Introducing DevOps

Adopting DevOps to achieve Continuous Innovation

IBM Worldwide Lead – DevOps Technical Sales
Executive IT Specialist, IBM Software Group
Sanjeev Sharma

ここ数年、ソフトウェア・デリバリー機能に対する要求は日々厳しさを増してきています。

- ●顧客はアジリティー（俊敏性）を求めている
- ●開発者は毎日大量の機能を開発している
- ●運用担当者は要求に応じて環境を準備している
- ●顧客は使用中のどのプラットフォームからでも、最新の機能にアクセスし、それらを使用できることを求めている

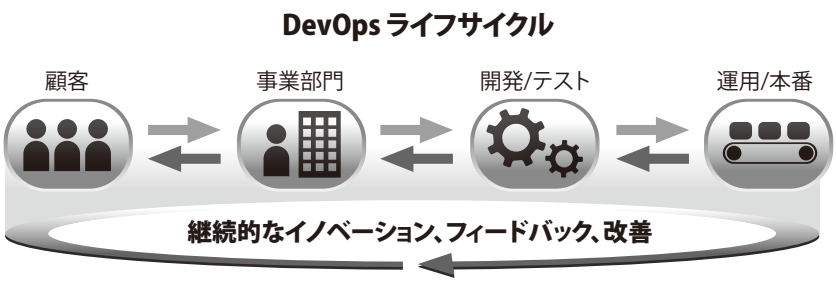
これらは、DevOpsムーブメントの原動力となっている要因の一部に過ぎません。今日のアプリケーションは複雑化しています。顧客側に表示されるのは思いどおりの操作を可能にする最新のインターフェースですが、これらのアプリケーションは、顧客の要望に合ったビジネス機能を提供するために、エンタープライズ・アプリケーション、既存のデータウェアハウス、および外部のサード・パーティー APIにアクセスする必要があります。

DevOpsは、IT業界の他の新しいテクノロジーや技術関連ムーブメントと同じように、一つの流行語（buzzword）となりました。誰もがDevOpsという言葉を口にしますが、それが一体何であるかを必ずしも理解しているわけではありません。何よりも困るのが、DevOpsを主張する人たちの多くがそれをうまく実践できていないことです。DevOpsムーブメントにおいて素晴らしい成果を残し、その最先端をいく優れた企業はいくつかあります。DevOpsの代表的な存在としてよく引き合いに出されるのは、Etsy社や、Facebook社、そしてNetflix社でしょう。しかし、これらの企業でも、DevOpsを実現する最も優れた手法については、いまだ議論と検討が続けられています。Netflix社が取り組んでいるのは、開発者が運用担当者の職務を引き受ける「NoOps」です。しかし、このような状況は無秩序を招き可視性が失われる可能性があるとして、大企業に適用できる方法ではないという反論の声も上がっています。業界はいま、DevOpsとは何かを定義し、各企業におけるリスク、価値、ならびに要件のバランスに基づいてさまざまな手法を確立しつつある状況で、こうした議論には期待が寄せられています。

本稿では、DevOpsの概要を説明し、組織がDevOpsによって実現できるビジネス価値を探り、DevOpsを導入するためのアプローチを示します。

DevOpsとは

DevOpsとは、リーンおよびアジャイル原則に基づくソフトウェア・デリバリーの手法です（**図1**）。事業部門から開発、品質保証、運用に至るまですべての利害関係者が協力して、顧客から実際に寄せられたフィードバックに基づき、より効率的にソフトウェアを提供することを目的としています。DevOpsの機能と原則を導入し実施することによって、デリバリー・プロセスを継続的に改善し、顧客から実際に寄せられたフィードバックに基づいてソフトウェアの変更や拡張ができるため、ソフトウェア・アプリケーションの効率的なデリバリーを実現できます。



- ●ソフトウェア・デリバリーの加速
- ●スピード、コスト、品質、およびリスクのバランス
- ●顧客からフィードバックを得るまでの時間の短縮

図1. DevOpsの概念

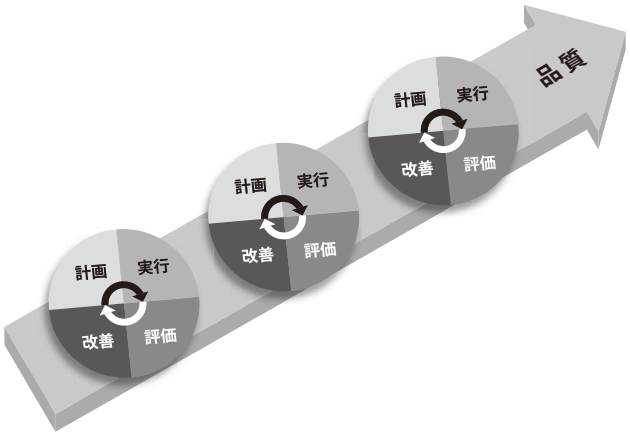


図2. 継続的な品質の改善

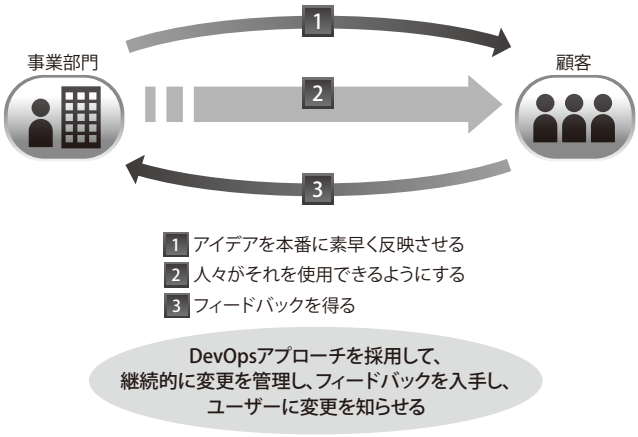


図3. リーン原則を適用したDevOps

デミング、リーン、およびカイゼン（Kaizen）

これらの原則は目新しいものではありませんが、リーン生産方式のムーブメントが起こった時期に、ウィリアム・デミングなどの思想リーダーによって業界に広まった原則が基になっています。デミングが提唱したのは、継続的な生産品質の改善を目的とした、計画：Plan、実行：Do、評価：Check、改善：Act（または調整：Adjust）からなるPDCAサイクルです（**図2**）。リーン生産方式では、この核となる手法を基に生産する製品を継続的に改善し、その生産プロセスにおいて無駄を減らすことを目指しています。日本の自動車業界ではこれらの原則を数十年前に導入し、「カイゼン」（改善）と名付けました。

近年のソフトウェア開発業界では、これらのリーン原則をソフトウェア開発に適用した「アジャイル方法論」が導入されています（**図3**）。基本的な考え方は、一定の期間（スクラム方法論ではスプリントと呼ばれます）に区切られた反復作業の中で、その期間ごとにソフトウェアとして動作する小規模なプログラムを開発し、顧客やその代理人からすぐさまそれに関するフィードバックを受け取り、開発作業を繰り返すことによって、提供するソフトウェアの要件を満たすというものです。アジャイル方法論には、その手法の基になる4つの重要な価値基準があります。これらはアジャイル・マニフェストで次のように表されています。

1. プロセスやツールよりも個人と対話を
2. 包括的なドキュメントよりも動くソフトウェアを
3. 契約交渉よりも顧客との協調を
4. 計画に従うことよりも変化への対応を

DevOpsでは、このリーン手法とアジャイル手法の基本的な考え方を、ソフトウェア・デリバリー・ライフサイクル全体、およびすべての利害関係者（開発と品質保証［QA］だけでなく、事業部門［LOB］から運用に至るまで）に適用しています。

DevOpsムーブメントは、その名のとおり、組織とチームが開発（Dev）と運用（Ops）の間にある障壁やすれ違いの解消を求めたことから始まりました。このような組織間におけるコミュニケーションや信頼の欠如は、ソフトウェアのデプロイに課題を残しました。このため、ソフトウェアをデプロイするプロセスは、「あまり頻繁に実施したくない、複雑で混乱を招く厄介な作業」と考えられるようになりました。一方で、コミュニケーション不足により、開発者と事業部門が、実際にソフトウェアを使用する顧客からフィードバックを受け取る作業も煩雑化しました。このため、顧客の運用環境でソフトウェアを使用した場合の動作とパフォーマンスに応じた対応プロセスは困難な作業となりました。

そうした中、Systems of Engagement（市場や顧客とつながるシステム）アプリケーションのニーズに後押しされて、DevOpsムーブメントの勢いは加速しました。このようなアプリケーションでは、継続的に、かつ素早く変更を適用し、フォーカスすべき製品市場の需要の変化に対応する必要があります。ソフトウェアを使用するのは顧客自身であり、通常そのソフトウェアを開発することは、顧客にイノベーションを提供するチャンスとなります。

DevOpsのすべての原則に共通する目的は、開発と運用の間にある障壁を取り除き、コミュニケーションと信頼を確立することです。これにより、小規模な一連の機能をできるだけ早く顧客の運用環境で稼働させ、それに関するフィードバックを受け取ることが可能になり、開発部門ではそのフィードバックを基にソフトウェアの調整と処置を行い、さらにソフトウェア・デリバリー・プロセスも強化して、無駄なやり直しを減らすことができます。

DevOpsに関するIBMの見解

DevOpsが成熟するにしたがって、その原則はソフトウェア・デリバリー・ライフサイクル全体に適用され、顧客や事業部門にもフィードバックされるようになりました。これにより、PDCAサイクルの「調整および改善」プロセスは、ソフトウェアの機能だけでなく、リリース計画、要件、テスト、環境、ソフトウェア自体のデリバリー・プロセスにも影響を与えるようになりました。ソフトウェア・デリバリーに関しては、PDCAサイクル全体があらゆる部分に適用されま

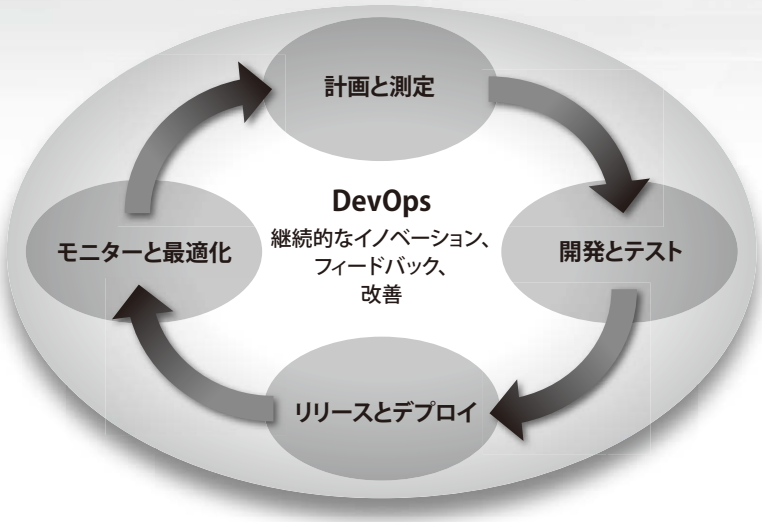


図4. 4つのDevOps導入アプローチ

す。こうしたDevOpsの成熟により、その原則を、Systems of Engagementアプリケーションだけでなく、従来のSystems of Records（記録するためのシステム）アプリケーションにも適用できるようになりました。市場が急速に進化し、競争が激化する中で、企業は顧客に提供する無数のアプリケーションにおいてイノベーションと安定性のバランスを維持しようと努力しています。

IBMではDevOpsに関して、この総体的な見方を採用しています。IBMによるDevOpsの定義を以下に示します。

“市場機会を捉える時間と顧客のフィードバックを得る時間を短縮することを可能とするための継続的なソフトウェア・デリバリーを実現する企業規模の能力”

IBMが考える4つのDevOps導入アプローチ

DevOpsは、ソフトウェア・デリバリー・ライフサイクル全体に及ぶ、広範な一連の機能によって構成されます。組織がどの部分からDevOpsの導入を開始するかは、その組織のビジネス目標、つまり解決しようとしている課題や、強化が必要なソフトウェア・デリバリー機能によって異なります。

組織がDevOpsを導入する際に役立つよう、IBMでは4つの導入アプローチを提案しています（**図4**）。それらは以下のとおりです。

●計画と測定
●開発とテスト
●リリースとデプロイ
●モニターと最適化

1.計画と測定

この導入アプローチには、事業部門とその計画プロセスに重点を置いた「継続的なビジネス計画」という一つの実施項目が含まれます。企業は高いアジリティーを維持し、顧客からのフィードバックに素早く対応する必要があります。このため、今日では多くの企業がリーン原則に基づく手法を採用しています。この手法では、初期段階では規模を広げず、ビジネス上のビジョンや価値をテストするために必要な成果とリソースを特定した後、顧客からのフィードバックに基づいて継続的に対応と調整を行います。

ビジネス目標を達成するには、進捗状況を把握し、顧客が本当に必要としているものを特定した後、それに従ってビジネス計画を更新することによって、方向性を決定していきます。これにより、顧客はリソース上の制約のある環境においても利点と欠点を比較しながら、継続的な判断を行うことができます。

2.開発とテスト

この導入アプローチには、コラボレーティブ開発と継続的なテストという二つの実施項目があります。これらを実施することにより、開発および品質保証（QA）機能の根幹が形成されます。

<コラボレーティブ開発>

ソフトウェア・デリバリーには、部門の枠を超えた多くのチームが関与します。これには、顧客、ビジネス・アナリスト、エンタープライズ・アーキテクト、ソフトウェア・アーキテクト、開発者、品質保証担当者、運用担当者、セキュリティー専門家、サプライヤー、パートナーなどが含まれます。これらのチームの作業担当者は、複数のプラットフォームで作業を行うため、別々の場所に分散して配置される場合があります。コラボレーティブ開発では、一連の

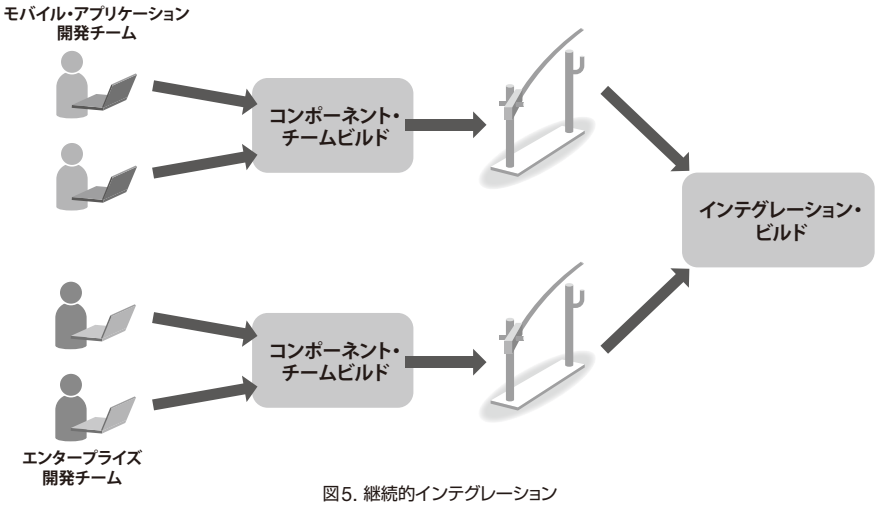


図5. 継続的インテグレーション

共通実施項目と、ソフトウェアの作成およびデリバリーに使用できる共通のプラットフォームが提供されるため、これらの担当者は協力して作業を行うことができます。

コラボレーティブ開発に含まれる一つの重要な機能は「継続的インテグレーション」（**図5**）です。これを実施することにより、ソフトウェア開発者は作業成果を継続的に、または頻繁に、開発チームの他のメンバーの成果と統合できます。継続的インテグレーションはアジャイル・ムーブメントによって広まった考え方で、開発者が作業成果を定期的に他の開発者の成果と統合し、その上で統合された成果物をテストすることを目的としています。複数のシステムとサービスから構成される複雑なシステムの場合、開発者は作業成果を他のシステムやサービスとも定期的に統合します。作業成果を定期的に統合することで、統合によるリスクを早期に発見できます。複雑なシステムでは、技術とスケジュールの両方に関する既知および未知のリスクを確認することもできます。

<継続的テスト>

継続的インテグレーションには、以下のようないくつかの目的があります。

- ●継続的なコードのテストと検証を可能にする
- ●作成したコードを、他の開発者が作成したコードや他のアプリケーション・コンポーネントのコードと統合し、それらが仕様どおりに動作するかどうかを検証する
- ●開発中のアプリケーションを継続的にテストする

継続的テストとは、ライフ・サイクル全体にわたって、早期かつ継続的にテストを実施することを意味します。これにより、コストを削減し、テスト・サイクルを短縮するとともに、品質に関するフィードバックを継続的に受け取ることができます。このようなテストを実施するには、自動化されたテストやサービスの仮想化といった機能を導入する必要があります。サービスの仮想化は、運用環境に近い環境のシミュレーションを行うための新機能であり、継続的なテストの実施を可能にします。

3.リリースとデプロイ

リリースとデプロイは、DevOpsの根幹をなす多くの機能を提供する導入アプローチです。継続的インテグレーションの概念は、継続的リリースとデプロイによって次のステップへと進みます。これにより、リリースとデプロイだけでなく、デリバリー・パイプラインの作成も可能になります。デリバリー・パイプラインを使用すると、品質保証、それに続く運用環境への展開といったソフトウェアの継続的なデプロイを効率的かつ自動化された方法で行うことができます。継続的リリースとデプロイの目的は、新機能をできるだけ早く顧客とユーザーにリリースすることです。

DevOpsテクノロジーの核となるツールやプロセスの多くは、継続的インテグレーション、継続的リリース、継続的デプロイの促進を目的としています。

4.モニターと最適化

モニターと最適化は二つの実施項目を含む導入アプローチであり、これらを実施することによって、企業ではリリースしたアプリケーションが運用環境でどのように動作するかをモニターし、顧客からのフィードバックを受け取ることができます。このデータは、迅速な対応や必要に応じたビジネス計画の変更を行う上で役立ちます。

<継続的モニタリング>

継続的モニタリングでは、データとメトリクス（評価指標）が運用担当者、品質保証担当者、開発者、事業部門、デリバリー・サイクルのさまざまな段階でアプリケーションに関与するその他の利害関係者に提供されます。これらのメトリクスの対象は運用環境に限定されません。利害関係者はこうしたメトリクスに従って、提供する機能やそれらの提供に必要なビジネス計画を拡張または変更できます。

<継続的フィードバックと最適化>

ソフトウェア・デリバリー・チームが入手できる最も重要な情報は、「顧客がどのようにアプリケーションを使用するかを示すデータ」と「そのアプリケーションを使用した顧客からのフィードバック」です。企業では、新しいテクノロジーを使用することによって、顧客のアプリケーションの操作パターンや問題点を正しく把握することができます。このフィードバックに基づき、さまざまな利害関係者が適切な対応を取ることによって、アプリケーションを改善し、顧客の操作性を向上させることができます。事業部門はビジネス計画の調整、開発部門は提供する機能の調整、運用部門はアプリケーションのデプロイ先となる環境の拡張などを行います。この継続的なフィードバック・ループはDevOpsに不可欠な要素であり、これによって企業は、高いアジリティーを維持し、顧客の要求に素早く対応することができるのです。

まとめ

DevOpsでは、リーンとアジャイルに基づく一連の実施項目が提供されます。組織ではこれらを使用して、ソフトウェア・デリバリー機能の効率と効果を高め、リスクを低減できます。その結果、顧客に最適なタイミングでイノベーションを提供し、その顧客からのフィードバックに基づいて、提供するイノベーションを迅速に強化できるようになります。今日のようなソフトウェア主導の世界では、ほとんどの組織が自社のソフトウェア・デリバリー機能によって顧客の要求に対処しています。その際、顧客が必要とするビジネス機能の安定性と、市場が求めるイノベーションとのバランスを取る必要があります。DevOpsによって、組織はこの微妙なバランスを維持し、市場を獲得することができるのです。