業務とアーキテクチャの依存関係分析に基づく Web システムの再設計手法の提案

M2020SE004　小澤 司
指導教員　沢田 篤史

1　はじめに

モノリシックシステムの維持では，単一のコンポーネントゆえの問題が大きくなり，それらの解決手段としてマイクロサービスアーキテクチャ（以下，MSA）への移行事例が増加している．MSA への移行に着手する企業は，巨大な単一コンポーネントを持つモノリシックシステムを，維持管理に適した大きさのサービスに分割することで，ビジネスのアジリティやケイパビリティの加速を期待する．

一方，企画段階においては，MSA を適用する利点と，難易度を評価するための検討プロセスが確立しておらず，計画者は見定めが難しい問題を抱えている．特に，移行後の安定した品質を担保するには，適切なコンポーネント分割に向けて調査するべき観点が不明瞭なので，依存関係の調査に多くの人員と期間を要する．それにより，計画者がそれらを確保できず検討を断念したり，投資判断に足る十分な仮説を用意できないことが起きる．

もう一つの問題として，効率的なシステム維持に向けては，分割後の複数のシステムを単一組織で維持するために，横断的関心事を独立して取り扱うことが理想であるが，再設計の方法は少ない．

本研究では，一般的な MVVM アーキテクチャ[1] を対象とし，この 2 つの問題点の解決に向けてモノリシックシステムから MSA への移行に向けた明確な検討プロセスを提案し，移行後の品質について形式的に見定めることを目的とする．

2　マイクロサービスアーキテクチャへの再設計とその問題点

2.1　先行研究

MSA 適用に向けた検討プロセスについては，いくつかの先行研究で提案されている．それらの調査結果を表 1 に示す．

Levcovitz ら [2] の提案では，不適切なコンポーネントの分割を防ぐための依存関係分析を提案している．この提案方法では，MSA の候補の定義に詳細な分析が必要になることから，多くの調査工数が必要となり，企画段階においての適用が難しい点は課題である．横断する品質要求に対して，言及している研究はないこともわかる．

2.2　横断的関心事に関する関連研究

Ossher ら [6] が提案した関心事の多次元分離（Multidimensional separation of concern（以下 MDSOC））は，品質に関する要求（関心事）のモジュール化に有効であると言われている．MDSOC の概念は，ハイパースライスが関心事に相当し，既存システムにおいて抽象化した複数のディメンジョン上にあるハイパースライスを，モジュール化し，再利用する．モノリシックシステムの MSA 適用に向けた再設計においても，横断する関心事を抽出し，モジュールにできると考える．

表 1　先行研究の調査結果（抜粋）

番号	業務分析精度	分割精度	作業負荷	横断的関心事の対処
[3]	○:DFD 作成	×：記載なし	×：負荷が高い	×：記載なし
[2]	△：依存関係分析	○：詳細分析	×：詳細分析必要	×:記載なし
[4]	×：記載なし	△：変更履歴に依存	○:MSA 抽出を自動化	×：記載なし

2.3　研究課題

本稿では，先行研究を踏まえ以下の 3 点を研究課題として設定する．

RQ1 MVVM アーキテクチャを前提とした，MSA への移行に利用できる依存関係情報は何かを明らかにする．

RQ2 サービスをまたぐ品質要求に対する処理機能を独立したモジュールにする方法を明らかにする．

RQ3 RQ1，RQ2 の結果を統合した分析手法を確立し，移行計画段階における効果予測に対する有効性を評価する．

3　ビジネスおよびシステムの依存関係分析を用いた再設計手法の提案

3.1　本研究の全体像

図 1 に本研究の全体像を示す．図の再構築手法プロセスは情報処理推進機構（IPA）の「システム再構築を成功に導くユーザガイド第 2 版」[10] を引用している．

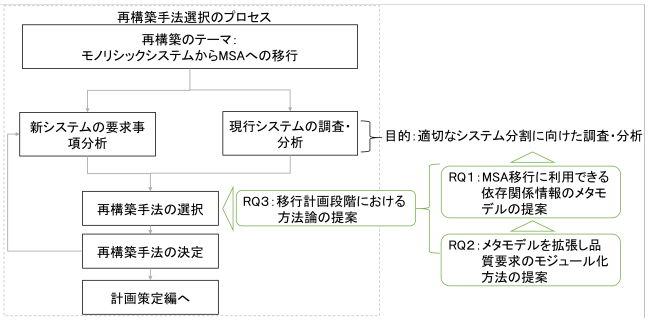

図 1　再構築手法選択のプロセスと本研究の全体像

3.2　利用できる依存関係情報の抽出に向けて

RQ1 では，MSA への移行に利用できる依存関係情報をソフトウェアアーキテクチャ（以下，アーキテクチャ）とデプロイメントの観点からメタモデルを用いて明らかにする．2 つの観点とする理由は，Clements ら [7] が，アーキテクチャを文書化する共通の枠組み（Views and Beyond）でまとめており，モジュールおよびコンポーネントとコネクタはアーキテクチャ，アロケーションはデプロイメントに相当する．

本研究の対象であるモノリシックシステムは，コンポーネントが単一デプロイメントに集約されていることが多いため，アーキテクチャの観点から依存関係分析を行う．ただし，システム分割および MSA 適用に向けての再設計では，デプロイメントの観点が必要となる．

3.3　横断する品質要求のモジュール化に向けて

RQ2 では，図 2 に品質要求に対する処理機能を独立したモジュールにする方法を示す．図 2 の 2 の横断的関心事の抽出と，3 のモジュール化検討では，それぞれアーキテクチャおよびデプロイメントの構成要素と品質要求，およびモジュール化の対応表を用いる．そのため，RQ1 のメタモデルを拡張し各構成要素と品質要求の対応表を作成する．対応表とコンポーネント分析結果を組み合わせ，横断する品質要求の抽出およびモジュール化を行う．

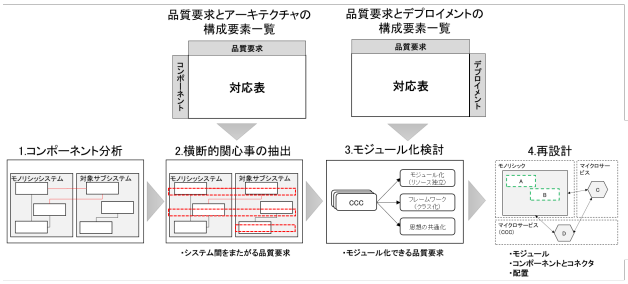

図 2　品質要求のモジュール化に向けてのプロセス

3.4　依存関係情報のメタモデル

3.4.1　アーキテクチャのメタモデルと品質要求の関係

アーキテクチャに対するメタモデルおよび品質要求との関係を，図 3 と表 2 に示す．

図 3 は業務とアーキテクチャおよび品質要求の間の依存関係を示したメタモデルである．Levcovitz ら [2] は，MSA の分析に必要な依存関係の構成要素はファサード，ビジネスファンクション，データの 3 つの層としている．ファサードはフロントエンドとバックエンドが分かれている場合に用いるが，本研究では一般的なモノリシックシステムに適用するため，ユースケース（Usecase），ファンクション（Function），データ（Data）として層別する．

表 2 はアーキテクチャの構成要素と品質要求との対応表を抜粋したものである．品質要求の各項目については，SQuaRE[9] における製品品質特性および副特性と情報処理推進機構（IPA）の非機能要求グレード [8] より，一般的な WEB サービスを想定した場合の項目を洗い出した．

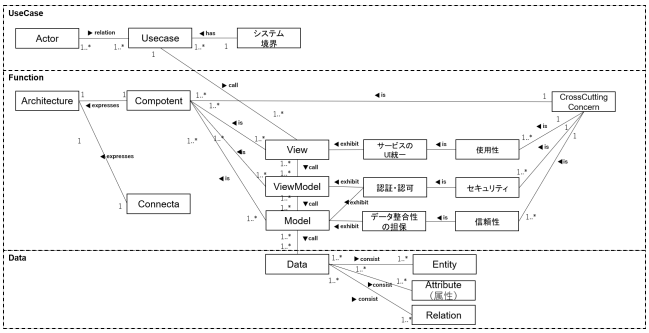

図 3　業務とアーキテクチャメタモデル

表 2　品質要求とアーキテクチャの構成要素一覧（抜粋）

品質要求	品質副特性	コンポーネント	機能例
サービスの UI 統一	ユーザインターフェイス快美性	V，VM	UI の統一
認証・認可	セキュリティ	VM，M	認証・認可機能
データ整合性の担保	可用性	VM，M	強整合性，結果整合性

3.4.2　デプロイメントのメタモデルと品質要求の関係

デプロイメントに対するメタモデルを図 4 に示す．アーキテクチャと同様，ユースケース，ファンクションデータの 3 層で表している．品質要求との対応表についてもアーキテクチャと同様に作成している．

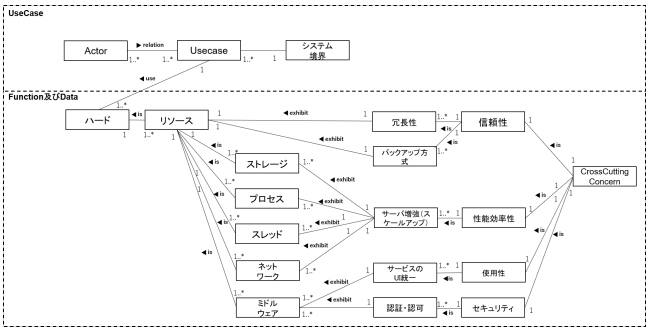

図 4　業務とデプロイメントメタモデル

3.4.3　モジュール化の可能な品質要求

システムをまたがる品質要求のモジュール化の対応表を表 3 に示す．全ての品質要求に対してモジュール化が難しいことから，「モジュール化」，「フレームワーク共通化」，「ガイドライン共通化」から選択することを想定する．

例えば，「サービスの UI 統一」においては，モジュール化が可能な品質要求である．システムを分割する際に，View の部分をモジュール化することで，横断的関心事を実現することができる．場合によってフレームワーク共通化およびガイドライン共通化も選択できる．

表 3 各品質要求におけるモジュール化の可否（抜粋）

品質要求	モジュール化	フレームワーク共通化	ガイドライン共通化
サービスの UI 統一	○	○	○
認証・認可	○	○	○
バックアップ方式	-	-	○

3.5 業務およびシステムの依存関係分析を用いた再設計手法

RQ3 に対するアプローチとして，MSA 移行のための検討プロセスを図 5 に示す.

3 つの手順を定義しており，手順 2 の完了時に MSA 化する対象を選定し，手順 3 で再設計のための詳細分析を行う．手順を分けることで，システム全体から詳細へ段階的に進めることができる.

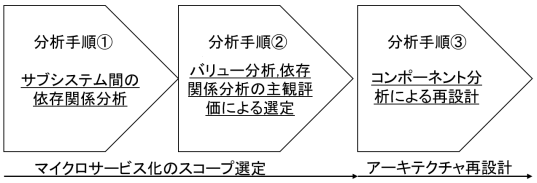

図 5 提案プロセス

手順 1 では，モノリシックシステムをサブシステムに分割し，RQ1 で示したユースケース，ファンクション，データの 3 層から，依存関係の強さを分析する．サブシステム間の依存関係の強さを定量化することで，各サブシステムの MSA 適用の難易度を測る.

手順 2 では，バリュー分析を行うことで，各サブシステムの MSA 適用に向けた価値（動機）を評価する．バリュー分析の結果と手順 1 の依存関係の強さを総合評価し，MSA 適用対象を選定する．本研究では，バリュー分析の評価観点を，Newman の著書 [5] の MSA 適用の 7 つの動機を,「変更頻度」,「利用頻度」,「再利用性」,「SLA の差異」,「技術的特異性」の 5 つに分類して利用する.

手順 3 では，RQ2 で示した図 2 の手順で進め，MSA 適用対象のサブシステム内およびモノリシックシステムとの間のコンポーネントの依存関係分析および品質要求のモジュール化を含めた再設計を行う.

4 実システムへの適用

4.1 対象システム

3 章で提案した分析手法について，会員制 WEB サービスに適用し評価する．一般的な MVVM を採用しており，表 4 の機能を持つ．サービスには，80 画面があり，各サブシステムに 5〜30 画面が属する.

4.2 分析手法の適用

以下に，MSA 適用に向けての 3 つの手順を適用した内容を詳述する.

表 4 評価システムの機能一覧 (抜粋)

機能	概要
会員管理	会員登録および変更を行う
問合せ	会員の問い合わせを受け付ける
コンテンツ管理	会員にコンテンツを提供する
アンケート	会員にアンケートを実施する
料金照会	利用料金を照会する

4.2.1 手順 1:サブシステム間の依存関係分析

サブシステム間の依存関係の強さを定量化した結果を表 5 に示す.「業務の影響度」は，各ユースケースと利用するファンクション（画面）数であり,「システムの影響度」は各ファンクションが参照または更新するデータ（テーブル）数である．ユースケースとファンクションの依存関係は全体的に弱く，ファンクションとデータの依存関係は,「会員管理」および「料金照会」が強い.

表 5 サブシステム間の依存関係分析結果（抜粋）

サブシステム	業務の影響度		システムの影響	
	被参照	参照	被参照	参照
会員管理	2	0	65	69
問合せ	0	0	1	2
コンテンツ管理	2	3	0	14
アンケート	3	5	1	3
料金照会	1	1	48	28

4.2.2 手順 2:バリュー分析を用いた対象の特定

バリュー分析と手順 1 を組み合わせた結果，移行の対象として MSA 適用のバリュー（価値）が高い「料金照会」を選択した．他のサブシステムとの依存関係が強いものの，参照機能が多いことから影響が少ないと判断した.

4.2.3 手順 3:コンポーネント分析および再設計

サービスとしては,「料金照会サービス」,「高負荷バッチサービス」,「管理・ログ管理サービス」の 3 つを定義した．必要な Web-API の開発数は最大 584 である.「高負荷バッチサービス」および「管理・ログ管理サービス」,「フロントエンド UI」,「セキュア対策（Web Application Firewall)」について，モジュールにしている.

5 評価および考察

5.1 評価

提案手法を評価するために，以下の 2 つの方法を用いる.

- 4 章の結果に対して，品質機能展開表（以下，QFD）を用いて，モジュール化の有効性を確認する.
- 先行研究と比較することで，計画者にとって，見定めが難しい問題の解決を行っているか確認する.

QFD の結果を図 6 に示す．モジュール化によって，機能を横断する品質要求への対応ができていることを確認

した．ただし，対象システムの状況によっては，モジュール化の対象とならない品質要求があることも確認した．

要求品質展開表		信頼性					使用性	性能効率性			セキュリティ	
		障害時許容性		可用性			ユーザインターフェイス快美性	性能・拡張性			機密性	否認防止性
		障害時の閉塞機能	障害の検知	エラー時処理	データの整合性担保	分散トランザクション対応等	サービスのUI統一	オンラインレスポンス	バッチレスポンス	リソース拡張性	認証・認可	WAFの導入の有無
モノリシック	会員登録をする	△	◎	△	◎		◎	△		△	◎	◎
モノリシック	属性情報変更する	△	◎	△	◎		◎	△		△	◎	◎
モノリシック	メルマガを送る		◎	△								
料金照会	料金照会を確認する	△	◎	△	◎		◎	△		△	◎	◎
高負荷バッチ処理	料金に応じたポイントをもらう	△	◎	△	◎		◎		◎	△	◎	◎
監視ログ管理	エラーの検知を行う		◎	△								

◎:モジュール化　○:フレームワーク共通化　△:ガイドライン共通化　空白:対象システムでは該当せず

図 6　QFD の評価（抜粋）

2.1 節の表 1 と同じ観点での評価結果を表 6 に示す．特に，Levcovitz ら [2] の研究と比較した場合，コンポーネント間の詳細分析前に，依存関係分析結果を主観評価することで，効率的に MSA の選定ができる．これは企画段階の調査に人員と期間を要する課題の解決に寄与する．

一方，Levcovitz ら [2] の手法は，ファサード，ビジネスファンクション，データの依存関係の詳細を調べた上で，MSA を定義していることから，MSA の粒度を小さくする点においては，本手法より優位性が高い．

表 6　先行研究調査の観点における本研究の評価

業務分析精度	分割精度	作業負荷	横断的関心事の対処
△：依存関係分析	○：MSA 選定後詳細分析	△：主観評価	○：対処

5.2　考察

本研究の検討プロセスは，ビジネスとシステムの依存関係分析および主観評価によって，企画段階の調査の効率化に貢献できる．あらかじめサブシステムを定義する必要があるが，日本企業では，初期構築時に定義することが多いため，評価システム以外への適用も可能である．また，横断する品質要求の対処を含めて再設計ができる．

一方，依存関係の強さの定量化が評価者の主観に依存している点から，依存関係の強さの値が大きく移行の判断ができない状況に陥る可能性がある．それらの解決には，方法の追加が必要である．加えて，更新機能の多いシステムの評価はしていない．

以上まとめると課題は以下の 2 点である．

- 更新機能が多く複雑なシステムへの適用
- 依存関係の強さの値が大きく移行の判断ができない場合の方法の追加および検証

6　おわりに

本研究では，一般的な MVVM アーキテクチャのシステムに向けて，MSA 適用の検討プロセスを提案した．企画段階に対して依存関係分析による主観評価と，横断的関心事を独立して取り扱う再設計手法として有効である．

今後の課題は 2 つである．1 つ目は，本検証は参照機能が多い比較的簡易なシステムで検証していることから，更新機能が多く複雑な実システムで検証を行う必要がある．

2 つ目は，本手法では，依存関係の評価が大きく MSA の適用判断ができない時の解決策の提示ができていない．例えば原因が，特定のサブシステム間の依存関係のみ強い場合，1 つのサブシステムにまとめる必要もあり，グラフの活用も有効であると考えられる．これらの方法の追加には有効性および妥当性の検証が必要である．

参考文献

[1] J. Smith, “WPF Apps with the Model-View-ViewModel Design Pattern”, https://docs.microsoft.com/en-us/archive/msdn-magazine/2009/february/patterns-wpf-apps-with-the-model-view-viewmodel-design-pattern. (accessed 2022.1.20)

[2] A. Levcovitz, R. Terra, M.T. Valente, “Towards a Technique for Extracting Microservices from Monolithic Enterprise Systems”, Proc. of the 3rd Brazilian Workshop on Software Visualization, Evolution and Maintenance (VEM), pp. 97–104, 2015.

[3] R. Chen, S. Li, Z. Li, “From Monolith to Microservices: A Dataflow-Driven Approach”, Proc. of the 24th Asia-Pacific Software Engineering Conference, pp. 466-475, 2017.

[4] G. Mazlami, J. Cito, P. Leitner, “Extraction of Microservices from Monolithic Software Architectures”, IEEE 24th International Conference on Web Services, pp. 524-531, 2017.

[5] S. Newman，佐藤直生（監訳），木下哲也（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.

[6] H. Ossher, P. Tarr, “Multi-Dimensional Separation of Concerns in Hyperspace”. Technical Report RC 21452 (96717) 16APR99, IBM Thomas J. Watson Research Center, Yorktown Heights, NY., 1999.

[7] P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, P. Merson, R. Nord, J. Stafford. Documenting Software Architectures: Views and Beyond, Second Edition, Addison-Wesley, 2010.

[8] 独立行政法人情報処理推進機構（IPA），非機能要求グレード本体（日本語版），https://www.ipa.go.jp/sec/softwareengineering/std/ent03-b.html.（accessed 2022.1.20）

[9] 日本規格協会グループ，ISO/IEC 25010:2011 SQuaRE，2013-06-20,

[10] 独立行政法人情報処理推進機構（IPA/SEC），システム再構築を成功に導くユーザガイド第 2 版，2018，https://www.ipa.go.jp/files/000057294.pdf.（accessed 2022.1.20）