World Academy of Science, Engineering and Technology
International Journal of Economics and Management Engineering
Vol:5, No:11, 2011

A study on a Generic Development Process for the BPM+SOA Design and Implementation

Toshimi Munehira

***Abstract*—**In order to optimize annual IT spending and to reduce the complexity of an entire system architecture, SOA trials have been started. It is common knowledge that to design an SOA system we have to adopt the top-down approach, but in reality silo systems are being made, so these companies cannot reuse newly designed services, and cannot enjoy SOA's economic benefits. To prevent this situation, we designed a generic SOA development process referred to as the architecture of "mass customization."

To define the generic detail development processes, we did a case study on an imaginary company. Through the case study, we could define the practical development processes and found this could vastly reduce updating development costs.

***Keywords*—**SOA, BPM, Generic Model, Mass Customization

I. Introduction

RECENTLY we have found some SOA examples in the media or at seminars, but most of them adopt a workflow approach within a specific business process (for example, at the IT Japan Awards 2008).

It is common knowledge that to design SOA systems we have to adopt a top-down approach that begins with business modeling (Yoshida, Tanaka, and Une 2007[1], Dugan 2007[2], Yanagisawa and Mutoh 2008[3]). These cited examples accommodate this principle, but because of the limited business domain, they might be so-called "silo SOAs."

If these IT professionals continue building silo SOA systems, they will be sure to confront the problem that when interconnecting some SOA systems they find it very difficult because of different service definitions and master data, and as a result they cannot enjoy SOA's economic benefits.

Outside Japan, it seems common to use BPM suites to implement SOA systems, but we are afraid we would be easily locked in by the BPM vendor.

Originally, SOA allows us flexibility, but the SOA implementing tool, that is BPM suites, may decrease that flexibility.

To prevent such problems, we tried to develop a generic development process. In this report, we introduce the outline of this method and the results of validation of our so-called BPM+SOA designing methodology.

II. Requirements for the SOA Development Process

A. Problems of the Present System Development Model

During the orthodox system development process, engineers are changed at each development stage. Additionally, a long time is required to move from a requirement definition step to a system release step. As a result, context gaps and time gaps occur between a real business and an implemented system (Latronico and Battista 2007[4]) as shown in Fig. 1 (Munehira and Shimada 2008[5]). If gaps are large, maintenance costs rise considerably.

Besides this, it is a prerequisite condition in the present system development model that business requirements not be changed, whereas in the real world a business may be under quite severe competitive conditions, so that the environment in which it operates is always changing, and it is forced to rebuild its business model. However, the present system model is defective when it comes to adapting to these changes quickly.

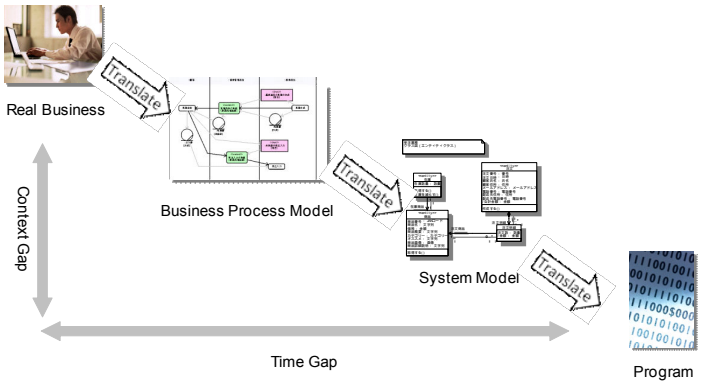


Fig. 1 Gaps between a real business and an implemented system

In other words, a BPM+SOA development method would overcome these gaps.

B. Purposes of SOA

SOA was developed in response to users' need to adapt agilely to a change of business circumstances and to reduce system maintenance costs that might become a fixed cost, as well as to fulfill software engineers' desire to create a flexible and stable system architecture (Nomura and Hara 2006[6]).

From the perspective of system development, we can find these common keywords for SOA.

1) Combine instead of coding
2) Immediately change systems according to a business' change
3) Users do system modifications

C. Differences between the SOA-oriented system and the legacy system

Table I shows the differences between the traditional system development approach and the SOA-oriented system development approach (Schmelzer 2007[7]). As described before, the orthodox systems are basically designed to last, but the SOA-oriented systems (SOA) are designed to change. This is the fundamental difference. Other items shown in Table 1 arise from this point.

TABLE I
THE DIFFERENCES BETWEEN SOA-ORIENTED SYSTEM AND LEGACY SYSTEM

Traditional Distributed Approach	Service Oriented Approach
Designed to last	Designed to change
Tightly Coupled	Loosely Coupled, Agile and Adaptive
Integrate Silos	Compose Services
Code Oriented	Metadata Oriented
Long development cycle	Interactive and iterative development
Middleware makes it work	Architecture makes it work
Favor Homogeneous Technology	Leverage Heterogeneous Technology

Source: Schmelzer 2007[7],pp12

D. The Economics of SOA

Fig. 2 shows the expected cost model of SOA (Schmelzer 2007[7], Munehira and Shimada 2008[5]). One of the important reasons to choose SOA is to reduce maintenance costs. To realize this cost model is the most important requirement for an SOA development model.

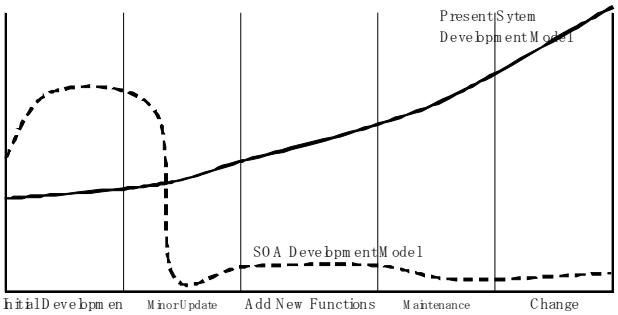


Fig. 2 The SOA economics

III. MASS CUSTOMIZATION AND SOA

Just as the waterfall development methods originated in the construction industry, we can find the similar solution of BPM+SOA in the manufacturing industry. That solution is based on "Mass Customization."

A. The definition of Mass Customization

Mass Customization is defined as follows (Pine, Victor, and Boynton 1993[8]).

"Mass customization calls for flexibility and quick responsiveness. In an ever-changing environment, people, processes, units, and technology reconfigure to give customers exactly what they want. Managers coordinate independent, capable individuals, and an efficient linkage system is crucial. Result: low-cost, high-quality, customized goods and services."

This definition has many similarities with those in Table 1.

In "mass customization," there are these four customization approaches (Gilmore and Pine 1997[9])

Collaborative: Collaborative customizers conduct a dialogue with individual customers to help them articulate their needs, to identify the precise offering that fulfills those needs, and to make customized products for them.

Adaptive: Adaptive customizers offer one standard, but customizable, product that is designed so that users can alter it themselves.

Cosmetic: Cosmetic customizers present a standard product differently to different customers.

Transparent: Transparent customizers provide individual customers with unique goods or services without letting them know explicitly that those products and services have been customized for them.

B. Enablers of Mass Customization

Mass Customization serves to achieve low costs, high quality, and highly varied, often individually customized products.

To achieve successful mass customization, the company has to turn its processes into modules and to create an architecture for linking them that will permit them to integrate rapidly in the best combination or sequence required to tailor products or services. This means that the company has to build a linkage system with these four key attributes. (Pine, Victor, and Boynton 1993[8])

1. Instantaneous.
   Processes must be able to be linked together as quickly as possible.
2. Costless.
   Beyond the initial investment required to create it, the linkage system must add as little as possible to the cost of making the product or service.
3. Seamless.
   Since a dynamic network is essentially constructing a new, instant team to deal with every customer interaction, the occasions for "showing the seams" are many indeed.
4. Frictionless.
   The instant teams must be frictionless from the moment of their creation, so information and communications technologies are mandatory for achieving this attribute.

C. Requirements for Mass Customization

Feitzinger and Lee pointed out three organizational-design principles for an effective mass-customization program (Feitzinger and Lee 1997[10]).

1) A product should be designed so it consists of independent modules that can be assembled into different forms of the product easily and inexpensively.
2) Manufacturing processes should be designed so that they consist of independent modules that can be moved or rearranged easily to support different distribution-network designs.
3) The supply network—the positioning of inventory and the location, number, and structure of manufacturing and distribution facilities—should be designed to provide two capabilities. First, it must be able to supply the basic product to the facilities performing the customization in a cost-effective manner. Second, it must have the flexibility and the responsiveness to take individual customers' orders and deliver the finished, customized goods quickly.

D.An Example of Mass Customization in SaaS

Salesforce.com typically shows the characteristics of Mass Customization (Munehira 2008[11]).

1) Customization

Salesforce.com adopts 4 customization approaches.

Collaborative

By introducing applied samples, it can clarify prospective customers' subconscious requirements. Since prospective customers can experience actual working systems, Salesforce.com can very effectively define their requirements.

Adaptive

After taking in a brief lecture, customers can make their own customizations themselves. If they need complicated customizations, such as an interconnection with their own systems, then an integrated development platform will be provided, which allows them to make their own programs.

Cosmetic

Users feel like they are using specially customized software, but Salesforce.com supplies only the usual service.

Transparent

As Salesforce.com adapts to various customers' needs, it continues updating systems without stopping the services. One day customers suddenly find that they can use new services without any configuration change.

2) Architecture

Using software, Salesforce.com has built a required architecture, that is, it can "turn its processes into modules and create an architecture for linking them that will permit them to integrate rapidly in the best combination or sequence required to tailor products or services."

This software achieved the requirements of a linkage system as follows.

Instantaneous: Customization finishes within 2 or 3 days. It used to take 3 to 6 months.

Costless: No SE supports are required, so implementation costs decrease dramatically.

Seamless and Frictionless: All processes are implemented in the software, so there is no miscommunication or friction.

E. Requirements from Mass Customization

As the SaaS example shows, mass customization is exactly what we want to realize in SOA.

From the software engineering view, we translated Feitzinger and Lee's three principles into these two modularization requirements.

Modularized business processes and services

Design service systems as compounds of independent modules, and by combining these components, make it possible to provide each service with lower costs and fewer efforts.

Modularized service design and production process

In the design process, let business or system designers design the appropriate combinations of those individual modules, so that the business systems thus designed by business users are implemented as system services without any further business level modifications.

IV. The Outline of the Generic Process

Fig. 3 shows a BPM+SOA development process model we designed to fulfill the requirements described in chapters II and III.

The 1st step is business modeling. According to the company's strategy (BSC: Balanced Score Card), we design To-Be business processes that would achieve business goals.

In this business process modeling, we use the process model references to adjust the grain level of services (Munehira 2009[12]).. Since this 1st step is required, we added BPM to SOA and decided to describe as "BPM+SOA".

The 2nd step is service mapping. Activities in the system swim lanes are detailed into the BCE (Boundary, Control, and Entity) models. Controls are mapped to services stored in the service repository.

The 3rd step is service development. If proper services are not found in the service repository or mapped services lack some functionality, we will produce new services or modify the existing services. In this step, service modification is allowed but replication is strongly forbidden. Replication will seriously decrease the reusability of services.

From step 1 through to step 3, we do data designing using an orthodox method.

The 4th step is system implementation. Screens are designed simultaneously but independently. Designed business process data and related screens are set in a way that a process engine can understand. ESB-related configurations are also set.

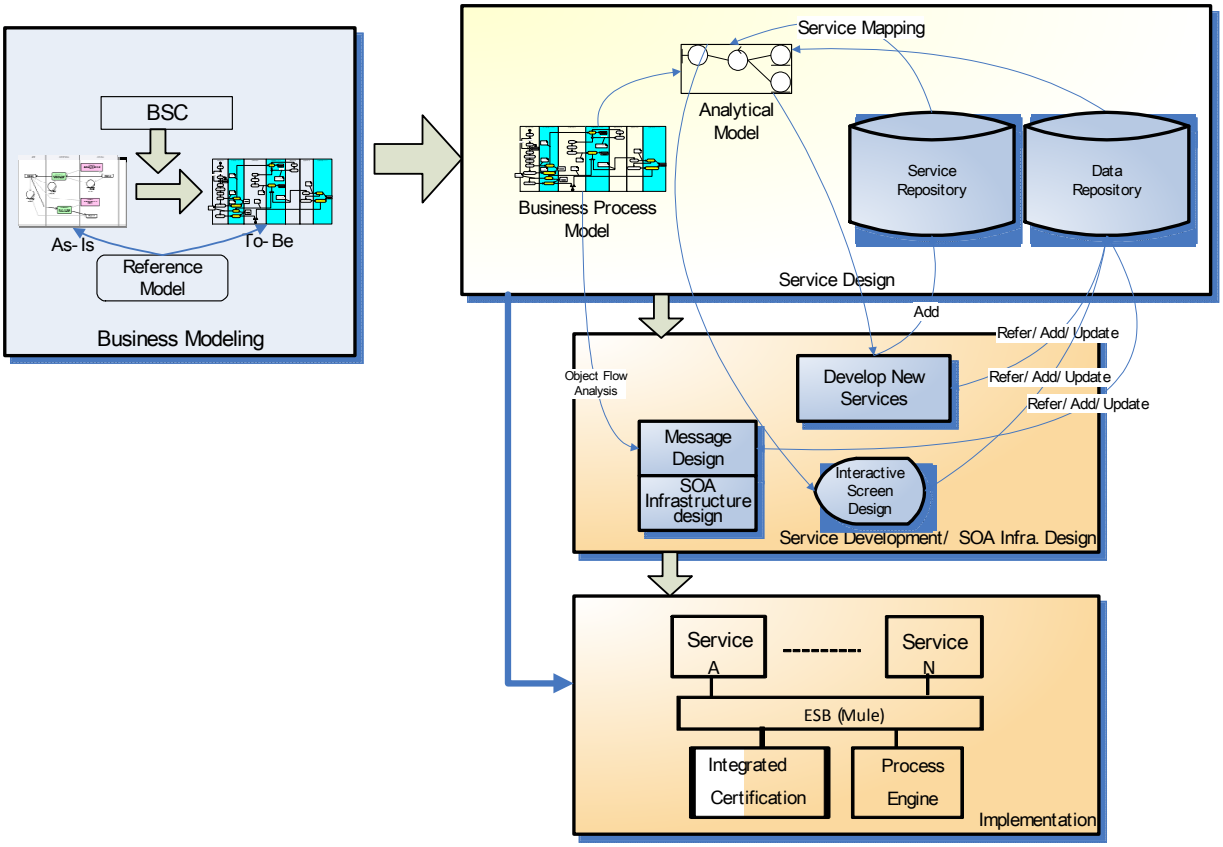


Fig. 3 The BPM+SOA development model

V. Case Study

In order to define the detail development processes and to evaluate the SOA economics, we did a case study on an imaginary company.

A. About an imaginary company

This imaginary company deals in sports bicycles and bicycle parts. This company has a wholesale department, nation-wide franchise shops, and a Web site, but it does not have any factory. This company imports bicycle parts from all over the world and sells them to bicycle fans through three channels, that is, specialty shops, franchise shops, and the Web site.

As shown in Fig. 3, To-Be business processes should align with the company strategy, so first we had to define this company's new strategy. We described the new strategy according to the instructions written in "Strategy Maps" (Kaplan and Norton 2004[13]).

BSC (Balanced Score Card) including strategy maps has four perspectives. The financial and customer perspective are business goals. The internal process perspective has a strong relation to the business processes. The learning and growth perspective describes intangible assets.

In this book, business processes are divided into four process categories—Operations Management, Customer Management, Innovation, Regulatory and Social—and we have to describe strategy maps for each business process category.

One of the management problems stems from the frequent shortage of some cycle parts. Table II shows objectives for the operation management processes.

TABLE II
OBJECTIVES FOR OPERATION MANAGEMENT PROCESSES

Process Category : Operation Management Process

Perspective	Financial		
Objectives	Increase income from A rank customers, Acquire income from local prospects, Decrease shop operating costs		
Perspective	Customer		
Objectives	Decrease customer's costs and time loss, Prepare impressive items, Provide agile and timely purchase experience		
Perspective	Internal Process		
Objectives	Develop and Sustain Supplier Relationships	Achieve supplier partnership	Achieve agile updating of supplier and cycle parts data
	Cycle Parts Procurement	Obtain agile and low cost contract with new suppliers	Have shorter lead time from sales order to purchase order
	Shop Operation	Make operation changes at low costs to suit customers' characteristics	Raise efficiency of shop operations

B. Business Process Modeling

Fig. 4 is the To-Be process of "procure cycle parts." This process corresponds to the objective "Shorten the lead time from sales order to purchase order." The upper activity diagram is the output of the 1st step, and the lower subactivity diagram is the output of the 2nd step. Screen, Mainprocess, and DB correspond to BCE (Boundary, Control, Entity).

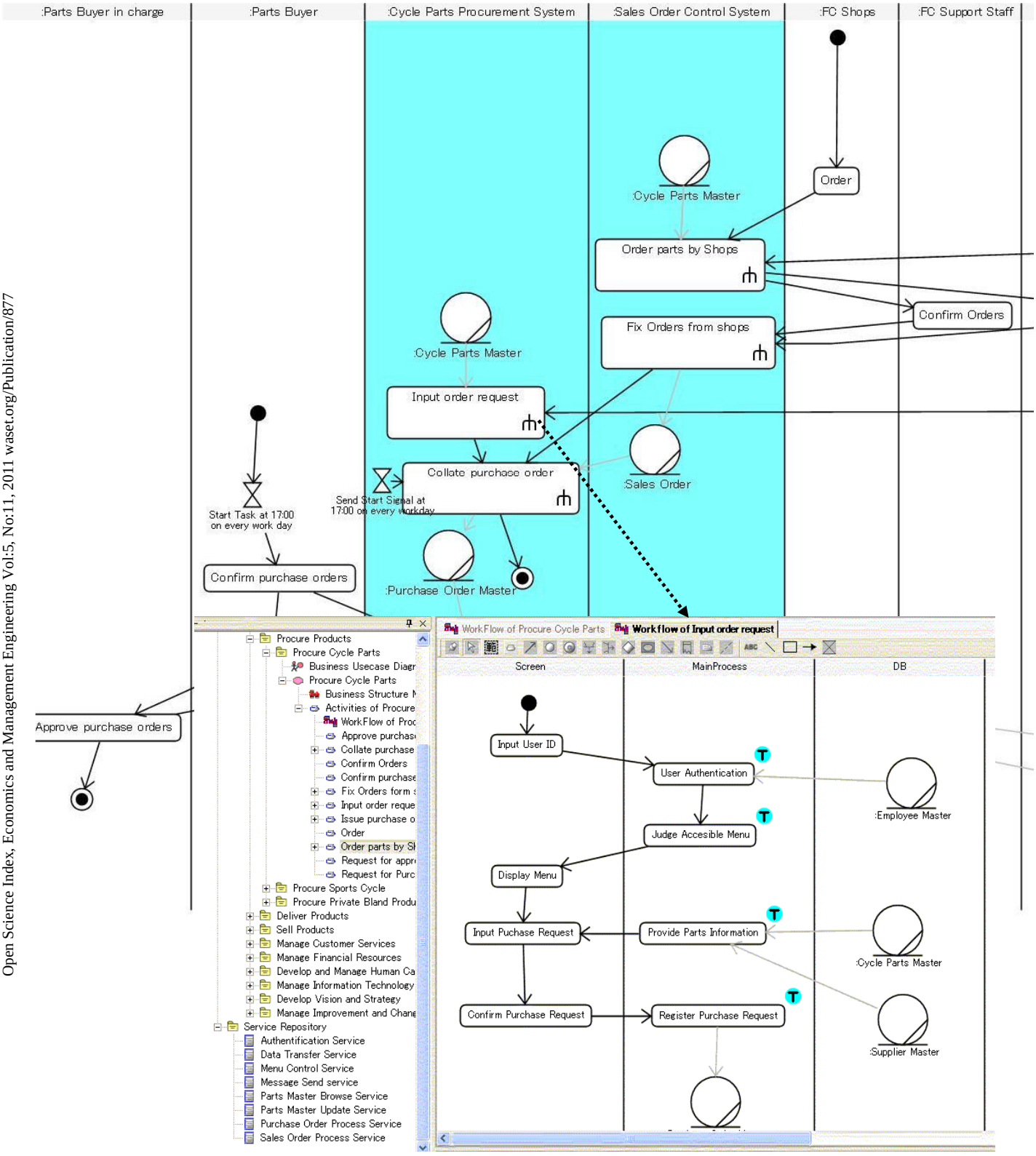


Fig. 4 To-Be business Process Model

C. OSS-based Development Environment

The purpose of this study is to design a generic development process. For BPM+SOA, we have to design and implement both human processes and system processes. When we considered using COTS to build this environment, we found the following problems.

- It is a de facto standard to adopt BPEL for implementing a system process.
- Because BPEL engines are usually supplied from software vendors coupled with ESB, these engines usually have vendor original specifications. This would force us to implement special functions that work only under specific circumstances.
- BPEL for People is prepared for implementing human processes under BPEL conditions, but few engines are supplied, and it requires high-level technical skills. Therefore, it does not fulfill the requirements written in II-B.
- BPM suites support both processes. However, if we started to use a BPM suite, we would be locked in afterward by this tool.
- Many BPM suites require us to develop screens with this tool.
- The development process is also dependent on this tool.

In order to avoid the vendor-lock-in, we prepared OSS-based SOA circumstances for the implementation of this imaginary company's model as shown in Fig. 5.

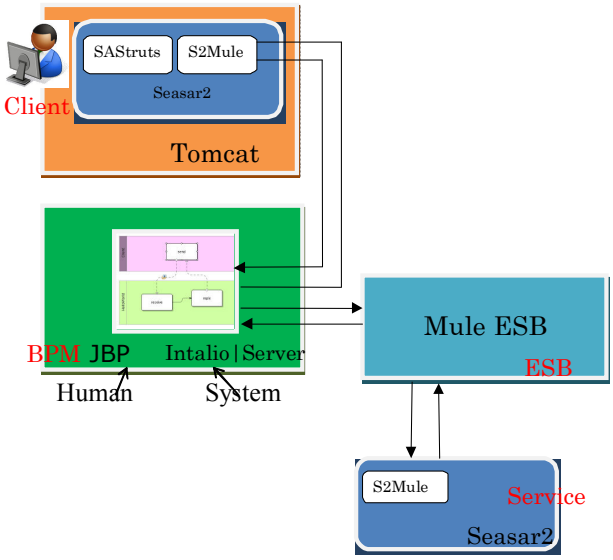


Fig. 5 OSS-based Environment

We adopted Mule for ESB, the core of SOA. It was very hard to find an OSS BPM engine, so we had to choose the BPEL engine that Intalio provides to Apache ODE. For a human workflow engine, we had to use jBPM because we could not find any OSS XPDL Engine.

VI. A Generic Development Process

A. The detail definition of the BPM+SOA design process

For BPM+SOA development, these three players are required. We organized an international team.

- Business Modeler: Osaka
- SOA Designer: Tokyo
- SOA Implementer: Shanghai

The end of a business modeler's task is to register desirable services in a service repository. In Fig. 4, the 'T' mark shows this activity is mapped to some registered service. If a business modeler cannot find a proper service, then he or she creates a new service class in the "Service Repository package" and makes a mapping. After a business modeler finishes the business process modeling, a SOA designer's task begins.

The SOA designer's tasks are defined as follows (Kranfzig, Banke, and Slama 2004[14], Oba et al. 2005[15]).

1.Service Analysis
Service candidate analysis
Conceptual model design
Service candidate refining
Service analysis
Message analysis
Service protocol analysis

2.Service design
Service design
Message design
Service definition documentation

Through a development process, we can define the generic and practical detail processes as shown in Fig. 6 and Tabel3.

We also found that services are classified into these three categories.

1) Process Services
2) Business Services
3) Fundamental Services

Process services are almost equal to BPEL or workflows, and business services are called from process services. Fundamental services are security, ID management, access control, and so on. Business services and fundamental services should be loosely coupled, independent, and reusable because they are called from various processes. However, process services are not reusable. This policy is very important to guarantee the flexibility and agility.

This new knowledge is one of the important points of "Service Analysis."

We knew independence and reusability are CSFs for the SOA repository, but we did not know how to assure them.

Through this case study, we found this generic process would allow us to design proper services.

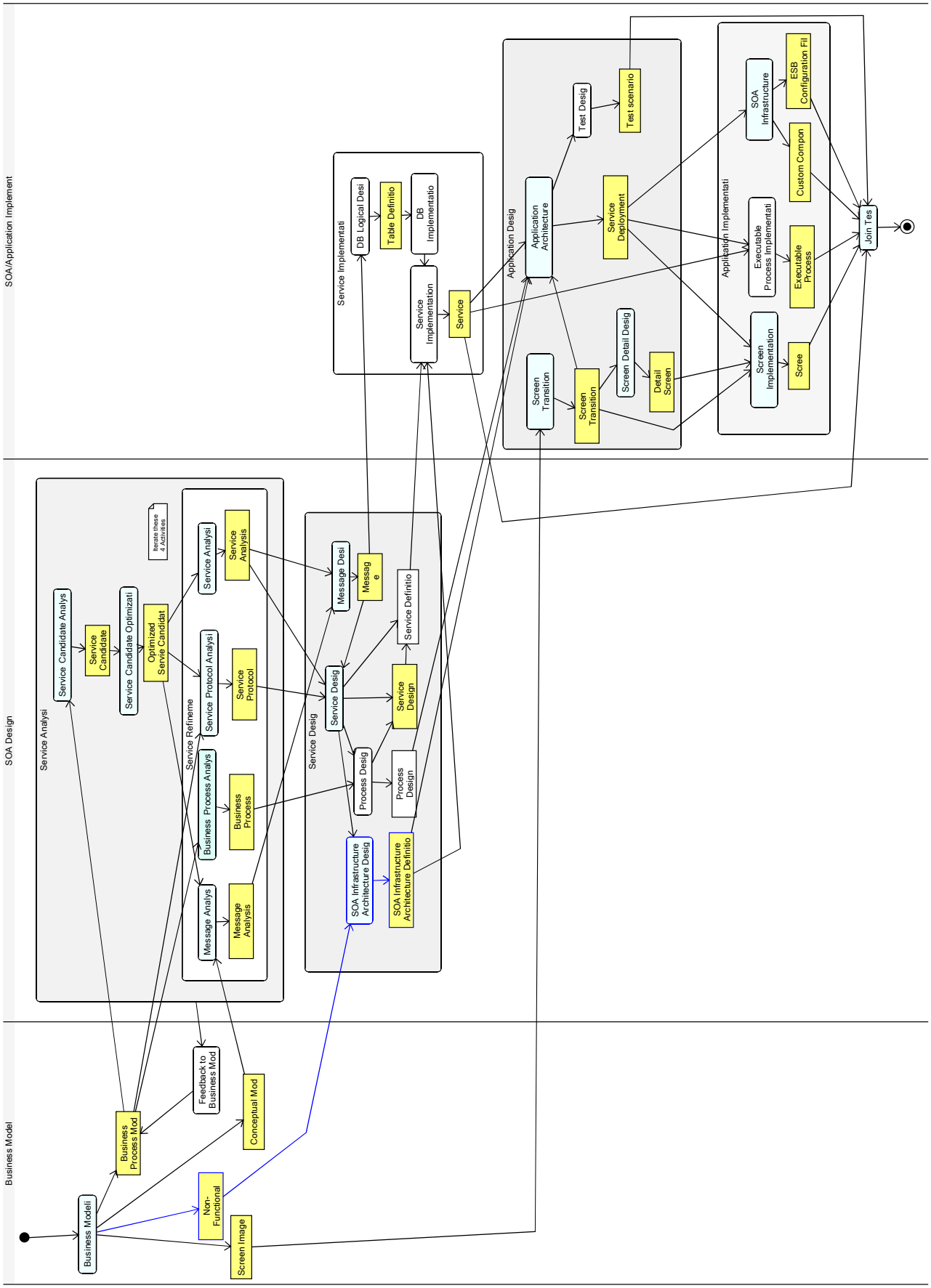


Fig. 6 The Generic Development Processes

TABLE III
THE GENERIC DEVELOPMENT PROCESSES

Activity	Subactivity	Details	Role	Input	Output
Service Analysis	Service Candidate Analysis	Extract service candidates from these elements -Services from the service repository -Activities from activity diagrams -Business Entities from activity diagrams	Service Analyst	Business Process Model	Service Candidates (Class Diagram)
	Service Candidate Optimization	Redefine service candidates based on relations, that is CRUD, with business entities.	Service Analyst	Service Candidates (Class Diagram)	Optimized Service Candidates (Class Diagram)
	Service Analysis	Define services from optimized service candidates reflecting results of Message Analysis and Business Process Analysis and Service Protocol Analysis. Services are classified into 3 categories. -Process Service -Business Services -Fundamental Services	Service Analyst	Optimized Service Candidates Message analysis model Business Process Analysis Model Service Protocol Analysis Model	Service analysis model (Class Diagram)
	Message Analysis	Create a message analysis model from the conceptual model by reflecting service analysis results.	Service Analyst	Conceptual Model Service Analysis Model Business Process Analysis Model Service Protocol Analysis Model	Message analysis model (Class Diagram)
	Business Process Analysis	Create a business process model to show how each business process is processed by service methods. This activity is to evaluate the validity of service analysis	Service Analyst	Business Process Model Service Analysis Model Service Protocol Analysis Model	Business Process Analysis Model (Activity Diagram) Update requests to the business process model
	Service Protocol	Create a sequence diagram from the	Service Analyst	Conceptual Model	Service Protocol

Activity	Subactivity	Details	Role	Input	Output
	Analysis	conceptual model and the service analysis model and the message analysis model. This diagram shows messages and interfaces between services.		Service Analysis Model Message Analysis Model	Analysis Model (Sequence Diagram)
	Feedback to Business Model	Feedback the results of service analysis to the business process model via the service repository. This activity is to keep consistency between the business process modeling and the service modeling. This consistency is very important for an iterative BPM.	Service Analyst Business Modeler	Service Analysis Model Business Process Model	Business Process Model (Services are updated) (Activity Diagram)
Service Design	Message Design	Define message profiles and types by checking whether enough items are defined for service implementation.	Service Designer	Service Analysis Model Message Analysis Model	Message model (Class Diagram)
	Service Design	Design services from the service analytic model and the message model and describe results into a service definition document.	Service Designer	Service Analysis Model Service Protocol Analysis Model Message Analysis Model	Service Design Model Service Definitions (***See Appendix***)
	Process Design	Rewrite the business process model (activity diagrams) into BPMN or XPDL or other languages that an adopted BPM engine can read.	Service Designer	Business Process Analysis Model	Process Design Model Service Design Model
	SOA Infrastructure Architecture Design	Describe the entire architecture of SOA components and structure including	Service Designer	Non-Functional Requirements Service Design Model	SOA Infrastructure Architecture Definition

Activity	Subactivity	Details	Role	Input	Output
		implementation technology for achieving non-functional requirements			
Service Implementation	DB Logical Design	Describe logical DB table definitions from the message model	Service Implementer	Message Model	Table Definition
	DB Implementation	Create DB according to DB tables	Service Implementer	Table Definition	DB Schema
	Service Implementation & Unit Test	Implement services according to service definitions. Also execute unit tests.	Service Implementer	Service Definition SOA Infrastructure Architecture Definition	Service Codes ESB Configurations Unit Test Codes
Application Design	Screen Transition Design	Define screen transition diagrams according to the process design model,	Application Designer	Screen Image Business Process Analysis Model	Screen Transition Diagram
	Screen Detail Design	With considering usability, define details of screens such as items, layouts, operation orders.	Application Designer	Screen Image	Detail Screen Diagram
	Application Architecture Design	Determine which services to use for realizing applications. Also determine interfaces between screens, processes, services and databases and then determine URI of them.	Application Designer	Process Design Diagram SOA Infrastructure Architecture Definition Service Definition Screen Transition Diagram	Service Deployment Diagram
	Test Design	Define the goals and processes of unit tests, function tests and join tests	Application Designer	Service Definition Table Definition Screen Transition Definition Detail Screen Diagram	Test scenarios
	Screen Implementation & Unit Test	Implement screens according to detail screen diagrams	Application Implementer	Screen Transition Diagram Detail Screen Diagram	Service Source Codes ESB Configurations Unit Test

Activity	Subactivity	Details	Role	Input	Output
Application Implementation				Service Deployment Diagram	Codes
	Executable Process Implementation	Describe process execute programs with using process execution language according to the process design diagrams	Application Implementer	Service Codes Service Deployment Diagram	Executable Process Process Engine Configuration Files
	SOA Infrastructure Implementation	Implement custom components and ESB configuration according to Service Deployment Diagram	Application Implementer	Service Deployment Diagram	ESB Configuration File Custom Components
Test	Join Test	Execute tests according to test scenarios	Application Implementer	Test scenarios Implementation outputs	Application

B. Validation of economics of SOA

Table4 shows the difference between the FP (Function Points) based estimation and the real data.

By means of this comparison we found the following:

- Initial SOA development cost is almost equal to that of the Orthodox method.
- In updating the system, the SOA development cost is lower than that of the orthodox method as the theory suggests (Fig. 2.)

TABLE IV
ACTUAL RESULTS AND ESTIMATION

	FP method	1st-Step	2nd-Step*
Estimation	Data-Func	36	20+36*
	Tran-Func	69	37+60
	No-Adjustment	105fpt	57+96fpt
Translate to man days		141.13	179.18

Actual Results	Analysis	10.00	20.00
	Design	30.75	
	Coding	46.55	49.10
	Test	49.40	55.60
	Total(man days)	136.70	124.70

*New+Update

VII. CONCLUSION

SOA has been developed to meet users' needs to adjust smoothly to the changes of their business environment and to reduce system maintenance costs that might become a fixed cost, as well as to fulfill the software engineers' desire to create a flexible and stable system architecture. The orthodox system development models cannot meet these needs.The purpose and characteristics of SOA are very clear, but the present situation in Japan with regard to SOA implementation methods is not good. In the USA, BPM suite users consider how to achieve the entire optimization of business processes and systems, but in Japan, companies introducing SOA are making silo SOAs. If they continue in this manner, they will not achieve the purpose of SOA.

To prevent this situation, we designed a desirable model for introducing SOA, referring to the architecture of "mass customization."

To define the generic detail development processes, we did a case study on an imaginary company. In order to maintain independence from vendor software, we prepared the OSS-based SOA environment. Through this case study, we could define the practical detail development processes of BPM+SOA and found this would vastly reduce updating development costs.

We are now creating a method of feedback from the SOA designer to the business modeler. After this, our model will be widely used, especially for offshore SOA development.

REFERENCES

[1] K. Yoshida, S. Tanaka, and Y. Une, "System development method based on EA and SOA", *Toshiba review,* Vol.62, No.9. 2007, pp72-75.

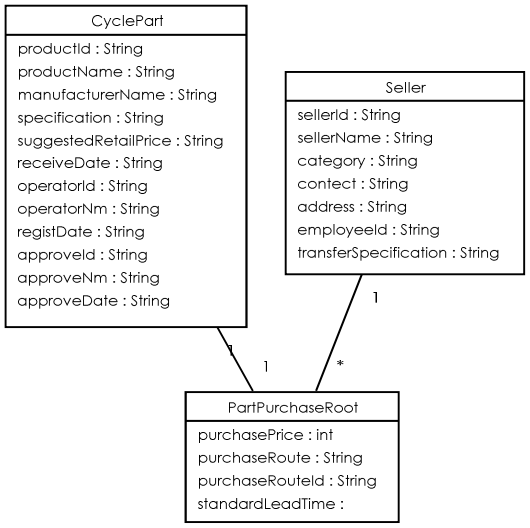
[2] L. Dugan, *Use of SOA and web Services Technologies for EA Migration—Lessons Learned on How To Sort It All Out*, *Transformation* and Innovation 2007.

[3] H. Yanagisawa and T. Muto, "Application of SOA in Software Product Development", *Unisys Technology Review*, Vol.93, August 2007, pp89-100.

[4] S. Latronico, F. Battista, *How a business process vision may boost innovative ideas*, Transformation and Innovation 2007.

[5] T. Munehira, and T. Shimada, *Analysis and Procurement of Information System*, JUSE Press, Japan, 2008.
[6] Y. Nomura, and H. Hara, "What is SOA? Now and Future," *The Journal of the Institute of Electronics, Information and Communication Engineers*, Vol.89, No.6, June 2006, pp506-510.
[7] R. Schmelzer, *Finding the Lowest Risk Path to SOA Adoption*, Transformation and Innovation 2007.
[8] B. J. Pine II, B. Victor, and A. C. Boynton, "Making Mass Customization Work", *Harvard Business Review*, September 1993.
[9] J. H. Gilmore, and B. J. Pine II, "Four Faces of Mass Customization", *Harvard Business Review*, January 1997.
[10] E. Feitzinger, and H. L. Lee, "Mass Customization at Hewlett-Packard: The Power of Postponement", *Harvard Business Review*, January 1997.
[11] T. Munehira, "A study on applying the mass-customization model on service businesses", *27th National conference of The Japan Society for Production Management*, 2008.
[12] T. Munehira, "A Study for Refining a Business Modeling Method for SOA and SaaS Designing," *APCIM2009 in Beijing*, 2009.
[13] R. S. Kaplan, and David P. Norton, *Strategy Maps: Converting Intangible Assets into Tangible Outcomes*, Harvard Business School Press, 2004.
[14] D. Kranfzig, K. Banke, and D. Slama, *Enterprise SOA: Service-Oriented Architecture Best Practice*, Prentice Hall, 2004.
[15] K. Oba, M. Hashimoto, and S. Fujikura et al., "The Status Quo and Challenges of Service-Oriented Architecture (SOA) Based Application Design," *IPSJ SIG technical report*, No.75, 2005, pp73-80.

APPENDIX
SERVICE DEFINITION

	Analytical Contents				Design Contents
Service Name	Parts Master Service				CyclePartsLedgerService
Service Consumer	Process Service				
Explanation	Parts Master Control for ordering				
Business Rule	New parts registration has these 3 types; temporary create, temporary update, update.				
Characteristics	This service is for reading updating cycle parts data				
Notes					
Methods		**Analytical Contents**			**Design Contents**
	Method Name	Get Cycle parts data			getCyclePartsData
	Business Rule	get detail data that matches required ID			
	Characteristics				
	Precondition	No			
	Post-condition	No			
	Irregular Process	Reply blank for no existing ID			
	Notes				
	Message Design Model				
	Inputs	**Analytical Name**	**Design Name:Type**	**Pre-Condition**	**Explanation**
		PartsCode	productId: string		
	Outputs	**Analytical Name**	**Design Name:Type**	**Pre-Condition**	**Explanation**
		CycleParts	part: CyclePart		