アジャイル統合アーキテクチャー

コンテナベースでマイクロサービスと連携した軽量統合ランタイム

目次

デジタル・トランスフォーメーションとアジャイル手法を中心としたアジャイル統合を採用して、マルチクラウド、非集中化、マイクロサービスの要求をすばやく満たす能力をもたらします。

エグゼクティブ・サマリー

デジタル・トランスフォーメーションを追求する組織は統合テクノロジーの利用、展開に新たな方法を採用しなければなりません。そうすることで、マルチクラウド、非集中化、マイクロサービスという目的に向かって適切なやり方ですばやく動くことができます。アプリケーション統合層は組織が新たなカスタマー・エクスペリエンス構築に向かって大胆な動きをとれるように変革しなければなりません。組織の生産性を最大化する方向からそらすようなアークテクチャーと開発のモデル化を強制してはいけません。

多くの組織がマイクロサービス・アーキテクチャーなどのアジャイル・アプリケーション手法を採用し始めています。そして現在はその動きの成果を確認し始めています。この手法は企業のAPI戦略を補完し、加速するものです。企業はこの手法を利用して、既存のESBインフラストラクチャーを近代化して、自社のプライベート、パブリック・クラウドの統合サービスを管理、運用するより効率的な方法を達成する手法を探さなければいけません。

このホワイトペーパーはある書籍に由来するものです。**それはデジタル・トランスフォーメーションに必須のアジャイル、スケーラビリティ、レジリエンスという要求を満たす統合ソリューションが、**コンテナベースで、非集中化された、マイクロサービスからなるアジャイル統合アーキテクチャーと提携する手法のメリットを探求した書籍です。

統合は変化しました

IDCの試算では、デジタル・トランスフォーメーション活動への支出は今後5年間で20兆ドルになります[1]。どうして支出がこんなに爆発的に増えるのでしょうか?あらゆる種類のデータを活用したアプリケーション・ネットワーク全体でつながったエクスペリエンスを通じて、新たなカスタマー・エクスペリエンスを構築するニーズはずっと存在していましたし、どんどん大きくなってきています。

簡単なタスクではありません – 適切なときに適切なコンテキストでプロセスと情報ソースを一緒にするのはうまくやれたとしてはかなり困難です。とくにSaaS業務アプリケーションを積極的に採用しようと考えているときには。新たなデータソースを業務プロセスに投入して、競合との差別化を図らなければなりません。

「新しいカスタマー・エクスペリエンスを推進するために、組織は成長を続けるアプリケーション、プロセス、情報ソース群を活用しなければなりません – どれも企業の統合機能のニーズ、それに対する投資を大幅に拡大します。」

デジタル変革に対するアプリケーション統合の価値

新しいカスタマー・エクスペリエンス構築の予定表を考えるとき、データにどうアクセスして、これらの活動の原動力となるサービスやAPIが利用できるようにするのかに集中すると、アプリケーション統合がもたらす重要な利点がわかります:

- 不均衡への効果的な対応–あらゆるフォーマットのあらゆるシステムからデータにアクセスして、マルチクラウド環境がどんな風に多様化していってもそれを等質化します。
- エンドポイントの専門知識 – 近代的な統合には、複雑なプロトコルやデータ・フォーマットに関する自動化が含まれていますが、エンド・システム内の実際のオブジェクト、業務、機能などに関する情報も組み込まれています。
- データによるイノベーション – アプリケーションのイノベーションは境界の外側のデータを組み合わせて、そこに意味を見つけられるかどうか次第です。これはマイクロサービス・アーキテクチャーではとくに顕著な特徴です。
- エンタープライズ級のアーティファクト – 統合フローは膨大な量のバリューをランタイムから引き継ぎます。これにはエンタープライズ級の機能、エラー回復、フォルト・トレランス、ログ捕捉、パフォーマンス分析、その他が含まれています。

統合の分野は企業や市場のコンピューティング需要に合わせて変化していますが、SOAとESBからどうやって現代的な、コンテナ化された、アジャイル統合アーキテクチャーに到達したのでしょうか?

これまでの経過 – SOA、ESBとAPI

アジャイル統合の未来を眺める前に、まずは過去がどうだったのか理解する必要があります。SOA（サービス指向アーキテクチャー）パターンは今世紀の初頭に出現しました。当初標準SOAが広く受け入れられたのは、すべてのシステムが同期公開パターンによって他のあらゆるシステムを発見し、会話できる輝かしい未来の先駆けとして構築されていたからです。

少し先に進むとESB（エンタープライズ・サービス・バス）運動のまっただなかに達します – このテクノロジーはバックエンド・システムへの接続を提供するはずのもので、先行のハブ・アンド・スポーク・パターンから生じたものです。多くの企業がESBパターンの実装に成功しましたが、この用語がクラウド・ネイティブの空間から愛されることはありませんでした。重すぎて、機動力に欠けていたのです。では、極端から極端へどうやって移ったのでしょうか?

煎じ詰めると真実はほんのいくつかの、相互に関連することの多い要素になります:

- SOAは単なるESB、とくに全社プログラムへの投資の実装よりは複雑なものです。
- ESBパターンは本番サーバー・クラスター上に数十、ないしは数百の統合をインストールした、全社向けの単一インフラストラクチャーを形成します。ESBパターンでは過度の集中化は必須ではありませんが、結果的にはトポロジーはほとんど常にその犠牲となります。

[1]IDC MaturityScape Benchmark：Digital Transformation Worldwide, 2017, Shawn Fitzgerald. Golluscio.

- 集中ESBパターンではたいてい、企業が求めていた大幅削減に失敗します。インターフェイスはあるプロジェクトから他のプロジェクトに再利用できないのです。
- ESBなどの企業間活動は資金調達に苦労しますが、たいていはその資金は制作コストをまかなえるだけ再利用可能なサービスにしか適用されません。

ESBパターンには業務活動のコンテキスト内での具体性がないので、エンタープライズ横断活動に継続的な資金を保証する点に問題があります。

結果として、この専門家SOAチームによるサービスの作成が本来意図していたプロジェクトの支え手ではなく、ボトルネックになってしまいました。これによって集中ESBパターンに結合という誤った名前を与えてしまいます。

ESBパターンに適用されるサービス指向アーキテクチャーはエンタープライズ全体の活動で、再利用可能で、同時利用可能なサービス、APIを作成するものです。これによって新アプリケーションは他システムからのデータを組み込めるようによりすばやく作成できます。

これに対してマイクロサービス・アーキテクチャーは個々のアプリケーションを書く際に、そのアプリケーションの機動力、スケーラビリティ、レジリエンスを高めるために選択できるオプションです。

アジャイル統合アーキテクチャーの事例

どうしてマイクロサービスという概念がアプリケーションの世界でこんなにも普及したのでしょうか?これは構造化アプリケーションに対する代替手法です。同一サーバーで実行される大規模なコードのサイロと化したアプリケーションではなく、より小さな、完全に独立して実行されるコンポーネントの集合体としてアプリケーションは設計されます。

マイクロサービス・アーキテクチャーでは以下の3つの大きな利点が可能です:

1. 機動力の**向上** – マイクロサービスは十分に小さいので、それだけで完全に理解できますし、個別に変更できます。
2. 弾力のある**スケーラビリティ** – リソース利用はビジネス・モデルに完全に結びつけられます。
3. **独立レジリエンス** – 適切な切り離しによって、一つのマイクロサービスに対する変更は実行時に他に影響を与えません。

こうした利点を念頭に、集中化されたサイロに展開されていることの多い統合をマイクロサービス・アーキテクチャーをベースにした視点で再創造するとしたらどんなものになるのでしょうか。これが **「アジャイル統合アーキテクチャー」です。**

アジャイル統合アーキテクチャーは「統合ソリューション向けのコンテナベースの、非集中化された、マイクロサービスと親和性の高いアーキテクチャー」と定義されます。

アジャイル統合アーキテクチャーには3つの関連しているが別々の側面があります:

側面1:微粒子化された統合展開。
サイロ化されたESBを個別のランタイムに分解する統合で何が得られるのでしょうか?

側面 2:非集中化された統合所有権。
微粒子化手法をより活用できるように、組織構造をどう変更すべきでしょうか?

側面3:クラウドネイティブ統合インフラストラクチャー。
統合の完全クラウドネイティブ手法にさらにどんな利点があるのでしょうか。

側面1:
微粒子化された統合展開

統合ハブ、またはESBパターンの集中化された展開ではすべての統合はしっかりと世話された(HA)単一の統合サーバー・ペアに展開されますが、これがプロジェクトにボトルネックをもたらすことが明らかになっています。共有サーバーへの展開には既存の重要インターフェイスを不安定化させるリスクがあります。個々のプロジェクトで新機能にアクセスするために統合ミドルウェアのバージョンをアップグレードすることは選択できません。

エンタープライズ全体のESBコンポーネントはもっと小さく、もっと管理しやすい、特定用途のピースに分割できます。場合によっては、公開するインターフェイスそれぞれに対してランタイム１つにまでばらせるかもしれません。こうした「微粒子化された統合展開」パターンは専用の、適切な規模のコンテナを提供して、機動力、スケーラビリティ、レジリエンスを改善するので、これまでの集中ESBパターンとはまるで別のものに見えます。図1に示したのは単純な用語で、集中ESBと微粒子化された統合展開の違いを示しています。

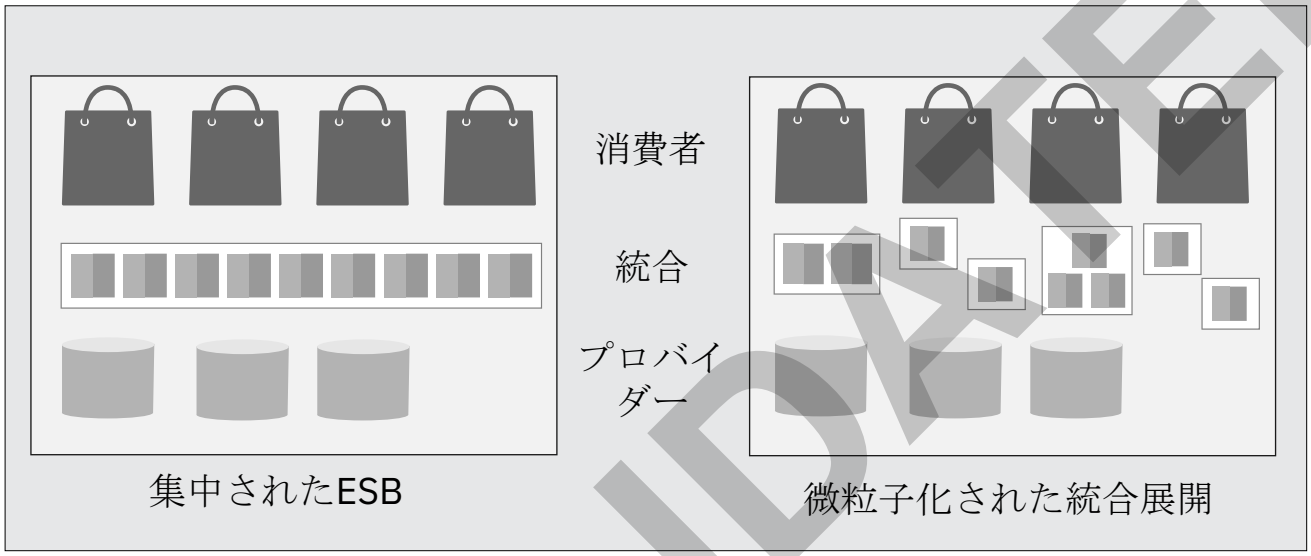

図１:集中ESBと微粒子化した統合展開の単純化した比較

微粒子化した統合展開はマイクロサービス・アーキテクチャーの利点を活用しています。微粒子化した統合展開に照らして、マイクロサービスの利点の一覧を振り返ってみましょう:

- **機動力** – すぐにボトルネックになる集中グループやインフラストラクチャーを後回しにすることなく、別々のチームが独立して統合作業を行えます。個々の統合フローは他のフローとは独立して変更、リビルド、展開できるので、アプリケーション変更はより安全なものになり、稼働までの速度を最大化できます。
- **スケーラビリティ** – 個々のフローはそれだけでスケールアップできるので、クラウド・インフラストラクチャーの効果的なエラスティック・スケーリングを活用できます。
- **レジリエンス** – 独立した統合フローは個々のコンテナで展開されるので、メモリー、接続、CPUなどの共有リソースを独り占めして、相手に影響を与えることがありません。

機動力、スケーラビリティ、レジリエンスを考慮すると、非集中化した統合なしに微粒子化した統合の利点を生かすことはできません。

微粒子化した統合について当社のアジャイル・インフラストラクチャー・アーキテクチャーをご覧ください。今すぐダウンロードを!

側面2:非集中化された統合所有権

サービス指向アーキテクチャーの直面する大きな課題は、集中統合チームの設定、サービスを造るためのインフラストラクチャーが強制される傾向にある点です。

これは継続的な摩擦をプロジェクト実施速度に対して生み出します。集中統合チームに依存するために。チームは自分たちの統合テクノロジーをよく知っていますが、統合対象のアプリケーションについては理解していないことが多いので、要件の翻訳が遅く、エラーを起こしがちです。

多くの組織はアプリケーション・チームが自分のサービス作成を支配できるのを好んだでしょう。しかし当時のテクノロジーとインフラストラクチャーでは不可能でした。

微粒子化した統合展開への動きが、統合の創造と保守の所有権分配への扉を開きました。業務アプリケーション・チームが統合作業を行って、新規機能の実装を合理化することは非現実的ではありません。

微粒子化された統合展開に興味はありませんか?当社のアジャイル統合アーキテクチャー・ブックが質問にお答えします。

側面3:クラウドネイティブ統合インフラストラクチャー

統合ランタイムは近年劇的に変化しました。これらの軽量ランタイムは真にクラウドネイティブな方法で利用できます。ここで言及している能力によって、クラスター管理、スケーリング、可用性などのこれまでプロプライエタリな機構だったものの重荷の多くを手放して、実行しているクラウド・プラットフォームに渡します。

これはコンテナ化された環境で実行するだけでないさまざまなものを必要とします。つまり「ペットではなく家畜」として機能すべきだ、ということで、Kubernetesやその他の一般的なクラウド標準フレームワークのオーケストレーション機能を最大限に活用すべきなのです。

「家畜手法」の採用はDevOpsチームの環境やソリューション全体との対話に影響して、軽量アーチテクチャーにアプリケーションを移行するほど効率が向上します。

近代的な統合ランタイムはアジャイル統合アーキテクチャーにどうやって対応したのでしょうか?

明らかなことですが、アジャイル統合アーキテクチャーではまったく別の形で統合トポロジーを展開する必要があります。その重要な側面として、コンテナベースの環境で動作できる近代的な統合ランタイムはクラウドネイティブな展開技法に最適です。近代的な統合ランタイムは従来のものからはほとんど認識できません。こうした違いのいくつかについて触れましょう:

- **高速な軽量ランタイム:**Dockerなどのコンテナで動作して、十分に軽量なので、数秒で開始、終了できますし、 Kubernetesなどのオーケストレーション・フレームワークで容易に管理できます。
- **依存性がない:**データベースやメッセージ・キューはもう必要ありません。とはいえ明白なことですが、必要があれば、それらに対する接続にも充分な能力があります。
- **ファイル・システム・ベースのインストール:**インストールは単純でバイナリーをファイル・システムに配置して、起動するだけです–Dockerイメージの多層ファイル・システムに最適です。
- **DevOpsツールのサポート:**ランタイムは継続的な統合、展開対応でなければいけません。スクリプトとプロパティ・ファイルベースのインストール、ビルド、展開、構成で「インフラストラクチャー・アズ・コード」手法に対応します。標準ビルド、展開ツール用のテンプレート・スクリプトを提供して、DevOpsパイプラインへの包含を加速すべきです。
- **API–ファースト:主要な通信プロトコルは**RESTful APIにすべきです。RESTful APIとしての統合公開は明確なもの、Open API仕様などの共通慣行に基づくものでなければなりません。ダウンストリームRESTful APIの呼び出しも定義ファイルによるディスカバリーを含む明確なものでなければなりません。
- **デジタル接続:**統合ランタイムが常に提供してきた豊富なエンタープライズ接続に加えて、近代的なリソースにも接続できなければなりません。たとえば、NoSQLデータベース(MongoDb、Cloudantなど)や、Kafkaなどのメッセージング・サービス。さらには、SalesforceなどのSaaS（ソフトウェア・アズ・ア・サービス)アプリケーションとのアプリケーション・インテリジェント・コネクタの豊富なカタログにもアクセスする必要があります。

- **継続的な展開:**継続的な展開は標準DevOpsパイプライン・ツールに組み込まれたコマンドライン・インターフェイスとテンプレート・スクリプトによって可能になります。これによってインターフェイス実装に必要な知識をさらに減らし、展開のペースを上げられます。
- **強化されたツール:**統合用の強化されたツールはほとんどのインターフェイスが構成するだけで、統合に関する背景知識を持たない個人であってもビルトできることを意味します。共通統合パターン用のテンプレートの追加によって、統合のベストプラクティスはツールに焼き付けられているので、さらにタスクを単純化できます。深い知識を持つ統合専門家が必要になることが少なくなり、統合の一部は非集中化統合に関する次のセクションで触れるように、アプリケーション・チームが実施できる可能性があります。

近代的な統合ランタイムはアジャイル統合アーキテクチャーの次の3つの側面にうまく適合します:微粒子化された展開、非集中化された所有権、真にクラウド・ネイティブなインフラストラクチャー。

クラウドネイティブなインフラストラクチャーについてさらに詳しくお知りになりたいですか?
当社のアジャイル統合アーキテクチャー・ブックをダウンロードしてください!

統合プラットフォーム向けのアジャイル統合アーキテクチャー

このホワイトペーパー全体で、アジャイル統合アーキテクチャーで展開されるアプリケーション統合機能に焦点を当てます。しかしながら、多くのエンタープライズ・ソリューションはいくつかの重要な統合機能を適用することでのみ解決できます。統合プラットフォーム(または「ハイブリッド統合プラットフォーム」と一部のアナリストが呼ぶもの)はこれらの機能を一緒に提供するので、組織は業務ソリューションをより効率的に、一貫性を保った形で構築できます。

多くの業界専門家はこの統合プラットフォームの価値について同意しています。Gartnerの注記:

ハイブリッド統合プラットフォーム(HIP)はオンプレミスとクラウドベースの統合フレームワーク兼ガバナンス機能で、これによって、 異なるスキルを持つ人物(統合専門家と非専門家)が広範な統合ユースケースをサポートできます…統合に責任を持つアプリケーション・リーダーはHIP機能フレームワークを活用して、統合戦略とインフラストラクチャーを近代化すべきです。そうすればデジタル・ビジネスの新たなユースケースに対応できます[2]。

Gartnerノートの重要なものの一つは、統合プラットフォームによってさまざまな組織からやってきた人々が自らのニーズにもっとも適合したユーザー・エクスペリエンスで協力できることです。つまり、業務ユーザーが簡単な問題を解決する指針となるより単純なエクスペリエンスで生産性を向上する間に、IT専門家は専門家レベルの制御でより複雑なエンタープライズ・シナリオを取り扱えます。その後、これらのユーザーが共有した資産の再利用で協力しながら、全体のガバナンスを維持します。

デジタル・トランスフォーメーションの新興のユースケースを満足させることは、さまざまなユーザー・コミュニティのサポートと同じくらいに重要です。このホワイトペーパーの大部分はこれらの新興ユースケースを探索します。しかしまずは統合プラットフォームの一部でなければならない重要機能について詳述します。

IBM Cloud Integration Platform

IBM Cloud™ Integrationは重要な統合機能群を、単純で、高速で、信頼できる首尾一貫したプラットフォームにまとめます。強力な統合、APIを数分で容易にビルドできて、先進のパフォーマンスとスケーラビリティを提供し、エンタープライズ級のセキュリティを備えた他に例のないエンドツーエンドの機能をもたらします。

IBM Cloud Integrationプラットフォームには、6つの、それぞれベストオブブリード機能の重要な統合上の特徴を組み合わせました。それらは以下の通りです:

API管理:
再利用可能なAPIとしてビジネス・サービスを組織内外の一部限定の開発者コミュニティに公開、管理します。組織はAPI戦略を採用して、独自データとサービス資産の効率的な共有を加速して、新たなアプリケーションや新たなビジネス・チャンスを活気づけます。

セキュリティ・ゲートウェイ:
エンタープライズ外に拡張された接続性と統合はDMZ対応のエッジ機能を備えていて、API、それが移動しているデータ、その背後にあるシステムを守ります

アプリケーション統合:
オンプレミスやクラウドのアプリケーションとデータ・ソースをつないで、業務情報交換を協調させて、必要なとき、必要な場所でそのデータが利用できるようにします。

[2]Hype Cycle for Application Infrastructure and Integration, 2017, Elizabeth Golluscio.

メッセージング:
システムやネットワークで問題が発生したときにメッセージが失われたり、だぶったり、複雑なリカバリーが必要になったりしない信頼性の高いメッセージ配送を提供して、いつでもどこでもリアルタイム情報の利用を保証します。

データ統合:
データにアクセスし、それをきれいにして、整えて、分析用のデータ・ウェアハウスやデータ・レーク内に業務の一貫したビューを作成します。

高速転送:
オンプレミスとクラウド間、ないしはクラウド相互間で大量データをセキュリティ・レベルを強化してすばやく、予想通りに移動します。データがきわめて大きいと、組織がクラウド・プラットフォームを採用するすばやさが促進されます。

このハイライトだけのホワイトペーパーで、統合プラットフォームの一部として必要なさまざまな重要機能、同時に動作させるためのこれらの機能要件に関する感覚、さらにはアジャイル統合アーキテクチャーがプラットフォームのより大きな機動力、スケーラビリティ、レジリエンスを可能にするため採用できることについて広範な視点を持っていただければ幸いです。

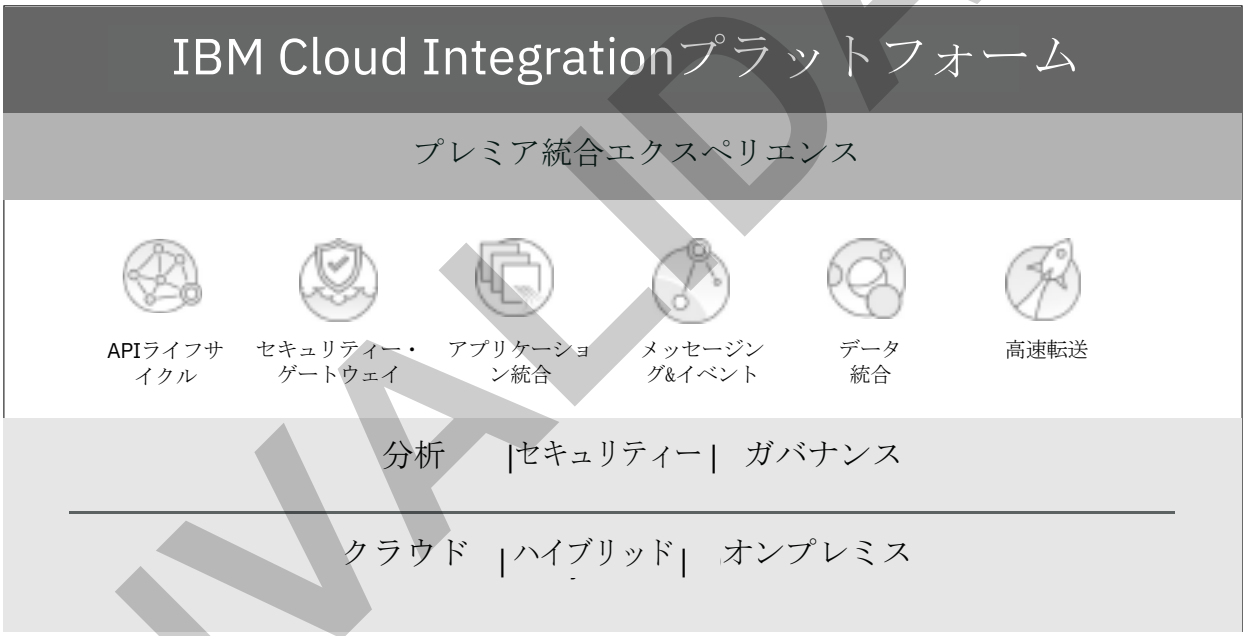

図 2: IBM Cloud Integrationプラットフォーム

わかりやすい電子ブックをダウンロードして、アジャイル統合アーキテクチャーをさらに学んでください。

日本アイ・ビー・エム株式会社
〒103-8510
東京都中央区日本橋箱崎町 19-21

Produced in the United States of America
August 2018

Please Recycle

82018782-JPJA-00