多様化が進むソフトウェア開発スタイル
～サスティナブルなデジタルビジネスのために～

SEC システムグループリーダー
山下　博之

SEC 調査役
室　修治

開発スピードや品質などに関する要求の高まりに応えるために、ソフトウェア開発の現場では様々な開発スタイルが採用されている。本稿では、IPA/SEC の調査・検討の成果や内外の公開調査結果などを通して、このような状況を概観すると共に、今後ますます進展する IoT 時代に向けた課題について考える。

1 ICT の進展とシステム開発

コンピュータがこの世に出現して以降、当初は単なる業務支援のために使われていた情報通信技術（ICT）は、図 1 に示すように、その技術自身の進展と時代の要請とが相まって、企業の基幹業務や社会生活のコモディティなどへと、その利用が質・量ともに拡大してきている。ICT の持つ優れた能力を認めた私たちは、ビジネス戦略の武器や社会サービス、生活必需品などへの利用と、ますます ICT への依存を高めている。このような流れの中で、ICT の中核をなすソフトウェアはその重要性が増し、ソフトウェア開発への要求事項もまた変遷してきた。業務支援システムに使われ始めた頃は、品質、とくに信頼性や操作性が厳しく評価されていた。その後、ビジネス戦略の要として利用されるようになると、競争優位を維持するために開発スピードが重視されるようになってきた [1]。そして最近では、いわゆる QCD（品質、コスト、納期）の目標達成よりも、ビジネスに価値をもたらすことがソフトウェア開発プロジェクトの成功を意味すると考えられるようになってきている [2]。

ただし、ソフトウェア開発への要求は画一的なものではなく、ICT システムの対象やそれを使うビジネスの状況に応じ、重視する評価ポイントは異なる。例えば、システム種別の観点では、企業の基幹系システムでは品質や継承性が、Web・フロント系システムでは開発スピードや変更容易性が、それぞれ重視されるという調査結果が報告されている [3]。また、ビジネス・ステージの観点では、黎明期には短い間隔で新サービスを継続的にリリースして激しい競争を勝ち抜くための開発が、安定期には利用者の評価に対応した更新版を定期的にリリースするための開発が、それぞれ求められる [4]。更に、IPA/SEC が公開したソフトウェア開発データの分析結果によれば、金融･保険業は他業種に比べ、生産性が低く信頼性が高い傾向が見られる [5]。

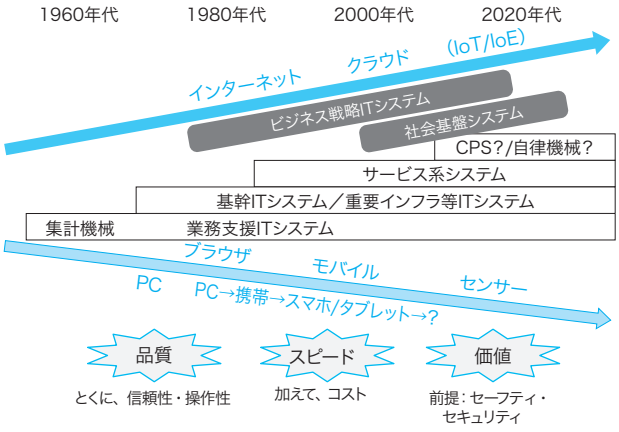

図 1　ICT 活用と ICT システムの変遷

2 ソフトウェア開発スタイル

ソフトウェア開発に対する要求の変遷に伴い、それに対応した開発スタイルの採用や各種技術の利用が行われてきている。システム全体の要求を確定した後に設計、製造、試験と順に開発を進める従来のウォーターフォール型開発に加えて、少しずつ作って確かめながら開発を進めるアジャイル開発、作らないで使うことを基本とするクラウド・コンピューティング、パラメータを変更するだけの自動コード生成やビジネスルールマネジメントシステム（BRMS）の利用などである。このように、ICT システムの対象やそれを使うビジネスの状況に応じ、最適な開発スタイルは異なる。もちろん、1 つのシステム全体を単一の手法で開発することが適切ではない場合もある。本特集では、それらの中から、アジャイル開発、リーン開発、超高速開発、派生開発を取り上げ、それらの動向についてそれぞれの専門家に寄稿していただいた記事を掲載する。

（1）アジャイル開発

アジャイル開発は、ビジネス環境の変化に対する俊敏なシステムの対応が求められる場合の開発手法である [6]。シ

ステム全体の機能を小さな単位に分割し、ビジネス要求に従って優先度の高い機能から短い周期で繰り返し開発し、順次リリースしながらシステム全体を構築していく。

（2）リーン開発

リーン開発は、トヨタ生産方式を源流とし、ソフトウェア製品を素早く提供するために、「ムダの徹底的な排除」や「品質の作り込み」、「全体を最適化する」などの原則に加え、現場で働く人々が考える環境の醸成、人間性尊重の概念も重要な基盤としている [7]。ただし、固有の手法またはプロセスがあるわけではない。

（3）超高速開発

超高速開発は、自動コード生成や BRMS などのツール類を活用してソフトウェア開発期間を著しく短縮することにより、企業のスピード経営の実現を目指す考え方や取り組みを指す。その代表的な推進団体は、超高速開発コミュニティ [8] である。

（4）派生開発

派生開発は、既存のソフトウェアに変更・追加・削除を行うことにより新たなソフトウェアを作るという保守開発を指し、主に組込みシステムの分野で採用されている。代表的な手法では、適切なドキュメンテーションとトレーサビリティを利用する。

3 IoT 時代に向けた課題

IPA/SEC では、主にソフトウェア開発時における先進的な設計・検証技術の適用事例を数多く収集し、それらを整理して公開している [9]。筆者らは、これらの情報及び内外の公開情報を通して、システム開発・運用時に考慮すべき要素として、アーキテクチャ、エンジニアリング、環境・ツール及びマネジメントの 4（3+1）つを挙げる（開発スタイルはエンジニアリングに含まれる）。そして、これらの要素は相互に深く関連することから、それぞれ整合していることが重要である（図 2 参照）。

例えば、複数の小さな “ サービス ” を組み合わせてアプリケーションを構築する「マイクロサービス」[10] と呼ばれるソフトウェア・アーキテクチャでは、各サービスは独立にデプロイされ、疎結合で独立動作することから、俊敏な変更が可能であると共に、独立チームによる開発でチームの特徴を活かすことができるため、アジャイル開発に向くと言われる。また、アジャイル開発の生産性を高めるためには、とくに、頻繁なリリースごとに必要となる回帰テストの効率化（自動化など）が必須であることから、そのためのツールの活用が欠かせない [1]。

このように、効率的なシステム開発・運用のためには、前述の各要素間の整合を図るマネジメントが求められる。適切なマネジメントに必要とされるスキルは、「デザイン力」である。これは、ICT システムの対象や、それを使うビジネスの状況などを俯瞰し、採用するアーキテクチャ、開発スタイルやプロセス／プラクティスの組み合わせ、及び環境・ツールを決定する能力のことである。

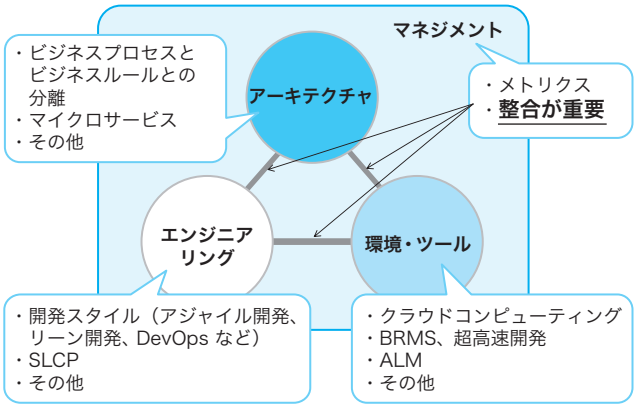

図 2　システム開発・運用時に考慮すべき 3 + 1 要素

あらゆるモノがインターネットに接続する IoT（Internet of Things）／ IoE（Internet of Everything）の時代が始まり、今後ますます進展すると予測されている。このような時代で生き残るには、デジタルビジネスを一層推進しなければならない。デジタルビジネスを持続的に発展させるためには、ICT システムの開発・運用に、優れた「デザイン力」によるマネジメントが重要となる。

最後に、組織・人材について考えを述べておきたい。IoT 時代におけるビジネスレイヤの人材には、IT 知識を背景に革新的なビジネスを創出するスキルが求められる。一方、IT レイヤの人材に求められるスキルは、更に 3 層に分けられる。ビジネスレイヤに近い層から順にそれぞれ、ビジネス知識を背景に適切な ICT システムをデザインするスキル、ビジネスニーズに対応して開発スタイルをデザインするスキル、エンジニアリングとツールなどを駆使してシステムを開発するスキルである。IoT 時代には、全体を見極めてビジネスやシステムについて適切にデザインできる、柔軟な組織・人材が求められる。

【参考文献】

[1] ソフトウェアメトリックス調査 2014, 一般社団法人 日本情報システム・ユーザー協会（JUAS）.
[2] CHAOS MANIFESTO 2014, The Standish Group International.
[3] IT 動向調査 2014, 一般社団法人 日本情報システム・ユーザー協会（JUAS）.
[4] Ram Chillarege: The Marriage of Business Dynamics and Software Engineering, IEEE SOFTWARE, November/December 2002.
[5] ソフトウェア開発データが語るメッセージ 2015, IPA/SEC, http://www.ipa.go.jp/sec/reports/20150925.html, 2015 年 9 月 25 日 .
[6] 山下博之 , 柏木雅之 : 非ウォーターフォール型（アジャイル）開発の動向と課題 , SEC journal, Vol. 8, No. 4, IPA/SEC, 2012 年 12 月 14 日 .
[7] 非ウォーターフォール型開発 WG 活動報告書 , IPA/SEC, http://www.ipa.go.jp/sec/softwareengineering/reports/20110407.html, 2011 年 3 月 31 日 .
[8] 超高速開発コミュニティ , https://www.x-rad.jp/.
[9]「先進的な設計・検証技術の適用事例報告書 2015 年度版」を公開 , IPA/SEC, http://www.ipa.go.jp/sec/reports/20151118.html, 2015 年 11 月 18 日 .
[10] James Lewis: Microservices, ThoughtWorks, http://martinfowler.com/articles/microservices.html, 25 March 2014.