

データ駆動型ユニケージャーキテクチャの提案

當仲寛哲¹ S. ブヤンジャルガル¹ 鈴木明夫^{1,2} 山本修一郎³

¹ 有限会社ユニバーサルシェルプログラミング研究所

² 一般社団法人持続可能なモノづくり・人づくり支援協会

³ 名古屋国際工科専門職大学 工科学部

¹〒105-0003 東京都港区西新橋 3-3-3 ²〒460-0008 愛知県名古屋市中区栄 3-1-26

³〒450-0002 名古屋市中村区名駅 4-27-1

あらまし 従来のコンポーネントアーキテクチャには、コンポーネント間の依存関係があるため、疎結合アーキテクチャの実現が難しいという問題があった。そこで、本稿ではコンポーネント間の依存関係を機能共通性、データ結合性の点から①ライナーによる共通機能の分離、②パイプによる共通機能のデータ結合する疎結合アーキテクチャの構成を可能とするユニケージャーキテクチャを提案する。さらに、具体例に提案手法を適用することにより有効性があることを確認する。

キーワード：エンタープライズアーキテクチャ、データ駆動、データレイクハウス、データ駆動型工程設計法

A Proposal of Data Driven Unicage Architecture

Nobuaki Tounaka¹, Buyanjargal.S¹, Akio Suzuki^{1,2} and Shuichiro Yamamoto³

¹Universal Shell Programming Laboratory LLC,

²Association for Support of Economic Sustainable Development for 21st Century,

³ IPUT in Nagoya

¹3-3-3 Nishi Shinbashi, Minato-ku, Tokyo, 105-0003 Japan

²1-4-16, Nishiki, Naka-ku, Nagoya, Aichi, 460-0003 Japan

³4-27-1 Meieki, Nakamura-ku, Nagoya, 450-0002 Japan

Abstract: It is an urging problem for the conventional IT system architecture with high dependencies between their comprising components to realize the loosely-coupled architecture which is required in this DX era. Thus, in this paper we propose a loosely-coupled architecture- Unicage which makes the dependencies loosen extracting the common functionality in the components into a liner and having the liners connect through data only. We also apply the proposing architecture into the common software design problem to test its advantage.

Keywords: Enterprise Architecture, Data Driven, Data Lakehouse, Data Driven Process Design

1. はじめに

近年、コロナ禍の影響で世界中にデジタル変革（DX）の推進が一躍しているが IT・データ分析系人材の不足、データやその分析のサイロ化、システム老朽化など課題が顕著化している。世界の経済活動がよりつながっている今日、市場のダイナミクスも速く企業には迅速なビジネス変革、

決断力とそれらを支えるデジタルエンタープライズアーキテクチャが求められている。

本稿では、既存システムに与える影響が低く、疎結合アーキテクチャを実現できるデータ駆動型ユニケージャーキテクチャを提案する。

以下では、まず、2 節で関連研究を説明する。3 節で

データ駆動ユニケージアーキテクチャを提案する。4 節でソフトウェア設計の共通問題に提案したアーキテクチャを適用する。5 節で考察について述べ、6 節でまとめと今後の課題について説明する。

2. 関連研究

山本ら[1]は老朽システムを抱えたままでは迅速なビジネス変革が難しく、まず老朽システムを解体し、ビジネスと厳密に整合する独立性の高い疎結合のコンポーネントによって IT システムを構成するような変革すべきであると指摘しており、その実現にマイクロサービスアーキテクチャを推奨している。

Harvard Business School レビュー [2]はデータ駆動型経営実現の障壁として①分析結果による経営層とのコミュニケーション不足、②分析技術力を有する人材不足、③データ分析のサイロ化、④分析結果共有の時間遅れを挙げている。

Bill Inmon ら[3]は Data Lake, DWH の拡張版として Data Lakehouse を提案しており、これは①生テキストデータ、②エンドユーザー向け分析用インフラ、③構造、非構造、テキストデータの集合体から構成されるとしている。また、分析用インフラについて、エンドユーザーはビジネスプロセスを常時円滑に運営し利益を追求するために用いるが、データサイエンティストは企業内データの何か新しいトレンドやパターンを見つけ出そうとして見方が違っていることを指摘している。そのため、企業は、従業員、市民データアナリストを社内で育成し内製化を進める取り組みをとっており、その実現に柔軟インフラシステムが求められているという[2]、[3]。

山本[4]は、企業における知識資本の構成法の一つとして OHRR 型組織を挙げている。OHRR 型組織は従来の組織構造とは別に、組織横断的な新たな関係を用意するという組織構成である。例えば、企業内 SNS などの CMC の活用で、この組織形態を実現できるという。この組織形態では知識資本が縦横断的に流動するため、組織の境界を越えたコミュニケーションが可能になる。その結果、迅速な経営決断、企業の社会関係資本の強化、特に目標の共有化が強化されることが期待できる。しかし、この推奨内容を実現するために具体的かつ柔軟な疎結合アーキテクチャとその実装方法が提案されていないという課題がある。

3. データ駆動型ユニケージアーキテクチャの提案

データ駆動型ユニケージアーキテクチャは、エンタープライズシステムアーキテクチャの 1 種であり、現場で発生している生のデータをそのまま保存するデジタルツイン志向でそのデータを直接処理することにより、必要な時に必要なデータを高速に引き出すシステム技術を有する。この特徴により、DX 時代におけるデータドリブン経営に必要な機能を現実的なカタチで提供できる。表 1 に、そのア

ーキテクチャの階層とそれにおける機能をまとめる。

表 1 データ駆動型ユニケージアーキテクチャ

アーキテクチャ階層	機能	データ
業務システム	複数コンポーネントの集合体	<ul style="list-style-type: none"> ・収集データ ・テキストデータ ・集計データ ・インデックス化データ ・出力データ
コンポーネント	複数ライナーの集合体	
ライナー	複数コマンドの集合体	
コマンド	データ処理	

*DPL: Data Processing Language

3.1. 業務システム

業務システムは、独立性の高いファイル配置と、ファイルを介してのみ連携する疎結合コンポーネントの組み合わせで構成されるため、柔軟性が高い。これは、山本ら[1]が指摘する独立性の高い疎結合のコンポーネントによって IT システムを構成するマイクロサービスアーキテクチャの一つといえる。

また、この業務システムを機能別に外部システムと連携するデータ収集系、データの保管・整理を賄うデータ保管系、業務処理、工程管理を行うデータ分析系の 3 つに分割することができる。系統同士は全てファイルを介して連携するため、システム開発、システム運用プロセスは独立性が高く運用管理しやすい特徴がある。

データ収集系では発生する構造、非構造、テキスト、バイナリなど様々の種類の生データをデータ発生セッション単位でファイル化する。つまり、データ発生源ごとにファイルは分化し、互いに独立している。データ連携の仕方や種類、サイズに制約はなく、企業活動のデジタルツインを構成するデータとなる。

データ保管系では、生データの意味内容を変えずに、同期的にテキスト形式(フィールド形式、タグ形式、ネーム形式)などの数少ない種類のファイル形式に変換する。ユニケージアーキテクチャ内部で扱うデータ形式の種類を絞り込むことにより、データを扱うソフトウェアの種類を減らすことができる。このアプローチは米国 IT 大手 Splunk Inc[6]が提供するシステムで実装されている。また、この保管系には生データをクレンジングし業務利用に資する使いやすい単位で集計したファイル、検索、分析、可視化を高速に行うためにインデックス化されたファイル、外部出力(対ユーザー、対システム)したファイルがある。これらのファイルは、互いに独立性を保っており、Bill Inmon ら [3]が提唱する Data Lakehouse アーキテクチャにおける Data Lineage 機能、Data version 機能、Data time travel 機能という重要な機能を既に実現されている。

3.2. コンポーネント

コンポーネントとは、1 業務工程を実装したシェルスクリプトのことで、図 1 に示す要素から構成する。1 工程は複数の工程処理から構成されるため、シェルスクリプトも複数の処理から構成される。この処理の単位は 1 ライナーで 1 シェルスクリプトは複数のライナーから構成される。1 業務は、例えば、5 節で取り上げる倉庫管理業務のように複数の工程から構成されるため、複数のコンポーネントで構築する。ここでもコンポーネント同士の連携は全て

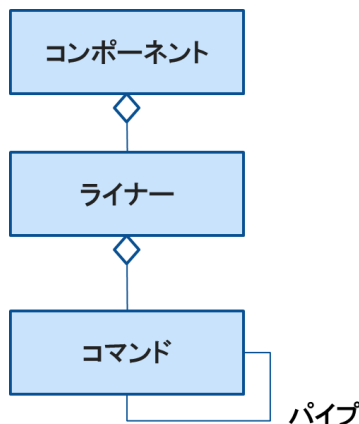


図 1 コンポーネントの構成

ファイルを介して行われるため、コンポーネントの結合が疎結合となっている。ユニケーザーキテクチャによる企業システム構築では、1 ライナーが 20-30 行のプログラム、1 コンポーネントが 100 行程度のプログラムになる。

図 2 は販売管理業務の 1 例で商品カテゴリー別に売上を集計する 1 ライナーからなる 1 コンポーネントの記述例を示す。ユニケーザーキテクチャにおけるプログラムを以下の構文に従って記述することによりコンポーネントを実装する。

〔構文〕コンポーネント

```
#####ライナーの説明#####
#コマンドの説明
<コマンド名>, <パラメータ並び>, <データ並び>,
<パイプで結合>
#####ライナーの終了#####
# 出力ファイルレイアウト記述
```

この構文に従ってプログラムを記述することにより、プログラムの可読性、保守運用性が高くなる。また、1 コマンド、1 ライナー毎に実行しながら、コンポーネント開発を進めるためバグが少ないという利点がある。

```
#####商品カテゴリー別に売上集計ライナー#####
# 原価 / 売価を連結
join1 key=2 PRICE SALES      |
# カテゴリーコードを連結
join1 key=2 CATEGORY         |
# 売数 / 売上 / 荒利計算
lcalc '$3,$7,$8,$8-$7*$4'    |
# カテゴリーコードでソート
msort -p4 key=1              |
# 売数 / 売上 / 荒利集計
sm2 1 1 2 4                  |
# 合計行の付加
sm5 1 1 2 4                  |
# 千で除算
divsen 2 3 4                 |
# 荒利率を求める
lcalc '$1,$2,$3,$4,100*$4/$3' |
# 四捨五入
marume 5.1                   |
# 商品カテゴリー名を連結して販売レポート出力
join2 key=1 CATEGORY_NAME    > REPORT.SALES
#####商品カテゴリー別に売上集計ライナー終了#####
# REPORT.SALES ファイルレイアウト
# ①カテゴリーコード ②カテゴリー名 ③売数
# ④売上 ⑤荒利 ⑥荒利率
```

図 2 1 ライナーから 1 コンポーネントの記述例

3.3. ライナー

1 ライナーは、1 業務処理を 1 単位としパイプで繋がれた DPL コマンド群である。Unix®/Linux パイプ（| - 縦棒）は標準入出力ファイルをまとめた機能であるため、コマンド間はファイルを介して連携している。すなわち、コマンド間の連携はデータ結合でソフトウェアのモジュールとして独立度が高く疎結合である。また、1 ライナー同士の連携も全てファイルを介して行われるため、疎結合となっている。

3.4. コマンド

コマンドとは Unix®/Linux シェルコマンドを拡張した DPL コマンドを指す。各コマンドは単一共通機能として作成されている。コマンドには表 2 に示す 5 種類があり、C 言語などの言語で入出力のデータ構造に従い処理が記述されている。

業務システム構築には、これまでのシステム構築経験によれば、平均して 20 から 40 個程度のコマンドが必要であることが分かっている。このコマンドの開発アプローチは、山本ら [5] が述べるデータ駆動型工程設計法と同様な考え方に基づいており、業務システムを構築する上で入出力データに良く現れ、コアとなる共通機能をコマンド化してい

る。このコマンド群は、筆者らの 20 年あまりの業務システム開発のノウハウに基づいて実装されている。

このような Unix®/Linux シェルコマンドを拡張した言語は新しいアプローチではない。例えば、米国 IT 大手 Splunk Inc[6]がこの 20 年間同様なアプローチで SPL™ (Search Processing Language) を提供している。

表 2 コマンドの種類

コマンド分類	用途
加工系	データの選択、削除、ソート、結合、分割
計算系	数学・統計計算、集計、関連性、傾向性の計算、比較
形式変換	データの形式変換、文字列加工
入出力	データの収集、出力
システム	システムの連携、状況確認、アクセスコントロール

4. 適用例

以下では提案アーキテクチャの適用事例を示すために、ソフトウェア設計の共通問題[7]に対して提案アーキテクチャに基づく実装を説明する。データ駆動工程設計法に基づいてこの共通問題を分析すると、図 3 に示すとおりである。なお、支払い請求・決済・領収などの処理は、共通問題では除外されている。

図 3 には、以下の 3 工程の処理がある。

- ① 銘柄在庫管理。出庫依頼票が届いたら依頼銘柄の在庫を確認する。在庫がある銘柄は銘柄出庫指示、在庫がなければ銘柄不足連絡を行う
- ② 銘柄出庫。出庫指示を受け、コンテナ内蔵銘柄の数量を減算する。
- ③ コンテナ管理。出庫対象コンテナの内蔵銘柄の数量を確認し全銘柄の数量が 0 の場合にコンテナ搬出連絡を行う。

各工程の入出力データは以下のように定義する。このデータは業務に資する集計データに相当する。

<出庫依頼票：4 フィールドデータ>

- ① 銘柄コード ② 銘柄名 ③ 数量 ④ 送り先

<積荷票：3 フィールドデータ>

- ① 内蔵銘柄コード ② コンテナ番号 ③ 在庫数量

<出庫指示書：7 フィールドデータ>

- ① 注文番号 ② 送り先 ③ コンテナ番号 ④ 銘柄コード
⑤ 銘柄名 ⑥ 数量 ⑦ 搬出空コンテナ番号

<在庫不足連絡：4 フィールドデータ>

- ① 注文番号 ② 銘柄 ③ 不足数量 ④ 送り先

<出庫銘柄：4 フィールドデータ>

- ① 注文番号 ② 銘柄 ③ 数量 ④ 送り先

<コンテナ：3 フィールドデータ>

- ① コンテナ番号 ② コンテナ空状況 ③ コンテナ搬出状況

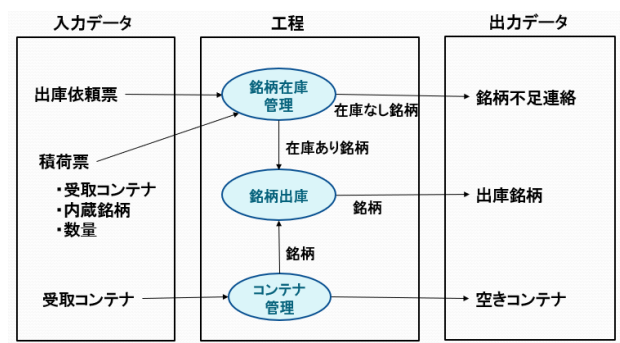


図 3 倉庫管理における工程

倉庫管理業務の実装例：

```
#####銘柄在庫管理#####
# 在庫計算処理・ライナー
# 出庫依頼票と積荷票を商品コードで結合
join1 key=1 積荷票 出庫依頼票 |
# 依頼数量とコンテナ内蔵品の差異計算
lcalc '内蔵品数量-依頼数量' 標準入力 |
# 銘柄在庫の確認
awk '$差異>0' > 中間ファイル
# 中間ファイルレイアウト
①内蔵銘柄コード ②コンテナ番号 ③在庫数量
④銘柄名 ⑤数量 送り先 ⑥差異

# 在庫確認処理・ライナー
# 在庫数量が不足の場合に銘柄不足連絡通知
# (メールか自動発注処理を実行)
# 中間ファイルが空か確認
if [ ! -s 中間ファイル ]; then
# 空の場合
不足通知処理実行
終了処理
Fi
# 在庫確認処理・ライナー終了

#####銘柄出庫#####
# コンテナ内蔵銘柄の出庫処理・ライナー
# 内蔵銘柄コード コンテナ番号 差異を選択
self 内蔵銘柄コード コンテナ番号¥
差異 中間ファイル |
# 積荷票データの更新
up3 key=内蔵銘柄コード/コンテナ番号 積荷票データ
標準入力 >積荷票.UPDATE.20220822 |
# 銘柄出庫処理・ライナー終了

#####コンテナ管理#####
# コンテナ内蔵全銘柄の数量集計処理・ライナー
# 全ての数量が0であれば、空状況を更新
# 銘柄コード、コンテナコードをキーに
# 銘柄数量の集計を計算
sm2 1 2 3 3 積荷票.UPDATE.20220822 |
# 集計が0のコンテナを抽出
selr 3 0 > 中間ファイル
# 中間ファイルレイアウト
# ①銘柄コード ②コンテナコード
```

倉庫管理業務の実装例（続き）：

```
# 空コンテナデータの状況更新処理・ライナー
if [ -s 中間ファイル ]; then
# 中間ファイルが空ではない場合、
# 空となったコンテナコード選択
self 2 中間ファイル |
# 空となった状況、搬出指示を追加
strcat -e 1 "" 空"" "" 搬出指示なし"" |
# コンテナデータの更新
up3 key=コンテナ番号 コンテナデータ 標準入力¥
> コンテナデータ.UPDATE
fi
# コンテナ状況更新ライナー終了
```

この例では、プログラムエラーハンドリング、終了処理、データ更新時のロックなどの処理を省き、アーキテクチャの基本的なコンポーネントのみ示した。また、データ収集、データクレンジングはこの例では省略した。

5. 考察

以下では、提案したアーキテクチャの新規性、有効性、限界について説明する。

5.1. 新規性

従来の IT システムを構築するコンポーネントアーキテクチャについて論じる研究はこれまでもあった。しかし、その疎結合アーキテクチャを実現する具体的な設計法は明確ではなかった。

本稿で提案したコンポーネント、ライナー、コマンドからなるデータ駆動型ユニケージャーキテクチャ (Data Driven Unicage Architecture, DDUA) は、コンポーネント間の依存関係をライナーによる共通機能の分離、パイプによる共通機能をデータ結合することで疎結合アーキテクチャを実現している点に、新規性がある。

5.2. 有効性

提案した DDUA アーキテクチャを共通例題に適用することにより、DDUA の有効性を示した。

5.3. 限界

(1) 機能における限界

提案アーキテクチャによって実現できる処理の種類は、選択、ソート、統計、比較、結合、関連性、類似性、形式変換と、その組み合わせであり、それ以外のものは現在実装されていない。今後のニーズに対応したコマンド群を開発する必要がある。

(2) 適用分野における限界

提案アーキテクチャが有効となる適用分野は ERP (Enterprise Resource Planning) である。自動運転、ロボット、FAなどのオートメーションシステムの作成には向かない。外部システムからのデータ取り込みやコントロールについては、コマンド化することによって、徐々に適用

範囲を増やすことはできるが、今後の課題である。

6. まとめと今後の課題

6.1. まとめ

本稿でデータ駆動型ユニケージャーキテクチャ DDUA は、データを介して連携するコンポーネントからなる疎結合アーキテクチャを実現できることを示した。これは、山本ら[1]が推奨する「ビジネスと整合する独立性の高い疎結合のコンポーネントによって IT システムを構成すべき」という点と共通性がある。したがって、そのような IT システム構築を可能にするアーキテクチャの一つが DDUA であることが分かった。

6.2. 今後の課題

今後の課題には、以下がある。

(1) 本稿では触れていないが、提案アーキテクチャは参考文献[7]、[8]のような、多くの導入実績がある。今後は、その適応事例を元に提案アーキテクチャによるエンタープライズシステム構築、既存システムとの融合性について解明する必要がある。

(2) 提案した疎結合アーキテクチャによる企業のビジネス変革、組織変革、社内 DX 人材育成を客観的に説明する必要がある。

(3) 提案アーキテクチャに基づいて開発した、生データからデータクレンジング、データ分析、可視化、そのリアルタイム共有、対話機能をワンストップで提供するツールを企業のビジネス変革、組織変革、DX 人材育成の観点から評価する必要がある。

謝辞

DDUA を導入していただいている協和工業株式会社 代表取締役社長 鬼頭佑治氏はじめ関係者の皆様に深謝します。

参考文献

- [1] 山本修一郎, DX の基礎知識 具体的なデジタル変革事例と方法論, 近代科学社 Digital, 2010
- [2] “UNCOVERING THE KEYS TO BECOMING TRULY ANALYTICS-DRIVEN”, Harvard Business School Review, 2018
- [3] Bill Inmon, Mary Levins, Ranjeet Srivastava, Building the Data Lakehouse, Technics Publications, 2021
- [4] 山本修一郎, CMC で変わる組織コミュニケーション, NTT 出版株式会社, 2010
- [5] 山本修一郎, 細見純子, データ駆動工程設計法の提案, 電子情報通信学会, 2022
- [6] David Carasso, Splunk' s Chief Mind, Exploring Splunk, Search Processing Language (SPL), CITO

Research, 2012

- [7] 山崎利治, 共通問題によるプログラム設計技法解説, 情報処理, vol. 25, No. 9, pp. 934, 1984
- [8] 鬼頭佑治, 小林英治, 中小製造業の DX 化課題解決の事例発表 : NKS: New KYOWA System, 一般社団法人持続可能なモノづくり・人づくり支援協会, 2022
- [9] 朝比奈史樹, ハンズラボとユニケージのあゆみ, ハンズラボ株式会社, 2020