大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics

アドバンス・トップエスイー プロフェッショナルスタディ

ドメイン駆動設計に基づいたマイクロサービス・アーキテクチャの効果的なモデリング・設計手法

富士通株式会社　酒井 響平　sakai.kyohei@fujitsu.com

開発における問題点

クラウドコンピューティングやコンテナ技術の進展を受け、Microservices Architecture (MSA) という設計思想が注目されている. しかし, その適用の難しさに加え, 適用に際し開発プロセスや組織構造の改変を伴うことから, MSAの導入が躊躇されることが存在する.

手法・ツールの適用による解決

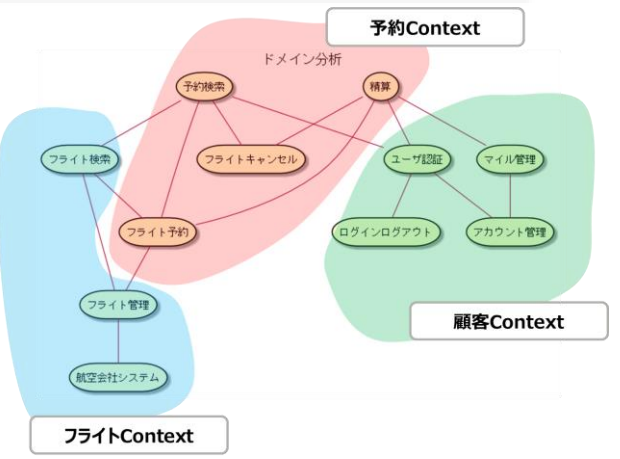
MSAのモデリング・設計に有効と言及されることの多いドメイン駆動設計 (DDD) を利用することでシステマチックなMSA設計プロセスを確立した. 本研究ではそのプロセスを適用することにより, MSAの利点をさらに受けることができるかをケーススタディを通じて評価した.

MSAモデリング・設計プロセス

今回扱ったプロセスの概要:
Microsoft社のAzureのドキュメントとして公開されたガイドライン
= DDDをベースにマイクロサービスを識別することに重点

1. ドメイン分析の実施
2. Bounded contextの定義
3. エンティティ・集約・ドメインサービスの定義
4. マイクロサービスの識別

このプロセスをよりシステマチックに実行するため, 相互作用分析等のプラクティスも本プロセスへと組み入れ, その補強を図った.



Bounded contextの作成例

評価

以下の2シナリオによってプロセスの評価を実施した.

①: コア機能を担うコンテナのプロセスが停止する (障害)
②: アプリロジックを修正して再デプロイする (アプリ改版)

①は障害発生時のシステムの振る舞いと修正作業を比較することを, ②は既存機能の修正・再デプロイ時の手番や作業中の可用性を比較することを目的とする.

ベンチマークアプリケーションに対して, 上記シナリオによる評価を実施し、次の結果が得られた.

- より細かいサービスへ分割するほどシステムの回復性は向上する一方, サービス間の通信のトレースが難しくなるなどのデメリットも生じる.
- DDDベースでMSAをを設計するとデプロイの容易性が向上し, リリーススピードを上げることに貢献する. ただし, アジャイル開発が前提となるため開発プロセスを改める必要性が生じる場合がある.

結論と今後の展望

◆ 結論
- DDDをベースとすることにより, MSAの利点(回復性・デプロイ容易性)をより享受することができた. ただし, 各利点にトレードオフ性が認められる場合があることに留意する必要がある.
- DDDをベースとすることでシステマチックにMSAを設計できた. 更に技術スタック選定をシステマチックに行うための知識ベースの構築が求められる.

◆ 今後の展望
- MSAの初心者・熟練者によるプロセスの第三者評価
- 大規模システムの実運用を想定とした評価
- 技術スタック選定のための横断型の知識ベースの構築