エンタープライズ DevOps

2017 年 12 月

著者紹介:
戸田　孝一郎
株式会社戦略スタッフ・サービス　代表取締役
一般社団法人　TPS 検定協会　理事

三井　伸行
株式会社戦略スタッフ・サービス　取締役
一般社団法人　TPS 検定協会　認定 TMS 講師

目次

エンタープライズ DevOps
DevOps によるビジネスを支援する IT サービス

1. はじめに

私たちはアジャイル開発や TMS のコーチングの提供を通して経験を積み重ね、DevOps は真にビジネスを支援するものである、と確信するに至りました。DevOps は IT のための物ではありません。ビジネス戦略やビジネス・プロセスを支援するための物です。DevOps に関する色々な書籍が出版されておりますが、それらの書籍は、その著者の経験から、彼らの視点で、開発系の人は開発中心に、運用系の人は運用を中心に、管理者の人は組織やプロジェクト管理を中心に書かれています。これらは誤りではありませんが、DevOps の全体に渡るプロセスをしっかりと説明できていません。

このホワイトペーパーは、ビジネス・プロセス中心にDevOpsの全てのプロセスを網羅した内容です。たぶん読者の皆様はこのホワイトペーパーを読んで頂くことで、DevOpsの全体像を明確に描けるようになるでしょう。なぜDevOpsのプロセス中心にこのホワイトペーパーを書いたのか？　その理由は、DevOpsとはITサービスのサプライチェーンを構築する事であり、また全体のプロセスを通して全体最適な運営を行うものだからです。そして管理者はこの開発から運用に至る全プロセスを整流化する仕組みを構築し、プロセスの成熟度を高めてビジネスを真に支援する必要があるからです。

プロローグ　DevOps への道筋

我々の DevOps プロセス構築は、2009 年までさかのぼります。あるクライアント企業で我々が指導してアジャイル開発（スクラムと XP）を開発部門に導入しました。その導入は大変うまくいき、開発期間も開発コストも従来に比べて半減する結果を得ました。
ある日アジャイル開発を導入したその事業部長が、私の所に来られて問題を提起しました。事業部長は、次のようにおっしゃいました。

> 「アジャイル開発を導入して、開発部門の生産性は飛躍的に向上し、作業期間もお陰様で半減しました。アジャイル開発に取り組んだのは、今まで開発期間がかかるとかバグが多くて稼働が遅れるとか、開発以外の人から開発を問題視する意見が多かったので、私はアジャイル開発を導入して期間短縮と品質向上を狙ったのです。確かに開発作業に対しては、当初目論んでいた以上の効果が得られました。しかし残念な事に事業部のビジネス・スピードは今までと変わらないのです。開発期間が短縮されれば、自ずとビジネス・スピードが上がると目論んでいました。ところがその様にはならず、ビジネス・スピードは従来と何ら変わりません。何故でしょうか？これではアジャイル開発を導入した価値が得られません。」

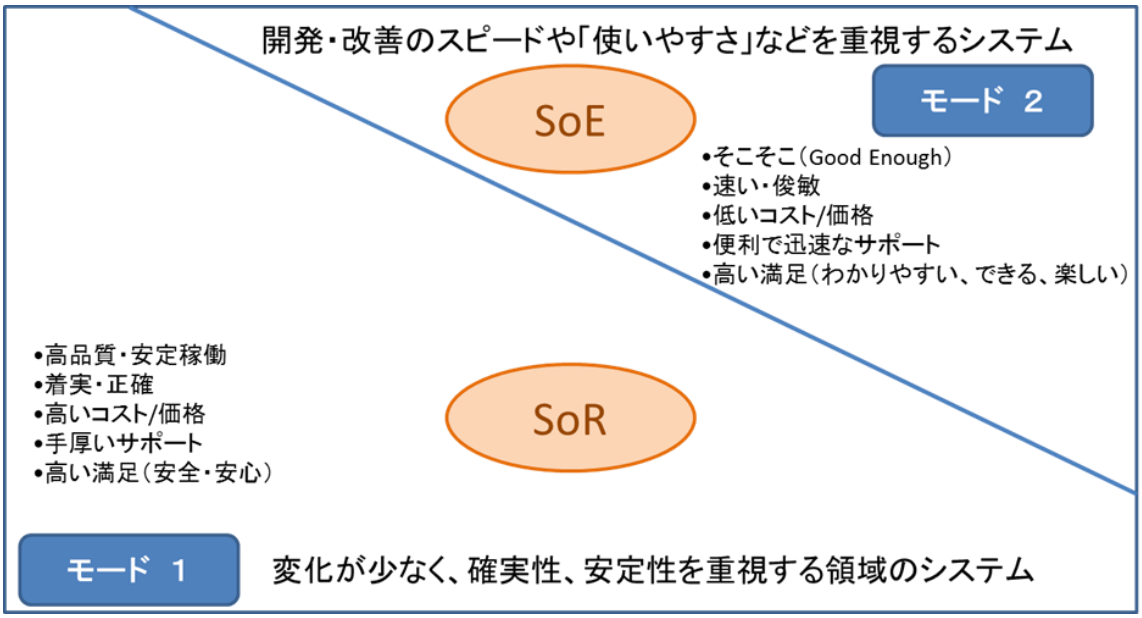
今まではこの事業部のビジネス・スピードを上げる阻害要件（ボトルネック）は開発作業だと信じていましたが、どうやら異なるようです。開発作業がビジネスのスピードを上げる阻害要件ではなかったのです。

そこで、事業部長と我々は、ビジネス全体のプロセスについて詳細に検討を加えました。そして、まだそれほど有名ではありませんでしたが、継続的デリバリーとか DevOps と言う新しい考え方が世の中に出て来たところでしたので、この考え方を導入して全体プロセスを全体最適で整流化する仕組みと働き方を検討しました。丁度我々の身近にこのような考え方で大成功を収めている日本の企業がありましたので、その企業の仕組みを事業部長と共に検討し、導入するプロジェクトを開始しました。
その雛形になったのは TMS（Total Management System; TPS(トヨタ生産方式)のコア思想）です。そこでトヨタで採用されている大部屋システムをその事業部に導入し、DevOps のプロジェクトが 2012 年に完成いたしました。この仕組みは全プロセスの端から端に至るまで全てのプロセスを見える化し、迅速、適宜な意思決定が出来る仕組みを構築してアジャイル開発チームのスピードを殺

さない前後工程（プロセス）の運営方法を確立しました。その結果、一週間のスプリント（イテレーション）で開発するアジャイル・チームのスピードに合わせて運用やお客様サービス、更には営業、企画、管理の部門まで同期化したオペレーションができるようになり、この事業部のビジネス・スピードは 3 倍まで早くできました。同じ事業部要員数で、同じ期間で売り上げは 3 倍以上に急成長出来たわけです。
これこそ DevOps がビジネスに貢献する真の効果です。単に開発と運用が協調する事が DevOps ではありません。従って DevOps が上手くできたかどうかの判断は、単にリリースまでの時間が早くなるとか、日に何回も頻繁にリリースできると言った IT 視点での評価では無く、ビジネスの成果（売上、利益等）で評価されるべき物です。

2. 企業における DevOps とは何か？

やアプリケーション・ライフサイクル管理の視点も考慮しなければなりません。ですから DevOps の名称の通り、開発(Development)と運用(Operation)が対等に協力して新しい IT サービスをタイムリーに提供し続ける仕組み（プロセス）を構築して、そのプロセスの成熟度を高めていく活動です。ガートナーのレポートではバイモーダル（Bimodal）と言う形でこのように説明しております。
不確実性の高まるビジネス環境において、事前に要件を完全に固めることはできません。そのような状況にあってもビジネス現場のニーズに臨機応変に対応し、俊敏・迅速に開発や変更の要求に応えなくてはなりません。そのためには、アジャイル開発や DevOps を積極的に取り入れ、この状況に対応するしかありません。変化が少なく、確実性、安定性を重視する領域のシステム（モード１）では、「現場の要求は中長期的に変わらない」ことを前提に要求仕様を固められますので、ビジネスとソフトウエア開発を一旦切り離して作業を進めることも可能です。しかし、変化を許容する開発・改善のスピードや「使いやすさ」などを重視するシステム（モード２）は、IT サービスとビジネスの一体化がすすんでいることもあり、ビジネスの現場とシステムの開発は密であることが求められます。そして、IT の立場からビジネスへの提案をしてゆくことが、これまでに増して求められるようになります。つまり、ビジネスの成功にどう貢献すればいいのかを IT の立場で考え、そのゴールを業務の現場と共有して開発や保守、運用をすすめてゆかなければならないのです。



モード 1 のシステムは、効率化によるコスト削減を目指す場合が多く、人事や会計、生産管理などの基幹系業務が中心となります。そして、次の要件を満たすことが求められます。

- ·高品質・安定稼働
- ·着実・正確
- ·高いコスト/価格
- ·手厚いサポート
- ·高い満足（安全・安心）

一方、モード 2 は、差別化による競争力強化と収益の拡大を目指す場合が多く、IT サービと一体化したデジタル・ビジネスや顧客とのコミュニケーションが必要なサービスが中心となります。そして、次の要件を満たすことが求められます。

- ·そこそこ（Good Enough）
- ·速い・俊敏
- ·低いコスト/価格
- ·便利で迅速なサポート
- ·高い満足（わかりやすい、できる、楽しい）

3. DevOps のゴール、その目指すもの

DevOps のゴールは、ジャストインタイム（JIT）での IT サービス提供プロセスを確立する事です。従って DevOps を導入する狙いは、ビジネス成果を上げる事です。例えば売上、利益、スピードを最大化させ、コストを極小化させる事です。別な表現をすれば、IT サービスのサプライチェーンを構築する事です。
ですから、従来の考えの様な『ソフトウエア（プログラム）を届ける』という考え方から、『IT サービスを提供し続ける』という考え方への大きなパラダイムシフトが要求されます。DevOps をアーキテクチャや技術側面で見ると、多種多様な手法やツールが紹介されております。これは自働化されたデプロイメント・パイプラインを構築する事の様に見えます。ですが、DevOps はこのような自働化されたデプロイメント・パイプラインを構築する為のテンプレートやツールを提供するものではありません。各々の組織において、自身の DevOps プロセスを構築し、そのプロセスを整流化させて、よどみなく IT サービスをビジネスに届け続ける IT サービス・サプライチェーンを構築するものです。
従って、DevOps とは、個々の手法やツールを選定して導入することではありません。DevOps の目的、意義をよく理解し、プロセスの効率を高めていく”プロセス改善”を行う事です。ですから DevOps の評価（成功/失敗）は、プロジェクトの成果（期間や費用等）では無く、対象となるビジネスの成果（業績）その物で、判断する必要があります。

4. DevOps に必要な知識領域

DevOps を導入する為には、種々な知識、手法、ツール等を選定する必要がありますので、IT の開発、運用にまたがる総合的な知識が要求されます。
DevOps を成功させる基本は、3 つの大きな知識領域です。この 3 つの知識のどれが欠けても DevOps プロセスを完成させる事はできません。

i. **規律あるアジャイル開発（Disciplined Agile）**

規律あるアジャイル開発は DevOps 成功の最大の要素です。
規律あるアジャイル開発を一言で説明すると、次の三点が継続できるチームです。

1. 安定したベロシティー（持続可能なペース）で開発できる
2. 常に変化への対応（対処）が出来る
3. 常にバグフリーな高品質なプログラムをリリースできる。

ビジネスに対応して頻繁なかつ素早いリリース・サイクルを維持しようとすると開発スピードが重要ですが、品質を高める努力が更に重要です。このような状態を創り出す為には、

出来るだけ小さな、均一化されたタスクの定義が求められます。アジャイル開発チームがタスクの粒度にかかわる姿勢が重要です。

また質の良いプログラム（IT サービス）をタイムリーに提供する為にはアジャイル開発チームの作業は、TPS で言う所の自工程完結（JKK）での仕事に臨む姿勢を求められます。これは DoD（Definition of Done; 完了の定義）をしっかりと決め、チーム全員がこのルールを厳守する事です。

また、品質を高める自工程完結での開発作業には、アジャイル・チームの定義するタスク粒度をできるだけ細かく定義できる事が成功要因です。理想的には 1 時間のタスクの大きさにタスク定義を均一化する事です。このような品質に拘りを持てる開発チームが自律したアジャイル開発に繋がります。

一方、プロダクトオーナーは単にプロダクトバックログの優先度や機能を管理するだけでは無く、更に運用時のコスト（手間）も加味して IT サービスの優先順位や、プロダクトバックログの粒度も考慮する必要があります。

ii. 継続的デリバリー（Continuous Delivery）

継続的デリバリーとは、ビルドを作成し、デプロイ、テストを行ってリリースするまでの一連のプロセスを自動化するパイプラインを構築する事です。

更に重要な事は、この様な自動化されたプロセスを構築した後に、継続的にプロセスの品質、成熟度を高める改善、改良を行う事です。特にテスト工程（受入テストやパフォーマンス・テスト）についてはテスト・プロセス成熟化の手法である*TPI NEXT®が参考になります。

このデプロイメント・パイプラインは各々の組織で異なりますので、これと言ったテンプレートや既成のツールが存在するわけではありません。それぞれ自身に合ったデプロイメント・パイプラインを設計、構築する必要があります。ただ一つの重要な共通点は、このデプロイメント・パイプラインは単一のプロセスで構築する事であり、複数存在させない事です。

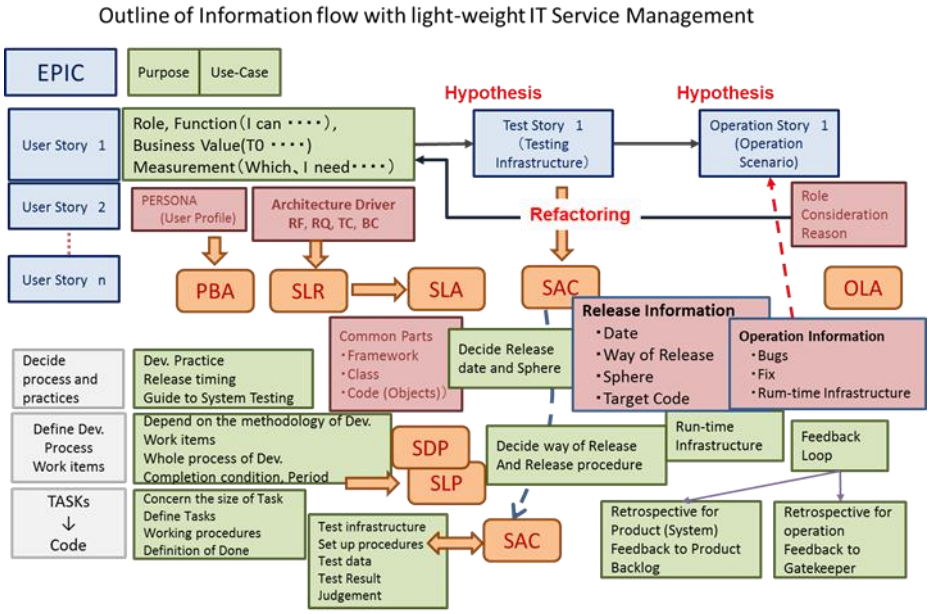
iii. 軽量化された IT サービスマネジメント（ITSM）

軽量な IT サービスマネジメントの特徴はアジャイル開発のスピードを殺さずに、ビジネス継続性を担保できる必要最低限の管理が出来る事です。その上で、ITIL の中でビジネス継続性に関係するデータ、すなわち PBA（Patterns of Business Activity）、SLR（Service Level Requirements）、SLA（Service Level Agreements）、SDP（Service Design Package）/SLP（Service Level Package）、SAC（Service Acceptance Criteria）、OLA（Operational Level Agreements）についてデータの発生源、発生時期を検討しアジャイル開発プロセス、DevOps プロセスのどのステージに対応するかマッピングしました。

軽量化された IT サービスマネジメントの基本的な考え方は、運用に必要な最低限の情報（MRI；Minimum required Information）を、そのデータ発生源で発生した時点で記録（ログ）しておき、後に管理に必要となった時点で、そのデータを整理して IT サービスマネジメントとして利用するという考え方です。できるだけデータ発生源での記録に手間がかからない方法でこれらを行い、本来の開発やその他の作業の邪魔にならない様にすべきだと考えています。

このようにこまめにメモを取る程度の負担で記録できる事が重要です。従って軽量な IT サービスマネジメントとは何か文書や管理の手順を規定するものではありません。後に管理で必要となる情報やデータを特別な労力を掛けずに記録・保持しておく仕組みです。例えばアジャイル開発チームが通常のプロセスの中で作成するメモや情報類（チケット等）を、そのまま捨てずに記録保持する事です。

軽量化された IT サービスマネジメントの概略図



iv. TPS (リーン) の考え方が基本

よどみのない流水化した IT サービス・サプライチェーンを構築するという事は、従来までのそれぞれの担当部門のサイロを壊し、一気通貫のプロセス・フローを作り、同期の取れたプロセスの運用管理を行う事になります。組織（担当部門）のエゴを排除して全体最適でのプロセスの運用を行うために、TPS（トヨタ生産方式）の概念を利用すると成功します。

TPS の基本は、ご存知の通り JIT（ジャストインタイム）と自働化と言う重要な二つの概念をプロセス内に埋め込む事になります。

DevOps プロセスでの JIT の概念の導入は、とりもなおさず一個流しのプロセス・フローを確立する事です。一方、自動化の概念の導入とは、単にプロセスを自動化（オートメーション）して自動で流れるプロセスを構築することではありません。自働化と表現をあえて変えているのは、自動化されたプロセスに、自律神経を埋め込むことを意味しています。自律神経を持った自動化プロセスとは、何かプロセス内の作業で不具合が発生した、もしくはその恐れが生じる事を感知した時に、全てのプロセスを停止できる仕組みを埋め込む事です。この仕組みにより不良を作らず、不良を流さない自工程完結での仕事が実現します。

また TPS の概念の導入で、異なる管理サイクルの仕事であっても全体最適なプロセス構築で、全ての関係部門の同期を取る事ができます。プロセスに関わる全ての関係部門が常に全体最適の視点で行動できる様になります。

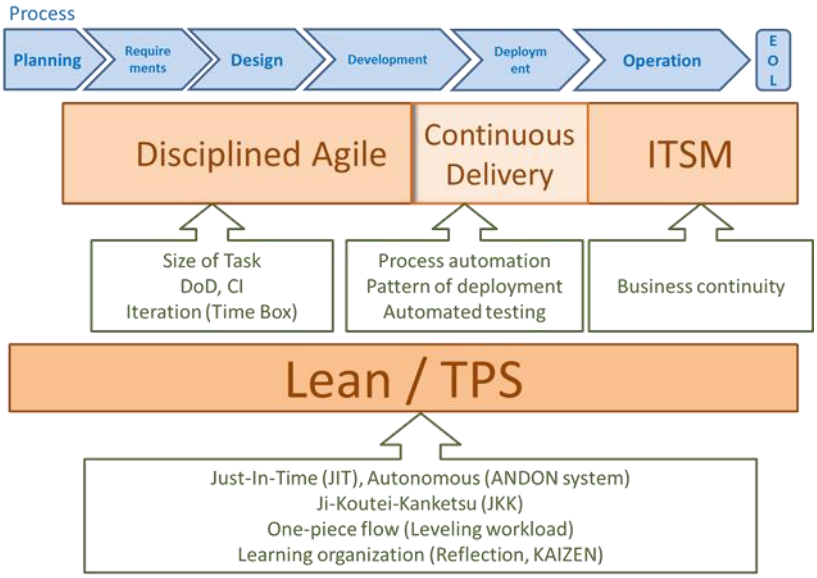
以上 4 つの知識領域が、DevOps プロセスを構築し運営する為に必須です。

2010 年にカルフォルニア州、マウンテンビューで初めて開催された DevOpsDay で John Willis 氏と Damon Edwards 氏が CAMS を提唱し、その後に Jez Humble 氏によって L が加えられた CALMS モデルで言われて居る様に、これらの知識領域が DevOps 成功への基本的条件といえます。

CALMS モデルとは、Culture, Automation, Lean, Measurement, Sharing の五文字の頭文字を並べたものです。

- Culture - コラボレーションとコミュニケーションを推進するための変革を負う
- Automation - バリューチェーン内から手作業のステップをなくす
- Lean - リーン原則の採用により，サイクル周期を高める
- Metrics - すべてについて計測し，データをサイクルの洗練化に活用する
- Sharing - 他者が学べるように，経験や成功の可否を共有する

DevOps の知識領域概念図



5. DevOps チーム — 新たな役割

DevOps チームを編成する場合には、コミュニケーションが充分に取れる小さなチームを編成されることを推奨します。丁度アマゾン社で言われる「2 ピザパイのルール」が良いでしょう。（2 つのピザで食事できる人数；10 名以内）

新たに生まれる役割について次に表します。

プロセスマスター:

スクラムでのスクラムマスターの役割と同様にチームをファシリテートして最大のパフォーマンスを出すようにリードします。また全プロセスを『見える化』して、一個流しのプロセスの流水化を指導します。

大部屋の設営について全責任を持ちます。

推奨経験；スクラムマスター、アジャイル管理者、PRINCE2

サービスマスター:

IT サービスをジャストインタイム（JIT）でユーザー（ビジネス）に提供する全ての責任を持ちます。

ちょうどスクラムでのプロダクトオーナーに似て居ますがプロダクトオーナーの責任に加えて、IT サービスを提供するコストについても責任を持ちます。

推奨経験；プロダクトオーナー（スクラム）、サービスオーナー（ITIL）

DevOps エンジニア:

自働化されたプロセスの構築及び維持管理の責任を持ちます。

自働化された DevOps プロセスの設計、ツールの選定と導入、プロセスの維持・管理に関する仕事をします。

推奨経験；アジャイル開発、各種開発ツール、管理ツール、テストツール.

ゲートキーパー / リリース管理者:

運用状態の監視と開発中、テスト中の IT サービスの進捗状況の管理に責任を持ち、最終的にリリースの可否判断に責任を持ちます。

勿論、セキュリティー、コンプライアンス、各種規制、運用チームの能力等を勘案して最終的なリリース可否判断をします。大事な事はリリースする IT サービスが現在の運用に与える影響を考慮することです。

推奨経験；IT サービスマネジメント、運用管理

リライアビリティーエンジニア:

ディプロイメント・プロセスにおける特にテスト品質について責任を持ちます。そしてテスト・プロセスの成熟化を指導します。

継続的インテグレーション（CI）、継続的デリバリー（CD）に対して開発チームへの支援、指導を提供します。

推奨経験；ソフトウエア・テスト、品質管理、テストツール

開発チーム:

DevOps の重要な成功要因は、自律したアジャイル開発チームです。自律したアジャイル開発チームとは、リリース計画に合わせて、品質を確保したプロダクト（IT サービス）を持続可能なペースで提供できるチームです。

推奨経験；アジャイル開発、オブジェクト指向、テスト

運用チーム:

軽量化された IT サービスマネジメントに基づき、設計、開発、導入の各プロセスに適宜フィードバックできるチームです。
特に ISO20000 の各種情報の欠落を指摘、フィードバックする事が、IT サービスの品質を高める事になります。

これは TPS での先行改善にあたります。積極的なプロダクトバックログへの提言、提案が求められます。

推奨経験；運用管理、改善

6. DevOps の組織

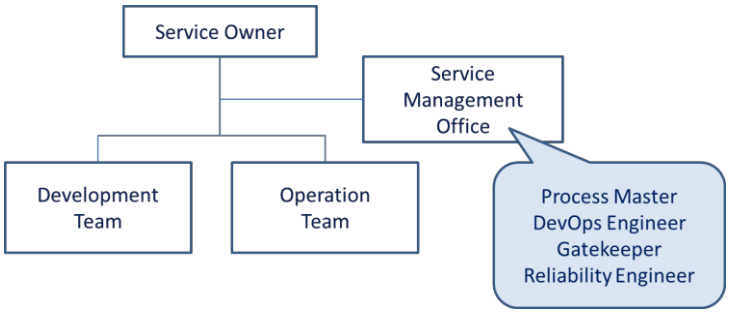
DevOps を推進する組織形態はどの様な形が望ましいでしょうか。

基本的にはサービスマネジメントオフィス（SMO）と言う形態のスタッフ部門として、DevOps プロセスの参加する人、チームを支援する形が考えられます。特にサービスマスターを支援する、従来のプロジェクト管理オフィス（PMO）をイメージして頂ければ良いと思います。

次に二つの形態例を示します。

i. 小規模でフラットな組織

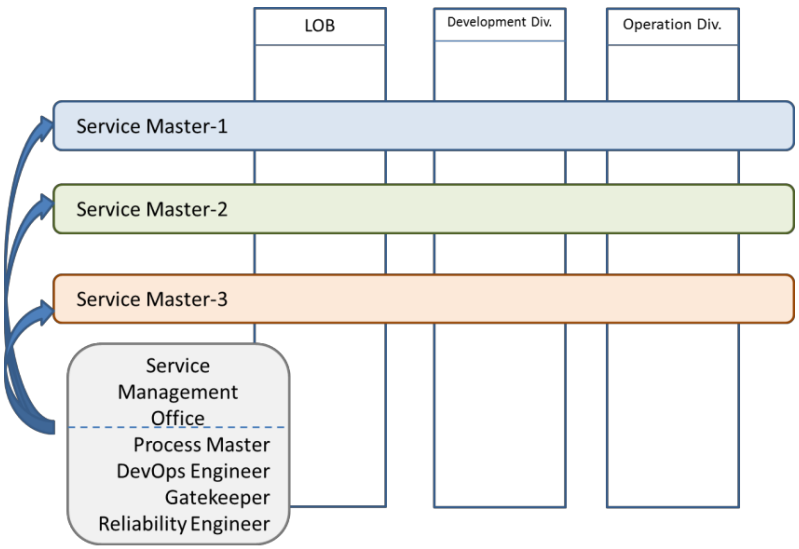
小規模な組織での代表的な例です。
開発チーム・運用チームを、サービスマスターを支援しながら指導するスタッフ組織



ii. 大規模な組織でのマトリックス組織

DevOps チームのエキスパートを集めて、プロジェクト（サービスマスター）毎にチームを編成して支援する形態

この形態はトヨタの主査制度（チーフエンジニア）の形態と同じです。



7. DevOps プロセス

全体最適での整流化された DevOps プロセスを構築する上で、最も重要な視点は、自工程完結（JKK）での作業（仕事）の習慣を DevOps チーム全員に浸透させる事です。

自工程完結（JKK）とは、チーム全員がチームの目標を共有し、その実現のために自分の担当する作業（仕事）を正しく行った事を確認して次のプロセスに渡す事です。自分の行った作業（仕事）が正しいのか？　次のステップ（プロセス）に渡して良いのか？　を判定できる基準を各ステップ（プロセス）の完了時点で持つことです。このような仕組みが実装されないと、DevOps の全体プロセスを整流化（流水化）できません。

では DevOps の主要なステップ（プロセス）の順を追って説明ます。

i. ビジネス戦略/企画

現代の IT サービスはビジネス戦略や企画と大変密な関係性を持っています。特に SoE のシステムでは、IT サービス戦略/企画その物がビジネス戦略である場合も多いでしょう。したがって DevOps チームのサービスマスターは、このビジネス戦略立案フェーズから関与する必要が出てきます。そしてビジネス戦略立案時に、どの様な IT サービスを提供する事がビジネスの差別化や戦略上の優位性を保てるのか、技術的・ビジネス的見地から提案することが望まれます。

ii. マーケティング＆販売企画

サービスマスターはこの段階でも積極的に関与する必要があります。推奨する IT サービスがどの様に販売戦略上優位性を保たれるのか？競合企業の戦略と比較してどの様な優位性を顧客に提供できるのか？IT サービスの基本戦略を策定すると共に、顧客や販売部門の要望をシステムに実装して提供できる IT サービス要求に変換します。この時点では、提供しようとしている IT サービスのビジネス価値を明確化し、その実現時期を合意します。

iii. 企画/管理

DevOps チームのプロセスマスターは、まず対象となる自身が担当する DevOps プロジェクトの全てのプロセスに渡って『見える化』する仕組みを構築します。ここでは、TPS（TMS）の大部屋方式の導入が大変有益です。

『大部屋』とは、関係する全てのプロセスを一目瞭然に把握できる『見える化』されたプロジェクトに関する全ての情報を一か所で全て把握でき、その上でその場で決断を下せるようにしなければなりません。この仕組みにより効率よく JIT でプロジェクトに関するあらゆる意思決定が正しく行えます。この大部屋には、プロジェクトを遂行する上で必要な全ての情報が『見える化』されて収集、蓄積されそれらの情報は適宜更新されていなければなりません。

『大部屋』の効果は、迅速、機敏な情報収集で、現状に合った現地現物での意思決定ができ、チーム内外ステークホルダーとの円滑なコミュニケーションが図れます、チーム内外ステークホルダー間での問題/課題の早期共有ができます、チームとしての一体感が醸成されます。

iv. プロジェクト計画

サービスマスターはプロジェクトを円滑に実施する自身のスタッフ組織であるサービスマネジメントオフィス（SMO）を編成します。SMO を編成するという事は、単に役割分担をするスタッフを集める事ではありません。

まずチームとしての目標を明確に定義し、その目標達成に向けたビジョンを策定してチーム全員と共有して納得を得なければなりません。そのためのグランド・ルールも策定します。更に対象とする IT サービスの実装環境（運用環境）も事前に定義します。

その上で DevOps プロセス全般に渡ってバリューストリームマップ（どの工程でどの様な価値を作り込むのか？）を作成します。これがこのプロジェクトでのプロセス設計となります。

v. 要求と設計

サービスマスターは、実現すべき IT サービスの要求事項（プロダクトバックログ）を収集、整理して、それぞれに実現するビジネス価値を基にした優先順位付を行います。

DevOps チームはプロダクトバックログから三種類のストーリーを作成します。

開発チームはプロダクトバックログからユーザー・ストーリーを確認します。

開発チームとリライアビリティーエンジニアは協働して、ユーザー・ストーリー毎にその対象となる受入テスト項目からテスト・ストーリーを作成します。

運用チームは、ユーザー・ストーリーからそれが実装された時の運用環境と運用方法をオペレーション・ストーリーと言う形で整理、作成します。

以下に各ストーリーの包含される具体的な内容を記述します。

- ユーザー・ストーリー(User Story):
  対象となるユーザー（役割)、必要な機能（実現したい事)、実現したい理由もしくは実現した時のビジネス価値、使用環境、使用方法

- テスト・ストーリー(Test Story):
  受入テストケース（テスト方法、テストデータ)、SAC のチェック項目

- オペレーション・ストーリー(Operation Story):

  企画されている IT サービスの運用上の優先順位、事業継続性確保の為の条件（EOL の条件)、SLA、OLA の基本的チェック項目し、SLA、OLA を作成

またこのステージで、DevOps エンジニアと運用チームは協力して、移行環境、テスト環境、開発環境を作成します。

開発チームは、リリース計画とイテレーション期間等アジャイル開発チームとしての計画を立案します。

ゲートキーパーは各種規制事項（コンプライアンス・ルール）等が IT サービスに及ぼす影響を検討します。

リライアビリティーエンジニアは、テスト方法、テストケースをテスト・ストーリーから定義します。

vi. 開発

このステージでは、Scrum（スクラム）が最も適した開発手法です。

開発チームはリリース計画をコミットしなければなりません。そのリリース計画を守る上でチームは自律したチームに成長しなければなりません。

イテレーション（スプリント）の期間は、ビジネス・ニーズ（必要とされるスピード）か

ら決められなければなりません。

また品質を確保するという視点からは、XP の各種手法例えばペア・プログラミング、テスト駆動開発（TDD）、リファクタリング、10 分間ビルドなどの利用は、大変有益です。

vii. デプロイメント

開発チームでの継続的インテグレーションが完了した後、自動化された受入テスト、パフォーマンス・テストが実施されデプロイメントのフェーズに入ります。

これらを総称してデプロイメント・パイプラインと言います。

DevOps エンジニアは、このデプロイメント・パイプラインを単一のパイプラインで設計しなければなりません。そして一個流しのプロセス・フローを構築します。

リライアビリティーエンジニアは、DevOps エンジニアの協力を得てテスト・プロセスの成熟化に努めなければなりません。

ゲートキーパーは、運用状況、開発プロセスの進捗などを常に監視（モニター）し、その IT サービスを、この時点でリリースして良いか？最終決定をします。

運用チームは、このステージで、どの様な運用が事業継続性を確保できるか確認します。

viii. 運用

運用チームは、軽量な IT サービスマネジメントプロセスを活用して、常時 IT サービスの稼働状況を監視（モニター）し、重大な危機に直面したならば、即座にリライアビリティーエンジニアと共に対応できる様に準備します。この際二つの重要な情報 RPO(Recovery point objective)と RTO(Recovery time objective)を管理します。

特に運用チームが注意を払う点は、運用の管理サイクルをアジャイル開発のイテレーションと同期が取れる様に短縮する事です。

DevOps プロセスに置いて運用チームに課せられた重大な任務は、各 IT サービスの稼働運用状況、事業継続性の確保と言う視点から、常にサービスマスターや開発チームに対して先行改善要求をフィードバックする事です。このフィードバックは、プロダクトバックログと言う形で行われます。

ix. 保守・維持

サービスマスターとリライアビリティーエンジニアは、ユーザーからの変更要求に対して対象となる IT サービスのメンテナンス（変更）を実施するかどうかの判断を下します。

変更すべきとの意思決定が下された後は、その変更要求は開発チームのプロダクトバックログに追記され、サービスマスターはその優先順位を管理します。

x. カストマー・サービス（コールセンター）

サービスマスターとリライアビリティーエンジニアは、常にユーザー（お客様）の評価に気を配らなくてなりません。ユーザー（お客様）からの IT サービスに対するフィードバック、特に運用上の課題、ユーザー・エクスペリエンスの課題、品質の課題に対して注意を払います。

またカストマー・サービスからも運用チーム同様にサービスマスターに対して先行改善要求や変更要求をプロダクトバックログという形でフィードバックします。

xi. EOL（IT サービスの終焉）

サービスマスターは、稼働中の IT サービスを終焉させる意思決定をします。
これは要求と設計ステージであらかじめ決めた EOL の条件に照らし合わせて意思決定されます。

万が一 IT サービスを延命させる決定を下す場合には、再度 EOL の条件と今後のサービス提供上の条件を明示してビジネス側の了解を得ます。

以上がエンタープライズ DevOps のプロセスです。

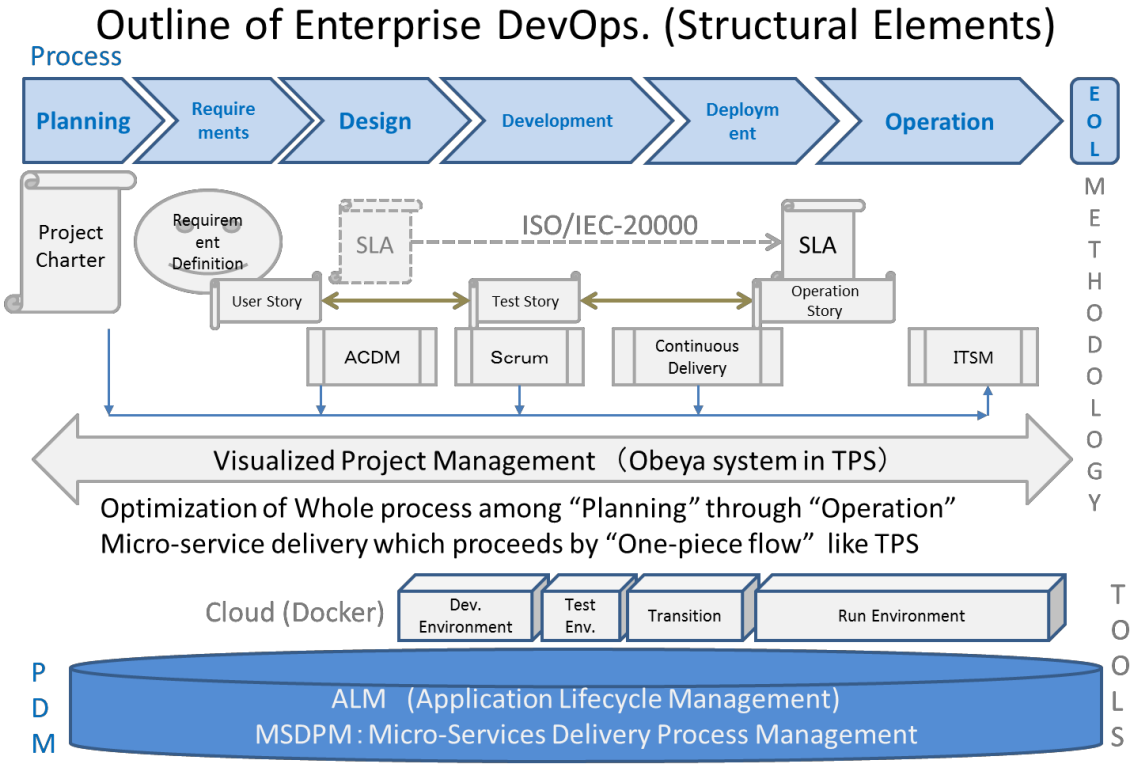
重要な事は、各局面（個々のプロセス）での最適化ではなく、全体の DevOps プロセスを通しての全体最適なプロセス、運営を図ることです。その設定されたプロセスの成熟化を常に行う努力を続ける事が、真にユーザー（お客様）に評価頂ける IT サービスとなります。

何度も触れますが、この設計された DevOps プロセスを一個流しの整流化（流水化）されたプル型のフロー（流れ）を作る事が、ビジネスに対して機敏性を増大させる仕組みになります。

最後にこれら全体の DevOps プロセスが、一か所で全て『見える化』されている事が、DevOps 成功の鍵になります。

注）『見える化』とは、誰でも即座に異常が解る事です。

下にエンタープライズ DevOps 全体像（概念図）を示します。



8. DevOps の導入形態

最後になりましたが、DevOps を企業内に導入する場合に想定される代表的な三つのパターンを紹介します。

i. トヨタウェイ(TOYOTA way)型：（大規模＆複合）

ビジネスオーナー（事業部長）やサービスマスターによって実施される戦略的に企画されたシステム（IT サービス）や複数のビジネスの優位性を確保する複数の IT サービスにて採用されます。大規模な組織（大企業）向けといえます。

導入組織形態としては、マトリックス型組織が適切です。また IT サービス戦略とビジネス戦略が密に関係性を持つビジネスに適します。IT サービスを提供するビジネスを行う企業にはこのような形態が望まれます。

ii. Collaboration 開発・運用協働型：（一般的な DevOps）

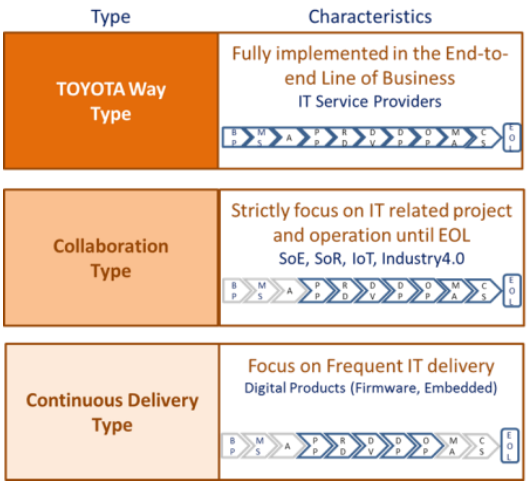
DevOps の代表的な形態です。IT サービスの機敏なそして頻繁なリリースへの対応と信頼性の高い事業継続性を考慮したいプロジェクト向きです。サービスマスターによって実施できます。一般企業での SoE と SoR の連動する仕組み構築に適します。

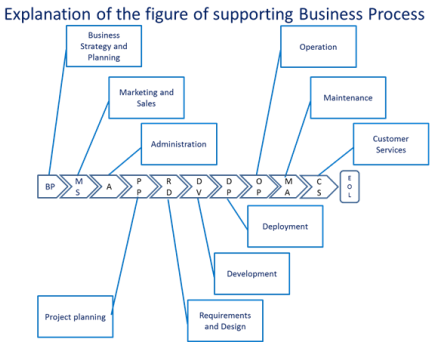
iii. 継続的デリバリー型：（基本形）

上記の二例と異なり DevOps 専任チームを設置せず、従来のアジャイル開発の延長線上に位置します。従って、サービスマスターを設置せずアジャイル開発のプロダクトオーナーがサービスマスターの機能を兼任します。

IT サービスの機敏な、そして頻繁なリリースが主目的となります。主にはデジタル製品メーカーのソフトウエア開発部門などに適します。

下に上記の 3 つのパターンを比較例示します。





9. まとめ

DevOps の導入、実践で成功を収める為には、CALMS モデルで示された 5 つのキーワードが重要です。とりわけ先頭の C(Culture)が大変重要な要素であると、欧米でも言われております。とりもなおさずカルチャーを変える、パラダイムシフトを起こす事が DevOps 成功への第一歩となります。
“DevOps is not Tool, **DevOps is People**, People use Tool. (カーネギーメロン SEI)”
DevOps のゴッド・ファーザー（命名親）である Patrick Debois 氏も『人間が問題だ』と言っています。

推奨する出版物（本）

- The Phoenix Project
  – A Novel about IT, DevOps, and helping your business.
  Gene Kim, Kevin Behr, George Spafford.
  ISBN9784822285357
  (邦訳版) The DevOps　逆転だ！究極の継承デリバリー
  日経 BP 社
  For general understanding

- Continuous Delivery
  – Reliable Software Releases through Build, Test, and Deployment automation.
  Jez Humble, David Farley
  ISBN9780321601919
  For DevOps Engineer, Reliability Engineer, Development team, Process Master.

- Effective DevOps: Building a Culture of Collaboration, Affinity, and Tooling at Scale
  Jennifer Davis, Katherine Daniels
  ISBN9781491926307
  For: Process Master, Development team

- DevOps: A Software Architect's perspective
  (SEI Series in Software Engineering)
  Len Bass, Ingo Weber, Liming Zhu.
  ISBN9780134049847
  (邦訳版) DevOps 教科書
  日経 BP 社
  For Development team, Process Master, Service Master, DevOps Engineer, Reliability Engineer, Gatekeeper.

- The DevOps 2.0 Toolkit:
  - Automating the Continuous Deployment Pipeline with Containerized Microservices.
  Viktor Fracic.
  ISBN9781523917440
  For DevOps Engineer, Development team, Process Master.

- Architecting Software Intensive Systems (ACDM)
  - A Practitioner's Guide
  Anthony J. Lattanze.
  ISBN9781420045697
  For Development team, Process Master.

- DevOps Automation Cookbook
  Michael Duffy.
  ISBN9781784392826
  For DevOps Engineer, Operation team, Process Master.

- The Visible Ops Handbook
  - Implementing ITIL in 4 Practical and Auditable steps
  Kevin Behr, Gene Kim, George Spafford.
  ISBN978097556813
  For Operation team, Gatekeeper, Process Master.

- TPI NEXT: Business Driven Test Process Improvement
  Alexander van Ewijk, Bert Linker, Marcel van Oosterwijk, Ben Visser, Gerrit de Vries, Loek Wilhelmus. Tik Marselis.
  ISBN9789072194978
  (邦訳版) TPI NEXT® ビジネス主導のテストプロセス改善
  株式会社トリフォリオ
  For DevOps Engineer, Reliability Engineer, Development team, Process Master.

- The TOYOTA Way
  Jeffrey K. Liker
  ISBN9780071392310
  For general understanding, Executive level, Manager, Service Master, Process Master.

Contact EXIN

www.exin.com

YOUR ICT COMPETENCE PARTNER