O-018

Agile/DevOps の導入について
The Introduction of Agile and DevOps

原 清己†
Kiyomi Hara

1. はじめに

日本の多くの情報システムは、過去数十年間にわたり技術的負債を累積してきました。今までと同じような取り組みを続けていては、これからも技術的負債は累積し続けると懸念されます。技術的負債以外にもその裏には累積された人的負債があります。技術負債を解決するためには、人的負債の解決を合わせて行うことが求められます。人やチームの働き方・価値観の変革を進める必要があると考えます。その為に、ここでは Agile および DevOps の導入を提案します。Agile/DevOps とは素早く変化し続ける能力を開発するためのアプローチであり、これらを導入することで価値ある情報システムの構築ができると考えます。

2. 情報システムの課題点

日本の情報システムの課題点を考える時、経産省の『DX レポート』にて重要な点が指摘されました。要点は、情報システムのサイロ化、複雑化、そしてブラックボックス化です。その結果として情報システムの柔軟性が欠如し、情報システム自体がビジネス上のボトルネックになっているという重大な課題点です。昨今ではこのような情報システムを象徴的に「モノリシックシステム」略して「モノリス」と呼んでいます。こうした情報システムは、ビジネスを牽引するどころか、逆にビジネスの足を引っ張るボトルネックになっていると言わざるを得ません。

2.1 代表的な技術的負債

この分野の過去３０年間を振り返ってみると、３０年前に著者が感じていた課題と、今でも多くの企業が抱える課題点は大差ないように思われます。その代表的なものを挙げると、以下になります。

- 機能変更（システム開発）の所要時間が長い
  システムの企画からリリースまでのリードタイムが長い。開発作業/テスト作業の多くが未だに手作業である。
- システムリリースの頻度や回数が少ない
  上記の結果として、新機能のリリース頻度が少ない、ビジネスとリリースとのスピードギャップがある。
- システム障害の発生確率
  低品質のソフトウェア、特にリリース時、あるいは直後の障害発生が多発する。
- システム障害の回復時間（MTTR）
  障害対応能力、あるいはレジリエンス能力の欠如、開発と運用チームの協業不足が想定される。
- システム投資額の抑制
  システム投資額の不足、情報システムの開発・運用は間接費用として基本的にコスト削減対象となっている。
- IT ベンダー企業中心のシステム開発
  いわゆるベンダー丸投げのシステム開発が未だに主流になっている。
- エンジニアのスキル不足
  早い技術革新と広範囲なスキルエリアが求められるが、タスクのたらい回しなどで、エンジニアのスキル不足が顕著となっている（以下を参照）

以上は、日本の IT 業界では、積み上げてきた技術的負債の根本的な解決を怠ってきたことによる必然的な結果として理解することができます。IT 業界という言葉の意味は、ユーザー企業のＩＴ部門、および情報処理サービス業界の両方を含めています。

2.2 代表的な人的負債

情報システムとは、人やチームの能力、あるいは組織的な取り組みによって構築されます。日本の IT エンジニアの現状を考えてみた時、同志社大学の中田教授の報告書が参考になります。専門職力、経営組織管理力、および基礎的思考力における、日本の水準を他国と較べたものが報告されています。３つの能力すべてが５ヶ国（米中独仏日）平均以下は日本のみであり、とりわけ専門職力と基礎的思考力の低さが際立っている。（詳細は中田教授の報告書を参照）同様の内容が、他の調査機関からも報告されていますが、こうした報告書から判明するのは日本の IT 技術者のスキルレベルが相対的に非常に低いという事実です。価値のある情報システムを構築する上で、厳しい現実を突きつけています。

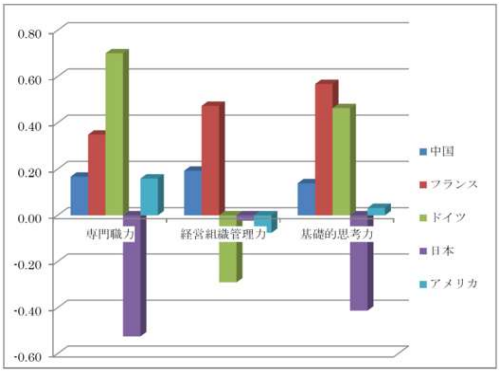

図１ ３つの能力変数の５ケ国比較
専門能力・経営組織管理力・基礎的思考力
「日本のソフトウェア技術者の生産性及び処遇の向上効果研究：アジア，欧米諸国との国際比較分析のフレームワークを用いて」に関する成果報告書より抜粋

2.2.1 日本の人材育成投資

海外の IT 先進企業では、新しいアイデア、新しい行動や取り組みを実験するアイデアソン、ハッカソンなどのイベントが開催されて、従業員の労働時間の最大 20%をイノベーションや改善への自由な取り組みに投資していると言われます。しかし、中田教授の報告によると、日本の技術者は自主的な能力開発のための自己啓発が低く、週当たり

の自己啓発時間が 0 時間という割合が 28%、週当たり自己啓発時間が 10 時間を超えている者の割合も非常に低いと指摘されています。また、日本企業の人材投資が海外に比べて非常に低いことがわかります。多くの技術者が日常業務に埋没し、次々に新しい業務にたらい回しにされている姿がうかがえます。これではスキルや能力を底上げするのは難しく、向上しようとする動機さえ生まれません。

3. デジタル時代のシステム開発・運用形態

ここでは、日本のこうした時代遅れのシステム開発・運用、あるいは仕事環境の現状を理解したうえで、米国の先進企業の導入事例を見てみます。彼らは何が違うのでしょうか？簡単に言えば、従来のようなベンダー主導型のウォーターフォール開発などは行っておらず、ビジネス駆動型の素早いアジャイル開発、その発展形である DevOps を導入している点だということです。

3.1 Agile/DevOps の導入事例

GAFA といったデジタル技術を最大限に活用している企業、つまりデジタルトランスフォーマーと呼ばれる企業群があります。彼らはデジタル技術を最大限に活用してビジネスを展開している企業群です。これらの企業の特徴としては、ソフトウェア開発手法としてアジャイル開発を採用し、さらに、その発展形である DevOps を全面的に導入している企業群です。それだけではなく、そのアジャイルや DevOps の背後にある考え方や価値観を全社に展開しているといった共通の特徴があります。例えば、グーグル社は 2013 年に、アマゾンは 2011 年に DevOps を導入したと発表しています。DevOps を導入したことで、何が実現できたのか。以下に公表された内容をご紹介します。Googl 社ワーナー・ボーゲル氏の講演から抜粋します。

- >10,000 developers in 40+ offices
- 5,000+ Projects under active development
- 17k submission per day (1 every 5 seconds)
- Single monolithic code tree with mixed language code
- Development on one branch – submission at head
- All builds from source
- 20+ sustained code changes per minute with 60+ peaks
- 50% of code change monthly
- 100+ million test cases run per day

この内容で驚くべきことは、平均 5 秒毎に新機能がリリースされ、毎月全体のソースコードの半分が変更され、1 日に 1 億テストケースが実行されているという事実です。その並外れたソフトウェア開発能力と開発スピード、それを頻繁にリリースできるという能力は信じがたいものです。そうしたことを実践しているプロジェクト体制として、5,000 以上のプロジェクトチームを編成して、それぞれのプロジェクトチームが平行的にシステム開発を行っているという点です。しかも、一つのブランチ上で開発を行っている。これらは、日本の現状からは想像もできないほどの格差があります。

3.2 IT サービスの本質

こうした数値を見た時は驚かされますが、そうした数値に驚くのではなく、このような事を達成するために、どんな目標を定めめてどんな取り組みをしてきたのかという点です、それを理解することが大事である、そのためには IT サービスの本質に遡って考えるべきだと考えます。IT サービスとは、組織のビジネス目標に貢献し、また、そのビジネス目標をいかに牽引することができか、それが IT サービスの本質です。現代のような変化の激しいデジタル化社会では、素早くジャストインタイムで新たな IT サービスを顧客に提供することが重要になります。また、安定した IT サービスを稼動させることが重要となります。システム障害による IT サービス停止は、組織にとって大きな損失を招くことに繋がります。

以上のことから、情報システムを評価するため目標基準、つまり、IT パフォーマンスの測定基準を新たに変えなければなりません。

3.2.1 新たな IT パフォーマンス指標

従来のシステム開発の管理指標として、QCD（品質、コスト、納期）が言われ続けてきました。これは IT プロジェクトを失敗しないようにするための指標です。従って、IT パフォーマンスの向上を測定するための指標ではありません。本来は、IT サービスの本質に基づいた評価基準となるように指標を設定すべきだと考えます。その指標とは以下のような主要 4 項目になると考えます。

IT パフォーマンスの評価基準

- システム変更に要する所要時間
- 本番環境へのリリースを行う頻度
- 障害発生時のサービス回復時間
- 障害発生の頻度（特にリリース時）

つまり、システム開発、システム変更に要する所要時間（リードタイム）をいかに短くするか、そして、それをいかに素早く頻繁に本番環境にリリースできるか、あるいはいかに頻繁にお客様にデリバリーできるか、ただし、その時に起こりえるシステム障害の可能性をいかに低く抑えるか、また、システム障害が起こったとしても、いかに短時間に復旧できるか、こうした能力を向上させるための指標であり、それによってビジネスパフォーマンスを向上させるための指標でもあります。これをここでは頭文字をとって LFMF と呼びます。

3.2.2 IT パフォーマンス指標：LFMF の向上

この新たな評価規準を向上させるためには、従来のウォーターフォールに代表される開発手法、それを管理するためのプロジェクトマネジメント手法、また、従来のソフトウェア品質管理手法、さらに、人や組織の働き方や価値観など、従来のあらゆる面を刷新するための包括的なアプローチや取り組みが必要となります。そうでないと達成することが困難であると考えます。

また、システム開発を従来のような単発的なプロジェクトとして捉えるのではなく、継続的なシステム改善活動、継続的なプロセス改善活動、そして、継続的な技術的／人的能力向上活動として捉える事が必要になります。そうした意味で、従来のようなプロジェクト志向の強い体質から、プロダクト志向あるいはサービス志向へのシフトが求められます。プロダクト志向、サービス志向になるということは、ライフサイクルマネジメントの考え方を IT サービスの中に導入することになります。これによって、今のような技術的負債や人的負債を、徐々にですが解決することができると考えます。

3.2.3 人的負債を改善することの重要性

技術的負債と人的負債はお互いが深く相互に関係しています。この観点では、コンウェイの法則があります。米国のコンピュータ科学者であったメルヴィン・コンウェイは、「システムを設計する組織は、そのコミュニケーション構造をそっくり真似た構造の設計を生み出してしまう。」と言っています。この法則によると、ソフトウェアのどの部分もそれを作り出した人や組織の構造を反映しているというもので、複雑な組織は複雑なアーキテクチャを生み出すことを意味しており、組織と情報システムとの密接な相互関連性を示しています。また、この法則はどんな力をもってしても破ることはできないと言っています。情報システムとは人や組織による創造物であり、プログラムのアルゴリズムからシステムの提供する価値まで、人や組織が作り出している創造物です。

4. Agile＆DevOps の導入

累積された技術的負債、累積された人的負債から脱却し、さらに新たな IT パフォーマンス指標を追求し向上させるための解決する具体的方法として、ここではアジャイル開発と DevOps の導入を提案します。アジャイル開発とは、反復的に且つ漸進的にソフトウェアを開発するための手法です。DevOps とは、開発チーム（Development）と運用チーム（Operations）が密接に連携することで、ビジネス価値を高め、かつ迅速にユーザーに価値を提供するための開発から運用までの新たな IT サービスマネジメントモデルです。ただ、その考え方の源流は同じものから生まれています。対象とする分野が少し違うだけで、アジャイルはソフトウェア開発分野であり、DevOps は IT サービスマネジメント全体になります。

そうした違いがあるものの、共に目指すものはシステムチーム（開発と運用）とビジネスチームが密接に連携しながら、情報システム部門のエンドツーエンドのバリューストリーム（プロセス）のフロー（流れ）を加速し、素早く頻繁に新たな機能をデリバリーすることであり、そのための包括的なアプローチである点です。

つまり、情報システム部門に存在するシステム企画、ソフトウェア開発、品質保証（QA）、デプロイメント、システム運用、情報セキュリティといったプロセスの全体である IT バリューストリームすべてを継続的に改善しながら変革を促すことであり、それによって、顧客価値の最大化、ビジネス価値の最大化を図るための実戦的手法のことであり、それを導入することが大切になります。

4.1 アジャイル開発の導入

アジャイル開発とはソフトウェア開発手法の１つとして捉えられており、従来のウォーターフォール開発と違って、システム開発のリードタイムを１カ月以内で反復的に開発するための手法です。ただし、アジャイル開発とは単なるソフトウェア開発手法ではありません。人の働き方そのものの変革を行うためのものであり、人づくり、チームづくり、組織づくりのためのフレームワークでもあります。人やチームに焦点をあて、現場で働く人が自ら継続的な改善活動を行うようにする考え方です。現場で働く人を尊重し信頼するためのフレームワークでもあります。その為には組織構造が変わらなければならず、組織マネジメントのためのフレームワークでもあると言えます。

従って、アジャイル開発のプロセスやプラクティスを理解するだけでなく、その背後にある原理・原則を理解することが大事になります。

4.1.1 Agile/DevOps の原理・原則

こうした Agile や DevOps の考え方の源流は TPS（トヨタ生産方式）とリーンであり、これらの原理・原則が背景にあります。その究極の姿が DevOps であり、DevOps とはアジャイルとリーンの発展形でもあります。

TPS（トヨタ生産方式）とは、より少ない資源で顧客価値を最大化するための車の生産方式ですが、顧客に価値を生まないもの、つまりムダを取ることによって顧客の注文から納車までの所要時間を短縮します。何がムダであるのかの捉え方が大事になります。ムダを取ることで生産に必要な原材料を最小限に抑えます。また、製品の品質を向上させるために、現場で働く人が自ら作業標準を作り出し、自ら品質向上に取り組みます。つまり、現場主導で徹底的にプロセス改善を継続的に行うことです。また、リーンとは TPS を学んだアメリカの MIT で生まれたものです。そこから派生したリーンマネジメントとは、市場に素早く連動して自律的に動くマネジメントスタイルを実現し、顧客価値を重視してマーケットに素早く対応するために、価値駆動型のマネジメントのことです。Agile や DevOps を採用し導入しているシリコンバレーの革新的なスタートアップ企業は、従来のような計画駆動型のマネジメント手法と相性が合わず、こうした考え方を採用しています。

4.1.2 アジャイルの代表的手法スクラム

アジャイルのスクラムの特徴の 1 つとして、開発の基本として 10 名以下のスクラムチームがあります。これは従来の組織には存在していません。小さなチームを基本としてフラットでオープン、かつシンプルな組織構造を採用しています。（グーグル社の 5,000 の開発チームを参照）
そのチームの特徴は、

- 小さなチーム (small team)
  メンバーが固定化した１０名以下の小さなチーム
- 適応主義 (Adaptive)
  変化に対応する、計画を柔軟に変えるチーム
- 自律的 (Self-Organized)
  権限委譲された自己管理型のチーム
- 機能横断的 (Cross Functional)
  広範な知識を持った自己完結型チーム

こうした開発体制は、TPS やリーンの考え方そのものです。人やチームのコラボレーションにフォーカスし、シンプルなコミュニケーションを行う密結合の小さいチーム、それに干渉領域の少ない疎結合のチームとチームの関係性、彼らが同時平行的にソフトウェア開発を進めます。リードタイムは１か月以内、ベストは１週間で素早く開発を繰り返します。モチベーションの高いやる気のあるチームだからこそ、短いリードタイムで、システムを反復的で漸進的に開発を行い、かつ、最高の品質を実現することができます。

4.1.3 スクラムと人的負債の解消

こうした特徴のある小さなスクラムチームによる反復開発を導入するということは、マネジメントスタイルの変革

が求めらます。つまり、従来の統制型マネジメントから新たな自律型マネジメントへの変革が必要になります。

統制型マネジメントとは、従来のコマンド&コントロール（指示と制御）といったマネジメントスタイルのことであり、詳細な指示をだすことでチームを導くスタイルです。これをマイクロマネジメントと呼ぶ人もいます。

一方、自律型マネジメントとは、チームが意思決定を行うように権限委譲し、彼らを最後まで信頼し結果を出すまで支援するマネジメントスタイルです。ビジョンを明確に伝えた上で権限委譲することでチームを導くマネジメントのことです。自律型マネジメントでは、ビジョンの完全な共有が重要なポイントになります。この自律型マネジメントによって、スクラムチームが自律化するよう育成・強化することになります。つまり、前で述べた人的負債の克服を、こうしたマネジメントによって解消することができます。つまり、現場のエンジニアの働き方の変革です。

4.1.4 プロジェクトマネジメントの変革

今までのシステム開発は、開始と終了が明確になっているプロジェクト型で進めています。アジャイルでは終了が原則として無い継続型になります。つまりライフサイクル型プロジェクトに変えることになります。従来では、課題を解決するためのシステム全体構造を描き、開発する具体的な機能を定義し、それに必要な費用、体制、期限を明確にします。それを精緻に文書化します。その後、プロジェクト計画（開発作業スケジュール）を作成して、その計画通りに作業が進むように管理・制御するのが基本です。

アジャイル開発でも従来と同様にプロジェクト計画は作成しますが、計画作りの相違点は、何を作るべきかにフォーカスする点です。ビジネス目標と顧客価値を重視するためです。どんな機能を作るべきかはプロジェクトを進めながら探究し調整をします。前提条件となるリソース（チーム）と短い開発期間の中で、優先順位の高いものから開発を継続的に進めながら、そこから生れる経験と学習というフィードバックを生み出しながら継続的に価値の探究をします。同時に様々な作業のプロセス改善を継続します。つまり、詳細な作業計画は１か月以内だけの計画になります。

つまり、アジャイルを導入することは、プロジェクトに対する大きな価値観の変革が伴います。このことは企業の経営システムである事業計画・予算計画の仕組みにも影響を与えます。１年毎の経営業績を重視する株主重視型経営システムから、長期的な経営業績、社員の成長を重視する経営システムへの変革が求められます。

4.2 DevOps の導入

DevOps は、アジャイル開発の発展形として導入されます。DevOps とは技術的側面だけではなく、文化的、組織的な実践アプローチとして様々な思考方法、あるいは管理手法を１つに収束させたものであり、アジャイル同様、TPS、リーン、制約条件理論（TOC）の原理・原則に依拠しています。その他にも、レジリエンス工学、学習する組織、セーフティカルチャー、などの知識体系などにも準拠し、また、高信頼マネジメント文化、サーバントリーダーシップ、組織的チェンジマネジメントなどからも影響を受けています。そうしたものが１つに収束することで、未だかつてない低コストと低労力で高い品質とスピードを手に入れています。IT バリューストリーム全体を通じてフローを加速するとともに、高い信頼性を確保します。先ほどのグーグル社の事例は、そのことを実証しています。

ただし、DevOps にはアジャイルのスクラムのような確定的なフレームワークがありません。そこで、それに代わるいくつかの重要なポイントを記します。

4.2.1 DevOps の定義とは

DevOps とは、様々なものが収束した包括的アプローチですが、そのことで捉えづらいものになっています。以下は、IBM 社の Eric Minick 氏の言葉を借りて、DevOps の全体を要約します。

- DevOps はビジネスの成功を支援するために存在する
- 適用範囲は広いが中心となるのは IT サービスである
- 基本はアジャイルとリーンである
- 組織カルチャーが非常に重要である
- フィードバックがイノベーションの原動力である
- 自動化を最大限に活用する

重要なのは、DevOps とは適用範囲が広く最終的には全社展開できるという点にあります。

4.2.2 DevOps 導入のための知識体系とケイパビリティ

DevOps の導入のために、どのようなケイパビリティが求められるのか。以下になります。

- 変化に素早く適応するアジャイル開発
- 開発から運用への継続的デリバリー
- 変化に素早く適応する IT サービスマネジメント
- TPS/リーンの組織カルチャー（人と組織のあり方）

つまり、技術的、人的環境の上にエンドツーエンドのプロセス改善を行い、品質とスピードを実現することです。

5. おわりに

Agile さらに DevOps を導入することで、IT 部門のバリューストリームすべての改善や変革を促すことで、新たな情報システムの価値が生れると考えます。さらに、全社展開することで顧客価値の最大化、ビジネス目標の最大化を図ることができると考えます。それを期待しながらおわりにしたいと思います。

参考文献

[1] 中田喜文, 日本のソフトウェア産業と技術者の現状を国際的に評価する： ソフトウェア技術者の 5 カ国調査結果の分析, SEC journal Vol.13 No.4Mar. 2018

[2] 大野 耐一, トヨタ生産方式――脱規模の経営をめざして, ダイヤモンド社, 1978/5/1

[3] Gen Kim, Jez Hamble, Patrick Debois and John Willis　The DevOps Handbook　ペーパーバック 2021/11/30