開発部門と運用部門の新たな関係作り・役割見直しにより
サービスリリースサイクルの“超短縮”を実現する

日本アクチュアリー会ＩＴ研究会　平成２９年度　第２Ｇ

【担当委員】

石亀　太郎	第一生命情報システム
須山　恭延	ライフネット生命

【メンバー】

宗岡　匠	明治安田生命
佐野　千紘	朝日生命
小林　正幸	住友生命
大高　武志	三井生命
穴田　絵理奈	アメリカンファミリー
小出　浩史	損保ジャパン日本興亜
Munusamy　Rajinikanth	ＡＩＵ損害
宮野　拓也	第一生命情報システム
中村　雄太	第一生命情報システム
岩本　一記	太陽生命
中川　佳彦	ニッセイ情報テクノロジー
本田　健輔	全共連
片岡　浩基	ニッセイ情報テクノロジー

目次

はじめに
序章．保険会社の利益源としての DevOps
　０．１．研究の目的
　　（１）保険会社における経営のスピードアップはなぜ必要か
　　（２）保険会社における経営のスピードアップには何が必要か
　０．２．研究の対象
　　（１）システム対応のスピードを上げるためには何が必要か
Ⅰ．DevOps への組織的アプローチ
　１．１．システム部門の組織の現状と課題
　　（１）組織の構造
　　（２）運用部門の参画タイミング
　１．２．DevOps の組織的アプローチ例
　　（１）組織構造改革
　　（２）運用部門が上流工程から参加する
　　（３）運用部門と開発部門が課題を話し合う場を設ける
　１．３．結論
Ⅱ．DevOps への技術的アプローチ
　２．１．リリースを早める上での技術的な課題
　２．２．DevOps の技術的アプローチ例
　　（１）継続的デリバリーの概要
　　（２）テストの自動化
　　（３）環境構築の自動化（Infrastructure as Code）
　　（４）デプロイの自動化
　　（５）その他の関連技術
　２．３．技術的アプローチを採用する上での課題
　２．４．結論
Ⅲ．保険業界における DevOps の適用
　３．１．保険業界にも DevOps は必要か
　３．２．保険業界でも DevOps の適用は可能か
　　（１）DevOps への組織的アプローチは、システム構成を問わず導入可能である
　　（２）メインフレーム向けの DevOps ツールが登場し始めている
　３．３．結論
終章
謝辞
参考文献

はじめに

私たちは、「開発部門と運用部門の新たな関係作り・役割見直しにより　サービスリリースサイクルの“超短縮”を実現する」というテーマで、DevOps という手法・思想について、また、保険会社に適用する価値やその実現可能性について研究を進めてきた。

DevOps という考えは、2009 年の Velocity カンファレンスにおいて、米国 Flickr 社の 2 名のエンジニアが行った“10 deploys per day -Dev & Ops cooperation at Flickr-”というタイトルのプレゼンテーションが起源とされる。Flickr 社では、IT 業界において長年、対立構造が課題になっていた開発部門と運用部門の役割を見直し、最新技術を利用した両者の協業を実現することにより、1 日に 10 回を超えるシステムのリリースを可能としたという。Flickr 社のこの思想は、開発を表す「Development」と運用を表す「Operation」の頭文字から「DevOps」という言葉で表され、システムリリースサイクルの高速化を実現するための手法として WEB 業界を中心とした IT 業界に広まっていくこととなった。

システム開発サイクルの高速化においては、しばしばアジャイル開発が取り上げられる。アジャイル開発のスコープは通常、要件定義からモジュールの完成を表し、システムの本番リリースは必ずしも含まれていなかった。一方 DevOps では、開発部門と運用部門の役割、協力体制の見直しを技術の進歩を利用することで実現し、システム開発の開始からリリースまでに潜在する「ムダ」を省いていくことが特徴である。日本国内ではメルカリ、楽天、Cookpad といった IT 先進企業においてすでに導入されている。システムの利便性を高めることが企業価値向上の源となっているこうした企業において、システム開発のスピードは企業の競争力そのものであり、DevOps によってシステムリリースサイクルの高速化を図ることは、そうした企業のニーズとマッチしていた。

保険業界におけるシステム開発のスピードは、Flickr 社やメルカリ、楽天、Cookpad といった企業ほど、競争力そのものに直結するとは考えにくいかもしれない。しかしながら、システムが大規模化・複雑化する中、そのシステムを保守・開発するスピードが、経営が求めるスピードに追いつかないという問題を抱えている保険会社は多い。現状の開発スピードをより速めていくニーズは保険業界全体に存在すると考えられる。

序章では、システム開発のスピードアップの必要性について述べる。本論では、システム開発のスピードアップの有効な手段となりうる DevOps について、実現方法を組織的アプローチと技術的なアプローチに分類し、考察していく。第一章では組織的なアプローチの例を、第二章では技術的なアプローチの方法を論じ、第三章では、そうしたアプローチが保険会社にも必要・有効であるか、また実現可能であるかを検証する。

序章．保険会社の利益源としての DevOps

０．１．研究の目的

「保険会社における経営のスピードアップ」が研究の目的である

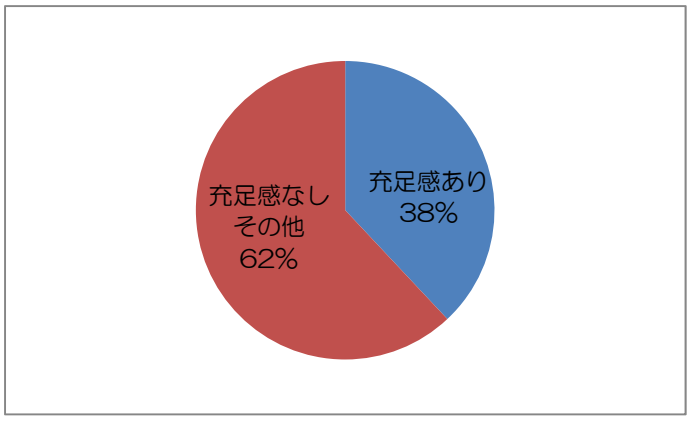
私たちの研究テーマは「開発部門と運用部門の新たな関係作り、役割見直しにより、サービスリリースサイクルの“超短縮”を実現する」という長いタイトルである。研究の目的を表すのは後半部分の「サービスリリースサイクルの“超短縮”を実現する」、つまり、「保険会社のサービスをより早く市場へ展開すること」である。さらにいえば、「保険会社における経営のスピードアップを目指すこと」の必要性と実現可能性を考察することが私たちの研究目的である。

（１）保険会社における経営のスピードアップはなぜ必要か

経営のスピードアップは、保険会社が利益を上げ続けるために必要である

国内の保険市場が飽和状態にある現在、保険会社が利益を上げ続けるためには、経営のスピードアップが欠かせないと私たちは考える。

【図表序-1】加入保険の保障内容に対する充足度



出典：公益財団法人生命保険文化センター「平成 27 年度 生命保険に関する全国実態調査」

【図表序-1】は、平成 27 年に行われた公益財団法人生命保険文化センターによる全国調査の結果である。保険契約者のうち、加入保障に対して「充足感あり」と回答した人の割合はわずか 38%にとどまり、充足感なし、およびそれ以外の回答が 6 割を超えている。同調査によれば、「加入保険の充足感がない」と答えた理由として「支払われると期待していた

給付金が支払われなかった」といった意見が多いという結果が出ている。
昨今の医療技術の進歩は目覚ましく、それに対応する医療保険に対する新たなニーズは常に生まれていく。また、損害保険においても、ドライブレコーダーの普及や、自動運転のモビリティ関連技術の進展が進む中で、保険に対するニーズも変容していくことが予想される。
そのような外部環境の中で、市場のニーズに合った保険商品やサービスをタイムリーに提供し続けることができなければ、上記のような加入者の不満はさらに増し、保険離れが進んでしまうことは想像に難くない。

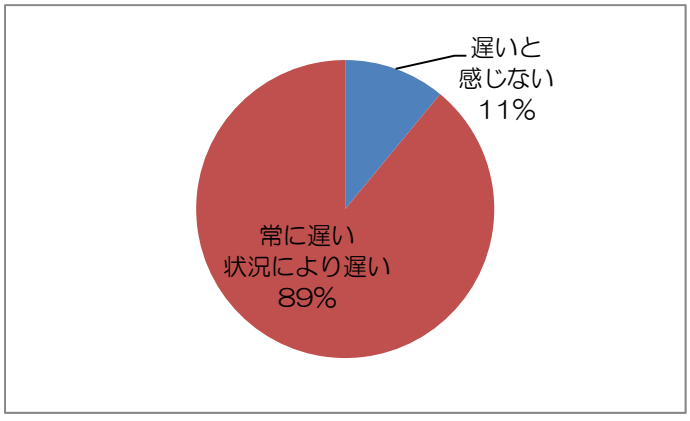
保険会社が利益を上げ続けるためには、新規加入者にとっても魅力的な商品やサービスをタイムリーに提供し続け、加入者の満足度を高める必要がある。そのために、そうした商品戦略・経営戦略をタイムリーに実行できる体制をかまえておく必要がある。

（２）保険会社における経営のスピードアップには何が必要か

システム対応のスピードアップは、経営のスピードアップに不可欠である

保険会社が新しい商品を売り出すためには、保険商品を販売し、契約の保全・保険金支払を正しく行うためのシステムが必要である。新しいサービスを始めるときも、やはりそのためのシステム対応が欠かせない。
つまり、保険会社が新たな商品戦略・経営戦略を十分なスピードで実行に移すためには、システム対応のスピードがそれに追いついている必要がある。

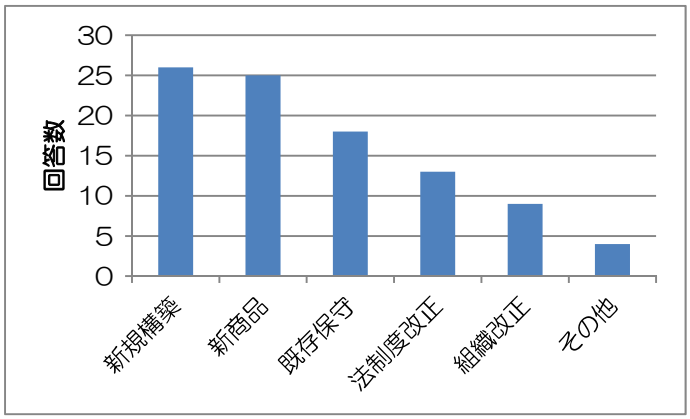
【図表序-2】システム開発のスピードに対する所感



出典：第２Ｇ アクチュアリー加盟企業へのアンケート

【図表序-2】は、我々が実施したアクチュアリー会加盟企業へのアンケートの結果である。自社のシステム開発のスピードに対する所感を尋ねたところ、およそ 9 割の保険会社が「常に遅い・状況により遅い」と感じると回答している。
「常に遅い・状況により遅い」と回答した各社に、特に遅いと感じる案件を尋ねたところ、結果は図表【図表序-3】に示す通りであった。新規サービス開始にあたってのシステムの新規構築や、新商品発売などが上位に挙がった。

【図表序-3】開発が特に遅いと感じる案件



出典：第２Ｇ アクチュアリー加盟企業へのアンケート

以上のアンケート結果より、多くの保険会社において、システム対応のスピードが追い付かないことで商品戦略・経営戦略を実行に移すスピードが鈍化しているという現状を読み取ることができる。
これより、保険会社が経営のスピードを上げるためには、システム対応のスピードを上げる必要があると言える。

０．２．研究の対象

私たちはシステム対応のスピードアップをどう実現するかについて研究した

前節までの議論を通じて、私たちは、保険会社が経営のスピードを上げて利益を上げ続けるためには、システム対応のスピードを上げる必要があると考えるに至った。そこで、保険会社において、システム対応をどのようにスピードアップしていくかを研究テーマの中心にした。

（1）システム対応のスピードを上げるためには何が必要か

システム対応のスピードアップの手段として、「DevOps」に着目した

ここでもう一度、研究のテーマに立ち返る。テーマの前半部分、「開発部門と運用部門の新たな関係作り、役割見直し」はその目的を達成する手段にあたる。
システム部門は通常、開発と運用という大きく二つの部門に分かれているが、システム対応を早いスピードで進めるためには、この両者が適切に連携をとっていく必要がある。

第一章から第三章にかけて、この「DevOps」がどういうものであり、保険会社ではどのように適用できるかを考察する。

Ⅰ．DevOps への組織的アプローチ

第一章では組織的なアプローチによって DevOps を実現する方法について論ずる。多くの会社のシステム部門が抱える組織的な課題を明らかにし、それらが DevOps の思想の元でどのように解決できるのかを考えることで、システムリリースサイクルの高速化の方法と実現可能性について探っていくこととする。

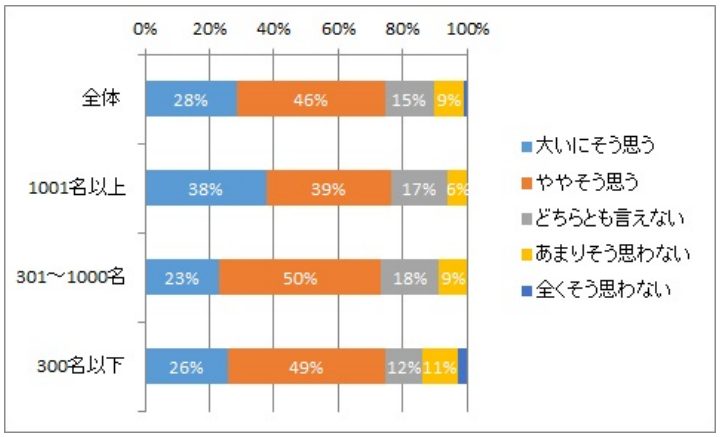
１．１．システム部門の組織の現状と課題

織の縦割り構造は、企業内のコミュニケーションの問題を発生させやすい

まず、システム部門の組織の現状と課題について整理してみる。システム部門に限らず、多くの会社でコミュニケーションが不足している状況に危機感を感じているという調査結果がある。

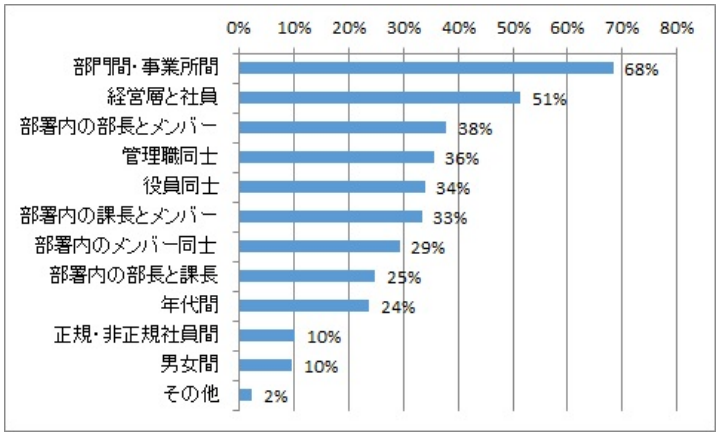
【図表 1-1】、【図表 1-2】は、HR 総研が実施した「社内コミュニケーションに関するアンケート」の結果の一部であるが、「社内のコミュニケーションに課題があると思うか」のアンケート結果を参照すると、コミュニケーションに課題を抱えた企業が多いことが分かる。また、「課題のあるコミュニケーションはどこか（全体）」のアンケートでは、多くの企業が部門間・事業所間と言った組織レベルでのコミュニケーション不足が原因であると回答している。縦割り組織の多い日本の企業において、部門間のコミュニケーションに不安を抱えていることが課題となっているととらえることができる。

【図表 1-1】社内のコミュニケーションに課題があると思うか



出典：HR 総研「社内コミュニケーションに関するアンケート」

【図表 1-2】課題のあるコミュニケーションはどこか（全体）



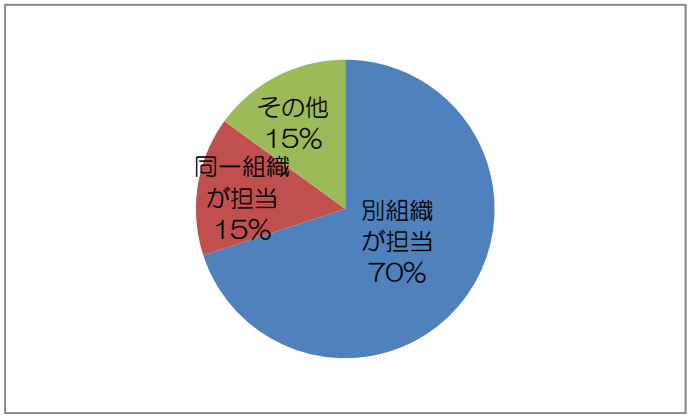
出典：HR 総研「社内コミュニケーションに関するアンケート」

（１）組織の構造

開発/運用の部門間においてコミュニケーションの断絶が発生しやすい

具体的に、保険会社のシステム部門における組織体制について着目してみる。【図表 1-3】はシステムの開発担当と運用担当がどのような組織体制をとっているのかについてのアンケート結果である。

【図表 1-3】開発と運用の組織体制



出典：第２Ｇ アクチュアリー加盟企業へのアンケート

7 割の企業がシステムの開発と運用を異なる組織が担当していると回答し、開発と運用が別組織に分かれている保険会社が多いことがわかった。

また保険会社におけるシステム開発は、ウォーターフォールモデルの流れにそって行われることが一般的である。要件定義は事業部門とシステム部門が共同で行い、システム設計以降の工程をシステム部門が担う。システム部門で行っている工程をさらに詳細化すると、要件定義から製造・テストまでを主に担うのが開発部門、本番移行や稼働後の運用・監視業務を主に担うのが運用部門となる。

このような業務分担の中で、開発部門は事業部門からの要望に接する機会も多いため、システム開発を早く、柔軟に進めたいというモチベーションが働きやすいと予想される。一方、運用部門は本番システムを安全に稼働させることに責任を負っているため、何事も慎重に進めたいというモチベーションが働くことが考えられる。このような現状は、海外では「開発部門は GO と言い、運用部門は NO と言う」とも表現される。言い換えれば、開発部門と運用部門では、それぞれの部門での主張、目標を達成するために、同じシステム部門であっても、対立構造が生まれやすくなっているという現状が浮かび上がる。

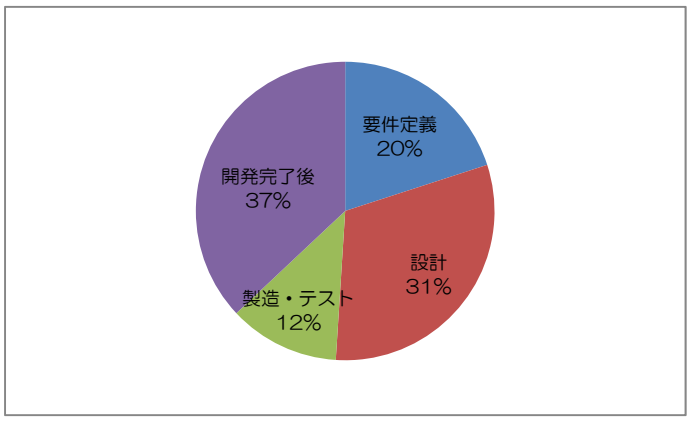
また、保険会社には「新契約」、「契約管理」、「経理」など、さまざまなシステムや事業部門がある。

開発部門の場合は、それぞれのシステムや事業部門に対応する開発チームが存在するのが通常である。一方、運用部門は、そのシステム全体をひとつのチームが担当していることが多い。そのため事業部門と開発部門の間では一対一の関係が成立し、コミュニケーションが比較的とりやすいのに対して、開発部門と運用部門の間では、多対一の関係となってしまいがちであり、コミュニケーションに断絶が起きてしまいやすい。このことが、先に説明した開発部門と運用部門の対立構造を深める要因にもなっている。また、開発部門と運用部門が多対一の関係において、運用部門はすべての事業部門のシステムの運用を担うことから、人手不足が常態化しやすいと言った課題も存在する。

（２）運用部門の参画タイミング

運用部門がシステム開発に参画するタイミングが遅い場合が多い

【図表 1-4】運用担当のシステム開発への参画タイミング



出典：第２Ｇ アクチュアリー会加盟企業へのアンケート

開発部門と運用部門のコミュニケーションが阻害される要因として、運用担当が開発に参画するタイミングにも、課題が存在しているようだ。運用担当がシステム開発にどのタイミングで参画しているかについてアンケートを行った結果、約 4 割の保険会社から開発完了後に初めて運用が参画しているという回答が得られた。

このアンケート結果からは、運用部門は本番移行が近づいたタイミングで初めてシステム開発に参画し、そこから非機能要件の検討を始める場合が多いことが推測できる。非機能要件とは、システムの性能やセキュリティーなど、ユーザーから見た業務要件には直接現れてこない要件を指している。

ウォーターフォールモデルでの開発にとって、手戻りは大きな痛手となる。特にシステム開発の終盤に近い局面での手戻りは大幅な遅延につながるリスクとなる。本番移行が近づいた段階で非機能要件上の問題が発覚した場合、まさに開発の終盤に近い局面での手戻りを発生させてしまう要因になる。あるいは、こうした手戻りのリスクを見込んで、開発期間に過剰なバッファを見積もることにもつながる。

手戻りと過剰なバッファはいずれも、スピーディーなシステム開発の阻害要因となる。

１．２．DevOps の組織的アプローチ例

組織改革やコミュニケーションの意図的な改善によって、問題を解決できる

これまでに、縦割り的な組織構造が開発と運用の対立構造やコミュニケーション不足をも

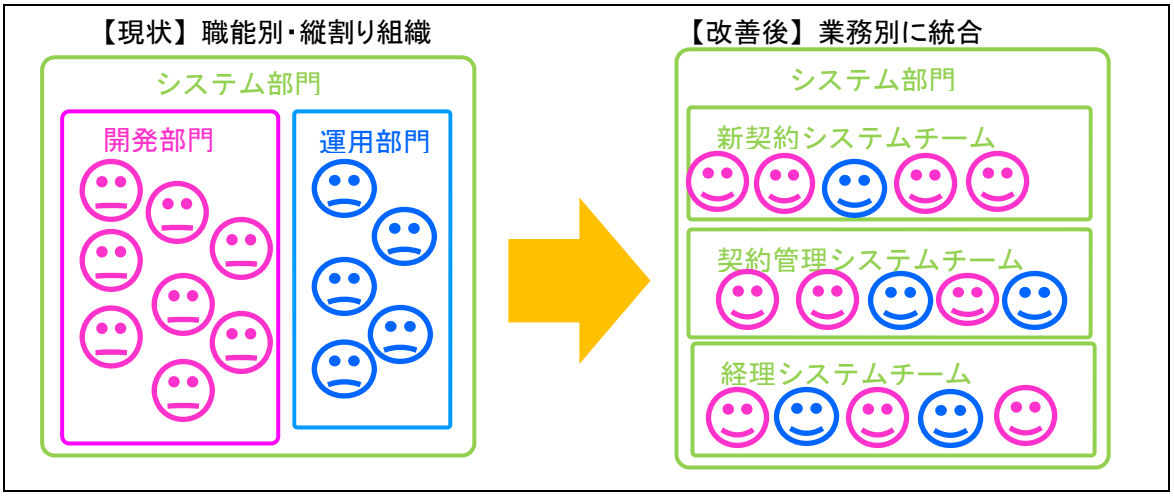
たらし、それらがシステム開発のスピードの阻害要因となっていることを見てきた。
こうした問題を、組織の構造を変えることで解決した例や、組織構造を大幅には変えなくても、コミュニケーションの仕方を意識的に変えることによって解決した例がある。
この節では、そうした成功例をもとに、組織や人の力による解決策を提示する。

（１）組織構造改革

解決策①：横割り組織への構造改革によって対立構造を解消する

抜本的な課題解決策として、現状は職能別の縦割りとなっている組織を、業務やシステム別に横串の組織に統合するという策が考えられる。これは実際に、日本国内の WEB 業界の企業が DevOps を成功させた実績のある事例である。

【図表 1-5】縦割り組織から横串の組織へ



コミュニケーションの断絶や対立を引き起こしやすい縦割り組織を廃することで、構造的にコミュニケーションの問題を起こさなくするという抜本的な解決策である。抜本的な改革であるために享受できるメリットも大きいが、実現のハードルやリスクといったデメリットを無視できないという保険会社も多いと考えられる。

【図表 1-6】組織構造改革に伴うメリットとデメリット

○	開発部門と運用部門を統合することで、チーム内での共通目標を持ちやすくなる
○	互いの領域の仕事を補完しあう協業体制の構築が容易になる
○	物理的に近くになることで互いのコミュニケーションが促進される
×	実現のハードルが高い（抜本的な組織改革や、開発・運用間の職務分掌に関する規程の改訂等も必要）
×	今までうまくいってきた組織構造を変えることに伴うリスク

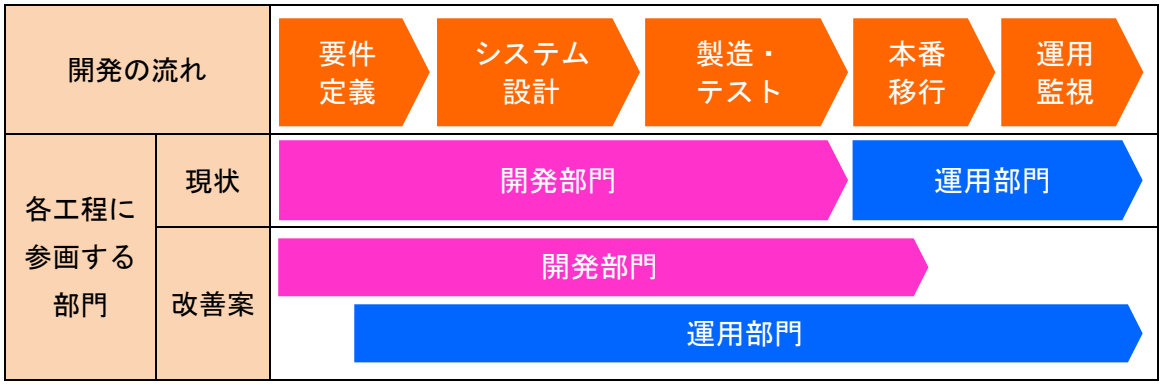
そこで次項からは、より実現のハードルが低く、デメリットの少ない解決策を紹介する。

（２）運用部門が上流工程から参加する

解決策②：運用部門の参画タイミングの早期化

今すぐ着手できる解決策の一つ目は、運用部門が開発の上流工程から参加することである。

【図表 1-7】運用部門が開発の上流工程から参加する



開発部門は上流工程の検討内容を運用部門にきちんと伝え、運用部門はそれを受けて、非機能要件に関して、早い段階で開発部門にアドバイスを行う。
これまで本番移行の直前になって初めて運用担当が参加していた会社であれば、運用担当が参加するタイミングを早めることによって、以下のメリットが得られる。

【図表 1-8】運用の参画タイミングを早期化するメリット

○　運用部門が早期に参画することで、開発部門とのコミュニケーションが促進される
○　システム開発終盤を迎えての手戻りが予防できる
○　過剰なバッファを見積もることによるスピードダウンが予防できる

問題となるのは、現状でさえ人手不足が問題となっている運用部門のさらなる参画をどのように促すのか、という点である。
それに対する答えのひとつとして、野村総合研究所（NRI）の例がある。同社では、開発部門から運用部門に歩み寄り、保守開発の上流で、運用の知見を述べる場を設けた。実際の保守を意識したシステム作りが可能となり、結果として利用部門からの追加要望が極小化され、開発の高速化につながったという成功事例である。
保険会社のシステム開発においても、この事例のように、開発部門と運用部門がお互いに歩み寄ることで、開発スピードの向上を実現できる可能性がある。

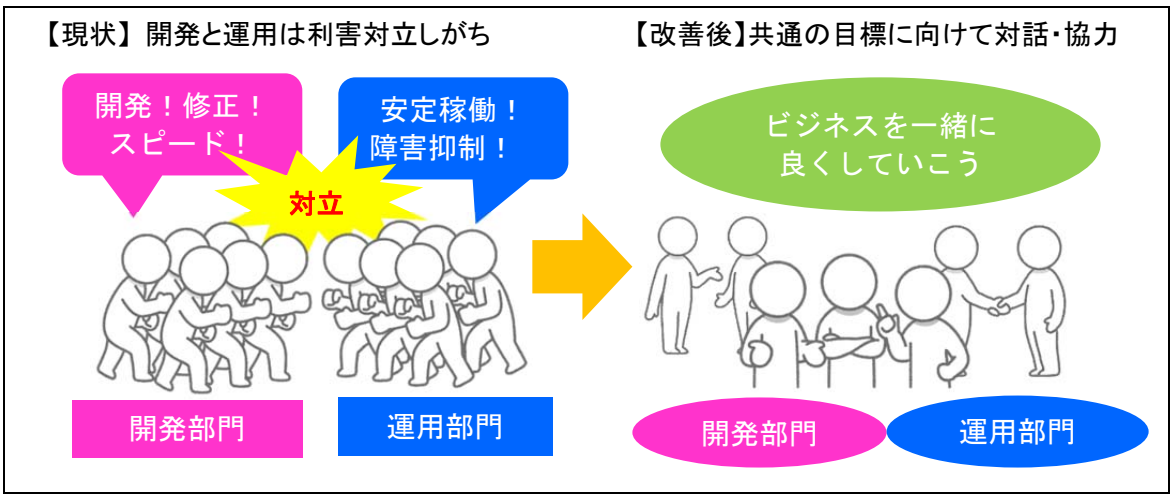
（３）運用部門と開発部門が課題を話し合う場を設ける

解決策③：開発/運用部門間の対話の場を設定、目標の統一化

今すぐ着手できる解決策の二つ目は、お互いに話し合う場を設けることである。この方法は一見、単純な策であるが、日ごろ開発部門と運用部門の間では、目先の業務に必要な最低限のやりとりしかしていない場合も多い。開発部門と運用部門が集まり、お互いの課題を率直に話し合う場を定期的に設けることで、自分たちに合った解決策を自分たちで見つけ出せるという利点がある。システム開発の現場の状況は組織によって千差万別であり、ある組織でうまくいった解決策が他の組織でも同じようにうまくいくとは限らない。自分たちの組織に合った解決策を見つけ出すのはとても重要なことだ。

日常的には、開発部門はシステムを作る、運用部門はそのシステムを安定稼働させ、障害を出さないといったようなそれぞれの部門ごとの目標を見て仕事をしている。その視点を一段上にあげて、「よりよいサービスをより早く、市場やお客様に届け続ける」という共通のビジネス目標を意識することで、対立構造の解消に成功した企業もある。

【図表 1-9】開発部門・運用部門は共通の目標を持つことで協業体制の構築を



先程紹介した NRI の例では、顧客により良いサービスを届けるために、運用作業の負荷軽減を開発部門、運用部門共通の目標としていた。そのことにより、システムを利用するユーザーが分かりやすいメッセージへの改善や、運用部門が担っていたリリース作業の自動化など、協業体制の成果によりリリースまでの時間の短縮化を実現している。

１．３．結論

組織面での DevOps へのアプローチは開発部門と運用部門のコミュニケーション促進から

市場の動向をいち早くシステムに反映していくには、リリースまでを含めた開発スピードの高速化が求められる。この高速化を達成するため、開発部門と運用部門の協業は避けて通ることができないが、一般的に部門間や組織間のコミュニケーションには課題があると認識されている。

従来のウォーターフォールモデルでの開発を実施してきた企業にとって、開発部門と運用部門はそれぞれに個別最適を目指すことが全体最適を達成すると考えられてきた。
しかし、開発部門と運用部門の協業が求められるなか、両部門がそれぞれの個別最適を目指すことは、対立関係を生む一因となっている。

開発部門と運用部門の協業を達成し、リリースサイクルの高速化を目指すためには両部門が「より良いサービスをより早く、市場・お客様へ届け続ける」といったような共通の目的に基づいて協力しあうことが求められる。そのための第一歩として、１．２．（２）や（３）で述べたように、まずは開発部門と運用部門のコミュニケーションを促進することが、最も動き出しやすく、効果の高い方法といえるのではないだろうか。

Ⅱ．DevOps への技術的アプローチ

第一章では組織の面での DevOps へのアプローチ方法を説明した。第二章ではツールを利用した技術的なアプローチ方法について検討する。

２．１．リリースを早める上での技術的な課題

「ビルド」と「リリース」を手作業に頼ってきたため、膨大な時間がかかっていた

システムが使われるようになるまでに、作成したアプリケーションを開発環境から準本番環境、本番環境に順次反映をする必要がある。このうち、作成したアプリケーションを稼動可能な状態にすることを「ビルド」といい、ある環境から別の環境へプログラムを移すことを「リリース」と呼んでいる。

数年前までは、こうしたリリースやビルドは手作業が当たり前であり、自動化は困難であると考えられてきた。そのため、多くの時間と人員を費やし、さらには手作業による作業

の失敗、つまり、システム障害を引き起こすリスクを抱えてきた。
本番環境へのビルドやリリースは運用部門が担当しているため、システム開発において運用部門が慎重な姿勢をとるのは、こうした理由がある。

２．２．DevOps の技術的アプローチ例

代表例は「継続的デリバリー」の実現である

昨今の技術進歩によって、ビルドから製品のリリースに至るまでの過程を、手作業ではなく自動で、かつ、一気通貫で行うことが可能となった。それが「継続的デリバリー」である。次項において「継続的デリバリー」の概要を説明する。

（１）継続的デリバリーの概要

開発からリリースまでを素早く継続的に実施できる仕組みが「継続的デリバリー」である

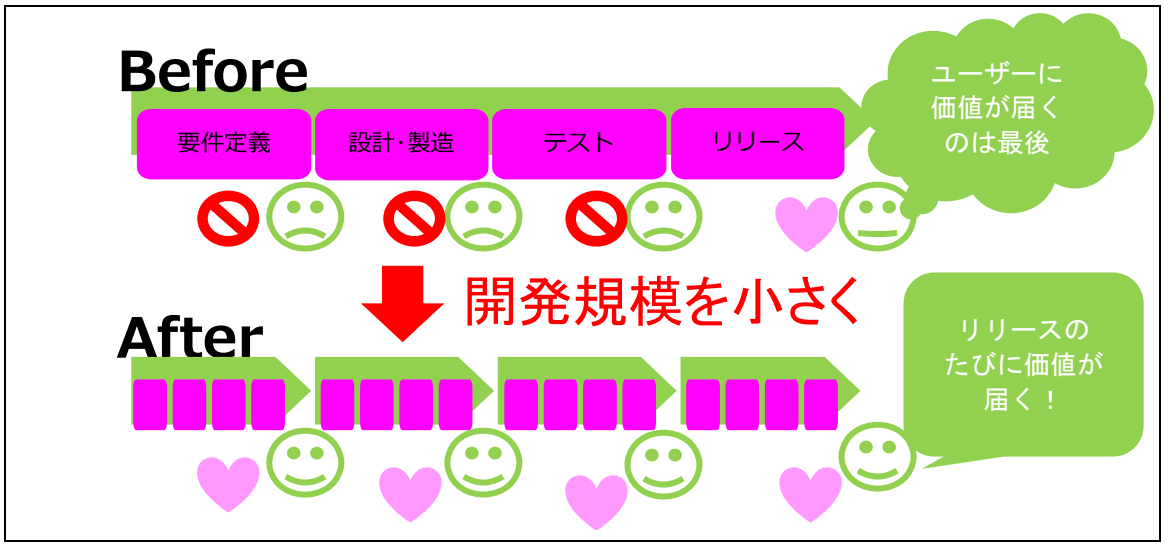
「継続的デリバリー」は、開発からリリースまでを素早く、かつ継続的に実施する仕組みのことである。
一般的なウォーターフォール開発と、継続的デリバリーには以下のような違いがある。

【図表 2-1】一般的なウォーターフォール開発と継続的デリバリーの違い：特徴のまとめ

	ウォーターフォール開発	継続的デリバリー
開発手法の思想	計画をしっかり立てた上で 開発に着手する	市場・ユーザーの反応を見ながら 継続的に改善する
リリース回数	少ない	多い
一回の開発規模	大規模	小規模

これらの違いを図で示すと以下のようになる。

【図表 2-2】一般的なウォーターフォール開発と継続的デリバリーの違い：イメージ図



システムはリリースされて初めて顧客やユーザーに価値を届けることができる。また、顧客やユーザーからシステムに対するフィードバックを得て、それに基づいてシステムを改良・改善していくことも、リリースされてからこそ実現できることである。
しかし、リリース回数を多くすることでデメリットも生じる。リリース回数を少なく抑えた場合と、多く実施した場合では、従来、以下のようなメリット・デメリットがあると考えられてきた。

【図表 2-3】リリース回数の大小によるメリット・デメリット

観点	リリース回数が少ない場合	リリース回数が多い場合
初期投資	× 多額の初期投資が必要 ⇒初期投資が無駄になるリスクが高い	○ 最初は小さなシステムから初めて、レベルアップできる ⇒初期投資が無駄になるリスクが抑えられる
変化への対応	× 柔軟・迅速に対応できない	○ 柔軟・迅速に対応できる
フィードバック	× 長い開発期間を終えるまで、フィードバックが得られない	○ フィードバックをこまめに取り入れながら改善していける
機能テスト	○ １データ・１ショットで複数の機能のテストをまとめて行える	× データの用意・ショットの実施がリリースの度に必要
回帰テスト	○ 回帰テストの回数を極小化でき、工数を低減できる	× リリースの度に回帰テストが必要となり、膨大な工数がかかる
システム環境	○ システム環境への修正を極小化できる	× システム環境への頻繁な修正が発生し、障害リスクにもつながる
移行作業に伴うリスク	○ リリースの回数を低減することで、リスクを最小化できる	× リリースの回数が増え、移行ミスによる障害のリスクが増える

数年前までの技術水準では、リリースを増やした場合のテストにかかる工数や、リリース時の障害のリスクといったデメリットは、リリース回数を少なく抑えた場合のデメリットを上回ると考えられていた。金融機関である保険会社においては特に、システムの品質は非常に重要であり、テストを省略するという選択はとれず、リリースに伴うリスクの増加も許容しにくい。
昨今の技術の進歩によって、リリース回数の増加に伴うこうしたデメリットを被ることなく、頻繁なリリースが実現できるようになっている。変化への柔軟な対応がビジネスにおいて重要性を増している現状とともに、そのような技術が誕生したことが、DevOps が隆盛している背景のひとつとなっている。次項より、その代表的な技術を紹介する。

（２）テストの自動化

テストを自動化すれば、品質を維持したまま、リリースを増やす際の大きなデメリットであるテスト工数を抑えられる

a.テストの自動化はなぜ必要か

以下は【図表 2-3】より、テストに関するメリット・デメリットを抜粋した表である。

【図表 2-4】リリース回数の大小によるテスト工数のメリット・デメリット

観点	リリース回数が少ない場合	リリース回数が多い場合
機能テスト	○　１データ・１ショットで複数の機能のテストをまとめて行える	×　データの用意・ショットの実施がリリースの度に必要
回帰テスト	○　回帰テストの回数を極小化でき、工数を低減できる	×　リリースの度に回帰テストが必要となり、膨大な工数がかかる

テストにかかる工数が、リリースを増やすにあたって大きなデメリットとなっていることがわかる。テストを人手に頼って行っている限り、こうしたデメリットは無視できない。また、品質が重要視される保険会社のシステムにおいて、テストを簡略化することによって工数を抑えるという選択肢は極めてとりにくい。
一方、テストを自動化すれば、これらのデメリットは解消する。多くの保険会社がテストを人手に頼っている現状の中、テストの自動化を取り入れる効果は大きい。

b.テスト自動化の仕組み

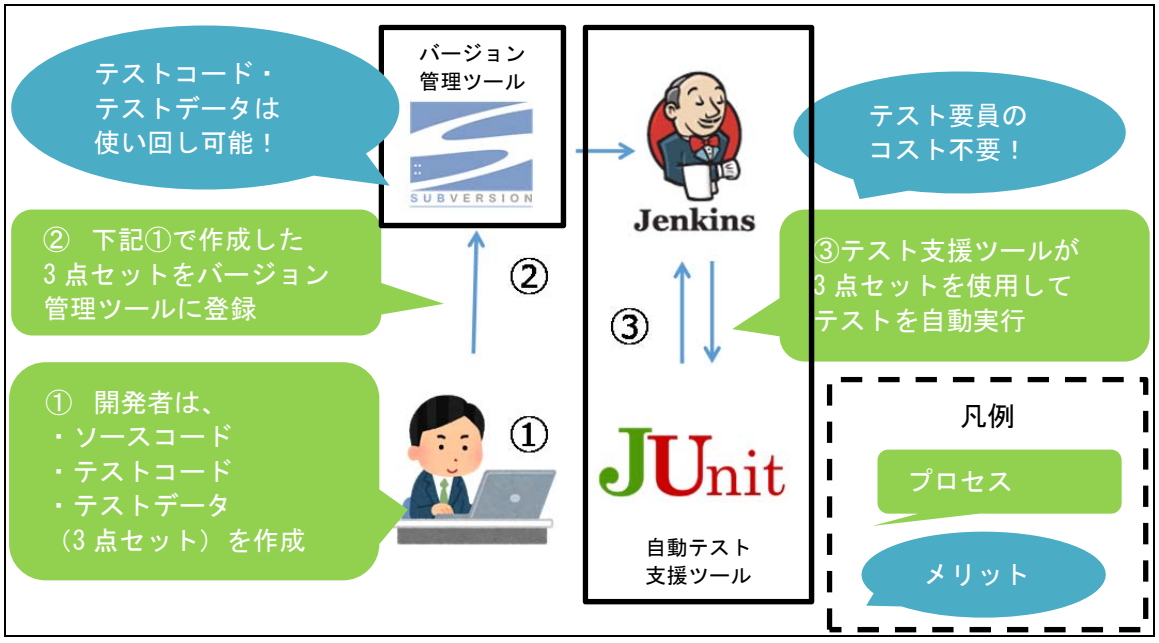
【図表 2-5】自動テストの実施内容

工程	実施内容
① ３点セットの作成	開発者はソースコードとともに、テストコードとテストデータを作成する。これを便宜的に「３点セット」と称する。
② バージョン管理ツールへの登録	開発者は３点セットをバージョン管理ツールに登録する。テストコードやテストデータも、ソースコードと同等の資産としてバージョン管理する。
③ 自動テストの実行	バージョン管理ツールに登録された３点セットを使用し、自動テスト支援ツールが自動的にテストを実行し、合否を判定する。

自動テストを行うにあたっては、表の①に記載の通り、開発者はソースコードとともにテストコードとテストデータを作成する必要がある。これは初回の開発時には手間のかかる作業となるが、テストコードやテストデータを②でバージョン管理ツールに登録し、資産として管理しておくことで、次回以降の開発時にテストデータやテストコードを使いまわすことができる。テストの実行は③にある通り、自動テスト支援ツールが自動的に行い、合否判定を返す。開発者は、すべてのテストケースに合格するまで開発と自動テストを繰り返す。

これを図で示すと、以下の通りとなる。
テストの自動化によって、開発者はテストの実行にほとんど手をかけることなく、品質の高いシステムを作り続けることができるようになる。

【図表 2-6】テスト自動化の仕組みとメリット



c.テスト自動化をどこから取り入れるか

テストには、単体テスト、機能テスト、UI テストなどさまざまな種類がある。いずれのテストも自動化は可能であるが、まず第一歩としては、どこから着手するのが妥当か。
我々は、単体テストからの導入を推奨する。
下表にまとめた各テストの性質を踏まえると、自動テストの効果が最も得やすいのが単体テストだと言える。単体テストは最も初期の段階で行われるため、この時点で障害を検出できれば後の工程への影響や手戻りが少なくて済み、障害が発生してもバグの発生箇所の特定もしやすい。

【図表 2-7】各種のテストの比較

観点	単体テスト （ユニットテスト）	機能テスト （ファンクショナルテスト）	UI テスト
概要	プログラム単体で、開発者の意図どおりに動作することを保障するテスト。	コード毎の連結する部分のテスト。コード間の結合条件によって引き起こされるバグを検出する。	実際のブラウザやアプリを動作させて行うテスト。業務的な観点で要件が実現できているか確認する。
テスト実施時期	製造の直後に実施	単体テストの後に実施	機能テストの後に実施
テスト粒度	細かい	中程度	大きい
障害箇所の特定	特定しやすい	単体テストよりも 特定しにくい	最も特定しにくい
障害発生の影響	小さい	中程度	大きい

また、他のテストに比べ、単体テストは最も自動テストの取り入れが容易であると言われており、単体テストを自動化しておくことは、以後のテスト工程の自動化への足がかりにもなる。

（３）環境構築の自動化（Infrastructure as Code）

環境構築の負荷や待ち時間を、自動化によってなくすことができる

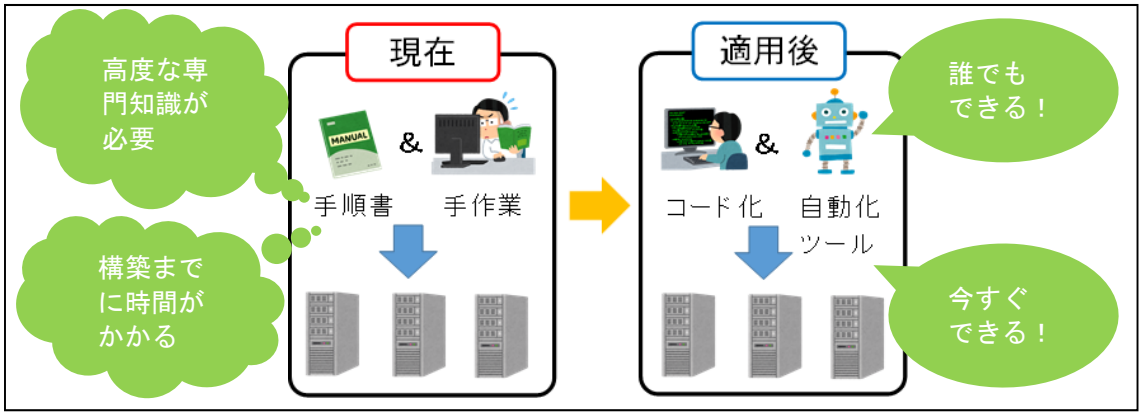
開発案件の内容によっては、プログラムを修正するだけでなく、プログラムの動作環境（＝システム環境）にも修正が必要な場合がある。以下は【図表 2-3】より、システム環境に関するメリット・デメリットを抜粋した表である。

【図表 2-8】リリース回数の大小によるシステム環境に関するメリット・デメリット

観点	リリース回数が少ない場合	リリース回数が多い場合
システム環境	○　システム環境への修正を極小化できる	×　システム環境への頻繁な修正が発生し、障害リスクにもつながる

システム環境への修正を手作業で行うことを前提とすれば、このデメリットも無視できない。システム環境の構築作業には、ハードウェアの調達、OS の設定、ミドルウェアのインストールと設定など様々な作業があり、これらの作業は手順書と手作業に頼って行うしかないと考えられてきた。多くの保険会社でも、インフラ基盤担当者がこうした作業を手作業で実施している。しかし、環境構築ツールの誕生により、これらのデメリットも解消されつつある。ハードウェアの仮想化や、環境構築ツールなどの技術進歩により、上述の作業をプログラム上の設定だけで行えるようになった。

【図表 2-9】環境構築の自動化



インフラ基盤担当者が環境に合わせた設定をコード化し、自動化ツールに適用しておくことで、開発担当者はインフラ基盤担当者と調整する必要や、構築までに長い時間待つ必要がなくなる。環境構築ツール上のボタンをクリックするだけで、すぐに開発に必要な環境を手に入れられるようになる。

（4）デプロイの自動化

デプロイの自動化によって、リリースを早く、かつ安全に行うことができる

環境構築と同様に、プログラムのリリースも、従来は人手に頼って行うしかないと考えられてきた。人手によるリリースはコストがかかるだけでなく、移行作業のミスに伴う障害のリスクを常にはらんでいる。以下は【図表 2-3】より、移行作業に関するメリット・デメ

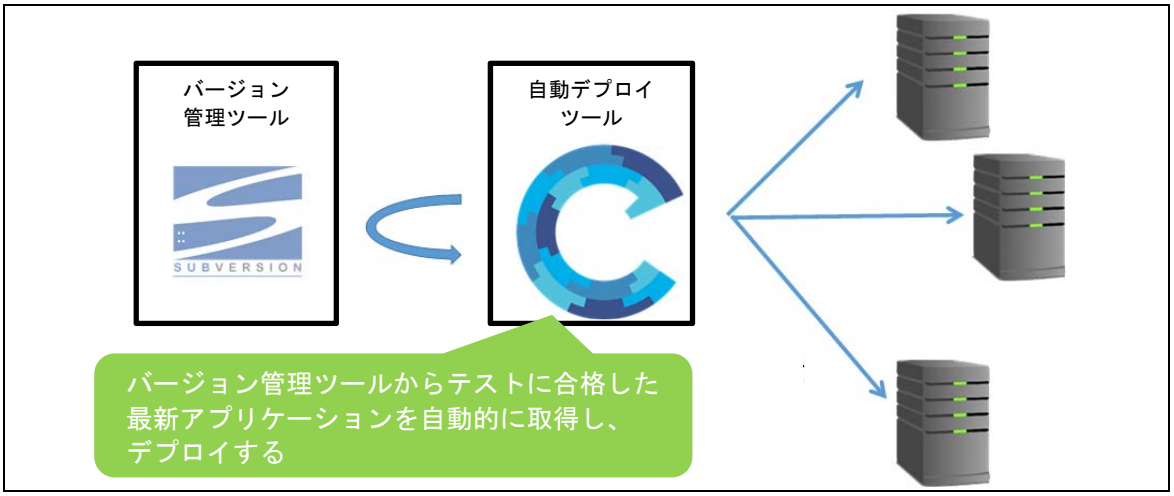
リットを抜粋した表である。

【図表 2-10】リリース回数の大小による移行時のメリット・デメリット

観点	リリース回数が少ない場合	リリース回数が多い場合
移行作業に伴うリスク	○ リリースの回数を低減することで、リスクを最小化できる	× リリースの回数が増え、移行ミスによる障害のリスクが増える

この問題も、デプロイの自動化の実現によって解消されつつある。自動デプロイの仕組みは下図の通りである。

【図 2-11】自動デプロイの仕組み



このツールを適用することで、バージョン管理ツール（SUBVERSION 等）で管理しているアプリケーションを自動で取得し、本番環境に自動でリリースすることができる。問題が発生した場合のロールバックもサポートしている。

デプロイを自動化することによって、移行作業のミスを懸念することなく、早く安全なリリースを継続的に行うことが可能になる。

（５）その他の関連技術

DevOps に関連するツールは他にも多様にある

前項までに、継続的デリバリーの実現に関連するツールを紹介してきたが、DevOps に関連するツールは他にも多様に存在する。たとえば、チャットツールなどのコミュニケーションツールを活用することによって、「Ⅰ．DevOps への組織的アプローチ」で紹介したよう

な、開発部門と運用部門の連携もより強めることができる。
開発現場でよく使われているチャットツールには Slack、Chatwork、Hipchat などのサービスがある。開発の進捗状況やリリース結果の連絡などはチャットに切り出すことによって、より即時的な連絡が可能になり、それらの情報に重要なメールが埋もれてしまうといった事態も防ぐことができる。メンバーの予定を管理し、会議の日程調整を自動で行える機能など、便利な機能を搭載したサービスもある。チャットボット機能によって、進捗やリリース結果の通知を自動で行うことも可能であり、開発・運用双方の情報発信の手間を削減するとともに、情報発信の遅れや漏れも防ぐことができる。

２．３．技術的アプローチを採用する上での課題

コスト削減だけでなく、利益や機会損失を考慮した適用検討が必要である

これまでに紹介したツールを、単純なコスト削減策として導入すると考えた場合、短期的に見ると導入コストが削減コストを上回ってしまう場合がある。
導入のコストには、ツール自体の購入費用の他に、開発・運用の担当者がそのツールを使いこなせるようになるまでの教育コストがかかる。また、テストの自動化を導入する場合には、テストコードやテストデータ作成するためのコストもかかる。
したがって、削減できるコストと導入にかかるコストを単純に比較すると、コストの回収期間を短期に見積もるほど、削減できるコストを導入のコストが上回ってしまう可能性が高まる。
一方で、序章でも述べた通り、DevOps の導入によって本来目指しているのは、コストの削減ではなく利益の増大である。システム開発のスピードが追いつかないことによって経営のスピードが鈍化することに伴う機会損失と、素早いシステム対応が経営のスピードを後押しし、市場機会をとらえた経営を展開することで得られる利益を考慮に入れた上で、適用の是非を検討する必要がある。

２．４．結論

DevOps の技術的アプローチを採用するメリットを広い視野でとらえ、導入を検討するべきである

DevOps への技術的アプローチを取り入れることで、テストや環境構築、リリースにかかっていた多くの人的作業を削減することができる。人的作業の削減は、スピードの向上や工数の削減だけでなく、人為的なミスの抑制にもつながる。開発担当者も運用担当者も、ミスを防ぐために行っていた膨大な作業から解放され、システムの価値を高めるための作業

に専念することが可能となる。
営業に関するシステムの価値を高めることによって、より効果的な営業活動につなげることもできる。事務システムの使い勝手が上がれば、事務部門の業務効率が高まり、より多くの人的資源を利益につながる活動へと振り向けることも可能になる。また、システム対応のスピードが上がることで、より柔軟な商品戦略の実施も可能になる。
システム開発コストの削減だけでなく、こうした利益を幅広い視野でとらえた上で、DevOps への技術的アプローチの導入を検討するべきではないだろうか。

Ⅲ．保険業界における DevOps の適用

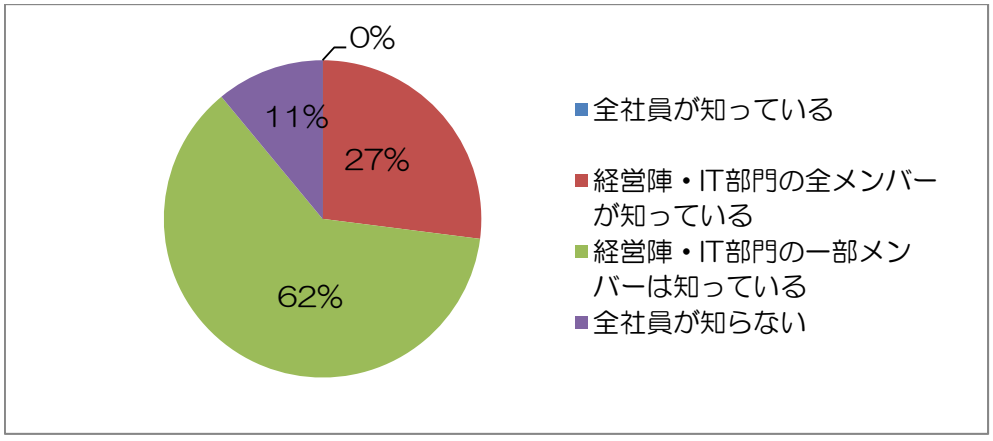
第一章、第二章では、システムのリリースサイクル短縮に寄与する DevOps の概念について、組織とツールの両面から紹介してきた。
これまでに紹介してきた DevOps の事例は、保険業界に限らず IT 業界全般にわたるものであるが、この章では保険業界に焦点を絞り、当業界における DevOps の必要性と、実現可能性について考察する。

３．１．保険業界にも DevOps は必要か

保険業界にも DevOps は必要である

WEB 業界など、スピードが重視される業界と比べ、金融・保険業界における業界全体の DevOps に対する認知度や導入事例は、まだ多くないのが現状である。

【図表 3-1】DevOps の認知度



出典：第２Ｇ アクチュアリー会加盟企業へのアンケート

また、DevOps の導入に成功したメルカリ、楽天、Cookpad は、「ユーザーに日常的に利用

されるサービスを提供している」点で共通している。このようなサービスであれば、ユーザーのニーズの変化を敏感にとらえ、スピーディーに対応する必要があることは容易に想像できる。これに比べ保険商品は前述の 3 社が提供するサービスほど顧客が日常的に触れるものではなく、顧客の新たなニーズが生まれるとは考えにくい。

また、保険会社をはじめとする金融業界は、他業界と比べて取り扱う商品の重大性、機密性が高く、システム障害が発生した場合には多大なる影響を顧客に与えてしまう。このため、リリースサイクルを早めることは品質の低下を招き、障害発生のリスクが高まるのではないかと考える企業も多い。

このように、顧客の新たなニーズが頻繁に生まれるとは考えにくく、品質維持が絶対とされる保険業界に、果たして DevOps が求められる場面はあるのだろうか。

我々は、あると考える。その理由は外部要因（顧客の視点）と内部要因（経営の視点）の両面から挙げられる。

外部要因－顧客の視点

序章の０．１．－（１）で述べた通り、顧客は現状の保障内容に満足していない場合がほとんどであり、保険に対して常に新しい保障を求めている。
保険商品に満足していただくには、医療技術の進展や高齢化社会への移行などを踏まえた時代の変化に合わせた商品を作ることが必要であり、そのことが新規契約獲得や保有契約の維持のためにも不可欠となる。
保険会社は時代の変化に遅れることなく、速やかに新商品の販売を進めていかなければならないため、スピードが重要視されるのは言うまでもない。

内部要因－経営の視点

序章の０．１．－（２）で述べた通り、保険会社の内部でも、現状のシステム開発のスピードに対する課題を感じている企業が 9 割に迫っている。
また、スピードに課題を感じていないと回答した企業にその理由を質問したところ、現状のスピードに合わせて業務を構築しているからと答えた企業が 8 割を超え、十分にスピード感のあるリリースができていると回答した企業はなかった。

以上の２つの理由から、保険業界にも DevOps は必要であると考える。

３．２．保険業界でも DevOps の適用は可能か

保険業界でも DevOps の適用は可能である

DevOps は主に WEB 業界で発達してきた手法のため、DevOps に関するツールはオープン系システムを前提としたものが多い。一方、我々保険業界のシステムは、基幹部分の業務ロジックをメインフレームに頼っている比重が依然高いという現状がある。そのような保険業界にあっても、DevOps の適用は可能なのだろうか。

私たちは、可能であると考える。理由を以下に述べる。

（１）DevOps への組織的アプローチは、システム構成を問わず導入可能である

開発部門と運用部門の組織体制および関わり方を見直すことで可能である

「Ⅰ．DevOps への組織的アプローチ」で紹介した通り、システム開発の組織を「職能別の縦割り組織」から「業務別に統合する」というような、抜本的な改革を行うことは、リスクにもなると考える企業は多いだろう。しかし、「運用担当が開発の上流工程から参加する」、「運用部門と開発部門がお互いの課題を率直に話し合う」といった方法は、メインフレーム中心のシステム領域であったとしても、今すぐにできる方法である。
組織の面からアプローチすることで、保険会社はすぐにでもシステム対応のスピードアップに向けて動き出すことが可能になる。

（２）メインフレーム向けの DevOps ツールが登場し始めている

メインフレーム環境でも導入可能なツールは存在する

DevOps は 2009 年の Velocity カンファレンスにおいて、米国 Flickr 社の 2 名のエンジニアが行った“10 deploys per day -Dev & Ops cooperation at Flickr-”というタイトルのプレゼンテーションが起源と言われている。それ以来、多くのオープン系システムの開発者たちがこの手法を取り入れ、実績をあげてきた。
一方で、メインフレーム向けの DevOps が話題になり始めたのは、2016 年頃からである。保険業界のようなメインフレームを基盤としている企業でも導入できる、DevOps ツールが登場し始めている。
その一例が、IBM 社の DevOps for Enterprize Systems のシリーズである。この中には、「Ⅱ．DevOps への技術的アプローチ」で説明したテストの自動化やリリースの自動化、環境構築の効率化に関するソリューションも含まれている。

画面やユーザーインターフェースにはオープン系システムを使用しているものの、基幹となる業務ロジックはメインフレームを使用している、というシステム構造は、金融機関を筆頭に依然多くの企業が抱えている。
そうした企業では、オープン系で開発している画面やユーザーインターフェースの部分は、アジャイルや DevOps などの手法を使って、速いスピードでの開発が実現できるようになっている一方で、メインフレームが抱えている業務ロジック部分は相変わらずウォーターフォールに基づいて、従来通りのスピードで開発を進めている。このため、両者の間で開発スピードに軋轢が生じる、というケースが頻発するようになった。

【図表 3-2】システム領域間の開発スピードの軋轢

システム領域	システムの種類	開発手法	開発スピード
画面・ＵＩ	オープン	アジャイル	早い
業務ロジック	メインフレーム	ウォーターフォール	遅い

ユーザーから見れば、画面やユーザーインターフェースの部分と業務ロジックの区別はあまり意味がなく、両者が組み合わさってはじめて一つのシステムとなる。いくら画面やユーザーインターフェースが速いスピードで開発できたとしても、業務ロジックの修正を伴わずに修正できる範囲は限られてくる。業務ロジック部分の開発が素早く行えないままでは、結局そのシステムはユーザーにとって、「変化への対応が遅いシステム」、ということになってしまう。

【図表 3-3】ユーザーから見た変化への対応スピードは、遅い方に引きずられてしまう

	現状	あるべき姿
画面・ＵＩの開発スピード	早い	早い
業務ロジックの開発スピード	遅い	早い
ユーザーから見た開発スピード	遅い	早い
全体像	画面やＵＩの開発は早くできても、業務ロジックの開発スピードがそれに追いつかないため、ユーザーにとっては変化への対応が遅いシステムとなっている	画面・ＵＩと業務ロジックの開発スピードがともに早いペースでかみ合い、変化にスピーディーに対応するシステムをユーザーに届けられる

そうした問題を解決するために、近年になってメインフレーム向けの DevOps ソリューションが誕生したのである。メインフレームが抱える業務ロジックをスピーディーに開発・改修できるようになってこそ、システム全体の開発スピードが上がり、ユーザー満足を実現できる。このような課題認識が、徐々に広がり始めている。
こうした課題認識は、我々保険会社が抱える問題とも一致しており、そのためのソリューションがまさに今、台頭しようとしていることは、我々にとって追い風といえるのではないか。

３．３．結論

保険会社も DevOps を導入すべきである

これまで説明してきたように、保険業界でも DevOps によるリリーススピードの向上は求められている。また、実現可能性の面でも保険業界で DevOps を導入することは可能である。我々保険会社は組織、ツールの両面から、今すぐにシステム対応のスピードアップに向けて動き出すべきである。
具体的なやり方として、組織面からのアプローチとしては、①組織の構造改革、②運用担当が開発の上流工程から参加する、③運用担当と開発担当が互いに話し合う、という３つの方法を提案した。①組織の構造改革は実現のハードルやリスクが高いと感じる保険会社でも、②や③の方法であれば、今すぐ取り入れることが可能である。
ツールの面では、画面やユーザーインターフェースなどのオープン系システムだけでなく、保険会社の多くが抱えているメインフレーム部分についても、DevOps のソリューションが台頭しつつあることを説明した。
保険会社がウォーターフォールモデルなどの従来の開発手法から脱却し、DevOps の様々な手法を用いた開発を成功させる日も、そう遠くない未来にやってくるのではないだろうか。

終章

保険会社にとって、経営のスピード向上は急務である。魅力的な商品やサービスをタイムリーに提供し続けるなど、スピードのある経営を実現するためには、それに見合うスピードでシステム対応を実施していく必要があることを序章で述べた。
その手段である DevOps について、組織的アプローチ・技術的アプローチの両面から、実現方法を紹介した。いずれのアプローチも、保険会社にとって実現可能であり、実践する価値があることを、本論の中で述べてきた。

本論の中では「DevOps」という研究テーマに基づき、システム部門の当事者である開発部門と運用部門の関係性に視点を絞って考察を行ってきた。
システム業界において、開発部門と運用部門の対立構造は積年の課題となっていた。その両者が共通のビジネス目標に向かって協力しあうことで、対立を乗り越え、よりスピーディーかつ利益につながるシステム対応を実現できたことを証明しているのが、DevOps の成功事例である。技術の進歩により、第二章で紹介したような様々なツールが誕生してきたことも、そうした流れを後押ししている。
一方で、本論の中では触れなかったが、当時者同士の関係改善が重要な意味を持つのは、システム部門の内部に限った話ではない。特に、大きな組織となりがちである保険会社にあっては、本社と現場、企画と営業、システム部門と事業部門、複数の事業部門間など、当事者同士の関係性の改善によって、よりよい経営のありかたに近づける場面はまだまだある。こうしたシステム以外の部門を含めた関係性の改善においても、DevOps のコンセプトから学べることは数多くある。

これからより多くの保険会社が、システム開発において DevOps の考え方を適用し、スピーディーな経営を実現するにあたり、当グループの研究が少しでも貢献できれば幸いである。また、DevOps のコンセプトの根幹である「複数の部門が同じ目標に向かって協力する」というマインドが、システム部門に限らず保険会社の各部門に浸透し、保険業界がますます発展していくことを願っている。

謝辞

本研究を進めるにあたり、製品レクチャー実施頂きました日本ＩＢM株式会社様、株式会社日立製作所様、富士通株式会社様、並びにアンケートにご協力いただきましたアクチュアリー会法人会員各社様、私たちの活動を支えて下さった多くの方々に、この場を借りて深く御礼申し上げます。

参考文献

<書籍>

・河村聖悟,北野太郎,中山貴尋,日下部貴章,株式会社リクルートテクノロジーズ『DevOps 導入指南 Infrastructure as Code でチーム開発・サービス運用を効率化する』翔泳社
・ジーン・キム,ジェズ・ハンブル,パトリック・ボア,ジョン・ウィリス『The DevOps ハンドブック 理論・原則・実践のすべて』日経 BP 社
・レン・バス,インゴ・ウェーバー,リーミン・チュー『DevOps 教科書』日経 BP 社
・Gary Gruver,Tommy Mouser『変革の軌跡【世界で戦える会社に変わる"アジャイル・DevOps"導入の原則】』技術評論社

<雑誌>

・『日経ＳＹＳＴＥＭＳ』 2014 年 2 月号 日経 BP 社
・『日経ＳＹＳＴＥＭＳ』2014 年 3 月号 日経 BP 社

<その他>

・公益財団法人生命保険文化センター「平成 27 年度生命保険に関する全国実態調査」
<http://www.jili.or.jp/research/report/zenkokujittai.html>
・Flickr 社のエンジニアによる 2009 年の Velocity カンファレンスでのプレゼンテーション資料
“10 deploys per day -Dev & Ops cooperation at Flickr-”
<https://cdn.oreillystatic.com/en/assets/1/event/29/10%2B%20Deploys%20Per%20Day_%20Dev%20and%20Ops%20Cooperation%20at%20Flickr%20Presentation.pdf>
・DevOps for Dummies - 3rd IBM Limited Edition
<https://www-01.ibm.com/common/ssi/cgi-bin/ssialias?htmlfid=RAM14026USEN