シェアリングビジネスプラットフォーム

The Sharing Business Platform

新　井　祐　也

要　約　シェアリングビジネスの広まりから，シェアリングビジネス向けアプリケーションの需要が見込まれる．シェアする対象は，オフィスや自動車など様々であるが，多くのサービスでは，シェアする対象を予約する機能や，利用後に実績を計上する機能を使う．日本ユニシスでは今後のシェアリングビジネスの拡大に向け，シェアリングビジネスで共通的に使われる機能を提供する「シェアリングビジネスプラットフォームサービス」を開発し，2017 年度からサービスの提供を開始した．

シェアリングビジネスプラットフォームサービスは，マイクロサービスアーキテクチャと WebAPI を採用し，アジャイル開発にスクラムを利用することにより，事業者の多様なビジネス要求に迅速に対応するとともに，システムを柔軟に運用することができる．

Abstract　Due to the spread of sharing business, demand for sharing business applications is expected to increase in the future. The target to be shared varies depending on services such as office and automobile, but for many services, it is necessary to reserve targets to be shared and to record actuals after use. Nihon Unisys has developed “Sharing Business Platform Service” that provides common functions necessary for sharing business to expand the future sharing business and started providing services from FY 2017.

The Sharing Business Platform Service adopts micro service architecture and Web API, and by using Scrum for Agile development, it can quickly respond to various business demands of business operators and can operate the system flexibly.

1　は　じ　め　に

シェアリングビジネスとは，個人や企業が保有する資産をインターネットを通じて個人や企業にシェア（主に時間貸し）するビジネスの総称である．国内市場規模は，2015 年度に約 285 億円であったものが，年平均成長率は 18％で推移し，2021 年度には 1070 億円まで拡大すると予測されている[1]．それに伴い利用者の増加や新たなビジネス要求が想定され，シェアリングビジネス向けアプリケーションの需要が今後ますます見込まれる．

シェアリングビジネスにおいてシェアする対象は，オフィスや自動車などサービスにより様々であるが，多くのサービスでは，シェアする対象を予約する機能や，利用後に実績を計上する機能を使う．また，シェアする対象だけでなく，サービスを利用するユーザを管理する機能も同様である．これらは様々なシェアリングサービスで共通する機能である．日本ユニシス株式会社（以降，日本ユニシス）はこれらの共通的な機能を提供する基盤として「シェアリングビジネスプラットフォーム」（以降，SBPF）を開発し，提供を開始した．

本稿では，まず 2 章において SBPF の概要を説明する．続く 3 章と 4 章では，シェアリングビジネスの変化に柔軟に対応するための考慮点を交えて，SBPF のアーキテクチャと開発・

運用プロセスについて述べる．

2　シェアリングビジネスプラットフォームの概要

SBPF は，シェアオフィスやカーシェアリングなど，様々なシェアリングビジネス向けアプリケーションの構築を支援するためのサービスである．SBPF は，シェアリングビジネスで共通的に利用される機能を WebAPI で提供する．シェアリングビジネス向けアプリケーションの開発者は，それらを組み合わせて，要求に応じたビジネスロジックやユーザインタフェースを開発する（図 1）．

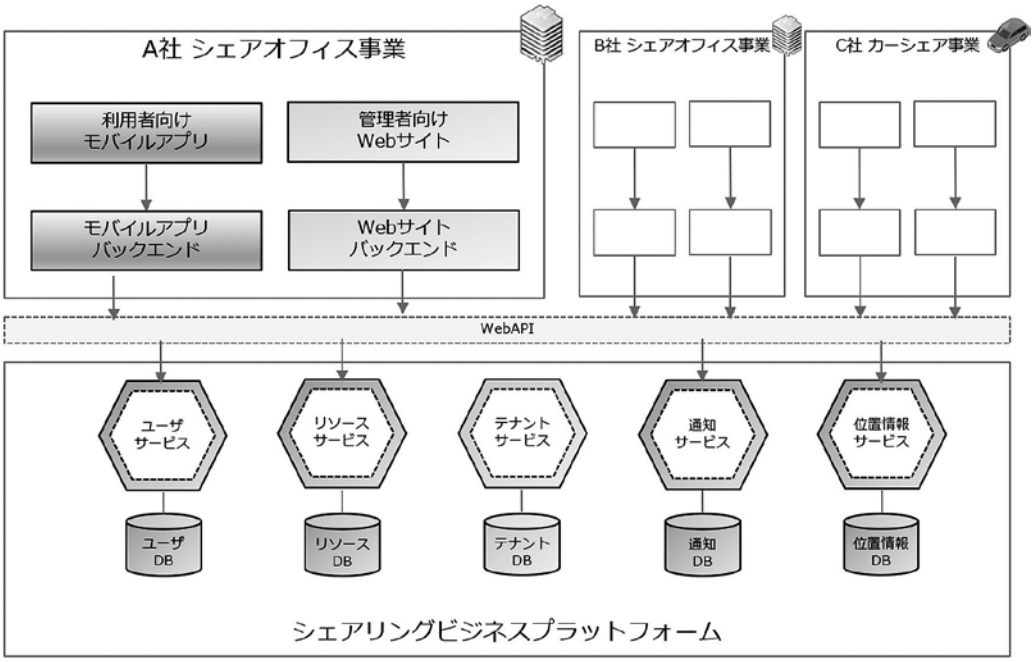

図 1　シェアリングビジネスプラットフォーム概要

SBPF はマイクロサービスアーキテクチャを採用し，関連するドメイン[*1] ごとにサービスを分割している．リソースサービス，ユーザサービス，位置情報サービス，通知サービス，テナントサービスの五つに分け，サービスの集合体として SBPF を構成している．各サービスの機能について，以下で説明する．

1）リソースサービス

自動車やオフィスなど，シェアリングビジネスの種類によってシェアする対象は様々である．例えばシェアオフィス事業の場合，シェアする対象は会議室やワークスペース，プロジェクタやホワイトボードなどの設備が考えられる．SBPF では，これらのシェアする対象すべてをリソースという概念に汎化して管理しており，リソース同士を柔軟な階層構造で表現できる（図 2）．また，リソースそれ自体の管理に加えて，リソースの予約を管理する機能や，リソースの利用後に実績を計上する機能を提供する．

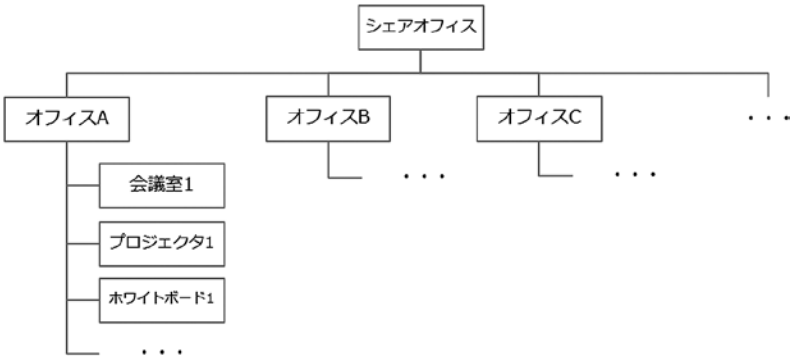

図 2　リソースの階層構造例

2）ユーザサービス

シェアリングビジネスにおいて，シェアする対象（リソース）は様々であるが，シェアする対象の利用者（ユーザ）が必ず存在する．そのため，会員登録や認証などのユーザを管理する機能が求められる．また，ユーザは個人と法人に大別できる．SBPF は，個人や法人を対象としたシェアリングビジネスの構築を支援するために，ユーザが所属する組織（企業），パーミッションなどの権限やロールを管理する機能を提供する．

3）位置情報サービス

カーシェアリング事業では自動車の現在位置，シェアオフィス事業ではオフィスの住所など，位置情報も扱う．ユーザとリソース（自動車やオフィス）の位置情報から，ユーザの近くにあるリソースの検索，ユーザとリソースの距離を計算する機能を提供する．

4）通知サービス

ユーザに通知を送るサービスである．位置情報サービスと合わせて利用することで，位置情報に連動したサービスを開発できる．

5）テナントサービス

SBPF は，複数のシェアリングビジネスを一つのプラットフォームで取り扱うマルチテナント型のサービスである．個々のシェアリングビジネス事業者をテナントとして取り扱うために，新規テナントの登録や，各テナントを識別するためのアクセスキーの発行などを行う．

3　シェアリングビジネスプラットフォームサービスのアーキテクチャ

2 章でも述べたように，SBPF はシェアオフィスやカーシェアリング事業など様々なシェアリングビジネスを対象としている．また同じシェアオフィス事業であっても，事業者によってビジネス要求は異なる．そのため，様々なビジネス要求に対応できるように汎用的かつカスタマイズ可能な形で，SBPF の機能をシェアリングビジネス向けアプリケーションの開発者に提供している．加えて，SBPF のサービス自体が，機能変更やトラフィックの増加に柔軟に対応できる構造を持つ．本章では，これらの要件を実現するために採用したアーキテクチャや，設計時の考慮点について述べる．

3.1　マイクロサービス

マイクロサービスはシステムを複数のサービスの組み合わせで構築するアプローチであり，

個々のサービスは独立したプロセス上で動作し，WebAPI などのシンプルで軽量な手段で通信する．2014 年に James Lewis 氏 /Martin Fowler 氏が公開した記事「Microservices」により，広く注目されるようになった[2]．マイクロサービスは，Amazon や Netflix といった大規模な Web サービスを提供する企業を中心に採用されているアーキテクチャスタイルである．

マイクロサービスに対して，従来のアーキテクチャは「モノリシック（一枚岩）アーキテクチャ」と表現される．シェアリングビジネス向けアプリケーションを例に，モノリシックアーキテクチャとマイクロサービスアーキテクチャの構成イメージを，図 3 に示す．

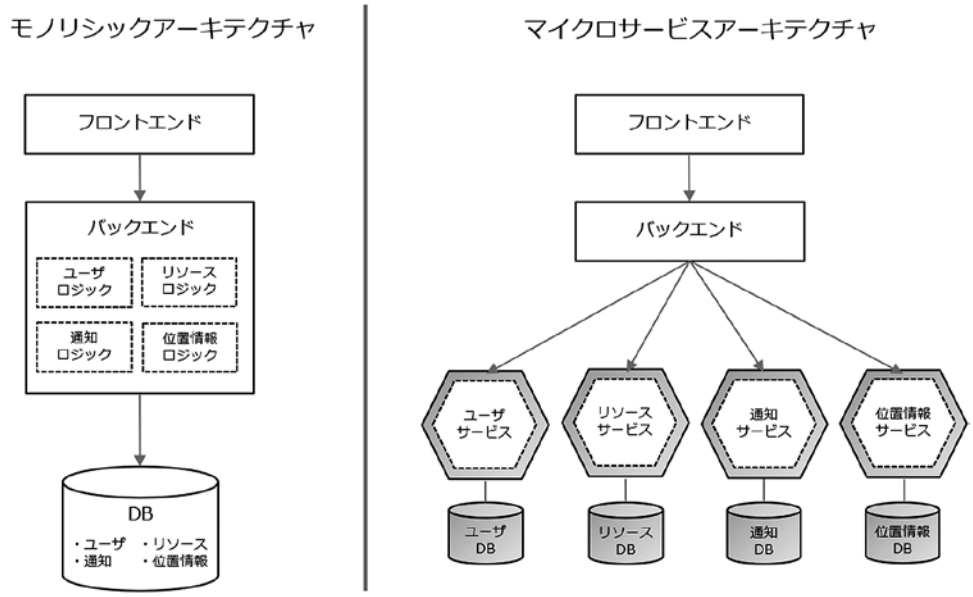

図 3　モノリシックアーキテクチャとマイクロサービスアーキテクチャ

モノリシックアーキテクチャと比較し，マイクロサービスアーキテクチャは次の利点が挙げられる．SBPF に当てはめた際のメリットを以降の本節で述べる．

- デプロイの容易性：他のサービスとは独立してデプロイできる
- スケーリング：システム全体ではなく，サービスごとにスケーリングできる
- 技術の多様性：サービスごとに異なる技術を採用できる

SBPF はマイクロサービスアーキテクチャを採用し，リソースサービスやユーザサービスを独立したサービスとして開発している．サービスごとに，コードベース，アプリケーションサーバ，データストアを分離しており，あるサービスの機能に変更を加えた場合，影響範囲はサービス内に限定できる．ビルドと自動テストについても，システム全体ではなく一つのサービスだけ実施すればよいので，実行時間を短縮できる．そして，変更したサービスを本番環境にデプロイする際も，他のサービスに影響を与えずに，独立したデプロイができる．

スケーリングの面でも利点がある．すべての機能を一つのシステムに含めると，特定の機能にトラフィックが増えた場合でも，システム全体をスケールさせなければならない．マイクロサービスを採用した SBPF では，例えばリソースサービスのトラフィックが増えた場合，シ

ステム全体をスケールするのではなく，ユーザサービスは現状のままで，リソースサービスだけスケールするといった対応ができる．

SBPF の各サービスは同時期に開発したため，同じプログラミング言語・データストアを利用しているが，今後はサービスごとに利用する技術を変えることができる．例えば，位置情報サービスのデータストアをより高速な NoSQL に変更するといったことが，他のサービスに影響を与えずにできる．

3.2 WebAPI

アプリケーション開発者に機能を提供する代表的な方法として，ライブラリによる提供と WebAPI による提供の 2 種類がある．ライブラリによる提供は，古くから採用されている実績のある方法であるが，プログラミング言語や実行環境がライブラリによって限定されてしまい，技術的多様性が失われるといった問題がある．また，ライブラリの機能がバージョンアップした際，最新のライブラリを配布し，アプリケーションに組み込まなければならない．

SBPF は，様々なシェアリングビジネス向けアプリケーションを対象とするため，プログラミング言語や実行環境は限定したくない．また，SBPF がバージョンアップした際に，アプリケーション側の手間をかけずにバージョンアップした機能を提供できることが望ましい．これらの理由から，SBPF は WebAPI によって機能を提供する方法を採用している．WebAPI とは，「HTTP プロトコルを利用してネットワーク越しに呼び出す API」である．そのため，HTTP 通信ができるアプリケーションであれば，プログラミング言語や実行環境は問わない．

SBPF は，WebAPI の設計方法としてデファクトスタンダード[*2] となっている REST 形式を採用している．REST とは，URI で表現されたネットワーク上のリソース（操作する対象）に対して，HTTP のメソッド（GET，POST，PUT など）でアクセスする WebAPI の設計方式である（表 1）．データフォーマットは，現在のスタンダードとなっている JSON を利用している．また，HTTP の仕様に沿い，WebAPI の実行結果を HTTP ステータスコードで表現している（表 2）．

デファクトスタンダードを遵守することで，シェアリングビジネス向けアプリケーションの開発者が，利用方法を容易に類推でき，既存のクライアントライブラリを流用できる．それによって，開発にかかる手間やストレスを軽減できる．

表 1　REST 形式に準拠したユーザ API の例

API	メソッド	URI	成功時のステータスコード
ユーザの一覧取得/検索	GET	/users	200
ユーザの取得	GET	/users/{id}	200
ユーザの新規登録	POST	/users	201
ユーザの情報の更新	PUT	/users/{id}	204
ユーザ情報の一部更新	PATCH	/users/{id}	204
ユーザの削除	DELETE	/users/{id}	204

表 2　HTTP の仕様に準拠したステータスコード

ステータスコード	状　態	説　明
200	OK	リクエストが正常に処理された
201	Created	リクエストが正常に処理され，新規リソースが作成された
204	No Content	リクエストが正常に処理されたが，返す新規情報はない
400	Bad Request	サーバーが理解できない無効な要求である
401	Unauthorized	要求されたリソースには認証が必要である
403	Forbidden	要求されたリクエストは拒否された
404	Not Found	要求されたリソースはサーバーに存在しない
500	Internal Server Error	サーバーでエラーが発生した

3. 3　LSUDs とオーケストレーション層

Netflix 社の Daniel Jacobson 氏は「The future of API design: The orchestration layer」という記事において，LSUDs（large set of unknown developers）と SSKDs（small set of known developers）について紹介している[3]．LSUDs = 未知のたくさんの開発者，SSKDs = 既知の少数の開発者という意味で，WebAPI がターゲットとする開発者を表す．

SBPF の WebAPI は，様々なシェアリングビジネス向けアプリケーションの開発者をターゲットとするため，LSUDs に分類できる．LSUDs の WebAPI は汎用的であるが故に，Web ブラウザやスマートフォンアプリケーションのクライアント層から利用する場合，一つのアクションを行うのに複数の WebAPI にアクセスするため，不要なデータも受け取らなければならずペイロードが大きくなるといったことが起こる．これらの問題を，クライアントと WebAPI の間にオーケストレーション層を設けることで解決する．オーケストレーション層で複数の WebAPI を組み合わせたビジネスロジックを実装し，またクライアントが望むデータのみフィルタリングすることで，シェアリングビジネス向けアプリケーションのクライアントごとに最適なエンドポイントを提供できる（図 4）．

3. 4　マイクロサービスにおけるデータ整合性

従来のモノリシックアーキテクチャで複数のデータを更新する場合，トランザクションによってデータの整合性を担保する．マイクロサービスアーキテクチャではサービス単位でデータストアを保持している．そこで，分散したデータストアの整合性を保つために SBPF は結果整合性に基づいた設計としている．結果整合性とは，「処理の途中ではデータの整合性が取れていないタイミングもあるが，最終的にデータの整合性が担保されていればよい」という考え方である．例えば，「一つの処理内でユーザサービスとリソースサービスのデータを更新するシェアリングビジネスアプリケーション」を開発する際は，処理の途中で失敗するケースに備えて，自動的なリトライや更新前の状態に戻す補償処理を実装する，あるいはアプリケーションの運用担当者が対処する，といった方針を決める．

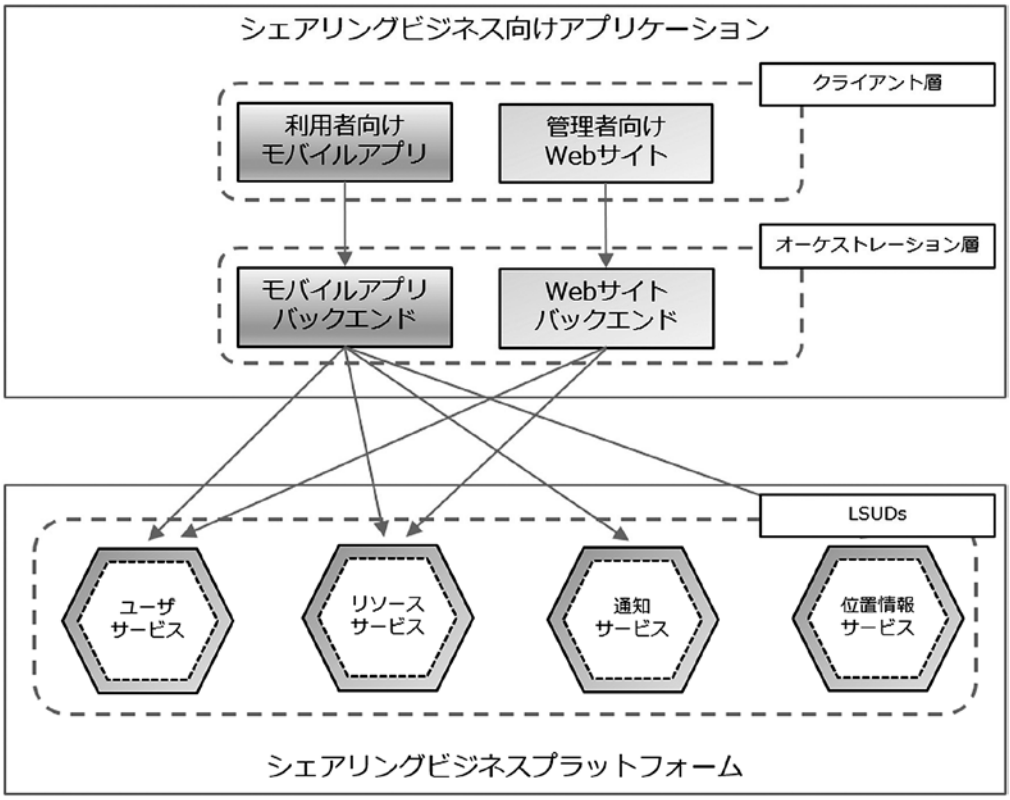

図 4　LSUDs とオーケストレーション層

3. 5　マルチテナントとセキュリティ

SBPF は，複数のシェアリングビジネス事業者を一つのプラットフォームで取り扱うマルチテナント型のサービスである．そのため，SBPF の WebAPI を利用するシェアリングビジネス事業者の認証を行う．一般的な認証の仕組みとして，API キー，アクセストークン，クライアント証明書を利用するといった方法がある．シェアリングビジネス向けアプリケーションにおいて，SBPF の WebAPI にアクセスするのはバックエンドプログラム（図 4 のオーケストレーション層）の役割となる．プログラムへの組み込みと管理のし易さの点から，SBPF では API キーによる認証方式を採用している．次に，API キーを利用した SBPF の認証の流れを説明する．

SBPF では，はじめに個々の事業者をテナントとして登録し，WebAPI を利用するための API キーを発行する．シェアリングビジネス向けアプリケーションは，発行された API キーをリクエストに付加し，SBPF の WebAPI にアクセスする．全ての WebAPI は HTTPS 接続によって暗号化される．SBPF は API キーの正当性をチェックし，どのテナントからのアクセスかを判断する．API キーの正当性をチェックする機能は，テナントサービスが担っている．アプリケーションからリソースサービスやユーザサービスが呼び出された際，テナントサービスに API キーの正当性を問い合わせる．また，SBPF の WebAPI をより堅牢にするために，API キーに対してアクセス可能な IP アドレスを制限する機能を設けている（図 5）．これにより，テナントの API キーが，何らかの理由で外部の第三者や他テナントの関係者に漏洩しても，WebAPI への不正なアクセスをブロックできる．加えて，API キーの正当性チェックの結果

はリアルタイムで監視し，不正アクセスの予兆を検知した際は，即時にアラートを発信する仕組みを用意している．

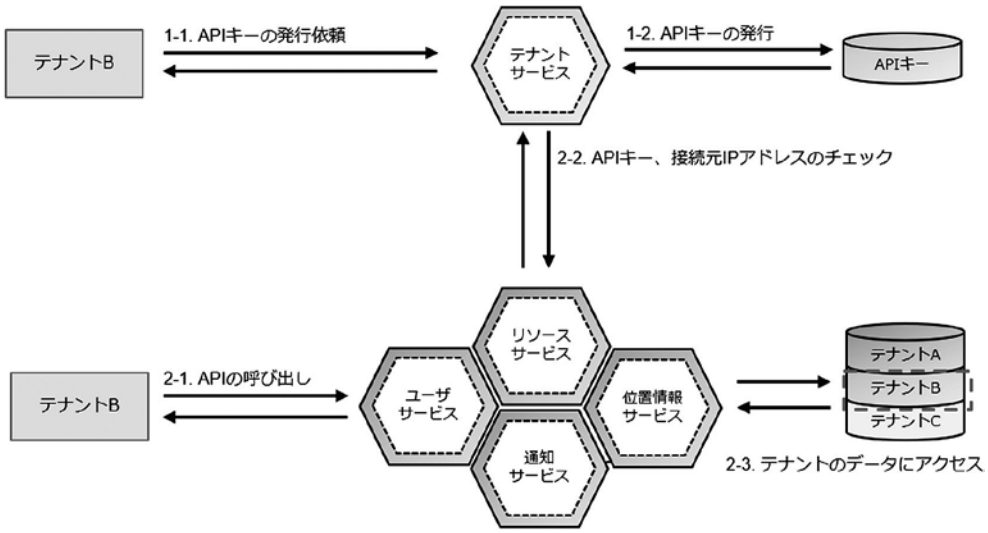

図5　マルチテナントとセキュリティ

3.6　柔軟なデータ構造の実現

シェアリングビジネス事業者ごとに実現したいビジネス要求が異なる．例えば，カーシェアリングの場合，シェア対象となる自動車をリソースとして登録するが，自動車の属性として，車種，乗車定員，排気量などが考えられる．シェアオフィス事業の会議室の場合，座席数，ホワイトボードの有無などの情報がある（図6）．このように，シェアリングビジネスの種類や事業者ごとに，取り扱う属性データが異なるため，属性データは柔軟に追加できるようにしたい．また，条件に一致する属性データを検索するシーンも想定されるため，格納した属性データは効率的に検索できるようにしたい．

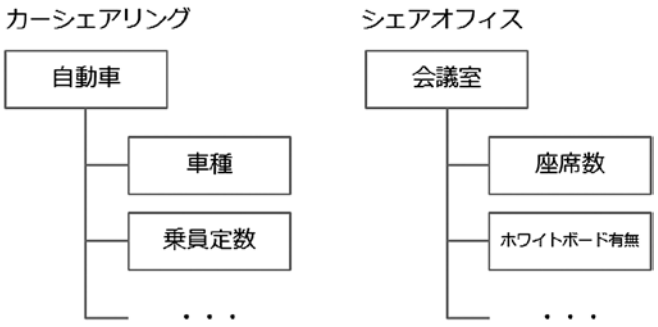

図6　属性データのイメージ

SBPF は，複数のシェアリングビジネス事業者が利用する共通のプラットフォームであり，事業者ごとのビジネス要求に応じた属性データを格納するために RDB のスキーマを変更することはできない．RDB でスキーマを変更せずに，可変データを格納する方法として，SBPF

は属性データを JSONB 型[*3] として格納する方法を採用している．PostgreSQL の JSONB は，他のデータと同じく SQL でデータにアクセスでき，かつインデックスを活用した効率的な検索ができる．

図 7 は，リソース登録の WebAPI の例である．“seats”と“whiteboard”の二つの属性データが JSON として登録される．

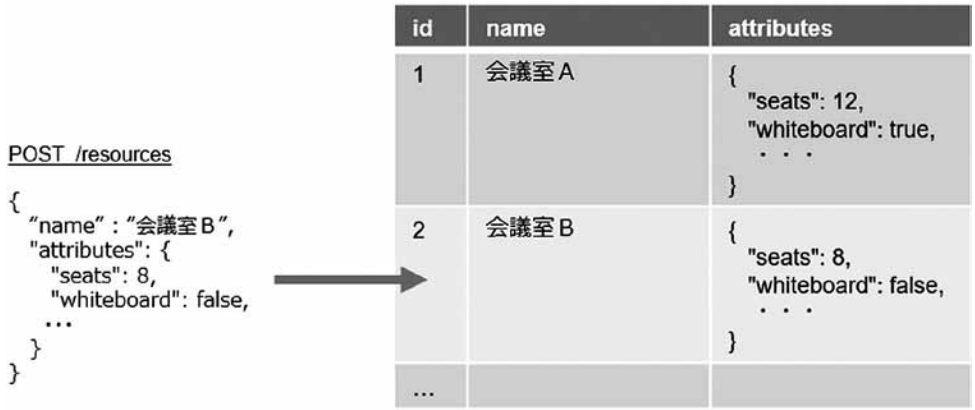

図 7　リソースの新規登録の例

一覧取得/検索の WebAPI は，データをフィルタリングするための条件を GET パラメータに指定して呼び出す．WebAPI 内部では，フィルタリング条件を解析して最適な SQL を組み立て，RDB にアクセスする．フィルタリング条件は，“[項目名] [比較演算子] [値]”の形で記述し，条件が複数ある場合は，論理演算子で条件をつなぐ．比較演算子は，eq（等しい），gt（より大きい），lt（より小さい）などが利用できる．フィルタリング条件と対応する SQL の例を，図 8 に示す（“10 人以上が入室できて，かつホワイトボードがある会議室”を検索）．

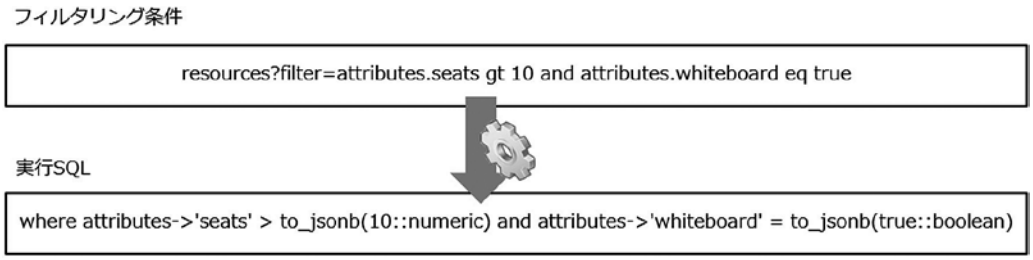

図 8　フィルタリング条件と実行 SQL

4　サービスのための開発・運用プロセス

シェアリングビジネスは急速に拡大を続ける市場であり，それに応じて新たなビジネス要求や利用者の増加が想定される．このような変化の激しい環境下で，持続的にサービスを成長させるため，日本ユニシスは SBPF の開発・運用プロセスで次のような要素を追求している．

- 要求の変化やフィードバックに柔軟に対応できる開発プロセスとツール
- 開発・改善したプロダクトを継続的にリリースできる体制と技術
- ビジネスの成長に合わせてシステムリソースをスケールできるクラウド環境

本章では，これらの要素を考慮したSBPFの開発・運用プロセスについて述べる．

4.1　開発プロセス

シェアリングビジネスは変化の激しい市場のため，開発途中であっても新たなニーズをキャッチアップし，SBPFのサービスを成長させたい．そのためSBPFでは，開発途中での変更を認めないウォーターフォール型の開発プロセスではなく，要求の変化やフィードバックに対応できるアジャイルな開発プロセス「スクラム」を利用している．スクラムはKen Schwaber氏とJeff Sutherland氏が提唱した具体的なアジャイル開発プロセスの一つであり[4]，2017年現在，世界中で最も利用されているアジャイル開発プロセスである．スクラムで開発プロセスを進めるにあたり，Atlassian社のJira SoftwareやConfluenceなどのツールを使って，バックログの管理とチームの情報共有を行っている（図9）．

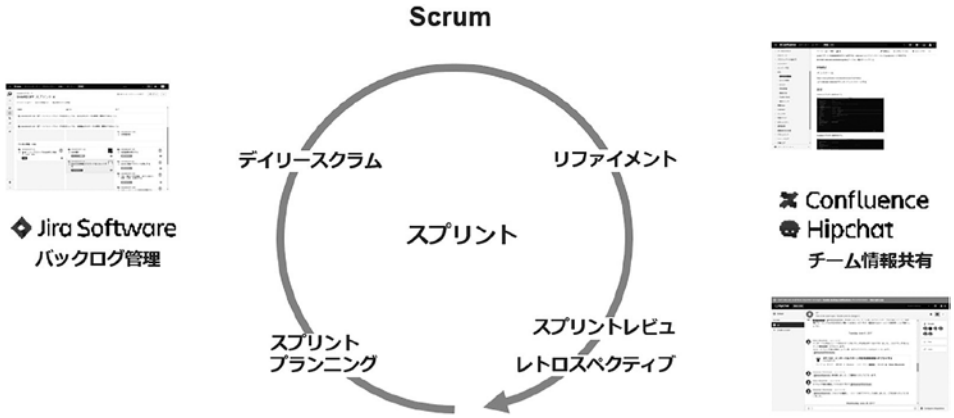

図9　開発プロセスとツール

アジャイル開発において，要求の変化に柔軟に対応するために，開発言語/フレームワークには，少ないコード量でビジネスロジックを記述でき，かつ欲しい機能がすぐに利用できるフルスタックなフレームワークを選定することが望ましい．SPBFでは，次の理由により「Grails」を採用している．

- プログラミング言語にGroovy[*4]を用いているため，Javaエンジニアが容易に習得できる
- 現在Javaのフレームワークとしてデファクトスタンダードとなりつつある「Spring Boot」をベースとしているため，Springのエコシステム[*5]を容易に利用できる

4.2　運用プロセス

アジャイル開発プロセスで改善したプロダクトを，迅速に，頻繁に，そして確実にプロダクション環境にリリースするためには，開発（Development）と運用（Operations）が共同する体制が望ましい．また体制面に加えて，デリバリープロセスの自動化の技術も有用である．

SBPFの開発・保守では，フルスタックなチーム体制を目指し，開発と運用を同じチームで行い，開発担当者と運用担当者の連携が密に取れるチーム体制にしている．技術面では，ビルド，テスト，パッケージ，デプロイのデリバリープロセスを，ツールを使って自動化している．

開発担当者はコードとテストを書き，開発用ブランチにプッシュする．レビューが終わると，メインブランチにマージされ，ビルドとテストが自動実行される．すべてのテストが成功すると，パッケージングされたモジュールが作成され，任意のタイミングで，運用担当者がプロダクション環境にデプロイする（図 10）．このようにデリバリープロセスを自動化しているため，人手による作業ミスや作業コストを抑制できる．

また，SBPF はマイクロサービスアーキテクチャの利点により，対象のサービスに限定してビルドとテストを実施するため，デプロイに要する時間を短縮できる．デプロイも対象のサービスに限定して行うので，プロダクション環境下での問題発生のリスクを低減する．

ユーザサービスやリソースサービスなど，SBPF の全てのサービスは，クラウド環境で稼働させている．今後，シェアリングビジネス市場の成長により利用者が増加した場合，ユーザサービスのシステムリソースだけ増強するといった戦略が，シームレスに実行できる環境となっている．

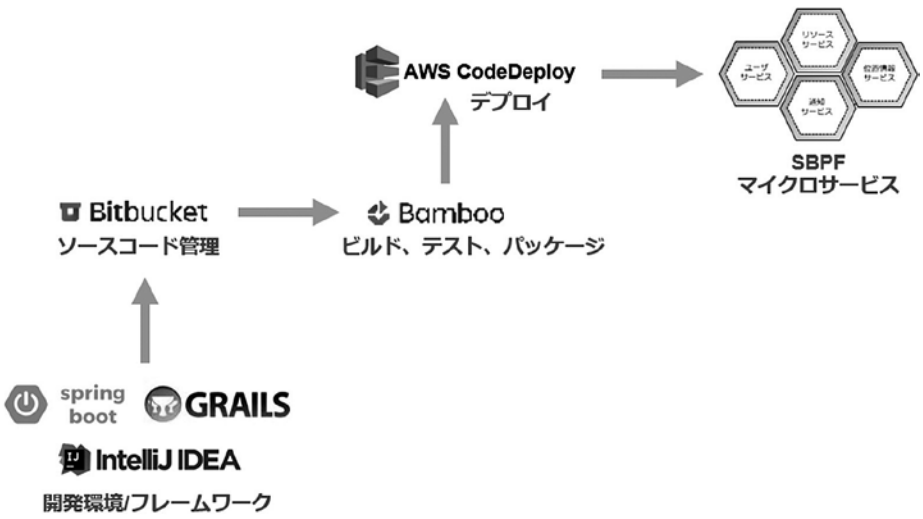

図 10　開発と運用のサイクル

5　お　わ　り　に

シェアリングビジネスの広まりから，今後ますますシェアリングビジネス向けアプリケーションが開発されるだろう．アプリケーションをゼロから開発するのではなく，SBPF の WebAPI を組み合わせて利用することで，それぞれのシェアリングビジネスに応じたロジックやユーザインタフェースの作成に注力できると考える．

SBPF は今後，アプリケーション開発者からのフィードバックや，テンプレートアプリケーション[*6]の開発を通じてノウハウを蓄積し，公開していく予定である．また，シェアリングビジネスはこれからも発展し，より新しいビジネスモデルも出てくるだろう．SBPF の開発・運用チームは，市場やビジネス要求の変化に対応し，サービスを成長させていく所存である．

＊1　ビジネス上の関心領域を指す.
＊2　標準化機関等が定めた規格ではなく，市場における競争や広く採用された「結果として事実上標準の基準」を指す.
＊3　バイナリ形式で JSON を格納する.
＊4　Java VM 上で動作する動的なスクリプト言語．構文は Java の上位互換である.
＊5　Spring Data や Spring Security などの周辺ライブラリを指す.
＊6　シェアオフィスやカーシェアなどのシェアリングビジネス向けアプリケーションのテンプレート.

参考文献

［1］ 矢野経済研究所,「シェアリングエコノミー（共有経済）に関する調査」，2017 年 11 月　http://www.yano.co.jp/press/press.php/001763
［2］ James Lewis/Martin Fowler, “Microservices”, MARTINFOWLER.COM, March 2014.　https://martinfowler.com/articles/microservices.html
［3］ Daniel Jacobson, “The future of API design: The orchestration layer”, The Next Web, Dec. 2013.
http://thenextweb.com/dd/2013/12/17/future-api-design-orchestration-layer/
［4］ Ken Schwaber，Jeff Sutherland,「スクラムガイド」，Scrum.org and Scrum Inc., 2013 年 7 月.
https://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-JA.pdf
［5］ Sam Newman 著，佐藤 直生 監訳,「マイクロサービスアーキテクチャ」，オライリー・ジャパン，2016 年 2 月.
［6］ 水野 貴明,「Web API: The Good Parts」，オライリー・ジャパン，2014 年 11 月.

※上記参考文献に含まれる URL のリンク先は，2018 年 4 月 19 日時点での存在を確認.

執筆者紹介

新　井　祐　也（Yuya Arai）
2004 年日本ユニシス・ソフトウェア(株)入社．通販向け会員/在庫/商品管理サービスの開発に従事．2016 年よりシェアリングビジネスプラットフォームの開発を担当．現在，プラットフォームサービス本部サービス開発部 DevOps 推進室に所属．認定スクラムマスター.