OSSを活用した新しいユーザー体験の実現

「IBM Garage Method」による製品の枠を超えたサービスの提供

さまざまなサービスがオンライン化され、ユーザーのニーズも多様化している現代社会において、ビジネスを拡大するために重要となるのは、「いかにしてユーザーの本質的課題を見つけ、かつその課題に対するサービスを迅速に提供することができるか」ということです。そのためにはOSSやクラウドの活用はもちろんのこと、従来とは異なるアプローチでの課題の洗い出し、設計・開発手法が必要となってきます。

本稿では、OSSやクラウド・プラットフォームを活用し、継続的にイノベーティブなサービスを生み出すためのオープンな手法である「IBM Garage Method」について説明します。

1. はじめに

近年、スマートフォンやクラウドの普及に伴い、ITを取り巻く市場は急速に変化しています。行政や医療を含むさまざまなサービスがオンライン化され、ITはビジネスだけではなく、われわれの普段の生活に必要不可欠な存在となりました。また、ユーザーはサービスを評価し、かつSNSなどを用いて他のユーザーと共有することが簡単に行えるようになりました。その結果、ユーザーのサービスに対する期待値も高まり、ユーザーの望む機能を然るべきタイミングで提供できない企業は淘汰される時代となっています。

このようにユーザーからの多様化する要求に対して、旧来の手法を用いていては激しい市場競争に勝ち残ることは不可能であり、新しい設計・開発手法やプラットフォームが必要とされています。

2. スタートアップ並の機敏さを実現する IBM Garage Method

皆さんはオープンソースという言葉で何を思い浮かべるでしょうか？　多くの方はOSS（オープンソースソフトウェア）を思い浮かべるのではないかと思います。実はソフトウェア以外にも、デザイン・ガイドライン、ハードウェアの設計、方法論やデータ（オープンデータ）など、さまざまなものがオープンソースとして公開されています。

IBMでは、急速に変化する市場に柔軟に対応し、お客様とともにイノベーティブなサービスを創造しビジネスに変革を起こすために、IBM Garage Methodと呼ばれる新しい手法を公開しています（**図1**）。手法の内容や情報を公開し、お客様自身でも広く使用してもらいフィードバックを得ることで発展を促すという意味では、IBM Garage Methodもオープンソースに通じるものがあります。

IBM Garage Methodは、IBMが長年培ってきたエンタープライズ製品開発のノウハウに加え、スタートアップ企業が採用しているような、デザイン・シンキング、アジャイル開発、OSSやクラウド・プラットフォームを

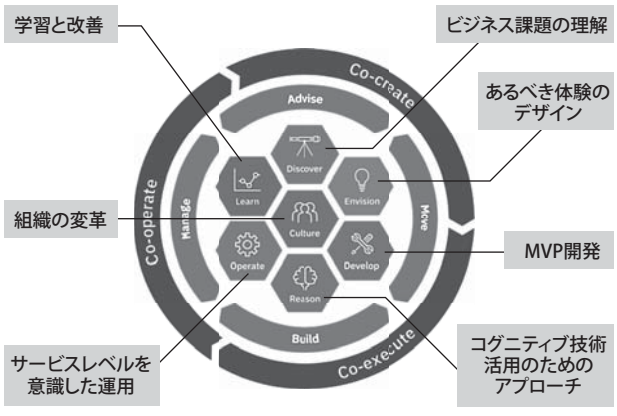

図1. IBM Garage Method概要

ベースにしています。

IBM Garage Methodでは、リーン・スタートアップ [1] の手法をベースとし、「構築」「計測」「学習」のプロセスを短いサイクルで繰り返し実施することでサービスの実装・改善を行います。その際、闇雲にトライ&エラーを繰り返すのではなく、デザイン・シンキングで得られたユーザー体験の改善という明確なビジョンを持ち、然るべきアーキテクチャーで、効率的な開発手法を用いることが求められます。

本稿では「デザイン・シンキング」「アーキテクチャー」「開発手法」の視点からIBM Garage Methodを解説します。その全容に関してはフィールドガイド[2]を参照ください。

3. オープン化により普及・発展するデザイン・シンキング

製品やサービスの企画・設計段階において、今やデザイン・シンキングの重要性は広く認知されています。IBM Garage（以下、ガレージ）が実践する“顧客との共創”と“かつてないスピード感”にも、デザイン・シンキングが重要な役割を果たしています。本章では、IBMがデザイン・シンキングを重要な施策の一つとして普及・発展させるために実施している、オープン化について解説します。

その前に、デザイン・シンキングについてよくある誤解を1点だけ解いておきます。デザイン・シンキングはブレインストーミングのためのワークショップではありません。エンドユーザーのより良い体験をゴールとする取り組み全般を指します。その取り組みを体系化し、短期集中型で成果を出すために実践されているのがデザイン・シンキングのワークショップです。本来“デザイン・シンキングを実践する”とは、ユーザーのためのアイデアを形にし、検証して、改善点を見つけ、また形にし、検証する。このループを自律的に回せている状態を指します。

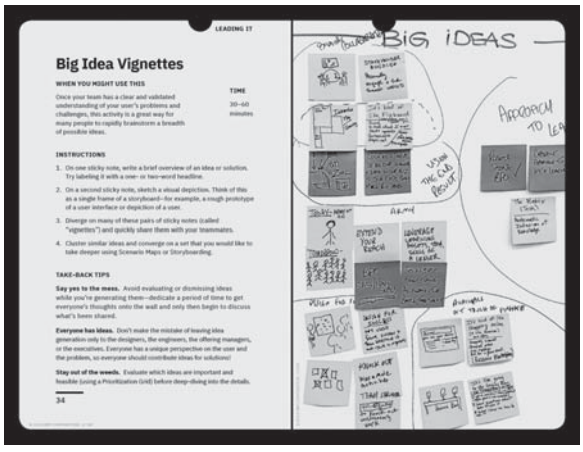

図2. Enterprise Design Thinking Field Guide

IBMではこの終わりなき改善のループを、より企業のビジネス・プロセスやプロジェクト・チームに適用しやすくするために、「エンタープライズ・デザイン・シンキング」（以下、EDT）として体系化し、アクティビティーのツール群やTipsを公開しています。どなたでもIBMの公開サイトからEDTの理念を閲覧し、ツールをダウンロードして自由に利用できます（無料でIBMidを作成しログインする必要があります）[3]。また、「フィールド・ガイド」というEDT活動をリードする人向けの小冊子[4]（**図2**）も発行しており、サイトからPDF版が入手可能です。

しかし、道具を手に入れ教科書を読めば実践できるかというと、そこはオープン化されたIT技術同様に難しい局面も多いものです。そこで、「IBMデジタル・バッジ・プログラム」というオープンなスキル認定制度を利用して実践的な学習が行えます[5]。

このプログラムでは、EDTのスキル・レベルを初期の「Practitioner（実務者）」から「Co-Creator（共創者）」「Coach」「Leader」へと段階的に認定しています。「Practitioner」ではEDTの理念とプロジェクトの進行を理解し、独自の共通言語を駆使してプロジェクトに貢献できることが目標で、e-ラーニングとクイズ形式の理解度チェックで比較的簡単に取得可能です。次の「Co-Creator」は実際にチームで取り組んだEDTプロジェクトの成果を段階ごとにWebフォームに記述するタフな認定ですが、このバッジ取得者の理解度・習熟度は信頼に足るものがあり、社内の研修部門でもCo-Creatorの増加はIBMビジネスの向上に効果的と見て育成に力を入れています。これらは社外の方にも取得いただけるオープンなバッジです（一部国内準備中）。

さらに、SNSツールを媒体として世界中のEDT実践者が情報交換を行っています。EDTプロジェクトで得た知見を基にした質疑応答や、ワークショップのファシリテーターの急募にも利用されています。一方向的な教育だけではなく、リアルな業務を通じたコミュニケーションがEDTの質を高め、仲間を増やし、IBMビジネス全体の価値を向上しています。

このように、EDTはアセットのオープン化、スキル認定のオープン化、専門家コミュニティーによって、オープン化されたI T技術同様に普及と発展を続けています。

4. 迅速なサービス立ち上げ、継続的なリリースを可能にするアーキテクチャー

デザイン・シンキング・ワークショップで導き出されたアイデアは可能な限り早く実装し、検証する必要があります。新しいサービスが本当にユーザーに受け入れられるかを一番早く正確に知る方法は、MVP（Minimum Viable Product、実用最小限の製品）を実装し実際にユーザーに使ってもらうことです。

そのためにはサービスを迅速に開発し、かつ継続的なリリースを効率的に行えるプラットフォームが必要となり、OSSや「Red Hat OpenShift」（以下、OpenShift）などのクラウド・プラットフォームの活用は必須条件です。**図3**にIBM Garage Methodで採用されているソフトウェア・コンポーネントの例を示します。

また、こららの技術を組み合わせ、どのようなアーキテクチャーでプラットフォームを構築するかも非常に重要となります。「IBM Cloud Architecture Center」[6]ではテクノロジーやユースケースごとにアーキテクチャー選択のガイドラインやベストプラクティスが公開されています。

5. ソフトウェアの設計思想の理解と検証の重要性

OSSを使用する上で注意すべき点もあります。ベンダーから提供された有償のパッケージ・ソフトウェアを使用する場合、ドキュメントに書かれた仕様はベンダーが保証してくれました。極端な例ですが、製品の設計思想に反した使い方をして問題が発生したとしても、そのような使い方がドキュメントで制限されていなければ製品の問題として修正してくれるかもしれません。

しかしOSSの場合、現存するランタイムの実装が基準となります。OSSを利用するユーザー側がその設計思想や実装に用いられている技術要素を正しく理解し、正しい使い方をする必要があります。

そのため、最終的なアーキテクチャーを決定する前に実際に動作させ、想定していた使い方が可能かを検証する必要があります。ここでもOpenShiftなどのクラウド・プラットフォームの活用が鍵となります。クラウド・プラットフォームは本番環境に近い環境でさまざまな組み合わせの検証が可能なサンドボックスを簡単に提供してくれます。

OSSやクラウド・プラットフォームを効率的に活用するためには、エンジニアにより高いスキルが求められることになりますが、技術を正しく理解し、最適なアーキテクチャーをデザインすることは中長期的に見ればリスクを減らし、変化に対応するスピード、ひいてはユーザー満足度の向上につながります。

6. MVPの開発

本章では、ガレージで実際に行われているMVP開発の一連の流れとともに、そこで活用されているOSSについて紹介します。開発言語やフレームワークの多くはOSSを用いており、選定の際には要件に応じて柔軟に選択します。GitHubのスター数も一つの指標とすることができます。人気の高いOSSを選択することで、使用中に疑問が生じた際など、Webで検索すると多くの参考情報がヒットする場合が多いため迅速に問題解決できる場合が多くなります。

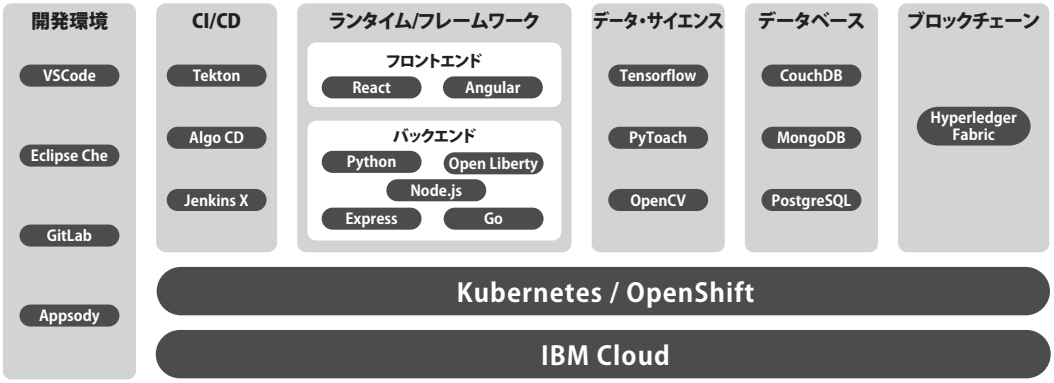

図3. IBM Garage Methodで採用されているソフトウェア・コンポーネント例

6-1. スプリント0

スプリント0は、実開発を円滑に進める準備として1〜2週間かけて実施するものです。ワークショップではまだ人間の体験をベースとした抽象的な物語だったものを、アプリの利用を通して実現させるための具体的な肉付けを行う作業とも言えます。同時に、開発環境のセットアップや技術検証、与えられた全体の開発期間で実現できそうなユーザー体験の範囲（MVPゴール）をより詳細に定義することも行います。ストーリーマップ、システム・コンテキスト、画面遷移図、基本的な画面構成、アーキテクチャー概要、データモデル、プロダクト・バックログなどを作成しますが、時には実際のユーザーやスポンサーユーザーと呼ばれるユーザーの代弁が可能な人物にこれらの計画を評価してもらうインタビューなどを行うこともあります。

このようにして、スプリント0では、技術者やアーキテクトだけではなく、プロダクト・オーナー、UX/UIデザイナー、ユーザーなど引き続き複数のスキルと職責を持ったメンバーとの共創が実践されています。個々のメンバーが与えられた責任を高度に高速に果たすためには、全員とのコミュニケーションと合意形成のプロセスが必須であり、これを実践できることがプロジェクトの成功やチームの能力の鍵であると言っても過言ではありません。ガレージでは**図4**のように、開発者やアーキテクト、デザイナーが協力して付箋とペンでストーリーマップの作成に取り組みます。フィジカルな共創スペースを確保できる場合はこれが最も早く、常に視界に入るのでコミュニケーションも活性化され頻繁な改善に効果を発揮します。

ストーリーマップの作成後にはデベロッパーが主体となりストーリーマップ内に定義された個々のユーザーストーリー（実装を前提としたより細かい粒度で定義された一連のユーザータスク）にかかるワークロードの見積もりを行います。それらをプロダクト・バックログとして整理し、プロダクト・オーナーが優先順位付けし、開発者はスプリント・バックログを定義します。RedmineやGitLab Issuesなどのタスク管理ツールがよく活用されています。

図4. ストーリーマップ作成の様子

デザイナーは、ストーリーマップに表現されたアプリケーションの設計を理解し、それをユーザーに自然に利用させるための画面遷移と各画面の基本構成をまとめます。スピードを重視するため、まずは手描きのスケッチで表現し、早い段階で開発者に説明して技術的な実現性を確認してもらいます。また、簡易なデザインガイドで画面の基本構造の理念を文章で著したり、サイズや比率や基幹色を数値で指定したり、プロトタイピング・ツールを使ってリアルなサンプルを提示することも重要です。具体的な指定により、複数の開発者がそれぞれに異なる解釈をしてしまうリスクを軽減し、人に依存しやすいデザインの再現性をある程度コントロールすることができます。デザイナーと開発者の画面設計に対する理解と優先度の理解が常に一致するよう、適切なコミュニケーションを保つことが重要です。アジャイルの文化において、すべてのメンバーはお互いの成果に意見を言うことを制限されません。デザイナーはそれらの会話を通じてユーザーにとって最も良い画面へと段階的に洗練させる責任を負います。

スプリント0のみならず、すべてのスプリントの最後にはワークショップ同様にプレイバックの機会を持ち、プロダクト・オーナーなど必要なメンバー全員で成果の確認を行います。さらにレトロスペクティブと呼ばれる振り返りも実践し、以降のスプリントで改善すべき行動を発見します。

6-2. スプリント1以降

スプリント1以降は、MVPの開発と、それを用いたユーザー体験の検証の繰り返しを行います。

6-2-1. ペア・プログラミング

IBM Garage Methodではペア・プログラミングによる開発を推奨しています。1台のコンピューターに2台のキーボードを接続し、デベロッパーはペアを作って同じ画面に向き合います。そして、一方が実際にコーディングを行うドライバー、もう一方がそれに対して助言を出すオブザーバーとなり、実装を進めていきます。1つのユーザーストーリーに

対しペアで向き合いコミュニケーションを取りながら実装を進めることで、より迅速に、質の高い結果を得ることができるのです。エディターにはVisual Studio CodeやEclipseを使用し、必要に応じてLive Share拡張機能やWeb会議ツールを用いてリモートで作業します（**図5**）。

6-2-2. テスト駆動開発

コーディングにあたってはテスト駆動開発を採用します。Jest、Mocha/Chaiなどのフレームワークによってテストが自動化された状態で、すべての振る舞いについて、テストを最初に記述し、その後にロジックを実装する形で進めます。テストを記述したら実行し、まず失敗することを確認します。さらにロジックの実装、テスト、リファクタリングを繰り返し、テストをパスする状態になったらその部分の実装は完了です。テストはそのロジックの仕様を記述したものとも言え、コードを高品質に保つために重要な役割を果たします。

6-2-3. バックエンドAPIの実装

API実装の際にはNode.js上のフレームワークであるExpress、LoopBack、Spring Bootなどのフレームワークを用いる場合もあります。また、APIを経由してアクセスするバックエンド・サービスはIBM Cloud上のAI（Watson）サービスやIBM Blockchain Platformなどのほか、Hyperledger FabricやMongoDBなど、多岐にわたります。いずれもコンテナとして稼働し、ポータビリティーの高いマイクロサービス[7]として実装します。

6-2-4. フロントエンドUIの実装

Webアプリケーション向けのUIフレームワークで近年メジャーなものとしては、Angular、React、Vue.jsなどがあります。MongoDB、Express、Node.jsと組み合わせて使う場合が多いことから、「MEAN（MongoDB、Express、Angular、Node.js）スタック」や「MERN（MongoDB、Express、React、Node.js）スタック」と呼ばれることもあります。

図5. ペア・プログラミングの様子

6-2-5. デザイン・ウォークスルー

どんなに良いチームでも、完璧なコラボレーションが継続され常にベストの品質を短期に開発できるわけではありません。デザイナーと開発者においても同様で、ちょっとした認識のずれや会話の不足から、アプリの使い勝手やユーザーの満足度（UI/UX）の品質は、いとも簡単に劣化していきます。チームが意図していなかったアプリがプレイバックのタイミングでデモされてしまうリスクを軽減するために行うのがデザイン・ウォークスルーです。

ある程度動作するところまで実装されたアプリを、デザイナーまたは実装者とは異なる人物がユーザーに代わり、ユーザーストーリーに従って操作してみます。説明や介助なしにタスクを遂行できるか、迷うポイントがないか、実際に使ってみて冗長と思われるシナリオがないか、画面の構造や色彩計画は妥当だったか、デザインガイドどおりに実装され、かつデザイン計画の意図がユーザーのタスクに適切に反映できているか、デザイナーはこれらを判断し、改善点を列挙します。改善点は開発者による技術的難易度も考慮の上、優先度が決定されます。

6-2-6. 実行環境の構築、アプリケーションのデプロイ

MVPの開発、テストが完了し、ユーザーに検証してもらうための実行環境を準備します。近年ではクラウド上のOpenShiftやKubernetesクラスターのマネージド・サービスであるCloud Foundryなどを基盤とするアプリケーション実行環境（PaaS）、サーバーレス環境（FaaS）などを用いることによって、環境構築に時間と手間をかけることなくMVPに実装した機能をユーザーに検証してもらえるようになりました。ユースケースや規模、ビジネス上の要件にあった環境を選択し活用するとよいでしょう。このような基盤を用いることによって管理や運用の仕組みづくりも行いやすくなります。また、MVPを検証する際のA/Bテストやカナリアリリースなどの実施が容易になります。

コンテナでの稼働を前提として作成したマイクロサービスは、TektonやJenkins Xなどのツールを用いてCI/CDパイプラインを構成し、ソースコードからコンテナ・イメージへの変換（S2I）、クラスターへのデプロイを自動的に

行えます。このような効率的かつ迅速なリリースによって、ユーザーのあるべき体験に必要な機能をMVPへ素早く柔軟に反映することが理想的です。

7. 実プロジェクトにおけるOSSの活用

このように、OSSは今やガレージをはじめあらゆるシステム開発において活用されており、多くのメリットをもたらしています。例えば、複数のビジネス・パートナーと協業してプロジェクトを進める際に、IBM製品を含む従来のプロプライエタリー・ソフトウェアを用いていた場合、ビジネス・パートナー側での開発環境構築のために前提ソフトウェアを含めたライセンスの貸し出しなどが必要であり、それなりの手続きや時間を要していました。OSSを使用することでこれらの手続きや時間を大幅に短縮することが可能です。

また、OSSの品質に関しても事前検証が容易であり、前述のように設計思想に則った使い方をすれば大きな問題が生じることはありません。冗長化や性能などの非機能要件に関しても、Kubernetesのレプリケーションやオートヒーリングを使用することでOSS側にその機能がなくても必要な非機能要件の担保が可能です。それだけでなく、OSSは非機能面をクラウド・プラットフォーム側に移譲することを前提としたシンプルな作りであることが多く、それゆえアプリケーション開発を非常に効率的に進めることが可能になります。冒頭で述べたように、ユーザーの期待に迅速に応えるためにガレージやOSSのようなオープンな手法、ソフトウェアの活用はこれからの時代に必要不可欠だと言えるでしょう。

8. おわりに

最後に、日本国内におけるOSSやクラウド活用の動向に関して考察します。冒頭でITサービスが日々の生活に必要不可欠になったと述べましたが、そのほとんどのシェアは米国を中心とした多国籍企業によって提供されているサービスに占められています。

残念ながら日本国内からはそれらに匹敵するほどイノベーティブなサービスをスピーディーに提供できていないのが現状でしょう。われわれの経験として、まだ日本国内にはOSSやクラウドをビジネスに利用することへの抵抗感がある企業が多いと感じています。

確かにOSSやクラウドを採用することに関してリスクがないとは言いきれません。ただ、旧来の開発・設計手法、プラットフォームを踏襲し、魅力的なサービスを迅速に提供できないことは確実なリスクとなります。

イノベーティブなサービスを迅速に、かつ継続的に提供するためには新しい手法や技術、プラットフォームを柔軟に取り込み、そして何より自社のサービスを変えていく決定のスピード、ひいては組織や企業文化の変革が必要であると考えます。

[参考文献]
[1] Eric Ries:The Lean Startup. Crown Publishing. ISBN 978-0-307-88789-4.(2011).
[2] IBM Garage Method for Cloud Field Guide: https://www.ibm.com/cloud/garage/content/field-guide/garage-method-field-guide/
[3] Enterprise Design Thinking Toolkit: https://www.ibm.com/design/thinking/page/toolkit
[4] Enterprise Design Thinking Field Guide : https://www.ibm.com/cloud/architecture/content/field-guide/design-thinking-field-guide
[5] IBM Skills Gateway, Badges: https://www.ibm.com/services/learning/M425350C34234U21
[6] IBM Cloud Architecture Center: https://www.ibm.com/cloud/architecture
[7] I. Nadareishvili : Microservices Architecture – Aligning Principles, Practices and Culture, O'Reilly, ISBN 978-1-491-95625-0. (2016)

日本アイ・ビー・エム株式会社
IBM Garage
アーキテクト

古郷 誠
Kogoh Makoto

IBM入社以来、業務用アプリケーション基盤やHPC、ブロックチェーンなど、さまざまな開発プロジェクトに従事。全人類がプログラミングを学習すれば世界は平和になると信じている、酒と猫を愛するエンジニア。

日本アイ・ビー・エム株式会社
IBM Garage
ガレージ・デザイナー

藤枝 久美子
Fujieda Kumiko

IBM製品のデザイン業務を通じユーザー中心設計を学んだのち、業種を問わず多くのお客様との共創を推進。「失敗係」「〜からの建て直し」を立ち回れてこそ一流のデザイナーと信じて活動中。

日本アイ・ビー・エム株式会社
IBM Garage
デベロッパー

森 智章
Mori Tomoaki

2015年日本IBM入社。製品サポート部門、インフラ構築・運用部門を経てIBM Garageへ異動。日中はMVP開発、夜間は自宅サーバー管理、週末はIBMの仲間と立ち上げたサークル「ぴゅあぴゅあ倶楽部」で量子コンピューターの面白さを広める活動に奮闘している。