



Agile Japan 2015



“失敗から学ぶアジャイル、成功につなげるアジャイル”

米国事例にみる 「継続的デリバリー」によるアジャイル開発の拡張と、 それを実現するツールに求められる要件

日本CA株式会社
DevOps&アプリケーションデリバリ
プリンシパルコンサルタント
西野 寛史

2015/04/16

自己紹介

- 西野 寛史
- 趣味：
 - 飛行機の操縦（普段はシミュレータですが）
- 経歴：
 - SIerでアプリ開発者（16年間）
 - 自治体の税務・財務システム開発に従事
 - ウォーターフォールな日々
 - Java が大好き
 - 日本CAで **DevOps** を担当
 - “サービス仮想化” – テスト自動化
 - “継続的デリバリ” – デプロイ自動化



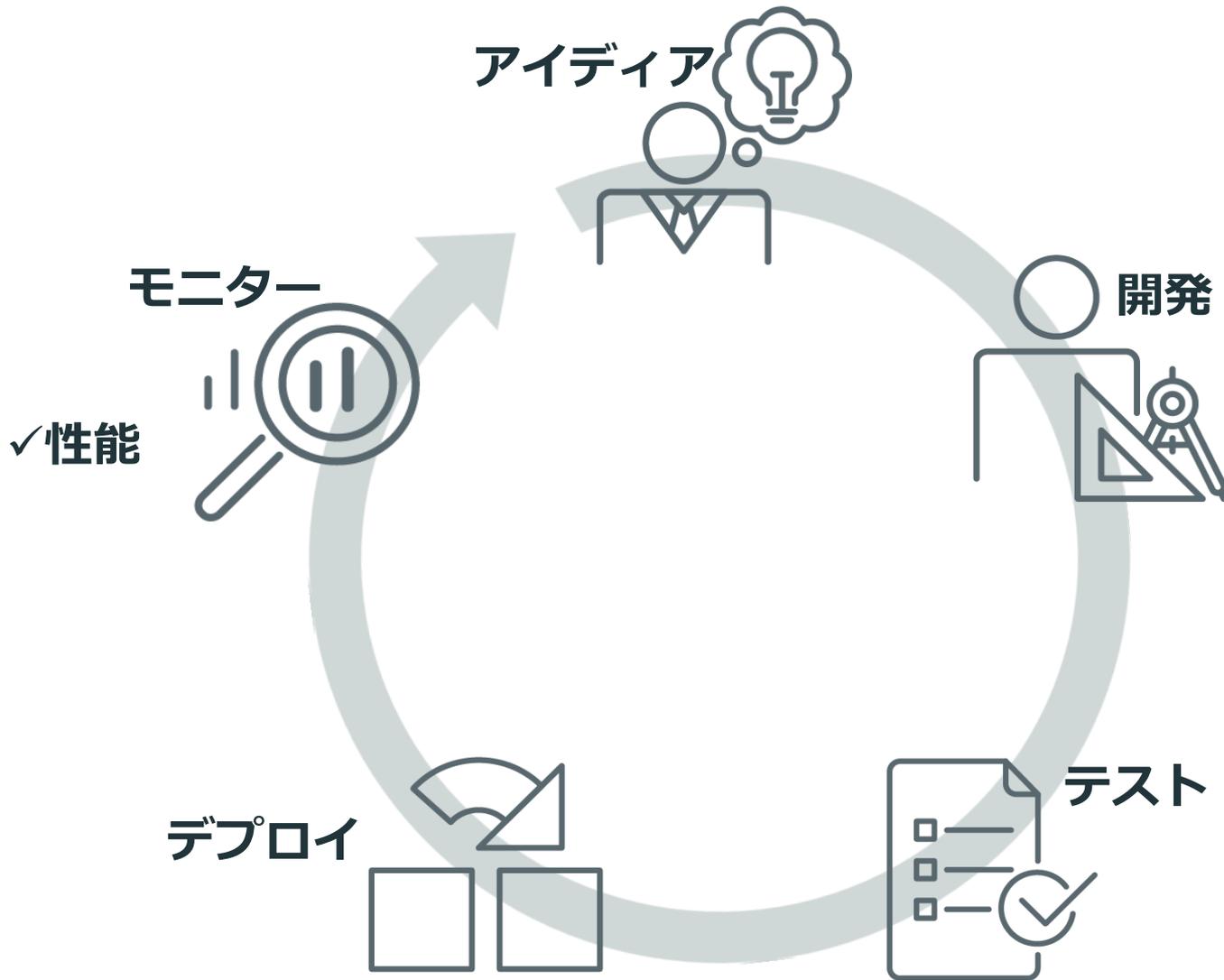
あらためて、DevOpsとは？

DevOps（デブオプス）は、ソフトウェア開発手法の一つ。開発（Development）と運用（Operations）を組み合わせたかばん語であり、開発担当者と運用担当者が連携して協力する開発手法をさす。ただし2013年現時点では厳密な定義は存在しておらず、抽象的な概念に留まっている。

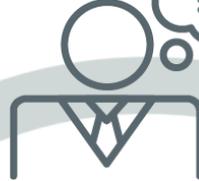
Dev and Ops

DevOpsという単語は2009年のオライリー主催のイベント「Velocity 2009」において、Flickrのエンジニアにより初めて公の場で用いられた。

このプレゼンテーションでは「開発と運用が協力することで、**1日に10回以上のペースでリリースが可能になる**」という発表とともにDevOpsという単語が用いられた。



アイデア  ✓リーンスタートアップ



開発



✓アジャイル開発
✓APIの活用

モニター



✓性能
✓ユーザー体験

DevOps

✓継続的
インテグレーション

今日の
フォーカス

デプロイ
✓継続的
デリバリー



テスト
✓自動化
✓サービス仮想化



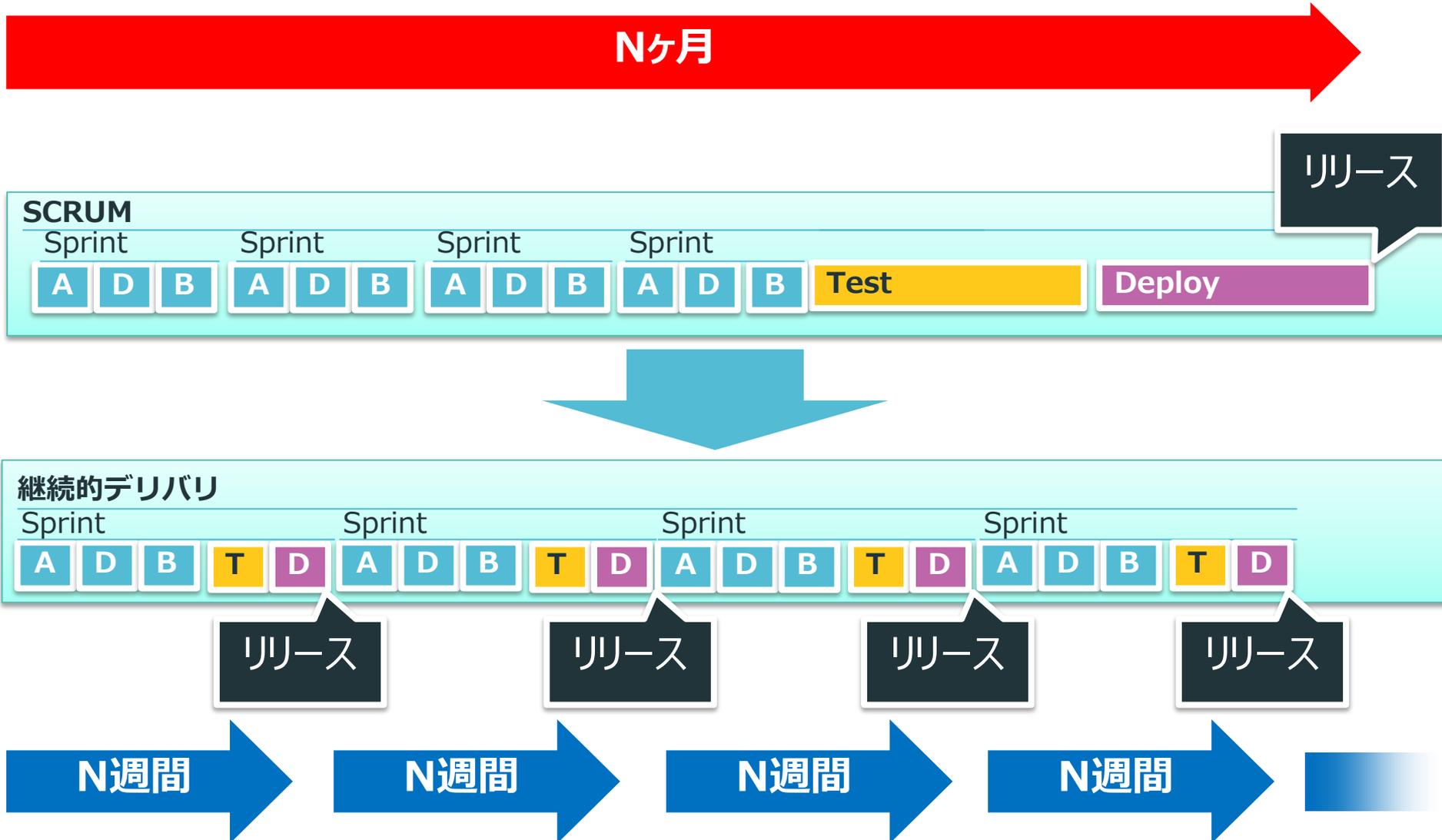
海外事例（グローバルな大手銀行）

1ヶ月あたり平均17,000デプロイ！

グローバルな大手銀行 DevOps Journey

- 2009 – モバイルバンキング App (アジャイル スクラム初導入)
 - スモールスタート
 - モバイルAppのリリースサイクルを13週から1週に短縮 → ビジネス価値を実証した
- 2010 – インターネットバンキングに拡張
- 2011 – コアバンキングシステムに拡張
 - 自動テストと自動デプロイ
- 2012 – 継続的デリバリー (CA Release Automation 導入)
- 2013 – DevOps
- 2014 – インフラをクラウドに移行

継続的デリバリーでアジャイル開発を“拡張” 市場投入時間を短縮



レッスン・ラウンド (1)

- スモールスタート、そして“let it grow” – 劇的な大変化を作らない
 - “grow”することには時間がかかることを学んだ – 上からの強制力はネガティブインパクトになる
 - 小さな成功体験で、周囲の人々の同意とサポートを獲得 – **成功を称賛!**
- DevOpsチームは、ビジネス部門、開発者、運用担当で構成され、**同じ上司のもとにある** (チームサイズは10人程度が良い)
- 各チームには小さくとも**顧客の目に触れる機能**を持たせ、**同じゴール**と (本番で問題が起きた場合の) **インシデント管理**を共有する

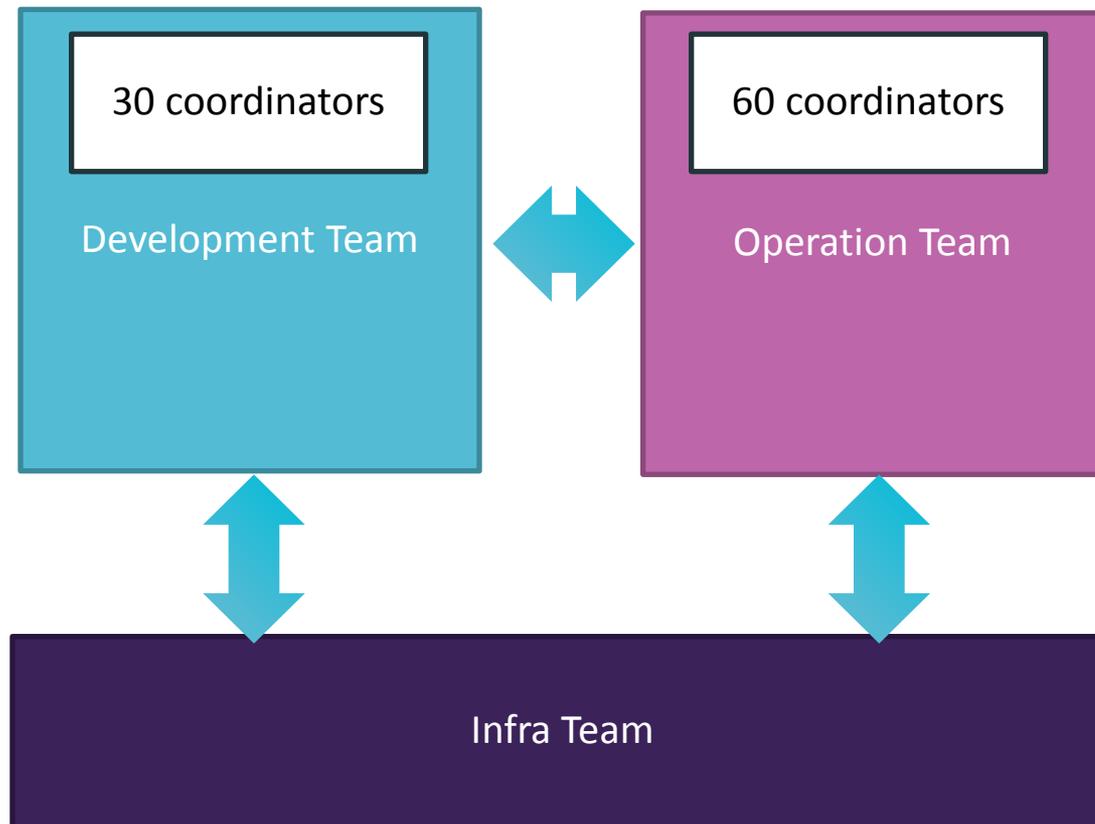
レッスン・ラウンド (2)

- プロセスの中に“センサー”を仕込む（計測すること）
間が発生しているか、どこが**スタック**しているか
 - コアバンキング – デプロイプロセス
 - モバイル – 手動テスト
 - インターネットバンキング – 構成管理プロセス
- クールな文化を醸成して、優秀な人材を惹きつける（優秀な人材を採用することは難しい）
- ナレッジ管理（コミュニケーションと情報共有を促進）

– どこで“待機時

以前のチーム構成

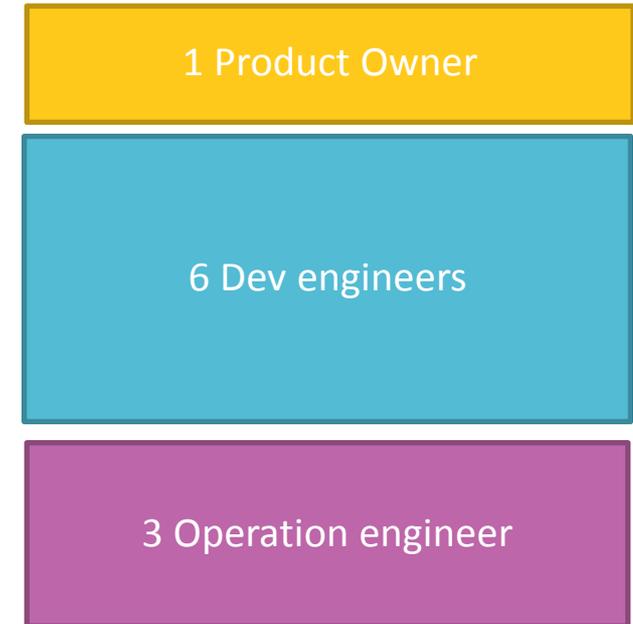
DevとOpsの間のアクティビティをコーディネートするために、多くのコーディネーター（技術者ではない）を擁していた



DevOpsチームとして再構成（コラボレーション促進のため）

各DevOpsチームは8～10人構成

- 現在、180のDevOpsチーム
- 1～5のアプリケーションをマネージ
- チームにプロダクトオーナーが1人。優先度判断を担う
- チームで、アプリケーションの開発とデプロイの責任を持つ
- アウトソースパートナー（インド）のメンバーも、チームにアサインされる
- コーディネーターはいない（不要）



複数チームでのコラボレーション

インテグレーターは、チームを
跨って責任を負う役割

同時にリリースチームの一員

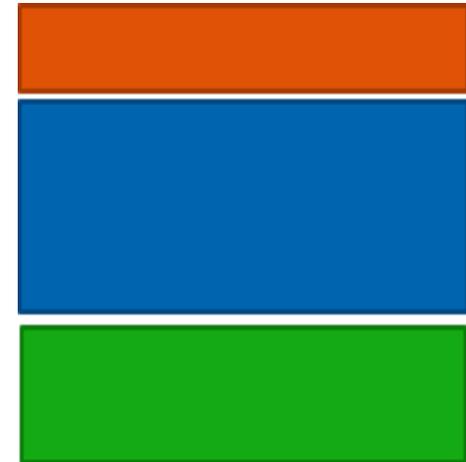
Integrators



Payment DevOps
team



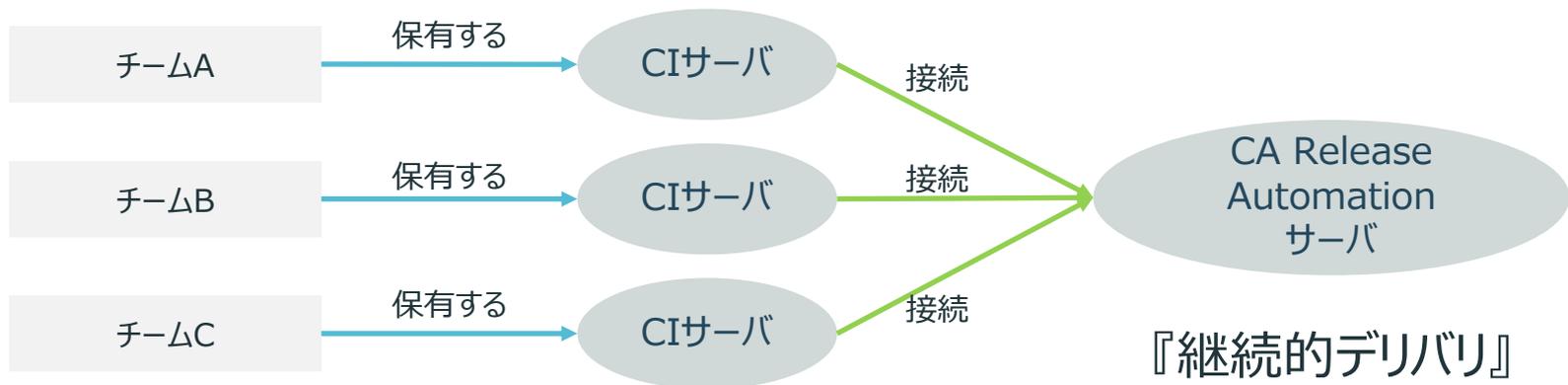
Channels DevOps
team



Savings DevOps
team

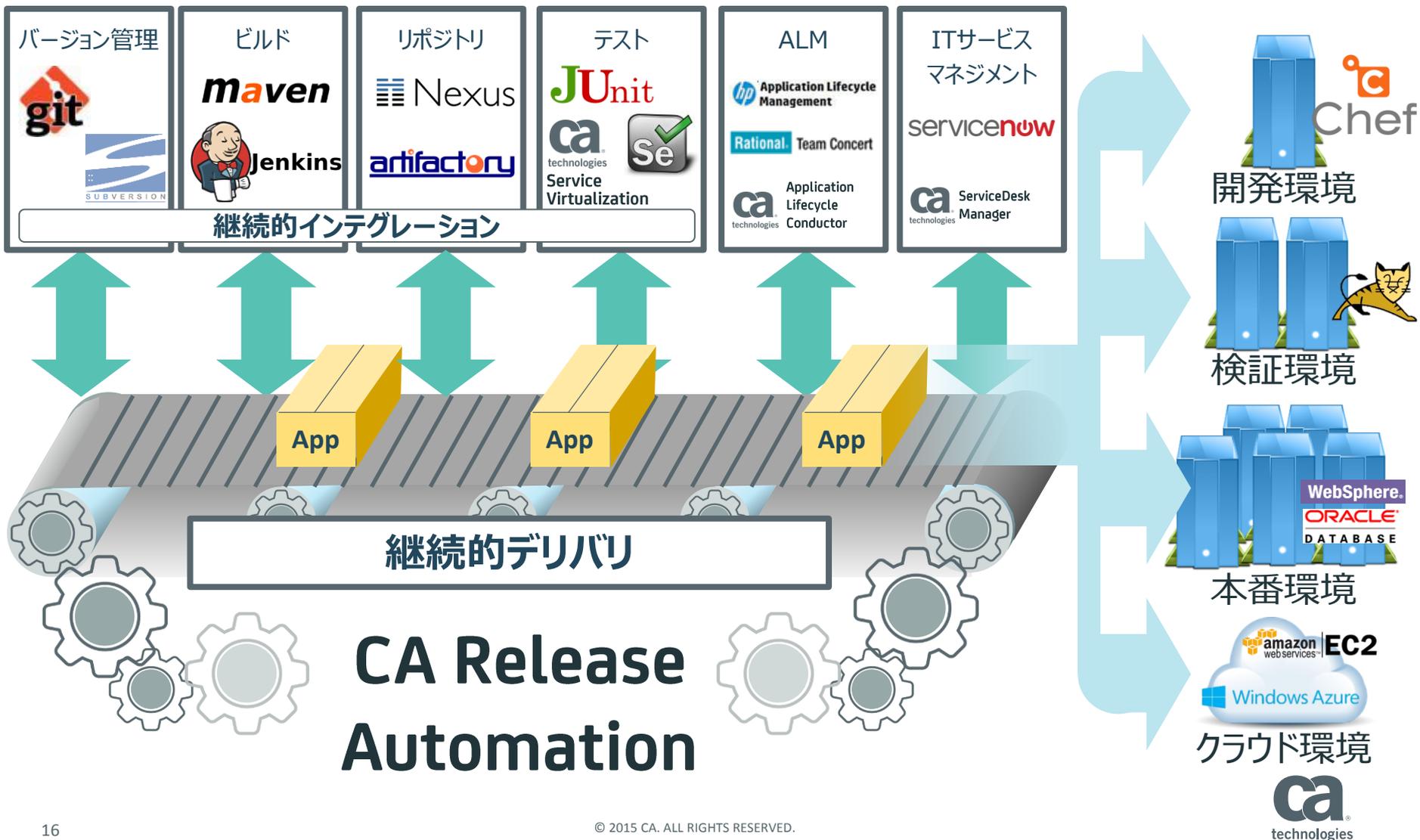
新規DevOpsチームに提供されるもの

- 経験のあるメンバーが、2～3スプリントの間だけ参加し、チームが自律するのを支援
- 単一のサーバに、継続的インテグレーションに必要なすべてのソフトウェアをインストールして提供（サーバに含まれるソフトウェア） → 「チーム用CIサーバ」
 - Jenkins
 - Maven
 - Sonar
- すべてのDevOpsチームはCIサーバを持ち、それらは同じ CA Release Automation サーバに接続されている



継続的デリバリ

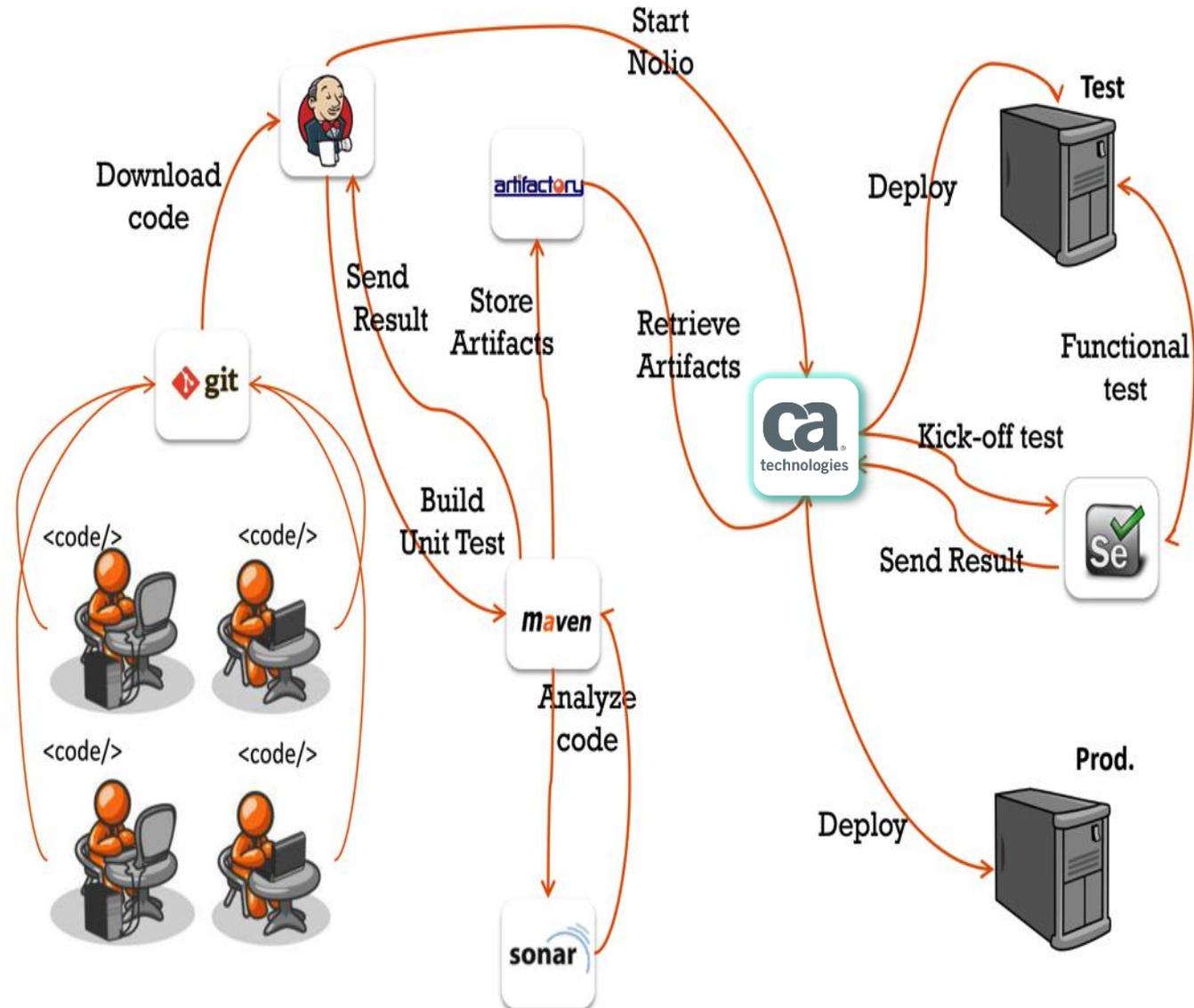
開発環境、テスト環境、ステー징／本番環境に対してベルトコンベアに乗せるように、成果物をリリースできるメカニズム



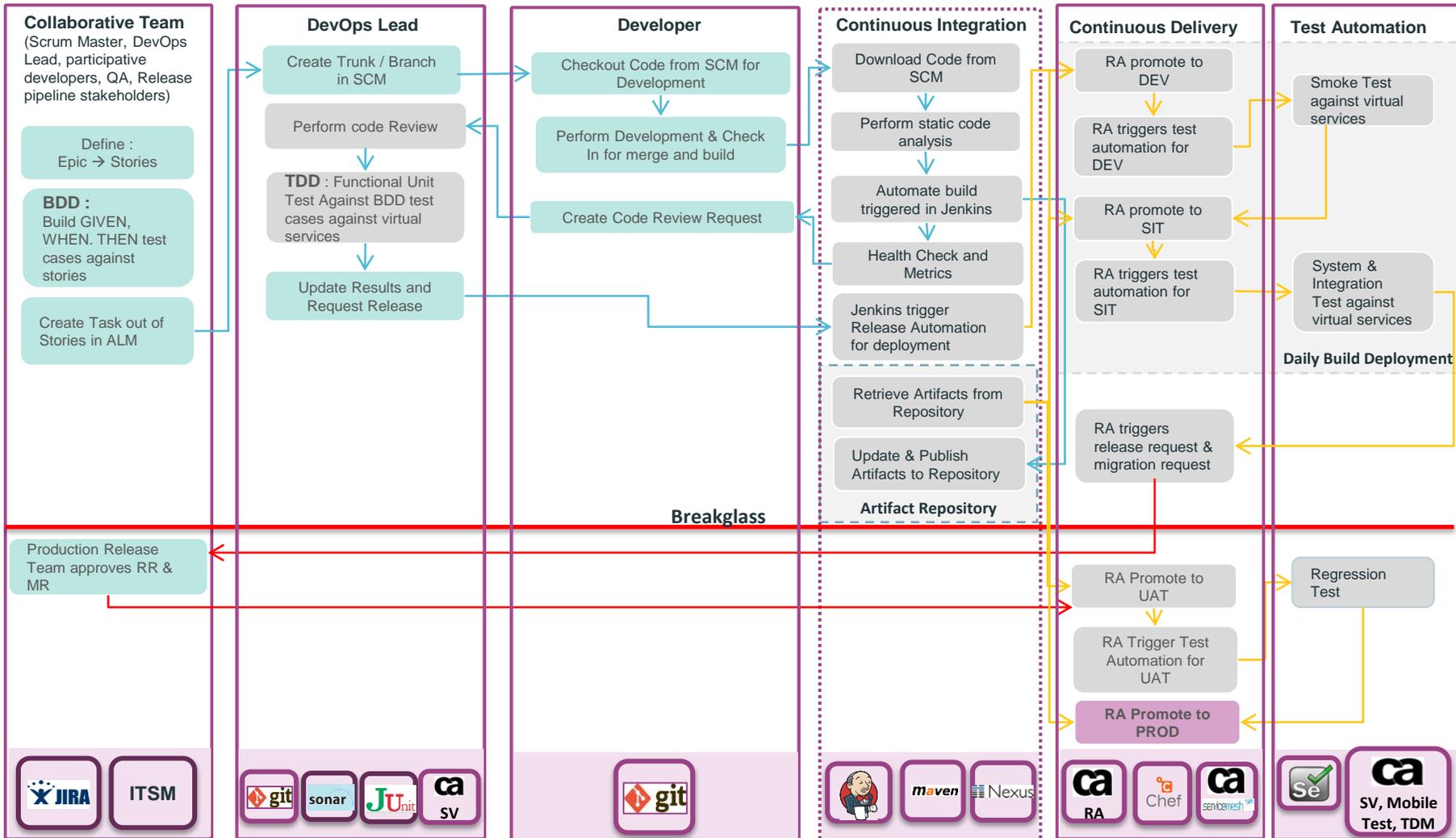
継続的デリバリー パイプライン

Global Bank preference for:

- Git – supports decentralized development better
- Artifactory – Works well with Java and non-Java
- ServiceNow
- CA RA for Automated deployment
- Evaluating provisioning tool to replace TSAM



Likely Process and Tools Landscape



グローバルな大手銀行

CA Release Automation による継続的デリバリ

2 週間で 1 デプロイ
リリースサイクル：13週間

1週間で 52 デプロイ
リリースサイクル：2 週間

(複数環境にわたって)
1週間で 1000 以上のデプロイ
リリースサイクル：1日あたり 350

WHY CA Release Automation :

- ✓ 標準機能でユーザの権限管理ができる
- ✓ 物理的なデプロイターゲットを抽象化できる



コアバンキング、インターネットバンキング、B2Bバンキングへ展開：

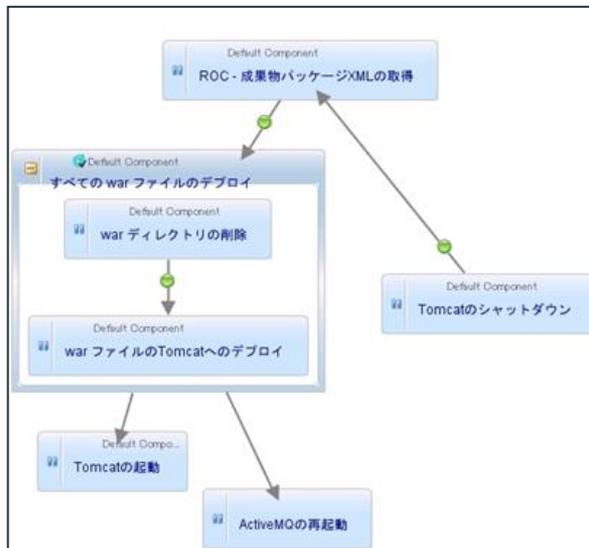
- **インターネット**
デリリーで全ての開発ソフトウェアの全自動テストが、新機能の開発スピード向上をリード
- **モバイル**
顧客フィードバックをもとに定期的にモバイルアプリを更新

- ✓ GUIですべてができる (他ツールはスクリプト必要)
- ✓ ダッシュボードやレポートが提供されている

「継続的デリバリ」実現ツールに求められる「要件」 - 1

物理的なデプロイターゲットを抽象化

プロセスは
1つ



ターゲットは
複数



プロセスの“抽象化”

「継続的デリバリ」実現ツールに求められる「要件」 - 2

GUIですべてができる（スクリプトを排除）

The screenshot shows the Automation Studio interface. On the left is a navigation tree for 'Customer Portal' with sub-items like 'Default Architecture', 'Deployment', and 'Application Full Deploy'. The main workspace displays a process flow for 'Web Server Type' with steps like 'IIS Server - Stop'. On the right, an 'Actions' panel lists various actions such as 'Amazon EC2', 'WebLogic', and 'WebSphere'. Red annotations highlight the 'デプロイ・プロセス' (Deployment Process) and 'アクション' (Action) areas.

アクション

デプロイ・プロセス

スクリプト排除できると、なにが良いか？

- ➔ 属人性の排除
- ➔ 開発と運用のコラボレーション

CA Release Automation

継続的デリバリーを実践するためのリリース自動化ツール

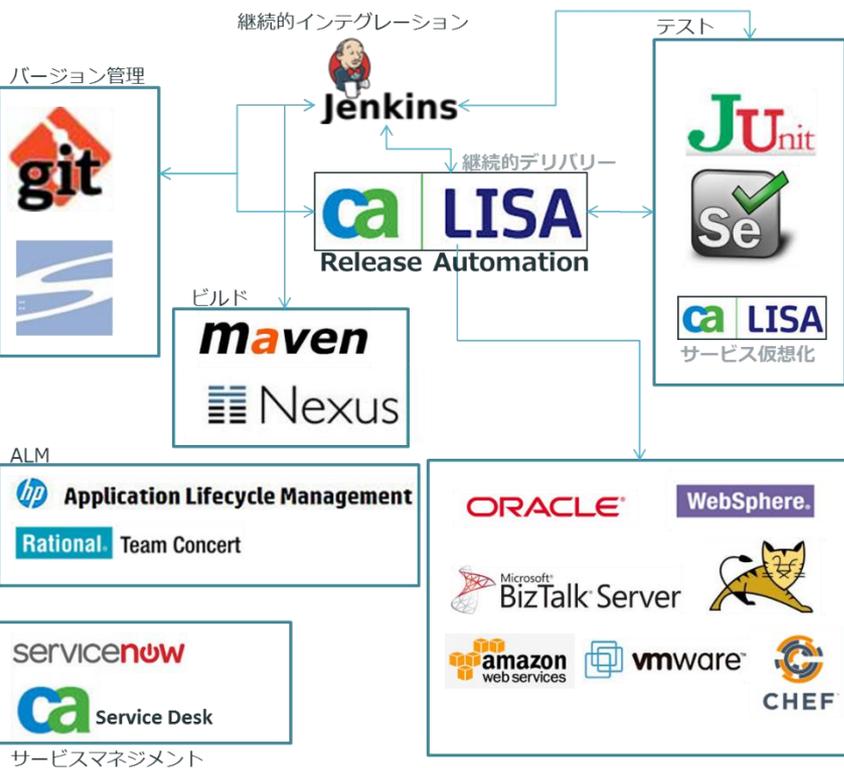
アプリケーションの開発から運用にいたるライフサイクル全体にわたり、複数層から構成されるWebシステムやクラウド環境に対して、配布作業フローの設計、テスト、実行といった、アプリケーション配布の自動化を行うための包括的なソリューション

ユースケース

1. メインフレーム、オープンシステムを含む複雑で大規模なアプリケーションのリリースを一元管理
2. ALMツールやITサービス管理ツールとの連携により監査ログを強化
3. モバイルやクラウドアプリケーションのような変更頻度の高いアプリケーションのリリースを自動化
4. アジャイル開発を継続的デリバリーで補完

ビジネスバリュー

1. リリースに要する要員と時間、エラーの低減
2. Time to Marketの改善



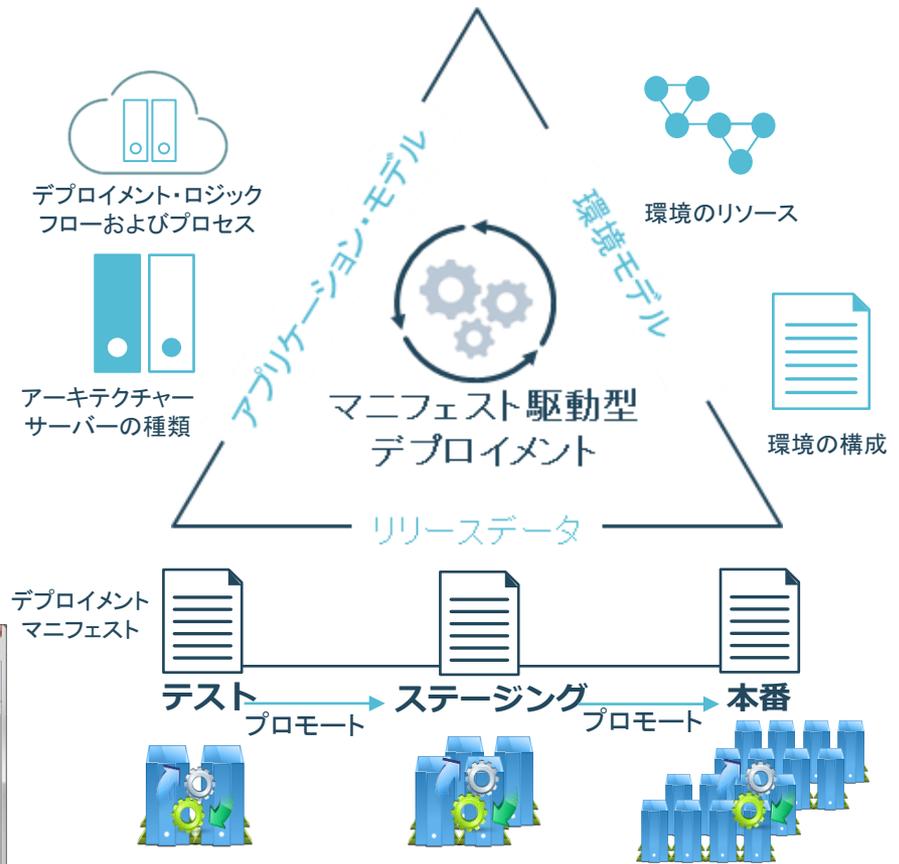
特徴1 直観的なグラフィカル・ワークフローエンジン

The screenshot displays the Automation Studio interface. On the left is a 'Navigation Panel' with a tree view showing a project named 'Lisa Cars' with sub-items like 'DEV (Default Architecture)', 'Production (Default Architecture)', and 'Processes (7)'. The 'Deploy Lisa Cars App' process is selected. The main workspace shows a 'utility server Application Server' environment with a graphical workflow. The workflow consists of several 'Default Component' boxes: 'ROC - 成果物パッケージXMLの取得' (top), 'すべての war ファイルのデプロイ' (middle), 'war ディレクトリの削除' (bottom), and 'Tomcatのシャットダウン' (right). Arrows indicate the flow from the 'ROC' component to the 'Deployment' component, and from the 'Deployment' component to the 'Tomcat Shutdown' component. The 'Tomcat Shutdown' component also has an arrow pointing back to the 'ROC' component, suggesting a loop or a specific dependency. The interface includes a menu bar (File, Activity, Administration, Help), buttons for 'New Process', 'Edit Process', and 'Run Process', and a status bar at the bottom showing 'http://localhost:8080'.

- アプリケーションリリースプロセスをグラフィカルに定義
- 多彩なミドルウェアやプラットフォームに対するアクション(900種類が事前定義済み)を選択しながらプロセスを設計
- スクリプト作成の手間とエラーを削減

特徴2 マニフェスト駆動型によるリリースの“抽象化”

- 再利用可能な「汎用リリースモデル」により、迅速でスケーラブルで信頼性の高いリリース操作を行うことが可能
- 汎用モデルとデプロイメント・データを分離(成果物、リリースデータと環境の構成など)することで、あらゆるデプロイメントタイプに対して繰り返し可能/再利用可能なリリースプロセスを構築可能



```

<?xml version="1.0" encoding="UTF-8"?>
<ApplicationDeploymentManifest Version="1.1" Status="Approved">
  <ProcessRun>
    <!-- This node contains information about the process run. This in case a manifest file is created for every process run. It can also contain a Result node where Nello can update the information during/after the process run to have both the process setup info as well as the results in 1 place-->
    <ProcessRun Version="3.1" Status="Approved" Buildnr="B127" Destination="Test 1" Number="001"/>
    <Result> </Result>
  </ProcessRun>
  <ProcessNames>
    <Environments>
    <Locations>
    <!-- this node contains the key application resources per version of the product - this is typically the items to be checked out, distributed and installed, best with version control as this changes over time-->
    <!-- the product version number can be higher in the hierarchy in case all other nodes are really dependent on the product version number - to be considered by project-->
    <Product Name="FinancialTradingPlatform">
      <Versions>
        <Version Nbr="1.0">
          <Components>
            <Component Version="1.2" Name="Transaction Service"/>
            <Component Version="1.2" Name="Brokerage Service"/>
          </Components>
        </Version>
        <Version Nbr="2.0">
          <Components>
            <Component Version="1.2" Name="Transaction Service"/>
            <Component Version="1.5" Name="Brokerage Service"/>
            <Component Version="3.1" Name="Gateway Service"/>
          </Components>
        </Version>
        <Version Nbr="3.1">
          <Components>
            <Component Version="1.2" Name="Transaction Service"/>
            <Component Version="1.6" Name="Brokerage Service"/>
            <Component Version="3.1" Name="Gateway Service"/>
            <Component Version="1.1" Name="CRM Service"/>
          </Components>
        </Version>
      </Versions>
    </Product>
  </ApplicationDeploymentManifest>
  
```

特徴3 豊富なアクションパックとプラグイン

- スクリプトを書かずに、主要なサードパーティ製エンタープライズ・ソリューション (開発、運用、サービスマネジメントツールなど)との連携が可能
- Rapid Development Kitにより、コード生成とパッケージングを迅速かつ一貫性をもって実装可能
- お客様の既存の投資を保護しながらリリースの自動化を実現



デモンストレーション

1. デプロイプロセスのデザインと実行 みどころ

- ・物理ターゲットの抽象化
- ・スクリプト不要の「動く手順書」

2. 継続的デリバリ みどころ

- ・CI(Jenkins)と自動デプロイ・自動テストの連携

The image displays two screenshots from a CI/CD pipeline demonstration. The top screenshot shows the Automation Studio interface, where a deployment process for 'Customer Portal' is being configured. The process includes steps like 'Web Server Type', 'IS Server', 'Deploy Artifact', 'Artifact Clean-up', and 'Delete Staged Artifact'. The bottom screenshot shows the Jenkins dashboard, which provides a real-time overview of the pipeline's status. It includes a 'Deployments' table and a 'Build Queue' section.

Deployment name	Environment	Deployment Plan	Description
Derivatives Platform Refresh FY14 Update 1.28.533	Development Ver 1.28.533 (Minor)		Derivatives Platform Refresh FY14 Update 1.28.533

#	Status	Name	Last Success	Last Failure	Last Du
1	Idle				
2	Building	Derivatives Platform #533	19 days - #531	3 mo 0 days - #514	6 min 4

まとめ

- DevOps – 俊敏性（アジリティ）を高める、が主語
- グローバルな大手銀行でのDevOps
 - アジャイル スクラムは2009年から着手し展開
 - 課題：テスト と デプロイ → 俊敏性のボトルネック！
- 解決に向けて・・・
 - 組織論（DevOpsチーム）、そして文化論
 - 技術論（継続的デリバリ）
- 継続的デリバリ – DevOpsツールチェーンを統べるベルトコンベアー
- 継続的デリバリーを実現するツールに求められる要件
 - 『抽象化』
 - 『GUI志向』 → 属人化低減 と コラボレーション
- CA Release Automation ご紹介

話は変わりますが...

尺の関係で、今日は詳しくご紹介できませんが・・・



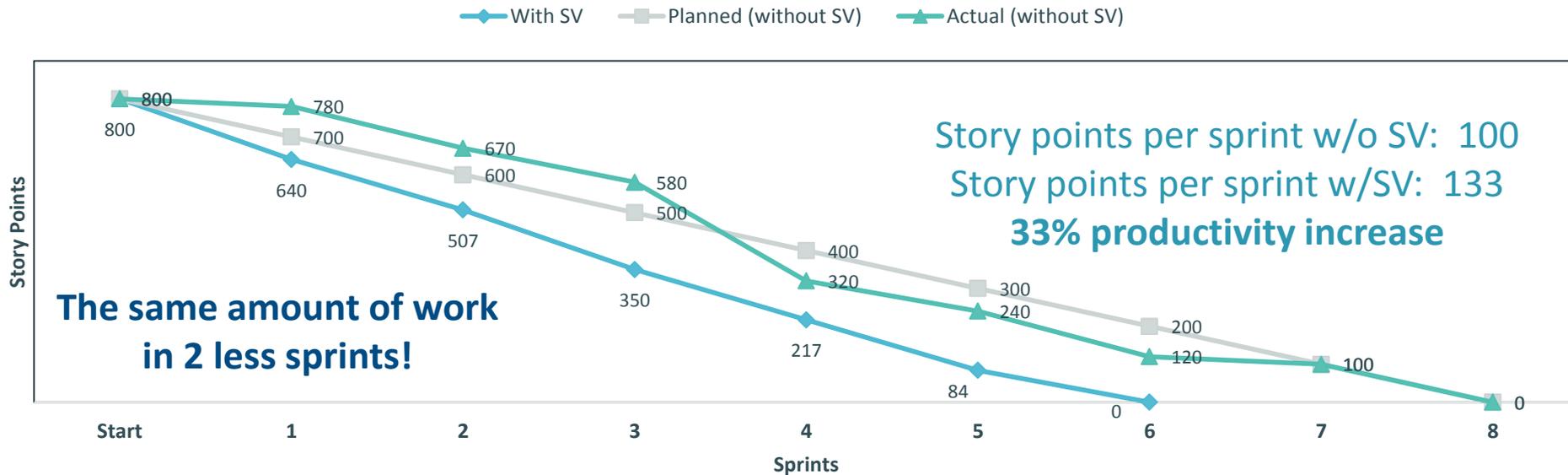
テストの自動化って、難しくないですか・・・？

環境が揃わない、データが合わない、連携先がまだできてない・・・

CA Service Virtualization が解決します

CA Service Virtualization はアジャイルの俊敏性を向上させます

- Service Virtualization **removes constraints**, making developers more productive and increasing the volume of work they can complete in a sprint. Service Virtualization **increases sprint velocity**



CA Service Virtualization - 詳しくはこちらで！

開催概要

名称	新クラウドサービス「QTaaS」ご紹介セミナー ふるまい仮想化が変革するアプリケーション・テスト
開催日	2015年4月24日(金) 15:00～17:30 (14:30受付開始)
場所	クオリカ本社セミナールーム >>アクセス
対象者	<ul style="list-style-type: none">・アプリケーション開発標準化に携わる方・アプリケーションのテストプロセスを構築・管理されている方・オープンソースのテストツールを利用している方・負荷テスト、結合テストを実施するエンジニアの方
主催	クオリカ株式会社
共催	CA Technologies
参加費	無料
定員	50名

“QTaaS”で検索！



米国事例にみる
「継続的デリバリー」によるアジャイル開発の拡張と、
それを実現するツールに求められる要件

日本CA株式会社

Hirofumi.Nishino@ca.com



in

ca.com