サーバレスアーキテクチャーの適用によるシステム開発期間短縮効果の検証

－ビジネスアジリティは向上するのか－

IA−001：サーバレスアーキテクチャーのビジネス適用：石野 大輔

Verifing the effectiveness in reducing development period on serverless architecture

－　Can business agility be improved　－

IA-001：Applying Serverless Architecture to Business：Daisuke Ishino

新たなデジタル技術が日々開発されている状況の中，企業は競争力強化のため，新技術を活用してビジネスモデルを素早く創出することが求められている．近年，ビジネスアジリティを向上させるアーキテクチャーとしてサーバレスアーキテクチャーが注目されている．本論文では，サーバレスアーキテクチャーの適用により，ビジネスアジリティが向上するのか検証した．検証で得られた結果から，ビジネスアジリティ向上の是非と今後の課題・展望について述べる．

While new digital technologies are applied every day, companies are required to create business models quickly by utilizing these new technologies to enhance their competitiveness. In recent years, serverless architecture has been attracting attention as an architecture to improve business agility. In this paper, we examine whether business agility can be improved by applying a serverless architecture. This paper discusses the pros and cons of the improvement of business agility and future issues and prospects based on the results of the verification．

Key Words & Phrases ：デジタルトランスフォーメーション，サーバレスアーキテクチャー，ビジネスアジリティ，開発期間の短縮，ドキュメント作成量の削減

Digital Transformation, Serverless Architecture, improvement of business agility, shortened development period, Reducing the amount of documentation

１．はじめに

日々新たなデジタル技術が開発されている昨今，IT業界のみならず様々な業種でデジタル技術を駆使したビジネスが創出され企業価値を高めている．こうした中で，各企業は競争力維持・強化のために，デジタルトランスフォーメーション（DX：Digital Transformation）をスピーディに進めていくことが求められている．経済産業省の「DXデジタルトランスフォーメーションレポート」[1]によれば，我が国企業においては，多くの経営者が DX の必要性を認識し，DX を進めるべく，デジタル部門を設置する等の取組が見られる．しかしながら，PoCを繰り返す等，ある程度の投資は行われるものの実際のビジネス変革には繋がっていないという状況が多くの企業に見られる現状にある．既存システムが老朽化・複雑化・ブラックボックス化する中では，データを十分に活用しきれず，新しいデジタル技術を導入したとしても，データの利活用・連携が限定的であるため，その効果も限定的となってしまうという問題が指摘されている．老朽化・複雑化した既存システムがこのまま継続して使用された場合，2025年～2030年の間に最大12兆円の経済損失が生じると推定されている．

既存の IT システムを巡る問題を解消しない限りは，DXを本格的に展開することは困難であると考えられる．しかし既存のITシステムの改修は大規模かつ長期のプロジェクトとなるため，スピーディに効果を得ることができない．そこで，頻繁に改修が求められるビジネスロジック又は新規に作成するビジネスロジックについて，新たなデジタル技術を使用して構築することで，ビジネスモデルの変化に迅速に追従できるようにする必要がある．

近年，ビジネスアジリティを向上させるアーキテクチャーとしてサーバレスアーキテクチャーが注目されている．サーバレスアーキテクチャーは，クラウド事業者の責任範囲が広く，ユーザーは業務ロジック構築とアプリケーション開発に注力できる点が大きな特徴である．本論文では，サーバレスアーキテクチャーの適用により実際にモデルシステムを設計・構築することでビジネスアジリティが向上するのかについて検証を行う．なお，本論文におけるビジネスアジリティの向上とは，サービスリリースまでの開発期間が短縮されることで，迅速にサービス提供可能となることを指す．以下，2章でサーバレスアーキテクチャーの特徴を述べ，3章，4章にてITシステム構築時のドキュメント作成量及び構築作業時間の2つの観点での検証結果について述べる．また，5章，6章にて検証作業にて確認された課題と課題を踏まえた今後の展望を述べる．

２．サーバレスアーキテクチャーとは

　本論文では「サーバレス」，「サーバレスアーキテクチャー」の2つの用語は，2019年度JGS論文「サーバレスアーキテクチャーの適用検討プロセスの提案」[2]の定義と同意とする．サーバレスとは，「サーバ管理を必要としないアプリケーションの構築と実行の概念」[2]のことであり，代表的な実現方法としてFaaS（Function as a Service）とBaaS（Backend as a Service）がある．これらFaaSとBaaSの機能を組み合わせたシステム構成がサーバレスアーキテクチャーである[2]．

2.1 サーバレスの特徴

　サーバレスには， クラウド事業者ごとに様々な特徴が存在するが，ここでは代表的な以下2

点の特徴について述べる．

1. サーバの構築・運用が不要

サーバやランタイム等のアプリケーションの実行に必要な基盤環境は，図1の通り，クラウド事業者の責任の下，運用・管理されている．そのためユーザーは，サーバ構築やミドルウェア設定などの環境構築を行う必要がなく，運用・管理に掛かる作業負荷も大幅に削減される．



図1．サーバレスの責任モデル

1. 柔軟なスケーラビリティ

処理量に応じた自動的なスケーリングが可能となるため，高負荷時を想定した過剰なリソースの確保を行う必要がない．また，予想外の負荷に対しても柔軟に対応でき，安定的な業務運営を行うことができる．

３．サーバレスアーキテクチャー適用によるビジネスアジリティ向上の検証

本章では，サーバレスアーキテクチャーを使用してグッズ購買サイトを構築し，オンプレミス環境における構築作業と比較することで，開発期間が短縮するか検証した結果を述べる．

検証範囲は，意思決定された事業戦略を，IT部門がサーバレスアーキテクチャーを適用して開発するシステム開発部分とする．

　開発期間の短縮は，次の2つの観点で検証する．

・ドキュメント作成量

・構築作業時間

3.1仮説と前提

　検証作業における仮説と前提を述べる．

3.1.1仮説

サーバレスアーキテクチャー適用により，各開発工程のドキュメント作成量の減少，構築作業時間の短縮により，開発期間が短縮することを仮説とする．

3.1.2 前提

　検証作業における前提を述べる．

1. 利用するクラウドベンダー

本検証では，代表的なクラウド事業者[3]であるAmazon Web Services（以下AWS）が提供するサーバレスサービスを使用して，グッズ購買サイトのシステム構築を行う．

1. モデルシステム選択理由

　政府がキャッシュレス決済を推進しており，ECサイトの立ち上げが増加すると考えたためである．「経済産業省のキャッシュレスの現状及び意義」[4]では，キャッシュレス決済推進のメリットとして「店舗の効率化・売上拡大」，「データ利活用」が挙げられている．実店舗を持つ企業が，ECサイトを立ち上げる場合，販売チャネルの多角化による売上の拡大が期待できる．また，サイト内の購買情報のデータを分析・活用することで，マーケティングや商品開発に繋げることが可能になる．ECサイトはキャッシュレス決済が主な決済手段であり，キャッシュレス決済推進によるメリットを享受できることから，ECサイトが増加すると考える．

1. グッズ購買サイト構築

グッズ購買サイトに必要な要求事項を要求仕様書（別冊A-1）として纏め，要求を充足するシステムを構築する．適用するサーバレスアーキテクチャーは，AWSにて品質が保証されているAWS samples掲載のアーキテクチャー設計から，検証要件に類似した要件のServerless Airline Booking（航空券購買システム例）[4] をリファレンスアーキテクチャーとして再利用し本システムの要求事項を満たすシステム構成を設計した．

　　（図２）

スクリーンショットの画面

自動的に生成された説明

図2．グッズ購買サイトシステム構成図

3.2検証方法

構築における作業時間及びドキュメント作成量の測定はそれぞれ以下の手順で行った．

3.2.1構築作業時間

次のA-1からA-4の手順で構築作業を行った．なお，構築作業時間の測定は構築作業そのものである手順A-3と手順A-4に要した時間を計測した．

手順A-1：要求仕様書を基に，採用するAWSサービスで必要な設計や設定について，AWS開発者ガイドやガイド記載のチュートリアルを参照し，事前学習を行う．

手順A-2：確認結果を整理し，システムで扱うデータ定義をデータ定義書（別冊B-1）として纏める．AWSマネジメントコンソール作業で設定する値を，グッズ購買サイトで実装する機能ごとに，パラメータシート（別冊B-2）として纏める．

手順A-3：Lambdaがサポートするランタイムを利用して，グッズ購買システムのアプリケーションソースコードを開発する．コード開発は，メンバーで分担して開発する．

手順A-4：手順1で使用した開発者ガイドやチュートリアルの手順をベースに，準備したデータ定義書，パラメータシートに従い，AWSマネジメントコンソール，AWS CLIを利用してグッズ購買システムを構築し，作業時間を計測する．構築作業は，AppSyncクエリからStep Functionsのステートマシンを起動するのに必要な認証機能の実装を除き，AWSを利用した開発の未経験者であるメンバー1名が担当する．

3.2.2ドキュメント作成量

次のB-1からB-4の手順でドキュメント作成量の測定を行った．

手順B-1：オンプレミス環境でウォーターフォールモデルによる開発を行う場合に必要となる成果物一覧を作成する．なお，成果物一覧は，メンバーが所属する各社で標準化された成果物一覧を纏めたものである．これまでのシステム構築における知見に基づいているため成果物の網羅性は担保されていると考える．

手順B-2：3.2.1構築作業後に，手順B-1で作成した成果物一覧に対して，サーバレスアーキテクチャー適用による開発で必要となる成果物を設定した．

手順B-3：オンプレミス開発におけるドキュメント作成時の作業量を基準として，同じドキュメントをサーバレスアーキテクチャーの適用による開発を行った場合と比較，評価する．評価は次の3段階とする．

・作業量が同じ

・作業量が減少する

・ドキュメント作成が不要となる

手順B-4：手順B-3にて「作業量が減少する」，「ドキュメント作成が不要」と評価したドキュメントついて，サーバレスアーキテクチャー適用時のメリットとして根拠を記載する．

上述の手順B-1～手順B-4を実施した結果を評価シート（別冊C-1）として纏める．評価シートには，3．2．1構築作業時に顕在化した課題についても記載する．

3.3検証結果

構築作業時間及びドキュメント作業量の計測結果についてそれぞれ以下に述べる．

3.3.1 構築作業時間

グッズ購買システムの構築作業時間（3.2.1の手順A-3，手順A-4の合計）は10時間であった．内訳は，手順A-3：アプリケーションソースコードの開発時間がのべ7時間，手順A-4：AWSマネジメントコンソール，AWS CLIを利用した構築作業時間が3時間であった．

3.3.2ドキュメント作成量

　図3の通り，オンプレミス環境でウォーターフォールモデルによる開発を行う場合に必要なドキュメントは68本，サーバレスアーキテクチャーの適用による開発で必要なドキュメントは56本であった．サーバレスアーキテクチャーの開発ではオンプレミス開発との比較でドキュメント作成量が18％減少した．

オンプレミス開発におけるドキュメント作成量を基準として，同じドキュメントをサーバレスアーキテクチャーの適用による開発を行った場合の作業量は以下の通りであった．

・オンプレミス開発時と作業量が変わらないドキュメント：38本

・オンプレミス開発時より作業量が減少するドキュメント：18本（オンプレミス開発時に作成するドキュメントのうち26％）

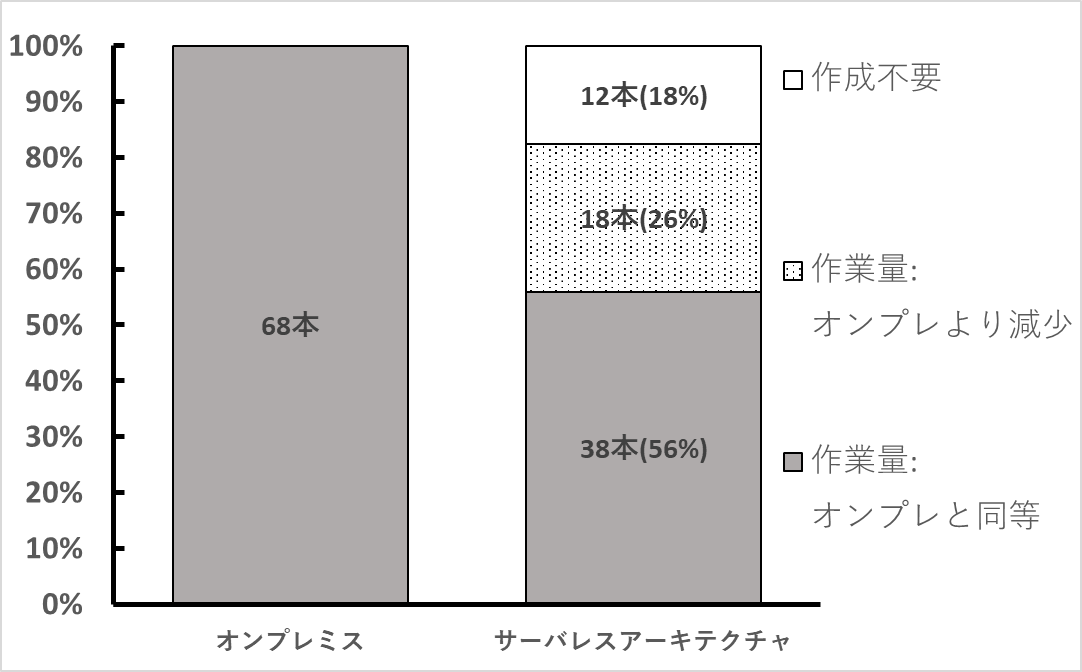


図3．ドキュメント作成量検証結果

４．考察

前述の検証結果から，サーバレスアーキテクチャーを適用した開発では，ドキュメント作成量及び構築作業時間が削減されることを確認できた．その理由は，非機能要求部分の実装が自動化されていることである．構築作業時間の削減については，関数の実現する機能が簡素であることも理由として挙げられる．それぞれの詳細について以下に述べる．また，サーバレスを適用してもオンプレミスと比較して開発期間が短縮されない点についても述べる．

4.1 非機能要求部分の実装自動化

　クラウドサービスの利用形態と図1．サーバレスの責任モデルを用いて説明する．

・IaaS（Infrastructure as a Service）

「仮想化」以下のインフラ部分は，クラウド事業者により高度に抽象化されており，利用者はインフラの導入・構築，管理が大幅に省力化される特徴がある．

・PaaS（Platform as a Service）

IaaSの特徴に加え，「ミドルウェア・ランタイム」が提供される．ベンダーが提供する仕様に依存するため，自由度や柔軟性に制限を受けるが，各種ミドルウェア・ランタイム導入・設定が省力化される特徴がある．また，拡張性，可用性を向上させる機能が提供されており， 設定も可能である．

・サーバレス

上述のIaaSやPaaSの特徴に加え，拡張性，可用性を向上させる機能がネイティブとして組み込まれている．利用者の設定に対する責任が低いため，設計量（ドキュメント作成量），構築作業量が削減される．

構築で利用したサービス（DynamoDB，Lambda）を例に，拡張性及び可用性が向上させる機能が組み込まれていることを確認する．

1. DynamoDB

・拡張性

DynamoDBはキャパシティユニットを増減させることで処理性能が決まる．キャパシティユニットとは，1秒あたりにN回の読み書きできる容量を指し，読み込み/書き込みの単位に設定可能である，ユーザーは，システムが要求するデータの読み込み頻度，書き込み頻度，データ項目のサイズ（読み込み/書き込み）に応じて，適切にキャパシティユニット数を設定する．またはAuto Scalingを利用可能である[6]．

・可用性

デフォルトで3つのアベイラビリティゾーンにデータが分散される．DynamoDBグローバルテーブル（マルチリージョンにマルチデータベースをデプロイ）の場合，サービスレベルアグリーメント（SLA）は99．999％以上となる[7]．

1. Lambda

・拡張性

　Lambda関数を処理するインスタンスは自動でスケーリングされ，並列で処理される[8]．

・可用性

　Lambdaは複数のアベイラビリティゾーンで関数を実行する．そのため１つのゾーンでサービスの中断が発生した場合にも，イベント処理の継続が保証されている[8]．

上記の通り，サーバレスには拡張性及び可用性を向上させる機能が組み込まれている．ユーザーは要求事項に合わせて，サーバレスの各サービスが提供する機能（バックエンド接続，拡張性，バックアップ等）を選択し，設定値を決めれば良い．加えて，クラウド事業者で実行が担保されており，ユーザーの設計範囲が限定されるため，オンプレミス環境で開発時より非機能部分の構築作業時間とドキュメントの作成量が削減される．

4.2 関数が実現する機能の簡素化

　各Lambda関数が「目的が1つもしくは2つで，ステートレス」な設計となっていることである．関数で実現することが簡素なので，プログラムステップ数が少ない．そのため，コーディング時間が短くなり，かつテストもし易くなる．また，ステートレスであるため，セッション情報の管理を考慮する必要も無くなる．これらにより開発時間が削減される．

4.3 結論

　サーバレスアーキテクチャーを適用することで，サービスリリースまでの開発期間が短縮され，迅速にサービス提供可能になる．結果，ビジネスアジリティが向上すると言える．

4.4 サーバレスアーキテクチャー適用により作業量が減少しない作業について

　本検証では，業務ロジック及びアプリケーション開発に対しては，開発期間の短縮効果は確認できなかった．2章で述べた通り，サーバレスとは「サーバ管理を必要としないアプリケーションの構築と実行の概念」である．業務要件定義「対象業務の現状を分析し，新たに実現すべき業務の流れを明確化する」や，アプリケーションの構造・設計，利用するデータの設計はユーザーの責任範囲である．

５．サーバレスアーキテクチャー適用における課題

今回の検証結果より，次のような課題が認められた．

5.1 開発言語の選択

Lambda関数の開発時に，開発者の得意な言語をそれぞれの関数で選定したため，複数言語が混在したシステムとなった．適切な言語を選択する自由度が増す反面，可読性が低くなるため，開発工程前に言語を選定するのが望ましい．

5.2 クラウド人材の育成

AWSを利用した開発の未経験者にて本検証を行ったため，構築作業前の事前学習に延べ1ヶ月程度費やした（AWSの基本概念，各サービス仕様等）．AWSサービスは日進月歩で進化しており，エンタープライズレベルの開発を行うためには，現場のエンジニアは当然，意思決定権のある経営者においても知識の習得が必要になる．しかし，有識者が存在していたとしても，各部門・領域に点在している状態では，組織としてエンタープライズレベルの開発を推進していくことは難しい．そのため，クラウドを用いた開発を進めていくためには，ノウハウを集約した専門組織を設立し，領域横断的な人材育成を行うことが望ましい．

６．今後の展望

サーバレスアーキテクチャーを適用することで，開発時に作成するドキュメント量と構築作業量が削減され，ビジネスアジリティが向上することが確認された．加えて，今回検証できていない以下の2つの観点を追求することで，更にビジネスアジリティを向上させる可能性があると考える．

1つ目は，開発業務への注力である．サーバレスアーキテクチャーを適用し運用業務を省力化することで，運用に割いていたコスト・人員を開発業務に充てられるため，よりスピーディな開発が可能になると考える．

2つ目は，各サービスの疎結合化である．各アプリケーションをマイクロサービスとして独立させ，機能追加・仕様変更時の際も対応範囲を局所化することで，低コスト・短納期で開発が可能であると考える．また上記の観点に加え，ドメイン駆動設計やアジャイル開発を組み合わせることで，より大きな効果が期待される．

ただし，上記の2点を実現させるためには，開発と運用の一体的体制やアジャイル開発を推進する組織風土の醸成に加え，高度な技術力とクラウドに対する豊富な知識を持つ人材の育成が必要である．引き続き，これらの観点を含めたサーバレスアーキテクチャーの適用によるビジネスアジリティの向上の可能性について，研究を進めていきたいと考えている．

謝辞

　本論文の作成にあたっては，日本アイ・ビー・ エム株式会社 北川衛様より多くの助言を頂きました．あらためて深謝いたします．

参考文献

1. デジタルトランスフォーメーションに向けた研究会，DXデジタルトランスフォーメーションレポート～IT システム「2025 年の崖」の克服と DX の本格的な展開～，2018年9月7日
2. JGS研究2019プロジェクトチームIA001 松尾 直弥，サーバレスアーキテクチャーの適用検討プロセスの提案，2019年7月31日
3. Gartner,Magic Quadrant for Cloud Infrastructure as a Service, Worldwide, 2019.07.16
4. 経済産業省商務・サービスグループ キャッシュレス推進室，キャッシュレスの現状及び意義，2020年1月
5. Serverless Airline Booking, <https://github.com/aws-samples/aws-serverless-airline-booking,2020.05>
6. AWS DynamoDB Read/Write Capacity Mode, https://docs.aws.amazon.com/ja\_jp/amazondynamodb/latest/developerguide/HowItWorks.ReadWriteCapacityMode.html, 2020.06
7. AWS DynamoDB Service Level Agreement, https://aws.amazon.com/jp/dynamodb/sla/, 2020.06
8. AWS Lambda function scaling, https://docs.aws.amazon.com/ja\_jp/lambda/latest/dg/invocation-scaling.html, 2020.06

添付１：メンバーリスト ( 会社名アイウエオ順 )

添付２：Member list

(alphabetical order of company name )

別冊A-1：グッズ購買サイト要求仕様書

別冊B-1：データ定義書

別冊B-2：パラメータシート

別冊C-1：評価シート

添付１：メンバーリスト ( 会社名アイウエオ順 )



添付２：Member list

(alphabetical order of company name )

