

Architectural Styles for Distributed Processing Systems and Practical Selection Method

Yoshitomi Morisawa^{a, c, *}, Katsuro Inoue^b, Koji Torii^c

^a Nihon Unisys, Ltd., Koto-ku, Tokyo, 135-8560 Japan

^b Graduate School of Engineering and Science, Osaka University, Toyonaka-shi, Osaka 560-8531, Japan

^c Graduate School of Information Science, Nara Institute of Science and Technology, Ikoma-shi, Nara 630-0101, Japan

Received xx August 2001; accepted xx yyyy 2002

Abstract

The software architecture of a system has influences against various software characteristics of the system such as efficiency, reliability, maintainability, and etc.. For supporting to design the software architecture, we have developed architectural styles for distributed processing systems. The styles classify the architecture for distributed processing systems into nine categories based on the location of data storage and the type of processing between a client and a server. This paper describes our architectural styles and proposes a simple but practical method to select an appropriate architectural style for developing an application system. The selection method introduces the characterization of architectural styles and the characteristic charts to visualize their characteristics of architectural styles. Next, we propose a method to select an appropriate architectural style using the conformity between characteristic charts of a system and architectural styles. We have verified the applicability of this selection method using our customers' real application systems.

Keywords: Architectural Style, Distributed Computing Model, Distributed Processing System, Software Architecture

1. Introduction

There are many software products commercially available for implementing Client/Server(C/S) systems. When a user implements an application system utilizing these products, it is common to spend a large amount of time and effort testing the interconnectivity and interoperability, known as conformance testing, among the products to be used. At Nihon Unisys, Ltd.(NUL), we have developed a body of expertise and experience in conformance testing by submitting proposals to customers, providing consultation in information technology, and supporting the implementation of our customers'

application systems. In order to share this expertise and experience, as well as to reduce the cost of developing new customer application systems, we needed to create a framework which includes: a software architecture for implementing C/S systems; an intuitive and simple model of distributed processing systems including C/S processing and proven combinations of products based on this model. NUL named this framework the Open Solution Framework (OSFW) and the distributed processing model the Client/Server Solution (C/SS) model, and announced the OSFW in January 1996 [11,12]. In order to take in the mobile and agent software technology to the C/SS model, we re-evaluated the model and announced new model as architectural styles for distributed processing systems in March 1999[13]. In this model, we classified distributed processing systems in the business application

* Corresponding author, Tel.: +81-3-4329-2164;
Fax: +81-3-5546-7865,
E-mail address: morisawa@dp.u-netsurf.ne.jp

domain into nine architectural styles.

The main motivation for introducing the architectural styles is to categorize the architecture of distributed processing systems, to provide proven software products called product sets for implementing an application system in each category of distributed processing systems, and to reduce the total cost of the application system. To achieve these goals, we developed the OSFW as a framework of distributed processing systems and are using architectural styles as reference architectures to design an architecture of an application system and to select product sets for implementing and administrating the application system. Product sets for each style are periodically revised to reflect the latest software products and are used with this framework for configuring our customers' services.

In the architectural design phase of an application system development, it is an important decision to design a system's software architecture that has influences against various software characteristics of the system such as efficiency, reliability, maintainability, and etc.. System architects have generally to collect various requirements of stakeholders such as users, developers, and managers for the system to be developed, and to design the software architecture from the requirements referring existing architectural styles and their previous experiences. Developers implement the system using the software architecture designed by architects. The software architecture is a result of technical, business, and social influence. The existence of the software architecture is in turn affects the technical, business, and social environments that subsequently influence future architecture. Bass calls this cycle of influences, from the environment to the architecture and back to the environment, the architectural business cycle (ABC) [1]. The ABC is showed in Figure 1 and its cycle is as follows;

- (1) Architect(s) creates architecture of an application system based on requirements, technical environment, and their experience.
- (2) Analyse and generate the architecture of the application system.
- (3) Develop the application system using the architecture.
- (4) Influence architecture experiences and architectural styles.

For supporting to design the software architecture and further to select software products, we had developed the intuitive and simple architecture styles

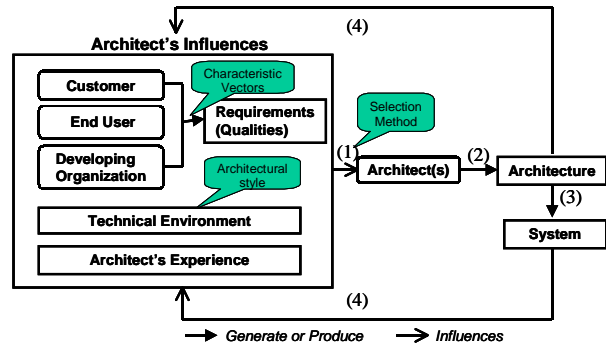


Figure 1. Our research works in the ABC

for distributed processing systems including C/S systems. The architectural styles themselves were designed so that field system engineers could understand intuitively and use them as reference architectures of distributed processing systems. However, at a time when system architects have to decide architecture of an application system, it is not unusual that data location and processing style between a client and a server (and among servers) are not settled. The architects generally have to decide architecture through a broad view of users, developers and managers. Therefore, they may select an appropriate architectural style from the viewpoint of managers, even if the model is not most suitable from the viewpoint of users and developers. We have been requested to provide a simple but practical method to select an architectural style for less experienced architects.

In this paper, we introduce our architectural style and propose a simple but practical method to select an architectural style for developing an application system[15]. Using balloons in Figure 1, we explain the positioning of our research works in the ABC. Our architecture styles, which are explained in Section 2, are used as reference architectures in the technical environment. The characterization of architectural styles is requirements from Customer and End User, and Developing Organization. Those are listed as characteristic vectors and the selection method is a proposal for architects how to select an appropriate architectural style for the application system that is the step (1) in Figure 1. This selection method is explained in Section 3. We apply this method in real business application systems in productions. This experience is described in Section 4. Finally, we provide the information of related works and the conclusions.

2. Architectural styles for distributed processing systems

The target domain of our architectural styles is the business application systems in the distributed processing environment. We classify the target domain into Information domain, Business domain and Office support domain[14]. Information domain is generally called the front office and its typical computing paradigm is a C/S processing. Business domain is generally called the back office and its typical computing paradigm is a transaction processing. Office support domain is generally called the center office or the middle office, and its typical computing paradigm is a collaborative processing such as groupware, workflow, and e-mail.

Next, we classify the distributed processing systems into nine architectural styles from the viewpoints of the location of data storage and the processing type between client and server. The location of data storage is classified as centralized or distributed. This distributed data is further classified as synchronous processing and asynchronous processing between servers. This location view and its classification is easy to understand and adopt by the field system engineers implementing practical systems. The processing type between client and server is classified as synchronous or asynchronous. Synchronous processing is further divided into two categories, Transaction type and Query type, depending on the characteristics of the messaging between the client and the server.

This classification scheme leads us to develop nine styles of distributed processing systems illustrated in Table 1. Note that primary purpose of the classification is to provide an intuitive, easy and

simple style for most field system engineers. So, the architectural styles and their classification should be simple, straightforward, and not too complicated.

As a general rule, we assume that "Presentation" is in a client side and "Data" is in a server side.

The meanings of the terms used in Table 1 are as follows.

- "Centralized" means that the data is stored in only one server, and "Distributed" means that the data is distributed into multiple servers. However, personal data or a personal database in a client side is not included.
- "Processing type" indicates the processing style between a client and server(s), and among servers. They are a synchronous processing and an asynchronous processing.
- "Transaction Type" is typical transaction processing with ACID (atomicity, consistency, isolation and durability) properties [6].
- "Query Type" indicates that a reply from the server is synchronized with a request from a client.
- For asynchronous processing as the processing type, "Notification Type" is assumed, which indicates that the server process is not synchronized with a client request.

The following sections provide a typical structure and a description of each architectural style based on the processing types between servers for supporting to understand our proposed architectural styles. The symbols used in typical structures and in this paper show in Figure 2.

2.1 Architectural styles for transaction types

Figure 2 shows typical structures of (a) centralized transaction style, (b) distributed

Table 1
Architectural Styles

Processing type between C/S	Location of Data		Distributed	
	Processing Type between Servers	Centralized	Synchronous Processing	Asynchronous Processing
Synchronous Processing	Transaction Type	Centralized Transaction Style	Distributed Transaction Style	Asynchronous Transaction Style
	Query Type	Centralized Query Style	Distributed Query Style	Asynchronous Query Style
Asynchronous Processing	Notification Type	Centralized Notification Style	Distributed Notification Style	Asynchronous Notification Style

Table 2
Symbols used in this paper

Symbol	Meaning
P	Presentation
AL _n	Application Logic
DM _n	Data Management
D _n	Data
↔	Transaction Message
↔	Query message
↔	Notification Message

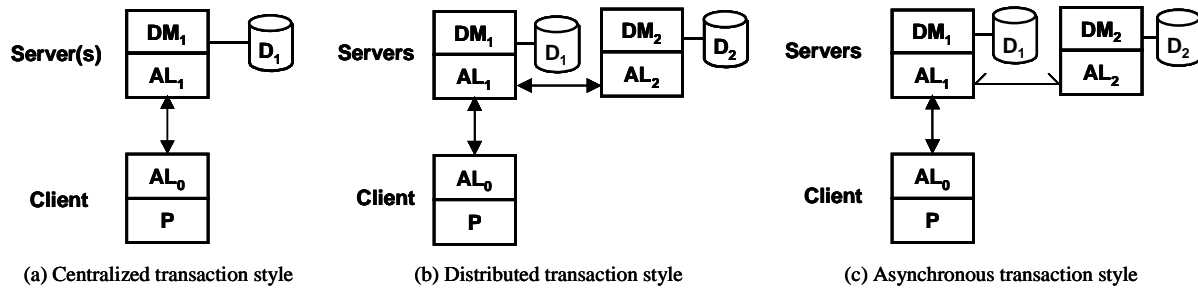


Figure 2. Architectural styles for transaction types

transaction style, and (c) asynchronous transaction style, respectively. Table 2 shows symbols used in the figure. These architectural styles mainly model a business domain called the back office. The centralized transaction style has a single database on a single server. The distributed transaction style has multiple databases on multiple servers and the processing type between servers is synchronous processing. The asynchronous transaction style has also multiple databases on multiple servers and the processing type between servers is asynchronous processing.

Message and process flow for these styles as follows: AL_0 client sends a transaction message to AL_1 . Using the processing result of AL_1 , DM_1 retrieves, updates, adds, and/or deletes data in D_1 . In the case of the centralized transaction style or the asynchronous transaction style, the result of the process is then returned to the client. Next, in the case of multiple databases, a new transaction or notification message generated by the processing result of AL_1 is sent to AL_2 . According to the processing result of AL_2 , DM_2 retrieves, updates, adds and/or deletes data in D_2 . Finally, in the case of the distributed transaction style, the processing result is sent back to the client and a mechanism is used to maintain the integrity of D_1 and D_2 if required.

The centralized transaction style is suited to transaction processing that uses a transaction control system and usually adds and updates data in a single database. Examples of this style include an order entry system, a stock management system, a stock ordering system, a production management system, a retail POS (Point of Sales) system, etc..

The distributed transaction style is suited to transaction processing for a core business involved complicated update and query of multiple databases. Examples of this include a core business system of banking, a seat reservation system, a factory production management system, etc..

The asynchronous transaction style is suited to a core application that promotes the management and utilization of information by asynchronous sharing of data. Examples of this style are asynchronous updating application of replicated data such as a personal information system or accounting system across multiple branches and uploading of transaction data collected at branches or departments such as combining the uploading of order data collected by departmental servers and the centralized order processing on an enterprise server.

2.2 Architectural styles for query types

Typical structures of (a) centralized query style, (b) distributed query style, and (c) asynchronous query style are similar structures in Figure 2 except message type between a client and server should be a query message instead of a transaction message. These architectural styles mainly model an information domain called the front office and are described details in our papers [14,15].

The centralized query style is suited to End User Computing (EUC) such as in a decision support system and to query and reply processing. Examples of EUC include various statistics, analysis and reporting, such as in budget planning, financial analysis, market research and analysis, sales analysis, capacity planning, demand forecast, etc.. Examples of query and reply processing include customer services, sales support, various inquiry processing, information providing services, etc..

The distributed query style is suited to EUC with simultaneous access to multiple databases and files and to inquiry-intensive immediate processing. This enables effective utilization of information by sharing existing databases. This style includes almost all of applications of the centralized query style. Other applications are an enterprise sales statistics, an enterprise productivity data analysis, etc..

The asynchronous query style is suited to an application to promote the management and utilization of information by asynchronous sharing of data. An example is an application of information utilization by downloading a part of database in business domains such as a decision support system (DSS) and an enterprise information system (EIS) using Daifukuchō (an old-fashioned account book in Japan) and multiple dimensional databases.

2.3 Architectural styles for notification types

Typical structures of (a) centralized notification style, (b) distributed notification style, and (c) asynchronous notification style are similar structures in Figure 2 except message type between a client and server should be a notification message instead of a transaction message. These architectural styles mainly model an office support domain called the center office and are described details in our papers [14,15] also.

The centralized notification style is suited to the automation of a simple workflow within a group or an organization. Examples of this style include the shipping and forwarding of internal memos and documents, events notification, internal document filing, execution, monitoring, and reporting of business workflow, etc..

The distributed notification style is suited to an application of distributed transaction processing and/or distributed data processing using an agent application on a server from mobile clients.

The asynchronous notification style is suited to the loose integration by cooperation with independent multiple applications or systems. Examples of this style are workflow between groups or organizations, integration of cooperating applications between enterprise systems, and cooperating systems between enterprises, such as electronic data interchange (EDI).

3. Selecting an architectural style

When implementing an application system in a distributed computing environment, an architectural style of the application system plays an important role. However, the application system may use various architectural styles according to the requirements of users, developers and managers, and the constraints on budgets and periods of time even if the system has to solve the same issue. In the present

situation, architects are intuitively selecting an appropriate architectural style for the application system using their own perceptions and experiences, and consultants are selecting architectural styles using the consulting methodology when our customer requests a consultation of a new system.

In this section, we propose a simple but practical method to help to select an appropriate architectural style for less experienced architects in designing their architecture at the early stage of an application system development. At the early stage of the development, architects do not become always clear the location of data storage and processing type between a client and a server from the requirements of users, developers, and managers. Our method may support to select an appropriate architectural style based on available requirements of users, developers, and managers at the early stage of the development. The requirements shows as quality requirements in the step (1) of the ABC. The quality requirements are represented as characteristic vectors. Architects design the architecture of an application system referencing the appropriate architectural style in the technical environment. Developers then implement an application system based on the architecture using proven software tools, called product sets in the OSFW, related with the architectural style.

3.1 Characterization of architectural styles

An architectural style prescribes a processing structure and a processing style of constitutional elements for an application system to develop. The architectural style, which defines a framework of an application system is chosen by various participants to use, to develop and to administer the system, and has a large influence in quality, cost of development and administration, and the development period of the system. Bass reports that architectural requirements are not as numerous as functional requirements; there should be a maximum of approximately 20 architectural requirements [2]. For practical use in the actual system development field, we restricted our study to seven viewpoints, rather than an exhaustive range of viewpoints because fewer viewpoints are practically applicable.

First, we discuss software qualities. The ISO standard defines functionality, reliability, usability, efficiency, maintainability, and portability as quality characteristics of software [7]. These quality characteristics are subdivided into 21 quality

sub-characteristics. They are as follows; Functionality is subdivided into suitability, accuracy, interoperability, compliance and security. Reliability is subdivided into maturity, fault tolerance and recoverability. Usability is subdivided into understandability, learnability and operability. Efficiency is subdivided into time behaviour and resource behaviour. Maintainability is subdivided into analyzability, changeability, stability and testability. Portability is subdivided into adaptability, installability, conformance and replaceability. These characteristics are used to define and evaluate quality requirements of software throughout a life cycle of the software.

We consider characteristics of an architectural style from the viewpoint of the people who participated in the life cycle of a system. When implementing a distributed processing type of an application system, system architects select an architectural style from a bird's-eye view based on the requirements of users, developers and managers and design the application system referencing this style.

From the viewpoint of users of a system, functionality of required features, reliability, efficiency, usability and portability to other environments becomes a target of consideration for quality characteristics. However, in the selection stage of the architectural style, functionality other than security sub-characteristics is an assumed condition. Reliability, usability, and portability are characteristics at a detailed design stage, and efficiency becomes a target of consideration at this stage. Therefore, "Data Security" and "Reply to User" become characteristics important to the security and response time behaviour requirements of a system of users.

From the viewpoint of developers of a system, all of the software quality characteristics become a target of consideration. However, in the selection stage of the architectural style, functionality is an assumed condition of development. Reliability, usability, maintainability and portability are characteristics at a detailed design stage. Efficiency becomes a target of consideration. Therefore, "Scope of Client/Server", "Independency among Servers" and "Data Distribution" become characteristics due to the requirements of system structure and processing type.

Managers of a system optimize software quality within a limited budget of human resources and time frames. Therefore, "Budget of System" and "Delivery

of System" become characteristics due to the requirements of managing system development.

Based on the above observations, we have created the following seven characteristic vectors. For visualizing characteristics of an architectural style using a radar chart form, we use a numerical value in parentheses attached to every characteristic value of each characteristic vector. The attached value is based on the general requirements of software quality and system development so that we can develop an efficient and usable system at a lower cost within a shorter period of time. We assigned a large number as the characteristic value when a system is developed within a shorter period at a lower cost and with higher functionality and more efficiency. For less experienced architects to use these styles, we have set simple and understandable characteristic values such as, "Yes/Undefined/No".

1) Data Security

This data security is a characteristic vector from a users' point of view when we develop a system with a fixed budget and within a fixed time of delivery. This characteristic value is classified into "High(3)", so that a user can request a secured process; "Low(1)", so that a user can request a good enough process, and "Either(2)", so that a user does not need to decide a security level of a process or has no information about the security.

Accordingly, for the centralized data of the architectural style, we may characterize such security as "High" regardless of transaction type, query type, and notification type because we can easily take security measures when data is centralized. For the distributed, the characteristic value is relatively "Low". When data may be either distributed or centralized, or we have no description on data distribution, the security is "Either".

2) Reply to User

This reply to user is a characteristic vector used whether a user of a system requests a reply of the processing of the system or not. This characteristic value is classified into "Reply(3)" to demand a reply, "Notification(1)" without demanding a reply, and "Either(2)", when deciding a reply for a processing demand is not necessary.

Accordingly, architectural styles of transaction type and query type are characterized as "Reply" and notification type as "Notification".

3) Scope of Client/Server

This scope of client/server is a characteristic vector for the scope of a client (user) and servers. This character value is classified into “LAN(3)”, so that a client can have servers at a short distance and is connected to servers with only LAN; “WAN(1)”, so that a client and servers are dispersed to a wide area and connected with a WAN; and “LAN/WAN(2)”, so that a network is uneven in distance among a client and servers and a network is a mix of LAN and WAN.

A WAN type of network usually uses transaction type as the processing type between a client and servers. A LAN type of network has less limitation compared with a WAN type for transmission rate and communication volume. Therefore, a LAN type may use any processing type between a client and servers. In a style of query type, the LAN type is generally used for a network between a client and servers due to the communication load.

4) Independency among Servers

This independency among servers is a characteristic vector of the relationship among servers when a system consists of more than one server and accomplishes a process on servers. This characteristic value is classified into “Independent(3)”, so that processes on servers are executed asynchronously for a requester; “Dependent(1)”, so that processes on servers are executed synchronously for a requester. Finally, in “Either(2)” a relationship among servers cannot be decided.

The characteristic value of the asynchronous styles is “Independent”. In other architectural styles, the characteristic value is “Dependent”.

5) Data Distribution

This data distribution is a characteristic vector of data storage. This characteristic value is classified into “Distributed(3)”, so that data is distributed and may be handled locally; “Centralized(1)”, so that a data volume is centralized in one place and easy to handle for business use in one place; and “Either(2)”, so that we do not need to decide to adapt centralized and distributed data.

The architectural style with a centralized type has “Centralized”. A distributed type of an architectural style is “Distributed”.

6) Budget of System

This budget of system is a characteristic vector of a budget (cost) to develop and to administer a system. This characteristic value is classified into “Reasonable(3)”, for less budget than average implementation; “Enough(1)”, for more budget than average implementation; and “Either(2)”, for no information of budget or average implementation.

Characteristic values for each architectural style after applying the following rules learned by our experiences are: development of an application system with transaction type that has relatively extra cost, and development with query type that also has a relatively lower cost. Development with notification type requires an average cost.

7) Delivery of System

This delivery of system is a characteristic vector of a term of system delivery. This characteristic value is classified into “Short(3)”, when requesting a short term delivery of a system; “Enough(1)”, when allowing a sufficient development period of time; and “Either(2)”, when not decided.

Table 3
Characteristic values of architectural styles

Characteristic Vector Architectural Style	Data Security	Reply to User	Scope of Client/Server	Independency among Servers	Data Distribution	Budget of System	Delivery of System
Centralized Transaction Style	High(3)	Reply(3)	WAN(1)	Dependent(1)	Centralized(1)	Enough(1)	Enough(1)
Distributed Transaction Style	Low(1)	Reply(3)	WAN(1)	Dependent(1)	Distributed(3)	Enough(1)	Enough(1)
Asynchronous Transaction Style	Low(1)	Reply(3)	LAN/WAN(2)	Independent(3)	Distributed(3)	Enough(1)	Enough(1)
Centralized Query Style	High(3)	Reply(3)	LAN(3)	Dependent(1)	Centralized(1)	Reasonable(3)	Short(3)
Distributed Query Style	Low(1)	Reply(3)	LAN(3)	Dependent(1)	Distributed(3)	Reasonable(3)	Either(2)
Asynchronous Query Style	Low(1)	Reply(3)	LAN/WAN(2)	Independent(3)	Distributed(3)	Reasonable(3)	Short(3)
Centralized Notification Style	High(3)	Notification(1)	LAN/WAN(2)	Dependent(1)	Centralized(1)	Either(2)	Short(3)
Distributed Notification Style	Low(1)	Notification(1)	LAN/WAN(2)	Dependent(1)	Distributed(3)	Either(2)	Either(2)
Asynchronous Notification Style	Low(1)	Notification(1)	LAN/WAN(2)	Independent(3)	Distributed(3)	Either(2)	Short(3)

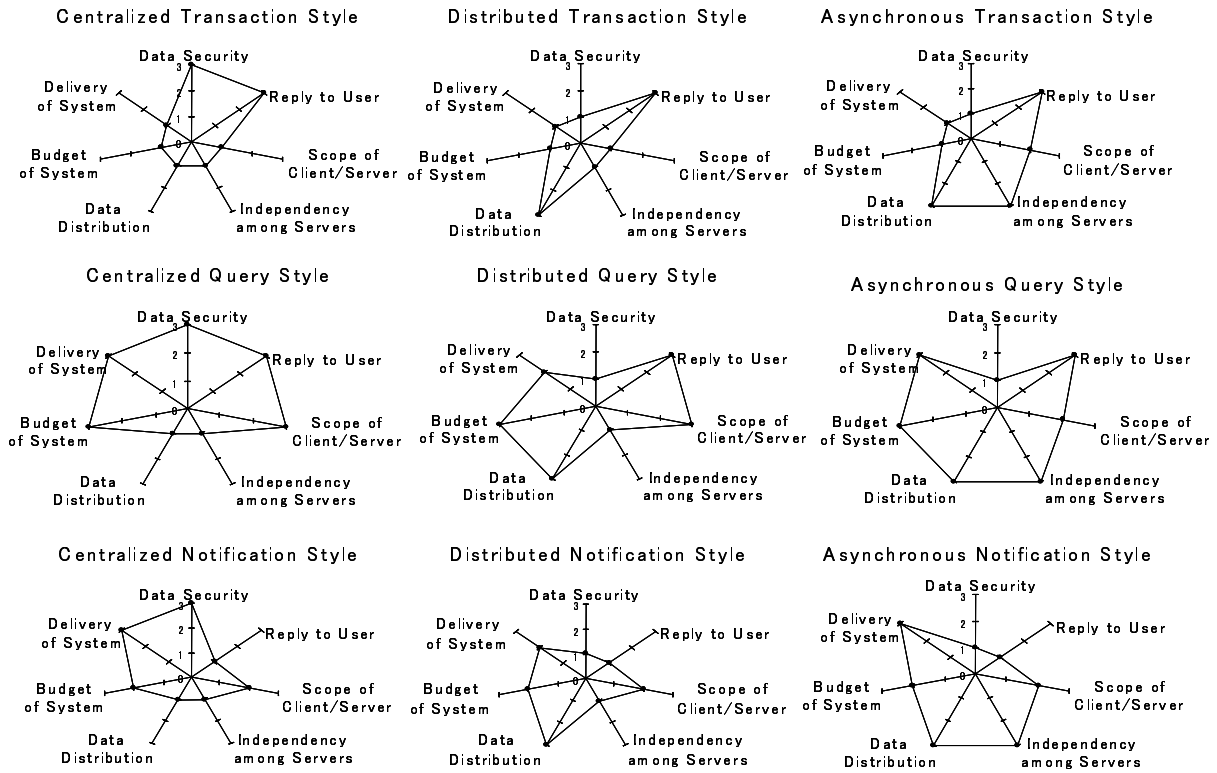


Figure 3. Characteristic charts of architectural styles

Characteristic values for each architectural style after applying the following rules learned by our experiences are: a development period of an application system with transaction type or a distributed style that has a relatively longer period of time, and a development period with query type and notification type that has a relatively shorter period of time.

Table 3 shows the characteristic values after applying the above rules to each architectural style described in the previous section. The radar charts to visualize characteristics of each architectural style are shown in Figure 3.

3.2 Selection method of architectural style

When selecting an architectural style, system architects consider requirements of users, developers, and managers from bird's-eye viewpoints. Usually, all requirements are not satisfied. The selection of an architectural style is important in deciding an

implementation method, a processing structure and style of an application system, and for selecting software products to implement and to administer the system.

In Section 3.2, we explain a method to select which architectural style conforms to an application. We use resemblance between characteristic charts represented by the characteristic vectors and the characteristic values of each architectural style introduced in the previous section. For supporting to find a resemblance chart, we introduce measurement criteria of "Distance" and "Size" and then we propose a selection method.

3.2.1 Measurement criteria

We define measurement criteria of Distance and Size as follows:

1) Criterion of Distance

A criterion of distance is a measure of the difference between the characteristic values of each of the seven characteristic vectors of an application

system (X) and each of the nine architectural styles (S_i). The distance of an architectural style is defined as the sum of the absolute values of the difference of each characteristic value.

■ Distance of X and i^{th} architectural style =

$$\sum_{j=1}^7 |S_{ij} - X_j|$$

where S_{ij} = Value of j^{th} vector of i^{th} architectural style,

X_j = Value of j^{th} vector of X.

2) Criterion of Size

The Size of an architectural style is defined as sum of the characteristic values of the characteristic vectors.

■ Size of i^{th} architectural style = $\sum_{j=1}^7 S_{ij}$

where S_{ij} = Value of j^{th} vector of i^{th} architectural style.

Using the numerical value as a characteristic value, we assume that a large value has higher functionality, efficiency, a shorter period of time, and a lower development cost. Therefore, the style with a large size meets the general requirements of software quality and systems development, and more than one architectural style may be selected. This selection method becomes the same intuitively as examining the resemblance degree of the radar charts of the architectural styles of an application system. Generally, when the distance is near, the resemblance degree of the characteristic charts becomes higher.

3.2.2 A method to select the architectural style

The following steps are applied to the selection of an architectural style:

Step 1. Decide a characteristic value for each characteristic vector of an application system.

Step 2. Measure distance between the requirements of an application system and each architectural style using the characteristic values of the application system and the characteristic values of each architectural style defined by Table 3.

Step 3. The architectural style with the shortest distance is the architectural style that should be used to implement this system.

Step 4. If the distance between more than one architectural style and the application system is the same, an architectural style with the largest size is chosen for the system.

Step 5. If more than one architectural style has the same distance and the same size, a resemblance style is chosen for the system.

After applying this selection method, architectural styles to an application system are ordered from the most suitable architectural style using the distance, size and resemblance degree between the requirements of an application system and each architectural style. Sometimes, an application system may have multiple clients. In this case, it may divide into subsystems for each client and apply the above five steps to select an architectural style based on the requirements of each client.

4. Experience necessary to apply real application systems

To verify the selection method of architectural styles proposed in Section 3, we have tried to apply the selection method to application systems used in production. In this section to apply our architectural style, we look at four typical application systems in products.

4.1 Ticket reservation system of P corp.

This system reserves and sells tickets for events such as movies and concerts to the membership of P corporation using 800 terminals in 620 shops located in the country. The ticket terminals are connected using a dedicated communication line to regional servers. Tickets start to sell all at once at all shops in the country on the mornings of Saturday and Sunday. The membership is informed of the reservation result at the time of the sale. From each sales point, tickets in the head office servers are reserved and sold via regional servers located in the Osaka area, Hokkaido area, Nagoya area and Tokyo area. This system is used to manage and to inquire about membership information, as well as for the invoicing and accounting of tickets. The extensibility of the system and the degree of security are not specifically required. However, recovery features of reservation data are a required characteristic of the system.

The characteristic values of this system are shown in Table 4. The centralized transaction style and the distributed transaction style are architectural styles for this system due to having a minimal distance as shown in Table 5. Furthermore, we show

in Figure 4 that the radar chart characterizes this system. This chart resembles the distributed transaction style more than the centralized transaction style.

Figure 5 shows the distributed transaction style adapted for the ticket reservation system utilizing local servers for the load dispersion of the head office server.

4.2 Accounts entry system of K medical corp.

This system inputs account data from ten local offices located in the country. It regularly gathers data accumulated in offices and transfers it to the host system of the main office, and processes the data on the host system. The input in each local office has time checks of the input data. Account data collected

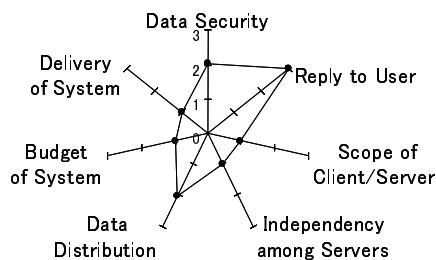


Figure 4. Characteristic chart of ticket reservation system

in each local office is accumulated into the host at appropriate times. Equipment in the local offices is connected with a LAN, and a WAN connection is used between local offices and the head office. Extensibility of the system is required due to the expansion of local offices. Security features are required to handle account data.

The characteristic values of this system are shown in Table 4. The asynchronous transaction style and asynchronous query style are architectural styles for this system due to having a minimal distance as shown in Table 5. The asynchronous query style has larger size than the asynchronous transaction style. The radar chart of this system is shown in Figure 6. This chart resembles the asynchronous query style.

The production system shown in Figure 7 includes the asynchronous query style.

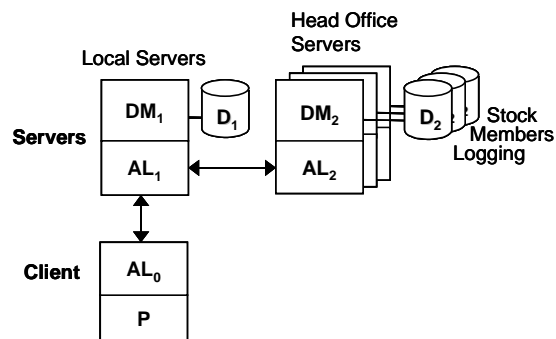


Figure 5. Ticket reservation system

Table 4

Characteristic values of examples

Characteristic Vector Evaluated System	Data Security	Reply to User	Scope of Client/Server	Independency among Servers	Data Distribution	Budget of System	Delivery of System
Ticket reservation	Either(2)	Reply(3)	WAN(1)	Dependent(1)	Either(2)	Enough(1)	Enough(1)
Accounts Entry	High(3)	Reply(3)	LAN/WAN(2)	Independent(3)	Distributed(3)	Either(2)	Either(2)
Field Engineer Support	Low(1)	Notification(1)	LAN/WAN(2)	Dependent(1)	Distributed(3)	Reasonable(3)	Short(3)
Customers' Application Support	Low(1)	Notification(1)	LAN/WAN(2)	Independent(3)	Distributed(3)	Either(2)	Short(3)

Table 5

Distances between architectural styles

Application System Architectural Style	Ticket reservation	Accounts Entry	Field Engineer Support	Customers' Application Support
Centralized Transaction Style	2	7	11	12
Distributed Transaction Style	2	7	7	8
Asynchronous Transaction Style	5	4	8	5
Centralized Query Style	8	7	7	10
Distributed Query Style	7	6	4	7
Asynchronous Query Style	9	4	4	3
Centralized Notification Style	8	7	5	6
Distributed Notification Style	7	6	2	3
Asynchronous Notification Style	10	5	3	0

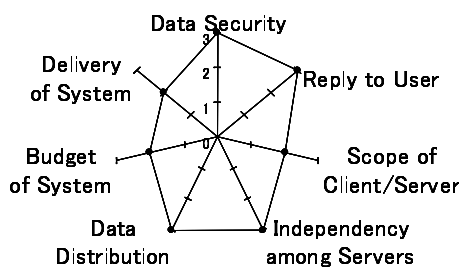


Figure 6. Characteristic chart of accounts entry system

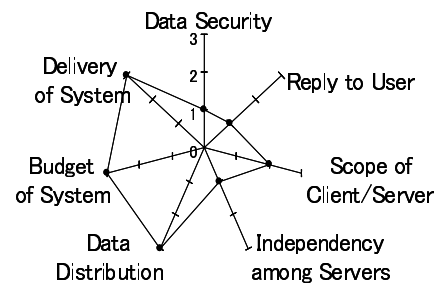


Figure 8. Characteristic chart of FE support system

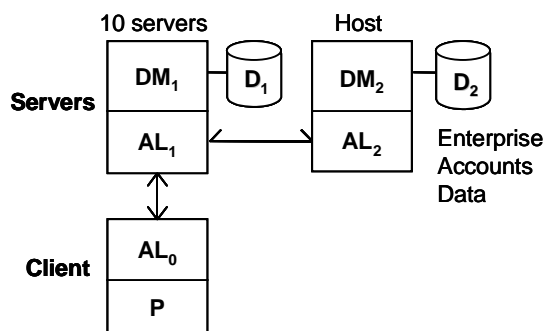


Figure 7. Accounts entry system

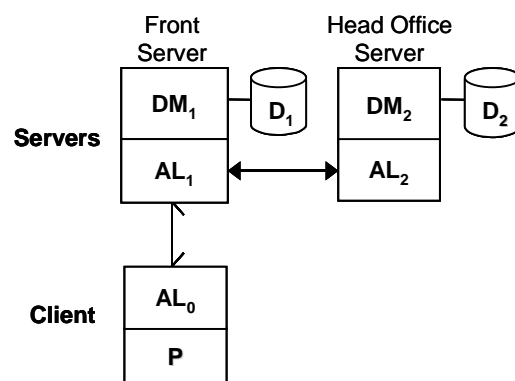


Figure 9. Field engineers support system

4.3 Field engineer support system of C corp.

This system helps business processes of a FE (Field Engineer) taking care of the repair of sold products. A front desk person, who is in charge of an information desk, receives repair request of customers, inputs the requested data on a server, and orders a visit to a place on an FE's handy terminal via the server. The FEs go straight to the spot and do repair work, on the basis of repair designation data. The FEs input a repair result into a handy terminal on the spot, and output an activity completion report. This report is transmitted to the front desk server on that day or the next day. Data transmitted to the front desk server is accumulated in the head office server in Tokyo. Processing between a handy terminal and the front desk server is not synchronized, but data between a front desk server and the head office server needs to be synchronized. The network has both LAN and WAN connections. Nothing is mentioned by the company about the extensibility of the system. Strong security it is not demanded, but strong recovery is required.

The radar chart of this system is shown in Figure

8. This chart resembles the distributed notification style.

The production system shown in Figure 9 includes the distributed notification style.

4.4 C power customers' application support system

This system supports a customers' application to request a construction. A request of the construction application from a customer is entered into an application database on the host computer. This data is downloaded automatically at a fixed time to a branch office server. In the branch office, the server uses the database on the server and automatically produces execution schedule data for that day according to the region, and produces execution information for each service engineer. This data is downloaded to each mobile terminal, and a service engineer works on the basis of the data. After the work ends, the service engineer inputs an execution result into the mobile terminal. The service engineer uploads an execution result in the mobile terminal to the branch office server after returning to the office.

The information regarding completion of the construction confirms that the work has been taken into the host computer. The customers' application support system consists of the receptionist entry function and the construction support function. For the construction support function, the necessary data is distributed. Mobile terminals are used in construction support, but responsiveness is not needed, and works between a branch office server and the head office host are done asynchronously. LAN and WAN connections coexist on the network. Extensibility of the system and facility are important, but security and cost are not demanded.

The radar chart of the construction support function of this system is shown in Figure 10. This chart resembles the asynchronous notification style.

In the customers' application support system, shown in Figure 11, the centralized transaction style applies to the function of the receptionist, and the asynchronous notification style applies to the construction support system.

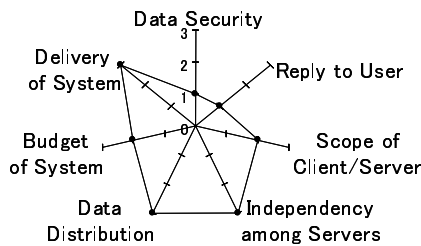


Figure 10. Characteristic chart of customers' application support system

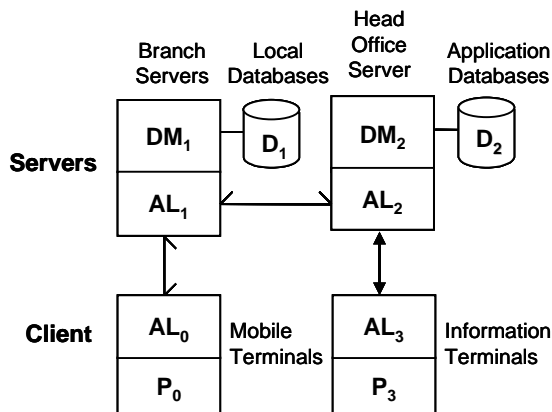


Figure 11. Customers' application support system

4.5 Considerations

In each example, deciding the characteristic values of seven characteristic vectors is relatively easy task even if a relationship among each characteristic vector exists due to their simple values such as, "Yes/Undefined/No". Using a simple inquiry system of the question and answer type, system architects may assign a value "Yes" or "No" if they understand the requirements related characteristic vectors, and will assign a value "Undefined" if they have no information or do not understand the requirements related character vectors.

Distance between architectural styles introduces a notation of priority when we have another viewpoint. We can easily select the next candidates if we use the priority order of the selection. Therefore, keeping a next possible choice in the selection is important.

In this paper, we measured the distance and size by assuming that there are no differences of importance among characteristic vectors. However, this produces a system that has to maintain its security with the lowest development cost even if it has limited features. For such strong requirements, we would measure the distance and size by adding a weight account to each characteristic vector. In this case we may have several kinds of coefficient values of weight account under the enterprise and system environment. However, in many cases such as in the examples shown in this paper, our selection method was able to be used practically without adding a weight account to characteristic vectors.

5. Related works

Alex Berson classified cooperative processing systems into five models: Distributed Presentation, Remote Presentation, Distributed Business Logic, Remote Data Management and Distributed Data Management [3]. The Gartner group has also defined five models of C/S computing which are very similar to Berson's model and is using the model in their research reports and conference presentations to aid in the discussion of C/S applications [4]. This model is based on the distribution points of presentation functions, application logic functions, and data management functions. However, this model does not account for asynchronous processing, making it difficult to model groupware applications and e-mail type applications.

IBM Corporation classified C/S systems into six templates in their guide for C/S systems. They are: Front-ending, Resource centric, Host-distributed logic, LAN-distributed, Data staging, and Multi-application. These templates are used as a reference model in C/S engineering [5]. The templates were the result of a survey of approximately 50 real-world solutions designed or implemented by IBM's typical customers in the early 1990s [18]. However, the criteria for applying these templates are not clear to users.

The architectural styles are announced based on modeling actual software in the past different from multiple layered platform and cooperative C/S processing. Shaw introduces typical seven architectural styles in her book [16]. They are: Pipes and filters, Data abstraction and object-oriented organization, Event-based and implicit invocation, Layered systems, Repositories, Interpreters, and Process control. These styles are the collection of experiences. Shaw also reports the feature-based classification of architectural styles [17]. They are: Data flow styles, Call-and-return styles, Interactive process styles, Data-centered repository styles, Data sharing styles, and Hierarchical styles. When considering these styles from the point of view to build a business application, it is difficult to use them as a reference model due to unclear principles at selecting an architectural style.

As the analysis method of software architecture, there are the SAAM [8] to make a scoring based on scenarios and the ATAM [9] to use the model of analogizing quality characteristics. The ATAM is a method for evaluating architectural-level designs that considers multiple quality attributes such as modifiability, performance, reliability, and security in gaining in sight as to whether the fully fleshed out incarnation of the architecture will meet its requirements. The method identifies trade-off points between these attributes, facilitates communication between stakeholders such as user, developer, customer, and maintainer. The ATAM is meant to be a risk mitigation method; a means of detecting areas of potential risk within the architecture of a complex software intensive system. It is not a design method of the architecture.

According to the selection of the architecture, Kishi reports the adoptability of the method to select the architecture using the method for decision-making, AHP (Analytic Hierarchy Process) [10]. This method is still in the examination phase

and difficult to use for less experienced architects.

6. Conclusions

We have focused a distributed processing system on the location of data and the processing type between a client and a server and among servers, and have classified it into nine architectural styles. This is the result to re-examine the C/SS model reported by Morisawa [11] in order to take in the mobile and agent information technology.

Using the architectural style explained in Section 2, we have proposed a method to select an appropriate architectural style for developing an application system. For this selection method, we have introduced characteristic vectors and their values to characterize an architectural style from the viewpoint of users, developers, and managers. When absolute criteria of superiority and inferiority did not exist such as among our architectural styles, we have introduced measurement criteria of distance and size and proposed a simple but practical selection method of conformity. We have expressed a radar chart for each architectural style, and showed resemblance degrees to be an effective concept. We have applied this resemblance degree to business application systems in production, and furthermore validated its effectiveness. Our less experienced architects and consultants have been using our architectural styles for selecting products sets and for writing proposals to our customers as reference architectures to design architecture of an application system. We have positive feedbacks for the selection method to support the designing of architecture of an application system in the system development field.

Our selection method will be applied to other architectural styles also. We will evaluate it in the next step.

In this paper, we do not mention how we divide application logic and how we deploy them into a distributed environment. This subject is an important point of view for a distributed processing system. We will do it as a future study.

References

- [1] Bass, L., et.al., *Software Architecture in Practice* (Addison-Wesley, 1998).
- [2] Bass, L. and Kazman, R.: *Architectural-Based Development*, (SEI, CMU/SEI-99-TR-007, 1999).
- [3] Berson, A., *Client/Server Architecture* (McGraw-Hill, 1992).

- [4] Cassell, J., The Total Cost of Client/Server: A Comprehensive Model (A Gartner Group Conference on the Future of Information Technology Industry, 1994).
- [5] IBM, A Guide to OPEN CLIENT/SERVER (Open Enterprise group of IBM Europe in Basingstoke UK, 1994).
- [6] ISO/IEC 10026-1: Information technology – Open Systems Interconnection – Distributed Transaction Processing – Part 1: OSI TP Model (1992)
- [7] ISO/IEC 9126: Information technology – Software product evaluation – Quality characteristics and guidelines for their use (1991)
- [8] Kazman, R., et.al., Scenario-Based Analysis of Software Architecture, IEEE Software, Vol.13 No.6 (1996) 47-55.
- [9] Kazman, R., et.al., The Architecture Tradeoff Analysis Method, Proceedings of 4th International Conference on Engineering of Complex Computer Systems (ICECCS98) (1998).
- [10] Kishi, T., On Software Architecture – Architecture Selection based on AHP -, Technical Report on IPSJ-SIG Software Engineering, 2001-SE-130, (2001) 101-108 in Japanese.
- [11] Morisawa, Y., Iwata, H. and Toyama, H., A Proposal of Computing Models for Distributed Processing Systems, Technical Report on IPSJ-SIG Software Engineering, 96-SE-109 (1996) 17-24 in Japanese.
- [12] Morisawa, Y., Okada, H., Iwata, H and Toyama, H., A Computing Model for Distributed Processing Systems and Its Application, Proceeding of 1998 Asia Pacific Software Engineering Conference (1998) 314-321.
- [13] Morisawa, Y. and Torii, K., Architectural Styles for Distributed Processing Systems, Technical Report on IPSJ-SIG Software Engineering, 99-SE-122 (1999) 9-16 in Japanese.
- [14] Morisawa, Y., A Computing Model of Product Lines for Distributed Processing Systems, Its Product Sets, and Its Applications, Proceedings of the First Software Product Lines Conference (SPLC1) (2000) 371-394.
This proceedings is published as; Patrick Donohoe, SOFTWARE PRODUCT LINES – Experience and Research Directions, (Kluwer Academic Publishers, 2000) 371-394.
- [15] Morisawa, Y. and Torii, K., An Architectural Style of Product Lines for Distributed Processing Systems, and Practical Selection Method, Proceedings of Joint 8th European Software Engineering Conference (ESEC) and 9th ACM SIGSOFT International Symposium on the Foundations of Software Engineering (FSE-9) (2001) 11-20.
This proceedings is published as; Morisawa, Y. and Torii, K., An Architectural Style of Product Lines for Distributed Processing Systems, and Practical Selection Method, ACM Software Engineering Notes, Vol.26 No.5 (2001) 11-20.
- [16] Shaw, M. and Garlan, D., Software Architecture (Prentice Hall, 1996).
- [17] Shaw, M. and Clements, P., A Field Guide to Boxology: Preliminary Classification of Architectural Styles for Software Systems, Proceedings of 21st International Computer Software and Application Conference (COMPSAC'97) (1997) 6-13.
- [18] Shedletsky, J.J. and Rofrano, J.J., Application reference designed for distributed system, IBM Systems Journal, Vol.32, No.4 (1993) 625-646.