解説/Review
特集：ディジタリゼーションにおけるシステムイノベーション

IT とシステム化

松本 隆明*

An Overview of IT Needed in the DX Era and Its Systematic Aspects

Takaaki MATSUMOTO*

Abstract– In the DX (Digital Transformation) era, the significance of IT, which underpins the socio-economic activities, will continue to increase. IT must be used as an engine for creating new smart services and businesses, instead of tools for improving business efficiency and reducing costs as in the past. For that purpose, it is necessary to grasp IT as an entire system. This paper describes an overview of IT needed in the DX era from the technical viewpoints of software, which is an indispensable component of IT.

Keywords– SoR (System of Record), SoE (System of Engagement), Software Engineering, SOA (Service-oriented Architecture), Microservices, Legacy Modernization, Agile Development, STAMP (Systems-Theoretic Accident Model and Process), System

1. はじめに

2019 年 10 月にガートナー社が発表したサーベイ結果 [1] によると，調査対象企業の 9 割において IT 部門は経営トップからビジネスの拡大に寄与していないと考えられており，IT 部門はビジネスのサポート役であってビジネスのリード役とはなっていないことが報告されている．同社では，IT がほとんどすべてのサービスやシステムのエンジンとなっている現代においてのバイモーダル IT の重要性を 2015 年から呼び掛けている．IT システムの活用は SoR（System of Record）と称されるように，業務内容の記録・保管・遂行を効率化・省力化することを主目的とする活用と，SoE（System of Engagement）と呼ぶ，システム同士や顧客との連携により付加価値を生み出すことを主目的とする活用の 2 つの捉え方があり，これまでは SoR を目指すケースが多かったが，今後は SoE を目指すことも考えていく必要があるということを強調している．

わが国では，SoR を「守りの IT」，SoE を「攻めの IT」と呼び，企業における IT 活用の目的を，社内の業務効率化やコスト削減を中心とした「守り」だけでなく，新たな製品やサービスの展開やビジネスモデル変革を通じた付加価値の創出や競争力の強化を目的とする「攻め」の活用にも生かしていくことの必要性が叫ばれてきた．わが国においては，これまで「守りの IT」の強化に主眼が置かれるケースが多かったこともあり，経済産業省と東京証券取引所が 2015 年から毎年，東京証券取引所の上場企業の中から「攻めの IT」に積極的に取り組んでいる企業を「攻めの IT 経営銘柄」として選定し，企業における「攻めの IT」の推進を推奨する取り組みを行っている [2].

このように「攻めの IT」の活用を増やしていくことの必要性が何年も前から叫ばれているにもかかわらず，冒頭のサーベイ結果にあるように，現在でも依然として IT がビジネスの推進役になっていないのはなぜであろうか．急激な環境変化への迅速な対応が求められる DX（デジタルトランスフォーメーション）時代においてより求められる「攻めの IT」に変えていくためには，IT をシステム的に俯瞰的にとらえるとともに，ビジネスモデルの変革や組織改革，さらには企業風土の刷新など経営的な側面も含めて全体最適を実現する手段として考えていく必要がある．本稿では，特にソフトウェアを中心とした技術的な側面から「攻めの IT」への変革について概観してみたい．

*独立行政法人情報処理推進機構　東京都文京区本駒込 2-28-8 文京グリーンコートセンタオフィス
*Information-technology Promotion Agency, Japan, Bunkyo Green Court Center Office 2-28-8 Honkomagome, Bunkyo-ku, Tokyo

Received: 27 February 2020, Accepted: 25 March 2020.

2. IT 開発の難しさ

IT に限らず，様々な分野におけるソフトウェアの役割の増大とその重要性は今さら述べるまでもないことである．NCSA Mosaic の開発者として有名なマーク・アンドルーセンが 2011 年の 8 月に The Wall Street Journal に寄稿した “Why Software Is Eating The World”という記事は当時のソフトウェアの急速な席巻ぶりを端的に表しており，記事ではこれからはソフトウェアをベースとする企業が世界のビジネスを制していくだろうと述べられている．

ハードウェアも組込みソフトウェアと呼ばれる内蔵ソフトウェアで制御されるケースがほとんどである．2015 年 3 月に，家庭用の液晶テレビが深夜から翌日の昼ごろまで電源のオン・オフを繰り返すというトラブルが発生した．トラブル対象機種数は 162 万台に達し，多くの利用者が不気味な思いを経験することとなった．トラブルの原因は，一般放送波に乗せて送られる特定放送データを受け取るテレビ側のソフトウェアの不具合で，後日修正ソフトウェアが送信されて自動更新により修正された．本事象は，一般の利用者にとっても，改めて家電製品がソフトウェアによって制御されていることを思い知らされた出来事である．

このようにソフトウェアは中心的役割を果たすようになってきたが，ソフトウェアの開発には一般の工業製品の開発とは異なる，例えば以下のような難しさが存在する．

- ソフトウェアは可視化できず，その開発の進捗状況も直接的に目に見えない
- 人間の知的著作物であるため，作成者の能力に大きく依存する
- 人間同士で連携して開発する必要があるため，開発プロジェクトのメンバ間の密なコミュニケーションが必要である
- ソフトウェアの不具合は，ハードウェアのように冗長化することで防ぐことはできない
- 設計仕様を正しく満たすソフトウェアは無数に存在する
- 開発工程（要件定義，基本設計，詳細設計，製造，テスト）の分割に任意性が高い

こうした難しさは，システムを構成するソフトウェアの規模が急速に増大しつつあった 1960 年代後半からすでに認識されており，「ソフトウェア危機」という言葉が叫ばれ，それに対処すべくソフトウェア・エンジニアリングの考え方が急速に広まった．ソフトウェア・エンジニアリングにおけるバイブルとも言われる，1975 年のフレデリック・ブルックス著「人月の神話」[3] では，遅れているソフトウェア開発プロジェクトに要員を投入してもさらに開発プロジェクトが遅れるだけであるというブルックスの法則が述べられている．人月の神話という意味は，「人」と「月」は等価ではない，すなわち人（要員）を増やすことは月（工期）を減らすことには必ずしもつながらないということを表している．10 人で 10 か月かかる 100 人月のソフトウェア開発プロジェクトを，20 人でやれば 5 ヶ月で完了するかというとそうではないということである．

IPA（独立行政法人情報処理推進機構）では，ソフトウェア・エンジニアリング推進の取り組みの一環として，ソフトウェア開発現場における定量的プロジェクト管理を推奨するため「ソフトウェア開発データ白書」[4] を定期的に刊行しているが，それによると工数と工期の関係は比例関係ではなく，工期が長くなると工数は指数関数的に増加することが示されている．ブルックスの法則が半世紀近く経った現在でも成り立っていることを示唆している．また，ソフトウェア開発においては要件定義などの上流工程の出来の良し悪しがプロジェクトの成否を大きく左右する．ちなみに，要件定義工程は，事業戦略・事業計画の策定やシステム化計画などと合わせて上流よりさらに上の超上流工程と呼ぶこともある．

要件定義工程では，ソフトウェアで実現する業務の内容を要件として整理するが，下流の製造工程とは異なる以下のような難しさを内在している．

- 要件の策定にあたっては，業務部門と IT 部門，さらには開発を外部に委託する場合には開発ベンダなど様々なバックグラウンドを持つステークホルダ間で意思疎通を図り，合意を形成する必要がある
- 要件の記述方法が形式化されていない場合が多い（日本語で記述する場合が多い）ため，ステークホルダ間で要件の内容理解に齟齬が生じやすい
- 要求と要件は異なるため，現場から出された様々な要求は実現性やコスト等の観点から優先度付けを行って要件としてまとめられる．この際見送りとなった要求については運用での対応策まで含めて予め設計しておく必要がある
- 本来上流工程では時間をかけて要求分析や要件整理を行うべきであるが，全体の工期の制約から往々にしてなるべく早く設計・製造工程に移行しがちで，上流工程に十分な検討期間が取れず曖昧なまま進めてしまうケースが多い

ソフトウェア開発の持つこうした本質的な難しさに対応すべく，各種の開発方法論や CASE（Computer Aided

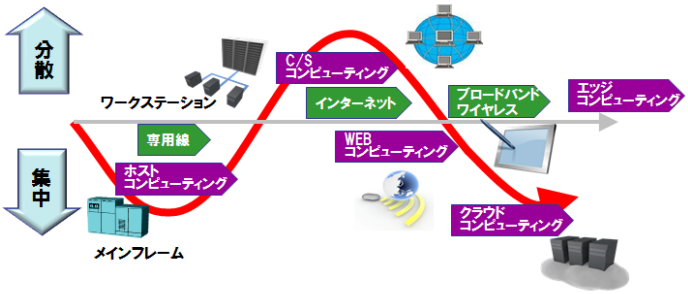

Fig. 1: Transition of IT Architecture.

Software Engineering）ツールの整備などソフトウェア・エンジニアリングも進化を遂げてきたが，より複雑化・大規模化する IT システムによって，IT システムの障害件数は残念ながら年々増加の傾向にある．IPA の調査 [5] によれば，新聞等で報道された社会生活に大きな影響を与えた IT システムの障害件数は，2019 年 1 月から 6 月までの半年で，金融機関のシステム障害や決済サービスの障害など 33 件発生し，これはすでに 2018 年の 1 年間分の件数と同数である．2019 年は改元に伴う障害が多く発生したという特殊な事情もあるが，障害件数の増加傾向は 2009 年以降続いている．

いずれにしても，これまではソフトウェアを中心とした IT システムの QCD（Quality，Cost，Delivery）をいかに高めるかが IT にとって最も重要な課題であり，この QCD を高めるという「守りの IT」の高度化に多大な労力をかけてきた．こうした努力により，わが国は QCD の向上という面では米国などに比べて優位性を保つことができた [6]．しかしながら，QCD に対する過剰な期待や要求，そしてそれに対する過大な投資がゆえに，急激に環境変化が起きつつある現代においては，それがかえって足かせとなり，冒頭述べたようにわが国では IT がビジネスの推進役になりきれなくなっている要因の一つとなっているのではないだろうか．

3. IT を取り巻く環境の変化

IT アーキテクチャのこれまでの変遷を振り返ると，**Fig. 1** に示すように集中と分散の形態を繰り返していることに気が付く．

当初，IT システムはメインフレームコンピュータに専用線を介して入力と表示機能のみを有する簡易な端末（ダム端末）がつながる集中型のシステム形態であった．その後，トランザクション量の増大に対応すべく端末の機能が向上し，ワークステーションとしてインテリジェント化することでクライアント・サーバ（C/S）コンピューティングの分散型に移行した．1990 年代後半に入り，インターネットの爆発的な普及，オープンソースの普及，仮想化技術の高度化などによってオープンコンピューティングが主流となり，仮想空間上での相互連携がより求められるようになることでクラウドコンピューティングによる集中型が主流となった．さらに，今後は IoT の進展により，リアルタイム性を求められる単純な処理はエッジ側で行うというエッジコンピューティングの分散型にまた進んでいくものと想定される．ハードウェアの進歩やネットワークの高度化により，歴史は繰り返しているが，大きな転換点は IT システムが単独で動作していた時代から，連携して複雑に動作する時代になってきたという点であろう．

クラウドコンピューティングの幅広い普及によって，IT システムを構成する業務アプリケーションの開発も様々な方法で行われるようになった．特に最近注目されているのがマイクロサービスという考え方である．マイクロサービスとは，業務アプリケーションを複数のサービスと呼ばれる機能単位に分割して，それらのサービスをクラウド上で連携させて全体の業務を実現するという方法である．

ソフトウェアの部品化やモジュール化の考え方は 1980 年代からある考え方であるが，部品に分割する基準が曖昧で粒度もバラバラとなりコードベースの分割に陥りがちなため，これまで部品化・再利用化という取り組みはあまり広がりを見せてこなかった．しかし，IT システムの肥大化に伴う維持管理コストの増大やビジネス環境変化への迅速な対応がより求められるようになり，2000 年代中ごろからサービス指向アーキテクチャ（Service-Oriented Architecture：SOA）の考え方が広まってきた．マイクロサービスは，構造中心でなくサービス視点で考えるというサービス指向アーキテクチャの 1 つの形態と考えることができ，さらには最近ではコンテナや REST API（REpresentational State Transfer API）など実装面での環境も整いつつあり，幅広い分野で普及が進みつつある．

マイクロサービスを効果的に活用するためには，業務プロセスを形式知化し，適切な単位でサービスとして分割し，なるべく汎用化することが重要となる．しかしながら，わが国の多くの業務ソフトウェアでは現場最適に対応した暗黙知が数多く存在するケースが多い．2018 年の調査によれば，わが国における ERP（Enterprise Resource Planning）パッケージソフトウェアの普及率は，財務会計や人事給与といった業務では 50 %程度で，販売管理や在庫管理といった業務では 30 %程度とのことである．ERP パッケージを導入しようとしても，現行の業務が汎用化されておらずに暗黙知が多く，カスタマイズ部分が膨大となるため，依然として多くの業務が専

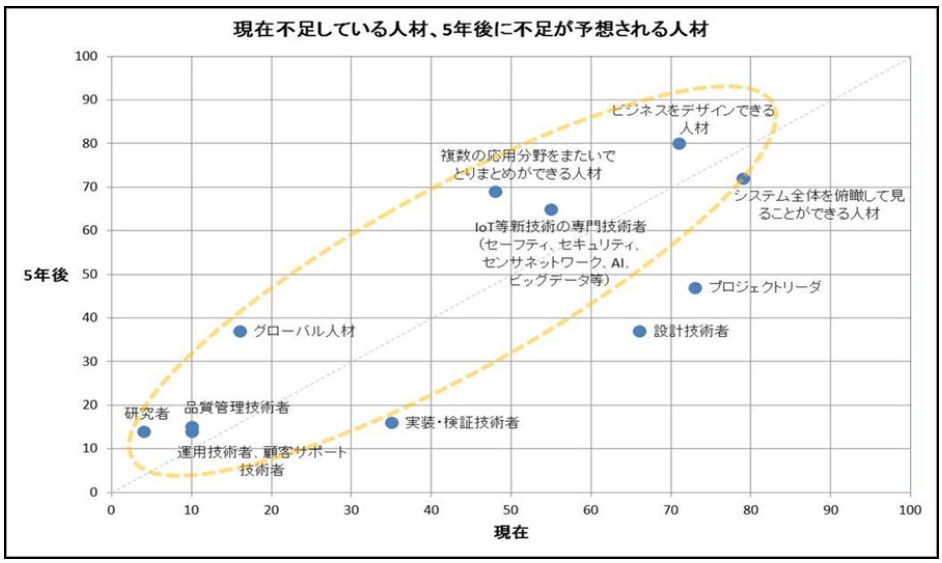

Fig. 2: Lack of Talent in Embedded Software Industry （出典：2016 年度組込みソフトウェア産業の動向把握等に関する調査，IPA）．

用の業務ソフトウェアで実装されることとなる．もちろん，専用のソフトウェアにより顧客にとっては画一的でない融通の利いたサービスが受けられるという利点があり，またサービス提供企業にとっても現場の創意工夫が直接的に生かせるという利点もあるが，「攻めの IT」に変えていくという点ではかえって障壁となってしまう可能性がある．

一方，海外では，むしろ ERP パッケージに合わせて業務のやり方を変えていくそうであるが，わが国では現場のやり方を変えることに依然として相当抵抗があるようである．

マイクロサービスのように分散化されたサービスでは，性能，信頼性，セキュリティ，運用性などの非機能に対する考慮がより重要となる．機能は分散化されても，非機能はサービス全体で設計する必要がある．例えば，業務のクリティカル度に応じて，連携する機能全体のトランザクション制御やデータベースのコミットメント制御を行うモジュールを作り込んでおくことも考えておかなければならない．モノリシックアーキテクチャから疎結合化され機能やデータが分散化されるマイクロサービスアーキテクチャになることで，非機能に関する部分はシステム的に全体俯瞰でとらえることがより求められるようになる．

ちなみに，全体を俯瞰的に見ることの必要性は，組込みソフトウェア産業界でも高まりつつある．組込みソフトウェアは各種の製品や機器に組み込まれて制御を行うソフトウェアであるが，**Fig. 2** に示す IPA の調査 [7] によれば，その開発にあたっても，組み込まれる製品や機器に関する知識や個々の設計技術や実装技術，あるいはプロジェクトマネージメントといった能力を有する人材よりも，システム全体を俯瞰できる人材やビジネスをデザインできる人材の方が不足しており，今後も不足が予想されるという報告がなされている．組込みソフトウェアの開発も，組み込まれる機器に適した部分最適の開発から，現在では，システムとして捉える全体最適で考える開発にシフトしつつある．

4. 攻めの IT 中心に変えていくためには

「守りの IT」中心から「攻めの IT」中心に変えていくために何が必要になってくるのか．本稿では，特に以下に述べる 3 つの視点から見てみたい．

4.1 レガシーシステム

JUAS（一般社団法人日本情報システム・ユーザ協会）の「企業 IT 動向調査報告書 2016」[8] によれば，技術面で老朽化した基幹系システムが半数以上あると答えた企業は全体の 31.1 %，システムの肥大化・複雑化を抱えた基幹システムが半数以上あると答えた企業は 35.6 %，ブラックボックス化した基幹システムが半数以上あると答えた企業は 27.8 %であると報告されている．約 3 割の企業が何らかの形のレガシーシステムを半数以上保有していることになる．「攻めの IT」中心に変えていく上では，こうしたレガシーシステムのモダナイゼーションが大きな足かせとなる可能性がある．

経済産業省が 2018 年 9 月に発表した，DX レポート～IT システム「2025 年の壁」克服と DX の本格的な展開～[9] においても，DX 推進のためには複雑化・ブラッ

クボックス化したレガシーシステムへの対応が急務であり，この課題が克服できなければ 2025 年以降最大年 12 兆円の経済損失が発生する可能性がある（2025 年の壁）としている．2025 年の壁の克服に向けて，既存システムの見える化指標や DX 推進システムガイドラインの策定などの取組みを今後実施していくとしている．

企業活動を支える IT システムは，すべてが「攻めの IT」となるわけではなく「守りの IT」も当然必要となる．レガシーシステムの問題では，「守りの IT」として存続させるシステムを効率よくまた正しく再構築して維持運用コストを削減し，その分を「攻めの IT」の投資に回すことが重要である．その意味で，再構築はモダナイゼーションの活動の中の 1 つと位置づけられる．

IT システムのライフサイクルは平均的に 10 年から 15 年程度と言われており，IT システムの再構築はこれまでも幾度となく行われてきたことであるが，最近は AI や IoT，あるいは 5G（第 5 世代移動通信システム）による高速通信など IT を取り巻く環境の急激な変化や過去の技術を持つエンジニアのリタイアにより，再構築に伴うリスクはより高まりつつある．

IT システムの再構築を行う上での最大のリスク要因は現行業務知識の不足であり，現行踏襲の問題や品質保証の問題を引き起こす [10]．現行踏襲では，そもそも現行とは何かがステークホルダ間で正しく共有されていないことにより，再構築後に齟齬が生じるケースが多い．また，設計書やソースコードが正しくメンテナンスされていない場合などでは，既存の設計書に基づいて再構築を行うと現行とは違ったシステムが出来てしまうといった問題も発生する．品質保証に関しては，業務知識の不足により，テスト項目の抽出やテスト範囲の設定などが不十分となり品質問題を引き起こすといったことも生じる．

文献 [10] では，再構築時に注意すべきポイントとして，以下の 7 つの落とし穴に気を付けるべきであるとしている．

- 「再構築だから」と企画・要件定義フェーズを軽視していないか
- 「今と同じ」という要件定義になっていないか
- 現行システムの調査が「表面的」になっていないか
- 業務部門はメンバの一員として上流工程から参画しているか
- 現行システムが動いているからといって，品質保証を簡単に考えていないか
- 担保すべき「業務継続性」は明確になっているか
- モダナイゼーションのリスクを甘く見ていないか

いずれにしても，DX レポートにも記載されているように，既存システムの見える化やモデル化，分析を十分行って，現行システムの全体像を明らかにしておくことが求められる．システム全体を俯瞰的にとらえるシステム思考の考え方がレガシー対策においても極めて重要となってくる．

4.2 アジャイル型の開発[1]

現代は VUCA（Volatility，Uncertainty，Complexity，Ambiguity）の時代であると言われるように，予測不能で不確実な社会になりつつある．IT システムにとっても，一部のミッションクリティカルなシステムを除けば，これまでのようにあらかじめ上流工程において要件をきちんと定義してから開発を進めるやり方だけでは通用しなくなる．ウォーターフォール型の開発からアジャイル型の開発へのシフトが急速に進むであろう．

アジャイル型の開発の成否のカギを握るのはコミュニケーションである．ユーザ企業がベンダ企業に IT システムの開発をアジャイル型で委託する場合には，ユーザ企業とベンダ企業との間で密接なコミュニケーションが無ければ，継続的な評価が行えず，最終成果物のイメージがお互いに全く違ったものとなってしまうリスクが大きくなる．

IPA の IT 人材白書 2017[11] によれば，IT エンジニアの所属先企業は，日本では 72 %が IT 企業であるのに対して，米国では 34.6 %が IT 企業で残りはユーザ企業であると報告されている．日本の場合には，ユーザ企業側に IT エンジニアが少ないためにアジャイル型で相互にコミュニケーションをとろうとしても，技術的な側面については十分な意思疎通が図れないといった懸念が想定される．日本においても今後は人材の流動性を高め，ユーザ企業側に IT エンジニアを増やすことによって，よりユーザ企業が開発に関与でき，相互に密なコミュニケーションが図れるようにしていく必要がある．

Web アプリケーション開発では，部品を組み立てて自動的に開発する超高速開発が注目され，そのためのツールも数多く提供されるようになってきた．こうした流れも開発にユーザ企業が関与しやすくすることが大きな目的である．

ユーザ企業内においても業務部門と IT 部門間の風通しを良くするとともに，トップ層の強力なリーダーシップのもとに目指すべきシステムのビジョンを明確にして進めるべきである．単に業務部門と IT 部門が協調して進めればよいと安易に考えるのではなく，組織改革も含めてシステム開発に取り組んでいくことが重要である．

1. アジャイル開発と言うと，通常はスクラムやエクストリーム・プログラミング（XP）などの開発手法の総称を指す場合が多い．ここでは開発手法に特定せず，幅広くブラッシュアップをしながら開発を進めていくやり方という意味で，アジャイル型の開発と呼ぶことにする．

アジャイル型になればなるほど，利用者視点に立って目指すべきサービスやシステムのあるべき姿を明確にして進めないと，永遠に設計と開発のサイクルを繰り返してしまうリスクが生じると認識しておくべきである．

ベンダ企業側も設計方法の共通化に取り組む必要がある．現状，設計書の記述方法やプログラムのコーディング規約など開発に関わるいわゆる「お作法」ともいうべき細かなルールは，ベンダ企業個別に社内標準を定めて適用している場合が多い．このため，ベンダごとに表現が異なり，コミュニケーションに齟齬を引き起こす要因ともなり兼ねず，またベンダ間での人材流動を阻害することにもつながる．こうしたルールの業界標準化をさらに進める必要がある．

ユーザ企業とベンダ企業間での開発に関わる契約の形態も，現状は確定請負契約で結ぶ場合が多いが，要件が曖昧なまま開発を委託するケースが増えることから，あらかじめ契約額を定める確定契約ではなく，準委任契約のような仕事量に応じた報酬や，あるいは最終成果物の良し悪しで価値ベースの報酬とするなどの開発契約の仕方の見直しも今後は求められていくであろう．

ちなみに，契約に関しては，2020 年 4 月に施行された民法改正によってこれまで使われていた瑕疵という表現は契約不適合という表現に改められた．表現が変わっても本質的には変わらないという意見もあるが，ソフトウェアの場合には，不具合を全く生じさせないことはほとんど困難である．従来のように曖昧になりがちな瑕疵の責任を瑕疵担保責任としてベンダ企業に全て負わせるのではなく，契約内容に合わない部分についてベンダ企業が責任を持つという方向に変わることになる．今般の改正により，ベンダが責任を負うべき期間が延び，開発や維持管理のコスト増につながるという懸念もあるにはあるが，契約に基づいたユーザ企業とベンダ企業間の責任の所在がより明確になり，両者の協力がしやすくなるものと期待したい．

4.3 安全性の確保

システムが非常に多くのコンポーネントから構成され複雑さが増すと，単純な構成では容易に確認できたシステム特性を説明することが非常に難しくなってくる．特に，安全性，性能，セキュリティなどのいわゆる非機能要件は，システム全体で検証し説明できなければならない．ここでは，これらのうち複雑化するシステムで求められていく安全性の確保について考えてみる．

1990 年代には高度化したシステムと人間との関与に関連した事故が数多く発生した．1994 年 3 月に航空機がシベリアに墜落し，乗客乗員 75 名全員が亡くなるという事故が起きた．直接の原因は，機長が自分の息子を操縦席に座らせて操縦かんを操作させたことであり，機長は自動操縦状態であるために操縦かんを操作しても問題ないと認識していた．しかしながら自動操縦システムには，フェールセーフのため，操縦かんを 30 秒以上操作し続けると自動操縦の一部が自動的に解除されるという機能があり，これにより知らないうちに自動操縦が解除され墜落に至った．

自動操縦システムとしては，万が一自動操縦が手動で解除できなくなった場合を考慮したフェールセーフという極めて高信頼なシステムになっていたにも関わらず，人間の関与への考慮が不足していた．さらにこの 1 か月後には空港への着陸中に航空機が墜落するという事故も発生し，これも自動操縦と人間の操作のどちらを優先するかによることが事故の原因と言われており，安全設計への人的要素の取り込みの必要性が高まることとなった．

20 世紀までのシステムの安全の考え方は，事故には機器の故障や人間の誤操作などの根本原因があって起こるもので，こうした根本原因を究明し排除するため FTA（Fault Tree Analysis）や FMEA（Fault Mode and Effect Analysis）といった安全性解析手法が用いられてきた．基本的には，システムを構成するコンポーネントの信頼度を上げ，根本原因が生じないようにすればシステム全体としての信頼性も上がるという発想である．これに対し，21 世紀に入ると人間も含めてコンポーネントが相互に連携して複雑に動作するようになり，個々のコンポーネントがいくら高信頼でもシステム全体の安全には必ずしもつながらないようになってきた．

マサチューセッツ工科大学のナンシー・レブソンは，システムの安全性はコンポーネント同士の相互作用から創発されるものであるという考え方に基づいた STAMP（Systems-Theoretic Accident Model and Process）を提唱している [12]．システムの障害は，システムの中で安全のための制御を行う Controller と制御される Controlled Process との間の相互作用（コントロールアクション）がうまく働かないことにより発生するとして，相互作用を記述した構造図（Control Structure）を作成してモデル化することが安全設計上重要であるとしている．もちろん STAMP モデルでは人間も一つの Controller や Controlled Process として扱われる．

STAMP のように，事故は単なるコンポーネントの故障で起きるものではなく，コンポーネント間の相互連携が安全制約に違反することにより起こるという考え方は，まさにより複雑化するシステムにおいては，システム的に全体をとらえて安全設計を行わなければいけないということを示している．STAMP の詳細については，文献 [13] を参照されたい．

また，様々なドメインのコンポーネントが相互に連携して動作する場合，信頼性や品質に関する考え方や物差しがドメインごとに異なることが想定される．シス

テムやソフトウェアの品質を規定する国際規格 SQuaRE（Systems and software Quality Requirements and Evaluation：ISO/IEC 25000 シリーズ）などに基づいて物差し合わせを行ってから全体の品質検証を行うことも重要である．

5. システム化に向けた政府の取組み

政府は，2019 年 10 月 15 日に「情報処理の促進に関する法律の一部を改正する法律案」を閣議決定し，その後国会での承認を経て，12 月 6 日公布された [14]．本改正は，Society 5.0 の実現を目指して，官民双方で取り組むべき以下の措置事項を規定している．

- 企業のデジタル経営改革
  レガシーシステムからの脱却を目指し，戦略的なシステムの活用を実現するための指針を国が策定し，それに基づいて優良企業の選定を行う
- 産業の基盤づくり
  組織を跨ってデータを連携・共有するために必要な共通の技術仕様（アーキテクチャ）を設計し，専門家の集約育成を行う組織（産業アーキテクチャ・デザインセンター（仮称））を IPA に設置する
- 安全性の確保
  政府調達におけるクラウドサービスの安全性評価を行う機能を IPA に追加するとともに，情報処理安全確保支援士の登録を更新制とする

このうち，産業アーキテクチャ・デザインセンター（仮称）が具備する主な機能としては，政府や事業者からの依頼を受けた重要分野のアーキテクチャ設計，アーキテクチャ設計にかかわるプロセスや手法の確立，アーキテクチャ人材の育成などが想定されている．

いずれにしても，これまで政府が提唱してきた Connected Industries の実現に向けて，その仕組み作りに国として支援を行うことで，システム化に向けた具体的な社会実装が始まるものと期待できる．一般社団法人システムイノベーションセンターや産業アーキテクチャ・デザインセンター（仮称）とで密接に連携してシステム化の推進に向けた取り組みを進めていくことが今後求められていくことになろう．

6. おわりに

DX 時代の IT は，企業活動を効率化するための道具である「守りの IT」から，新たなビジネスモデルや付加価値を生み出すエンジンとしての「攻めの IT」へと変貌していく必要があり，IT 開発に内在する様々な課題を克服して，サイバー空間とリアル空間の融合の高度化を実現させていかなければならない．「攻めの IT」への変革のためには、これまで「守りの IT」として企業活動を支えてきたレガシーシステムの全体像をシステムとして俯瞰的にとらえて視覚化し全体最適の視点から刷新していくモダナイゼーションへの取組み、アジャイル型の開発形態が増えることに対応したより目的指向の IT 開発への取組み、そして安全性などの非機能に対してはシステム思考に基づいた設計方法への取組みが今後特に重要となる．なかなか進まない「攻めの IT」へシフトしていくためにも IT のシステム化への取り組みを強化していくことがより一層求められる．

参考文献

[1] ガートナージャパン株式会社，プレスリリース https://www.gartner.com/jp/newsroom/press-releases/pr-20191023

[2] 経済産業省，攻めの IT 経営銘柄 https://www.meti.go.jp/policy/it_policy/investment/keiei_meigara/keiei_meigara.html

[3] フレデリック・P・ブルックス Jr.：滝沢 徹，富沢 昇，牧野 祐子 翻訳，「人月の神話 – 狼人間を撃つ銀の弾はない」，丸善出版 (2014).

[4] 独立行政法人情報処理推進機構，ソフトウェア開発データ白書 2018-2019 https://www.ipa.go.jp/sec/publish/tn12-002.html

[5] 独立行政法人情報処理推進機構，情報システムの障害状況 2019 年前半データ https://www.ipa.go.jp/files/000077486.pdf

[6] マイケル A. クスマノ，サイコム・インターナショナル 監訳，「ソフトウェア企業の競争戦略」，ダイヤモンド社 (2004).

[7] 独立行政法人情報処理推進機構，2016 年度 組込みソフトウェア産業の動向把握等に関する調査 http://www.ipa.go.jp/sec/reports/20170502.html

[8] 一般社団法人日本情報システム・ユーザ協会，企業 IT 動向調査報告書 2016 https://www.juas.or.jp/cms/media/2017/02/16itdoukou.pdf

[9] 経済産業省，DX レポート～IT システム「2025 年の壁」克服と DX の本格的な展開～https://www.meti.go.jp/shingikai/mono_info_service/digital_transformation/20180907_report.html

[10] 独立行政法人情報処理推進機構，システム再構築を成功に導くユーザガイド～デジタル変革に向けた IT モダナイゼーション企画のポイント集 https://www.ipa.go.jp/sec/publish/tn16-009.html

[11] 独立行政法人情報処理推進機構，IT 人材白書 2017 https://www.ipa.go.jp/files/000059086.pdf

[12] Nancy G. Leveson，“Engineering a Safer World: Systems Thinking Applied to Safety (Engineering Systems)”，The MIT Press (2012).

[13] 独立行政法人情報処理推進機構，はじめての STAMP/STPA ～システム思考に基づく新しい安全性解析手法～https://www.ipa.go.jp/sec/reports/20160428.html

[14] 経済産業省，ニュースリリース https://www.meti.go.jp/press/2019/10/20191015002/20191015002.html

松本 隆明

1978 年東京工業大学大学院理工学研究科修士課程修了. 同年日本電信電話公社（現 NTT）に入社. その後 NTT データ技術開発本部長, NTT データ先端技術常務取締役を経て, 2012 年より独立行政法人情報処理推進機構ソフトウェアエンジニアリングセンター所長, 現在同顧問. 博士（工学）.