ソフトウェア品質とシステム信頼性を考える

A Consideration on Software Quality and System Reliability

TFOS.SG 航空システム部会　松　田　　宏
（松田宏コンサルティング㈱　代表取締役）

キーワード： システム工学、信頼性工学、ソフトウェア工学、システム分析、業務要件、要件定義、開発手法、プロジェクト管理、品質管理、運用管理、システム障害、災害復旧、事業継続計画、ライフサイクル、費用対効果、システム投資

はじめに

大規模集積回路（Very Large Scale Integration: VLSI）技術の進歩によりコンピュータの性能は著しく向上[1]し、今もその勢いは衰えていない。その結果、コンピュータは劇的に小型化し、大量生産により価格も急激に低下したため、企業や官公庁から家電製品に至るまで、社会のあらゆる分野で高度に利用されるようになった。しかし、コンピュータを応用したシステムの信頼性はソフトウェアの品質によって大きく左右されるため、新たな問題が起きている。システムが複雑になり、ソフトウェアが巨大なブラックボックス化したため、品質の確保が非常に困難になってきたからだ。

技術進歩にもかかわらず、ソフトウェアに起因するシステム障害は昔から同じようなパターンが繰り返されている。また、分野は異なっても障害パターンにはかなり類似性がある。他分野の事例は航空システム[2]の信頼性を考えるうえで良い参考になるので、典型的なパターンの障害事例を通してソフトウェアの品質問題を考えてみたい。

1　システムの基礎知識

ソフトウェアの品質とシステムの信頼性を議論する前提として、システム、信頼性、ソフトウェア、品質管理等に関する基礎的な知識をここで整理しておきたい。

---

[1] 1965年に米インテル社の共同創業者ゴードン・ムーアが集積回路上のトランジスタ数は2年で倍になると予言し、「ムーアの法則」と呼ばれている。その後、2倍になる期間は18カ月に短縮され、未だに有効だとされている。この速度は10年間で32~97倍に相当する。

[2] TFOS.SG 航空システム部会では、①航空機搭載システム、②運航管理システム、③航空交通管理システムに大別して研究を行っている。

1.1　システム(System)とは何か

システム[3]とは「複数の要素から成り、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」のことである。分野により系、体系、制度、方式、機構、組織などの訳語がある。

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

システムはある環境内に存在する。そして、外部との境界を通して入力や出力、制御[4]が出入りする「開かれた(Open)」システムと、内部だけで完結している「閉じた(Close)」システムに大別される。宇宙空間にある宇宙船、ガラス容器に水、水草、空気、金魚を密封した生態系などはクローズド・システムと考えることができる。ただし、前者の場合は太陽電池による発電、後者の場合は太陽光による水草の葉緑素による光合成を行っているので、厳密に見れば完全なクローズド・システムではない。

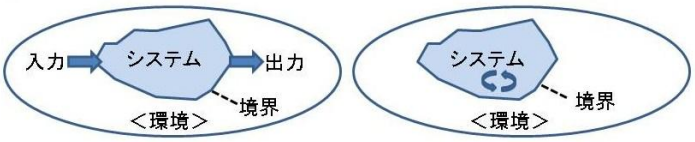

図 1　オープン（左）とクローズ（右）

なお、情報技術業界ではソフトウェアやデータが特定のメーカーやシリーズ、機種などに限定されないという意味の「オープン・システム（Open System）[5]」という概念があるので、区別が必要である。

また、ひとつの大きなシステムが複数の小さなシステムから構成される場合、その構

[3] ISO 9000-2005(JIS Q 9000-2006) 3.2.1 項では、「相互に関連する又は相互に作用する要素の集まり」と定義されている。
[4] 制御もシステム入力のひとつだが、一般的なシステムモデルでは制御を分けて表現する。
[5] 典型的な例として UNIX や LINUX、Windows などの基本ソフト（詳細は後述）がある。

成要素である小さなシステムをサブシステムと呼ぶ。

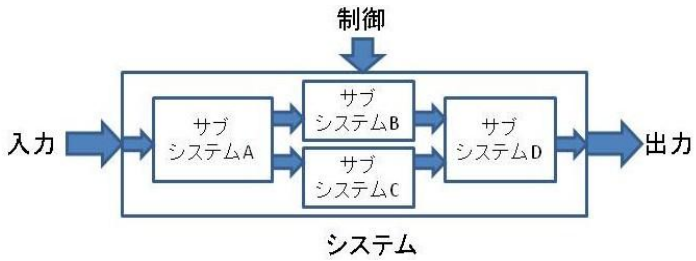

図 2　システムとサブシステム

1.2　システムの信頼性

システムの信頼性は一般に「平均故間隔[6]（Mean Time Between Failure: MTBF)」で表される。また、保守性には「平均修復時間（Mean Time To Repair)」が使われる。可用性は「稼働率（Availability)」で示される。一定時間内にシステムが稼働する時間の割合[7]を%である。稼働率のレベル毎に、1 年間[8]の停止時間を以下に示す。

＜稼働率＞	＜停止時間＞
99.9%	8 時間 46 分間
99.99%	53 分間
99.999%	5 分 36 秒間
99.9999%	32 秒間
99.99999%	3 秒間

システムの信頼性を高めるために、さまざまな設計手法が採用されている。「フェイルセーフ（Fail Safe)」は、システムに障害が発生した場合、常に安全側に制御する設計手法である。例えば鉄道車両はブレーキが故障した場合に必ず非常ブレーキがかかり、鉄道信号は異常時や停電から復旧した場合は必ず「赤」になる。

「フェイルソフト（Fail Soft)」は、故障個所を切り離して被害を最小限に抑え、機能や性能が低下してもシステム全体は停止しないようにする設計方式である。

---

[6] 保守などの計画的な停止時間は MTBF の計算から除外する。

[7] 稼働率＝MTBF/(MTBF+MTTR)の式で計算される。

[8] 4 年に一度のうるう年を考慮した単純計算で 1 年間≒365.25 日≒31,557,600 秒。実際には 100 年ごとと 400 年ごとに例外の補正があり、1 太陽年＝365.24219 日である。

「フォールトトレラント(Fault Tolerant)」は、機器の多重化や並行運転により、部分的な障害が起きても全体の動作を継続可能にする設計手法である。フォールトトレラント・コンピュータ[9]は主に金融機関などの重要なシステムで使われている。

1.3　構成要素の多重化

信頼性を高めるためには、システム構成要素の多重化が有効である。例えば、航空機のエンジンや搭載航法計器、航空管制用の航空路監視レーダー、対空通信設備、金融機関のオンライン系情報システム、電話会社の交換設備などである。

ネットワークの多重化も信頼性向上に効果的である。予備の通信回線を設置する、電力会社の別々な配電系統から２系統で電源を引き込む、などである。インターネットは、通信回線を網目状にし、中継点が故障してもメッセージを自動的に迂回させる機能をもつが、冷戦時代に核攻撃を想定して軍事目的で開発されたものである。

他に、本来とは別種の機器など代替的な手段で業務機能を継続する場合もある。米国FAA ではレーダー情報処理システムが故障した場合に備え、Direct Access Radar Channel: DARC)という簡易システムを設置している。空港の管制塔にはバッテリーで動作するライトガンや非常用無線機が設置されている。重要な施設には無停電電源装置（Uninterrupted Power Supply:　UPS）[10]や非常用発電機が設置されている。

1.4　情報システムの構成

情報システムはハードウェアとソフトウェアとネットワークで構成される。地上に設置される一般的な情報システムでは、標準的なハードウェア、汎用の基本ソフトウェア(Operating System : OS)[11]とミドルウェア（Middleware）[12]をプラットフォームとし、その上で業務ソフトウェア[13]を動作させる。近年のオープン・システムの普及に伴い、業務ソフトウェアの主要部分は市販のパッケージ・ソフトウェアを使い、独自業務の部分だけ新規のソフトウェアを開発する場合が増えている。

---

9 主要部分が複数ユニットで構成され、電源や信号ラインも二重化された高信頼性コンピュータ。動作中にユニットを取り外したり挿入したりしても動作は正常に続けられる。

10 バッテリーと定電圧定周波数装置を組み合わせ、非常時に一定時間、電源を供給する装置。

11 ハードウェアの制御や入出力の管理などの基本機能を業務ソフトウェアに提供し、開発や運用を容易にする汎用のソフトウェア。

12 データベース管理／ネットワーク管理、開発／管理ツールなど、OS と業務ソフトの中間に位置するソフトウェアの総称。

13 アプリケーションソフトウェアとも呼ばれる。データを即時処理するオンライン処理と、まとめて処理するバッチ処理とに大別される。

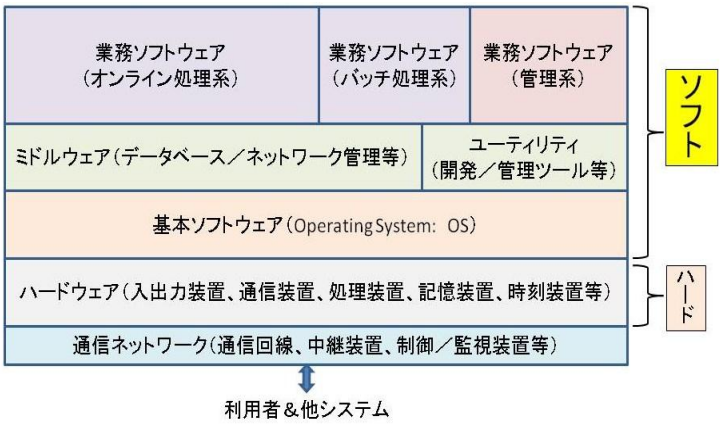

図 3　一般的な情報システムの構成

制御システム、特に航空機や鉄道、自動車などに搭載されるシステムの場合は機器の重量や容積、振動や動作温度範囲などの環境条件が厳しいので、特殊な専用コンピュータが使われることが多い。また、センサー類からの信号や制御装置への信号を入出力するための特殊なインターフェイス[14]を備えている。

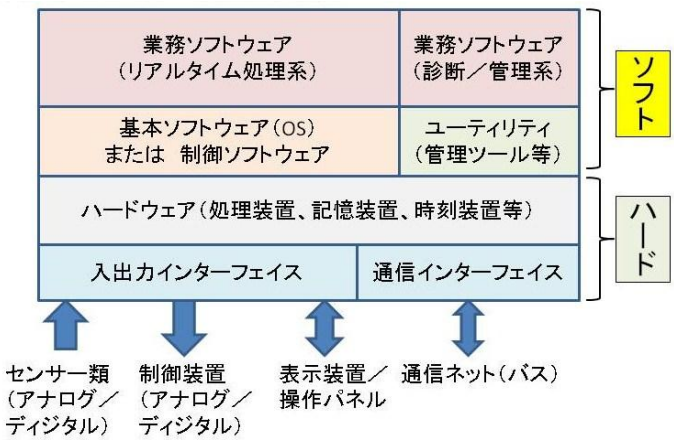

図 4　標準的な制御システムの構成

制御システムでは処理速度や記憶容量など性能面の条件も厳しいため、汎用ソフトウェアは冗長すぎて使えない。基本ソフトウェアとしては効率が高く小規模な制御用 OS か、モニターと呼ばれる必要機能だけを凝縮した専用ソフトウェアを使用する。業務ソ

[14] アナログ信号をディジタル信号に変換する AD（Analogue／Digital）変換装置や、ディジタル信号をアナログ信号に変換する DA(Digital/Analogue)変換装置など。

フトウェアは、プロセスの並列処理や同期を管理できるセマフォ（semaphore）など特殊な機能があり、処理速度が速いプログラミング言語で開発されることが多い。

1.5　ソフトウェアの品質

フトウェアとは、コンピュータ・プログラム（ソースコード[15]およびオブジェクト[16]）、実行に必要なパラメータ、データテーブル、制御情報[17]などを指す場合が多い。しかし、広義には、設計書（基本設計書、詳細設計書）や試験仕様（データ、結果を含む）、保守履歴、保守／運用管理／利用マニュアルなどのドキュメントや各種ノウハウを含む全体を指す。品質の高いソフトウェアとは、一般には次のようなものをいう。

- システムが停止しない[18]
- 必要なときに確実に動作し、処理時間が短い
- 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　（前後関係や同期など）
- 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
- 操作／設定が容易　が容易である
- 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、・・・）

巨大システムの場合、プログラムの記述行数は膨大な量となる。尺度として厳密ではないが概略の規模をコメント行を除くステップ数で表すと、自動車で 700 万行（カーナビを除く）、携帯電話で 1,000 万行、社会保険庁システムで 2,100 万行、MS-Windows XP で 4,000 万行、郵貯システムで 5,130 万行と言われている。初期の銀行システム[19]では 50 万～100 万行であったが、今では 1 億行に近いといわれている。印刷したソースプロ

15 人間が読むことができるプログラム言語で記述されたもの。
16 コンピュータが読み取り、実行できる形（機械語）に翻訳されたもの。
17 ジョブ制御言語（Job Control Language: JCL）による定型処理の手順（Catalog）やプログラムモジュールのリンクを指定する情報など。
18 プログラム無限ループ、メモリ不正アクセス、入出力異常、不正演算（数値を 0 で割算）、その他の異常があると、システムは停止する。
19 1960 年代の銀行第一次オンラインシステムの場合。

グラムリストを運ぶにはフォークリフトが必要、というのは本当の話である。

このような膨大なプログラムを多数の技術者が分担して開発するので、ヒューマンエラーによる記述ミスや相互矛盾は避けられない。プログラムに潜む誤りを「虫(bug)」、虫を探して修正する作業を「虫取り(debug)」と呼ぶが、論理的な条件の組合せの数は天文学的ですべてを試験することはできず、完全に根絶[20]することは不可能に近い。

図 5　ソフトウェアのバグ(虫)

システム開発費用も莫大で、大銀行の合併プロジェクトでは参加技術者数 6,000 人、費用 4,000 億円という例[21]がある。地方銀行では単独で開発費用を負担できないので、共同開発が増えている。自動車の車載電子制御システム[22]も規模が大きくなり、共通部分は複数の会社がコンソーシアムを結成して共同開発する例が増えている。

1.6　設計品質と製造品質

ソフトウェア製品の品質は、設計品質と製造品質とに大別される。しかし、それ以前に重要なのは、設計の前提となる業務要件やシステム要件の定義段階である。適切な仕様が設定されずに開発されたシステムは実際の業務には適合しないからである。

予算や納期の制約で、設計工程が不十分なまま製造工程に進めてしまうことがあるが、設計の手戻りが多発し、設計品質と製造品質の両方が低下することが多い。

製造段階では、適切な開発手法と最新の自動化ツールを使い、必要な知識と経験を持つ開発要員を確保し、適切なプロジェクト管理[23]を行えば、ソフトウェアの品質は十分なものになるはずである。事実、市販パッケージの開発では品質が利益の源泉であるため、技術レベルの高い（給料の高い）優秀な人材を確保し、開発効率の高い最新の手法やツールを駆使し、高い生産性と品質を確保し、十分な利益を上げている。

---

[20] 開発段階では、バグの発見頻度の変化から統計的に潜在バグ数を推定したり、第三者が故意にバグを挿入し，発見率から全体を推計したりする方法でプログラムの品質を検証する方法も使われる。
[21] 東京三菱 UFJ 銀行の事例。既存のものを利用し、新規開発は必要最小限とした場合の規模。
[22] 新機能の 90%が電子制御部品で実現され、開発費の約 60%がソフトウェアである。トヨタクラウンに実装されたマイコンは約 75 個だったが、最新型のレクサスでは約 100 個に増えている。
[23] 工程管理、資源管理、技術管理、変更管理、品質管理などの総称。

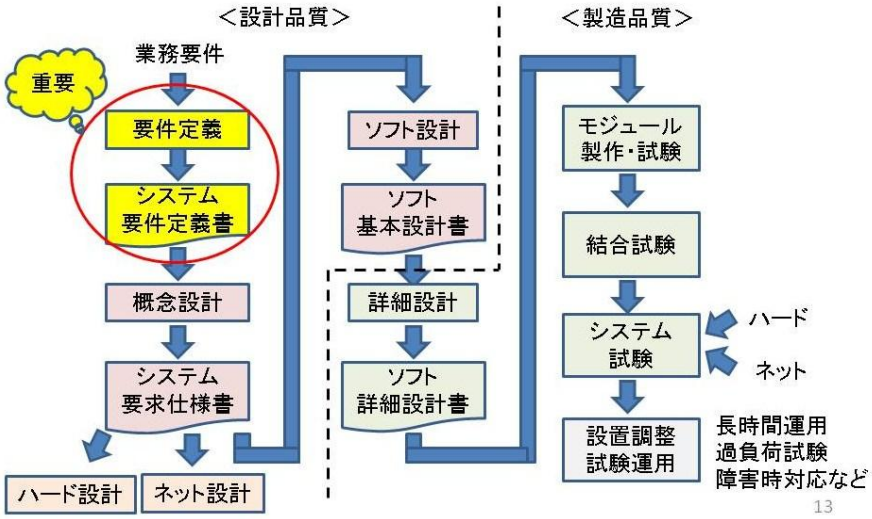

図 6　ソフトの設計品質と製造品質

しかし受託によるシステム開発では、予算不足や期間の制約、コスト重視偏重による品質に対する認識不足により、品質は理想とはほど遠いのが実態である。昔ながらの開発手法や古いプログラミング言語に固執し、効率の良いツールも使わず、相対的に技術レベルの低い（給料の安い）多数の開発要員による人海戦術が多いためである。もちろん品質管理も不十分で、システム信頼性向上の大きな阻害要因になっている。

1.7　運用管理の品質

運用管理の良し悪しもシステムの信頼性を大きく左右する。設計時に運用管理の方式を十分に検討していないことに起因する問題もあるが、体制の不備によるシステム障害も少なくない。運用管理は定常業務なので、コスト削減により人、モノ、金などの必要資源が十分に配分されない傾向が強いからである。

システムの起動や停止、稼働状態の変更、保守など運用管理の方式や手順が明確にマニュアル化されていないこともある。また、担当要員が十分な教育訓練を受けていないことがある。結果として、初歩的な操作ミスなどによるシステム障害が起きる。特に異常事態の場合、対応がマニュアル化され定期的に訓練を繰り返していなければ、設計や開発に携わった技術者でなければ適切な対応は困難で、事態を悪化させてしまう。

システムの稼働に必須の電源や空調、通信ネットワークなどのインフラの管理も軽視されがちである。そのため、管理上の怠慢や操作上の不注意などによる初歩的ミスの影

響が拡大し、重大な障害が長時間継続するという事故が絶えない。

さらに、システムを長期間運用していればソフトウェアに様々な追加や変更が加えられるが、履歴が適切に文書化され保管されていないことが多い。過去の変更を知らずに新たな変更を加えれば矛盾が発生し、重大なシステム障害が起きる。

一部の変更が他に影響を与えないことを確認するのは非常に困難で、開発の最終段階と同様の大規模で総合的な試験が必要だが、人手だけでは困難である。自動的にテストデータを作成したり、結果の妥当性を自動的に照合したり、過負荷状態を作り出したりする試験用シミュレータが必須だが、そうした準備がない場合が多い。

２．システム障害の事例

ここでは、ソフトウェアに起因する典型的システム障害の事例を、パターン別に紹介する。なお、実名が示されている事例は報道された公開情報によるもの、匿名の事例は個人的な情報源によるものである。

2.1　設計上のミスや想定漏れ

（1）スペースシャトル打上げ延期

NASA のスペースシャトル初号機コロンビアは 1981 年 4 月に打ち上げられたが、実は搭載している飛行制御システムのソフトウェアに不具合があり、原因調査と修正に時間かかかり、当初予定よりも 1 カ月遅れであった。

システムは、IBM 製の 32 ビット汎用コンピュータを 5 台使い、そのうちの 4 台が同じソフトゥエアでデータを処理し、結果を照合する多数決方式を採用していた。残りの 1 台は上記 4 台による多数決が成立しない場合に備え、全く別に開発されたソフトウェアによって飛行制御のためのデータ処理を行うことになっていた。

当初予定日に起きたトラブルは、各コンピュータにプログラムをロードし、起動するタイミングが何度やり直しても一致しなかったという初歩的なものであった。

（2）うるう年の大晦日にアルミ溶解炉が停止

1996 年はうるう年で、元旦から大晦日までの日数は 366 日だった。その大晦日に、ニュージーランド南島 Tiwai Point にある Aluminum Smelter Ltd.社のアルミ精錬工場の全ての溶解炉が停止した。工場の制御システムが、1 年は 365 日であると想定して設

計[24]されていたので、運転開始後の最初のうるう年であったこの年に366日目のデータを処理できなかったのである。損害額は100万ニュージーランド・ドルであった。

2.2　想定以上の過負荷（低すぎた処理負荷想定）

（1）東京証券取引所の取引停止

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、全銘柄の取引を停止した。旧式の汎用大型システムであったため拡張性がなく、処理能力を増強した新システム[25]移行は2010年1月までかかった。

（2）みずほ銀行の全ATMが停止

2011年3月19～21日の3連休に、みずほ銀行のすべてのATMが停止した。東日本大震災の義捐金振込が特定の口座に集中し、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ためである。口座データの容量上限の設定ミスに気付かなかったこと、処理が遅れたデータの日付更新処理の際のオペレータの操作ミスなどが重なり、復旧が遅れた（後述するシステム運用管理上の怠慢にも該当する）。

2.3　操作ミス等の複合

（1）　ジェイコ株誤注文事件（利用者の入力ミス＋過負荷＋不適切な対応）

2005年12月8日に、みずほ証券の担当者が新規上場のジェイコム株の「61万円で1株」とすべき売注文を東証システムに「1円で61万株」と誤入力した。誤入力ではないかとの警告が表示されたが、よく見ずに無視したという。

破格の安値の売り注文であったため買注文が殺到し、東証の売買システムはパンクした。そのため注文の取消ができず、東証と電話でやりとりしている間に多額の損害が発生した。みずほ証券は東証に対し414億円の損害賠償を求める訴訟を起こし、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であるが、東証は金利を含む１３２億円を仮払いした。最終判決はまだ出ていない。

---

[24] 日付や時刻によるソフト障害は非常に多い。事前対策により大きな事故はなかった2000年問題、不定期に実施される「うるう秒」問題、UNIX時刻の2038年問題など。多くは仕様上の問題。

[25] オープン・システム技術を初採用した東証アローズ。ニューヨーク初め、香港、上海、シンガポール、ロンドン、フランクフルト（現在はユーロネクスト）など、世界の主な証券取引所はかなり以前から拡張性の高いオープン・システムを使っていた。

（2）大量の顧客データを誤消去（運用操作者の操作ミス＋設計上の考慮不足）

2012 年 6 月 20 日に、ファーストサーバ社（本社大阪）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、復旧不可能になった。社内の作業ミスがきっかけで次々にデータが消えたとされているが、詳細は不明。

2.4　障害時の処理ソフトウェアの不具合

（1）機器障害時の自動切替ソフト不具合（設計上の考慮不足＋試験不足）

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した。原因はデリバティブ取引システムと構内ネットワークの間にあるＬ３スイッチ[26]に障害が起き、二重化されている他の系に自動切替するソフトに不具合があったため。

（2） 異常処理ソフトの不備（設計上の考慮不足＋試験不足）

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にくい状態になった。制御基板のメモリ処理容量不足で動作異常となったが、冗長系（予備系）への切替え機能[27]を果たすソフトウェアが不備だったため。

2.5　運用管理上の怠慢

（1）空港管制塔の電源停止（運用管理方式の不備、要員の教育訓練不足）

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。商用電源２系統と予備発電機があるにも関わらず、保守作業時に誤って無停電電源装置から電源を供給する設定にし、バッテリーを使い果たしたため。 担当者は電源系統の構成を知らず、保守作業用の手順書等もなかった。

（2）データセンターで突然の停電（施設管理者の怠慢＋システム技術者の無知）

インターネットプロバイダＡ社のデータセンターで全サーバの電源が突然止まった。

---

[26] ネットワーク中継器の一種。ISO モデルの第 3 層（データパケット）を切替える装置。

[27] ハードウェアの障害を検出し、正常な予備系に切り替える機能を果たすソフトウェアが設計上の考慮漏れで欠けていた。携帯電話に限らず、電話会社の交換機器の一部障害時に予備系への切替処理ソフトの不具合で大規模（広域、長時間）障害となる事故は昔から繰り返されている。

業務量の増大に合わせてサーバを次々に増設してきたため、ついに電力消費量が限度を超えたため。センターの管理者は電源容量のことを全く意識していなかった。

（3）操作記録ファイルのパンク（インテグレーターの怠慢、運用管理者の無知と怠慢）

クレジット会社B社で、稼働間もない新システムのコンソール操作記録ファイルがパンクし、営業システムが停止した。全体の容量は十分にあるのに、システムインテグレータの開発者が個別容量の上限設定を忘れ、運用者もチェックしなかったため。

（4）（将来）高潮／津波で水没確実（施設管理者の無知と怠慢、経営者の無責任）

グローバル企業C社の名古屋データセンターは、伊勢湾台風で水没し、ハザードマップでも浸水予想地域にある。堤防にかこまれ、すぐ隣にはポンプ所がある場所。災害に無関心な管理者が安易に遊休地を利用したためだが、 経営者も無責任である。

2.6　事前対策による業務継続

（1） バックアップセンター[28]

1989年10月に発生したサンフランシスコの地震[29]で、大手通販会社の受注システムが停止した。しかし、かねてから災害時に備えて大手バックアップセンターと契約していたので、最新データの磁気テープと電話オペレータ 200 人を飛行機でシカゴに移し、オヘア空港隣接のバックアップセンターでシステムを稼働させ業務を再開した。

（2） クラスタ構成による相互バックアップ

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、鎮火するまでの 12 時間に 2 フロアを全焼し、金融商品や外国為替取引のシステムとディーリングルームが焼失した。しかしブリュッセル支店とロンドン支店にも同じシステムが設置されており、パリ本店のデータは相互バックアップ[30]されていたので、担当者が飛行機や鉄道[31]でロンドンとブリュッセルに移動し、翌日から平常通りの業務を再開した。

---

[28] 同時に災害に逢う可能性の小さい遠隔地の共用コンピュータ設備を複数の顧客企業が契約し、ソフトウェアと最新の業務データを保管しておき、災害時にシステム機能を代替稼働させる施設。

[29] 正式名称は断層名によるロマ・プリータ地震。1909年のサンフランシスコ地震と区別するため。

[30] 信号伝達に時間のかかる遠隔地でもデータ更新を完全に同期させることができるコンチテンタルクラスタ方式と呼ばれた技術を本格的に採用した初期の事例。

[31] 当時、ドーバー海峡海底トンネルは未開通だった。

（３）業務継続計画[32]（BCP）

2001年9月11日に発生したアメリカの同時多発テロ事件では、崩壊したNY世界貿易センターに入居していた多くの企業が壊滅的な被害を受けた。しかし、米国の大手金融機関はバックアップセンターを含む業務継続計画を義務付けられているため、別な場所で代替要員により業務を継続できた。しかし中小企業はそうした備えが無かったため、貴重な人材だけでなく重要な業務データも喪失し、多くが倒産したといわれている。

2.7　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逆噴射しながら月面に降下中、着陸予定地点の地形に問題があることがわかり着陸地点を変更したら、飛行制御システムがオーバーフローした。燃料が残り少なかったので、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させた。それ以降、NASAの研究所の廊下には「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である。また、NASAのOBが設立したAbacus Technology （算盤技術！）という関連企業の製品である解析ソフトAbacusは多方面で使われている。

3　ソフトウェア品質の向上策

これまでのシステム工学的視点とソフトウェアの特性、紹介したシステム障害の事例を踏まえ、以下に航空システムのソフトウェア品質を向上させる対策を提案する。

3.1　適切な業務／システム要件の設定

設計に先立ち、システム化する業務の分析や業務プロセスの抜本的な見直し[33]、業務の標準化など、システム以前の問題を整理し、解決しておく必要がある。また、どのようなシステムを開発し、業務上どのように利用するのかという業務要件／システム要件の設定のために、十分な経営資源を投入すべきである。この段階で時間と金を節約した結果、開発費用が膨らんでしまう場合が少なくないからである。

さらに、この段階で業務を熟知した利用者が積極的に関与する必要がある。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。そのためには、業務、人、モノ、金

---

[32] Business Continuity Program。米国では投資家保護を目的とするSarbanes-Oxley Act 2002：SOX法により、企業の業務継続計画が義務付けられている。

[33] Business Process Re-engineering：BPR

に決定権のある経営者／管理者など、組織トップが積極的に関与しなければならない。

3.2　斬新的改善改良／単純なシステム

新しいシステムを企画する際は、経営管理者も利用者も開発技術者も、せっかくの機会だからろ画期的、革新的なものを望みがちである。しかし、革新的なシステム計画は技術的にも運用的にも実現が難しい。期待した性能が出ない、予算オーバー、開発の遅延などにより失敗するリスクが非常に高いので注意が必要である。

システム開発では、経験を積み重ねながら実績に基づいて斬新的に改善／改良していく手法を採用することが望ましい。新技術／新製品の採用にはかなりの技術力が必要なので、最新技術の経験者の採用やコンサルタント契約など発注側も体制強化が必要である。最新のものは費用対効果が大きいが、技術的なリスクも大きいからである。

また、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、信頼性を低下させ、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を計画し、確実に稼働させることが望ましい。

3.3　例外的な事象の適切な想定

設計時にあらゆる例外事象を想定することは不可能だが、機器故障や誤入力、例外データ、操作ミスなど、過去の事例などを収集するだけでも起こりうる事態のかなりの部分がリストアップできる。災害対策などは費用の制約で完璧な対応ができないことが少なくないが、被害を減らす減災は可能である。過去に起きたことはまた起こる可能性が高いし、論理的に考えれば起こり得るリスクは漏らさず想定すべきである。

そのためには、対象業務を熟知した人と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幅広く検証する必要がある。技術は進んでも障害のパターンは昔から変わらないので、想定外というのは不勉強か怠慢であることが多い。

また、異常が発生した際の人間系との整合性をとっておくことも重要である。あらゆる異常事業に自動的に対応をする完璧なシステムを設計することは不可能である。また、例外処理を全ての条件の組合せで試験することも困難で、試験漏れが障害の原因になることが少なくないからである。

3.4　構造化設計と設計開発ツール

わが国では開発費を「人数×月数」で見積るため開発効率や品質を問われない受託開発が中心で、開発の効率と品質を高める構造化設計や設計開発ツールが使われないこと

が少なくない。設計手法やツールに対する理解を深め、積極的な採用が望ましい。

データ／プロセスのモデリング、モジュール化された構造設計、標準化などソフトウェア工学的な（合理的な）開発手法を積極的に活用することは、開発効率と品質の向上に非常に有効である。それによる処理効率の低下は、多くの場合、昨今のハードウェア性能向上でカバーすることができる。ただし、リアルタイムシステム、特に機器の重量や容積が制約される搭載型システムの場合は、開発効率や処理効率と処理性能や記憶容量などの必要資源との間で適切なトレードオフを行う必要がある。

また、処理論理[34]の検証とソースコードの自動生成を行う設計開発ツールや、特定の業務に特化した目的向専用言語を活用することも有効である。膨大な行数のソースコードを全て人間が書く方法では、ヒューマンエラーが避けられないからである。

さらに、できるだけ実績のある市販パッケージや稼働実績のあるプログラムモジュールを活用することも品質向上の観点で重要である。技術者は他人が作ったものを嫌い、すべてを自分で作りたがる傾向(Not Invented Here: NIH)が強いので、技術者の恣意的な設計（＝趣味）にゆだねてはいけない。

3.5　ライフサイクルの費用と効果

そのシステムを何年間使うのかという想定により、開発費などの初期費用と運用管理や保守などの維持費用の妥当性は変わる。開発費を節約した結果、後々の維持費がかさむ例が少なくないからである。また、古く高価なハードウェアを使った陳腐化したシステムを、多くの人手と費用をかけ延命するよりも、全体を最新のハードウェアとソフトウェア・パッケージや開発手法を活用して更新した方が経済的な場合もある。

また、技術の進歩や業務量の変動に対応できるよう、柔軟性のあるシステム投資計画を立てることも重要である。これまでに急激に進歩してきた情報通信技術だが、今後もまだまた機器の性能向上やコストの低下が予想されるので、あらかじめ対応策を考えておくべきであろう。税制上の耐用年数は実態に合わないので、陳腐化したシステムは廃棄する決断も必要である。金融派生商品取引のように業務の変化が大きい分野では、当初から短期間で使い捨てにすることを前提にしたシステム投資が行われている。

さらに、システム投資を計画する場合は費用対効果と障害リスク、特に損害額を的確に想定しておく必要がある。システム費用を節減し過ぎた結果、重大な障害を起こして業務上大きな直接被害や間接被害（機会損害や信用失墜など）を出すことが多い。

---

[34] プログラムにおける条件判断や演算処理などの論理の流れ（Algorithm）。

3.6　ソフトウェア品質とシステム投資

わが国の場合、業界の歴史的な経緯により独自仕様の手造りソフトウェアが多く、品質に関する問題が少なくない。コンピュータのメーカーが多く、ソフトウェアの互換性が無かったため、パッケージ市場が育たなかったからである。昨今ではオープン・システム市場が確立しているので、業務ソフトウェアは自社で開発しなければならないという常識を見直し、合理的なシステム投資をする必要がある。

また、ソフトウェアの品質保証が実質的には行われていないという問題もある。受託開発の場合、間違いがあっても瑕疵担保契約による補修義務があるのみで、システムの不具合による損害に関するシステムインテグレータの賠償責任は免責という契約しか行われない。さらに、システム障害のリスク評価がほとんど行われていないため、費用の節減が優先し、適切なシステム投資計画が行われない。保険制度も未整備である。

開発費が工数で見積もられ品質が評価されないため、十分な教育訓練を受けていない低単価の未熟な技術者が頭数（人数×期間）で勝負している実態があり、品質向上の経済的インセンティブがないことが最大の問題であろう。経営者の理解も不十分なので、金融など社会的に重要なシステムでも投資が不十分かつ不適切である。

おわりに

以上の議論を踏まえ、将来の航空システム像を探る研究と並行して、それらシステムを実現する際のシステム工学的な側面での研究課題として、次の事項を提案したい。

① 無限の信頼性は実現できず、高い信頼性を実現するにはコストがかかる。信頼性要件を適切に設定するための考え方
② 障害時の対応方法と機械系と人間系の役割分担について、費用とヒューマンファクターを考慮しつつ適切に設定するための考え方
③ 利用者の知識やスキルのレベルによってシステム設計は違うはず。システム要件としての利用者レベルの適切な想定と教育訓練カリキュラムの考え方
④ ソフトウェアの標準化は経済的なメリットだけでなく、品質向上にも非常に効果的だが、陳腐化の問題が避けられない。両者を適切に整合させる考え方。
⑤ 人間系を含め、相互に連携して業務運用できるインターオペラビリティ（相互運用性基盤）の条件と実現方法

【著者略歴】

１９４７年（昭和２２年）山形市生まれ。山形大学理学部物理学科卒業。運輸省航空保安職員研修所（現航空保安大学校）管制課程終了。運輸省東京航空交通管制部航空管制官、日本電気株式会社システムエンジニア（航空管制システム担当）、株式会社三菱総合研究所主任研究員（宇宙開発システム担当、計測システム室長、情報システム室長、戦略情報システム部長代理）、日本ヒューレット・パッカード株式会社シニアコンサルタント（コンサルティングビジネス推進室長、システム戦略コンサルタント、コンサルティング事業本部人材開発部長）を経て独立し、現在に至る。

【推奨図書】詳しく学びたい方のために

「システム工学　問題発見・解決の方法」 井上 雅裕／陳 新開／長谷川 浩志　著、
オーム社 (2011/9/14)。 ISBN-13: 978-4274210921

「システム工学（新世代工学シリーズ)」 田村 坦之 著
オーム社 (1999/02) 、ISBN-13: 978-4274131677

「情報理論（ちくま学芸文庫）」甘利俊一　著
筑摩書房 (2011/4/8) 、ISBN-13: 978-4480093585

「信頼性工学入門」 真壁 肇 編
日本規格協会; 新版 (2010/07)、ISBN-13: 978-4542503489

「入門 信頼性工学 － 確率・統計の信頼性への適用」 福井 泰好 著
森北出版株式会社 (2006/7/13) 、ISBN-13: 978-4627665712

「ソフトウェア工学（情報工学レクチャーシリーズ）」 高橋 直久／丸山 勝久 著
森北出版 (2010/8/12)、ISBN-13: 978-4627810617

「ソフトウェア社会のゆくえ」 玉井 哲雄　著
岩波書店 (2012/1/28)、ISBN-13: 978-4000056199

「これだけは理解しておきたい ソフトウェア開発の知識」 白井豊 著
ゆたか創造舎; 初版 (2008/12/25) 、ISBN-13: 978-4904551004

「ソフトウェア構造化技法－ダイアグラム法による」J.マーチン／C.マックルーア著、
国友義久／渡辺 純一 訳、近代科学社 (1986/11)、ISBN-13: 978-4764901247

「品質管理のための統計手法（日経文庫)」 永田 靖 著
日本経済新聞社 (2006/01) 、ISBN-13: 978-4532110895