

情報システム開発の開発方法論における 課題に関する研究

Study on problems of
Information systems development methodologies

小 川 裕 克*

Hirokatsu OGAWA

はじめに

情報技術（IT）の進展と共に、情報システムは企業にとって、なくてはならない存在となった。そして今では情報システムの良し悪しや活用の仕方が企業収益を大きく左右するまでになってきている。

一方で情報システムの重要性の増大と共に、様々な問題や課題が指摘されている。例えばその開発スピードは遅く、多大な開発・維持コストがかかること、ユーザーが望む機能が提供されないこと等である。また大小さまざまなシステムトラブルが多発し、業務に支障をきたすことも多い。

情報システムの開発に関する問題を解決するために、1960年代からソフトウェアエンジニアリングとして、科学的工学的なアプローチがなされてきた。2000年以降もSWEBOK（the Software Engineering Body Of Knowledge：ソフトウェア工学基礎知識体系）などが発表されたり、オブジェクト指向モデリングやUML（Unified Modeling Language）が普及してきていたりしているものの、開発の生産性はここ20年でそれほど上がってきているようには見えない。これは、一言で情報システムと言っても千差万別であり、しかも効率的で品質の高いシステム開発方法論が必ずしも確立していないことも大きな原因であると考えられる。

日本国内において採用されているシステム開発方法論の大半はウォーターフォールモデルであるが、この開発方法論を採用していることが情報システムに様々な問題を与えているという声も多い。ユーザーニーズが必ずしも明確にならないまま、開発が進められる場合も多く、後々品質面で問題が出たり、システム開発プロジェクトに大きな負荷がかかったりしているからである。これに対して、海外ではアジャイル開発

* 経営総合学科－情報技術

Department of Management Synthesis－Information Technology

が主流になりつつある。それ故、日本国内でも、システム開発会社の中には、ウォーターフォールモデルの欠点を独自のやり方で克服しようという試みも出ている。また、クラウドサービスを活用することにより、開発を最小限に抑えるやり方も広がってきている。このような状況において、情報システムの開発に関しては個別の開発方法論を論じたものが多いものの、全体を俯瞰して整理したものは少ないように見える。

本研究では、上述のような情報システムの現状と将来動向を踏まえて、システム開発方法論全般に焦点をあてる。まず主なシステム開発方法論の長所・短所、及びシステム開発の現状を整理する。次にシステム開発の現場ではどういう課題があるか、そして情報システム開発のあるべき姿について考察する。

1. 情報システム開発におけるシステム開発方法論

1.1 現在利用されている主なシステム開発方法論

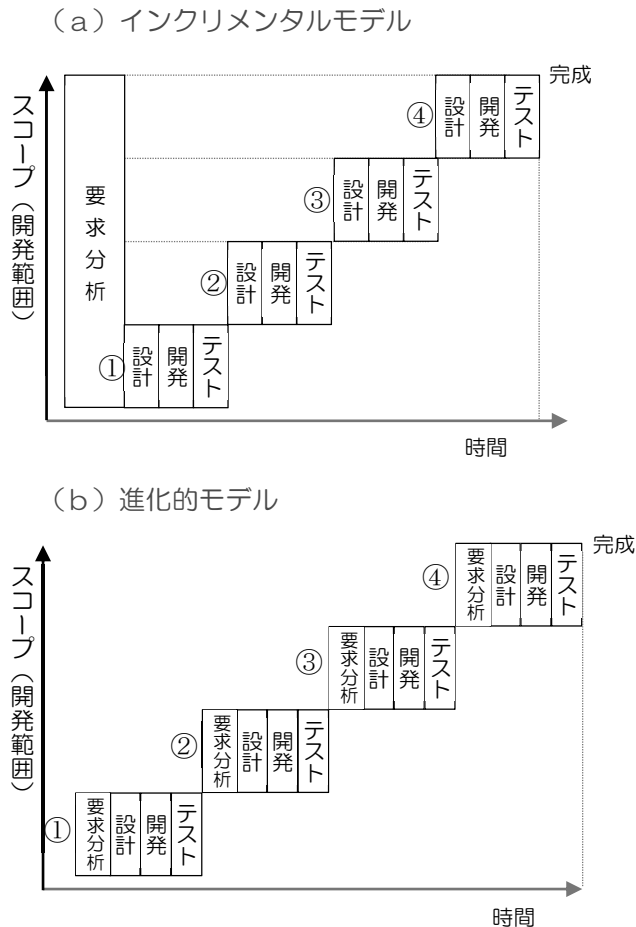
(1) ウォーターフォールモデル

ウォーターフォールモデル（以下、WFモデルと省略する）はシステム化の計画からシステムの完成までをいくつかの工程に分け、逐次進めて行く開発モデルである。まず業務アプリケーションの機能など、システム化における要件を定義する。これを受けてシステムの設計、プログラム開発、テストという順で作業を進める。そのため、前半の工程（設計）の結果が正しいことの確認は、最終的に後半の工程（テスト）で行うことになる。

(2) インクリメンタル（段階的）モデルと進化的モデル

インクリメンタルモデルは、まずすべての要求分析（及び要件定義）を行い、要求事項をいくつかに分割し、順次開発していく方法論である（図表1（a）参照）。インクリメンタルモデルの場合、すべての要求分析を最初に行うが、進化的モデルは要求分析も含めて順次開発を行っていくモデルである（図表1（b）参照）。両モデル共に全体をサブシステムに分け、順次リリースして行く。各サブシステムの開発は基本的にWFモデルを採る。大規模システムを開発する場合、開発リスクを減らすために、設計・開発をいくつかのフェーズに分けて行うことも多いが、このようなケースもインクリメンタルモデル、または進化的モデルを採用しているものと言える。

〔図表1〕 インクリメンタルモデルと進化的モデルの開発プロセス

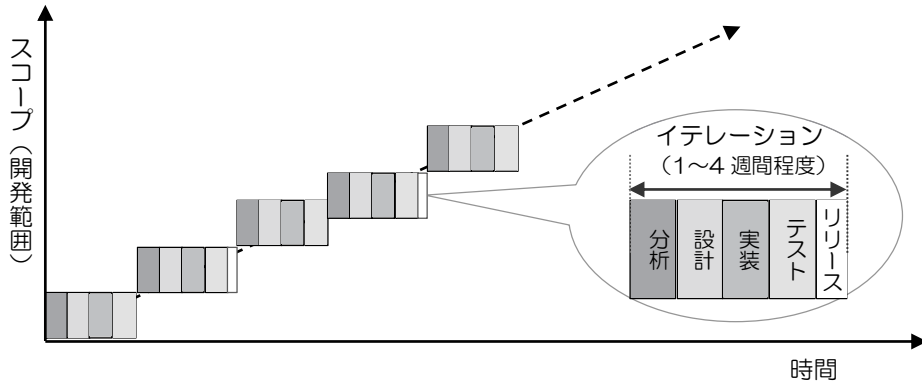


(出所) SEC『共通フレームワーク2013』(pp327-329)を参考に筆者作成

(3) プロトタイプモデル

プロトタイプモデルはユーザーにシステムの大まかな機能や操作性を把握してもらうため、まずプロトタイプ（試作品）を短期間で作成する。このプロトタイプをユーザーに確認・評価してもらった後、改めて正式に利用するシステム（完成品）を設計・開発していく方法論である。特にユーザーインターフェースとなる画面の設計・開発ではユーザーとの認識のギャップが少ないという利点がある。プロトタイプモデルでは、完成品の開発にあたり、改めて最初から作り直すやり方と、プロトタイプそのものに手を入れて行くやり方があるが、開発期間や開発コストの削減のため、後者の方法を採用しているケースが多いと考えられる。

〔図表2〕 アジャイル開発における開発工程



(出所) 筆者作成

(4) アジャイル開発（アジャイルモデル）

アジャイル開発はWFモデルにおける問題を解決するために考案された。このモデルは、ユーザーからのニーズやビジネスの変化等に迅速（Agile）に対応し、情報システムを短期間で開発しようという方法論である。1990年代、RAD（Rapid Application Development）が注目された時期があったが、アジャイル開発はその延長線上にあるものと考えることができる。アジャイル開発では1～4週間単位でイテレーション（反復）を繰り返し、一つのイテレーションの中で、分析と開発を繰り返していく（図表2参照）。「アジャイルソフトウェア開発宣言」に述べられているように、個人の対話（開発メンバー間のコミュニケーション）を重視し、開発チームが一定の権限を持って自発的に開発する。またWFモデルでは絶対的に重要な「計画や契約、ドキュメント」よりも、「変化への対応、ユーザーとの協調、動くソフトウェア」を重視する。

一括りにアジャイル開発といっても、多くの方法論が存在するが、スクラム（SCRUM）とXP（eXtreme Programming）が代表的であり、両者を組み合わせるケースも多い。“Version One State of Agile Development Survey 2009”によると、スクラムとXPを合わせるとアジャイル開発の8割を占める。スクラムは開発プロジェクトを進めるための方法論であり、XPは開発手法中心の方法論である。スクラムの場合、開発体制は、通常10人以内で、プロダクトに対して最終責任を持つプロダクトオーナー（ユーザーの代表者）と、開発プロジェクトを円滑に進める役割のスクラムマスター、チームメンバー（開発者）がスクラムを組んでプロジェクトを進める。スプリント（前述のイテレーションに相当）の初めに計画会議を開催し、最後にはレビュー会議と振り返り（成果物や課題の確認等）を行う。

(5) マイクロサービス（マイクロアーキテクチャ）

マイクロサービスは、小さく分割されたサービスを、独立した一つのプログラムとして開発するやり方である。これまで述べた開発方法論は開発手順についての方法論であるが、マイクロサービスはプログラムの構造に関する開発モデルである。マイクロサービスにおける各サービス（プログラム）は、独立したプロセスとして稼働し、サービス同士はHTTPなどで通信する、疎結合の状態のアーキテクチャである。各プログラムの言語やミドルソフトは問わない。各プログラムの開発はアジャイル開発に依るケースが多いという¹⁾。マイクロサービスアーキテクチャは、AmazonのWebサービスのよう、顧客へのサービスを素早く開発し提供しようとする中で生まれたという。

SOA（Service Oriented Architecture）とマイクロサービスアーキテクチャは共にサービスの連携として捉えることができるが、SOAはESB（Enterprise Service Bus）によりインフラ（システム基盤）を最適化するなど、統一的思想に基づいているが、マイクロサービスは小粒のサービス（プログラム）同士が連携していれば良く、サービスを素早くユーザーに提供できることを重視している。

(6) 超高速開発

超高速開発は、設計情報などを管理するデータベース（レポジトリ）や開発ツールを活用し、開発期間の短縮やシステムの品質向上を目指す方法論である。レポジトリを活用することにより、システムの保守等の生産性も高まることが期待できる。また開発ツールを利用することにより、ソースコードを自動生成する等、主に実装（プログラム設計からプログラミング・単体テスト）、テスト工程を効率化することが期待できる。超高速開発はWFモデルやアジャイル開発とは必ずしも排他的な関係とはならない。JUAS（日本情報システム・ユーザー協会）による調査『ユーザー企業 ソフトウェアメトリックス2015』（p27参照）によると、超高速開発の生産性は、WFモデルに比べて、実装工期が26%短縮（実装工数では42%削減）されているという。また同調査によると、開発ツールのカバー範囲（支援する開発工程）は開発ツールにより異なるが、内部設計から単体テストまでのものが多い。

1.2 各開発方法論の長所と短所

(1) WFモデル

WFモデルは設計書等の成果物を作成し、レビュー等で確認した上で次の工程へ進むため、一般的に次のような長所が挙げられる。

①開発方法論としてある程度確立しており、プロジェクトマネジメントが比較的容

易である²⁾。

- ②特に国内におけるシステム開発のほとんどは WF モデルを採用しているため、経験者が多く、開発者の調達が比較的容易である。
- ③開発内容や開発スケジュール、開発コストを前もって明確にするため、システム開発会社へ開発を委託しやすい。
- ④全ての要件定義を最初に行うため、全体のシステム構造も初期段階で確定することができ、(少なくとも新規開発時は) 継ぎはぎだらけのシステム構造 (アーキテクチャ) になることを避けることができる。
また以下のような短所も多く指摘されている。
- ①直列に並べた工程に沿って順次作業を進めるため、システムの完成に時間がかかる。
- ②計画段階ですべての要件が明確にならないことも多く、要件定義の確定に想定以上の時間がかかったり、要件を見誤るケースもある。
- ③システムが要件通り、または設計通りに出来上がっているかは、開発プロジェクト後半のテスト段階にならないと確認できない。プロジェクトの終盤に問題が発生すると、全体スケジュールの遅延や開発コストの増大が生じ、ユーザーの業務へ甚大な影響を与える可能性が高い。
- ④設計書等のドキュメントの作成負荷が大きい。

(2) インクリメンタル (段階的) モデルと進化的モデル

インクリメンタルモデルは、WF モデルと同様な長所がある。さらにサブシステムに分けて開発するため、開発要員や開発コストも平滑化でき、WF モデルよりもプロジェクトマネジメント上のリスクを低くできる。また早い段階で一部のシステムが利用できるというメリットもある。

短所も WF モデルと同様であるが、段階的开发を行う分、設計上のミスへの対応が早めに行えるため、WF モデルよりもリスクが低いと言える。ただし複数のサブシステムが完成するまで、ユーザーにとって完成したサブシステムは利用価値が無い場合もあり得る。

進化的モデルの場合、インクリメンタルモデルよりもユーザーニーズとのギャップは少なく、さらに開発リスクが下がるものと考えられる。ただし要求分析を段階的に行っていくため、全体の整合性やアーキテクチャの効率性等に問題が発生する可能性がある。

(3) プロトタイプモデル

プロトタイプモデルの利点としては、早い段階でユーザーがプロトタイプを見る

ことができるため、ユーザーインターフェースのイメージや必要な機能を具体的に掴むことが可能となり、完成したシステムがユーザーの認識から大きくずれることが避けられる。しかしプロトタイプ開発以降、WFモデルによる開発を採用した場合、開発工数や開発期間が大幅に削減されるわけでは無い。プロトタイプはコーディング基準等にあまりとらわれずに素早く開発されたものも多く、プロトタイプのパログラム自身に肉付けしてシステムを完成させた場合、ソースコードが複雑に絡み合っ
て保守しにくいプログラムになる可能性がある。

(4) アジャイル開発（アジャイルモデル）

アジャイル開発はWFモデルの欠点を改良するために考えられた開発方法論だけに、以下のような長所がある。

- ①最初にすべての要件定義を行う必要がないため、要件定義自体に想定以上の時間と手間がかかることを回避することができる。ユーザーは早い時期に開発結果を確認することができ、要件定義とのギャップを直ぐに埋めることが可能となる。
- ②順次機能の追加・変更を行っていくため、ユーザーニーズの追加・変更に柔軟に対応できる。
- ③詳細なドキュメントを作成しないため、開発期間が短縮できる。
- ④各イテレーションは期間が短いため、開発作業の見積りや計画が立て易い。
- ⑤開発チームメンバーの自主性を重んじるため、モラルの向上が期待できる。

しかし、アジャイル開発もWFモデルにない短所が存在する。

- ①最終到達地点がはっきりしない（見えない）ままプロジェクトが進むことになりかねない。
- ②（複数のユーザーからの）ニーズが次々と出てきて収拾がつかなくなり、システムが複雑化したり、想定以上の開発期間や開発コストがかかってしまうこともあり得る。
- ③全ての決定はユーザーの代表であるプロダクト・オーナーと開発チームによる協議によるため、強いリーダーシップと責任者を明確にする必要のある大規模プロジェクトになるほどプロジェクトマネジメント上の問題が出やすい。開発チームの数が多くなると、チーム間連携がうまく行かなくなる可能性も高い。
- ④特に日本国内の場合、アジャイル開発の経験のあるエンジニアはごく少数であるため、開発者の調達が困難であったり、開発者養成にも時間とコストがかかったりする。
- ⑤一般的に開発メンバーはユーザーの業務知識等に疎いケースが多く、また基盤系とアプリケーション系のエンジニアのスキル分野にも大きな隔たりがある。ところがアジャイル開発では開発メンバーが設計・開発・テストを一貫して行う場合

が多いため、設計から開発までの高いスキルと幅広い知識が要求される場合がある。

(5) マイクロサービス（マイクロアーキテクチャ）

マイクロサービスの長所としては次のような点を挙げることができる。

- ①サービス（プログラム）同士が疎結合の関係にあるため、開発やメンテナンスがやり易く、短時間で機能の強化・変更が可能である。システムトラブルによる他のサービスへの影響も少ない。
- ②サービスを積み上げていくことにより、多様なサービスもしくは多くの機能を顧客にタイムリーに提供することが可能であり、拡張性にも優れている。
- ③開発言語やミドルウェア等が比較的自由に選べるため、色々な技術を採用することができる。

また短所としては次のような点が挙げられる。

- ①サービス全体としてのシステムの冗長化や整合性を取ることが難しくなり、システムの性能等もサービスごとに考慮する必要がある。
- ②複数のサービスが連携するような場合、並列処理や非同期処理など、デバッグ等も含めて、プログラム開発や運用管理の難易度が高くなる。
- ③プロセス間通信のオーバーヘッドが大きい。

(6) 超高速開発

超高速開発は開発ツールを活用するため、長所として以下の点が挙げられる。

- ①開発工程の一部を効率化または省略化することが可能となり、開発期間の短縮化や開発コストの削減が図れる。
- ②開発ツールの使用により、手作業が少なくなり、開発手順や成果物の標準化が進み、ソフトウェアの品質向上が期待できる。
- ③開発ツールを使うことにより、高度なシステム開発技術の必要性が少なくなる。
- ④設計情報を管理できるレポジトリを持つツールの場合、設計の再利用を図ることができ、システム・メンテナンスを効率化できる。
- ⑤超高速開発はWFモデル、またはアジャイル開発と組み合わせることが可能なため、それぞれの長所を享受できる。
- ⑥プロトタイプの開発にも適用できる。

また短所としては次のような点が挙げられる。

- ①開発ツールにより、対応できる（強みを発揮できる）開発工程が異なるが、開発ツール間の連携が弱い（例えばレポジトリを共有できないなど）。
- ②開発ツールの機能の適用範囲が限定されていたりして、開発者にとって必ずしも

満足ができるとは限らない。開発ツールの将来性・サポートにも不安が残る。

- ③開発ツールに習熟するのに時間がかかる可能性がある。開発ツールを変えると習得のための学習期間が新たに必要となる。
- ④超高速開発による開発工数や開発期間の見積実績が少ないと、新システムの開発見積が難しい可能性がある。
- ⑤開発ツールの導入・保守費用がかかる。

2. システム開発の現状

2.1 システムへの投資とシステム構築時の重要事項

(1) システムへの投資

JUAS『企業IT動向調査報告書2015』(p52参照)によると、国内企業のシステム投資は2010年度以降増加傾向にあるが、2014年度のシステム投資計画(IT予算)の内訳は開発費が44%、保守運用費が56%と、システムの維持のための費用の方が多い。また同調査報告書は、国内企業のIT予算を新規ビジネスや事業拡大のためにITを積極的に活用する予算と、現行ビジネスを維持・運営のために継続的に使う保守的な予算に分けて調査している。この調査によると、2014年度IT予算配分は、ITを積極的に活用する予算が21%、保守的予算が79%と、現状維持に圧倒的に多くのIT予算が費やされている(同調査報告書62P参照)。

さらにIT投資で解決したい中期的な経営課題に関しては、顧客価値の遡及や拡張、新規事業等の展開と言った「攻め」のIT投資と、効率化やリスク低減と言った「守り」のIT投資に分けた場合、「攻め」が35%、「守り」が65%となっている(同調査報告書67P参照)。しかも「守り」のトップは業務プロセスの効率化(省力化、業務コスト削減)が22%と一番多く、事務処理でコンピュータが使われ始めた1950年代後半以降、ずっと同じ課題に対応してきていることになる。

以上から、「多くの企業にとって、ITコストが企業の収益に及ぼす影響が大きい。しかし情報システムの保守に多額のコストを掛けざるを得ない。そのため、新規開発を少なくしてITコストはなるべく抑えることが必要である」、または「企業経営者は、ITを積極的に活用して競争力強化やビジネスを変革して行きたいという意識が必ずしも高くない」という推測もできる。そういう中で、CIOもしくはIT部門は、「限られた予算の中で、いかに情報システムを効率的に開発・維持していくか」「情報システムをビジネスにどう役立てていくか」を考え、経営者に提案していくことが重要であると考えられる。

(2) 情報システム開発に際して重視すべき事項

企業が情報システムを構築する際、「品質 (Quality)」と「コスト (Cost)」, 「スケジュール (Delivery)」を重視していることは周知の通りである。しかし, 特に重要な要素はシステムによって若干異なる。JUAS「企業 IT 動向調査報告書 2014」(pp166-168)によると,「基幹系」「管理業務系」システムでは「品質」が一番重視され,次に「コスト」が続く。また「開発スピード」よりも「継承性」「変更容易性」が重視されている。「品質」が最重要視されているのは,企業のビジネスを支えているためと考えられる。さらに「継承性」と「変更容易性」が重視されているのはシステムのライフサイクルが長いためであると考えられる。ただしビジネスモデルがITに依存している企業ほど「品質」と共に「開発スピード」を重視する傾向が高いという結果も出ている(同調査報告書p171)。

一方「Web・フロント系」では「品質」「コスト」と共に「開発スピード」が重視され,「継承性」はそれほど重要視されていない。「Web・フロント系」は顧客と多く接するシステムでもあり,ビジネス環境の変化(顧客ニーズの変化)に応じてシステムを素早く変化させていく必要があるためと推定される。「業務支援・情報系」も「Web・フロント系」と同様な傾向にある。

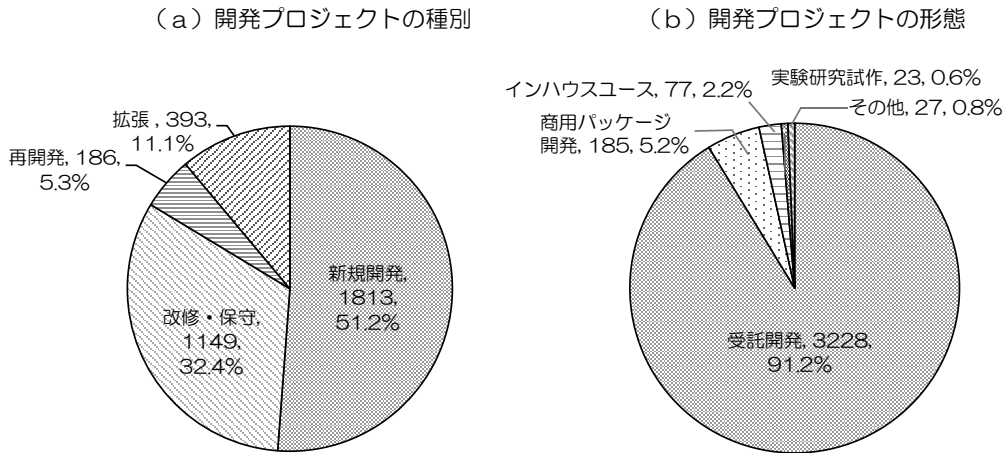
なお同調査報告書(p169)によると売上高の大きい大企業ほど「コスト」と共に「開発スピード」(小企業の場合はその逆)を重視する傾向がある。特に「業務支援・情報系」システムでは,「コスト」よりも「開発スピード」が重視されている。いずれにしてもITコストはどの企業にとっても無視できないコストである。つまりTCO(Total Cost of Ownership)の削減は大きな課題であり,「コスト」との兼ね合いで「開発スピード」を犠牲にせざるを得ない場合が多いということが窺える。

2.2 開発プロジェクトの概要

(1) 開発プロジェクトの特徴

主に大手システム開発会社を調査対象とした『ソフトウェア開発データ白書 2014-2015』³⁾によると,開発プロジェクトの種別は,新規開発が51%,改修・保守が33%となっている(図表3(a))。また比較的小規模プロジェクト(400FP未満,40KSLOC未満⁴⁾)でも,新規開発が42%,改修保守が37%と,新規開発の割合が若干少なくなるものの,同様な傾向にある。前述2.1の結果を合わせて考えると,新規開発の大部分は外部の業者に委託し,改修・保守の一部はユーザー企業内の社員が行う傾向があるようである。

【図表3】 開発プロジェクトの種別と形態



(出所)『ソフトウェア開発データ白書2014-2015』(p35-36)

またプロジェクトの開発形態を見ると、受託開発が91%となっている（図表3 (b)）。つまりユーザー企業とシステム開発会社とがシステム開発の契約を締結する場合、開発範囲、責任範囲のはっきりしていることが条件となっていることがわかる。従って外部委託の場合、採用する方法論も作業範囲、責任範囲が明確になるものが好まれているということになる。

(2) システムの特性

図表4 (a) はシステムの種別を分類したものである。アプリケーションシステムが9割以上占めている。しかもアプリケーション開発において、業務パッケージを利用しているケースは18%と少ない（図表4 (b)）。業務に適したパッケージが少ないためとの推定も可能だが、以前から言われているように、日本の企業は欧米に比べて「業務パッケージに自社業務を合わせるのではなく、自社の業務に合ったアプリケーションを要求する」傾向が強いためとも言える。

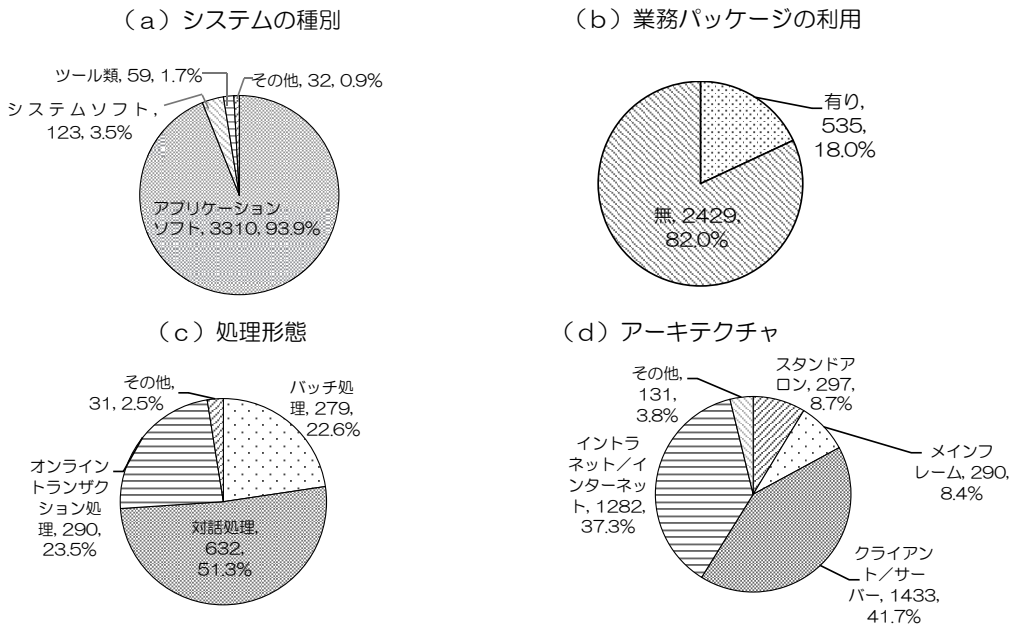
以上から、開発方法論について議論する場合、アプリケーションの開発を中心に行うことが重要であることがわかる。また業務パッケージを上手に活用することが、情報システムのQCDの観点から重要な課題であるとも言える。

図表4 (c) (d) はシステムの処理形態及び採用しているシステムのアーキテクチャを分類したものである。インターネットの普及や通信プロトコルTCP/IPによる通信手順の標準化により、対話型処理やインターネット／イントラネット、クライアント／サーバー方式が広まっていることがわかる。

なお同調査報告書によると、開発言語の割合はJava 30.4%、COBOL 15.3%、

VB 11.8%, C言語 10.7%, C++6.3%の順になっており、オブジェクト指向言語が普及してきているが、依然COBOLの割合も多く、メインフレームの利用は減少しているものの、クライアント／サーバーにおいても一部COBOLが使われていることが推定される。古いプログラムで書き換えが困難なものが残っているためであると考えられる。またDBMSではOracle（49.8%）を筆頭にほとんどのシステムでリレーショナルDBが採用されている。

〔図表4〕 システムの特性別分類

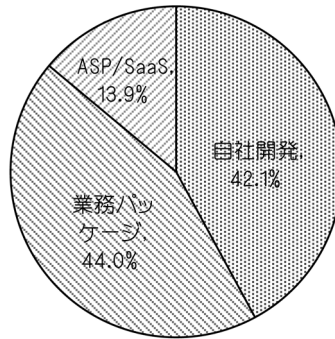


(出所)『ソフトウェア開発データ白書2014-2015』(pp40-41)

(3) アプリケーションの開発形態

図表5は自社開発か、または業務パッケージを利用するか等のアプリケーションの開発形態を表したものである。自社開発の中には、開発を外部委託しているケースや、PaaS等を利用した開発も入っているものと考えられる。また業務パッケージの利用は44%と、図表4 (b) の18%に比べてかなり多いが、業務パッケージをそのまま利用するケースと、業務パッケージをカスタマイズして使うケースの両方が混在しているものと推定される⁵⁾。いずれにしても、今後はPaaSと共にSaaSの利用も伸びていき、開発負荷を軽減する方向に行くものと推定される。

〔図表5〕 アプリケーションの形態



(出所) JUAS『企業IT動向調査報告書2014』(p184)をもとに筆者が集計

2.3 情報システムの規模と開発体制

(1) 情報システムの規模

図表6は情報システムの規模をいくつかの指標から見たものである。大規模開発プロジェクトもあるものの、プロジェクトの規模は小～中規模のものが多く、また新規開発と改良開発を比べた場合、当然の傾向として新規開発の方がSLOC、工期、工数共に増加する。

〔図表6〕 情報システムの規模

指標	分析項目ごとの結果	筆者コメント
KSLOC (ソースコードのライン数。単位 1000)	- 平均 174.2KSLOC、中央 42.5kSLOC、標準偏差 511.1 (n=2,248)。 50KSLOC 以下が約 53%。 200KSLOC 以上のプロジェクトも 400 件以上あり。	100～200KSLOC 程度を中規模システムとすると、比較的小規模の開発プロジェクトが多い。 一方で大規模システムの開発プロジェクトも多い。
工期 (月数)	- 平均 8.7、中央 7.0、最大 47.2、標準偏差 6.2 (n=3441)。 6～8 か月のプロジェクトが一番多く、全体の約 53%を占める。	1 年程度を中規模プロジェクトとすると、1 年を超えるプロジェクトは少ないため、中規模以下のプロジェクトが多い。
工数 (基本設計から総合テストまでの工数。人月換算)	- 平均 174.6 人月、中央値 48.11 人月、標準偏差 523.4 (n=2,568)。 50 人月以内が 51%、100 人月以内は約 70%。	300 人月以上を大規模プロジェクトとすると、比較的小規模もしくは中規模の開発プロジェクトが多いが、規模にはらつきがあると推定される。

(出所)『ソフトウェア開発データ白書2014-2015』(pp56-63) のデータに筆者がコメントを加えた。

(2) 開発体制

日本の国内企業は、システム開発を外部へ委託するケースが多い。しかもシステム開発を受託したシステム開発会社も、さらにソフトウェアハウスにプログラム開発等を再委託したり、オフショア開発⁶⁾を利用しているケースが多いのが現状である。

『企業IT動向調査報告書2014』(p179)によると、システム開発における内製比率は42%となっている。ここで内製比率は、

内製比率(%) = (社員要員数 + 情報子会社要員数) ÷ (全開発要員数)

と定義されている。情報子会社も「内製」の中に入れているが、実際には情報子会社がさらに外部に開発委託しているケースも多いと考えられるため、実質的な内製比率はもっと下がるものと考えられる⁷⁾。

またユーザー企業内のIT技術者は、システム開発会社内のIT技術者も入れた総数に対して25%程度⁸⁾である。しかも開発プロジェクトに(専任で)参加していないIT技術者も多い⁹⁾。さらに国内の情報システム開発においては、受託開発が91%(図表3(b)参照)であることから、日本の内製比率は実質10~20%程度と見る方が妥当であると考えられる(ちなみに同調査によると、米国の内製比率は37%である)。

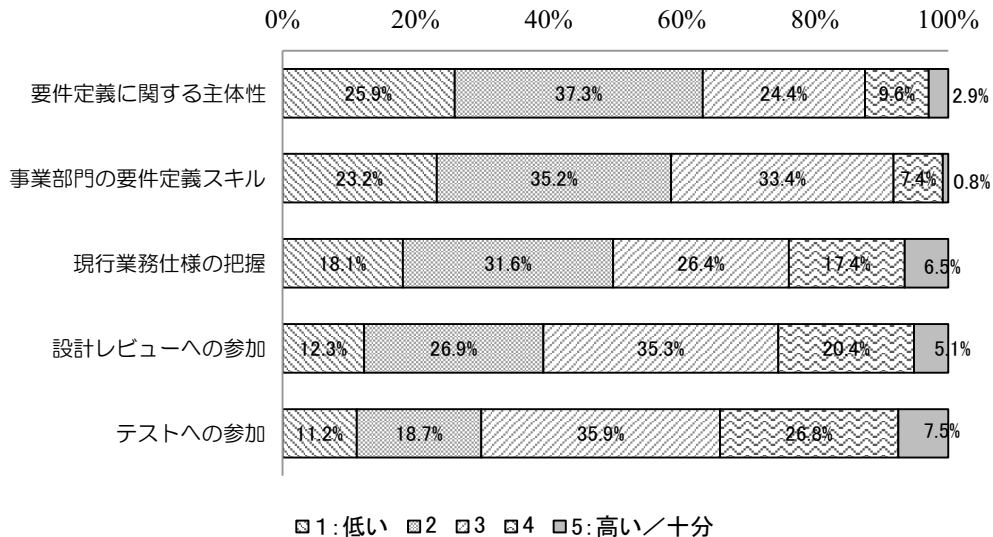
なお「企業IT動向調査報告書2014」(p183)によると、開発スピードを重視する企業ほど内製比率が高い。

(3) システム開発への事業部門の参画

システム開発、特に業務アプリケーションの開発においては、そのシステムのユーザー(事業部門)が要件定義やテストに主体的に参画することや、ユーザー部門の要件定義スキルと現行業務仕様の把握が重要な鍵となる。これらについては、WFモデル、アジャイル開発のどの方法論を採用しようと同様である。

ところがJUASによる調査結果を見ると、事業部門における「要件定義に関する主体性」は必ずしも高くない(図表7参照)。事業部門の主体性の低さをIT部門またはシステム開発会社がカバーしているという実態が見える。

[図表7] システム開発への事業部門の参画度



(出所) JUAS『企業IT動向調査報告書2014』(p172)

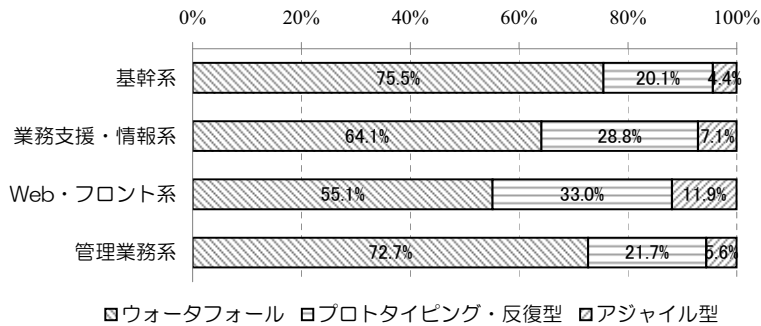
2.4 利用されている開発方法論の割合

国内で採用されている（ユーザー企業から見た）システム開発方法論の主流はWFモデルである（図表8参照）。特に「基幹系」「管理業務系」はシステム開発において試行錯誤的な要素が少なく、比較的業務仕様を確定し易いため、これに馴染みやすいWFモデルが採用されているものと見ることができる。

一方、図表9はシステム開発会社が採用している開発方法論の割合を示したものである。受託したシステム開発の97%がWFモデルである。WFモデルは受託契約に向いているということがわかる。

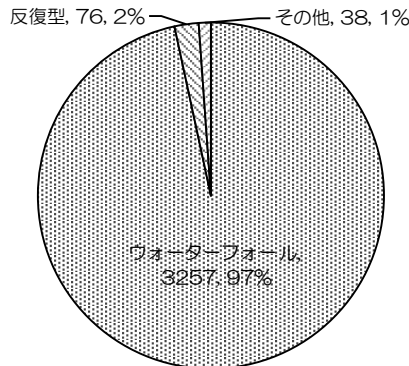
なお図表8、9を比較した場合、図表8の方が、プロトタイプモデル型及びアジャイル型の割合が高い。つまり、ユーザー企業内において、社員自らが開発しているケースや、システム開会社の技術者がユーザー企業内に常駐し、ユーザー側社員と一緒に（またはコミュニケーションを密に取りながら）開発しているケースも多いものと推定される。

〔図表8〕 採用しているシステム開発方法論（1）



（出所）JUAS『企業IT動向調査報告書2014』（p189）

〔図表9〕 採用しているシステム開発方法論（2）



（出所）『ソフトウェア開発データ白書2014-2015』（p48）

「2.1（1）システムへの投資」で述べたように、国内企業の場合、「守り」のIT投資が多いが、これは小回りの苦手な（コストのかかる）WFモデルが多いことも理由のひとつであると考えられる。また、比較的小規模のプロジェクトで、短期間で開発すべきものの一部は、図表8におけるプロトタイピング・反復型、またはアジャイル型で開発している可能性もある。

なお図表8、9の中に超高速開発が入っていないが、その理由としては適用事例が少ないためと、別のシステム開発方法論の中に含まれてしまっているためと推定される。

2.5 アジャイル開発適用の現状

2012年のIPA（情報処理推進機構）の調査¹⁰⁾によると、国内におけるアジャイル開発の適用分野はインターネット上のB2Cサービスが83%、市販ゲームソフトが

9%、基幹システムが8%となっている。つまりアジャイル開発は、特にサービス内容の追加・変更の多いB2Cサービスに適していると言える。また同調査によると、開発契約については、自社開発が58%（一部オフショア開発も含む）、準委任受託開発と請負受託開発が共に17%となっている。アジャイル開発では全ての要件が確定する前にスタートするため、開発契約に工夫が必要であるが、多くの場合、これを自社開発または準委任受託開発で乗り切っていることがわかる。また請負受託開発契約の場合も、開発内容を柔軟に入れ替える工夫がなされているものと推定される。ただし、開発期間や開発コストに厳格なプロジェクトにおいては、開発内容の入れ替えが柔軟に行えるかは疑問である。

さらに同調査によると、アジャイル開発適用にあたっては、プロジェクト内で様々な工夫がなされているようである。例えば、適用分野を、変化の必要な部分に限定したり、最初は開発メンバをユーザーと同一フロアに集めるなどしてコミュニケーションを密接にする、ユーザーの役割の多くを代替するメンバーを割当てることにより、ユーザー側負荷を軽減するよう工夫する、開発メンバーを徐々に増やすことによりアジャイル開発になじませる、等である。なおサブシステム同士が疎結合になるようにしたり、基盤システムの開発にアジャイル開発を採用する場合は、アプリケーションとは別サイクルにするなどの工夫をしているという。

また開発者から見ると、WFモデルに比べて、アジャイル開発はストレスが少なく、生きがいがあるという結果も出ている。これは、アジャイル開発はプロジェクトの透明度が高い上に、開発者にとって、単にある部分のプログラム開発ではなく、その上流も含めて参加できることが理由として挙げられる。さらに、適用プロジェクトの多くが小規模のため、大きな問題や課題が発生しにくいことも考えられる。

なお大手システム開発会社が、2015年7月にアジャイル専門組織を立ち上げたり、ユーザー企業がアジャイル開発を指定するケースが増加しているという¹¹⁾。つまりアジャイル開発の割合は現在も少ないものの、徐々に拡大している模様である。

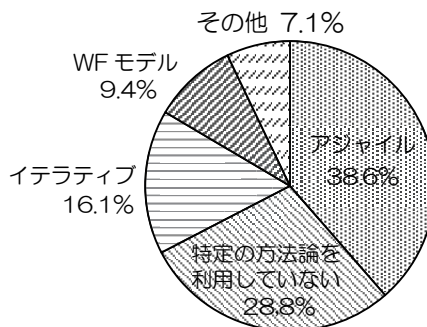
JUAS「ユーザー企業ソフトウェアメトリックス調査2015」(pp17-23)によると、アジャイル開発及び超高速開発が採用されているプロジェクトの50%以上が既存システムの再構築となっている。その採用理由を多い順（複数回答）に見ると、アジャイル開発の場合、「工期短縮」（68%）、「コスト削減」（46%）、「要件抽出が困難」（32%）であり、超高速開発の場合は「コスト削減」（92%）、「工期短縮」（86%）となっている。またアジャイル開発、超高速開発共に、開発担当者が引き続き保守運用を担当しているケースが多い（それぞれ55%、62%）。アジャイル開発及び超高速開発のメリットを活用し、かつデメリットを克服していることがわかる。

3. 海外におけるシステム開発の現状

3.1 利用している開発方法論とアジャイル開発の優位性

海外においては、米国中心にアジャイル開発が主流となっている。A B M MoniruzzamanとDr Syed Akhter Hossain¹²⁾によると、アジャイル開発とWFモデルの割合はそれぞれ39%、9%と、圧倒的にアジャイル開発の方が多い（図表10参照）。またアクチュエーション・コンサルティングによる2014年11月の調査¹³⁾によると、WFモデルとアジャイルの両者のブレンドが45%、Agile/Scrum が33%、WF モデル10%となっており、今後ブレンドとアジャイル開発との差は徐々に縮まって行くとしている。

〔図表10〕 海外において採用されている開発方法論



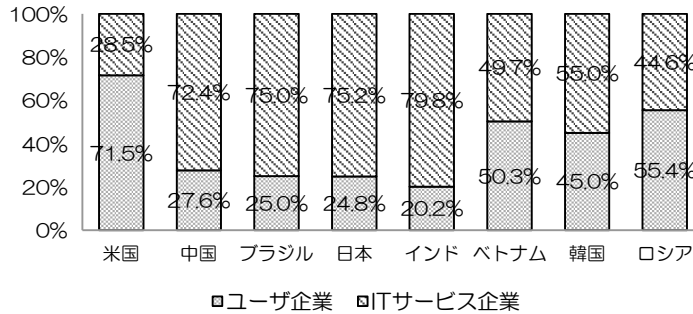
（出所）“Comparative Study on Agile software development methodologies” (p18)

前述A B M MoniruzzamanとDr Syed Akhter Hossainは同論文の中で、「開発途中における要求変更への対応の柔軟性」「仕様変更に伴う開発コスト変化」「ドキュメント作成のための負荷」「システムの複雑度とプロジェクトの成功度合」等についてアジャイルモデルとWFモデルとを比較し、いずれのケースもアジャイルモデルの方が優れているとしている。

3.2 IT技術者の属性

IPAの調査によると、米国の場合、日本と違いユーザー企業に属するIT技術者の割合のほうがシステム開発会社に属するIT技術者の割合よりも多い（図表11参照）。IT技術者がユーザー企業に属する場合、システム開発は比較的柔軟に進められる。つまりスケジュールや開発機能を柔軟に変更でき、かつユーザーとの連携をより緊密にすることが可能となるため、アジャイル開発の割合が多いと言える。

〔図表11〕 IT技術者の所属先



(出所) IPA『非ウォーターフォール型開発の普及要因と適用領域の拡大に関する調査 調査概要報告書』2012年6月11日 (p21)

しかし、これは日本が必ずしも特殊であるとは言えない。中国、ブラジル、インドも同様な傾向にあるからである。ただし、IPAによると、米国、中国、ブラジルでは、IT技術者は人気が高く、優秀な人材が集まっており、さらに流動性も高いという。逆に日本の場合はIT技術者の流動性が低く、学生の間では「仕事がきつい職業の一つ」であるという評価も存在する¹⁴⁾。この傾向が情報システムの開発に暗い影を落としている可能性がある。短絡的に米国を礼賛するのではなく、IT技術者にとって魅力ある労働環境を整えていくことが重要であると考えられる。

4. 情報システム開発における現状の問題または課題

4.1 開発方法論に関する問題または課題

(1) WFモデル

WFモデルには色々な問題や課題が内在している。まず開発に時間がかかり、柔軟性がないため、現在のようなビジネス環境の変化が速い時代には必ずしもそぐわないという点である。特に大規模システムの新規開発では要件の確定が難しく、スケジュール遅延やコストオーバーが発生し、しかもユーザーの業務に適合しないことも多い。さらにシステム開発契約（請負契約）後に、ユーザーから仕様の変更や機能追加を強いられることも多く、スケジュールの遅延やコスト増、さらには品質の低下を招いたりして、場合によっては、（システム開発会社からみると）赤字プロジェクトになってしまう危険性も高い。特に最近では、日本国内でもスケジュール遅延が訴訟問題に発展しているケースもある。WFモデルの短所が影響し、多くのユーザー企業の経営者には「システム開発には金（コスト）と時間（期間）がか

かるものだ。その割に効果が低い。」ということが定着してしまっているように見える。

次に、システム開発会社により各設計・開発工程の定義が異なることである。たとえば「基本設計＝外部設計，内部設計＝詳細設計」としているケースや「基本設計＝外部設計＋内部設計，詳細設計＝プログラム設計」としているケース，基本設計の前工程として概要設計の工程を入れているケースなどである。IPA が公表している「共通フレーム2013」も，各工程の定義は概念的には理解できるものの，必ずしも明確でない。これまでの基本設計，または外部設計，内部設計等とどう対応するのもよくわからない。結果として，現場では何も改善されないことになる。また開発言語としてCOBOLの割合がまだ多いことも開発工程の定義に影響を与えている可能性もある。生産性向上の一環として，統一的な設計・開発工程の標準化も課題の一つとして考えられる。共通フレームワーク2013を標準的なモデルとして提示するだけでなく，統一化に向けて具体的な改善を進めることが期待される。

(2) アジャイル開発及び超高速開発

アジャイル開発については，日本国内で普及していないということもあり，公に紹介されている事例が少なく，その効果がよく見えない部分もある。情報システムは当該企業の根幹に関わる部分であるため，詳細は社外秘のケースが多いこともその一因であると推定される。アジャイル開発については事例をできるだけ増やすことが必要である。

最近は大規模システムの開発でアジャイル開発を適用した例も出ているが，アジャイル開発の特性上，やはり小規模システムの開発の方が適しているものと考えられる。大規模システム開発においては，特にシステム全体の最適化が重要になる。単純にアジャイル開発を適用すると，いびつなシステムができたり，非機能要件が満たされないシステムが出来上がってしまう危険性もある。

アジャイル開発は，ユーザーと開発者が同じフロアで一緒に開発することにより，より大きい効果が期待される。しかし，比較的小規模な開発テーマをWFモデルで開発する場合，開発者がユーザー部署に常駐して短期間でかつ柔軟な対応を取りながら開発をしているケースも多い。この場合，アジャイル開発とWFモデルとの開発スピードの差はあまり出ない可能性もある¹⁵⁾。

アジャイル開発は，顧客及び開発者同士のコミュニケーションを重要視しており，WFモデルと比べて科学的なアプローチという点で弱いように見える。これは，ターゲット（システムの要件やスケジュール）がはっきりしないことも原因の一つと考えられる。また設計から開発まで一気通貫の開発となるため，システムアナリストやシステムエンジニア，さらにはプログラマー等，得意とする専門性を発揮しにく

い面もある。

超高速開発における課題については、まず開発ツールの機能の充実が挙げられる。開発ツールが提供する機能のレベルがユーザーニーズに達していない場合があるからである。例えば、開発ツールでは要求されたユーザーインターフェースを満たすことができない状況も発生しているようである。また複数の開発ツールの連携機能の充実も課題の一つである。既に「超高速開発コミュニティ」が設立され、ツール・メソドロジーの普及活動がなされているが、まずは自社の開発ツールを売り込むことの方が優先順位としては高いという声もある。今後は開発ツールを活かす開発方法論の明確化、標準化、そしてツール間の連携を推進する場としての機能も期待したい。なお超高速開発によるプロトタイプ開発への適用も提案されている¹⁶⁾。超高速開発をプロトタイプ開発に適用すれば、プロトタイプ開発における品質面での課題もクリアできるものと期待できる。

4.2 開発体制及び開発を進める上での問題または課題

(1) 情報システム開発における価値観の違い

情報システムの開発においては、ソフトウェア工学からのアプローチが必ずしも支持されているわけではない。システム化の目的やニーズが多種多様であることもあり、プログラマーの技芸や価値観が入りやすい。その結果、ソフトウェア開発の標準化や自動化が必ずしも進まず、生産性向上に結び付かない状況に陥ってしまう場合もある。

逆に「ソフトウェアファクトリー」的なアプローチを採用したために、ソフトウェアの生産性や品質向上に寄与したものの、エンジニアのモチベーション維持や、独創的（発展的）なソフトウェア、または高度で競争力のあるシステムを開発すると言った観点からはマイナスとなってしまうケースもあり得る。開発するシステム（ソフトウェア）の性格・位置づけを確定した上で最適な開発体制・開発方法論を採用する必要がある。

(2) 情報システムのブラックボックス化への対応

情報システムを効率的に維持するには、その情報システム（ソフトウェア）を熟知している技術者が存在することと、ドキュメント（設計書）やソースコードが最新の状態に維持されていることが重要となる。ところが現実には（ユーザー企業、システム開発会社のどちらにも）システムを熟知している人がいなかったり、ドキュメントが維持されていないケースが多い。情報システムのブラックボックス化を無くし、システムの維持・強化をどう効率的に行い、かつ高い品質を維持するかは、

今もって課題の一つして存在する。超高速開発におけるリポジトリの利用が解決策の一つとなる可能性が高い。

(3) (アプリケーション) パッケージの利用

ERP パッケージなど、パッケージを導入すれば、開発しなければならない部分が減り、情報システム開発のコストや開発期間を減らすことができる。しかし、現実にはパッケージを現状の業務に適合させたり、周りの情報システムと連携させるために、パッケージのカスタマイズやパッケージの周りに複数の新システムを開発する必要が出たりして、かえって開発コストが膨らんだり、開発期間も短縮できないような状況が多く発生している。クラウドサービス (SaaS) を利用する場合も、サービスそのものを単体で使う分にはあまり問題がないとしても、他のシステムと連携したい場合はやはり同様な問題が発生しうる。パッケージを利用する場合は、業務の標準化等を考慮して、ある程度割り切って利用する (カスタマイズはしない) 等の判断が必要となる。

(4) WF モデルを採用した開発プロジェクト

WF モデルでは最初に要件を確定しなければならないが、事業部門 (ユーザー) の要件定義に関する主体性は必ずしも高くないことがシステムトラブルや全体スケジュールを圧迫する要因となっている。またリリース日は、システムの規模や機能、採用するシステムアーキテクチャなどにより、演繹的に導かれるように見えるが、実際は経営者もしくは事業部門によりリリースする日が前もって決められてしまい、結果として開発プロジェクトへしわ寄せがきているケースも多い。

さらに突き詰めると、どこまでが要件定義かという問題もある。ユーザーによる詳細な要求は外部設計が完了するまで確定しないからである。

WF モデルは、ユーザー企業がシステム開発会社に開発委託し易い (開発内容とスケジュールを明確にして契約できる) というメリットがあるように見えるが、開発内容の詳細が見えない段階で請負契約を締結することが多く、後々問題が発生しやすい。請負契約でなく準委任契約にすればシステム開発会社側のリスクは減るが、その分ユーザー側のリスクが高くなる。また契約金額は付加価値ベースではなく、投入予定工数 (人月) がベースとなっているケースが多く、システム開発プロジェクトを労働集約型にしている原因の一つになっていると考えられる。

規模の大きい新規開発プロジェクトの場合、プロジェクトの経験と能力がプロジェクトの成否を左右するが、特に大規模プロジェクトを担当できるプロジェクトマネージャの養成が大きな課題となっている¹⁷⁾。

さらにシステム開発において、多重下請け構造を採っているケースが多いことも

問題である。この場合、下請け会社のスキルが低いとスケジュールの遅延やソフトウェアの品質上の問題が発生し易い。下請け会社に所属する開発者（特に若者）の参画意識や達成感が低いという問題もある。

(5) アジャイル開発を採用した開発プロジェクト

アジャイル開発については、システム開発会社との請負契約に必ずしも適合しない点をどう克服するかという課題がある。システム開発会社にシステムの設計開発を依頼する場合、準委任契約か投入人数を基準とした請負契約が考えられるが、顧客サイドへの人材派遣の要素が強くなってしまう。当初想定の開発費からの変動分を最後に調整することも考えられるが、顧客側からすると予算オーバー、システム開発会社側からすると利益の縮小、リリース時期の遅延については双方にとってのリスクとなる。また準委任契約の場合、成果物等は全て発注者側の責任となってしまうため、稼働後の情報システムに何らかの齟齬があった場合でも開発側に責任を問えなくなる可能性がある。さらにRFP（Request For Proposal）による開発も工夫が必要となってくる。

アジャイル開発を経験した技術者を増やすことも課題の一つである。顧客側がアジャイル開発経験者の投入を条件にして発注することにより、開発リスクを減らすケースもあるようだが、この場合、システム開発会社のアジャイル開発経験者を増やすという観点ではプラスとならない。

またアジャイル開発の場合、想定以上にシステムが複雑化し、開発コストが増大する可能性がある。特に大規模プロジェクトの場合、関係ユーザーが増えたり、開発者間の連携が難しくなったりするため、このような問題が発生しやすい。

長期的な方向性を決めるIT戦略とアジャイル開発によって完成したシステムとの整合性を合わせることも必要となる。アジャイル開発による短期開発、ユーザーズに沿った柔軟な対応ばかりに気を取られると、近視眼的なシステム開発になり、長期的な方向性を見失う可能性もあるからである。

アジャイル開発により完成したシステムの効率的な維持・強化も課題となる。EUC（エンドユーザーコンピューティング）と同様に、設計書が無かったり、コーディングもルール化・標準化されていなかったりすると開発者の退社や転勤などにより、プログラムの修正がままならなくなるからである。

(6) IT内部統制の整備

IT内部統制の整備と開発方法論をどう関係づけるかも課題の一つであると考えられる。特にビジネスニーズの変化に応じて情報システムを素早く変更していく必要がある場合、その分ビジネスのITへの依存度が高いということになり、IT内部

統制の整備も重要な課題となる¹⁸⁾。情報システムの開発・維持においては、採用すべき開発方法論や、標準化、セキュリティ確保、リリースルール等も含めた開発規準を定め、それらに沿った開発を進める必要がある。この観点からすると、プログラムの高頻度リリースやDevOps¹⁹⁾は、IT 内部統制に関連した問題が発生する可能性が高い。例えば、プログラムのリリースを急ぐあまり、不十分なテストのまま、開発者が勝手にリリースし、トラブルを誘発してしまうケースなどである。また開発者と運用者の相互けん制機能が必要である²⁰⁾と考えられるが、DevOps の推進により、相互けん制機能が効かなくなることもあり得る。

(7) IT 部門またはシステム開発者

日本国内においては、ユーザー企業における IT 部門のエンジニアが持つスキルと、システム開発会社におけるエンジニアが持つスキルに大きな違いがある場合が多い。求められているスキルも同様である。

ユーザー企業の IT 部門の場合、IT に関する専門知識や経験が少なく、自ら IT 部門に希望して異動してきた社員が少ないことも多い。しかもシステムトラブルへの対応や、ユーザーからの要望・問合せ対応等に追われており、新技術の調査・研究もままならないケースも少なくない。これは、情報システムの実質開発者がシステム開発会社であり、細かい日常の IT 対応は IT 部門が担当していることによるところが多いためと考えられる。いかに IT 部門に属する社員のモチベーションを高め IT スキル（特に IT に関する企画能力、IT 動向の把握、プロジェクトのマネジメント能力）を高めていくかが課題であると考えられる²¹⁾。つまり情報システムの内製化を図るには大きな壁があるということになる。

システム開発会社のエンジニアはシステムの設計・開発スキルを高めることが重要目標となっているため、顧客の業界動向や業務知識に疎い上に、これらを必ずしも重要視していない場合も多い。そのため顧客と一体となった開発形態を採ると、エンジニアは手足となってしまったり、エンジニア自身の帰属意識の問題等の様々な問題が生じやすい。また作業形態によっては派遣法上の問題も生じる可能性がある。

5. 情報システム開発において採用すべき開発方法論等について

ユーザー企業にとって、情報システムの品質をいかに向上させるか、いかに素早く開発するか、いかにコストをかけずに開発するかは重要な課題である。しかし IT 予算の制限や自社の IT 部門の開発能力のレベル等の問題により解決できていない場合

が多いのが現状である。

この課題を解決するためには、他社と差別化できない部分、もしくは省力化・効率化が目的の部分ではできるだけコストを抑えるために、クラウドサービスや既存のパッケージを活用することが重要な選択肢の一つとなる。一方でこれらを活用できない部分や他社との差別化のための独自部分は自社で開発することが望ましいものと考えられる。

情報システムの開発にあたっては、その性格や位置づけ等により、特に重視すべきQCDが変わり、それに伴い採用すべき開発方法論等も変わってくる。総合的に検討の上、最適な方法論を選べばよいが、複数の方法論を組み合わせることも選択肢の一つである。また長期的な観点から、品質の向上とメンテナンスの容易性を図るために、開発基準を設けると共に、開発の標準化、部品のコンポーネント化、エンジニアの能力向上を推進することが必要である。そして各開発方法論の長所・短所、課題・問題を十分認識しておく必要がある。

WFモデルを採用する場合、実質的に進化的モデルで進めれば、要件定義の遅れを避けることが期待できる。またアプリケーションの設計とシステム全体の方式設計を並行して進めることより設計期間の短縮を図ることもできる²²⁾。さらに開発ツールやリポジトリを活用することにより、超高速開発のメリットを享受でき、開発期間を大幅に短縮できる可能性が高い。特にユーザーインターフェース関連はできるだけ早い段階で確認が取れるようにツールを使うことが望ましい。

ニーズの変化への柔軟な対応や素早い開発を目的にアジャイル開発を採用する場合も何らかの工夫が必要になる。例えば外部に開発を依頼する場合、実績のある開発会社であるか、契約形態をどうするか等を考慮する必要がある。また情報システムを内製化する場合は、社内のIT要員の態勢に問題のないようにしなければならない。

開発を受託するシステム開発会社としても、アジャイル開発や超高速開発を使ったプロジェクトの経験を積むと共に、これらの方法論に関しても開発規準や標準化を確立する必要がある。一方で最新のITの動向をウォッチし、これらを適用できるだけの能力を培っておく必要がある。

なお情報システムの開発・維持、運用にあたっては、オペレーショナルリスクへの対応やIT内部統制（特にIT全般統制）の構築・整備に十分留意する必要がある。

おわりに

今後、IoT（Internet of Things）の拡大や技術革新と共に、情報システムはこれまで以上に複雑かつ広大なネットワークシステムの中に組み込まれて行き、大きく変貌していくものと予想される。例えば製造業におけるドイツのインダストリー4.0の

進展により、情報システムはCyber Physical Systemとして大きく発展して行く。金融の分野でも、IT企業が先導するフィンテック革命が進行して行くと思われる。

ユーザー企業は、IT革新のメリットを素早く取り入れて自社のビジネスをどう変化させ、どう強化させていくか、ビジネスの変化に沿って情報システムをどう対応させていくかが問われてくる。そのためには、IT戦略の一環で、情報システムを内製化するのかアウトソースするのかを越えて、有望IT企業と緊密に提携する（場合によっては自社内に取り込む）ことも必要になって来るものと予想される。このような変化に適確に対応していくためには、ITに造詣の深い経営者がIT部門を担当²³⁾し、今後のIT化の方針やIT部門の位置づけ・体制、システム開発態勢のあり方を見直していく必要がある。一方でIT部門の社員のモチベーションを高める努力も必要であろう。

同様にシステム開発会社においては、まずは基礎体力として、プロジェクトマネージャの育成や、先端技術のトレンドを見極めながら、その先端技術を適用できる能力と積極性のあるエンジニアの育成していく必要がある。さらにユーザー企業（顧客）との強い連携、場合によってはユーザー企業のビジネスを代替する（クラウドサービスとして提供する）ことも考えられる。

いずれにしても情報システムは既製品やパターンオーダー、イージーオーダーが増加して行くものの、オーダーメイドが無くなることは無いものと思われる。従ってシステム開発という業務（ビジネス）は今後も重要であり、当分なくなることは無いと考えられる。

注

- 1) James Lewis, Martin Fowler “Microservices”, 25 March 2014
<<http://martinfowler.com/articles/microservices.html>>参照。
- 2) しかしWFモデルが持つ短所ゆえに、実際はプロジェクトマネジメントが容易でない場合が多い。
- 3) 情報処理推進機構（IPA）技術本部ソフトウェア高信頼化センター（SEC）監修。
- 4) FPはファンクションポイントを、KSOLはソースコードのライン数（1000ステップ単位）を表す。
- 5) 日本国内において、業務パッケージのカスタマイズの割合は、欧米に比べて多い。IPAは『グローバル化を支えるIT人材確保・育成施策に関する調査』概要報告書（2011年3月）において、地域別パッケージの利用実態を調査している（p12）。これによると、2006年におけるERPパッケージのカスタマイズの割合は、20%未満もしくはそのまま使用が、日本では46%に対して、北米では55%、ヨーロッパ84%となっている。
- 6) IPA『「IT人材白書2013」グローバル／オフショア動向調査』によると2011年度のオフショア開発総額は1,000億円規模となっている。
- 7) 大手システムインテグレータ等を調査対象とした「ソフトウェア開発データ白書2014-2015」

によると、外部委託工数比率は平均61.7%（中央68.5%）、外部委託金額比率では平均56.1%（中央64.0%）となっている。つまり設計・開発が下請け会社に再委託されているケースが多いことを示している。

- 8) IPA「グローバル化を支えるIT人材確保。育成施策に関する調査」概要報告書（2011年3月）（p22）によると、システム開発プロジェクトにおけるIT技術者の所属先は25%がユーザー企業となっている。つまり、

$$\text{内製比率（\%）} = \frac{\text{ユーザー企業所属のIT技術者254,721人}}{\text{ユーザー企業所属のIT技術者} + \text{ITサービス企業所属のIT技術者771,426人}}$$
と定義すると、内製比率は24.8%になる。
- 9) ユーザー企業のIT部門の社員はIT企画業務や、ユーザーからと問合せ等への対応、システムトラブルのフォローなどに追われている時間が多い。
- 10) IPA『非ウォーターフォールモデルの普及要因と適用領域の拡大に関する調査 調査概要報告書』（p9）参照。
- 11) 日経Systems『さらば教科書アジャイル よく回るプロジェクト』2015年9月、日経BP（pp24-43）
- 12) “Comparative Study on Agile software development methodologies”, 12 July, 2013, Cornell University Library
- 13) Actuation consulting “Product Development Methods: Is Agile The Most Widely Used?”, 4 Nov, 2014
<http://www.actuationconsulting.com/product-development-methods-popular/>
- 14) 政井寛『IT技術者に未来はあるか？ — 第1回』ASCII.jp×ビジネス, 2011年4月28日
<http://ascii.jp/elem/000/000/587/587632/>参照。
- 15) このような開発形態自身もアジャイル開発に含めている可能性もある。なお契約形態は請負型でも、実際にSEまたはプログラマーを何人投入しているか、それぞれの単価を掛け合わせて契約金額を決めるが場合が多いものと推定される。なおユーザー部署常駐型の場合、システムの完成が優先され、ドキュメント（成果物）の品質に問題が出る可能性もある。
- 16) ICT経営パートナーズ協会『超高速開発が企業システムに革命を起こす』日経BP社、参照。
- 17) 筆者がシステムインテグレータ数社にヒアリングした結果である。
- 18) 金融商品取引法は上場会社に対して有価証券報告書と併せて内部統制報告書の提出を義務付けている。この内部統制報告に関しては、財務報告に関係している重要システムのみが対象である。しかしながら財務報告に係わらず、企業にとって重要なシステムはすべて内部統制を整備・運用する必要であると考えられる。また開発方法論からすると、IT全般統制の整備が特に重要であると考えられる。
- 19) DevOpsとは「ソフトウェアの開発担当と導入・運用担当が密接に協力する体制を構築し、ソフトウェアの導入や更新を迅速に進めること」である。
- 20) 例えば金融庁による金融検査マニュアル（平成27年11月）にはシステム開発・運用部門の相互牽制態勢の構築や、開発担当者によるオペレーションへのアクセス原則禁止などが記述されている。
- 21) IT部門に長く所属していると、ユーザー部門に異動して活躍させることが難しくなるということも考えられる。またユーザー企業がシステム開発会社からエンジニアを採用した場合、ユーザー企業で必要とされる業務知識のキャッチアップも課題になる可能性がある。
- 22) アプリケーションの設計と方式設計は当然連携する必要があるが、求められるエンジニアのスキルが違うため、通常別のエンジニアが設計を担当する。
- 23) 筆者がこれまでヒアリングしたユーザー企業の中には、どちらかというITの苦手な人が部門長になるケースも多い。もしくはITに造詣の深い人がIT部（門）長になる場合でも、その上にITに弱い役員が付き、結果として効率の悪い二重構造を形成しているケースもある。

参考文献

- 日本情報システム・ユーザー協会『ユーザー企業ソフトウェアメトリックス調査2015』2015年4月
- 情報処理推進機構ソフトウェア高信頼化センター『ソフトウェア開発データ白書』2014年10月1日
- 日本情報システム・ユーザー協会『企業IT動向調査報告書2015』2015年4月30日
- 日本情報システム・ユーザー協会『企業IT動向調査報告書2014』2014年4月24日
- 山下博之『非ウォーターフォール型（アジャイル）開発の動向と課題』2013年3月4日，IPA／SEC
<<https://www.ipa.go.jp/files/000027309.pdf>>
- 山下博之『非ウォーターフォール型開発WG活動報告書のポイント』2011年12月9日，IPA／SEC
<<http://www.ipa.go.jp/files/000004874.pdf>>
- 『非ウォーターフォール型開発の普及要因と適用領域の拡大に関する調査 調査報告書』2012年3月28日，IPA／SEC
- A B M Moniruzzamann, Dr Syed Akhter Hossain “Comparative Study on Agile software development methodologies” 12 Jul 2013, Coronell University Library
<<http://arxiv.org/abs/1307.3356>>
- James Lewis, Martin Fowler “Microservices” 25 March 2014
<<http://martinfowler.com/articles/microservices.html>>
- 情報処理推進機構ソフトウェア高信頼化センター『共通フレームワーク2013』2013年3月4日
- テクノロジーアート『応用編アジャイル概論』2012年5月10日，東京電機大学出版局
- ICT経営パートナーズ協会『超高速開発が企業システムに革命を起こす』2014年5月19日，日経BP社
- ITソリューションフロンティア『特集たゆまぬ生産革新への取り組み』2015年10月20日，野村総合研究所
- 日経コンピュータ『特集 広がる「超高速開発」』2015年10月1日，日経BP社
- 日経コンピュータ『特集 団結しないシステム開発』2015年6月25日，日経BP社
- 日経コンピュータ『特集1 オルタナティブSI』2015年2月5日，日経BP社
- 日経SYSTEMS『特集1 さらば教科書アジャイル 良く回るプロジェクト』2015年9月，日経BP社