ハイブリッド環境における アプリケーション配信の加速

Accelerating Application Delivery in a Hybrid World

● James Weir　● Alban Richard　● 植田　譲

あらまし

今日のビジネスは，アジャイル方法論やDevOpsプロセスでICTの応答性を向上することに目を向けている。しかし，ガバナンスやコンプライアンスが損なわれないことも保証しなければならない。更に，ハイブリッドICTやマルチクラウド環境では，様々なインフラが開発や運用に影響を及ぼしており，これらの相反する要件を満たせなくなる可能性がある。DevOpsに共通の定義は存在しないが，DevOpsとはコミュニケーションと協力を促進する環境であり，アジャイルとはその環境内での働き方であると考えることができる。DevOpsでは，高品質で十分にテストされた製品のリリースサイクルを最適化しつつ，様々なチームや部門が効率的な協力やコミュニケーションを行うことを重視している。FUJITSU Software UForge AppCenterはDevOpsを念頭に置いた基盤であり，ソフトウェア配信を高速化するため，アプリケーション配信のライフサイクルの様々な段階における人手による調整作業を減らすことに注力している。

本稿では，アジャイル開発を説明するとともに，アジャイル開発がDevOpsにどのように関連するのか，コントロールや一貫性を犠牲にせずDevOpsから必要とするスピードと俊敏性を得るために，UForge AppCenterがどのような役割を果たせるのかについて述べる。

Abstract

Today's businesses are looking to increase IT responsiveness with agile methodologies and DevOps processes. But they must also ensure that governance and compliance are not compromised. Furthermore, hybrid IT or multi-cloud environments can exacerbate these conflicting requirements, where different infrastructures are in play for development, test, pre-prod and production. DevOps means many things to many people, with no common definition. It can be seen as being an environment that promotes communication and collaboration, while Agile is a method of working within the environment. DevOps stresses effective collaboration and communication between various teams and departments within a culture that optimizes release cycles of high-quality and thoroughly tested end products. By viewing the entire delivery process holistically, DevOps helps us identify and solve bottlenecks that traditionally happen when one role in the process is overloaded. Fujitsu Software UForge AppCenter is a platform with DevOps in mind, focused on reducing manual coordination across the different stages of the application delivery life cycle to maximize the velocity of software delivery. This article describes agile development, how it relates to DevOps and how UForge can help get the velocity and agility you need from DevOps without sacrificing control and consistency.

ま え が き

ビジネスがかつてないほどのスピードで変化する今日，企業はより俊敏で応答性に優れたICTを活用することで，自らを進化・適応させ，顧客の新たな要望に応えていく必要がある。しかし同時に，ガバナンスやコンプライアンスが損なわれないことも保証しなければならない。

近年はクラウドの活用によって，必要に応じてスピーディーにICTインフラを導入できるようになった。一方で,アプリケーションの構築や配備は，顧客の要望に応じた人手による作業であり，俊敏性やガバナンスが損なわれる原因となっている。

たとえ何らかのツールを持っていたとしても，個々のインフラ向けに一から人手で構築作業を行う必要があるとしたら，アプリケーション配信は非常に困難な作業となってしまうであろう。仮にこうしたプロセスを自動化できたらどうであろうか？もし，どんなインフラであろうと，一貫した繰り返し可能なアプリケーションの構築が保証されるならどうであろうか？更に，ソフトウェアに対する統制も，同時に手にできるなら？

富士通が提供するFUJITSU Software UForge AppCenterは，開発，テスト，試験運用，本運用を通じてマルチクラウドやデータセンターを横断して繰り返し利用可能なプロセスを実現することで，上記問いへの解を提示する。

本稿では,アジャイル開発[(1)]を説明するとともに，アジャイル開発がDevOpsにどのように関連するのか，UForge AppCenterがどのようにしてDevOpsのハイブリッド環境対応を実現するのかについて述べる。

アジャイル開発

本章では，筆者が参加したアジャイル開発に関するワークショップの概要を紹介し，そこで得られたアジャイル型開発について述べる。

● ワークショップの概要

このワークショップで取り組んだ課題は，顧客を演じるインストラクターが提示した絵（**図-1**）を，各チームが再現するというものであった。4人一組で三つのチームに分かれ，各チームは，2名の「デザイナー」と2名の「アーティスト」で構成され，その4名が協力し合って絵を再現することが求められた。デザイナーには様々な図形とパターンで構成された絵が提示され，アーティストに対して自分が見たものを説明する役割が与えられた。一方アーティストは，デザイナーの指示を基に10分以内に絵を描き起こすことが求められた。このとき，デザイナーはアーティストに口頭で説明することが許されなかった。筆者は，アーティストの役割を演じることとなった。ワークショップのインストラクターが絵を見せるためにデザイナーを部屋から連れ出すと同時に，タイマーが始動した。6分後，チーム1のデザイナーが指示を書き留めた付箋紙を手に現れた。彼はアーティストに付箋紙を手渡すと，すぐに部屋を立ち去った。その直後，チーム2のデザイナーが走り書きのメモを持って現れ，自分のチームに手渡し去っていった。8分後，少しほっとしたことに，筆者らのチームのデザイナーがA4ページに詳細な指示を書いたものを持って戻ってきた。この時点で,私たちアーティストには，指示に従って絵を完成させる時間は，わずか2分しか残されていなかった。デザイナーから渡されたメモには以下のような指示が書かれていた。

・紙面の4分の3を使って四角形を描く，点線
・四角形の上に，三角形を描く
・….リストは続く.,,

私たちアーティストは全力を尽くし，10分が経過するまでに指示の20％を終わらせたことに大変満足していた。しかし，インストラクターがオリジナルの絵を見せてくれたとき，私たちの描いたものが，全く的はずれであったことを知った。ほ

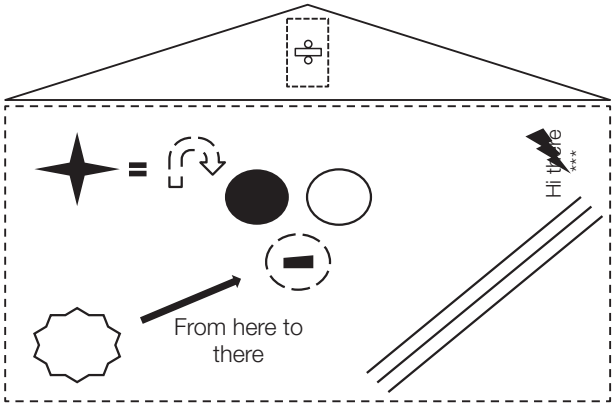

図-1　ワークショップで提示された絵

かのチームも同様であった。

この最初の取り組みの目的は，チームで共同作業をする際に，私たちがどのように振る舞うのかを明らかにすることであった。この中で，以下の三つの気付きが得られた。

第一に，デザイナーとアーティストに分けられた時点で，チーム内の力学に変化が生じた。デザイナーは絵の中の様々な形を書き留めることに責任を負い，アーティストはそれらを再現する役割となった。

第二に，6～8分が経過した時点で（アーティストが作業するために数分しか残されていない）アーティストは，ようやくデザイナーから必要なものを受け取った。

最後に，全てのチームが同じやり方で取り組んでいた。つまり，デザイナーが付箋紙や紙に書いた指示を手に戻ってくると，それらをアーティストに残し，すぐにまた取りこぼしていたデータを集めるために部屋を出ていった。このためアーティストは，与えられた指示を自分たちだけで理解し絵を描くしかなかった。

これら三つの気付きから分かることは，チームがもはや結束して問題解決に当たるのではなく，二つのサブチームとして動いているということであった。デザイナーは目にした図形を書き留めることによって自分の仕事をし，アーティストに作業全体を仕上げる仕事を受け渡した。そして，アーティストは，割り当てられた時間内で業務を終わらせるために，多大なプレッシャーを受けることとなった。

チームが全体の作業に責任を負うというよりも，むしろ責任を分割した結果，自然に壁が作られてしまった。つまり，自己中心的な仕事のやり方，すなわち「サイロ」が形成されてしまったのである。アーティストは，最初の情報を得るまでに非常に長い時間待たされた上，デザイナーが追加データを探すのに夢中になって部屋を後にしてしまったため，自分たちの理解が正しいのかフィードバックを得ることができなかった。

10分経過した後で，インストラクターは絵を見せながら「今なら，どのくらいの時間がかかると思うか」と尋ねた。振り返ってみると，私たちが描いた図形がお客様要件を満たしているのか，描き直したとして一体どれだけの時間がかかるのか，疑問を感じずにはいられなかった。私たちはこの取り組みの中で，デザイナーの誰もが，インストラクターに対して，顧客が求める成果のためにどの図形を描くことがもっとも重要なのか，聞こうとすら考えなかったことに気付いた。

● アジャイル型開発の解決

このワークショップにおいてチームが実践した方法は，ソフトウェア開発における伝統的なウォーターフォール型に非常に似ていた。

ウォーターフォール型開発において，アプリケーションはその開発プロセス内で複数のチーム間で受け渡しされながら，しばしば直線的に開発される。各チームは，次のチームにアプリケーションを手渡すより前に完了すべき作業の責任を負っている。プロジェクトマネージャー（ワークショップにおけるデザイナー）は，様々な情報源から要件を受け取る。そして，開発者チーム（ワークショップにおけるアーティスト）は，これらの要件を解釈し，アプリケーションを開発する。その後，アプリケーションはテストチームに送られて要件を満たしているか確認が行われる。要件を満たしていることが確認されると，リリースエンジニアチームは製品をパッケージ化する。最終的に，アプリケーションはお客様に届けられるか，監視された稼働環境に配信するために運用チームに手渡される。

ウォーターフォール型で開発する目的は，リスクを最小化することにあり，そして，そこにこそ問題がある。製品の各リリースに対してバラバラに働いているチームごとにチェックポイントが必要となるため，フィードバックが遅くなる。ウォーターフォール型開発では，開発チームはプロジェクト担当部分を改善する機会はたった一度しかない。

これに対してアジャイル型開発は，リスクを最小化するよりもアジリティ（機敏さ）を最大化することを優先する。それは，プロジェクトやプロダクトの範囲を限定することでもある。要求を最小限に絞った上で，提供可能な製品を作り上げる。アジャイル（機敏である）型開発とは文字どおり，速く効率的で，小規模・低コストで，少ない機能を持ち，短期間で完了する開発を意味する。[(2)]

● ワークショップの成果

これらを理解した上で，新しい絵で課題を繰り返したところ，違いは顕著なものであった。デザイナーが新しい絵を見に行ったとき，彼らがインストラクターに尋ねた最初の質問は，どの図形が最も重要かということであった。その後，30秒以内にデザイナー全員がどの絵を描くのかに関する簡単な指示書を持って，アーティストのもとへ戻ってきた。デザイナーはアーティストに指示書を渡して二つ目の図形のためにすぐに戻るのではなく，アーティストが図形を描き始めるのを見ていた。デザイナーたちは話すことを許されていないため，図形を改善するために「すこし左」「少し大きく」などの小さなメモを書き，フィードバックを提供した。こうすることで，二つ目の図形のために戻る前にアーティストが正しいサイズで，正しい場所に図形を描いたことを確かめることができた。

このようにして，チームは絵を描くという大きな仕事を小さなタスクに分解し，今取り組んでいるタスクに対してフィードバックを提供し，そのタスクを完了した後で，次のタスクに取り掛かった。仮に私たちが絵を完全に仕上げることができなかったとしても，私たちは最も重要な図形を仕上げることができた。言い換えれば，私たちはお客様が喜ぶ「最小の動く製品」（MVP：Minimum Viable Product）を作り上げた。

DevOps

DevOpsは人によって解釈が異なる。筆者らは，DevOpsとはコミュニケーションと協力を促進する環境であり，アジャイルとはその環境内での働き方であると考えている。DevOpsでは，高品質で十分にテストされた製品のリリースサイクルを最適化するという行動様式のもとで，様々なチームや部門が効率的な協力やコミュニケーションを行うことを重視している。[3), (4)]

このリリースサイクル，もしくはソフトウェア配信プロセスは，以下の七つのステップに要約される。

（1）BUILD

アプリケーションの開発フェーズである。開発チームが新しい機能の実装や拡張，バグ修正を実施する。

（2）TEST

アプリケーションの開発と並行して，テストが実行される。テストには，単体テスト（開発フェーズの一部とみなされることが多い），結合テスト，セキュリティテスト，性能テスト，受入テスト，シナリオテストなどがある。テストは，新しい開発によってリグレッション（機能の低下）が生じていないことを確認するのに極めて重要であり，現行機能の品質に関して開発チームにフィードバックする。テストは通常，開発チームと運用チームの双方で分担するが，専属の品質保証（QA）チームが担当することもある。

（3）PACKAGE

アプリケーションをテストした後，対象となる環境で動作させるために正しくアプリケーションをパッケージ化する必要がある。パッケージ化は，利用している技術もしくは環境に強く依存する。

（4）SHIP

アプリケーションをパッケージ化した後，ほかのチームやパートナー，購入するお客様が利用できるように出荷（SHIP）する必要がある。パッケージを登録し共有できるレジストリは，この目的のために使用される。レジストリの種類は，使われるパッケージの種類に依存する。

（5）DEPLOY

アプリケーションは，動作する場所を必要とする。抽象化されているとはいえ，コンピュータサーバ，ネットワーク，ストレージリソースを用意し，マシンイメージ，すなわちVMware環境におけるOVF（Open Virtualization Format）イメージや，KubernetesにおけるDockerイメージなど，起動可能なバイナリイメージからアプリケーションをインストールする必要がある。

（6）MONITOR

アプリケーションを実行した後は，正常に動作し続けていることを監視する必要がある。設定監視は，システムに設定ミスがないことを確認したり，許可されていない変更を検出したりするためにも使用される。

（7）UPDATE

アプリケーションが一旦配信された後も，時間の経過とともにセキュリティパッチや最新機能を配信し，アップデートされなければならない。最

終的には，不要になった時点でアプリケーションは破棄される。

アプリケーションを配信する時間を短縮するとしても，これらの七つのステップは省略できないため，各ステップの実行を最小化することが重要である。もし各チームが独立して動くための権限を与えられるなら，チーム間の調整は減少し，個人の生産性が向上する可能性がある。その結果，アプリケーションの配信時間を短縮できる。これがDevOpsの核心であると考える。一貫したプロセスを維持するために，ワークフローを重視したツールを選択する必要がある。そうすることで，ほとんどの組織で起こり得る様々な技術が混在する状況を，適切に扱うことができる。

UForge AppCenterを使ったDevOps

筆者らの所属するUShareSoftは，ソフトウェアの配信を容易にするために2008年に設立されたフランスのソフトウェア開発会社であり，2015年に富士通グループに加わった。主力ソフトウェア製品であるFUJITSU Software UForge AppCenterは，ハイブリッド環境におけるアプリケーションの自動配信と移行機能を提供する。

ハイブリッド環境においてDevOpsプロセスを実践するには，様々なインフラの上で，複数の部門・関係者が協同してアプリケーションを開発・運用できる仕組みが必要である。しかし，これらを人手で実現することは困難であり，複数のDevOpsツールを組み合わせても，部分的にしか実現できないことが多かった。UForge AppCenterは次のような技術を提供することで，前章で述べた七つのステップのうちPACKAGEからUPDATEまでの五つを自動化し，ハイブリッド環境におけるアプリケーション配信を加速できる（**図-2**）。[5]

（1）アプリケーションのテンプレート化技術

従来，ハイブリッド環境で利用されるアプリケーションは，OVFパッケージなどのようにOSやミドルウェアも含んだバイナリデータとして取り扱われることが多かった。しかしこのような形式では，可搬性に課題があるだけでなく，更新も容易ではないため，DevOpsプロセスで活用するのは困難であった。UForge AppCenterは，アプリケーションを構成するシステムをパッケージ構成などのメタ情報として記述すること（テンプレート化）により，可搬性を高めるだけでなく，パッケージの置き換えなどの更新も容易にする。

（2）対象環境に適したマシンイメージ生成技術

従来，ハイブリッド環境にアプリケーションを配信するには，対象環境に応じたドライバの適用やイメージ形式の選択など，動作環境ごとに異なる処理が必要であり，イメージ管理の負荷が増大する要因となっていた。UForge AppCenterは，メタ情報で記述されたテンプレートからイメージを生成する際，ユーザーが指定した動作環境に応じ

図-2　Uforgeの主要サービス

たドライバを自動的に適用し，その環境に適した形式でイメージを出力する。これにより，イメージ管理の負荷を大幅に削減する。

この特定のクラウドに依存しない「アプリケーションのテンプレート化技術」は，他社にない富士通独自の技術である。完全自動化の仕組み（BUILDおよびTESTを自動化する仕組みと，ステップ間をつないでリリースサイクル全体の進行管理を行う仕組み）とを組み合わせることで，リリースサイクルを短縮し，ハイブリッド環境におけるアプリケーション配信を飛躍的に加速できる。対象には，Fujitsu Cloud Service K5を始め，Amazon Web Services, Microsoft Azure, VMware, Dockerなど，多くの主要なクラウド環境が含まれる。

UForgeは，これらの技術により以下の機能を提供する。

(1) アプリケーションのテンプレート化技術を通じて，アプリケーション構造を管理者が理解可能な形式で「見える化」し，企業のDevOpsプロセスにソフトウェア統制をもたらす。

(2) パッケージを格納したリポジトリのモデル化を行う。これにより，アプリケーションに含まれるパッケージがリポジトリと同期して適切に更新されているか確認できる。一方で，パッケージ全体または一部を特定の日付に対応したバージョンに固定して，不用意に更新されないよう統制をかけることもできる。

(3) ソフトウェアスタックのモデル化を行う。そこには，OSのパラメーター（キーボード種別，タイムゾーン，パーティション設定など）をはじめとして，OSを構成するパッケージ間の依存関係，ミドルウェアやアプリケーションの構成，各種設定情報などが含まれる。これらをモデル化する技術は，クラウドを横断するフルスタックで一貫した開発への透明性をもたらす。加えて，顧客は企業内の全てのソフトウェアを監督し制御するための中核となる仕組みを構築できる。

(4) CI（Continuous Integration）や配信ツールなどを含むほかのDevOpsツールと組み合わせることで，開発から構築・テスト・リリース・配備に至る，自動化された再利用可能なDevOpsツールチェーンを企業内に構築できる（**表-1**）。

UShareSoft社の開発チームでは，DevOps分野における新たな取り組みの導入も継続的に行っている。直近ではハイブリッドICTにおけるブループリンティングの標準となっているApache Brooklyn[(6)]が該当する（**図-3**）。ブループリンティングとは，複数のノードで構成されたシステムをモデル化した記述形式であり，個々のノードはUForge AppCenterによってモデル化されたソフトウェアスタックである。このようなモデル化により，異なるクラウド環境に対して，多階層で構成されたソリューションを構築，配備できる。

UForge AppCenterを使った移行

UForgeは，オンプレミスからクラウド，クラウドからクラウドへのアプリケーションの移行を自動化できる。UForgeを用いて移行することにより，顧客は「ベンダーロックインの回避」と「アプリケーションのリファクタリング（外部仕様を保ったまま，保守性を高めるようにプログラムのソースコードを書き換えること）」を実現できる。

「アプリケーションのテンプレート化技術」と「対象環境に適したマシンイメージ生成技術」により，UForgeは移行対象となるアプリケーションを移行対象先環境への依存性を取り除いたテンプレート

表-1　UForgeを用いたDevOps

ステップ	実現手段
BUILD	Git，Gerrit，Jenkins，Sonarで実現
TEST	JUnit，Cucumber，Selenium，Jasmineなどで実現
PACKAGE	UForge AppCenterで実現
SHIP	UForge AppCenterとArtifactory，Nexus，DockerHubなどとの組み合わせで実現
DEPLOY	UForge AppCenter（ブループリントモジュール）で実現
MONITOR	UForge AppCenterとOSS（Apache Brooklynなど）との組み合わせで実現
UPDATE	UForge AppCenterとOSS（Apache Brooklynなど）との組み合わせで実現

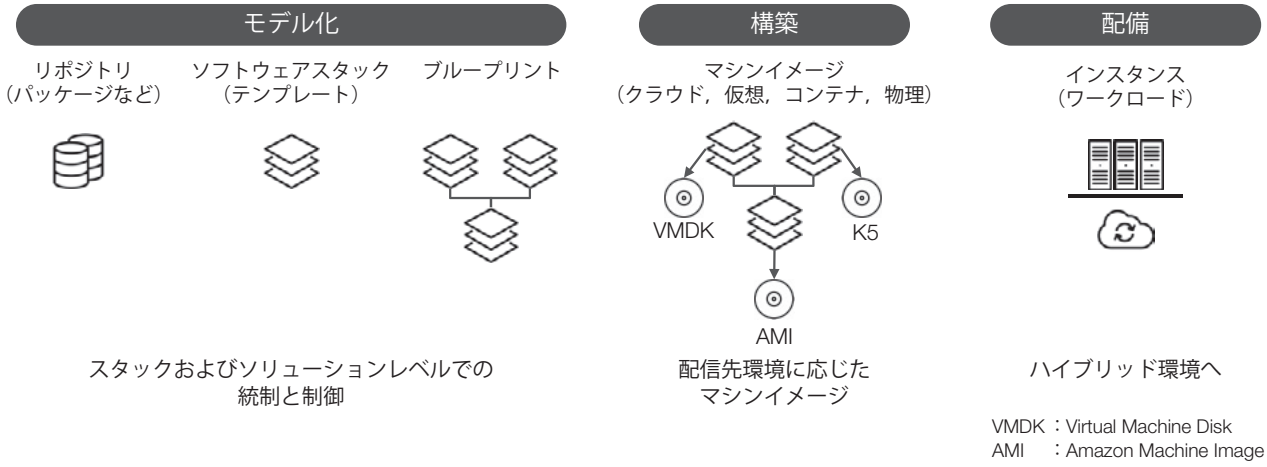

図-3　ブループリントのプロセス

として表現できる。これにより，プラットフォーム間でのアプリケーションの可搬性が高まり，顧客はUForgeがサポートする環境であればどのクラウドに対しても，対象となるシステムを移行できる。この結果，顧客は特定のクラウドやベンダーへのロックイン（過度に依存し，移行が困難になること）を回避できるようになる。

企業は，極力手間をかけずにアプリケーションを簡単に素早く移行でき（リフトアンドシフト），また移行中にアプリケーションをリファクタリングすることで，クラウド内の統制やライフサイクル管理，性能，そのほかのメリットを得ることもできる。このようにリファクタリングが可能となるのは，アプリケーションがテンプレートとして表現されているためである。テンプレート化されたアプリケーションに対しては，それを構成するパッケージの更新や削除，新たなパッケージの追加やユーザー定義のスクリプトの取り込みなど，リファクタリングに必要な操作が実施可能である。

む　す　び

DevOpsは，プロセス全体を俯瞰することで，プロセス内の役割の負荷が大きくなった際に発生するボトルネックを特定し，解決することを支援する。UForgeは，ソフトウェア配信を高速化するために，アプリケーション配信のライフサイクルの様々な段階で必要となる，人手による調整作業を減らすことに注力している。

UForgeを用いることで，DevOpsプロセスにハイブリッドな環境に対応する機能性を組み込むことができる。様々なクラウドで利用可能なアプリケーションのテンプレート化は，スピードと俊敏性に加えて統制と制御をもたらすことにより，開発，テスト，試験運用，本運用を通じてマルチクラウドやデータセンターを横断して繰り返し利用可能なプロセスを実現する。

参考文献

(1) Agile software development.
https://en.wikipedia.org/wiki/Agile_software_development
(2) An Introduction to Agile Methodology.
https://www.codeproject.com/Articles/704600/An-Introduction-to-Agile-Methodology?display=Print
(3) Demystifying DevOps：Difference between Agile and Devops.
http://www.agilebuddha.com/agile/demystifying-devops/
(4) Agile and DevOps：Friends or Foes?
https://www.atlassian.com/agile/devops
(5) UShareSoft website on UForge.
https://www.usharesoft.com
(6) Apache Brooklyn.
https://brooklyn.apache.org/

著者紹介

James Weir
UShareSoft，SAS.
Chief Technology Officer

Alban Richard
UShareSoft，SAS.
Chief Executive Officer

植田　譲（うえだ　ゆずる）
UShareSoft，SAS.
システムスキャンやイメージ生成など，ソフトウェアスタックのモデル化に関する開発に従事。

©2018 富士通株式会社