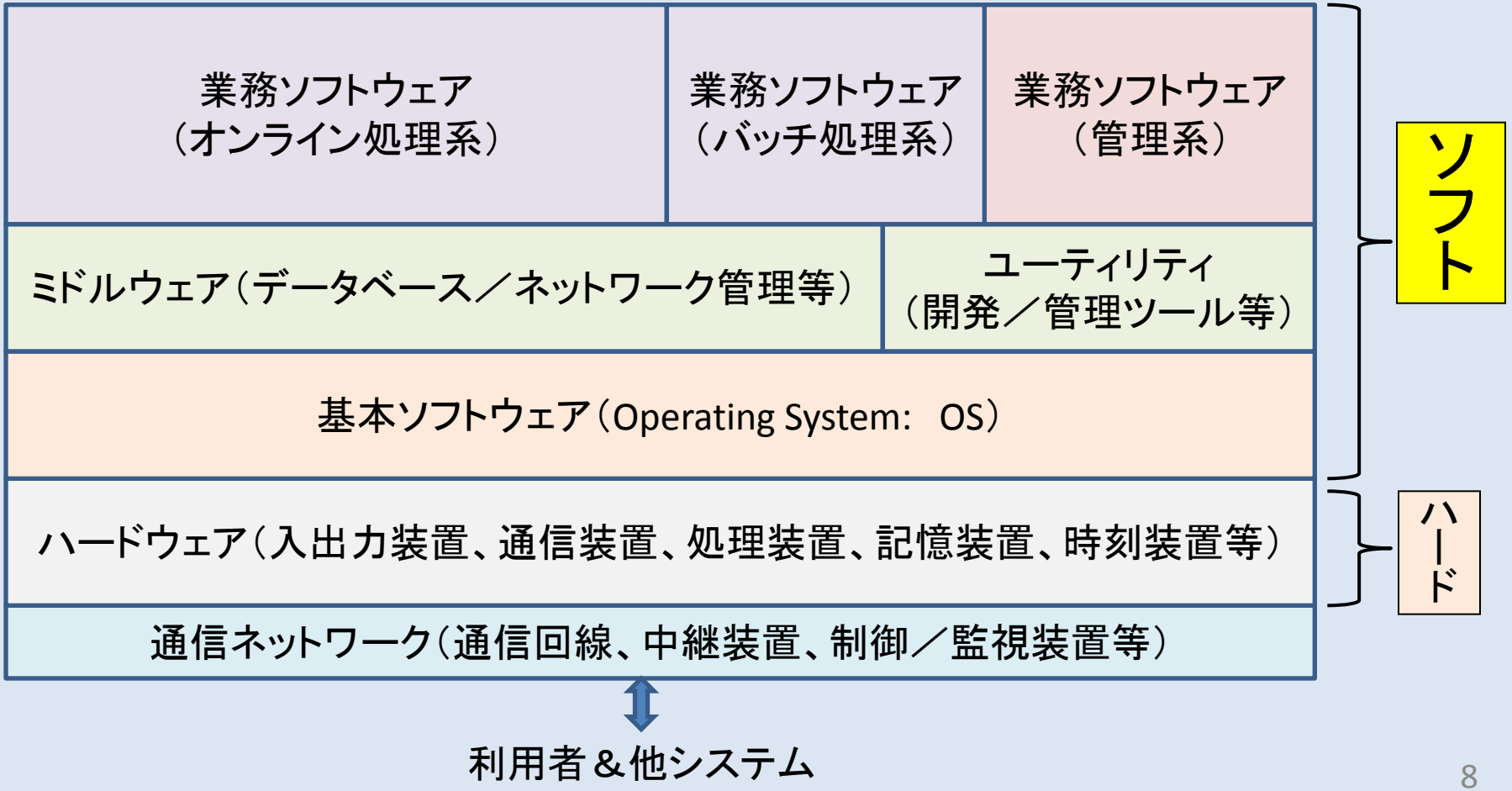
（例）予備回線／複数回線
　　インターネット（冷戦時代の核攻撃対策として開発された！）
　　２系統の商用電源

- 代替手段による業務機能の維持

（例）DARC（Direct Access Radar Channel)
　　管制塔のライトガン／バッテリー方式の非常用無線機
　　UPS（無停電電源装置）／非常用発電機

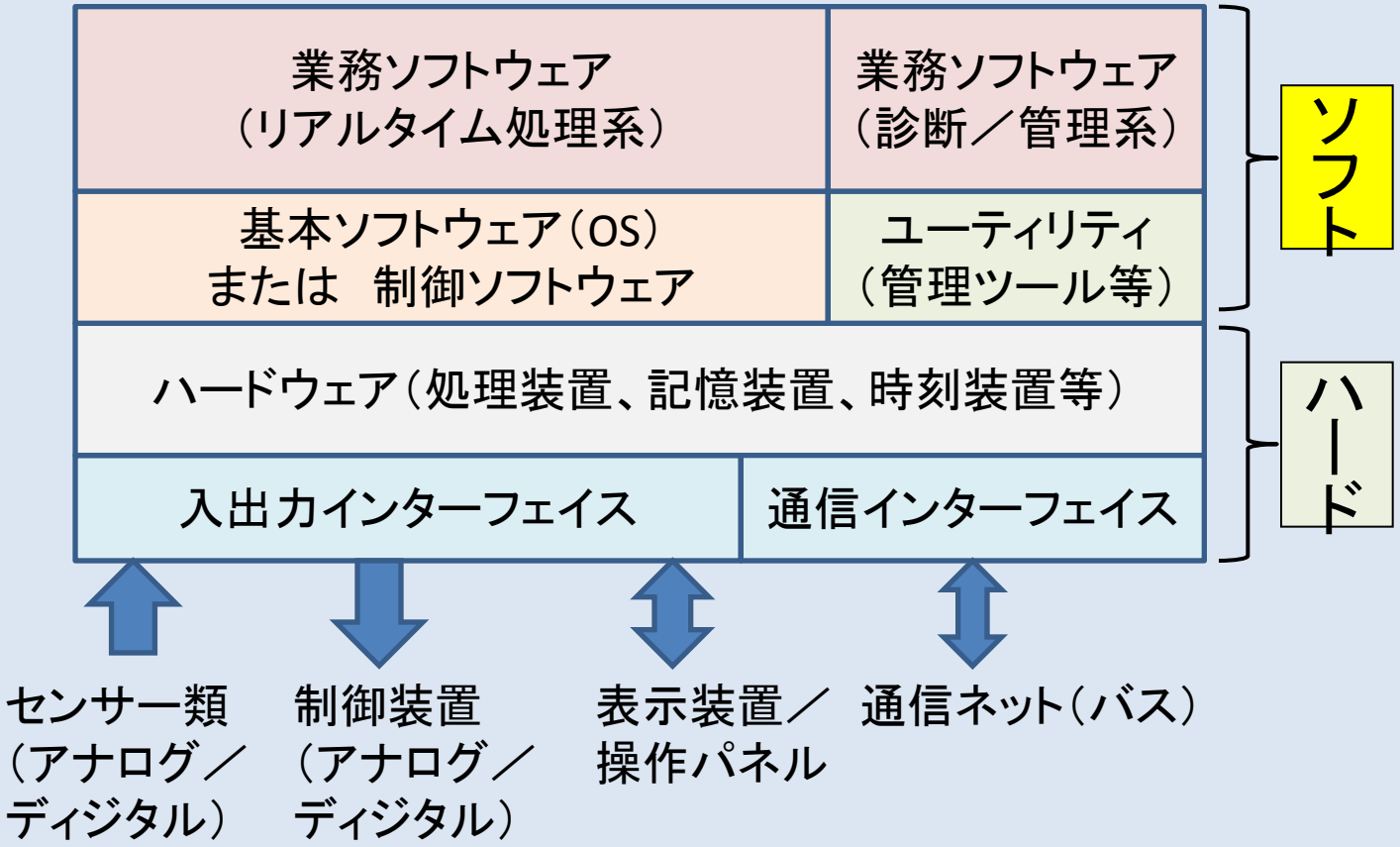
Part-1　システムの基礎知識
④情報システムの構成（1）

一般的な情報システムの構成
（ソフト比率が高い。汎用ソフトを多用）



Part-1　システムの基礎知識
④情報システムの構成（2）

制御システムの一般的な構成
（専用ソフトが多い）



Part-1　システムの基礎知識
⑤ソフトウェアの品質（1）

＜ソフトウェアの構成要素＞

- ソフトウェア設計書（基本設計書、詳細設計書、変更記録など）
- プログラムのソースコード（プログラム言語で記述）
  　標題等の基本情報、データ定義、実行コード、コメント
- プログラムのオブジェクト（実行可能な機械語形式）
- プログラムの実行に必要なパラメータ類
- プログラムが参照するデータテーブル
- ジョブ制御情報（JCL）／リンク情報

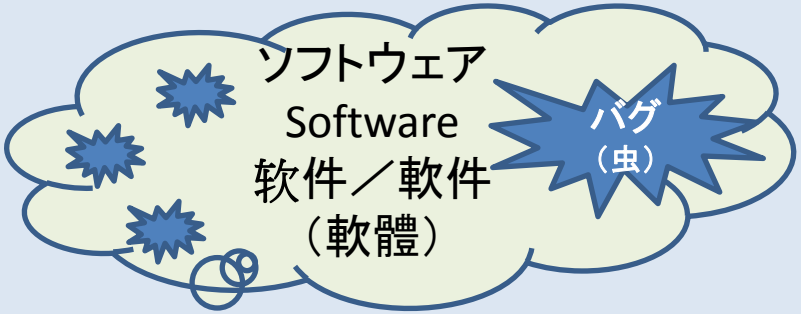
狭義のソフトウェア

- 試験仕様（試験の方法、データ、結果など）
- 保守履歴（障害記録、仕様変更／修正記録、運用記録など）
- 保守マニュアル／保守ツール
- 運用管理マニュアル／運用管理上のノウハウ
- 利用者用操作マニュアル／利用上のノウハウ

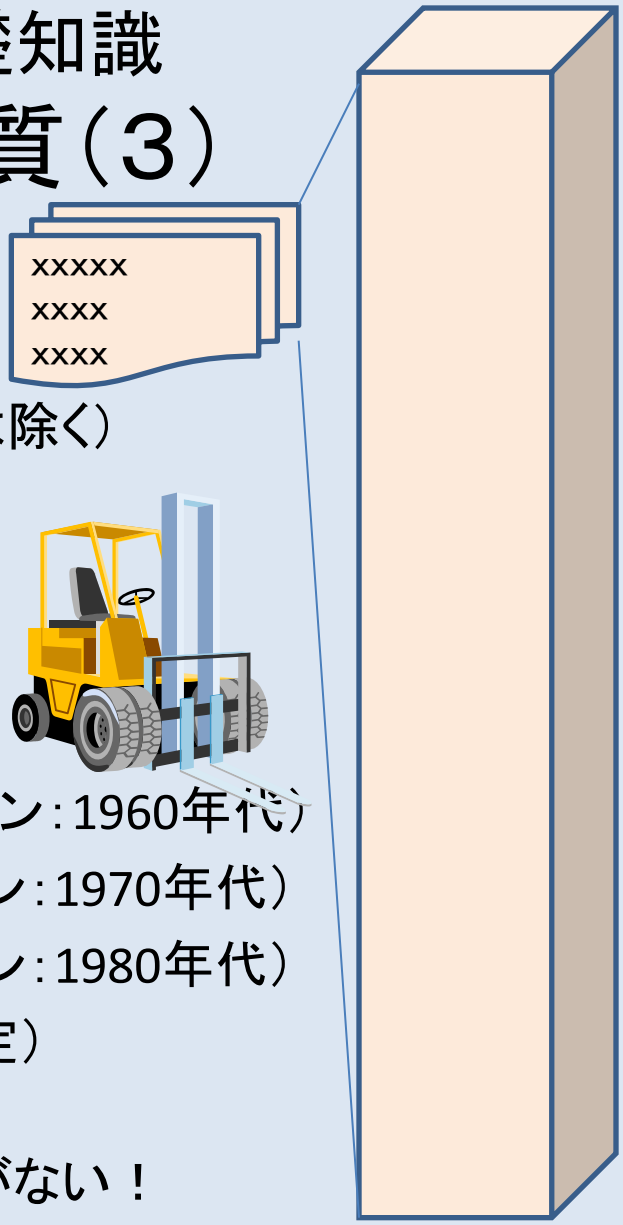
Part-1　システムの基礎知識
⑤ソフトウェアの品質（2）

＜品質の高いソフトウェアとは＞

- システムが停止しない
  （無限ループ、メモリ不正アクセス、入出力異常、不正演算などで停止）
- 必要なときに確実に動作する
- データが正しく入出力される
- 正しい処理結果が得られる（数値計算では一定の誤差範囲）
- 正しいタイミングで処理される
- 例外処理が適切に行われる
- データが無くならない
- データが正しく蓄積／更新される
- 操作／設定が容易
- 適切なチェックポイント／リカバリポイントが設定されている
- 機能の追加／変更、容量拡大などが容易（構造、論理、表現、・・・）



Part-1　システムの基礎知識
⑤ソフトウェアの品質（3）



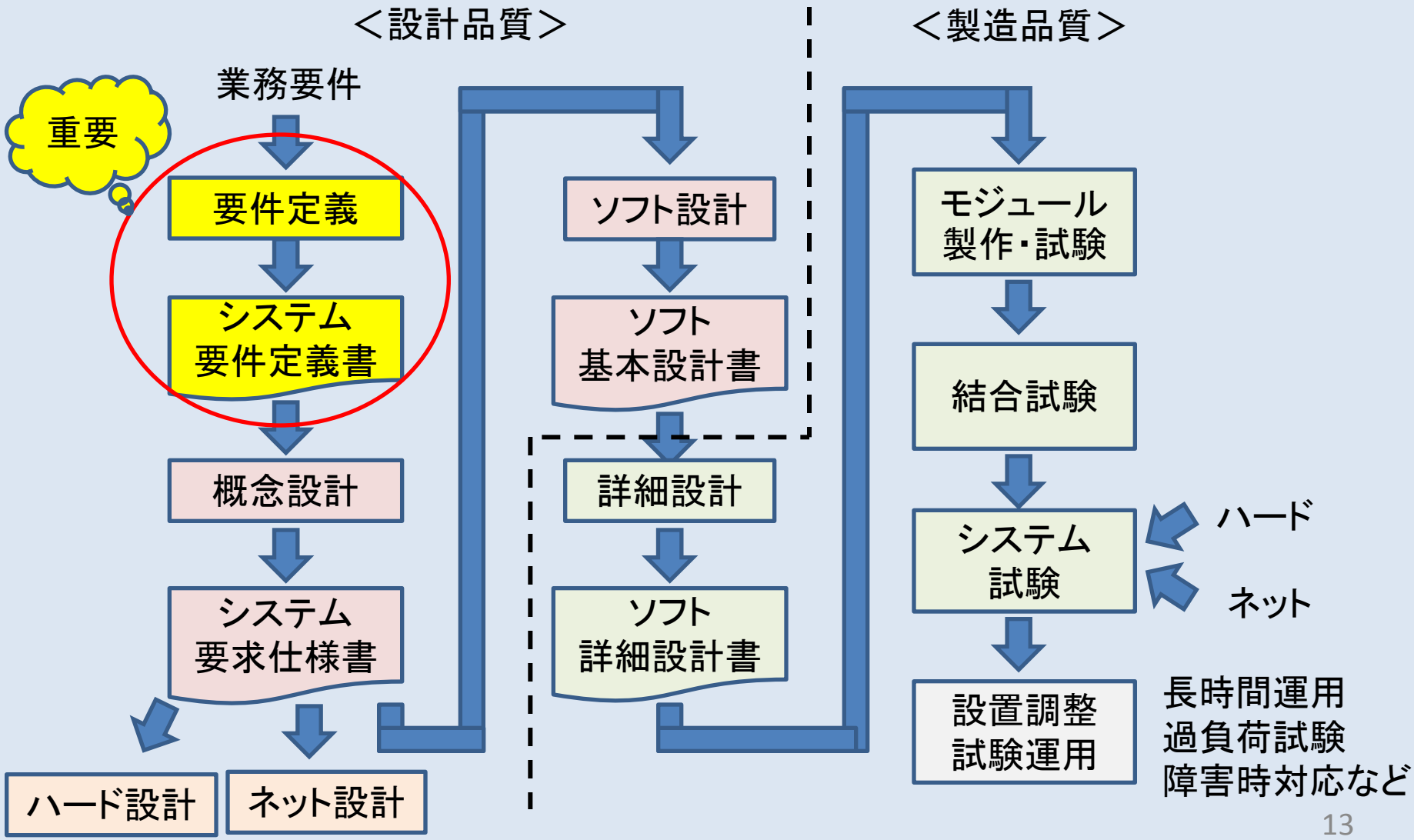
参考：プログラムの規模（おおその目安）

- 自動車　700万ステップ（カーナビは除く）
- 携帯電話　1,000万ステップ
- 社会保険庁　2,100万ステップ
- MS-Windows XP　4,000万ステップ
- ゆうちょ　5,130万ステップ
- 銀行システム　50万ステップ（1次オンライン：1960年代）
  200万ステップ（2次オンライン：1970年代）
  700万ステップ（3次オンライン：1980年代）
  10,000万ステップ（現在・・・推定）

＊これだけの規模で人為的な間違いがないはずがない！

大銀行合併時のシステムプロジェクト規模・・・6,000人（4,000億円）

Part-1　⑥設計品質と製造品質



Part-1　システムの基礎知識
⑦運用管理の品質

- 運用管理方式が未確立　　⇒　操作ミスなどによる障害
  要員の教育訓練が不十分　　（特に異常時対応）
- インフラ管理が不適正　　⇒　システム停止
  （電源／空調／ネットの障害）
- ソフト保守要員の技術力不足　⇒　初歩的なミス
- 変更履歴などの文書が不備　　⇒　保守後にトラブル
- 試験環境の整備が不十分　　⇒　変更の影響確認が困難

Part-2　システム障害事例
①設計ミス／想定漏れ

- 1981年4月に打ち上げられたスペースシャトル初号機（コロンビア）は、飛行制御システムのソフト不具合により1カ月遅れだった。

  多数決方式の複数コンピュータ間で、初期タイミングが何度やり直しても一致しなかったため。
  （主システム：4台＋予備システム：1台＝合計：5台）

- うるう年だった1996年の大晦日に、ニュージーランドのAluminum Smelter社のアルミ精錬工場ですべての溶解炉が停止。

  制御システムのプログラムが366日目の処理を想定していなかったため。損害額は100万NZドル。

Part-2　システム障害事例
②想定以上の過負荷

- 2006年1月18日に、東京証券取引所の売買システムがライブドア事件に伴う大量の売注文を処理できず、全銘柄の取引停止。

システムが旧式で拡張性がなく、処理能力を増強した新システムへの移行は2010年１月までかかった。

- 2011年3月19～21日の3連休に、みずほ銀行のすべてのATMが停止。

大震災の義捐金振込が集中し、夜間バッチ処理が翌朝のオンライン処理開始に間に合わなくなった。データ容量の上限設定ミスや操作ミスも重なった。

Part-2　システム障害事例
③操作ミス等の複合

- 2005年12月8日、みずほ証券が新規上場のジェイコム株の売注文を誤入力。「61万円で1株」⇒「1円で61万株」買注文の殺到で東証の売買システムがパンク。注文の取消ができずにいる間に多額の損害が発生。

  みずほ証券は414億円の損害賠償を求め、地裁は東証に約107億円の支払を命じた。みずほ証券はこれを不服として控訴中。東証は既に金利を含む１３２億円を仮払いした。

- 2012年6月20日、ファーストサーバ社のレンタルサーバに預けられていた5698社分のWEBデータが完全消失し、大部分が復旧不可能に。

  ファーストサーバ社内の作業ミスがきっかけ（詳細は不明）。

Part-2　システム障害事例
④障害処理の不具合

- 2012年8月7日、東証の派生商品（デリバティブ）売買システムの障害により全銘柄の取引を停止。

原因はデリバティブ取引システムと構内ネットの間のL3スイッチ（2重化）の障害。自動切替ソフトに不具合があった。

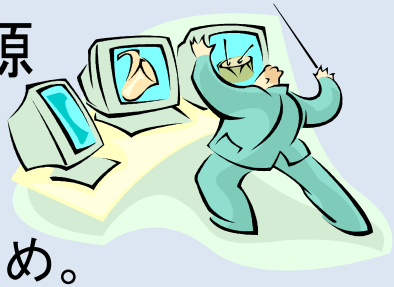
- 2012年1月25日、KDDIのau携帯電話、固定通信、法人系のサービスが東京都西部で利用しずらい状態になった。

制御基板のメモリ処理容量不足で動作異常となったが、冗長切替機能（ソフトウェア）が不備だった。

Part-2　システム障害事例
参考：東証のシステム障害

- 2001年6月12日　ソフト99コーポレーション株上場時に終日取引停止
- 2005年11月1日　プログラムミスにより全上場銘柄の取引を一時停止
- 2005年12月8日　みずほ証券が誤発注したジェイコム株の注文を取り消せず、多額の損害が発生（→事例紹介）
- 2006年1月18日　ライブドア事件で大量の売り注文にシステム処理が追いつかず、全銘柄取引停止（→事例紹介）
- 2008年2月8日　デリバティブ売買システムの障害で一部先物商品の取停止
- 2008年7月22日　デリバティブ売買システムの障害で、指数、株式・国政先物、オプション取引を一時停止
- 2012年2月2日　arrowhead内の3つのサーバで不具合が発生、午前9時より241銘柄の取引を一時停止
- 2012年8月7日　売買システムの障害により全派生商品（デリバティブ）銘柄の取引停止（→事例紹介）

Part-2　システム障害事例
⑤運用管理上の怠慢



- 2005年8月2日に、羽田空港の管制施設への電源供給が止まりレーダー等が使えなくなった。
  商用2系統、予備発電機があるにも関わらず、保守時の切替ミスで無停電電源装置のバッテリーを使い果たしたため。
- インターネットプロバイダA社データセンターで突然停電。
  サーバーを次々に増設し、電力消費量が限度を超えたため。管理者は容量の問題を全く意識していなかった。
- クレジット会社B社でシステム操作の記録ファイルがパンクし、システム停止。
  記憶装置の容量は十分にあるのに上限設定が不適切だったため。
- グローバル企業B社の名古屋データセンターは、伊勢湾台風で水没し、市のハザードマップの浸水予想地域にある。
  災害史を知らない管理者が安易に遊休地を利用したため。

Part-2　システム障害事例
⑥事前対策による業務継続（1）

- 1989年10月のサンフランシスコ大地震で通販会社の受注システムが停止。
  ⇒　電話オペレータ約200人を飛行機でシカゴに移動させ、オヘア空港隣接のバックアップセンターで業務を継続。

- 1996年5月にパリのクレディ・リヨネ銀行本店で火災が起き、金融商品取引システムが焼失した。
  ⇒　システムはブリュッセルとロンドンで相互バックアップされていたので、担当者が移動して翌日には業務を再開。

クレディ・リヨネ銀行の正面入口

Part-2　システム障害事例
⑥事前対策による業務継続（2）

- 2001年9月11日のアメリカ同時多発テロ事件でニューヨークの世界貿易センター入居企業は壊滅的な被害を受けた。

⇒　大手金融機関は、バックアップシステムを含む事業継続計画（BCP)が義務付けられていたため、業務機能もデータも喪失せず、代替要員により別な場所で業務継続できた。

＊ほとんどの中小企業はバックアップシステムがなかったため事業データを喪失し、その多くが倒産した。

Part-2　システム障害事例
⑦人的対応による解決

- 1969年7月20日、アポロ11号の月着陸船イーグルが月面降下中に、飛行制御システムがオーバーフローした。

  ⇒　エドウィン・オルドリン操縦士は飛行制御システムを切り、手動操縦で無事月面に着陸した。

- NASAの研究所の廊下に、「いざという時はこれを使え」と算盤（そろばん）が展示してあるという話は有名。
- NASAにAbacus Technology（算盤技術！）という関連企業があり、同社の解析ソフトAbacusは多方面で使われている。

2008年6月にスペースシャトル「ディスカバリー号」に搭乗した宇宙飛行士星出彰彦さんが携帯したアルミ製「宇宙ソロバン」（写真：慶応義塾大学理工学部）

Part-3　ソフトウェア品質の向上策
①適切な業務要件／システム要件

- システム以前に、業務分析やプロセスの見直し（Business Process Re-engineering：BPR）、業務の標準化などの問題を整理し、解決しておく。
- 業務要件／システム要件の定義に時間とお金をかける。
（結果的に開発費用の節約ができる場合が多い）
- 業務を熟知したシステム利用者が積極的に関与する。
（システム部門任せ／技術者任せで良いシステムはできない）
- 経営者／管理者など、組織トップが積極的に関与する。
（業務、人、モノ、金に決定権のある人が責任を持つ）

Part-3　ソフトウェア品質の向上策

②漸進的な改善改良／単純なシステム

- 一足飛びに画期的、革新的なシステムを開発計画は、途中で挫折することが多い。
（技術トラブル、所期の性能が出ない、予算オーバー、開発遅延など）

- 実績に基づく経験を積み重ねながら斬新的に改善／改良していく開発手法が望ましい。
（新技術／製品の採用は費用対効果が大きいがリスクも大きい）

- 多くの機能を盛り込んだ複雑なシステムは開発リスクが高くなり、信頼性を低下させる。操作ミスも避けられない。
（できるだけ単純で洗練されたシステムが望ましい）

Part-3　ソフトウェア品質の向上策

③例外的な事象の適切な想定

- 機器故障や誤入力、例外データ、操作ミスなど、起こりうる事象を可能な限り幅広く想定する。

- 経験豊富なベテラン技術者を関与させ、システム障害の基本的なパターンをすべて検証する。

  （技術は進んでも障害の基本的なパターンは昔から変わらない。想定外というのは不勉強か怠慢であることが多い）

- 異常が発生した際の人間系との整合性をとっておく。

  （あらゆる例外を想定し完璧な対応をすることは不可能。また、例外処理が正しく行われるかどうかを試験することも困難）

Part-3　ソフトウェア品質の向上策
④構造化設計と設計開発ツール

- ソフトウェア工学的な（合理的な）開発手法を使う。
  （例：データモデリング、プロセスモデリング、構造設計、標準化など）

- 処理効率の低下はハードウェア性能でカバーする。
  （リアルタイムシステム（特に搭載型システム）の場合は、開発効率と処理効率や必要資源とで適切にトレードオフする）

- 設計開発ツールや目的向専用言語を活用する。
  （膨大な行数のソースコードを人間が書くと間違いが避けられない）

- 実績のある市販パッケージを活用する。
  （技術者はすべて自分で作りたがる・・・
  Not Invented Here:　NIH)

Part-3　ソフトウェア品質の向上策
⑤ライフサイクルコストと効果の想定

- 寿命を考慮し開発と維持の費用バランスをとる。
（開発費を節約した結果、維持費がかさむ例が少なくない）

- 技術進歩や業務変化に対応できるよう、柔軟性のあるシステム投資計画を立てる。
（機器の性能向上やコストの低下を予想し、対応策を考えておく。
税制上の耐用年数は実態に合わない！）

- 投資の費用対効果と障害リスクを的確に想定しておく。
（費用節減して障害が起き、大損害を出すことが多い）

- 陳腐化したシステムは廃棄する決断も。
（短期間で使い捨てる前提のシステム投資もある）

わが国におけるソフトウェア品質の諸問題

- 独自仕様の一品料理的な手造りソフトウェアが多い。
  （コンピュータのメーカーが多く、パッケージ市場が育たなかった）
- システム障害時のリスクが評価されていない。
  （費用節減が優先し、適切なシステム投資計画が行われない）
- 開発費が工数で見積もられ、品質が評価されない。
  （低単価技術者の人海戦術で、品質向上のインセンティブがない）
- ソフトウェアの品質保証が行われていない。
  （瑕疵担保による補修義務のみ。損害は免責という契約）
- 経営者の理解が不十分で、システム投資が不適切。

今後の研究課題(提案)

- 適切な信頼性要件の設定方法
- 障害時の対応方法(人的な対応を含む)
- 要員の教育訓練(利用者、運用者、管理者、開発者)
- 標準化と陳腐化の矛盾の解決方法
- インターオペラビリティ(相互運用性基盤)の確保

＜ご静聴ありがとうございました。＞