Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 225 (2023) 609–616

Procedia Computer Science

www.elsevier.com/locate/procedia

27th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES 2023)

Data Driven Process-Based Unicage Architecture

Nobuaki Tounaka[a], Shuichiro Yamamoto[b], Buyanjargal Shirnen[a,b,] *

[a]Unversal Shell Programming Laboratory Ltd., 3-3-3 Nishishimbashi, Minato-ku ,Tokyo 105-0003, Japan
[b]IPUT in Nagoyo, 4-27-1 Meieki, Nakamura-ku, Nagoya 450-0002, Japan

Abstract

Digital transformation necessitates the creation of business value or innovation using digital technology in every industry. Enterprises are facing an unprecedented business environment and competition for resources and market, which forces them to be more adaptive. Consequently, their Enterprise Information Systems (EIS) must also be adaptive. The conventional information system architecture has high dependencies between their application components to realize the loosely-coupled, microservice application architecture required for adaptiveness. In this paper, we propose a loosely-coupled, process-based enterprise information system architecture - Unicage - which consists of business, application, data, and technology architectures, especially for Enterprise Resource Planning system, with only input and output data-based building blocks. As a result, it enables a Full Cycle Development. We also propose a model for the architecture using ArchiMate.

Keywords: Data Driven Enterprise Architecture, Microservice Architecture, Data Driven Process Design, TOGAF, ArchiMate.

1. Introduction

The objective of the latest EA is business transformation, not the overall optimization of IT systems. Furthermore, since the objective of EA is business transformation, the business capability-based EA development method starts implementing parts that generate business value in stages. Activities that do not generate business value are not recommended. Thus, architecture development is a continuous, cyclical process, and in executing the Architecture Development Method (ADM) repeatedly over time. TOGAF [3] accepts the iterative model, which is a critical component of Agile. In microservice architecture, microservices that correspond to business capabilities are loosely coupled to promote DX. To achieve this, it is necessary to design and implement overall business, data and application, and technology architectures, which are consisting the EA as suggested by TOGAF core architecture layers, in the loosely coupled manner. This type of architecture enables the flexibility to iterative development such as Agile or TOGAF ADM because it considers only input and output (IO) information to/from the business process. The IO data driven architecture also advantages system developers to achieve a full cycle system development since it requires only understanding input and output in every business process which directly maps to an IO data-based application component.

Based on the fact mentioned above, we propose Data Driven Process Based (DDPB) Architecture consisting of loosely coupled building blocks that compose business, application, and technology architecture layers. We also illustrate a practical model to implement the architecture with ArchiMate [1].

In the following, the related research will be explained in Section 2. Next, Section 3 proposes DDPB Enterprise Architecture model by ArchiMate. Then, Section 4 explains a concrete application example of the DDPB architecture. Section 5 describes the discussion, and Section 6 summarizes this paper and mentions future issues.

* Corresponding author. Tel.: +81-3-3432-1174; fax: +81-3-3432-1174.
E-mail address: s-buyan@usp-lab.com

10.1016/j.procs.2023.10.046

2. Related work

Steven H. Spewak [2] suggested a Business-Driven or Data-Driven Information System Architecture as Enterprise Architecture Planning (EAP). It recommends that data be defined before applications, and data dependency determines the sequence for implementing application systems. Data must be provided at a reasonable, affordable cost. The DBMS is merely a tool for accessing and storing data and simply using it cannot fulfill the promises of Database Management Systems. With EAP, the first architecture defines all the data needed to support the business. When that is completed, the second architecture defines the applications needed to manage that data.

TOGAF [3] suggests Architecture Development Method (ADM) which is iterative, over the whole process within phases in Requirements Management. In the ADM, phase C involves Information Systems Architectures, referring to Steven Spewak's Enterprise Architecture Planning (EAP) as a data-driven approach.

Yamamoto [4] proposed a concrete evaluation framework to compare EA frameworks and concluded that TOGAF is the most powerful EA framework. However, for the model and method features, TOGAF lacks clear description on model and method architects who conduct tailoring of model and method for organizations, although tailoring tasks are explained.

TOGAF [3] points out that the majority of Enterprise Architecture frameworks focus on the specific set of deliverables while relatively silent about the methods to be used to generate them.

Yamamoto [5] proposed the approach to visualize the business value, business model, and business process using ArchiMate. In the approach, Yamamoto proposed the integrated models in ArchiMate notation.

Yamamoto [6] proposed the Data Driven Process Design Method (DDPDM) which focuses on elimination of the business process which does not creates business value or overlapping the process.

Netflex [7] proposed a model that says "Operate what you built" or a Full Cycle Software Development. It mentioned that the purpose of the software life cycle is to optimize "time to value" by converting ideas into working products and services for customers. Developing and running a software service involves a full set of responsibilities, which can create inefficiencies across the entire life cycle when segmented. Full Cycle Developers are responsible for the full software life cycle, from design to support, by combining all of these ideas together into a single development team equipped with amazing developer productivity tools.

CRIBB [8], James P. Houghton [9], João M.P. Moreira [10] mentioned or evaluated the technology performance of Unicage Architecture.

3. Data Driven Process-Based Unicage Architecture (UA)

The UA architecture complies with TOGAF [3] and Table 1. describes its core building blocks for the core architecture layers as in the TOGAF, to propose a practical model for EIS implementation. The ArchiMate metamodels based on TOGAF ArchiMate Metamodels for the core layers are explained in the following chapters. As shown in Table 1., a business process design maps directly to Application and Technology building blocks since each block is designed based on IO data or information, which allows the Microservice Architecture and Full Cycle Development. Having the business process analyzed and designed means having the data and application design ready at time, which allows efficiency of development cycle and the full cycle development available. We here exclude the UA development method.

Table 1. Data Driven Process-Based Unicage Architecture (UA) Core Building Block

TOGAF Core Elements	UA Core Building Block Definition	Note (IO: Input and Output)
Business Architecture	IO data-based business process	Consider a business process as a function with an input and output data; The processes are connected through information or data only. Analyze and design it with Data Driven Process Design Method [6].
Data Architecture	Name indexed text file	Conform to data architecture design principles, file and directory indexing convention in the UA development method.
Application Architecture	IO data-based application component	Implement as-is business process into an application component. The components are connected through data only as so the business processes.
Technology Architecture	IO data-based command	A command is the smallest building block with the same IO data-based structure as a business process and application component. The command has a sophisticated single functionality to process an input data, being able to be implemented by any programming language.
Architecture Development Method	UA development method	Composed of Lean and Agile principles with UA, Full Cycle Development, project management method, and programming standard including error handle, low-code, code description, and directory tree conventions.

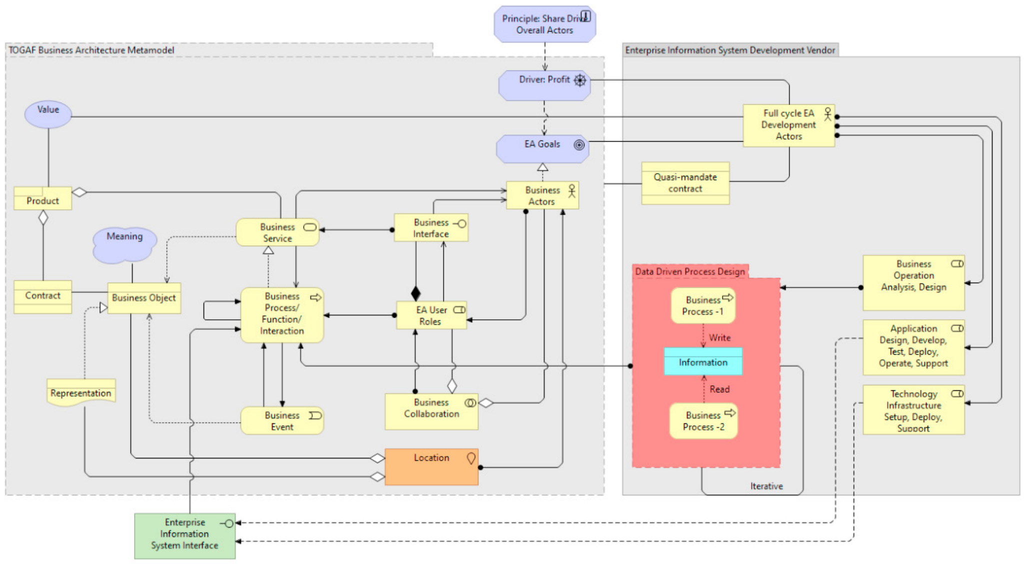
Principle: Share Drive Overall Actors
TOGAF Business Architecture Metamodel
Driver: Profit
Value
EA Goals
Product
Business Actors
Business Service
Business Interface
Meaning
Contract
Business Object
Business Process/ Function/ Interaction
EA User Roles
Representation
Business Event
Business Collaboration
Location
Enterprise Information System Interface
Enterprise Information System Development Vendor
Full cycle EA Development Actors
Quasi-mandate contract
Data Driven Process Design
Business Process -1
Write
Information
Read
Business Process -2
Iterative
Business Operation Analysis, Design
Application Design, Develop, Test, Deploy, Operate, Support
Technology Infrastructure Setup, Deploy, Support

Fig. 1. Business Architecture Metamodel in ArchiMate

3.1. Business Architecture Model in UA

The BA is along with TOGAF BA and the UA suggests the building block for implementation of business process. Fig 1. illustrates the relationship between EA and EIS vendor, especially in terms of business process analysis and design. The UA has principles in its EIS development.

- Principle 1: Profit oriented.
- Principle 2: Goal oriented and goal optimized overall development process toward Principle 1.
- Principle 3: Enterprise Architecture Development team be assembled of full cycle developers with a mission realizing Principle 2.
- Principle 4: Analyze and design every business process as an IO process.

The UA applies Data Driven Process Design Method [6] to analyze and design any business process conforming to Principle 4.

3.2. Data Architecture Model in UA

Based on the principles below, the entire data leaves a complete chronological record of the workings of EIS. This makes it easier to find out the cause of a problem when it occurs, since all the workings of the system can be visualized retroactively. Table 2. shows the data type, their application, and the data management capability. Having the few data types and the data management tool realized by data IO-based commands which are also used to realize an application component, allows an EIS developer less learning cost. Without using a middleware for data management, it also allows developers less learning and operation cost, and less cost for entire EIS development, which also supports EAP [2] suggestion.

- Principle 1: Allow redundancy, eliminate the unnecessary.
- Principle 2: Any data will NOT be overwritten, updated, or deleted, but spawned.
- Principle 3: Avoid data dependency.

Table 2. Data type in UA

Data type	Artifact	Application	Data management
Logical/layout data	Plain text file	Design, Coding, Documentation	Data IO-based software commands
Data	Plain text file	Business application	Data IO-based software commands
Media	Binary	Business application	Data IO-based software commands

The data architecture metamodel for the UA is depicted in Fig 2. which is along with the TOGAF metamodel.

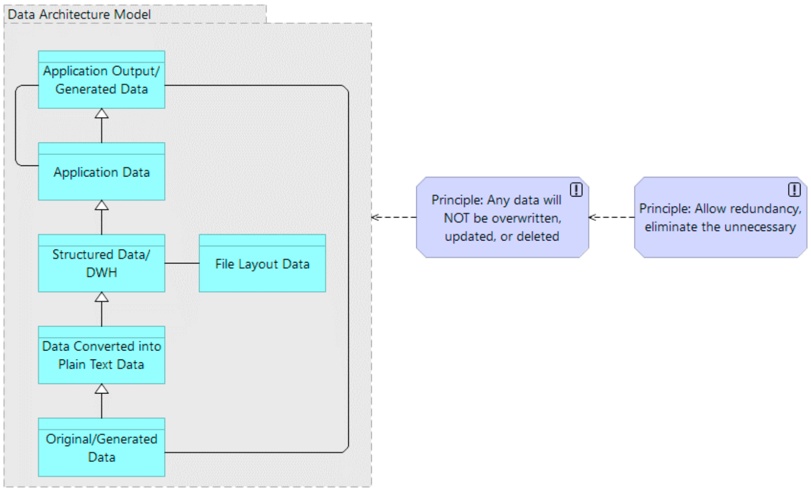
Data Architecture Model
Application Output/ Generated Data
Application Data
Structured Data/ DWH
File Layout Data
Data Converted into Plain Text Data
Original/Generated Data
Principle: Any data will NOT be overwritten, updated, or deleted
Principle: Allow redundancy, eliminate the unnecessary

Fig. 2. Data Architecture Metamodel in ArchiMate

3.3. Application Architecture Model in UA

An application component consists of component, liner, and command as shown in Fig. 3. A liner is a collection of commands which are connected by Unix Pipe and it expresses a stage that comprises a application component or a business process. Any application, liner, or command is connected through data object only. In other words, the application architecture is Microservice Architecture or loosely coupled. An IO-based application component matches its serving business process in structure with the same IO data, which implies that the business process design is ready so is the application design. The application component also conforms to the following principles in terms of code understandability, code simplicity, data management, and system maintenance.

- Principle 1: Write a description for input and output file layout into/from a command.
- Principle 2: Write a single command with input data in a line.
- Principle 3: Write a description for each liner and application component/business process.
- Principle 4: Applications that create data should be implemented before applications that use data.

The application component realized by the UA meets the following criterion that EAP [2] suggests, due to the IO data-based microservice application and technology architecture.

- Criterion 1: It should be understandable. The application definitions make sense and are easily understood by people throughout the enterprise.
- Criterion 2: It should be complete. Applications support most business functions and every data entity is being managed; and it should be consistent with no overlap or duplication of application capabilities.
- Criterion 3: It should be stable. Each application definition is based solely on the business model and data architecture and is independent of who uses the application, how the application works, where the application is located, or when the application is operated.

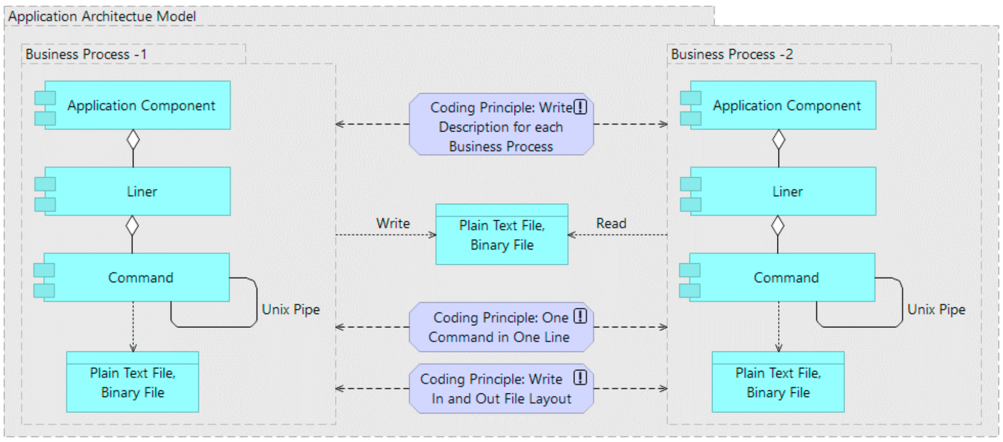
Application Architectue Model
Business Process -1
Application Component
Liner
Command
Unix Pipe
Plain Text File, Binary File
Coding Principle: Write Description for each Business Process
Write
Plain Text File, Binary File
Read
Coding Principle: One Command in One Line
Coding Principle: Write In and Out File Layout
Business Process -2
Application Component
Liner
Command
Unix Pipe
Plain Text File, Binary File

Fig. 3. Application Architecture Metamodel in ArchiMate

3.4. Technology Architecture Model in UA

Fig. 4 shows the Technology Architecture Model in which the IO data-based command lies at its core. The commands are implemented with the core libraries of an operation system based on POSIX (Portable Operating System Interface). The command can be implemented in any programming language and it executes sophisticated data processing algorithm while providing simple coding syntax or interface such as below. The command serves as a software cushion implementing complex algorithm which may take a long line of codes if it is implemented in a computer programming language.

- Command syntax: {command}, {parameters}, {data}

The command or tool must comply with the following principles.
- Principle 1: Use only core and sophisticated POSIX libraries to implement the command.
- Principle 2: Keep the command syntax normalized as in the command syntax above.
- Principle 3: Keep the number of commands as few as possible. Thus, try combination of commands to solve any required data processing before creating a new command.

In appreciation to the software cushion and principles above, the Technology Architecture in UA meets the criteria that EAP [2] suggests to apply open-systems concepts, meaning that operating systems should be as follow.
- Portable: run across multiple vendor platforms.
- Scalable: run across a wide power range from small to large computers.
- Interoperable: run in a heterogeneous environment.
- Compatible: preserve the investment in existing software and enable technology advances to be integrated with other components.

Table 3. explains the taxonomy of a command set and Fig 4. proposes the technology architecture model in ArchiMate.

Table 3. IO Data-based Command category

Category name	Usage
Data manipulation	Data selection, delete, sort, join, split
Calculation	Mathematical, statistical, summary, comparison
Formatization	Data conversion, string manipulation
Input and output controller	Data retrieving, sending
System process control	Job control, system collaboration, access control

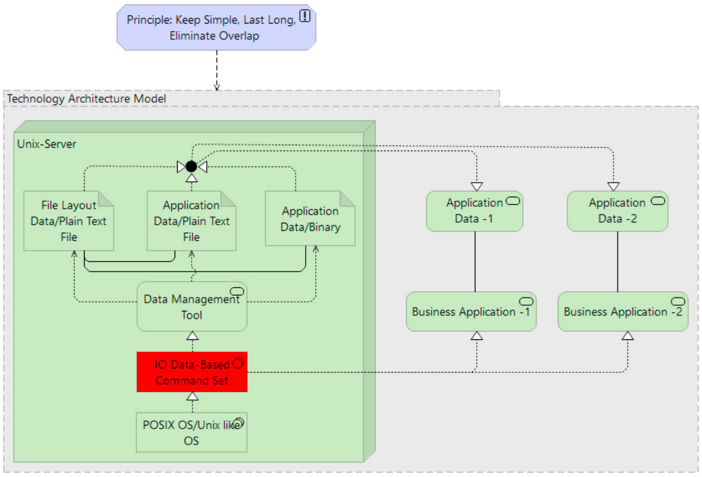
Principle: Keep Simple, Last Long, Eliminate Overlap
Technology Architecture Model
Unix-Server
File Layout Data/Plain Text File
Application Data/Plain Text File
Application Data/Binary
Data Management Tool
IO Data-Based Command Set
POSIX OS/Unix like OS
Application Data -1
Application Data -2
Business Application -1
Business Application -2

Fig. 4. Technology Architecture Metamodel in ArchiMate

4. Case study

4.1. Liquor Shop Problem

We apply the UA to a software design problem to demonstrate its adaptability, illustrating in ArchiMate. The problem is called a Liquor Shop Problem [6] and it challenges a software design to realize an order and inventory management process in the shop warehouse. Table 4. describes a flow of the management processes from receiving a container with liquor, dispatching an order of liquor, and empty container management to contact a customer who places the order.

Table 4. Liquor shop order and inventory management process

Process actor	Business process	Output or input information to next process
Warehouse manager	Receive a logistic container and place it in the warehouse.	Hand over a container receipt to an order dispatcher. A product receipt information: {Container number, Delivery date, Liquor name, Amount of liquor}
Order dispatcher	Receive 10 orders per day.	Request a warehouse manager to pick up liquor from the container to fill each order. The request is made through a telephone or a request form. An order information: {Liquor name, Amount, Delivery address} A request form information: {Order number, Delivery address, Container number, Liquor name, Amount, Mark for empty container} Make a phone call to a customer if ordered liquor is out of inventory or insufficient amount to fill and note that in an inventory shortage list. A shortage list information: {Delivery address, Liquor name, shortage amount}
Order dispatcher	Checks if the liquor in the shortage list is received.	Request a warehouse manager to fill the order in the shortage list once the insufficient liquor is received.
Order dispatcher	Check a container if it is scheduled to be empty.	Notify the container information to the warehouse manager to keep the number of containers in the warehouse as few as possible.

We assume the business processes in the problem analyzed and optimized by DDPDM [6] which results in optimizing the number of processes from 10 to 8, and have the result in the proposing ArchiMate metamodels.

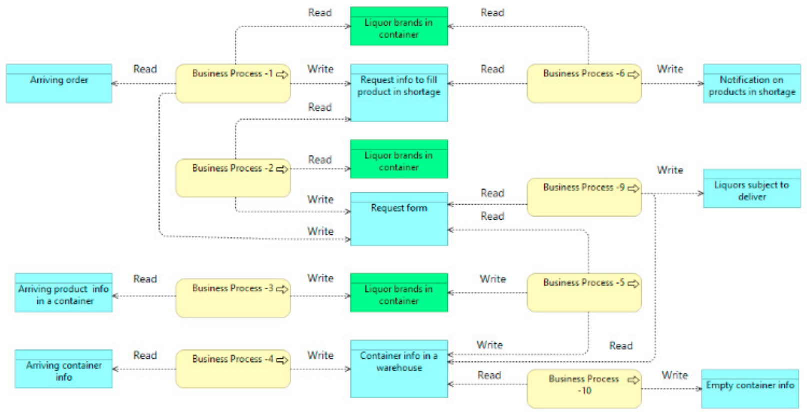

Fig.5.Liquor shop order and inventory management processes in ArchiMate

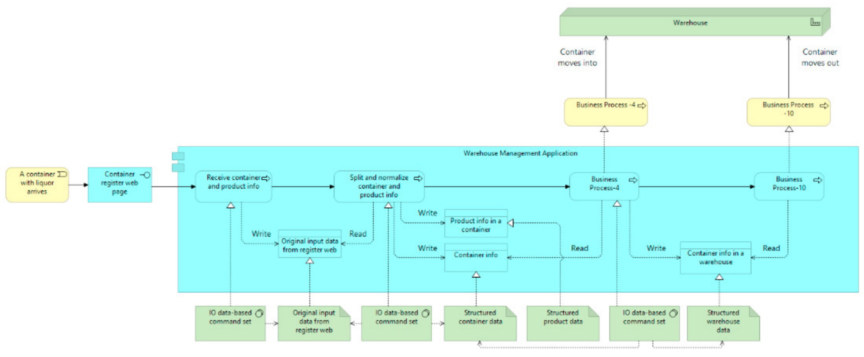
Warehouse
Container moves into
Container moves out
Business Process -4
Business Process -10
Warehouse Management Application
A container with liquor arrives
Container register web page
Receive container and product info
Split and normalize container and product info
Business Process-4
Business Process-10
Product info in a container
Original input data from register web
Container info
Container info in a warehouse
Write
Read
IO data-based command set
Original input data from register web
Structured container data
Structured product data
Structured warehouse data

Fig.6. A process implementation in the UA in ArchiMate

4.2. ArchiMate Notation

Fig 5. illustrates the order and inventory management processes after applying DDPDM [6]. For the simplicity of ArchiMate model, the "Liquor brands in container" data object is duplicated. Fig 6. demonstrates the UA capability implementing a part of the processes.

5. Discussion

5.1. Novelty

The major enterprise architecture frameworks define a set of deliverables of the framework while a practical implementation of building blocks is relatively mentioned less. This paper proposed the building blocks to implement the core architecture layers recommended by TOGAF [3] and demonstrated its adaptability through a case study.

5.2. Effectiveness

The proposed architecture has been effectively applied to analyze business processes based on IO data and implement application components directly mapping to the business processes, connecting through data only. It has also shown that the architecture is a microservice or loosely coupled architecture. The proposed metamodels along with TOGAF architecture layer metamodels in ArchiMate has also used to demonstrate the adaptability of the architecture on the case study.

5.3. Limitation

Although the proposed architecture has effectively applied to the case study with ArchiMate illustration, and adopted by many enterprises, the architecture is designed for ERP, especially. An integrated ArchiMate metamodel for the entire architecture is needed to illustrate. Furthermore, a quantitative evaluation on its effectiveness shall be conducted.

6. Conclusion

In this paper, we proposed the data driven enterprise architecture with practical implementation blocks with ArchiMate metamodels. Future work includes more case studies illustrated with proposed ArchiMate models and designing an integrated ArchiMate metamodel for the entire architecture.

References

[1] Phillip Beauvoir & Jean-Baptiste Sarrodie. (2022) "Archi -archimate modelling", Archi version:4.9.3 supporting the ArchiMate® 3.1 language.

[2] Steven H. Spewak and Steven C. Hill. (2008) "Enterprise Architecture Planning: Developing a Blueprint for Data, Applications, and Technology", John Wiley & Sons, Inc.

[3] The Open Group. (2018) "The TOGAF® Standard, Version 9.2"

[4] Shuichiro Yamamoto, Nada Ibrahem Olayan, Shuji Morisaki. (2018) "Another Look at Enterprise Architecture Framework", Journal of Business Theory and Practice: Vol. 6, No. 2, 2018.

[5] Shuichiro Yamamoto. (2020) "A DX Visualization Approach Using ArchiMate", IPSJ SIG Technical Report: Vol.2020-SE-204 No.18.

[6] Shuichiro Yamamoto, Junko Hosomi. (2022) "A Proposal on Data Driven Process Design Method", IEICE Technical Report KBSE2022-24: vol. 122, no. 139,79-84.

[7] Philip Fisher-Ogden, Greg Burrell, and Dianne Marsh. (2018) "Full Cycle Developers at Netflix – Operate What You Build", Netflix Technology Blog.

[8] COMPUTATIONAL RESEARCH in BOSTON and BEYOND (CRIBB). (2013) "How to Analyze 50 Billion Records in Less than a Second without Hadoop or Big Iron", https://math.mit.edu/crib/2013/aug2.html

[9] James P. Houghton et al. (2017) "Tracking the evolution of conversational clusters in social media", https://doi.org/10.1177/0049124117729705

[10] João M.P. MoreiraHelena GalhardasHelena GalhardasMiguel PardalMiguel Pardal. (2018) "LeanBench: comparing software stacks for batch and query processing of IoT data", DOI: 10.1016/j.procs.2018.04.067