複雑化した業務アプリケーションの分割と可視化

秦野 智臣[1,a)]　上村 学[1]　松尾 昭彦[1]

概要：本稿では，複雑化した業務アプリケーションを，プログラムとデータから成る複数の集合に分割する技術について紹介する．本技術では，集合間の依存関係が互いに少なくなるように分割されるため，その結果を可視化することで，既存アプリケーションの再設計に利用できる．

1. はじめに

企業で利用される業務アプリケーションは，ビジネス環境や情報技術の変化に迅速に対応しなければならない．

マイクロサービスアーキテクチャは，一枚岩のアプリケーション（モノリス）を機能に従って複数の小さいサービス（マイクロサービス）に分割し，それらを連携させることで迅速なデプロイを実現するものである [1]．マイクロサービスアーキテクチャでは，各サービスの独立性を高め，サービスごとにデータベースを所有することで，互いに疎結合な設計を行う．このような設計を行うことで，あるサービスに対する変更の影響を局所化することが可能になり，変化に対応しやすいアプリケーションを実現できる．

しかし，既存のモノリスでは，数多くのプログラムとデータが複雑に関係しており，機能に従った疎結合なサービスを再設計することが困難である．設計者は，どのような機能がどのようなプログラム要素で構成されており，それらがどのように連携しているかを理解しなければならない．また，どのサービスがどのデータを所有すべきかを適切に判断する必要がある．

我々は，業務アプリケーションを，プログラムとデータ（データベーステーブル）から成る複数の集合（サービス部品と呼ぶ）に分割する技術を開発した．サービス部品は，疎結合なサービスの候補となる機能のまとまりであり，プログラムの呼び出し関係とデータベースへのアクセス関係に基づいて抽出される．本技術による分割は，マイクロサービスアーキテクチャのような変更の影響を局所化する設計を行うためのスタートラインとして利用することができる．

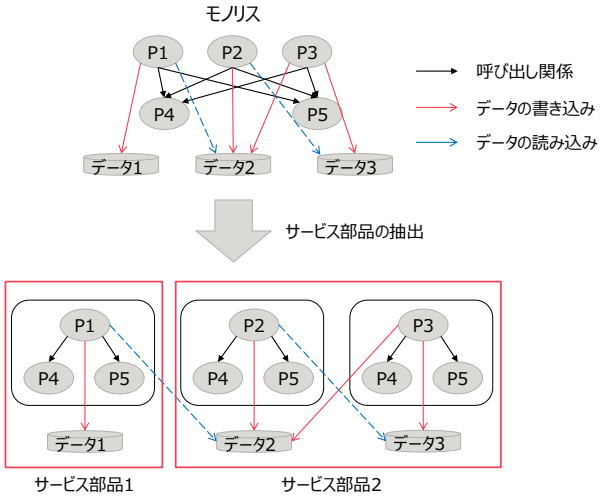

図 1　モノリスの分割

2. 開発技術

本技術では，アプリケーションの各機能を実現するために必要なプログラムとデータベーステーブルを特定し，それらをサービス部品として抽出する．各サービス部品が独立して動作するように，複数の機能から利用されるプログラムはサービス部品間に重複させてデプロイする．一方で，データを重複させると整合性の管理が困難になるため，特定のサービス部品に所有させる．以降では，COBOL で実装された業務アプリケーションを例として，サービス部品の具体的な抽出手順について説明する．

2.1 プログラム集合の特定

アプリケーションが提供する機能の起点となるプログラムを選定し，そのプログラムから呼び出されるすべてのプログラムをプログラム集合とする．プログラム集合は，他のプログラム集合に依存せずに単独で実行可能な単位であるため，あるプログラムが複数のプログラム集合に含まれ

1　株式会社富士通研究所
a)　hatano.tomomi@jp.fujitsu.com

る場合もある．本例では，COBOL ソースファイルをプログラムとし，分析対象内のどのファイルからも呼び出されていないファイルを起点プログラムとして選定する．このようなファイルは，アプリケーションの画面や JCL などのバッチから呼び出されるものである．

図 1 は，5 つのプログラム（P1 から P5）と 3 つのデータから成るアプリケーションの例を示している．この例では，P1，P2，P3 がそれぞれ起点プログラムとして選定され，3 つのプログラム集合が作られる．

2.2 サービス部品の抽出

各プログラム集合とデータベーステーブル間のアクセス関係を取得し，クラスタリングアルゴリズム [2] を用いて関係の強いプログラム集合とテーブルを集める．テーブルへのアクセス関係は読み込みと書き込みを区別し，書き込みがより強い関係であると考え，書き込みの重みを強くする．我々の適用事例では，書き込みの重みを読み込みの 2 倍に設定している．このクラスタリングによって，プログラム集合とテーブルから成るクラスタが抽出されるが，複数のクラスタから書き込まれるテーブルは汎用的なデータを保持している可能性が高いと考えられるため，このようなテーブルは特定のプログラム集合に配置せず，それ単独で 1 つのクラスタとする（これを共通データと呼ぶ）．そして，共通データと共通データへのアクセス関係を除外した状態で再度クラスタリングアルゴリズムを適用し，形成されたクラスタをサービス部品とする．

図 1 の下部は，サービス部品抽出の例を示している．P1 を起点とするプログラム集合とデータ 1 が 1 つのサービス部品を構成している．また，P2 と P3 をそれぞれ起点プログラム集合は，データ 2 とデータ 3 へのアクセス関係が強いため，1 つのサービス部品としてまとめられている．

3. 分割結果の利用

本技術による分割結果を可視化することで，アプリケーションの再設計に利用できる．**図 2** は，ある業務アプリケーションを対象に本技術を適用した結果を可視化したものである．図中の小さな黒い四角（ビル）が 1 つのデータベーステーブルを表しており，黒以外のビルがプログラム集合を表している．また，赤色の枠で囲まれた部分はサービス部品であり，枠で囲まれていない黒いビルは共通データである．ビル間の線はプログラム集合とテーブル間のアクセス関係を示している．

図 2 のような分割結果を分析することで，再設計に向けた様々な検討が考えられる．たとえば，共通データに対するアクセスが集中している場合は，それに対応するデータアクセス API を用意する工数が発生すると予想できる．また，データアクセスの関係線が少ないサービス部品は，実際に抽出することが容易であると考えられるため，そのようなサービス部品から再構築するという計画を立てることができる．一方で，関係線が多いサービス部品や構成要素が多いサービス部品は，サービスとして複雑すぎる可能性があるため，細分化の検討が必要であると考えられる．

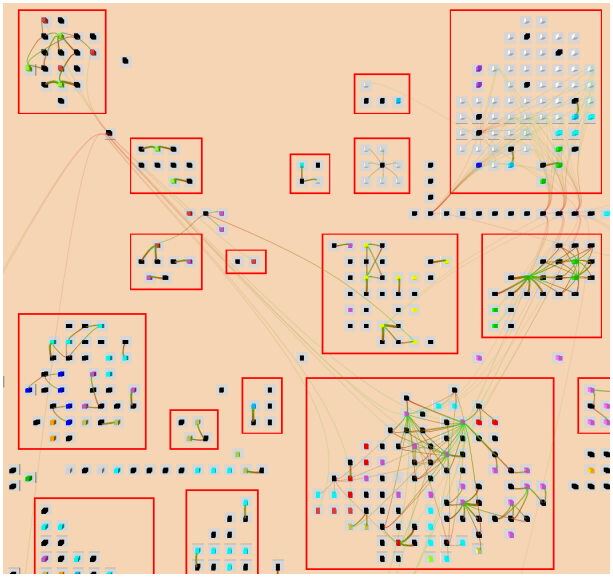

図 2 本技術の適用事例

4. 実用化に向けた課題

本ワークショップでは，本技術の実用化に向けた課題について議論する．たとえば，我々は以下のような点が課題になると考えている．

- 各サービス部品が具体的にどのような処理を行うものであるかを効率的に理解する方法．
- プログラミング言語による差異に対応する方法．COBOL のアプリケーションで実践した方法が，Java などの他言語には流用できない可能性がある．
- ソースコード以外の有益な情報がある場合に，それをどのように入力させ，分割にどう反映させるか．ドキュメントや過去の修正履歴だけでなく，事業戦略を考慮して設計を行う必要もある．

参考文献

[1] Newman, S.: 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). 佐藤 直生監修，木下 哲也訳.

[2] Kobayashi, K., Kamimura, M., Kato, K., Yano, K. and Matsuo, A.: Feature-gathering dependency-based software clustering using Dedication and Modularity, *Proceedings of the 28th IEEE International Conference on Software Maintenance*, pp. 462–471 (2012).