情報システム学会研究発表大会原稿
DevOpsの導入と効果

原　清己
Kiyomi Hara
イノベーション・エッジ（Innovation Edge）代表

要旨

多くの日本の企業情報システムが大きな技術的負債を抱え、身動きできない状況の中でそれを解決するための方策も見いだせない状況にあります。一方では、企業のデジタル化の波であるデジタルトランスフォーメーション（DX）時代が到来し、情報システムの新たな価値の創造が企業の命運を左右する時代になりました。しかし日本の IT 業界の対応は遅々としていて、今や米国などに比べて周回遅れの状況となっています。従来の一括請負型のサービス形態がいまだに主流となり、ユーザー企業の外部依存体質はいまだに続いています。

現代のようなデジタル化社会では、IT 投資をしていない企業はどこにもありません。言葉を変えるとすべての組織が IT 企業であり、またデジタル企業だと言っても過言ではありません。GAFA と呼ばれる米国のデジタル先端企業は、すべての企業がアジャイル開発と DevOps を導入している企業群であり、こうした企業の迅速なビジネス展開、高い成長性、あるいは時価総額の大きさに驚かされます。こうした動向を見ても、今後はアジャイル開発・DevOps の導入が IT サービスをビジネスの成長に連動させるためにも必須であると考えられます。こうした先端事例は実証的な証拠として我々にその有効性を示しています。

本文

１．現状の課題認識

経済産業省が２０１８年に「DX レポート～IT システム「2025 年の崖」の克服と DX の本格的な展開～」と題する報告書を発行しています。その中で以下が指摘されました。

多くの経営者が、将来の成長、競争力強化のために、新たなデジタル技術を活用して新たなビジネス・モデルを創出するデジタル・トランスフォーメーション（DX）の必要性について理解しているが、以下のような課題点を抱えている。

- 既存システムが、事業部門ごとに構築されて、全社横断的なデータ活用ができでいない。
- 過剰なカスタマイズがなされているなどにより、複雑化・ブラックボックス化している。
- 経営者が DX を望んでも、データ活用のために上記のような既存システムの問題を解決し、そのためには業務自体の見直しも求められる中（＝経営改革そのもの）、現場サイドの抵抗も大きく、いかにこれを実行するかが課題となっている

この課題を克服できない場合、DX が実現できないのみならず、2025 年以降、最大１２兆円／年（現在の約３倍）の経済損失が生じる可能性がある（2025 年の崖）。

情報システムのサイロ化が進んだことで横断的なサービスができていない点、あるいは柔軟性に欠けるモノリシックなソフトウェアであることの弊害が指摘されています。こうした課題に対して、新たな方向性と解決策を見いだす必要があります。ただし、今までの延長線上からはこうしたものは生まれてこないと考えています。

２．DevOps 導入事例

まず、米国の最新動向から見ていきます。要旨にもご紹介しましたが、DX 時代の口火を切った GAFA（ガーフャ：Google, Apple, Facebook, Amazon.com）と呼ばれる米国を代表する先端企業がありますが、この４社以外にもデジタル技術を最大限に利用した企業、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が、すべての企業が IT サービスによってビジネスが牽引されています。これらの企業のほぼ全てがアジャイル開発、その発展形である DevOps を導入している企業群でもあります。例えば、グーグルは 2013 年に、アマゾンは 2011 年に DevOps を導入したと公表しています。導入したことで、何が実現されたのかを以下にご紹介します。

Google のワーナー・ボーゲル氏の講演（2013 年に DevOps を導入）から抜粋
・>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

この事例で驚くべきことは、5 秒毎に新機能がリリースされ、毎月全体の半分のコード変更が行われ、１日に１億テストケースを回している点です。その開発スピードとリリース頻度は信じがたいものです。それを実践しているプロジェクト体制は、5,000 もの非常に多くのプロジェクトチームが平行して開発を行っています。Amazon もこれとまったく同様の発表をしていますが、開発プロジェクトのチーム編成を 2 Pizza rule と呼んで、2 枚のピザを分け合えるチームを編成していると言っています。つまり、10 名以下の小さなチームでプロジェクトを編成しています。Google や Amazon のような企業は、こうした開発アプローチを採用することで上記のような成果を達成しています。この DevOps を日本でも展開できれば、「2025 年の壁」という問題も解決できるのではないかとの思いから、このレポートを作成することに致しました。

４．IT サービスの価値基準

Google や Amazon が達成した数値に驚くのでなく、どのような目標を定めてこのような数値を達成したのかを考えることが大事です。企業のビジネス目標に貢献し、企業のビジネスを牽引することが IT サービスの本質であり、そのことが IT サービス価値の最大化になります。現代のような変化の激しいデジタル化社会では、いかに早く、ジャストインタイムで IT サービスを提供できるか、また、頻繁な変更にたいしても安定稼動を維持できることが重要になります。つまり、IT サービスを測定するための評価基準を変えるべきだと考えます。

IT パフォーマンスの評価基準（メトリックス）

- 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）

IT パフォーマンスの向上を行うためには、このような新たな評価基準を設定し直す必要があります。そうすれば、IT 部門の責任や役割に対する考え方が自ずと変わってきます。従来のように、それぞれのプロジェクトの QCD 達成を目的にする前に、この新たな規準を追い求めるべきです。それは、従来のプロジェクト志向の考え方から、プロダクト志向あるいはサービス志向へシフトすることになります。プロダクト志向、サービス志向になるということは、IT サービスのライフサイクルマネジメントの考え方を導入することだと考えます。この観点があれば、今のような技術的負債を抱え込むことはないと考えています。

このことは、IT 部門に存在する製品企画、ソフトウェア開発、品質保証（QA）、システム運用、情報セキュリティといった IT バリューストリームすべてに対する変革を促すことになります。これら IT サービスのマネジメントプロセスを顧客価値の最大化の観点から見直す活動に繋がると考えます。

３．DevOps とは

３.１ DevOps の定義

DevOps という言葉が生まれたのは 2009 年であり、まだ新しい言葉であると言えます。この言葉は Development と Operations の造語として生まれました。その発端は Velocity2009 というカンファレンスで 2 名の若き技術者 John Allspaw 氏と Paul Hammond 氏が、“10 Deploys per Day : Dev and Ops Cooperation at Flickr” の講演を行ったことから始まりました。しかし、その定義も様々で 100 名いれば 100 通りの定義があるとまで言われています。それは、DevOps が非常に広範囲なものを扱っているためで、その解釈がそれぞれの人の経験や立場を反映したものになっているからだと言えます。ここでご紹介するのは代表的な定義の１つとして IBM 社の Eric Minick 氏（IBM Program Director）の言葉です。以下のように具体的に表現しています。

・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
・文化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
・自動化を活用する

DevOps とは IT サービスに関する広範囲な活動を含んでおり、ビジネスを支援する IT サービスの価値を最大化することを目的とした一連の活動の総称が DevOps だということになります。従って、組織や人によって醸成される文化が非常に重要になります。カーネギーメロン大学のソフトウェア工学研究所の言葉をご紹介すると、”DevOps is not Tool, DevOps is People, People use Tool.” と言っています。したがって、特定の技術論、方法論、あるいは技法、ツールのみを論じるものではありません。

３.２　DevOps のカルチャー（組織、人の振舞い）

DevOps を実践するには、人々の働き方、振る舞い、あるいは組織の在り方が変わらないとなりません。そのためには、以下のようなリーンの基本思想を基にした新しいカルチャーの醸成が非常に重要であると言われています。それを具体的に表現すると以下になります。

１．顧客中心の原則と行動
　　顧客中心のマインドセットと官僚主義の排除
　　変革へのチャレンジ、リスクを取る勇気、行動を始める勇気
２．目標に対する共有意識
　　目標を共有してコラボレーションの促進
　　プロジェクト思考からプロダクト思考、あるいはサービス思考へのシフト
３．全体に対する責任と役割
　　エンドツーエンドの全体責任を共有
　　全体責任を共有した多能工
４．役割共有の自律化（自己組織的、機能横断的）チーム
　　明確な役割分担を排除したチーム
　　クロスファンクショナル（機能横断的）なチーム
５．継続的な改善活動
　　継続的に仕事を改善し、ムダ（７つのムダ）を削減する活動
　　すべての手作業を完全に自動化する活動

こうした文化の大きな特徴はオープンなカルチャーです。組織から官僚主義、階層構造を無くすことです。サイロ化した縦割り組織の壁を打破し顧客への価値を最大化するために、すべての人が協力し協調

し、すべての人が自ら考え、自ら行動できるようなカルチャーを生み出すことです。また、この事は、DX 時代に求められる素早い意思決定と素早い行動、つまり IT サービスのスピードある変化を実践するためには必須になります。

３.３　DevOps の知識分野

DevOps の導入には文化を含めて、どのようなケイパビリティが必要とされるのでしょうか。それは開発から運用までの IT 部門に存在するすべてのバリューストリームを変革するものになりますが、その中で重要なのが以下になります。

１．変化に対応できる規律あるアジャイル開発
２．開発から運用へのスムーズでムダのない移行プロセス
３．柔軟で信頼性のある事業継続性を保証する IT サービスマネジメント
４．リーンをベースとしたカルチャー（組織と人のあり方）

３.３.１　規律あるアジャイル開発

「システム変更に要する所要時間」を短くするにはアジャイル開発が必須となり、従来のウォーターフォール型の開発では所要時間を短縮することが困難です。DevOps 導入にとって、このアジャイル開発を確実に回すことが重要な鍵を握っています。アジャイル開発は、そのプラクティスを学ぶことは簡単だと言われていますが、これを確実に実践することが難しいと言われています。その理由は、人が変わらない限り規律あるチームにならないからです。人や組織が変わるためには適切なファシリテータとマネジメントサポート、それに人や組織を変えるための一定の時間が必要になります。以下のような規律あるアジャイル開発チームを構築することが重要です。

１．機能横断的で権限委譲された自律的なチーム
２．全体目標を共有し役割分担を明確にしない多能工のチーム
３．開発状況、進捗状況が完全に見える化された職場とそれを実践するチーム
４．短い反復期間でバグフリーの高品質なソフトウェアを開発できるチーム

何を導入していればアジャイル開発か、と言われる議論がありますが、わたしは技法にこだわる必要は無いと考えています。上記の４つの実践ができているかどうかが最も重要です。とは言え、これを確実に回すためには、以下のプラクティスに則って反復的でインクリメンタルなソフトウェアを開発すると、より確実に実践できるようになると思います。

１．アジャイルプロジェクト管理手法としてのスクラムの導入と実践
２．開発技法としてエクストリームプログラミング（XP）の導入と実践（以下の技法の導入）
　テスト駆動開発、リファクタリング、ペアプログラミング、継続的インテグレーション

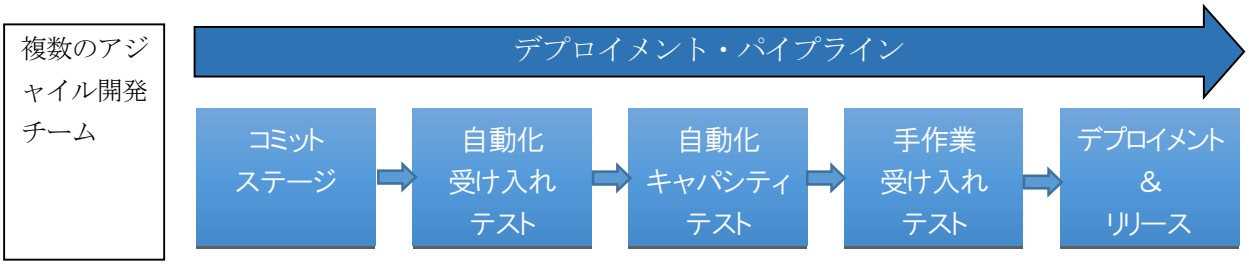
３.３.２　開発から運用への移行プロセス

「システム変更に要する所要時間」を短くし、開発から運用へのスムーズな移管を行なうことで、「本番へのリリースを行う頻度」を頻繁にすることができます。複数のアジャイル開発チームが平行的に開発をし、本番環境にスムーズでムダなくデプロイできれば、リリース頻度を最大限に引き上げることができます。この実践により一日に何回もリリースを行なうことができます。そのためには移行プロセスを完全に自動化し、本番への移管規定、変更規定などを変えなくてはなりません。

DevOps では、開発と運用を連携するために、以下の図のようなデプロイメントプロセスを自動化したパイプラインの構築が必要となります。従来の品質保証（QA）プロセスを自動化したデプロイメント・パイプラインの構築です。これによって、開発から本番環境へのフローを最大化することができます。ただし、これを導入する為には以下のような前提となる条件が必要になります。

１．開発工程に継続的インテグレーションを導入

２．デプロイメント・パイプラインのテスト自動化とテストプロセス改善
３．全てのソースを構成管理しバージョン管理システムを導入（Everything as Code）
４．同じシステム環境の構築（開発環境、テスト環境、ステージング環境、本番環境）



注意）書籍「継続的デリバリー」by Jez Hamble, David Fairley から引用したものを編集

３.３.３　事業継続性を保証する IT サービスマネジメント

IT サービスマネジメントの世界では、世界標準の ISO/IEC 20000 があります。そのベストプラクティスとして ITIL (Information Technology Infrastructure Library)があります。多くの日本企業もこの ITIL を導入してきました。ただし、そのまま ITIL の厳格なプラクティスを導入すると様々な弊害が生じます。それは、頻繁なリリースを行なうためのスピードを阻害することになるからです。DevOps の観点から考慮点を上げます。

１．常に変化する環境に素早く適応するためには、厳格なプロセス記述は変更に対処する際の妨げになる可能性があることを理解する。
２．開発チームと運用チームが協業するサービスチームを編成し、引継ぎの回数を少なくし、複雑なプロセス記述を不要にし、ドキュメントを最小限に抑える。
３．サービスチームは可能な限り自律的に活動し、他チームとの調整や第三者の承認なしに、サービスをリリースするために必要な活動を実施する。
４．プロセスをより迅速に、より信頼性を高く、再利用を可能にするために、自動化することに焦点を置く。
５．サービスデザイン、サービストランジション、サービス運用、およびその品質に対して、すべて個々のサービスチームが対応する。そのチームが対処しなくてはならない項目に関しての全体像は ITIL を尊重する。
６．サービスストラテジに関連する内容は、すべてのサービスチームが全員で話し合うものとする。

以上のように、プロダクト志向、サービス志向のアプローチを採用することで、チームがサービスに関する全責任を負うことが重要です。

３.４　DevOps の導入と展開

DevOps を導入するための第一段階として、IT サービスのバリューストリームである製品企画、ソフトウェア開発、品質保証（QA）、システム運用、情報セキュリティといったエンドツーエンドのプロセスのフローを最大化します。左から右への所要時間を短くすることです。そのためには、タスクや成果物のバッチサイズを小さくすることも重要になってきます。

第二段階として、より頻繁により多くのフィードバックをリアルタイムで発生させる情報フローの流れを構築し、そのフィードバックによってイノベーションを起こす原動力にすることです。事前に問題を解決し、事前に兆候を発見し、破壊的な結果を引起す前に問題を避ける努力をします。組織的な学習と改善の文化を作り出し、学習した内容を組織として活かすことです。失敗が起きたときにも責任追及や非難をする理由や対象とせずに、学習の良いチャンスと捉えます。

第三段階として、安全性の向上、継続的な改善、および学習を支える実践を根づかせます。その為に、

安全性を実現するジャストカルチャー（公正な文化）を確立し、ローカルな発見があればグローバルな改善につなげ、組織的な改善と学習のための時間を確保します。ある１つの組織で学習した新しい知見を、すばやく組織全体で活用する仕組みを作ります。

以上のような３段階で順番に導入することで、競合他社より早く学ぶべきことを学び、安全で回復力のある企業文化を作り出し、一人一人の潜在能力を最大限に引き出せるようにすることで、変化に迅速に対応できるスピードある組織を作ることができます。

４．現行システムの再構築

では、どうすれば複雑化・ブラックボックス化した既存システムをモダナイズすることができるかに立ち戻ってみます。ただこの点については、現実的には銀の弾などないとは思います。ただし、反復的にインクリメンタルな方法で既存のモノリシックなソフトウェアを、重要性の高い機能から順番に置き換えていくことです。小さなサービスコンポーネントから構成される疎結合なソフトウェア構造に置き換えていくことです。そのための継続した IT 投資を続ける必要があります。システム投資をプロダクトライフサイクル全体で考えることが必要だと思います。

また、従来からのソフトウェア開発に対する考え方を変えないとなりません。大型のプロジェクトを組んで一時に再構築するような事をしていては今までの失敗の繰り返しです。ここで参考になる考え方として Conway の法則があります。「ソフトウェアのどの部分も、それを作った組織の構造を反映する。」というものです。力づくで行っても Conway の法則は破れるものではなく、組織はアーキテクチャに影響を与え、複雑な組織は複雑なアーキテクチャを作り出します。その逆に、シンプルな組織はシンプルなアーキテクチャを作り出します。自律化したシンプルで小さなチームによる反復的でインクリメンタルな開発活動により、良いアーキテクチャ構造のシステムに移行できると考えます。インフラ環境だけのモダナイズでは、問題の解決にはなりません。

５．まとめ

限られた紙面の中で、限られた要点しか表現することはできませんでしたが、最後のまとめです。DevOps は、基本的に「リーン」や「トヨタ生産方式」の考え方を拠り所にしており、その他に、レジリエンス工学、学習する組織、セーフティカルチャー、ヒューマンファクターから得られた知識体系を取り入れています、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などの考え方からも影響を受けています。このように DevOps は、IT サービス価値の最大化を達成するために、様々な技術的、構造的、文化的なアプローチを採用し、多くの思考法や理論、あるいは管理手法が最終的に１つに収束した結果として生み出されています。そうした結果として、未だかつてない低コストと低労力で、最高の品質、信頼性、安定性、セキュリティを実現し、製品企画、ソフトウェア開発、品質保証（QA）、システム運用、情報セキュリティといった IT バリューストリーム全体を通じてフローを加速し、かつ信頼性を確保することができます。DevOps が企業のビジネスを牽引していることは、DevOps を導入した先端企業によって実証されその証拠として示されています。

参考文献

[1] Gen Kim, Jez Hamble, Patrick Debois and John Willis　The DevOps ハンドブック 理論・原則・実践のすべて　長尾高弘訳　榊原彰監修　日経 BP 社 2017/6/22
[2] Jez Hamble, David Farley　継続的デリバリー 信頼できるソフトウェアリリースのためのビルド･テスト・デプロイメントの自動化　和智右桂、高木正弘訳 KADOKAWA　2017/7/31
[3] Mary and Tom Poppendieck リーン開発の本質　高嶋優子、天野勝、平鍋健児訳　日経 BP 社 2008/2/7
[4] Kent Beck, Cynthia Andres　エクストリームプログラミング　角征典訳 オーム社 2015/6/26
[5] 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