

Toward Realization of Service-Oriented Architecture (SOA)

● Isao Morita

(Manuscript received April 25, 2006)

These days, companies must quickly identify changes that will affect them and quickly respond to those changes. In some cases, to secure global competitive superiority, companies must make rapid decisions to start new businesses as early as possible. On the other hand, because the IT systems at the heart of businesses are developed in response to individual requests, there are multiple platforms and application architectures, which causes complexity and bloating. This makes it hard to grasp the overall situation, making it difficult to adapt a system to a new business and ensure consistency when a system is maintained. Service-Oriented Architecture (SOA), which is a concept for making systems robust against change, is attracting attention as a way to solve these problems. Fujitsu made its system of SOA technologies public in July 2005 to facilitate the realization of IT systems that are robust against change. This system provides SOA development techniques (SDAS/Service Modeling). This paper describes the technologies used to realize SOA.

1. Introduction

Service-Oriented Architecture (SOA) has recently been in the spotlight. It is regarded as the only current solution for realizing information technology (IT) systems that can keep up with changes in the business environment and is expected to provide great benefits to businesses. Some say that SOA may outperform existing system development methods because of its advanced business analysis method and IT and its revolutionary productivity. However, others say that it is only a buzzword used by vendors in sales talks and it will change nothing. In short, as it now stands, SOA is not well understood.

The February 6, 2006 issue of *Nikkei Computer*¹⁾ says that, "SOA is a culmination of existing technologies for the purpose of problem solving and there is no breakthrough in its technology." Fujitsu is adapting and promoting SOA, and we were very impressed with this quote, especially the part, "for the purpose of problem

solving." We are often asked what problems SOA can solve; however, it is not easy to answer this question because the problems it can solve vary from business to business. The above quote gives us a renewed recognition of the fact that SOA is designed to solve problems in business.

This paper describes SOA in terms of the application of SOA technologies to solve problems.

2. What is SOA?

First, we will describe the concept of SOA.

SOA was first proposed in a report of Gartner, Inc. issued in 1996 (as long as 10 years ago). The report said that SOA is a software architecture that specifies system structures, where the whole information system is regarded as a collection of services interoperated and used freely through a common interface (the service bus) independent of hardware. In addition, the strongest effect of SOA is that it makes a system robust against change (**Figure 1**).

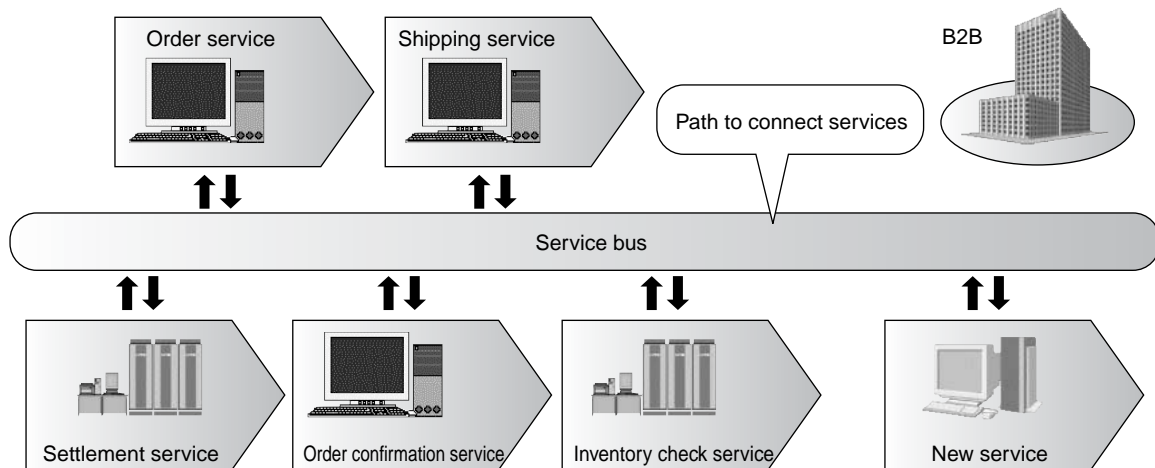


Figure 1
Image of SOA.

A “service” is an independent group of functions that is unrelated directly to anything except the service interface. There are various services that run on various platforms and OSs, are used at various locations, and are written with various languages. The service bus allows communication between these services. The service interface is a standard method of communication that anyone can use, regardless of the service vendor. The service bus hides which service the applications are communicating with. SOA makes a system robust against changes that occur in this structure.

Current IT systems are like spaghetti, and it is impossible to tell what effects a change will have in other parts of a system. Given the present circumstances, much time and expense are required to make changes in a system. In addition, system complexity has been increasing recently because individual systems have become integrated into company-wide systems using, for example, advanced networks, the Enterprise Architecture (EA) concept, and Enterprise Application Integration (EAI) tools. It now seems clear that SOA can make a system robust against changes in IT.

Let us compare the concept of SOA with the evolution of IT.

1) Simple components

There was a time when systems were thoroughly developed as components based on key concepts such as reuse, efficiency, and streamlining. The difference between a component and a service is that a component only works when embedded in the application that is using it, while a service works even if it is used by an application that is remote from the service or uses a different language.

2) EAI

EAI tools enable connections between systems with heterogeneous platforms and languages, enabling us to connect individual systems easily by converting the format and properties of the data in the heterogeneous connection process. However, this technology is based on data interoperability that is dependent on the vendor that provides the EAI tools.

3) Distributed object technology

Distributed object technology enables an object to work even if an application using the object is remote from the object or the application and object use different languages. CORBA is a well-known implementation of distributed object technology. Enterprise JavaBeans (EJB) is another implementation that has made it easy to implement distributed objects using Java

language. Because the interoperation method is standardized, objects can connect and communicate with other objects at any location. These technologies provide a basis for realizing SOA.

4) Web services

Web services are becoming popular because they make software functions available over a network. Web service technology is also standardized. The protocol SOAP, which is based on XML, is rapidly spreading because it is not only independent of hardware, OSs, and languages, but also easy to use compared to CORBA.

With the emergence of Web service technology, the concept of SOA, which was proposed 10 years ago, has become feasible and is attracting much attention.

5) Service bus

All major vendors now provide service bus products to realize SOA.

The service bus includes functions such as a messaging function and mediation function and acts as an intermediate interface between services.

3. Service unit

We said that the aim of SOA is to realize system architectures that are robust against

change, but is it possible to realize a robust system structure just by introducing a service bus product?

The reality is that everyone wants to know the best method for separating a required function into service units.

Service units are not clearly defined in the concept of SOA. As described in the beginning of this paper, this is because the problems that SOA can solve vary from business to business.

In cases where deregulation is in progress, as in the financial industry, it is necessary to respond to the elimination and integration of business processes and quickly make business improvements on site. In these cases, systems can be made robust against change if implemented using a service in which order controls are separated from business processes. This separation can be achieved using workflows that implement order controls and services that implement business processes (**Figure 2**).

If a system needs to handle changes or variety in the presentation layer in the future (e.g., an expansion of channels to provide screens focusing on operability for internal operators or publication on the Internet for customers), the underlying business logic should be implemented

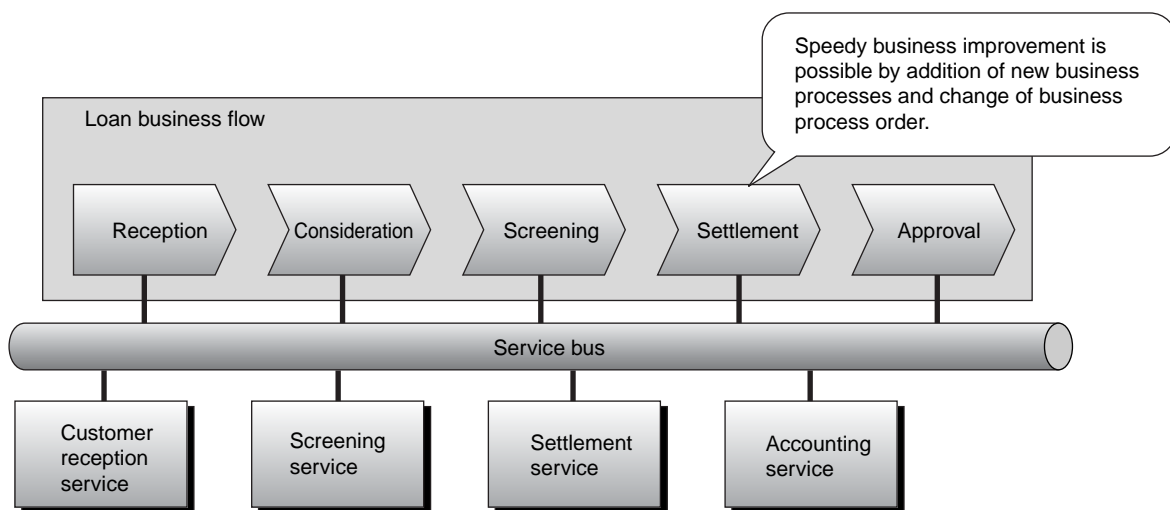


Figure 2
Image of workflow and services.

as a common service and shared by all types of channels to keep up with changes in the presentation layer (Figure 3).

Although not detailed here, the analysis results in the upper processes will greatly influence the service design and therefore these processes are important factors in deciding the service units.

4. Problems in existing system

The problems described above will be solved by clarifying requirements and designing common components as before and then defining the services. However, these approaches can lead to the following problems:

- Many changes are made in the system.
- Every requirement that is identified is temporarily made; however, their effect on the overall optimization is not considered.
- Because the effects of changes are unknown, a modified copy of the target application is added to the system and used under limited conditions.
- As a result of these problems, the system

becomes bloated and complex like spaghetti, making the effects of changes harder to grasp and requiring more time and expense to make them.

- Rebuilding is needed, but it is impossible to know what type of structures are required to build the system in order to respond to change quickly and prevent the system from becoming bloated and complex.

The developments that caused the bloating have reached their limit of effectiveness.

A number of companies are believed to have these problems in their systems.

5. Breakthrough

To prevent a system from becoming bloated and complex:

- 1) Decompose the system into an appropriate size.

In line with business, divide the system into services that can adapt to changes and have the appropriate sizes and no relation to each other.

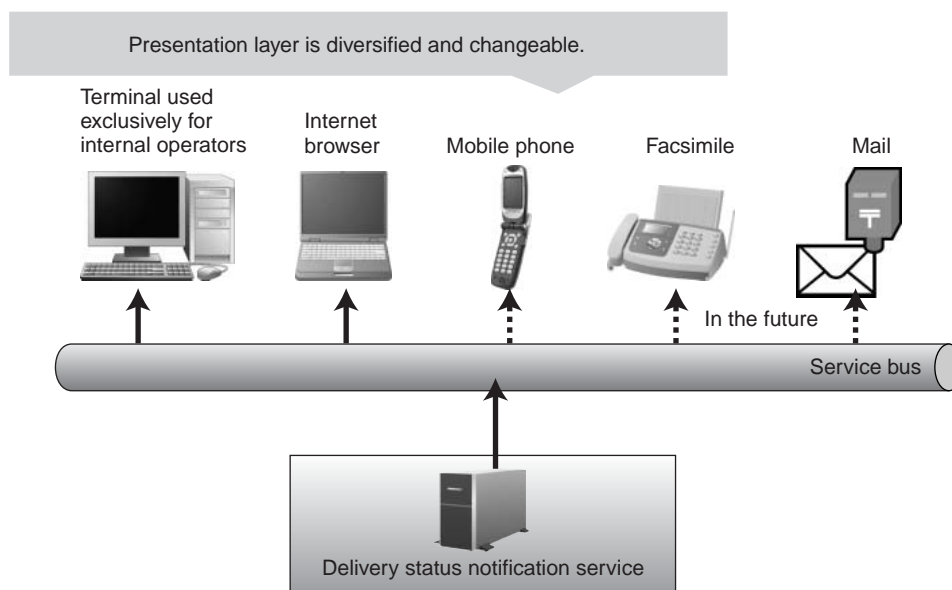


Figure 3
Responding to changes in presentation layer.

- 2) Separate the stable parts from the unstable parts.

Existing systems are composed of important processes for business and incidental processes (e.g., creation of management documents, infrequent exception handling, and functions that frequently change).

This paper proposes a modeling technology that identifies services so a bloated, complex, spaghetti-like state can be avoided.

6. Assuming causes

Systems become bloated, complex, and spaghetti-like because of important entities that exist from the start to the end of a business flow and attract many business processes.

In addition, an entity normalized by Data Oriented Approach (DOA) (**Figure 4**) attracts more processes. However, DOA is important for clarifying the true meaning and dependency of

business data and is an underlying assumption in the method of analysis introduced next.

We call these important, long-lasting entities “kaname entities.” A “kaname” is a Japanese term that means a nail or a place to secure the ribs of a Japanese fan and is compared to the most important things in a system.

Unless kaname entities are analyzed and decomposed, problems will recur. The decomposed units are set as services in SOA. Then, component-like services are considered. This is the first step in performing Kaname Analysis to realize SOA.

7. Kaname Analysis

Kaname Analysis is based on kaname entities and kaname events. Kaname entities are DOA entities and are the most important entities in a business flow. Kaname events are important business events that can change the state of an

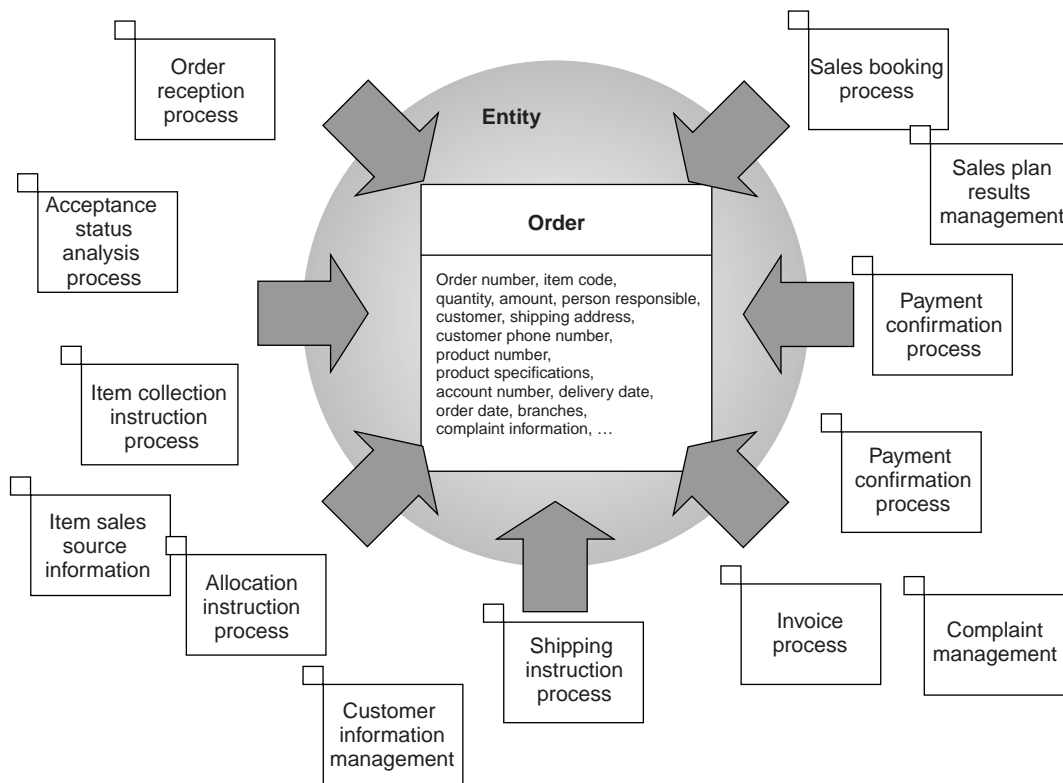


Figure 4
Kaname entity attracts processes.

entity.

These important entities and events need to be identified because we want to distinguish the essence of a business from its incidental aspects and stable things from changeable things.

Regarding the identification of the essence, Professor Zentaro Nakamura stated that, "To boil water, in the past, a pan was filled with water, put over a wood fire, and the water was brought to the boil. Today, we fill a kettle with water and put it on a stove or just use an electric kettle. In any case, the essence of this task is to change the status of the water. The entity here is the water, and the event is the change in status over time."²⁾

In, for example, the general sales and manufacturing business, an order is first received, the inventory is checked, the item is delivered to the customer, and the payment is received. This business process does not change. The DOA entities (e.g., Order, Customer, Item, Inventory, Invoice, and Payment) are stable and essential, and this is the main reason for performing DOA.

The first step in identifying the essence of a business is to construct a proper data model; that is, not a table relationship diagram based on the

normalization of data structures in screens, ledgers, and databases, but a business data model that has been clearly mapped from the essence of the business task. In short, the first step is to develop a data model centered on kaname entities (**Figure 5**).

After finding the kaname entities, the state transitions are analyzed. Here, there are two key points to keep in mind. The first is that analysis must be performed with the responsibilities of the kaname entities clarified. To take the example above, the Order entity is only responsible for the order. That is, the only purpose of Order is to ensure that the customer receives the requested item. If Order is also responsible for other tasks, the system becomes more complex, so there should be another entity, for example, Production Order, that produces the ordered item.

The other point is to find the events that change the states of the entities. This is easier to understand if it is considered that a state change is a change that people other than the changer need to know about.

The state is analyzed to analyze the business process. When writing business processes using

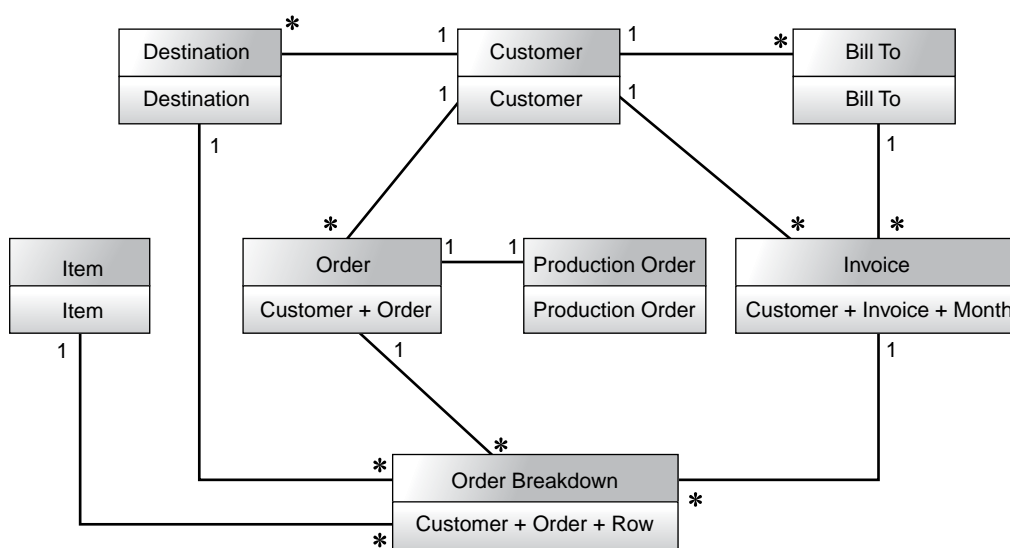


Figure 5
Data model centered on kaname entities.

a business flow or similar diagram, the following problems often occur:

- There are too many branches to draw them all.
- Infrequent exceptions need to be described.
- It is difficult to determine the granularity when writing processes.
- The flow chart exceeds 1000 pages, so nobody can or wants to review it.

When analyzing the states, we can focus on the essence of the business. For example, the acceptance of an order is an important state change from the viewpoint of the overall process required to complete the order, and this state change can be identified in the state analysis. On the other hand, the only people who need to know the individual steps required to accept an order are those who perform them; these steps are not the essence of the business and are not within the scope of state analysis (**Figure 6**).

Detailed coverage of all the points is one

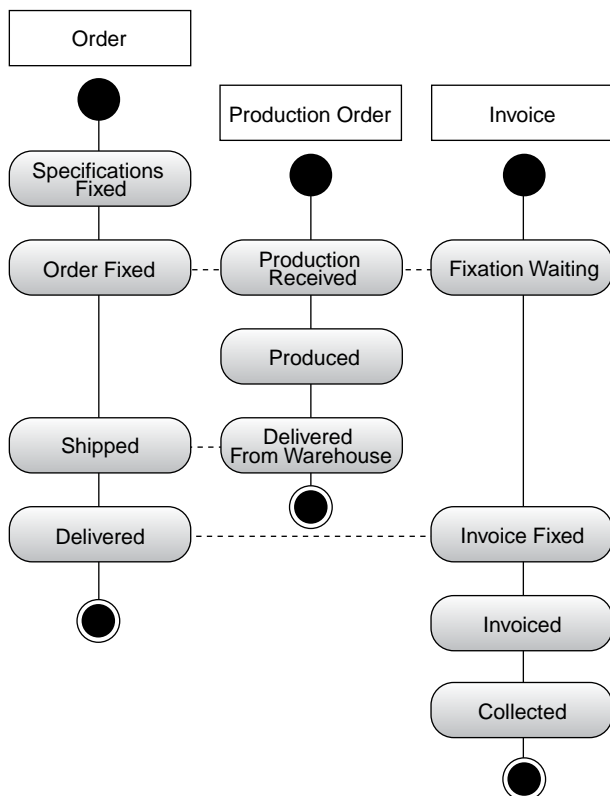


Figure 6
State transfer model of kaname entities.

issue, and the analysis of the essence is another. An effective way to identify the essence of a business and detect stable function units of business essence across organizations and businesses without falling into the traps described above is to clarify the kaname entities and their state transitions.

8. Service architecture model

So far, we have described the process up to the identification of state transitions of kaname entities. Next, we describe how to identify services using the identification results.

1) Step 1

The service architecture model (service modeling in SOA) begins by defining one service for each state of each kaname entity (**Figure 7**).

The screens, processes, and entities are deployed in service boxes. The caller and the callee are clearly distinguished in the interface between services.

Each kaname entity is deployed for each decomposed service that corresponds to a state of the kaname entity. Whether a database table is created for each service need not be considered in this analysis process because the design feasibility is reevaluated at a later stage. The major objective here is to find function units that are independent from each other and are robust against change. To do this, the system is decomposed into services, so a feasible design is considered later. In fact, there are various methods of implementation, for example, providing a virtual view. There might be cases where the processes have performance problems. The important point here is whether the process reduces the amount of interface between services and whether the system can be decomposed into independent functions.

2) Step 2

Perform a simulation of events. Simulate an order and keep track of how the process transits between services in the service architecture model. At this time, clarify the trigger of the pro-

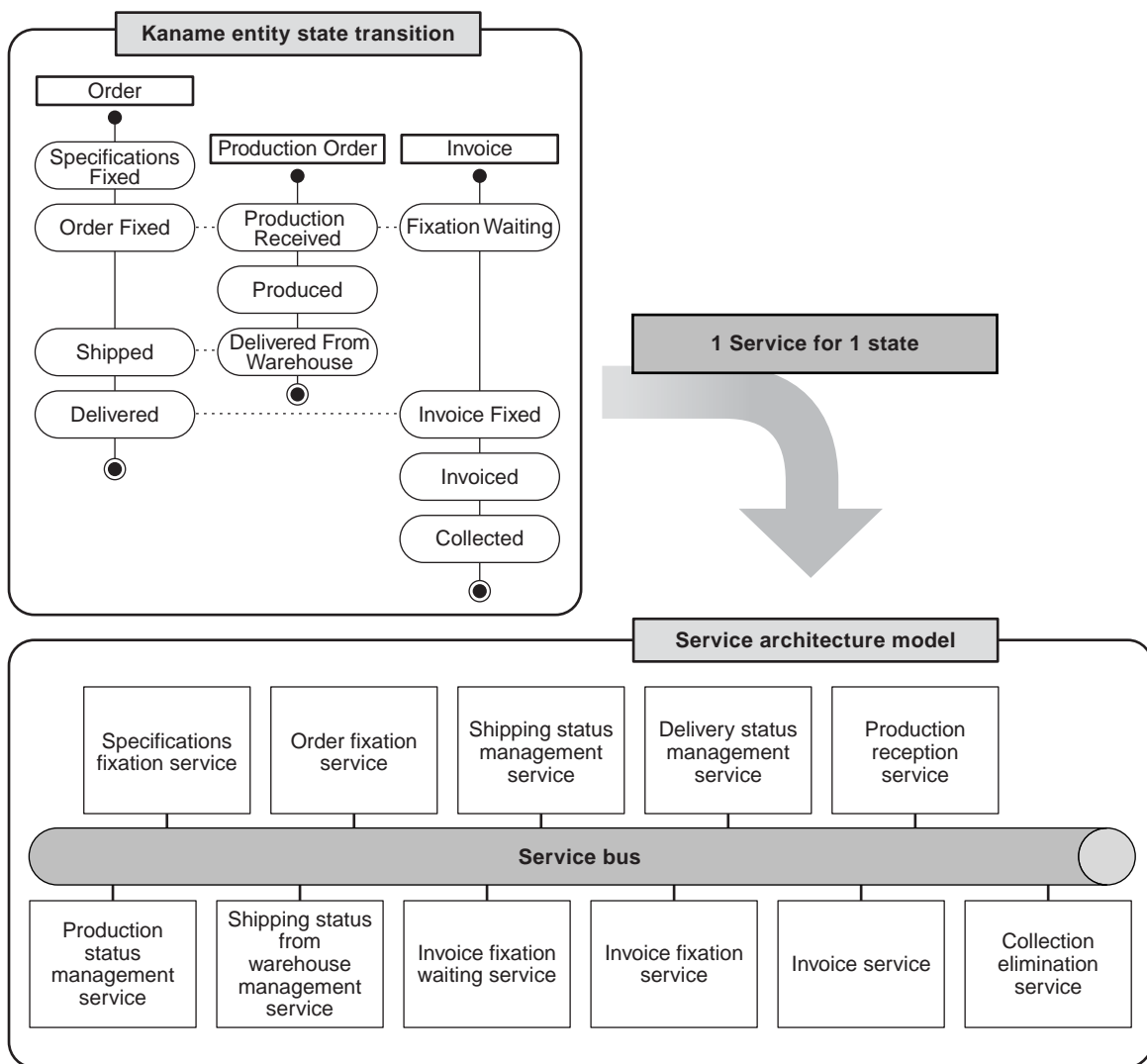


Figure 7
Service architecture model (step 1).

cess (e.g., a screen event or the time), as well as the arguments and return values of the interface. In addition, evaluate whether each entity is necessary for the service that contains it. Based on the results, improve the service architecture model (**Figure 8**).

3) Step 3

So far, the entities that make the business complex have been decomposed into services that are independent from each other. As described above, services representing the workflow control, channel response, and other functions are added according to the problems that exist. In addition,

in Step 3, the communication between other system interfaces is clarified.

4) Step 4

Verify the original problem (whether the system is robust against change) on paper. Because SOA is a new technique, it is important to study its effects and make improvements to its design.

With the steps described above, an analysis is introduced centered on the essence of the business. Naturally, it is also important to discuss and cover all the functions, screens, and data in the design. The analysis and design can be performed in the model.

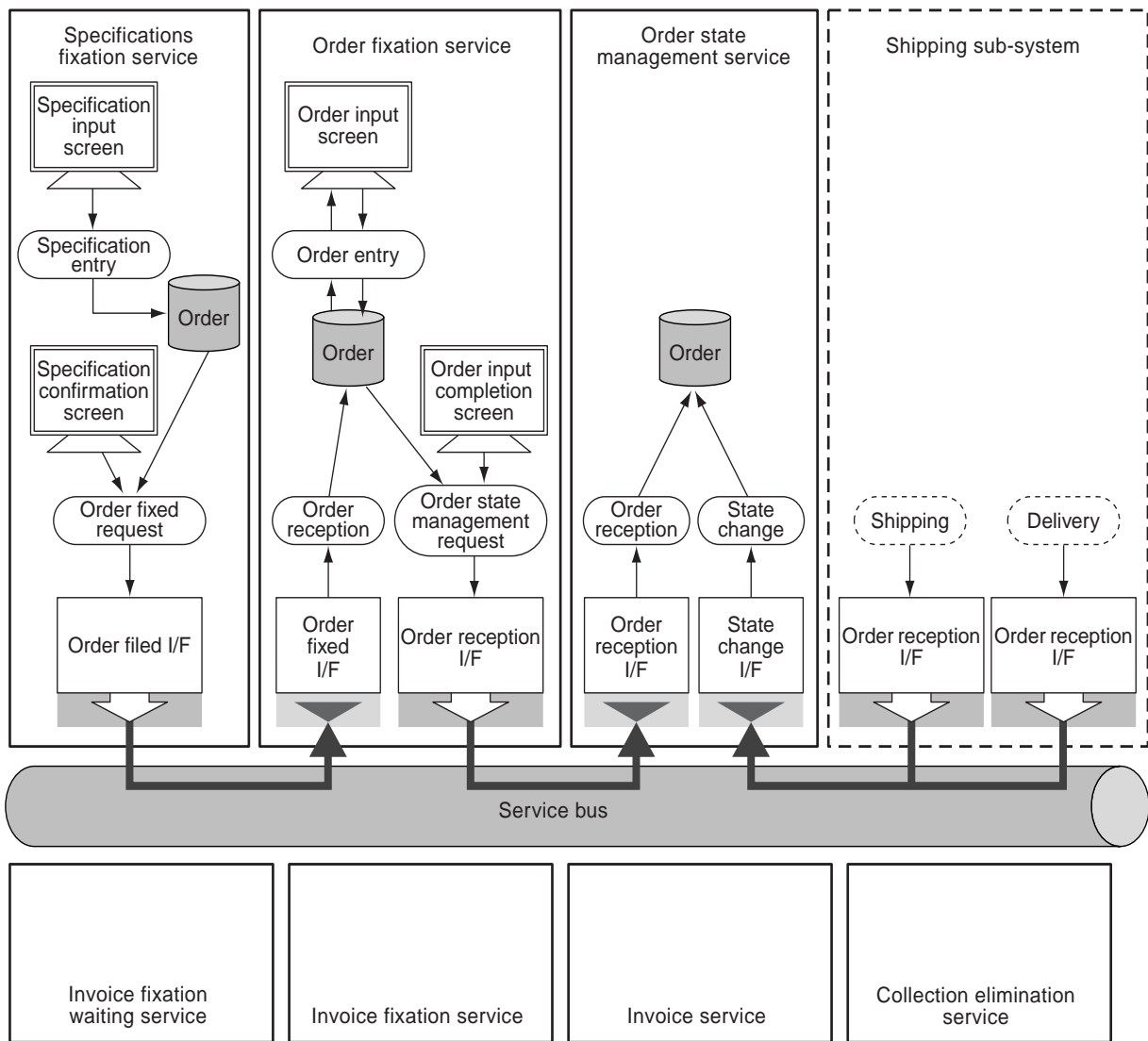


Figure 8
Service architecture model (step 2).

In addition, a feasibility design needs to be performed. There are still many points to consider in design, for example, the timing, data amount, transactions, and the physical deployment of tables. This paper has described how to design structures that are robust against change in the upper processes. Discussions about other SOA design topics will be published at a later date.

9. Conclusion

Fujitsu made its system of SOA technologies public in July 2005 to facilitate the realization of

IT systems that are robust against change. This system provides SOA development techniques (SDAS/Service Modeling) for business modeling and service configuration. This paper has mainly described service configuration.

References

- 1) Special Issue: SOA. (in Japanese), *Nikkei Computer*, February 6, 2006.
- 2) Z. Nakamura: A Planning Method of Successful Simple Work by Kaname Entity/Event Analysis. (in Japanese), The Nikkan Kogyo Shimbun, Ltd., 2003.



Isao Morita, Fujitsu Ltd.

Mr. Morita received the B.E. and M.E. degrees in System Engineering from Tokyo Denki University, Japan in 1985 and 1987, respectively. He joined Fujitsu Ltd., Tokyo, Japan in 1987, where he has been engaged in software development and support of technologies in the upper processes of Fujitsu's system development methodology, SDAS.

E-mail: morita.isao@jp.fujitsu.com