アプリケーションのライフサイクル全体を支えるOSSとその動向

「DevOpsの重要性はなんとなく認識しているが、その実態がつかめない」「どうやって推進すれば良いか分からない」という方も多いのではないでしょうか？　また、CI/CD（継続的インテグレーション/継続的デリバリー）はやっているが、開発を楽にする「あった方が良いもの」と捉えていて、「必須である」との認識をしていない方も多いようです。

本稿では、アプリケーション・ライフサイクルの観点から、なぜ今DevOpsが重要であり、CI/CDが「必須」とされているのかについて、それらを支えるオープンソースソフトウェア（以下、OSS）の動向とともにお伝えします。

1. はじめに

ITとビジネスの関係はこの20年で随分と様変わりしてきました。ITは、2000年代にはビジネスに「有効」とされていましたが、2010年代には「不可欠」に、そして2020年代は「コア」なものへと、その重要性がより増してきています。デジタルトランスフォーメーション（DX）の必要性が注目されたことで、既存の確立したビジネスモデルのIT化が主な役割だった時代から、市場の求めるアプリケーションをいかに迅速に、そして流動的に変化する要求に追随して継続的に改善してリリースするかが重要な時代になりました。

これに伴ってアプリケーションのライフサイクルの考え方も変化してきました。アプリケーション開発に至るアイデア段階から廃止までのすべての期間のプロセス管理をするApplication Lifecycle Management（以下、ALM）や、開発チームと運用チームがお互い協調するDevOps、それらを効率的に機能させるためのプロセスとしてのCI/CD（Continuous Integration/Continuous Delivery）といった考え方です。そして、CI/CDを実現するためのOSSとしては、後述する「Continuous Delivery Foundation」（以下、CDF）発足に多大な貢献をした川口耕介氏が産み出したJenkins[1]が有名ですが、この他にもさまざまなOSSが存在し時代のニーズに合わせて現在も進化を続けています。本稿ではこれらアプリケーション・ライフサイクル管理の考え方と、それらを支える各種OSSとその動向、さらには導入に際しての課題とその解決策の一つとしてIBMのソリューションを紹介します。

2. ALMとDevOps

アプリケーション・ライフサイクルの始まりはビジネスニーズから発生し、そこからアプリケーションの開発とリリース、運用と最終的にビジネス価値を持たなくなるまでこれが繰り返されます。この「ビジネス」「開発」「運用」という3つの要素をそれぞれバラバラに考えるのではなく、サイクル全体を総合的に管理することでアプリケーションの品質や開発生産性、市場の変化への対応力を向上させてビジネスニーズに合致させるように統制していくという考え方がALMです。ALMはソフトウェア開発ライフサイクル（以下、SDLC：Software Development Life Cycle）と混同されがちですが、SDLCよりも広い範囲を包含しており、SDLCは開発における1ライフサイクルを指しているだけであるということも意識しておく必要があります。つまり、ALMとはアプリケーションをリリースしたら運用に引き継いで終わりとするのではなく、ビジネスニーズに基づいて繰り返し改善しその価値を市場に届け続けるという考え方に根差したものであると認識すべきでしょう。

こうしたALMの考え方に基づいて、アプリケーション・ライフサイクルを繰り返し回すためには「開発」と「運用」が協調することが大事になってきます。開発（Dev）と運用（Ops）はその役割の違いから衝突する傾向にあり

ますが、より良いアプリケーションを迅速に、そして安定的に提供するという双方が同じゴールを目指して協調していくとする文化や組織の在り方、それを前提としたCI/CDの運用を含めてDevOpsと呼称します（**図1**）。

3. CI/CD

ALMの「ビジネス」「開発」「運用」の3つの要素は密接に関連しており、これらをつなげて適切に管理、実施することは非常に困難です。そして、冒頭でも述べたとおりこれらを関連付けて効率的にALMを機能させるためのプロセスがCI/CDです。

ご存知の方も多いとは思いますが、ここでCI/CDについておさらいしておきましょう。一般的にCI/CDは「継続的インテグレーション／継続的デリバリー」と定義されることが多いのですが、ALMの構成要素と対応付けるなら、以下の3つからなる定義とするのが適切ではないでしょうか。

●Continuous Integration（継続的インテグレーション）

「開発」向けの自動化プロセスです。アプリケーションコードを定期的に、あるいはGitなどのSCM（Source Code Management）へのプッシュをトリガーとしてビルド、テストが実施されます。これにより、開発者間でのコードの不整合によりビルドが壊れることを防止、または早期に発見することが可能になるだけでなく、SonarQube[2]などのツールと連携して自動テスト・静的コード解析を実施することで、コード品質を高めることが可能です。

●Continuous Delivery（継続的デリバリー）

「開発」と「ビジネス」の橋渡しをするプロセスです。ビジネス側に動作するアプリケーションを迅速に提供することでリリース予定の機能を可視化し、両者のコミュニケーション不足による仕様の齟齬（そご）を極小化します。場合によってはここでSelenium[3]などを利用してE2Eテストを実施、またそのテスト自体も自動化するケースもあります。

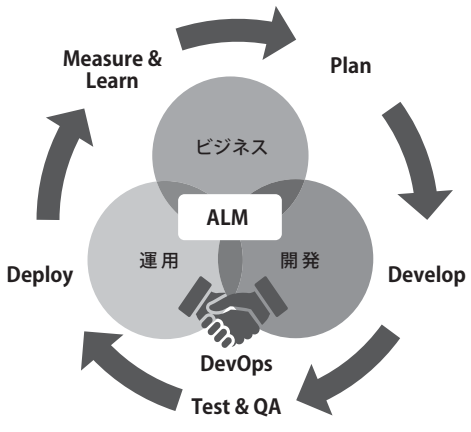

図1. ALMとDevOps

●Continuous Deployment（継続的デプロイメント）

「運用」によって実施される本番環境に対するアプリケーション導入作業を自動化するプロセスです。手動作業を排して自動化することで、人為的なリリース・ミスを削減するだけでなく、顧客に対してアプリケーションを提供するスピードを向上させます。

継続的デリバリーに継続的デプロイメントを含めて語られることもありますが、言葉の定義はどうであれ、CI/CDはALMにおける3要素（「ビジネス」「開発」「運用」）をつなげてそのサイクルを効率的に回すための、またはDevOpsを推進するためのプロセスであるということが分かるでしょう。

4. CI/CDのこれまでと昨今のトレンド

CI/CDは、Jenkinsやその機能を拡張するさまざまなプラグインがOSSとして公開され発展してきたことで、誰もが比較的容易にその環境を構築できるようになりました。こうしてCI/CDが一般化しその重要性が増してくると、新たな問題も発生してきます。適切に管理しないと、乱立するJenkins環境や肥大化したビルドジョブによるブラックボックス化で、メンテナンスや移行が困難となる状況も多くなってきました。さらにはクラウドネイティブ時代の到来により、プライベート／パブリック・クラウド、ベンダーを問わず状況に合わせて使い分けるマルチクラウド構成をとる企業も増えてきたことで、より一層その管理は困難な状況となりました。こうした状況の中で昨今のCI/CDを支えるOSSはどのようなトレンドとなってきているのか解説していきます。

(1)Configuration as CodeとSingle Source of Truth

JenkinsではGUIを使用して比較的簡単にジョブを構築可能であることがメリットでしたが、これは前述のとおりジョブを作成した本人しか分からないブラックボックス化を引き起こす問題があります。これに対応するため、

JenkinsではGUIでの設定ではなく、Jenkinsfileという設定ファイルにジョブの定義を宣言的にコードで記述していく「Configuration as Code」が推奨されます。ジョブ定義を決められたフォーマットでコードによって記述することにより、ジョブの見通しが良くなるだけでなく、これをGit上で管理することでそのジョブを「信頼できる唯一の情報源」(Single Source of Truth)として管理することが可能となります。つまり、GitからJenkinsfileを取得して実行することで異なるJenkins環境であっても同じジョブを実行できます。これにより乱立するJenkins環境のジョブを統一的に管理できるようになるだけでなく、ジョブの移行容易性も担保されることになります。これらはその他のCI/CDを実現するOSSツール(Concourse CI[4]、Gitlab CI/CD[5]など)もその記述方法は異なりますが同様の傾向となります。

(2)SaaS化/サーバーレス化

前述したConfiguration as CodeとSingle Source of Truthが前提となると、そのジョブの実行環境は場所を選ばなくなるだけでなく、必要な時にだけ利用するという考え方が一般的となります。つまりCI/CD環境を管理、運用する必要性がなくなってくるということです。CI/CD環境を提供するSaaSを利用して、Jenkinsfileのようなジョブを記述したコード(とビルド対象のソースコード)をSaaSに渡すことで、必要に応じてジョブを実行してくれるようになります。また、SaaSを利用しない場合であっても「Kubernetes」や「Red Hat OpenShift」のようなコンテナ・オーケストレーション・ツールと連携することで、必要に応じてジョブの実行環境コンテナが起動してジョブを実行し、不要になったらそのコンテナが破棄される、というようなサーバーレスなジョブの実行環境が提供されます。これらはSaaSとしては実にさまざまな種類が展開されており、サーバーレス対応やKubernetesネイティブなCI/CDとしてもJenkinsだけでなく、Jenkins X[6]、Tekton[7]といったOSSが登場してきています。

(3)マルチクラウド対応・標準化の推進

クラウドネイティブ時代にCI/CDを有効に機能させるには、マルチクラウドなDevOps戦略を志向して特定のクラウドにロックインされないようにさまざまなOSSツールを活用していく必要があります。また、このようにJenkinsだけでなくSaaSを含めたさまざまなCI/CDツールが登場するようになると、ツール間のジョブの移植性についても注意を払う必要が出てきます。このような状況下で、The Linux Foundationは2019年3月12日(米国時間)に先述のCDF[8]の発足を発表しました。プレスリリースでCDFの活動は「ベンダー中立的にCI/CDおよびDevOpsメソドロジーの啓蒙を行いながら、世界中のあらゆるソフトウェア開発チームがCI/CDのベストプラクティスを実装できるようにする」と記述されており、ジョブの記述方式を標準化してツール間の移植性を支援するとともに、Jenkins、Jenkins X、Tekton、Spinnaker[9]といったCI/CD関連のOSSをホストすることを発表しています。このようにCI/CDをベンダー中立の立場から推進していく、CDFの今後の動向に注目していく必要があるでしょう。

5. DevOpsを実現する際の課題

DevOpsを実現するためには、前述のとおりOSSツールなどを利用してCI/CD環境を構築して開発作業や運用作業の自動化を進める必要があります。また、作業の自

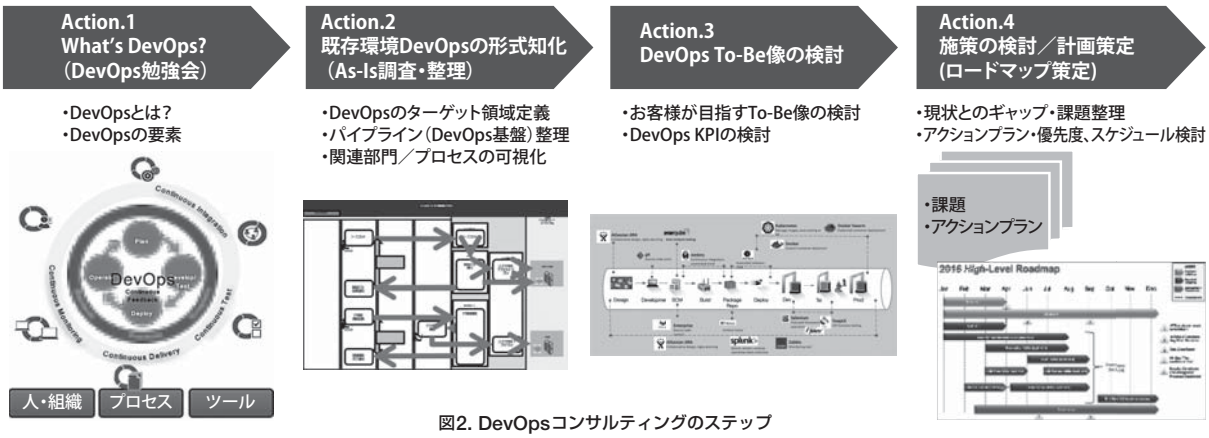

図2. DevOpsコンサルティングのステップ

動化に加え、開発と運用に関わるメンバーの意識改革や組織構造の変革といったDevOpsの推進にも取り組むことがより重要となります。本章では、DevOpsを実現する際に検討すべき代表的な課題について説明します。

(1)目指すべき姿の明確化

DevOpsの実現には、開発や運用のプロセスを定義して必要な作業を極力自動化する必要がありますが、自動化以外の取り組みも必要です。DevOpsの特徴は、ビジネス要件の変化にシステムが迅速に対応できることを目的として、開発担当者と運用担当者が連携して継続的にシステム開発を行うことです。そのため、開発と運用のチーム構成の見直しや、DevOps実現の目的を共有し、DevOpsの必要性について組織全体で共通認識を持つことも重要な成功要因となります。

また、開発や運用のプロセス定義や作業自動化の仕組みを構築するにあたり、現状プロセスの課題やボトルネックを特定した上で改善方法を検討していく必要もあります。さらに、開発対象となるシステム構成についても考慮が必要です。例えば、クラウドでのシステム開発を想定した場合、単一のクラウド・ベンダーのみを利用するシステムを前提とする場合は、ベンダーが提供するサービスを最大限活用することが効率的です。一方、システムごとにクラウド・ベンダーを使い分けるマルチクラウド構成を前提とする場合は、開発と運用のプロセスや自動化の方法をクラウド・ベンダーごとに標準化するのか、クラウド・ベンダーをまたがって標準化するのかといった方針についても検討する必要があります。

DevOpsの実現には、上記に挙げたような点を検討した上で、目指すべき開発や運用の姿を明確にし、実現に向けたロードマップを描くことが必要です。

(2) DevOpsの設計と実装に必要なスキルを持った要員の確保

DevOpsの実現に向けたロードマップの作成後は、開発と運用のプロセス定義や作業自動化のための設計と実装を進めます。設計にあたっては、OSSツールやクラウド・ベンダーの提供するサービスを組み合わせて、開発作業や運用作業の自動化を具体的にどう実現するかを検討します。OSSツールは種類が非常に多く、機能的に類似したものも数多く存在します。また、クラウド・ベンダーが提供するCI/CDやモニタリングのためのサービスもベンダーごとに固有なため、ツールやサービスの選定には多くの調査や知識が必要です。ツールの選定では、前提としているシステム構成がハイブリッド・クラウドやマルチクラウドの場合は、オンプレミスとクラウド間あるいは複数のクラウド・ベンダー間でどこまで共通的に利用できるか、できない場合はどのような代替ツールがあるかなどの検討が必要です。

このように、DevOpsの設計と実装にはOSSツールやクラウド・ベンダーのサービスに関する広範なスキルを持った要員が必要ですが、実際のシステム開発プロジェクトではコスト制約等の理由で専任のメンバーを抱えられない場合も多く、開発と運用の作業自動化が実現できない状況も少なくありません。

(3) 複数環境併用時の運用負荷軽減

クラウドを利用したシステム開発が広がりをみせ、ハイブリッド・クラウド構成やマルチクラウド構成をとる企業が増えています。これらの構成におけるシステム運用は、従来のオンプレミス環境のみでの運用に比べ運用負荷が高くなります。例えば、監視対象となるインフラ・リ

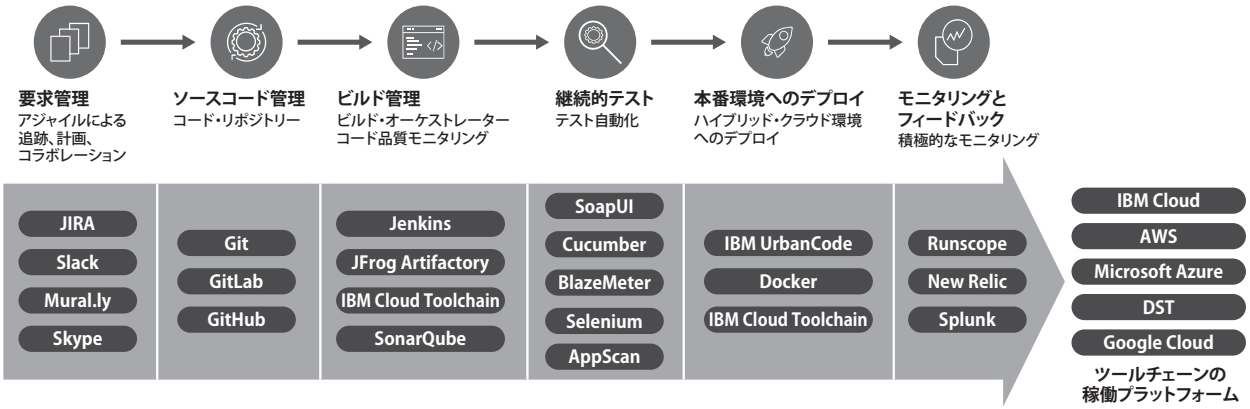

図3. DevOpsのための開発ツールチェーン

ソースやアプリケーションがオンプレミス環境と複数のクラウド環境に分散するうえに、監視方法も各環境で統一できない場合もあるため、複数の監視の仕組みで運用する必要があります。ログの収集と監視についても同様です。また、障害対応についても、特に複数クラウド・ベンダーを利用している場合は原因調査の方法や問い合わせ先、対応実施方法が異なるため運用負荷は高くなります。

ハイブリッド・クラウド構成やマルチクラウド構成でのシステム運用の負荷を抑えるためには、運用作業の自動化や監視状況の可視化および分析を統合的に実施する体制や仕組みを構築することが望ましいのですが、必要なスキルを持った要員の確保や、統合運用のための体制や仕組みを整えるにはそれなりのコストも必要となるため構築が難しい場合も少なくありません。

6. IBMが提供するDevOps実現ソリューション

前章で説明したDevOps実現時の課題に対し、IBMでは「AppOps on Cloud」というソリューションを提供することでお客様のDevOps実現を支援しています。AppOps on Cloudは、クラウドとオンプレミスの両方を対象としたDevOpsの構想策定・開発・運用を支援するソリューションで、大企業を中心に世界中で5,000を超えるプロジェクトへの提供実績があります。以降では、AppOps on Cloudが提供する3つのサービスについて説明します。

(1)構想策定サービス

構想策定サービスは、DevOps実現に向けた戦略コンサルティングを実施します。**図2**に示すステップで、お客様環境でのアプリケーション開発〜運用プロセスの現状を調査・整理し、To-Beモデルの検討を通じてお客様のDevOps実現に向けた施策と実行計画を含むロードマップを作成します。

また、開発と運用を含むエンドツーエンドでのDevOps環境設計や、マルチクラウド構成を前提とした運用アーキテクチャーのハイレベルなアーキテクチャー設計も行います。設計にあたっては、グローバルの先進事例やベストプラクティスを活用し、お客様のTo-Beモデルに応じてOSSツールやクラウド・ベンダー提供のサービスを選定します。

システム構成および開発担当者や運用担当者のスキルは企業によりさまざまですが、自社の状況に応じたロードマップを作成することで、DevOpsの実践に向けた取り組みをスムーズにスタートさせられるようになります。

(2)開発基盤構築サービス

開発基盤構築サービスは、DevOps実現のための開発ツールチェーン(**図3**)とアプリケーションの 開発・テスト環境を迅速かつ低コストで提供します。開発ツールチェーンの構築には、「IBM DevOps Commander」というAPIでアクセス可能なDevOpsプラットフォームの設計、構築、運用を自動化するサービスを利用し、DevOpsのベストプラクティスに基づいた環境を高速に構築、セットアップすることが可能です。また、構築した開発ツールチェーンで利用しているOSSツールの技術サポートやガイドの提供も可能です。構築する開発ツールチェーンには、アプリケーションの静的テストと動的テストを組み込むことで、セキュアなアプリケーションを開発するための仕組みを開発サイクルに組み込むことができます。

要員不足により構築作業が後回しになりがちなCI/CD

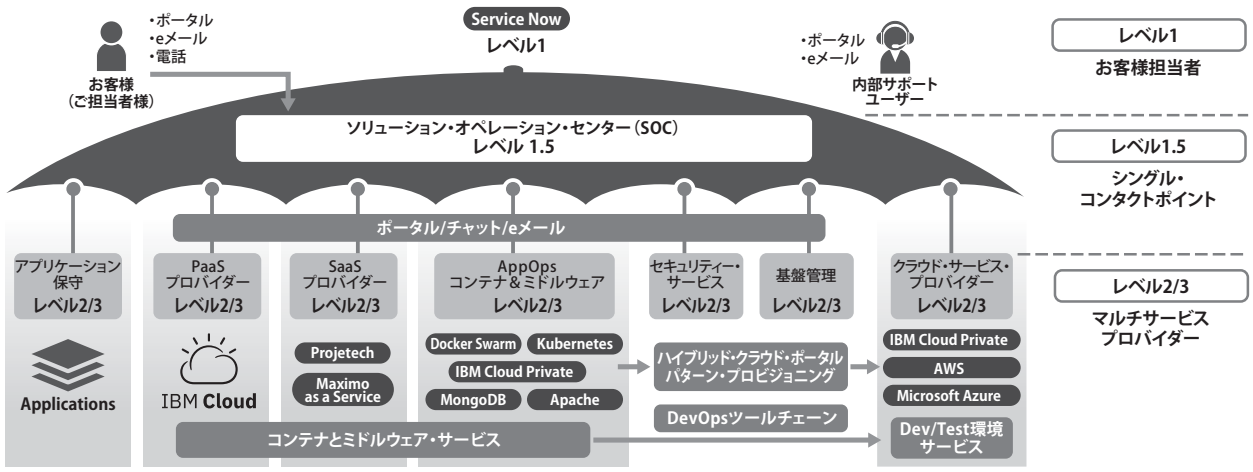

図4. ソリューション・オペレーション・センターの概要

パイプラインを開発の初期段階から利用できるようになるため、開発やテストの効率化とアプリケーション品質の向上を図ることができます。開発ツールチェーンの利用にあたり、技術サポートや各種OSSツールの利用ガイドも提供可能なため、利用経験のない開発メンバーでもすぐに使い始めることができます。

(3)運用・保守サービス

運用保守サービスは、ソリューション・オペレーション・センター(以下、SOC)によるクラウド・アプリケーションの統合運用・保守サービスを提供します。SOCは、レベル1.5としてのシングル・コンタクトポイント、レベル2/3としてのOSSツールやコンテナを含むミドルウェア・マネージド・サービスなど、マルチクラウド・ベンダー環境での統合運用を事前合意のSLAに基づいてシームレスに提供します(**図4**)。また、継続的な自動化と運用改善を実現するため、先進の統合運用管理ツール(セルフサービスポータル、プロアクティブなモニタリング、アナリティクス、ランブックオートメーション、自動レポーティング)を組み込んでいます(**図5**)。

SOCを利用することで、ハイブリッド構成やマルチクラウド構成のシステムに対する統合運用を低コストで実現することができます。さらに、DevOpsの実践において重要となる継続的な適用と改善のサイクルを回すことも可能となります。

7. おわりに

DX時代のビジネスでは、テクノロジーの力を利用して新しい価値をスピーディーに生み出すことが求められており、その手段の一つとしてALMやCI/CDの構築、DevOpsは必要不可欠なものとなりつつあります。本稿で紹介したOSSツールやSIベンダー各社が提供するソリューションを活用してそれらを実現していくことが、差別化された企業価値を生むための重要な要素であると考えています。

[参考文献]
[1] Jenkins: https://jenkins.io/
[2] SonarQube: https://www.sonarqube.org/
[3] Selenium: https://selenium.dev/
[4] Concourse CI: https://concourse-ci.org/
[5] Jenkins X: https://jenkins-x.io/
[6] Tekton: https://tekton.dev/
[7] Gitlab CI/CD: https://about.gitlab.com/stages-devops-lifecycle/continuous-integration/
[8] CDF: https://cd.foundation/
[9] Spinnaker: https://www.spinnaker.io/

日本アイ・ビー・エム株式会社
グローバル・ビジネス・サービス事業 クラウド・アプリケーション・イノベーション部
アプリケーション・アーキテクト

土井 聡
Doi Satoshi

金融系SIerを経て2019年日本IBMに入社。アプリケーション基盤技術がメインスキル。エンタープライズ企業の技術革新を手助けしたいという想いで複数の業種・業界でアプリケーションのモダナイゼーションの推進を実施中。

日本アイ・ビー・エム株式会社
グローバル・ビジネス・サービス事業 アドバンスド・ソフトウェアアセット クラウド・アプリケーション技術 担当
アドバイザリー・アーキテクト

八尾 智幸
Yao Tomoyuki

2004年日本IBM入社。ソフトウェアエ開発研究所でワークフロー製品の開発に従事。グローバル・ビジネス・サービス事業へ異動後は、複数の業種・業界でWebアプリケーション開発案件を中心としたシステム構築に従事。現在は、アプリケーションの高速開発ツールやDevOpsソリューションの展開と推進を実施。

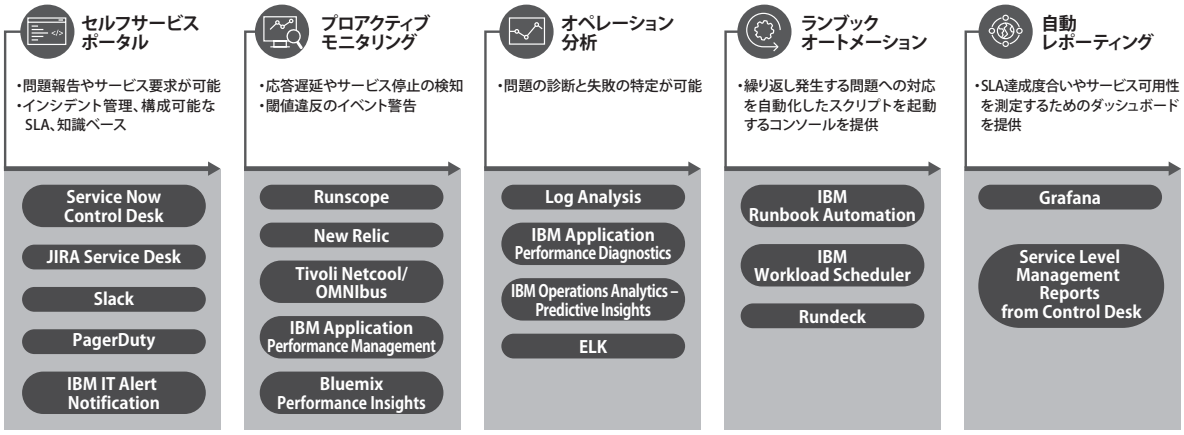

図5. SOCの統合運用管理ツール