多種多様なサービスを効率的に実現するマイクロサービス開発技術

Microservice Development Technologies to Realize Efficient Development of Various Services

小島　芳治 KOJIMA Yoshiharu　　斉藤　　稔 SAITO Minoru

東芝グループは，社会が直面する様々な課題をデジタル技術で解決するCPS（サイバーフィジカルシステム）テクノロジー企業を目指しており，CPSとして提供する多種多様なサービスを，効率的に開発する必要がある。マイクロサービスアーキテクチャーは，小規模なサービスの集合体としてサービスを作り上げるアーキテクチャーであり，再利用性が高くサービスの組み合わせが容易であるという特長を持つため，効率的なサービス開発に有効である。

東芝デジタルソリューションズ（株）は，マイクロサービスを適切な粒度で設計し，迅速に開発するマイクロサービス開発技術として，ガイドブック，アーキテクチャーパターン，及び実行基盤を標準化し，社内に適用している。また，開発現場への展開とフィードバックを繰り返すことで，マイクロサービス開発技術を継続的に改善して利用価値を高め，多種多様なサービスの迅速な開発と提供を可能にした。

The Toshiba Group has set the goal of becoming a cyber-physical systems (CPS) technology company that can offer solutions to various social problems based on digital technologies. In order to efficiently provide a broad range of CPS services, we have adopted a software architecture style known as microservices architecture, in which a service is constructed through the collection of small-scale services called microservices. A feature of this method is that it allows a service to be easily constructed as an aggregate of microservices. However, it is necessary to design microservices of the appropriate size and to develop them in an agile manner, for flexible transformation of the microservices themselves.

As part of this approach, Toshiba Digital Solutions Corporation is promoting the establishment of a development environment for the swift and agile development of microservices through the preparation of development guides, architecture patterns, and execution platforms. We are also responding to the sophistication of microservices by continuously improving related technologies and promoting the rapid expansion of their application based on a feedback cycle from development sites.

1. まえがき

マイクロサービスアーキテクチャーは，マイクロサービスという小規模のサービスの集合体として，より大きな規模のサービスを作り上げるアーキテクチャーであり，利用方法が汎用的かつ簡便で，再利用性が高い。このため，マイクロサービスを組み合わせることで，新たなサービスを容易に開発できる。

CPSの領域では，ソフトウェアコンポーネントに再利用性を持たせるため，マイクロサービスアーキテクチャーを重要な技術の一つに位置付けている。

一方で，個々のマイクロサービスは，一度開発すればよいというものではなく，ビジネスの変化への対応が継続的に求められる。そのため，マイクロサービスには高い変更容易性が求められるが，このためには適切な粒度での設計と迅速な開発が必要である。

ここでは，東芝デジタルソリューションズ（株）が展開している，変更容易性を高めるためのマイクロサービス開発技術と，その実践事例について述べる。

2. マイクロサービス開発の概要

2.1 マイクロサービスアーキテクチャー

マイクロサービスは，互いに連携して動作する小規模で自律的なサービス（機能）であり，マイクロサービスアーキテクチャーに基づくサービスは，マイクロサービスの組み合わせによって構築されるものである。マイクロサービス間の連携は，HTTP（Hypertext Transfer Protocol）などの汎用プロトコルによるネットワークを経由したAPI（Application Programming Interface）呼び出しで行われ，互いの依存度が小さい疎結合な作りになっている。そのため，マイクロサービス間の連携が容易であり，マイクロサービスを組み合わせて新たなサービスを柔軟に作ることができる。一方で，ネットワーク経由で複数のマイクロサービスがつながることによるサービス構成の複雑化などのデメリットがあり，あらゆるサービスに適しているわけではない。

CPSでは，そのフレームワークとして東芝IoT（Internet of Things）リファレンスアーキテクチャー（Toshiba IoT

Reference Architecture，TIRAと略記）が策定された。TIRAは，マイクロサービスによるAPI連携を規約としており，これに基づいて開発されたマイクロサービスは再利用可能な部品となる。インダストリアルIoTサービスであるTOSHIBA SPINEXは，TIRAに準拠しており，いち早くマイクロサービスアーキテクチャーを導入した事例の一つである。

一方で，マイクロサービスは一度の開発で完成するものではなく，ビジネス変化への対応が継続的に求められる。ビジネスの変化に柔軟に対応するためには，マイクロサービスに高い変更容易性を持たせる必要がある。

2.2　マイクロサービス開発の課題

マイクロサービスに高い変更容易性を持たせるためには，単一の機能改善時に他機能に影響を与えないような設計が求められる。更に，変更を素早く行うため，短期間で開発・リリースのサイクルを繰り返す必要がある。これらを実現するため，マイクロサービス開発では，次の2点の課題に対応する必要がある。

⑴　マイクロサービスの適切な粒度での設計

マイクロサービスの粒度が適切でないと，変更容易性が失われるおそれがある。例えば，マイクロサービスが複数の機能を持っているとすると，そのうち一つの機能を更新するときに，ほかの機能も含めて停止する必要が生じる。また，機能が適切に分割されていたとしても，それらの機能が利用するデータが適切に分割されていないと，ある機能を更新するときにほかの機能の停止が求められる場合がある。このような，機能とデータの境界を適切に定めた設計は，容易ではない。

⑵　迅速に開発を行うための技術やインフラの導入

短期間での開発には，インフラ調達を迅速に行うためのパブリッククラウド技術の利用や，サービスの実装を迅速に行うためのOSS（オープンソースソフトウェア）技術の活用が欠かせない。一方で，パブリッククラウド技術やOSS技術は進歩が早いため，常に最新技術に追従し，習得し続ける必要がある。

これらの課題に対応するためには，マイクロサービスの設計や，パブリッククラウド技術及びOSS技術の利用などのノウハウを，十分に蓄え，共有する必要がある。当社は，このようなノウハウをマイクロサービス開発技術として集約・標準化し，共通基盤として開発現場に展開している。

3.　マイクロサービス開発技術の標準化と共通基盤の適用

3.1　マイクロサービス開発技術の標準化

マイクロサービスの開発技術として，ガイドブック，アーキテクチャーパターン，及び実行基盤を標準化し，共通基盤として社内に展開している。これらを用いて，2.2節で挙げた課題のそれぞれについて，次のように対応している。

⑴　書籍やWebで公開されている技術情報をプロジェクトに適用した結果を基に，マイクロサービスの設計や開発方法をガイドブックにまとめている。ガイドブックに従うことで，適切な粒度でマイクロサービスを設計できる。

⑵　パブリッククラウド技術やOSS技術のノウハウをガイドブックにまとめている。更に，それらを利用した推奨アーキテクチャーを，アーキテクチャーパターンとして集めている。ガイドブックやアーキテクチャーパターンを利用することで，パブリッククラウドやOSSを用いた開発技術を素早く取得し，効果的に利用できる。

⑶　OSS技術を組み込んだフレームワークとコンポーネントを，実行基盤として整備している。実行基盤をベースにマイクロサービスを開発することで，迅速な開発が可能となる。

3.2　マイクロサービス設計への共通基盤の適用

マイクロサービス開発のガイドブックは，マイクロサービスの特徴や，マイクロサービスアーキテクチャーが適したサービス，マイクロサービス開発の課題とその解決方法などを提示している。2.2節に示したマイクロサービスの粒度については，その観点として，トランザクション（一連のデータ処理の単位），更新頻度，負荷，及び開発組織の四つを挙げている。これらの観点で，サービス利用時の業務を基準にマイクロサービスの分割検討を繰り返して，適切な粒度を決定する。このマイクロサービスの分割手順を，**図1**に示す。

3.3　マイクロサービス開発への共通基盤の適用

パブリッククラウド技術やOSS技術は，公式，非公式ともに多くの解説情報があふれているが，必要な情報を拾い上げることが難しい。そこで，当社は，パブリッククラウド技術やOSS技術について，これらを活用したマイクロサービス開発を検証，評価した上で，推奨する構成や，開発手順，各種ノウハウなどを社内ガイドブックとしてまとめている。社内ガイドブックを参考に，パブリッククラウド技術やOSS技術を活用することで，高品質なマイクロサービスを効率的に開発できる。

パブリッククラウド技術を利用したサービスでは，非機能要件について，従来の開発とは異なる観点が必要となる。具体的には，柔軟なコンピューティングリソース調達を前提とした可用性・拡張性や，パブリッククラウドへのアクセスを考慮したセキュリティーなどが挙げられる。当社は，パブリッククラウド技術を利用した推奨アーキテクチャーを集めたアーキテクチャーパターンを用意している。アーキテクチャー

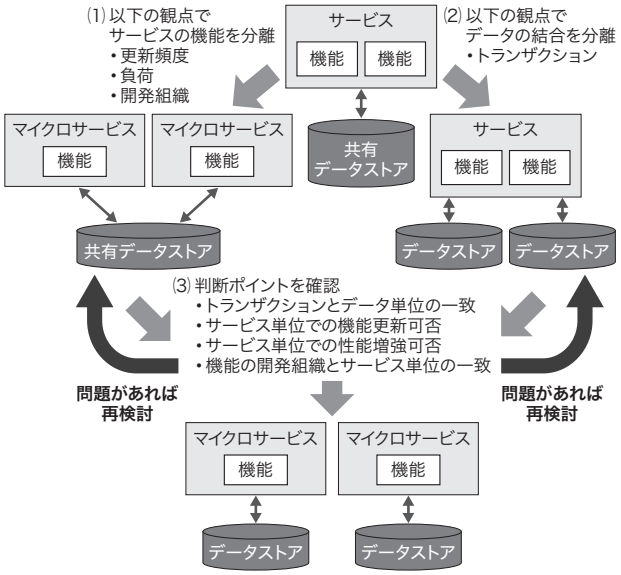
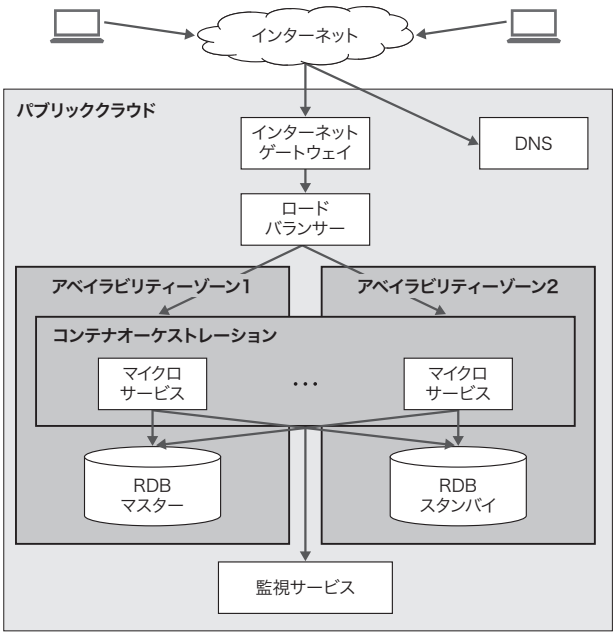


図1. マイクロサービスの分割手順

機能又はデータを分割し，判断ポイントに従って粒度が適切であるかどうかを判断する。適切でない場合は再検討を繰り返して，適切な粒度を決定する。

Procedure for separation of microservices



DNS：Domain Name System
RDB：リレーショナルデータベース

図2. パブリッククラウドを利用したマイクロサービスの構成例

コンテナオーケストレーションツール上にマイクロサービスを構築し，データベース（DB）とともにアベイラビリティーゾーン間で冗長化することで，可用性を高める。

Example of configuration of microservices using public cloud services

パターンは，非機能要求グレード[1]を基に，パブリッククラウド技術を利用する際，特に考慮が必要な要件をピックアップし，それらの要件を実現するアーキテクチャーを提示している。例として，アーキテクチャーパターンで提示している，パブリッククラウド技術を利用した構成例を，**図2**に示す。マイクロサービス開発においては，アーキテクチャーパターンをテンプレートとして設計を進めることで，非機能要件の抜けや漏れを防ぐことができ，それらの要件を満たす設計例を参考とすることで，設計を効率的に進めることができる。

更に，当社は，アプリケーションの実行基盤であり，当社が独自に開発しているStavewareを社内のサービス開発に適用している。これは，昨今では，マイクロサービスアーキテクチャーの普及を踏まえ，マイクロサービス開発にも適用可能なフレームワークやコンポーネントも備えている。マイクロサービスは個々に開発できるが，それらを組み合わせてサービスを構築するためには共通の機能が必要である。機能の例を以下に示す。

(1) HTTPをベースとしたAPIの，標準的な実装であるREST（Representational State Transfer）API
(2) シングルサインオンを実現する認証認可機能
(3) マイクロサービスを連携した処理をトレースするためのログ出力機能
(4) 全マイクロサービスの稼働状況を確認し，障害時に検知・分析する監視機能

Stavewareは，Webアプリケーションのデファクトスタンダードである OSSをベースとしており，先進技術に追従するとともに，マイクロサービスに必要な共通の機能を実現している。Stavewareのユーザーは，サービスごとの個別の機能だけを開発し，共通の機能はStavewareを利用することで，迅速な開発が実現できる。**図3**に，Stavewareを利用して実現されるマイクロサービスの構成を示す。Stavewareは，個々のマイクロサービスに必要な機能を提供するフレームワークであるStaveware Coreと，マイクロサービスとは別のコンポーネントとして横断的な機能を提供するStaveware API Gatewayによって構成される。

4. マイクロサービス開発技術の実践事例

4.1 開発現場との連携

当社のマイクロサービス開発技術は，ビジネスの変化に素早く対応し，技術革新に追従できるように，常に改善を図っている。改善は，先進技術のシーズと開発現場からのニーズの両輪から行っている。ニーズだけでは先進技術の適用が遅れ，シーズだけでは開発現場で適用できない技術となるおそれがある。ニーズとシーズを並行して取り込むことで，開発現場に適用可能な開発技術を作り上げている。

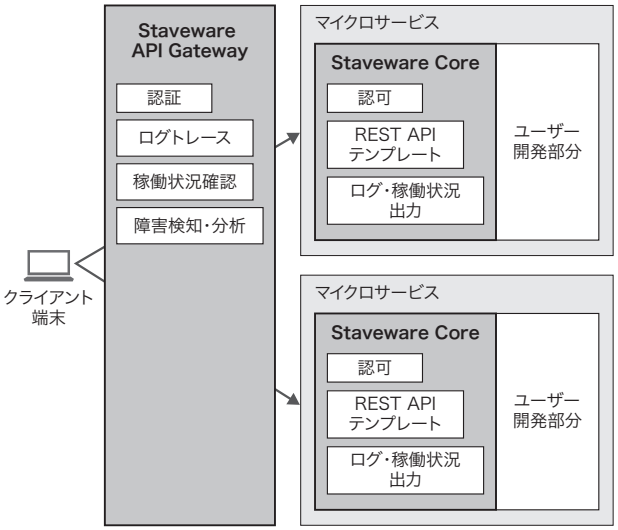


図3. Stavewareを用いたマイクロサービスの構成

個々のマイクロサービスに組み込む必要がある機能をStaveware Coreで，横断的に提供する機能を一つのコンポーネントとしてStaveware API Gatewayで，それぞれ実現している。

Microservices constructed using Staveware framework

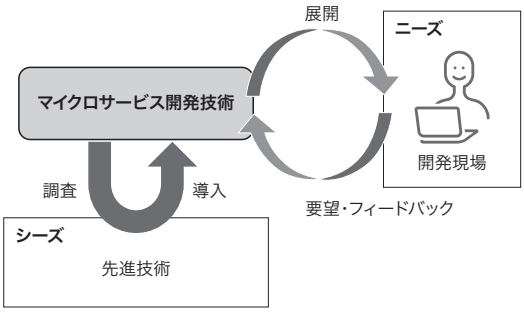


図4. マイクロサービス開発技術を改善するフィードバックサイクル

シーズとして先進技術を調査・導入するとともに，ニーズとして開発現場の要望を取り入れ，更にフィードバックサイクルを繰り返すことで，マイクロサービス開発技術を継続的に改善する。

Feedback cycle to improve microservice development technologies

更に，改善を継続的に実現するため，開発技術と開発現場とのフィードバックサイクルを確立した（**図4**）。

開発現場への適用によって得られた課題のフィードバックと，それに応えて改善した技術の再展開によって，開発現場で役立つ，価値の高い開発技術を素早く展開することを可能としている。

4.2 開発現場からのフィードバック例

開発現場からのフィードバックを得て共通基盤を改善した例として，TOSHIBA SPINEXの開発がある。TOSHIBA SPINEXでは，ユーザー及びIoT機器の管理やデータ分析といった様々なユーザーに，共通で利用される機能を提供するマイクロサービスと，ユーザーごとに異なる画面を提供するユーザーインターフェース（UI）サービスという2段階の構成を採用した。マイクロサービス開発のガイドブックでは，この構成を成功事例として提示し，Stavewareでは，マイクロサービス間の共通機能を持つマイクロサービスと画面を提供するUIサービスのサンプルを同梱（どうこん）し，リファレンスモデルとして提示している。

TOSHIBA SPINEXの開発での知見を基に改善したStavewareは，先進技術を取り込むとともに，ほかの開発現場でも適用を進めており，新たなフィードバックを得て更に改善が進められている。

5. あとがき

当社のマイクロサービス開発技術と，それを改善するためのフィードバックサイクルについて述べた。

今後も，先進技術のシーズと開発現場からのニーズを並行して取り込み，フィードバックサイクルを経て，継続的にマイクロサービス開発技術を改善することで，ビジネスの変化に対応したサービスの，迅速な提供を支えていく。

文　献

(1) 情報処理推進機構．“システム構築の上流工程強化（非機能要求グレード）”．社会基盤センター．<https://www.ipa.go.jp/sec/softwareengineering/std/ent03-b.html>，（参照 2020-06-12）．

小島　芳治　KOJIMA Yoshiharu, Ph.D.
東芝デジタルソリューションズ（株）ソフトウェアシステム技術開発センター システム・エンジニアリング開発部
博士（情報科学）
Toshiba Digital Solutions Corp.

斉藤　　稔　SAITO Minoru
東芝デジタルソリューションズ（株）ソフトウェアシステム技術開発センター システム・エンジニアリング開発部
Toshiba Digital Solutions Corp.